UNIVERSIDAD DE SEVILLA

ESCUELA SUPERIOR DE INGENIEROS

INGENIERÍA DE TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA:

ESTUDIO E IMPLEMENTACIÓN DE HERRAMIENTAS SOFTWARE BASADAS EN TÉCNICAS GENÉTICAS PARA EL DISEÑO DE REDES NEURONALES CELULARES

Autor: Miguel Ángel Preciado Díaz

Tutor: Eduardo Fernández Camacho

ÍNDICE

1 OBJETIVO DEL PROYECTO

Con el fin de cumplimentar lo especificado en los planes de estudio de la Titulación de Ingeniero de Telecomunicación de la Universidad de Sevilla, se presenta este Proyecto Fin de Carrera que tiene por objetivo ofrecer un paquete software integrado con la finalidad de aportar una herramienta basada en técnicas genéticas para ayuda en el diseño de algoritmos y plantillas para redes neuronales celulares, así como la obtención de parámetros indicativos de la calidad de las soluciones obtenidas y la verificación de las mismas con distintos ejemplos. Los algoritmos y plantillas nombrados anteriormente implican procesamientos de imágenes en sistemas basados en redes neuronales celulares, procesamientos que pueden ser elementales, en el caso de las plantillas, o más complejos para el caso de los algoritmos. Los procesamientos elementales se realizan mediante un procesado de un sistema basado en redes neuronales celulares configurado con una determinada plantilla que contiene los parámetros de la red. Los procesamientos complejos se realizan mediante la combinación de procesamientos elementales, y se representan en un programa o algoritmo.

Por otra parte resulta necesaria la exposición de las bases teóricas referentes a las redes neuronales celulares, las técnicas genéticas de optimización y las particularidades de la aplicación de dichas técnicas genéticas en el diseño de redes neuronales celulares. Se requiere la asimilación de estas bases teóricas por parte del usuario para la comprensión global del problema y para aplicar los programas del paquete software con propiedad, de manera que el usuario conozca el objetivo de aplicación, así como el significado y efecto de los distintos parámetros ajustables y demás elementos que intervienen en los mismos.

2 INTRODUCCIÓN

Las redes neuronales celulares, o CNNs (de *Celular Neural Network*) constituyen una nueva técnica de procesamiento de imágenes de naturaleza masivamente paralela que aporta una eficiencia en términos de tiempo de procesamiento insuperables por otras técnicas convencionales, como el uso de microprocesadores o DSPs. El principal problema a la hora de desarrollar sistemas basados en CNNs es la falta de herramientas analíticas a la hora de diseñar, siendo la prueba-error y la intuición las herramientas que se han venido utilizando principalmente.

En este contexto surge la necesidad de desarrollar algún método para la automatización de este trabajo, en muchos casos tedioso, y que permita evaluar la calidad del procesamiento en términos de velocidad y versatilidad en cuanto a capacidad de obtener resultados satisfactorios en distintas imágenes.

Este problema se ha abordado a dos niveles, a nivel de plantilla, que estará asociada a un determinado procesamiento elemental de la CNN, o a nivel de algoritmo, en el que se realiza una combinación de procesamientos elementales correspondientes a distintas plantillas para realizar un procesamiento más complejo.

La automatización del diseño se basa en técnicas genéticas de búsqueda, subconjunto de los denominados algoritmos evolutivos que constituyen una familia de sistemas basados en computador y que resuelven problemas gracias al uso de modelos computacionales de algunos de los conocidos mecanismos de evolución como clave para sus diseños e implementación.

Las dos principales técnicas de búsqueda genética son la técnicas de los Algoritmos Genéticos (G.A., Genetic Algorithm), y la técnica de la Programación Genética (GP, Genetic Programing). La principal diferencia entre ambas técnicas es la naturaleza de los elementos que van a manipular para su optimización. Los GA tratan cadenas fijas de bits, y no tienen ninguna consideración acerca de la estructura o lo que está representando estas cadenas en la manipulación de estas al aplicar los operadores de búsqueda. Los GP, sin embargo, tratan cadenas de tamaño variable que representan un algoritmo, y tienen en cuenta la estructura de la cadena a la hora de aplicar los operadores genéticos que manipulan estas cadenas.

Teniendo en cuenta que las plantillas tienen un tamaño fijo y los algoritmos tienen un tamaño variable y una estructura determinada, resulta evidente por qué se ha desarrollado el GA para la búsqueda de la plantilla, y el GP para la búsqueda del algoritmo.

Concretando, en el presente proyecto se ha desarrollado un paquete que consta de cuatro programas, que podemos clasificar dos a dos tanto por su naturaleza (síntesis / análisis) como por las resultados que obtiene y manipula (plantillas / algoritmos). Son los siguientes:

- ***GA Template***: Según la clasificación propuesta anteriormente sería síntesis-plantilla. Trata de obtener automáticamente la plantilla que introducida en la red realice la operación especificada por el usuario mediante el conjunto de imágenes de entrada y de salida deseada, con el mayor parecido posible entre las imágenes de salida resultante y deseada, con opción a tener en cuenta otros parámetros a optimizar, como son el tiempo de convergencia de la plantilla y la magnitud de los valores de la plantilla. Para ello utiliza la técnica de los algoritmos genéticos.

- ***Template Simulator***: Le correspondería el tipo análisis-plantilla. Realiza la simulación de la plantilla que introduzca el usuario, que puede ser una obtenida de forma automática mediante el programa anterior, o bien puede ser introducida directamente por el usuario. De manera que el programa podría ser utilizado para medir la calidad de la plantilla obtenida mediante técnicas genéticas, o bien para que el usuario refine una plantilla simplemente por prueba y error.

- ***GP algorithm***: Tipo síntesis-programa. El objetivo de este programa es obtener automáticamente un algoritmo que realice un procesado de imagen más complejo que el que se obtiene simplemente con una sola plantilla, mediante la combinación de procesado de varias plantillas. Las plantillas que entran en juego en el programa pertenecerán a un conjunto de plantillas base, que el usuario se encargará de configurar a partir de la biblioteca de plantillas que se ofrece con el programa, la cual puede ser ampliada por el usuario utilizando para ello los dos programas anteriores. El programa se representa con una estructura de árbol.

- ***Algorithm simulator***: Tipo análisis-programa. Con este programa se puede validar el funcionamiento adecuado del programa obtenido con distintas entradas, así como obtener el tiempo total de procesamiento, suma de los tiempos de procesamiento de las plantillas que utilice el programa. Además de cargar un programa obtenido automáticamente mediante la utilización del programa anterior se permite la edición por parte del usuario para la mejora o el diseño completo de un programa.

En la figura 2.1 se muestra un esquema en el que se pone de manifiesto la interacción de los distintos programas con el usuario y entre ellos.

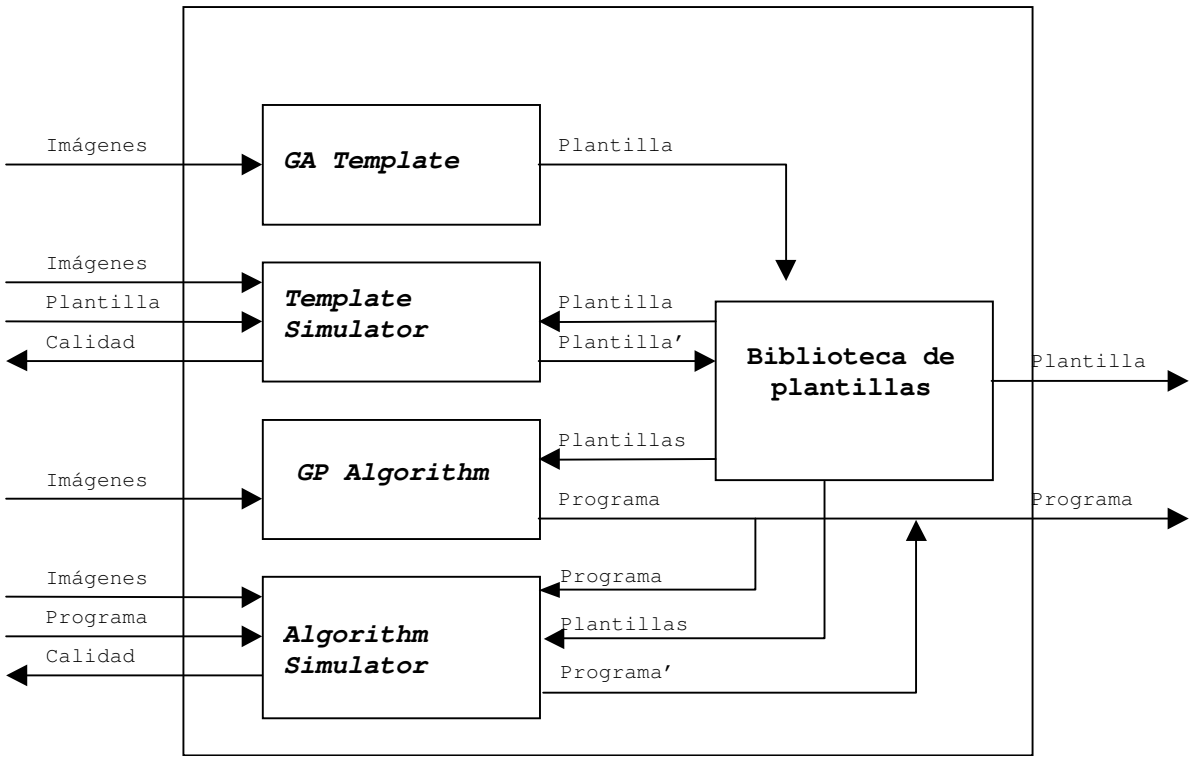


Figura 2.1 Estructura de la interacción de los programas

Como se puede observar en el esquema el resultado que se obtiene es, bien una plantilla que realice un procesamiento muy específico que no se encuentre entre las plantillas de la biblioteca proporcionada, o bien un programa, un algoritmo de procesamiento que combine el conjunto de plantillas base que el usuario haya configurado.

3 INTRODUCCIÓN A LAS CNN

3.1 CNNS: BASES TEÓRICAS

3.1.1 PRELIMINARES

El concepto de CNN es un concepto reciente, su introducción se debe a L.O. Chua y L. Yang que en 1988 [Chua 88a], [Chua 88b] sentaron las bases teóricas que lo definen, aparte de mostrar diferentes aplicaciones prácticas del mismo. La idea original ha evolucionado con gran rapidez pero siempre reteniendo sus dos características básicas: interconexión física, en un ámbito local, de elementos procesadores - o células - operando en paralelo, dispuestos en forma de matriz (usualmente de 2 dimensiones, pero extendiéndose a n dimensiones en el concepto general) y el carácter analógico de la dinámica del proceso y de las interacciones entre células.

Estas particulares redes de neuronas tienen a la visión artificial entre las más destacadas y desarrolladas de sus aplicaciones prácticas [Crounse 95] puesto que han probado ser realmente eficaces en tareas de procesamiento de imágenes. Pero el actual interés por esta tecnología no reside únicamente en esta capacidad por sí sola, sino en la velocidad de cómputo dada por la naturaleza del sistema, que es capaz de alcanzar el orden de 10^{12} operaciones por segundo en circuitos integrados de tan sólo 2 cm^2 [Espejo 94], [Espejo 96].

3.1.2 EL CONCEPTO BÁSICO DE CNN

En este apartado se describirá el concepto de CNN según la visión original de L. O. Chua y L. Yang [Chua 93b] en su acepción más simple. Una CNN puede considerarse como una matriz de circuitos dinámicos ("procesadores") idénticos y de naturaleza no lineal. En el caso más simple, que es el aquí considerado, la matriz se asimila a una red geométrica de 2 dimensiones, esto es; los circuitos básicos que componen la CNN se sitúan ordenados físicamente en una estructura de dos dimensiones. Aunque el concepto de CNN no se limita a una disposición en red rectangular en filas y columnas esto es lo más usual (otras posibilidades son las redes triangulares, redes hexagonales,…etc, como se puede observar en la figura 3.1). Cada nodo de la red, que supondremos rectangular de ahora en adelante, está ocupado por lo que hasta ahora hemos definido como circuito dinámico o "procesador" y al que nos referiremos en lo siguiente como célula (lo cual es más significativo si tenemos en cuenta que la traducción literal de CNN es: "red celular de neuronas").

La estructura de una célula se puede observar en la figura 3.2, donde es posible además observarla en forma de su equivalente eléctrico. Como se ve en dicha figura, la célula se puede considerar constituida por resistencias, condensadores y fuentes de corriente controladas en tensión. El núcleo de la célula está constituido por el condensador C y la resistencia R_x que al estar en paralelo soportarán una misma tensión a la que llamaremos tensión de estado de la célula: V_{xij}, suponiendo como hemos dicho anteriormente una red rectangular de células, esto es las células serán nodos de una red distribuida en filas (subíndice "i") y columnas (subíndice "j").

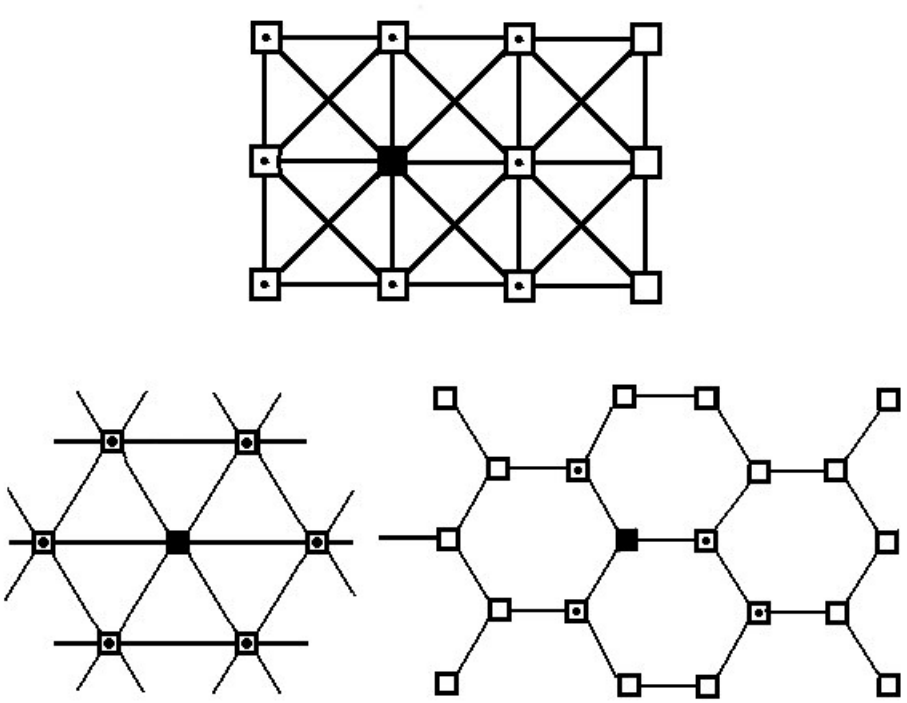
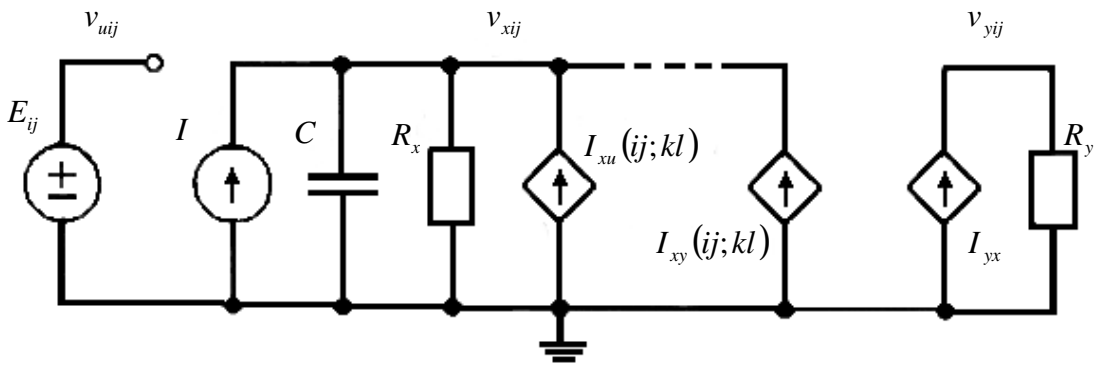


Figura 3.1. Distintas topologías de CNN



$$I_{xy}(ij;kl) = A_{ij;kl}(v_{ykl}, v_{yij}) \qquad I_{xu}(ij;kl) = B_{ij;kl}(v_{ukl}, v_{uij})$$

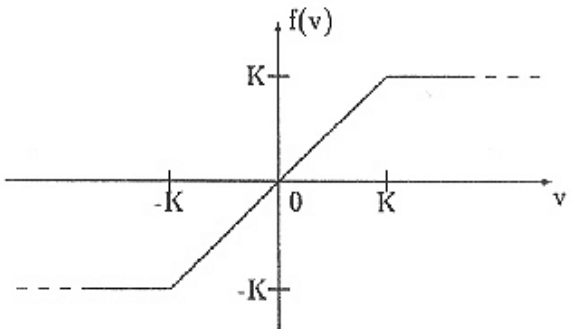
$$I_{yx} = f(v_{xij})$$

Figura 3.2 Estructura de una célula en forma de su equivalente eléctrico

Esta tensión V_{xij} será la que gobierne la tensión de salida de la célula (i,j) puesto que dicha tensión se obtiene como la tensión sobre una resistencia R_y obtenida al actuar sobre ella una fuente de corriente dependiente I_{yx} controlada por V_{xij}. La relación $I_{yx}=f(V_{xij})$ es no lineal (únicamente lineal en un intervalo) y matemáticamente viene dada por:

$$f(V) = \frac{1}{2} \cdot (|V + K| - |V - K|)$$

De esta forma los valores de V_{xij} menores en valor absoluto a K>0 nos darán una I_{yx} de valor V_{xij}, y para los valores que no cumplan la restricción anterior tendremos un efecto limitador de la corriente que quedará fijada en -K para aquellos valores de V<=-K y en K para tensiones tales que V>=K, según se ve más claramente en la figura 3.3.



$$f(v) = \frac{1}{2}\left(|v + K| - |v - K|\right)$$

Figura 3.3 Tensión de salida de una célula en función de su tensión de estado.

Además, cada célula recibe señales externas (señales analógicas de vídeo por ejemplo, en el caso de las aplicaciones de tratamiento de imágenes) V_{uij} y también recibe la corriente suministrada por una fuente de corriente independiente I_{ij}. Y no son sólo estos elementos los que influyen en la tensión de estado V_{xij} de la célula, sino que debemos sumar aún una realimentación mediante la cual la tensión de salida V_{yij} influirá en V_{xij} así como por efecto de las transconductancias de igual forma influirán las tensiones de salida de las células vecinas. Para la mejor comprensión de lo relatado basta con observar la célula como el circuito eléctrico de la figura 2, de esta figura es también fácil deducir *la ecuación de estado,* que es la ecuación diferencial que determina la dinámica de la tensión de estado de la célula Cel(ij) (fila i, columna j, supuesta una matriz rectangular de dimensiones m x n de células) :

$$C\frac{dV_{xij}(t)}{dt} = -R_x \cdot V_{xij}(t) + \sum_{Cel(kl)\in N_r(ij)} A(ij;kl)V_{ykl}(t) + \sum_{Cel(kl)\in N_r(ij)} B(ij;kl)V_{ukl} + I_{ij}$$

Donde:

$$V_{ykl} = \frac{1}{2} \cdot (|V_{xkl} + K| - |V_{xkl} - K|)$$

con i, j variando de 1 a m, y de 1 a n respectivamente. Considerando parámetros normalizados (esto es C=1, R_x=1, R_y=1), así como entrada y salida también normalizados (V_{uij} limitado al intervalo [-1,1] y K=1 en la Fórmula 3 con lo que se consigue también que V_{yij} tome valores únicamente dentro de [-1,1]), el modelo quedaría definido por:

$$\frac{dV_{xij}(t)}{dt} = -V_{xij}(t) + \sum_{Cel(kl)\in N_r(ij)} A(ij;kl)V_{ykl}(t) + \sum_{Cel(kl)\in N_r(ij)} B(ij;kl)V_{ukl} + I_{ij}$$

$$V_{ykl} = \frac{1}{2} \cdot (|V_{xkl} + 1| - |V_{xkl} - 1|)$$

Para comprender el significado de los subíndices de los sumatorios que aparecen en estas ecuaciones, deberemos antes definir el concepto de vecindad.

Diremos que una célula Cel(kl) pertenece al conjunto de vecinos de radio r; $N_r(ij)$ de una célula dada Cel(ij) si se verifica que:

$$Cel(kl) \in N_r(ij) \longrightarrow max\{|k-i|, |l-j|\} \leq r$$

Concepto que aunque ilustrado de forma matemática resulta fácilmente inteligible. No obstante las células "fronterizas", aquellas Cel(ij) en las que i, j ó ambas resultan ser menores de r, no poseen tantas "células vecinas" como el resto, por lo q ue para evitar problemas es práctica habitual suponer que la matriz de células de dimensiones m x n es de alguna forma real o virtual ampliada a (m+r) x (n+r) dimensiones, donde las células fronterizas toman un valor predeterminado que supone lo que conocemos como "condiciones de frontera".

Así pues, la evolución del estado (tensión de estado V_{xij}) de la célula Cel(ij) viene en gran parte condicionada por los coeficientes A(ij;kl) y B(ij;kl) que establecen perfectamente la interacción entre una célula y sus vecinas y que podemos suponer constituyen dos matrices, $A=\{a_{kl}\}$ y $B=\{b_{kl}\}$, de (2r+1)x(2r+1) dimensiones (siendo r el radio o grado de vecindad considerado). Nótese que se han definido dos matrices cuyos coeficientes a_{kl} y b_{kl} han sido considerados independientes de la célula en cuestión Cel(ij), pese a que en general no tiene por qué ser así sino que estos coeficientes podrían ser distintos para cada una de las células, es decir serían función de i y j, o lo que es lo mismo de la posición (fila, columna) de la célula dentro de la red. Si así se ha hecho ha sido porque es mucho más común encontrarse con que A(ij;kl) y B(ij;kl) son en realidad A(kl) y B(kl), es decir independientes de i y j.

También se da la misma situación en el caso del término I_{ij} de la ecuación que define la dinámica de la célula, es decir lo usual es que I_{ij} sea una constante igual para todas las células de la red, y por tanto en la mayoría de los casos podremos escribir:

$$C\frac{dV_{xij}(t)}{dt} = -R_x \cdot V_{xij}(t) + \sum_{Cel(kl)\in N_r(ij)} a_{kl} V_{ykl}(t) + \sum_{Cel(kl)\in N_r(ij)} b_{kl} V_{ukl} + I$$

Utilizando una traducción del inglés al castellano coherente con las nomenclaturas utilizadas usualmente en bibliografía podemos referirnos a A(ij;kl) como *matriz de realimentación*, a B(ij;kl) como *matriz de control*, y a I_{ij} como *término de desplazamiento* o *desplazamiento* a secas, y al conjunto de matriz de realimentación, matriz de control y desplazamiento: A(ij;kl), B(ij;kl) e I_{ij} como *plantilla.*

Una plantilla define completamente la dinámica del sistema así como la interacción entre células vecinas. Hasta ahora se han visto plantillas de coeficientes constantes bien sea el caso general A(ij;kl), B(ij;kl), I_{ij} como el caso de plantillas invariantes en el "espacio": A(kl), B(kl), I, como se apuntaba anteriormente. Pero las capacidades de las CNN pueden verse incrementadas considerando interacciones entre células vecinas no lineales e interacciones con los estados anteriores (en el tiempo) o de tipo retardado. Para representar una interacción no lineal utilizaremos la nomenclatura: $A_{ij;kl}(V_{ykl},V_{yij})$ y $B_{ij;kl}(V_{ukl},V_{uij})$, por otra parte las interacciones de tipo retardado se simbolizarán aquí como: $A^t_{ij,kl}$ y $B^t_{ij,kl}$, de forma que la ecuación de estado puede escribirse en su forma más general como:

$$C\frac{dV_{xij}(t)}{dt} = -R_x \cdot V_{xij}(t) + I_{ij} + \sum_{Cel(kl)\in N_r(ij)} A_{ij;kl}(V_{ykl},V_{yij}) +$$
$$+ \sum_{Cel(kl)\in N_r(ij)} B_{ij;kl}(V_{ukl},V_{uij}) + \sum_{Cel(kl)\in N_r(ij)} A^t{}_{ij;kl} V_{ykl} + \sum_{Cel(kl)\in N_r(ij)} B^t{}_{ij;kl} V_{ukl}$$

Un ejemplo de CNN con interacciones no lineales podría ser:

$$A = \begin{bmatrix} 0 & a & 0 \\ a & 1 & a \\ 0 & a & 0 \end{bmatrix}, B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, I = 0,$$

obteniéndose el parámetro "a" como:

$$a = \begin{cases} 0 \text{ si } V_{ykl} - V_{yij} < 0 \\ 0{,}25 \cdot (V_{ykl} - V_{yij}) \text{ si } V_{ykl} - V_{yij} >= 0 \end{cases}$$

La plantilla que constituye este ejemplo es como se puede observar invariante en el espacio, pues los coeficientes de las matrices A y B son idénticos para cualquier célula Cel(ij).

En la práctica se codifica una imagen en escala de grises de forma que un valor de tensión 1 en una célula representa a un pixel negro y un valor -1 a uno blanco, representando los valores intermedios a pixeles niveles de gris tanto más próximos al negro (blanco) tanto más próximos estén al valor de tensión 1 (-1). La aplicación de esta última plantilla a una CNN en la que los pixels de la imagen así codificada constituyan el estado inicial de la misma: $V_{xij}(t=0)$, se dará como resultado una salida $V_{yij}(8)$ tal que será igual al máximo de $V_{xij}(t=0)$ para cualquier i y j.

3.1.3 LOS MODELOS ISR Y FSR

Recientemente se han utilizado modelos diferentes al original de L.O. Chua y L. Yang a la hora de realizar implementaciones hardware de las CNNs. A continuación se comentarán los modelos *ISR (Improved Signal Range)* y *FSR (Full Signal Range)* [Espejo 96].

Hasta lo aquí expuesto únicamente se había considerado no linealidad en la función que define la salida del sistema a partir de la variable de estado del mismo, en el modelo que aquí se describe también se consideran no linealidades en el término al que en bibliografía se describe como *"pérdidas locales"*, ($-V_{xij}(t)$). De forma que $V_{xij}(t)$ se ve sustituido por una función lineal a trozos $g_m(x)$ definida, para el caso de variables de entrada V_{uij} y salida V_{yij} limitadas al intervalo [-1,1], como:

$$g_m(V) = \begin{cases} m \cdot (V-1) + 1, & \text{si } V > 1 \\ V, & \text{si } |V| \le 1 \\ m \cdot (V+1) - 1, & \text{si } V < -1 \end{cases}$$

De esta forma la dinámica de cada célula vendrá definida por la ecuación:

$$C\frac{dV_{xij}(t)}{dt} = -R_x \cdot g_m(V_{xij}(t)) + \sum_{Cel(kl)\in N_r(ij)} A(ij;kl)V_{ykl}(t) + \sum_{Cel(kl)\in N_r(ij)} B(ij;kl)V_{ukl} + I_{ij}$$

esta es la ecuación que define al modelo ISR. El modelo FSR no es más que un caso particular e ideal del modelo ISR, aquel en que el parámetro m que define g_m tiende a ser infinito. De igual forma se puede observar que el modelo original de L.O. Chua y L. Yang se corresponde con el descrito con el modelo ISR para m=1.

Mediante el modelo ideal FSR se consigue limitar el rango de variación de la variable de estado V_{xij} al mismo intervalo de variación de las variables de entrada, V_{uij}, y salida, V_{yij}, de hecho la variable de estado y la de salida se confundirán a todos efectos, es decir, $V_{xij}= V_{yij}$. Esta propiedad tiene múltiples ventajas a la hora de materializar un CNN en forma de circuito integrado.

A partir la ecuación (con parámetros, entrada y estado inicial normalizados) que define la dinámica del sistema en el modelo original se puede acotar el valor de la variable de estado del sistema V_{xij}:

$$V_{xij}(t) = V_{xij}(0)\cdot e^{-t} + \int_0^t \Big(\sum_{Cel(kl)\in N_r(ij)} A(ij;kl)V_{ykl}(t) + \sum_{Cel(kl)\in N_r(ij)} B(ij;kl)V_{ukl} + I_{ij} \Big) \cdot e^{\theta - t} d\theta$$

$$V_{xij}(t) = V_{xij}(0)\cdot e^{-t} + \left[\sum_{Cel(kl)\in N_r(ij)} B(ij;kl)V_{ukl} + I_{ij} \right](1-e^{-t}) + \int_0^t \sum_{Cel(kl)\in N_r(ij)} A(ij;kl)V_{ykl}(t) \cdot e^{\theta - t} d\theta$$

$$\left|V_{xij}(t)\right| \le \left|V_{xij}(0)\right| + \max_{ij}\left[\sum_{Cel(kl)\in N_r(ij)}\left|B(ij;kl)\right|\right] + \left|I_{ij}\right| + \int_0^t \max_{ij}\left[\sum_{Cel(kl)\in N_r(ij)}\left|A(ij;kl)\right|\right] e^{\theta - t} d\theta$$

$$\left|V_{xij}(t)\right| \le \left|V_{xij}(0)\right| + \max_{ij}\left[\sum_{Cel(kl)\in N_r(ij)}\left|B(ij;kl)\right|\right] + \left|I_{ij}\right| + \max_{ij}\left[\sum_{Cel(kl)\in N_r(ij)}\left|A(ij;kl)\right|\right]$$

Lo que en el caso de plantillas invariantes en el espacio (caso habitual) se traduce en:

$$\left|V_{xij}(t)\right| \le \left|V_{xij}(0)\right| + \sum_{Cel(kl)\in N_r(ij)}\left|b_{kl}\right| + \sum_{Cel(kl)\in N_r(ij)}\left|a_{kl}\right| + |I|$$

es decir:

$$\left|V_{xij}(t)\right| \le 1 + \sum_{Cel(kl)\in N_r(ij)}\left|b_{kl}\right| + \sum_{Cel(kl)\in N_r(ij)}\left|a_{kl}\right| + |I|$$

lo cual representa en la mayoría de las plantillas de aplicación encontradas una cota superior para el valor absoluto de V_{xij} en el intervalo [4, 20].

Este rango de valores posibles para V_{xij} obliga a usar bloques de circuitería que lo soporte, frente a otros bloques que únicamente han de soportar un rango de 4 a 20 veces menor. El uso de circuitería distinta para estos distintos bloques pone de manifiesto problemas de precisión a la hora de considerar los coeficientes (a_{kl}, $b_{kl,}$ I) frente al valor unitario negativo del término de pérdidas: $(-1)V_{xij}$. A este problema se une el propio hecho de la elección de esos bloques de circuitería soportando un valor fijo y alto de tensión, o la posibilidad de cambiar el parámetro de normalizado de los valores mediante circuitería programable. Cualquiera de estas alternativas supone un aumento de la complejidad del circuito. Además,

- La complejidad del circuito integrado disminuye aumentando así la densidad espacial de células en un circuito integrado.
- Al estar más limitado el rango de V_{xij} disminuye el consumo de energía.
- El circuito se hace menos sensible a variaciones uniformes en los parámetros del sistema.

En el modelo original los valores de la variable de estado en la región de saturación pueden ser considerados como una medida de la inercia de la variable de salida V_{yij} para permanecer en saturación. En el modelo FSR no ocurre así, en el momento en que la variable de estado toma el valor normalizado +1 (-1), sigue tomando ese valor hasta que su derivada sea no positiva (no negativa) de esta forma no se acumula ‘inercia” y la salida de la región de saturación será (caso de ocurrir) más rápida.

La velocidad de proceso es aproximadamente el doble que en el modelo original a igualdad de parámetros.

Estas propiedades son aplicables, como se ha dicho, al modelo FSR. El modelo ISR se aproxima a este comportamiento a medida que crece el parámetro m (recuérdese que tanto el modelo FSR como el modelo de L.O. Chua y L. Yang pueden considerarse casos particulares del modelo ISR). En el modelo ISR se puede probar que el valor final de la variable de estado también se encuentra acotado:

$$\lim_{t\to\infty}\left|V_{xij}(t)\right| \le \frac{m-1+M_{ij}}{m}$$

siendo:

$$M_{ij} = \left|\sum_{Cel(kl)\in N_r(ij)} b_{kl}V_{ukl} + I\right| + \sum_{Cel(kl)\in N_r(ij)} \left|a_{kl}\right|$$

En las implementaciones reales el valor de m será siempre finito, y por tanto en realidad es el modelo ISR el modelo a considerar. No obstante para valores de m suficientemente puede considerarse que la CNN posee las propiedades del modelo FSR.

Las figuras 3.4 y 3.5 muestran de forma esquemática posibles implementaciones de los modelos FSR e ISR considerando únicamente la realimentación (esto es suponiendo nulos tanto el término de desplazamiento I, como los coeficientes de la matriz de control b_{kl}).

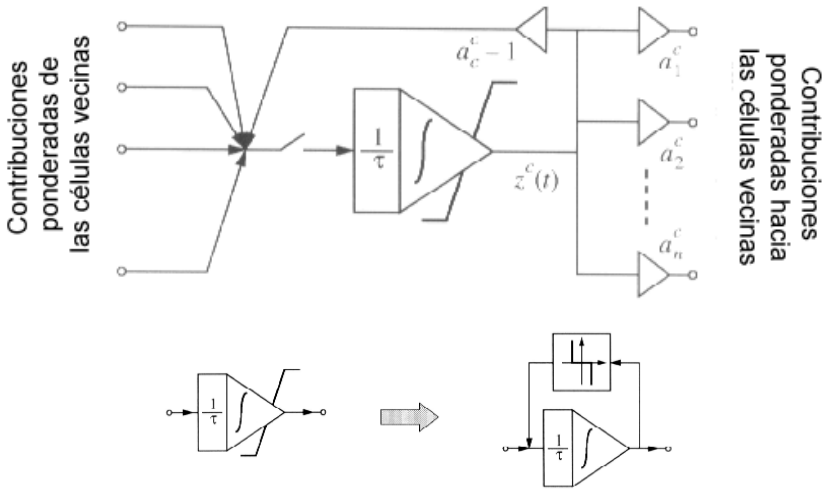


Figura 3.4 Estructura de una célula según el modelo FSR

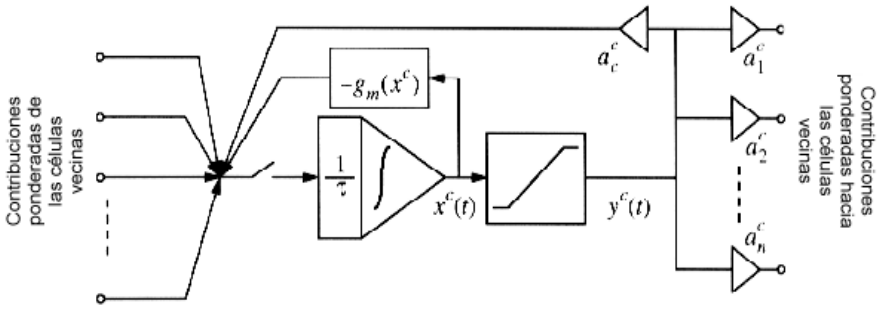


Figura 3.5 Estructura de una célula según el modelo ISR

3.1.4 EL MODELO DE CNN DISCRETA EN EL TIEMPO

El modelo de CNN discretas en el tiempo o DTCNN (Discrete-Time CNN) [Harrer 93] constituye un tipo especial de CNN en el que el comportamiento dinámico de la misma está definido por una realimentación de señales binarias en instantes temporales definidos por un elemento que actúa como reloj del sistema, es decir el tiempo será una variable discreta y consecuentemente las variables de estado y salida también lo serán.

La dinámica del sistema viene definida por las siguientes ecuaciones (suponiendo parámetros normalizados):

$$V_{xij}(K) = \sum_{Cel(kl)\in N_r(ij)} A(ij;kl)V_{ykl}(K) + \sum_{Cel(kl)\in N_r(ij)} B(ij;kl)V_{ukl} + I_{ij}$$

$$V_{yij}(K) = \begin{cases} 1 & \text{si } V_{xij}(K-1) > 0 \\ -1 & \text{si } V_{xij}(K-1) < 0 \end{cases}$$

Por tanto a diferencia de las CNNs, la dinámica del sistema está gobernada por un reloj y únicamente podrán tratarse valores binarios (-1 y +1), según el criterio de normalización seguido. Antes de comenzar a iterar se han de definir los valores iniciales $V_{yij}(0)$, estando los valores de la entrada V_{uij} limitados a un conjunto contínuo de valores. La salida está indefinida para $V_{xij}=0$ pero en la práctica la existencia de ruido hace que no se dé este caso.

Este tipo particular de CNNs posee, aparte sus limitaciones, 2 ventajas frente a los modelos con variables contínuas:

- El estado K+1 puede obtenerse fácilmente a partir del estado K mediante una serie de desigualdades lineales. Los coeficientes de las plantillas se obtienen resolviendo este sistema de desigualdades, partiendo para cada instante de tiempo de la salida a obtener.

- La simulación mediante software es muy sencilla, puesto que no es necesario un algoritmo que resuelva la ecuación diferencial que se plantea en los otros tipos de CNNs y el sistema es discreto en el tiempo.

3.1.5 MODELOS FÍSICOS QUE EMULAN CNNS EN PARALELO

Tal y como se ha definido una CNN es una matriz n-dimensional de sistemas dinámicos idénticos, pero a efectos de implementación práctica como circuito integrado VLSI, el integrar matrices tridimensionales (o multicapa) supone una grandísima dificultad frente lo que es el caso habitual: matrices bidimensionales (una sola capa), rectangulares. El precio que se ha de pagar por esta relativa sencillez de los circuitos es la pérdida de potencia, entendida esta como la capacidad de realizar operaciones más complejas, de la CNN.

Multitud de problemas de procesado de señales (imágenes en particular) no pueden ser resueltos con una CNN bidimensional siendo su resolución posible mediante circuitos más complejos como las CNN multicapa. Sin embargo, el problema a resolver puede ser reformulado como una serie de operaciones sencillas cada una de las cuales son realizables por CNNs bidimensionales. Estas operaciones o tareas pueden ser realizadas simultáneamente de forma que el valor de la variable de estado de cada célula se obtiene como una función lógica (lógica analógica) de los resultados parciales de cada operación. La estructura adecuada para operar de esta forma pasa por la conexión en paralelo de varias CNNs de una sola capa, a las que se "alimenta" con una misma entrada, mientras que para cada conjunto de células en paralelo un bloque (un circuito) ha de implementar la función lógica (lógica analógica) que define la salida del sistema a partir de la salida de cada una de esas células.

Llegar a realizar un circuito de CNNs en paralelo según se ha descrito implica enormes dificultades, no obstante se han propuesto [Slot 93] modelos en los que la estructura de la célula individual se ve modificada para poder procesar simultáneamente varias operaciones.

La célula modificada difiere del modelo original de célula en la parte que define la tensión de estado (estado) de la célula. El estado de la célula estará en este caso controlado por varios sub-circuitos a los que se les hará llegar la misma entrada, el valor analógico obtenido por cada uno de estos circuitos se hará llegar a una estructura compuesta de diodos y resistencias que implementará la "lógica analógica" que definirá el valor de la tensión de estado.

Para solventar un determinado problema de este tipo se habrá de definir una plantilla por cada una de las operaciones básicas a realizar en paralelo, más la estructura de la parte diodo-resistiva que implemente la función "lógica analógica" adecuada.

La figura 3.6 muestra la estructura de la célula modificada para obtener la variable estado del sistema como un **"OR analógico" ((V1)OR(V2)=máx{V1,V2})** de las variables de estado de dos operaciones distintas procesándose en paralelo.

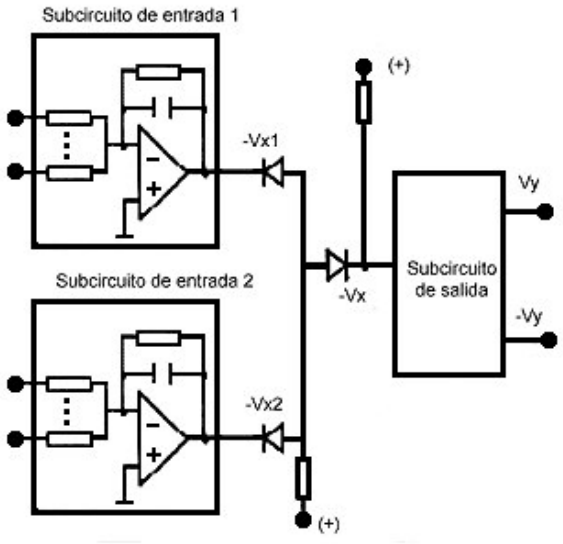


Figura 3.6 Estructura de la célula modificada.

3.1.6 OTROS MODELOS

Tomando como punto de partida el modelo original y como objetivo la solución de determinado problema se han desarrollado múltiples modelos de CNNs especialmente adaptados para la realizar la tarea objetivo. Tal es el caso de las CNNs con estado inicial dependiente de la entrada, especialmente adecuado para la clasificación en regiones no separables linealmente, en particular la detección de bordes [Genç 98] o el de el modelo CNN-UBN (CNN Universal Binary Neurons) que realiza la misma tarea de una forma mucho más eficaz que el primero [Aizemberg 98].

No existe sin embargo, hasta la fecha, un modelo que resulte más eficiente y eficaz de forma global que el resto lo cual hace que como fruto del contínuo esfuerzo investigador nuevas alternativas y variantes sigan viendo la luz. Para hacer aún más amplio el abanico de posibilidades a esta gran variedad de modelos se le une el hecho de las diferentes capacidades derivadas del uso de radios de vecindad mayores, el uso de plantillas con interacciones variantes en el espacio, retardadas en el tiempo, etc. que suponen una mayor dificultad (o incluso la imposibilidad) a la hora de la realización práctica.

3.1.7 EPÍLOGO

En resumen: una CNN es una matriz n-dimensional de sistemas dinámicos idénticos llamados células que interactúan con aquellas que constituyen su vecindad (aquellas de la matriz que distan de ella menos de un determinado radio de vecindad r) de forma que las variables de estado de cada célula varían de forma continua con el tiempo (salvo la excepción de las CNN discretas).

Se han considerado diferentes modelos de CNN definidos por la diferente forma de la ecuación que define la dinámica del sistema, de forma que es posible encontrarse variantes (con interacción lineal o no lineal, con interacción invariante o variante en el

espacio, con interacción con los estados anteriores -con retardo- o sin ella) de diferentes modelizaciones (original, ISR, FSR, …).

Partiendo de un estado inicial del valor de cada una de las células así como un valor de entrada supuesto constante (puede ser variable en el tiempo, pero su variación es mucho más lenta que la dinámica de cada célula, de forma que ante un cambio en el sistema los valores de las variables de estado de la célula alcanzan el régimen estacionario mucho antes de que pueda variar la entrada) y en base a una serie de parámetros, el valor de la variable de estado de cada célula evoluciona, originando un valor de salida. Estos parámetros definen la interacción de cada célula con sus vecinas a través de los valores de entrada así como a través de la realimentación de los valores de salida de las células en cada instante. El conjunto de estos parámetros que definen una salida particular de las células recibe el nombre de plantilla ("Cloning template" en la bibliografía original inglesa).

Entre las aplicaciones de las CNNs cabe destacar el tratamiento de imágenes, pudiendo estas realizar múltiples operaciones sobre las mismas. Una operación determinada puede conseguirse mediante la elección adecuada de una plantilla. De esta forma puede considerarse a la plantilla como una operación elemental pudiendo realizarse operaciones complejas concatenando la aplicación de varias plantillas.

La mayor parte de los estudios y descubrimientos hasta la fecha se basan, por su mayor simplicidad (o mejor dicho por su menor dificultad), en CNNs con interacciones lineales, invariantes en el espacio y sin retardo (lo que condiciona la naturaleza de las plantillas) con salidas binarias, es decir; la salida de cada célula de la CNN está limitada a un conjunto de 2 valores ("blanco" y "negro" en el caso del tratamiento de imágenes).

En el presente proyecto se ha trabajado con el modelo original de Chua y Yang, considerando parámetros normalizados (esto es C=1, R_x=1, R_y=1), así como entrada y salida también normalizados (V_{uij} limitado al intervalo [-1,1] y K=1 con lo que se consigue también que V_{yij} tome valores únicamente dentro de [-1,1]), y entorno de vecindad 1. El modelo es el más estudiado, con abundante bibliografía y con una importante biblioteca de plantillas de distinta funcionalidad, además de ser fácilmente implementable en una simulación por computador.

3.2 PARAMETRIZACIÓN DE UNA CNN: LAS PLANTILLAS

3.2.1 INTRODUCCIÓN

Según se ha definido anteriormente, haciendo referencia a la ecuación que gobierna la dinámica de una CNN, utilizando una traducción del inglés al castellano coherente con las nomenclaturas utilizadas usualmente en bibliografía podemos referirnos a A(ij;kl) como *matriz de realimentación*, a B(ij;kl) como *matriz de control*, y a I_{ij} como *término de desplazamiento* o *desplazamiento* a secas, y al conjunto de matriz de realimentación, matriz de control y desplazamiento: A(ij;kl), B(ij;kl) e I_{ij} como *plantilla.*

Esta definición no es en absoluto general, pues según se ha visto existen diferentes formulaciones y tipos de CNN que condicionan la definición de plantilla.

Esta definición se corresponde al caso de CNN con interacciones lineales, variantes en el espacio, sin retardo y según la formulación de L. O. Chua y L. Yang.

Aún a pesar de la falta de generalidad de la definición, esta es incluso demasiado general para lo que suele ser común en la práctica, en el que lo usual es encontrarse con interacciones invariantes en el espacio. De esta forma tendremos $\boldsymbol{a_{kl}}$, $\boldsymbol{b_{kl}}$ e $\boldsymbol{I}$ en lugar de **A(ij;kl), B(ij;kl)** e $\mathbf{I_{ij}}$.

Se supondrá en todo momento una CNN rectangular con una vecindad de radio 1. Consecuentemente las interacciones de cada célula tendrán lugar con las células de filas y columnas inmediatamente adyacentes a la fila y columna de la considerada. Esto es, cada célula interactuará con 9 vecinas (8 vecinas más ella misma), por lo que la dinámica del sistema quedará definida por 19 parámetros (9 de la matriz de realimentación [a_{kl}], 9 de la matriz de control [b_{kl}], más el término de desplazamiento I).

$$\begin{bmatrix} a_{i-1,j-1} & a_{i-1,j} & a_{i-1,j+1} \\ a_{i,j-1} & a_{i,j} & a_{i,j+1} \\ a_{i+1,j-1} & a_{i+1,j} & a_{i+1,j+1} \end{bmatrix} ; \begin{bmatrix} b_{i-1,j-1} & b_{i-1,j} & b_{i-1,j+1} \\ b_{i,j-1} & b_{i,j} & b_{i,j+1} \\ b_{i+1,j-1} & b_{i+1,j} & b_{i+1,j+1} \end{bmatrix} ; \text{ I}$$

Una plantilla por norma general está orientada y diseñada para la consecución de un determinado objetivo: es decir para obtener unos valores de salida ($V_{yij}(\bullet)$ en la nomenclatura aquí usada) a partir de un estado inicial de las células ($V_{xij}(0)$) y una determinada entrada (V_{uij}).

3.2.2 LAS PLANTILLAS Y LOS FENÓMENOS FÍSICOS

Si se examinan las ecuaciones que definen el estado de las células que componen una CNN, se puede observar que, bajo ciertas condiciones, este conjunto de ecuaciones se puede considerar como el que define el valor de una variable de estado de un sistema de partículas discreto en el espacio (2-D o 3-D según que consideremos una CNN con una sola capa o con varias).

Si se tiene en cuenta que una derivada direccional se puede aproximar por:

$$\frac{dx}{di} = \frac{x_{i+h} - x_i}{h}; \quad \frac{\mathrm{d}^2 x}{di^2} = \frac{x_{i+2h} - 2 \cdot x_{i+h} + x_i}{h^2}; \ldots\ldots$$

Considerando que la distancia h en "i" entre c ada célula es la unidad es fácil identificar los coeficientes de las ecuaciones en derivadas que definen el comportamiento un sistema de partículas discreto en el espacio con los coeficientes de una matriz de realimentación de una CNN. Para que la analogía pueda darse es preciso que la variable de estado de todas las células de la CNN se conserve dentro de los valores en que la función que define la salida es lineal.

Así autores como L.O. Chua y L. Yang [Chua 88a] han descrito la relación entre las CNNs y la ecuación de difusión del calor, J. Henseler y P.J. Braspenning [Henseler 93] han estudiado aplicaciones de CNNs simulando la propagación de ondas y también

F. Puffer, R. Tetzlaff y D. Wolf [Puffer 98] han hecho lo propio para modelar mediante CNNs el flujo de fluídos casi incompresibles.

Aunque pueda parecer que esta visión de las CNNs poco tenga que ver con sus posibles aplicaciones a tratamiento de imágenes y visión artificial, no es así, baste señalar la aplicación de autentificación de firmas manuscritas desarrollada por J. Henseler y P.J. Braspenning [Henseler 93], en la que el estudio en frecuencia de la salida obtenida mediante la firma de una persona como estado inicial de un sistema que simula la propagación de ondas, permite clasificar o distinguir al autor de la misma.

El estudio de las CNNs como sistemas en que ondas se propagan en función de los coeficientes de sus plantillas ha dado lugar a interesantes resultados en el campo de la segmentación activa de imágenes y la esqueletización de las mismas como los expuestos en [Vilariño 98] o el trabajo "Computing with front propagation:Adtive contour and skeleton models in continuous-time CNN" de C. Rekeczky y L.O. Chua. Los trabajos mencionados basan sus experiencias en la utilización de distintas plantillas en una secuencia temporal en la que el cambio de una a otra plantilla se realiza mucho antes de que se alcance el régimen estacionario para la plantilla en curso.

3.2.3 ALGUNAS PLANTILLAS DE PROBADA FUNCIONALIDAD

Hasta llegar a este punto se han tratado las plantillas como un conjunto de parámetros que definen inequívocamente una salida determinada a partir de una entrada y un estado inicial dados. Esto no es así, pues la naturaleza dinámica del sistema puede ocasionar que una plantilla dada aplicada a una CNN con un determinado estado inicial y una determinada entrada de lugar a una respuesta (salida) oscilante o bien con multitud de puntos de equilibrio. Multitud de autores han abordado el tema en busca de la determinación de características que garanticen una respuesta convergente hacia un punto de equilibrio único independientemente de la entrada y estado iniciales [Chua 93b], [Madlenov 98], [Arik 98].

Todas las plantillas probadas son plantillas con interacciones lineales sin retardo, de radio de vecindad uno y espacio-invariantes de dominio general que pueden encontrarse en la red Internet. A continuación se muestran algunas plantillas en las que se puede observar la capacidad de procesamiento de la CNN. Solamente se muestran unas pocas de todas las plantillas que se incluyen en la biblioteca de plantillas. En el anexo I se puede ver el resto de plantillas de la biblioteca.

3.2.3.1 Plantillas de umbralización

Esta plantilla convierte una imagen en escala de grises "P", que se cargará como estado inicial de las células de la CNN, en una imagen (salida) binaria donde cada pixel p_{ij} de P es convertido a negro si y sólo si el valor normalizado de gris de p_{ij} es superior al definido por el término de desplazamiento I de la plantilla cambiado de signo.

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix};\ B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix};\ I = \text{umbral}$$

La figura 3.7 muestra los resultados obtenidos sobre una imagen de entrada para los valores umbral 0,55 y 0,15 respectivamente.

Figura 3.7 Imagen inicial y resultado para un valor de umbral de 0,55 y 0,15.

3.2.3.2 Plantilla de Proyección de sombra

Proyecta en una dirección (definida por los elementos de la matriz de realimentación no nulos) los pixels negros de una imagen binaria de entrada partiendo de un estado inicial de pixels negros.

$$A = \begin{bmatrix} 0 & 2 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix};\ B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix};\ I = 0$$

La figura 3.8 muestra los efectos de la plantilla de proyección vertical (hacia abajo) de sombra.

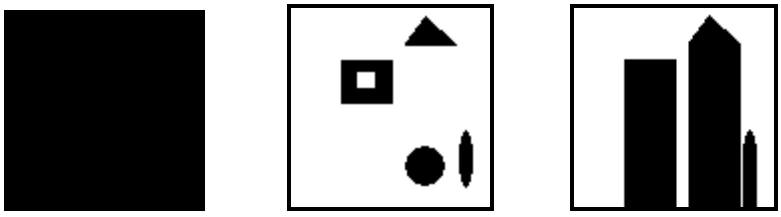


Figura 3.8 Imágenes estado inicial, entrada y salida para la plantilla de proyección de sombra.

3.2.3.3 Plantilla de extracción de los objetos seleccionados

El objetivo de esta plantilla es extraer de una imagen binaria cargada como entrada al sistema únicamente los objetos que se encuentren "marcados" mediante al menos un pixel negro en la imagen que se pasa al sistema como estado inicial.

$$A = \begin{bmatrix} 0{,}25 & 0{,}25 & 0{,}25 \\ 0{,}25 & 1 & 0{,}25 \\ 0{,}25 & 0{,}25 & 0{,}25 \end{bmatrix};\ B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1{,}75 & 0 \\ 0 & 0 & 0 \end{bmatrix};\ I = 0$$

Los resultados obtenidos con esta plantilla se pueden observar en la figura 3.9.

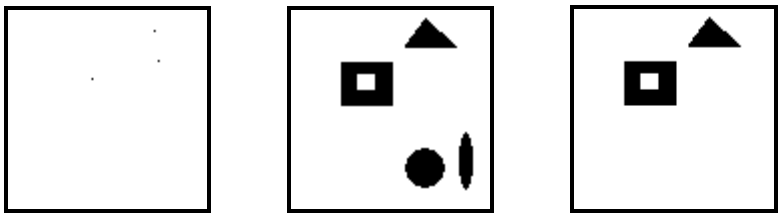


Figura 3.9 Imágenes estado inicial, entrada y salida para la plantilla de extracción de objetos seleccionados

3.2.3.4 Plantilla de detección de objetos conexos

Mediante esta plantilla es posible determinar si un objeto dado es realmente un único objeto, es decir, está formado por una única "pieza" o bien compuesto de dos o más componentes inconexos que podrían considerarse objetos individuales. Esta plantilla borra los objetos marcados de una imagen binaria. Un objeto se considerará marcado si al menos uno de sus pixels negro en la imagen de entrada es blanco en el estado inicial. En la figura 3.10 se muestra un ejemplo de aplicación de la plantilla.

$$A = \begin{bmatrix} 0 & 0{,}5 & 0 \\ 0{,}5 & 3 & 0{,}5 \\ 0 & 0{,}5 & 0 \end{bmatrix}; B = \begin{bmatrix} 0 & -0{,}5 & 0 \\ -0{,}5 & -3 & -0{,}5 \\ 0 & -0{,}5 & 0 \end{bmatrix}; I = -4{,}5$$

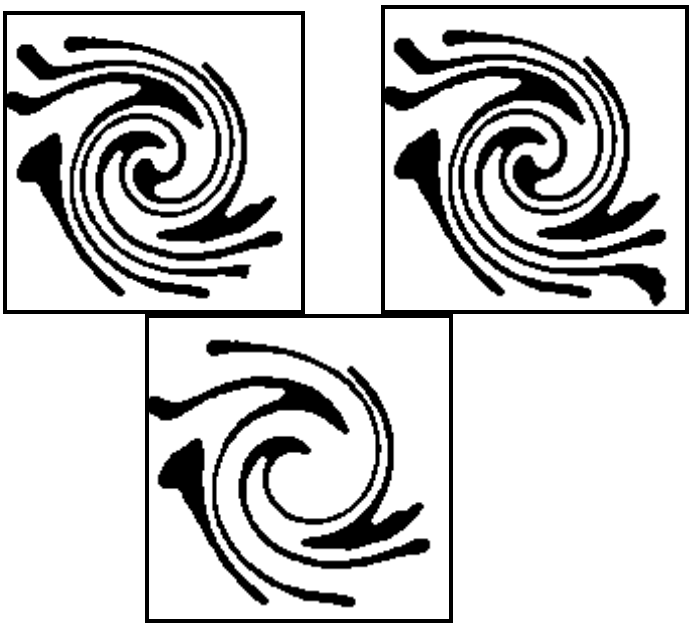


Figura 3.10. Estado inicial, entrada (encima) y salida (debajo) para la plantilla de detección de objetos conexos.

La figura 3.11 es útil a efectos de corroborar la efectividad de esta plantilla y también la plantilla que realiza la operación "diferencia" entre imág enes binarias. En dicha figura se muestra el resultado de la operación diferencia entre la imagen entrada

para la plantilla de detección de objetos conexos y su salida obteniendo, por tanto como resultado de esta sustracción, el objeto conexo borrado.

Figura 3.11 Objeto borrado en la aplicación de la plantilla de detección de objetos conexos.

3.2.3.5 Plantilla de Conectividad de componente direccional

La plantilla descrita en este apartado es capaz de detectar la existencia de una discontinuidad de componente direccional en una figura, en este caso discontinuidad de componente horizontal. Así descrito obteniendo un píxel negro por discontinuidad en cada fila (columna o diagonal) y trasladando adicionalmente esos píxeles hacia columnas alternas (los puntos que representen la detección de un agujero se encontrarán separados, dentro de una fila, por un píxel blanco) en la cercanía de una de las fronteras de la imagen. La plantilla bajo estas líneas realiza el procesamiento especificado trasladando los puntos de detección hacia la frontera derecha de la imagen.

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 2 & -1 \\ 0 & 0 & 0 \end{bmatrix};\ \mathrm{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix};\ \mathrm{I} = 0$$

Dado que la matriz de control tiene todos sus coeficientes nulos, es indiferente cual sea la imagen de entrada. La imagen a tratar se deberá pasar como estado inicial del sistema. En la siguiente figura se muestra un ejemplo, en la que la detección de continuidad de componente coincide con la detección de agujeros.

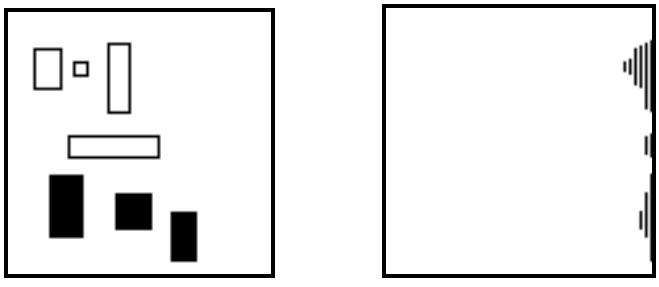


Figura 3.12. Estado inicial y salida para la plantilla de detección de agujeros horizontales

3.3 LA CNN UNIVERSAL MACHINE.

El paradigma de las CNNs ha permitido la definición de un ordenador analógico matricial algorítmicamente programable con capacidad de procesamiento en tiempo real comparable a los supercomputadores, pero en un único chip: la denominada Máquina virtual de CNN (CNN-UM). En las CNN-UM se combinan unidades analógicas programables a través de del peso de interacciones locales , con operaciones lógicas programables y memorias distribuidas tanto analógicas como digitales.

Hasta la aparición de la CNN-UM las ímplementaciones de CNN que se había conseguido desarrollar solamente permitían el procesamiento correspondiente a una única plantilla. La CNN-UM amplía enormemente las posibilidades de procesamiento, pues mediante la programación analógica es posible la combinación de distintas plantillas, así como el control de flujo del programa y la clasificación de resultados. Para ello es necesario añadir a la CNN una serie de elementos auxiliares. En concreto la CNN-UM consiste en una CNN con las siguientes capacidades adicionales:

- Memoria análoga local o distribuida a lo largo del array de células capaz de almacenar señales continuas.
- Memoria binaria local incrustada en cada célula, dándole a la CNNUM la capacidad de almacenar señales de tipo binario.
- Circuitería de control y comunicación en cada célula.
- Unidad global de comunicaciones capaz de comunicarse con cada célula y con el exterior.

Todo esto permite que la CNNUM haga un determinado procesamiento, configurando adecuadamente los valores de la plantilla, y almacenar resultados parciales que pueden ser utilizados para un procesamiento posterior. De esta forma se consigue una combinación de procesamientos con distintas plantillas.

La arquitectura global simplificada de la CNN Universal Machine se muestra en la figura 3.13:

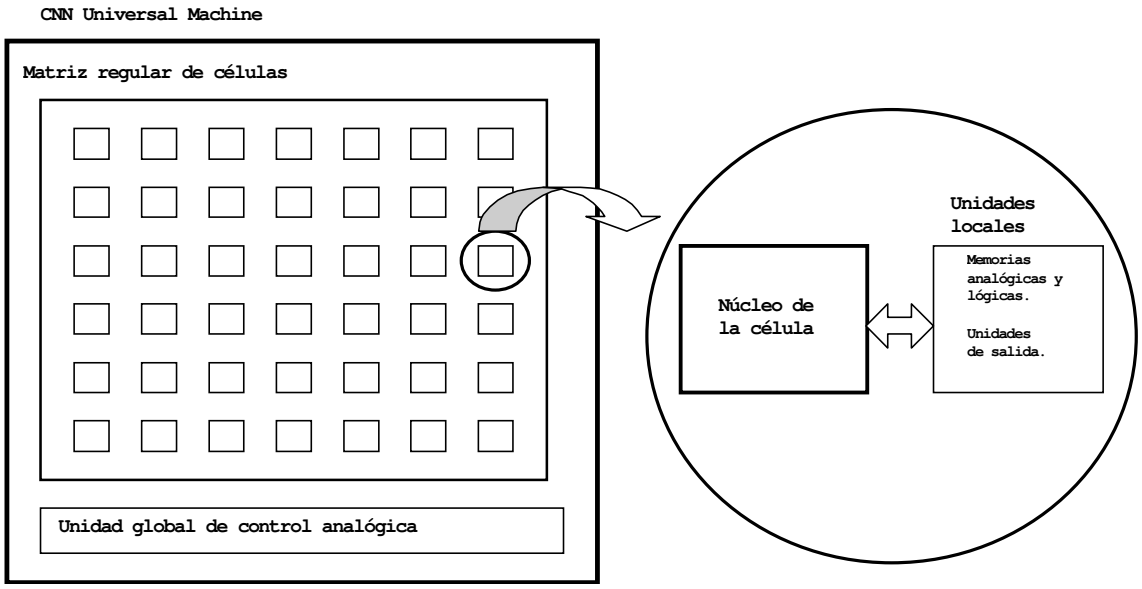


Figura 3.13 Estructura de la CNN-UM

Un programa analógico, consistente en una secuencia de operaciones analógicas y lógicas, describe las tareas de procesamiento. Los programas analógicos que se tratan en el proyecto consideran únicamente operaciones analógicas, que corresponden a la aplicación del procesamiento correspondiente a una determinada plantilla. Para la codificación del programa se ha empleado la estructura de árbol para representar los programas, de manera que la codificación del árbol resultante es el programa codificado. La estructura de árbol es la ideal para la aplicación de la programación genética. De esta forma es posible codificar un amplio conjunto de programas, los cuales deben cumplir que en su estructura no hay ningún tipo de bucle ni sentencia condicional, pues no sería posible codificarlo en un árbol. El resultado de estos programas siempre será la imagen obtenida de la aplicación de las distintas operaciones analógicas definidas en el programa. Los nodos terminales del árbol de programa serán las imágenes de entrada, mientras que los intermedios serán operadores analógicos, que corresponderán a la aplicación del procesamiento asociado a una determinada plantilla.

3.4 EJEMPLOS DE APLICACIÓN EN VISIÓN ARTIFICAL

3.4.1 GENERALIDADES

Para la mayor parte de los problemas de visión no existe una solución obtenida directamente a partir del uso de una plantilla determinada en una CNN, pues a pesar de que una sola plantilla puede ser capaz de realizar una operación relativamente compleja de tratamiento de imágenes, por regla general la complejidad de un problema de visión escapa de las posibilidades de la misma. De manera que en la mayoría de los casos se hace necesaria la aplicación de un programa analógico que implique la combinación del uso de varias plantillas en la determinación de la solución. En los apartados que siguen se ilustrará la resolución de varios problemas de visión artificial mediante CNNs.

En los diagramas que representan los programas se ha adoptado el criterio que se muestra en la figura 3.14 para las plantillas de dos entradas:

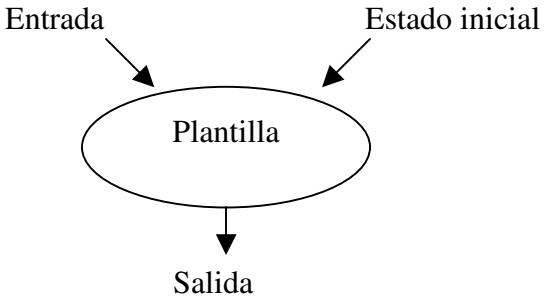


Figura 3.14 Criterio en representación de programas

3.4.2 RECONOCIMIENTO DE CARACTERES MANUSCRITOS

El problema de la extracción de características de caracteres manuscritos puede ser abordado mediante el uso de una CNN. El posterior procesado de las características así extraídas puede ser la entrada de un clasificador que decidirá a qué carácter corresponden tales características.

Las características extraídas deberán poseer un elevado grado de invariabilidad frente a parámetros como puedan ser la posición del carácter en la imagen, la rotación

del mismo, su tamaño y la persona autora del manuscrito. Además las características extraídas deben suponer un volumen de información mucho menor al del conjunto de la imagen que ha sido tratada.

La invariancia a considerar puede ser de dos tipos; invariancia geométrica e invariancia estadística. Mientras posición, rotación y tamaño se asocian a la invariancia geométrica, las consideraciones estadísticas de las imágenes son asociadas a la invariancia de ese mismo tipo. No obstante la invariancia total no es en ocasiones efectiva, pues la invariancia total frente a la rotación no permitiría discernir entre, por ejemplo, "6" y "9", "?" y "¿", "[" y "]" y muchos más caracteres.

Se han desarrollado multitud de algoritmos que realizan esta labor de extracción de características de caracteres manuscritos sin que ninguno se muestre realmente mucho más efectivo que el resto. En este apartado se describirá un extractor de características soportado por una CNN según [Suzuki 93], al que es aplicable el comentario anterior; no se muestra como de efectividad claramente superior al resto de algoritmos pero al menos es eficaz.

A la hora de entrenar el clasificador, el trabajo citado en bibliografía empleó conjuntos de entrenamiento extraídos de la serie de bases de datos para reconocimiento de caracteres manuscritos ETL, concretamente ETL3 que básicamente contiene el alfabeto latino los caracteres numéricos árabes y 12 símbolos más como; "-", "(",… etc. y ETL8B2 conteniendo los 71 caracteres japoneses Hirakana, de mayor dificultad de clasificación (incluso mediante la observación humana resultan difíciles de identificar). Los resultados obtenidos utilizando un 30% aleatorio del conjunto total como conjunto de entrenamiento para la red de clasificación fueron, una tasa del 94,8% de aciertos para los caracteres ETL3 y un 85,7% para los ETL8B2.

La base de la extracción de características es en este caso la aplicación en una CNN de dos clases de plantillas, las plantillas descritas en el presente trabajo como de detección de "agujeros" (horizontales, verticales y también en diagonal a 45º y -45º), plantillas denominadas en bibliografía plantillas CCD o de detección de componentes conexos, y las de proyección de sombra (hacia la derecha o la izquierda, y hacia arriba o abajo). En las figuras 3.15, 3.16, 3.17 se observan los distintos resultados.

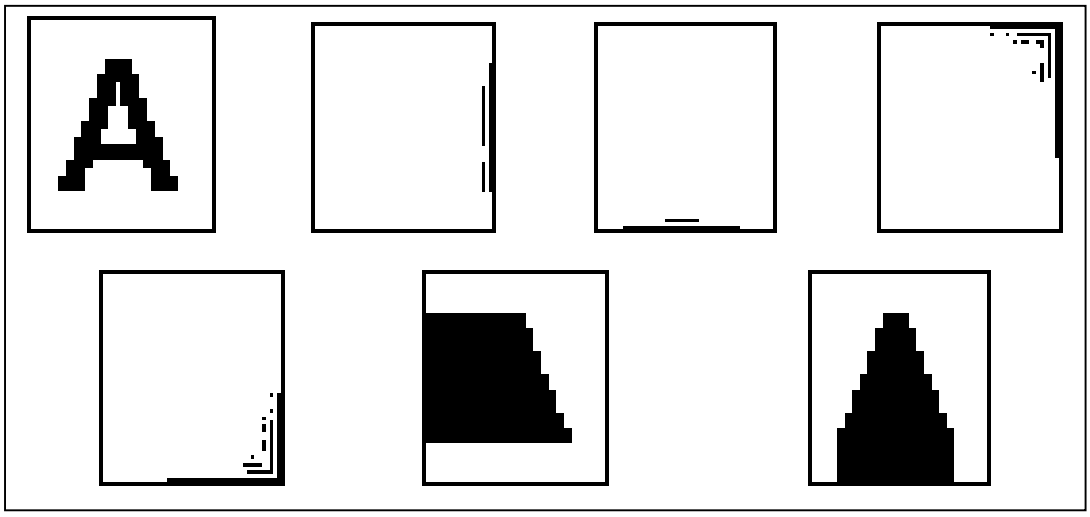


Figura 3.15 Extracción de características para un carácter "A" fuente Arial, negrita, tamaño 12 puntos.

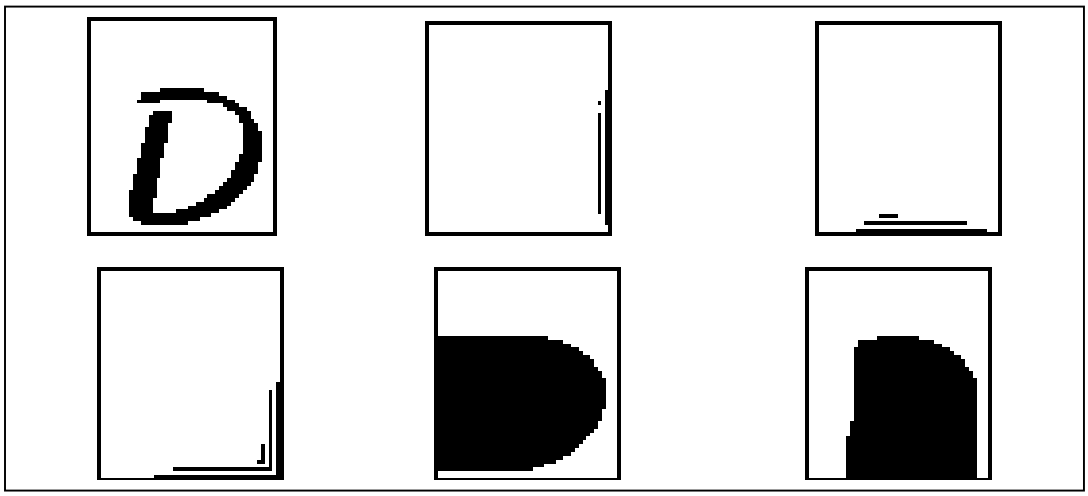


Figura 3.16 Extracción de características para un carácter "D" fuente Rosson SSK, negrita, tamaño 12 puntos.

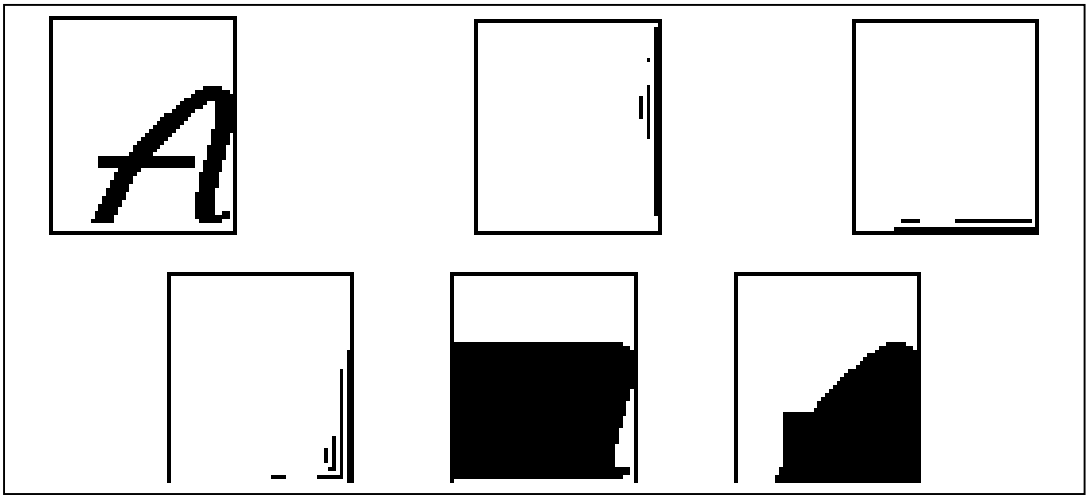


Figura 3.17 Extracción de características para un carácter "A" fuente Rosson SSK, negrita, tamaño 12 puntos.

Estos datos son tratados (básicamente relaciones geométricas entre ellos para conseguir la invariancia de tipo geométrico) y comprimidos de forma que a partir de ellos son el clasificador neuronal efectúa su labor con las cifras de eficacia antes comentadas.

Así con objeto de que los datos obtenidos mediante las plantillas de detección de "agujeros" sean independientes del tamaño del carácter (invariancia respecto al tamaño) el número de pixels de cada una de las capas en la salida (filas o columnas de pixels negros) es dividido entre el número de pixels de la primera capa.

3.4.3 MEDIDA DE RUGOSIDAD MEDIANTE LA BÚSQUEDA DE CONCAVIDADES

Se trata de un programa analógico de CNN que es capaz de medir la rugosidad de la superficie de una figura, para lo cual se trata de buscar las concavidades que se presenten en la misma.

La plantilla más importante del programa es la *Concave Location Filler*, que se encuentra en la biblioteca y rellena las concavidades de la imagen de entrada. El resto de plantillas empleadas son la *Erosion*, *Logic Difference* y *Theshold*. En la figura 3.18 se muestra el árbol correspondiente a este programa.

El resultado del programa es una imagen de salida en la que se muestra con trazo más grueso las zonas más irregulares del contorno de la figura, mientras que se muestra con trazó más fino, o sin trazo, las zonas más suaves. Debido a la poligonización de la plantilla *Concave location filler* se pueden mostrar trazos gruesos en zonas que no presentan ningún tipo de irregularidad. En cualquier caso, el número de píxeles a la salida da una medida aproximada de la rugosidad del contorno. En la figura 3.19 y 3.20 se muestran varios ejemplos. Se puede alterar el número de veces que se aplique la plantilla *Erosion* en dependiendo de la resolución de la imagen que se trate.

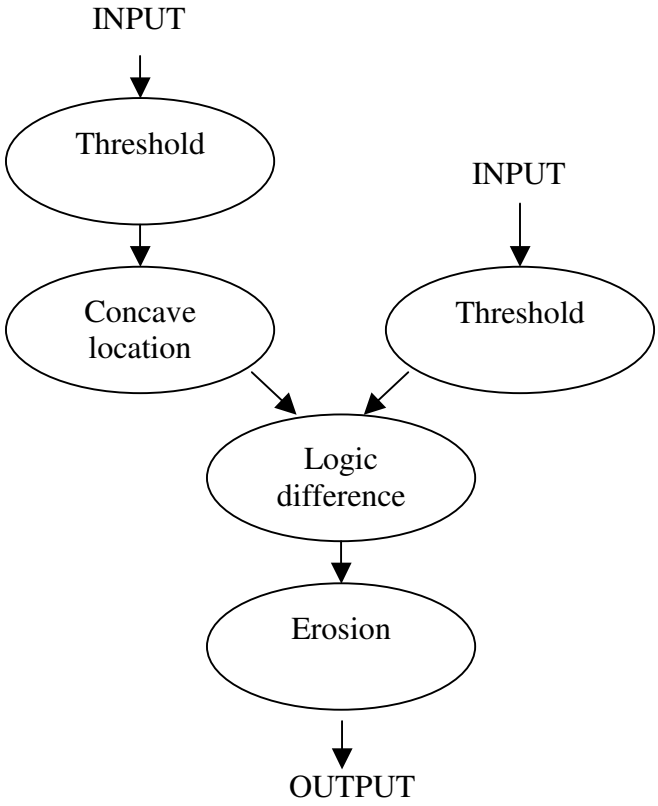


Figura 3.18 Estructura del programa

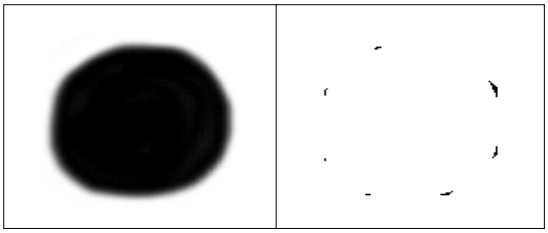


Figura 3.19 Ejemplo de aplicación del medidor de rugosidad

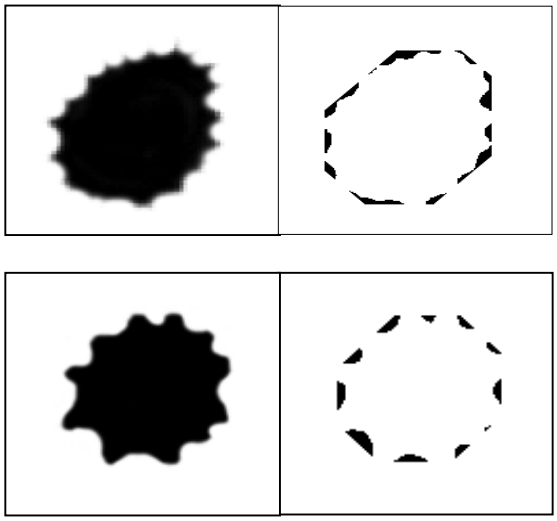


Figura 3.20 Ejemplos de aplicación del medidor de rugosidad

3.4.4 CUENTA DE OBJETOS

Mediante este simple programa se realiza la cuenta del número de objetos en la imagen. La extracción de los objetos dependerá de la situación que tengamos. Se puede utilizar un simple umbralizador en caso de que los objetos se distingan del fondo por su luminosidad. Para un caso más general podemos realizar la diferencia entre la imagen de fondo y la imagen con los objetos mediante la aplicación de la plantilla *Difference Theshold*. Esto es lo que vamos a utilizar para este ejemplo. Suele ser recomendable realizar un suavizado de las imágenes antes de realizar la diferencia, pues de lo contrario cualquier pequeño ruido o imprecisión se mostraría a la salida, alterando así el resultado. También es recomendable aplicar una plantilla para eliminación de objetos pequeños provocados por el ruido. Todo dependerá de las condiciones con que nos encontremos. En la figura 3.21 se muestran las imágenes de entrada y en la 3.22 la imagen de salida. Como se puede observar se indica el número de objetos mediante el mismo número de píxeles en el margen derecho, en la columna adyacente a la columna del borde izquierdo de la imagen. En caso de que se diera el caso de que a varios objetos les corresponden la misma posición en dicha columna, se mostrarían los correspondientes píxeles en columnas adyacentes de la misma fila de forma alternada, con un píxel blanco entre cada dos negros.

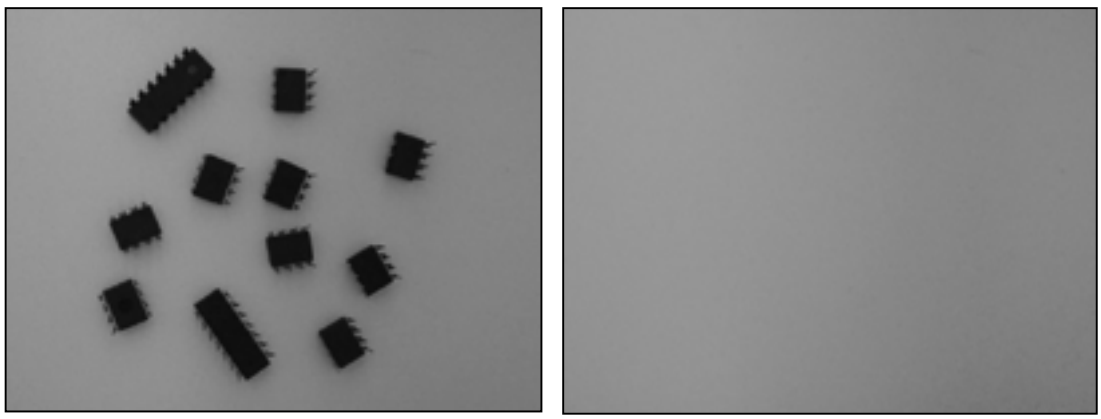


Figura 3.21 Imágenes de entrada. Imagen con objetos e imagen sólo con el fondo

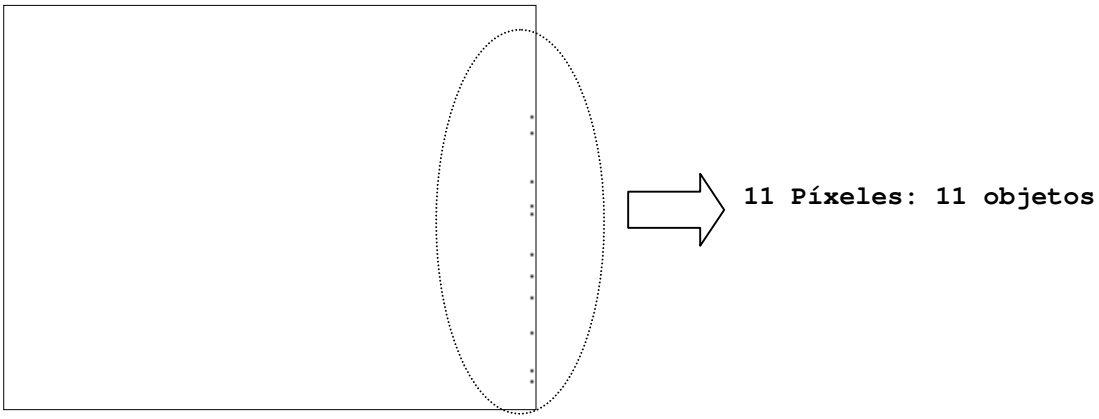


Figura 3.22 Imagenes de salida y resultado final

Para realizar la diferencia de imágenes grises en general se deben emplear las plantillas *Difference Threshold* y *Logic OR.* A partir de aquí se denominará el bloque *Difference x* al bloque que realiza la diferencia entre dos imágenes en grises, donde *x* es un parámetro que determina cuándo la diferencia entre dos píxeles es suficientemente significativa como para manifestarse en la imagen binaria de salida, que muestra estas diferencias con píxeles negros. La estructura de este bloque funcional multiplantilla, para un parámetro genérico *x,* es la que se muestra en la figura 3.23.

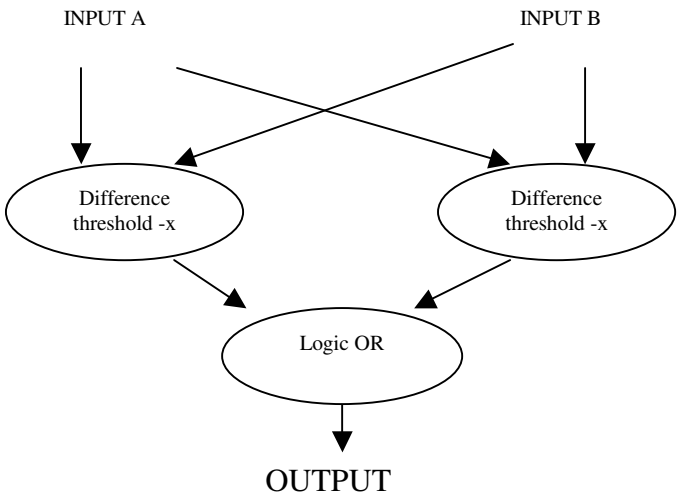


Figura 3.23 Estructura del bloque *Difference x*

Las plantillas requeridas en este algoritmo son las siguientes: *Smoothig Gray*, *Difference Threshold -0.25, Concave Location Filler, Local Southern Detector* y *Horizontal CCD*. En la figura 3.24 se muestra la estructura completa del algoritmo, con algunas imágenes parciales.

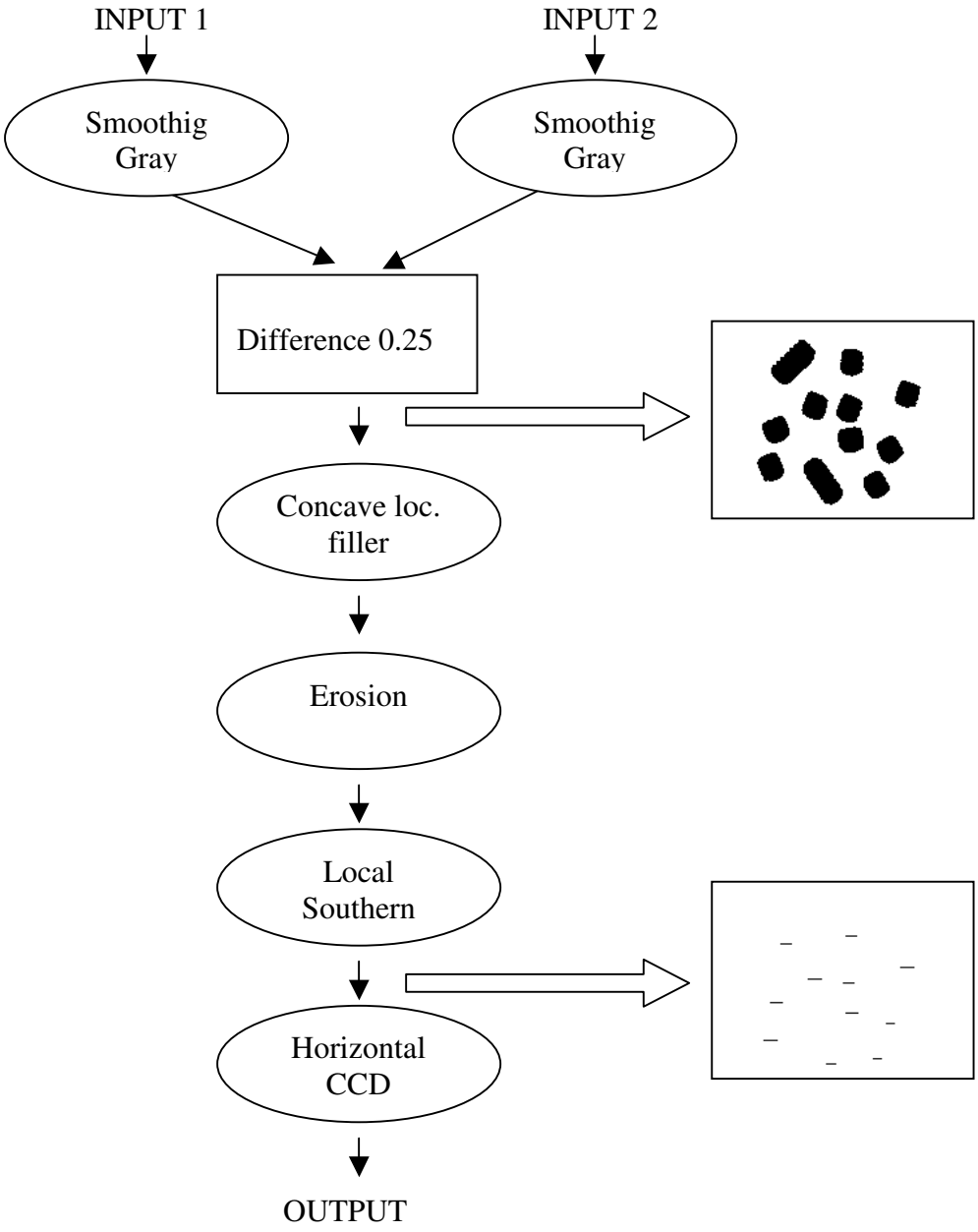


Figura 3.24 Estructura del algoritmo y algunas imágenes parciales

La estructura anterior es susceptible de modificación en función de la situación en la que se aplique el procesamiento. Si se requiere un mayor suavizado se podría haber empleado la plantilla *Heat Difusión* en lugar de la *Smoothing Gray*. En caso de que el fondo sea más heterogéneo puede ser recomendable la función *Dilation* para evitar que los objetos se fragmenten, dando lugar a un mayor número de objetos. En caso de ruido o imprecisiones significativos se debe aplicar varias plantillas *Erosion* en el lugar donde hay una sola plantilla *Erosion* en el esquema de la figura 3.24.

3.4.5 CLASIFICADOR DE VELOCIDAD HORIZONTAL

Este programa realiza una clasificación de los objetos de la imagen de entrada en función de su velocidad, a partir de dos imágenes consecutivas de la secuencia de movimiento. Se va a suponer una determinada dirección y sentido. Para cualquier otra dirección y sentido no hay más que rotar las plantillas usadas en este algoritmo. En este caso se va a analizar la situación en que el movimiento es horizontal y hacia la izquierda. En la figura 3.25 se muestran las imágenes de entrada del algoritmo. Estas corresponden al fondo supuesto estático, a los objetos en un instante y a los objetos en un instante posterior.

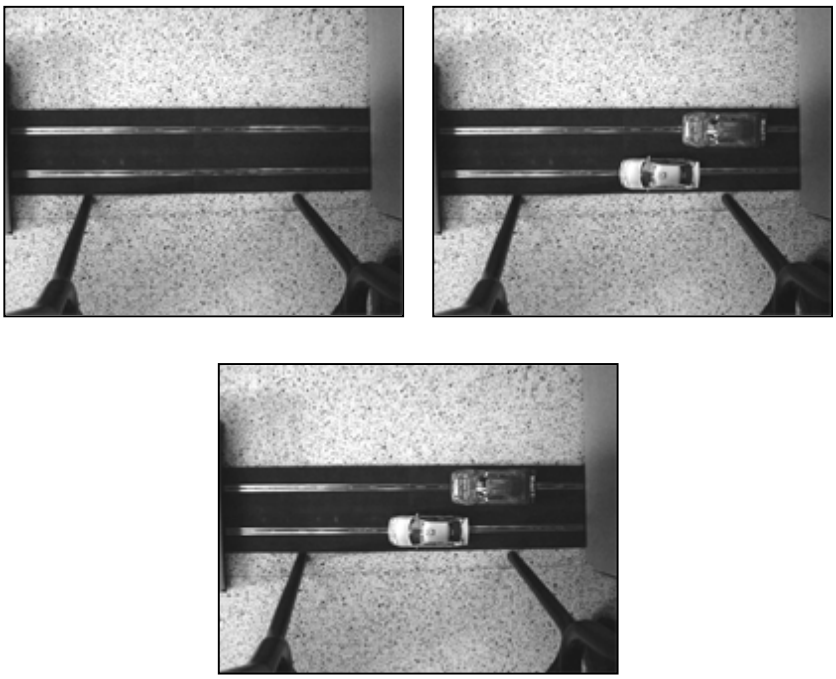


Figura 3.25 Imágenes de entrada del algoritmo

En la imagen de salida se muestran trazos horizontales, de largo proporcional a la velocidad detectada y de ancho el mismo que el del objeto en movimiento. En la figura 3.26 se muestra la imagen de salida que se obtiene de la aplicación de este algoritmo a las anteriores imágenes de entrada.

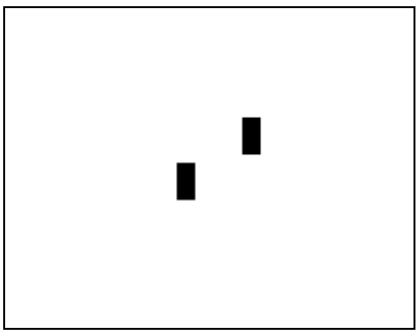


Figura 3.26 Imagen de salida del algoritmo

Las plantillas utilizadas son las siguientes: *Box Filler, Contour Detector Right, Logic Inverse, Masked Right Shadow, Difference Threshold* y *Logic Or*. Para la extracción de los objetos se realiza la diferencia de las imágenes con la imagen de fondo mediante la aplicación del bloque *Difference*, que ya ha sido comentado en el ejemplo anterior y muestra a la salida las diferencias significativas entre dos entradas. Para evitar que los objetos sean fragmentados se aplica el operador *Dilation*. No es necesario aplicar la plantilla *Erosion* para evitar falsos objetos debido a ruido e imprecisiones, puesto que esta planilla presenta una cierta inmunidad a estos. La estructura completa del programa se muestra en la figura 3.19.

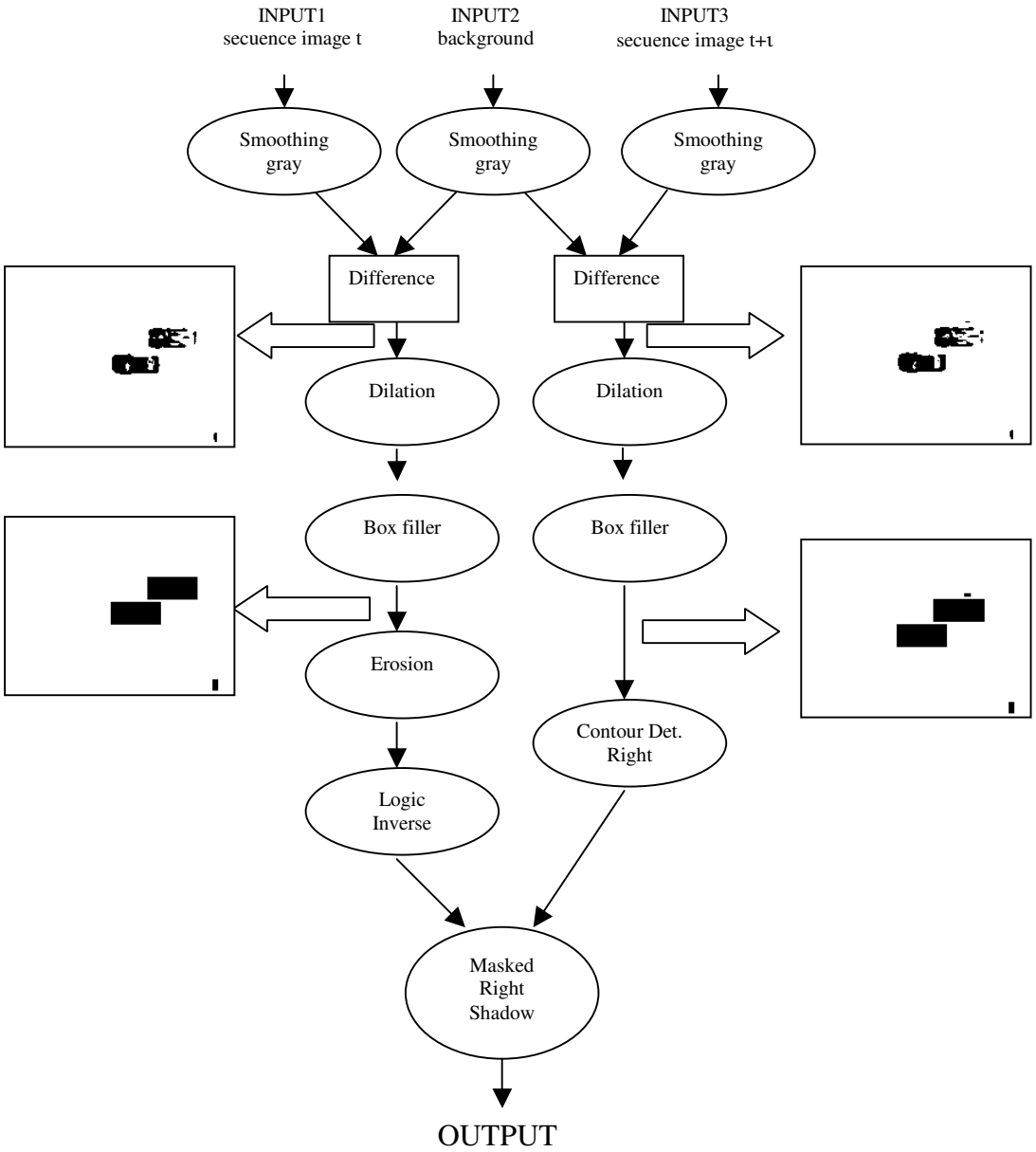


Figura 3.27 Estructura del programa y algunas imágenes parciales

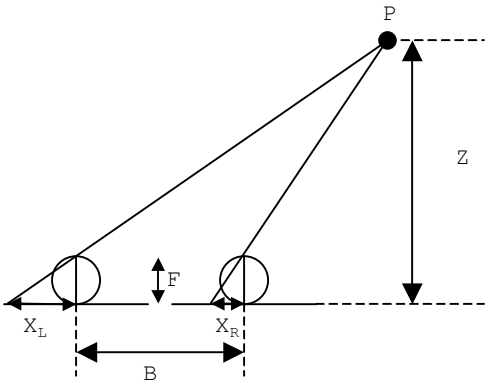
En las imágenes parciales podemos ver que aparecen dos falsos objetos, pero que no tienen influencia en la salida final. También se observa que los objetos que se extraen aparecen fragmentados en principio, pero el efecto reconstructor de la plantilla

Dilation y de la propia plantilla *Box Filler* los vuelve a agrupar en un solo objeto. Dependiendo de la situación es posible que sea necesaria la aplicación repetida de varias plantillas *Dilation* en lugar de una sola.

Este algoritmo tiene la restricción de que la velocidad de los objetos y el muestreo de las imágenes sean tales que el área de los objetos en movimiento presenten alguna intersección en dos imágenes consecutivas de la secuencia.

3.4.6 DETERMINACIÓN DE PROFUNDIDAD DE OBJETO MEDIANTE VISIÓN ESTÉREO

Mediante esta aplicación es posible realizar una estimación de profundidad de objeto, a partir de dos imágenes provenientes de dos puntos de vista separados, con fondo conocido en ambas. A la salida se muestra una imagen con marcaciones de una anchura proporcional a la disparidad encontrada en las dos imágenes para dicho objeto. Esta disparidad es inversamente proporcional a la profundidad, de manera que a mayor disparidad, anchura de las marcaciones a la salida, mayor profundidad corresponde al objeto. En la figura 3.28 se muestra la escena típica de la visión estéreo, así como la relación entre disparidad y profundidad. En esta escena se ha supuesto el caso más sencillo, con cámaras idénticas, ejes de cámara paralelos, línea base que une los centros de las lentes perpendicular a estos ejes, y referencias horizontal y vertical paralelas entre ambas cámaras.



$$disparidad = x_L - x_R = \frac{BF}{Z}$$

Figura 3.28 Geometría de visión estéreo

Se ha realizado una simplificación en el problema de correspondencia, de manera que se ha supuesto que en caso de haber varios objetos en la misma horizontal, estos deben estar lo suficientemente separados entre sí y lo suficientemente lejos de las cámaras tal que ,considerando la superposición de las imágenes que provienen de ambas cámaras, las proyecciones de un objeto aparezcan separadas de las proyecciones de cualquier otro objeto, y no se crucen. En las figura 3.29 y 3.30 se muestran dos ejemplos de correcta e incorrecta aplicación, respectivamente.

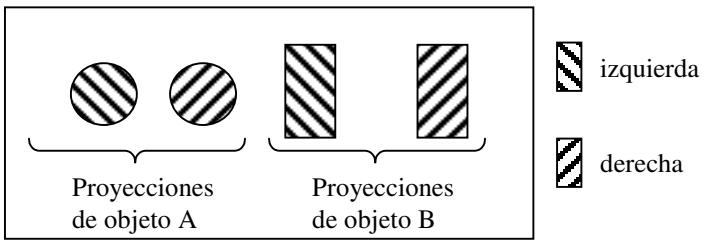


Figura 3.29 Superposición de imágenes izquierda y derecha en un ejemplo de correcta aplicación.

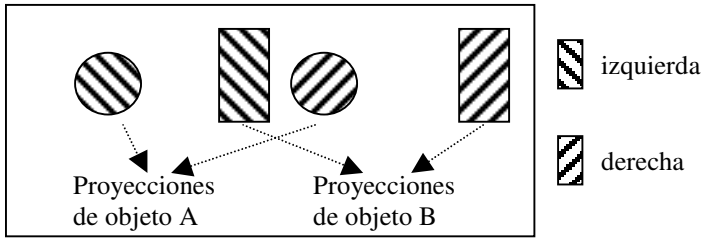


Figura 3.30 Superposición de imágenes izquierda y derecha en un ejemplo de incorrecta aplicación.

Las imágenes de entrada corresponden las tomas izquierda y derecha, con y sin los objetos, en total cuatro imágenes que se muestran en la figura 3.31.

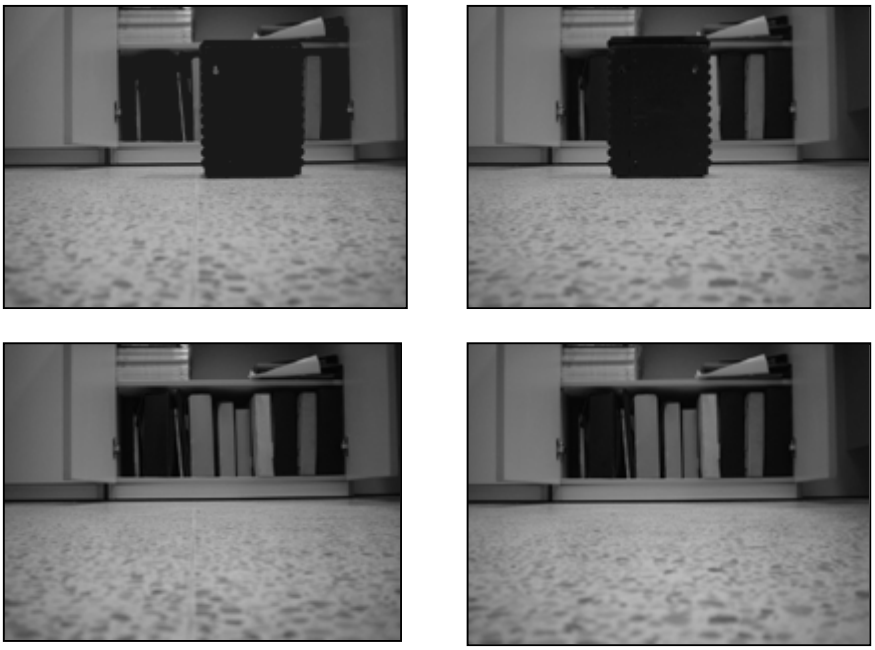


Figura 3.31 Imágenes de entrada

La imagen de salida del algoritmo muestra la disparidad del objeto de la entrada mediante un trazo horizontal rectangular de igual altura que el objeto y de ancho proporcional a la disparidad presentado por el objeto. En la figura 3.32 se muestra la imagen que se obtiene de la aplicación para las anteriores imágenes.

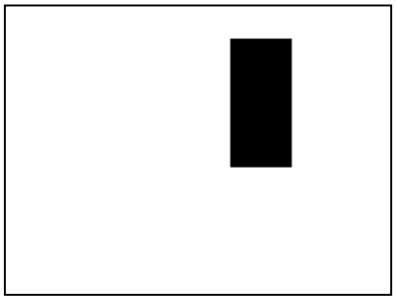


Figura 3.32 Imagen de salida del algoritmo

Las plantillas que entran en juego en este programa son las siguientes: *Box Filler, Right contour detector, Logic Inverse, Erosion, Dilation, Masked Right Shadow, Difference Threshold* y *Logic Or*. En el programa se utiliza el bloque *difference*, que es el mismo que se explico anteriormente y se muestra en la figura 3.23. La estructura del programa con algunas imágenes parciales se muestra en la figura 3.33.

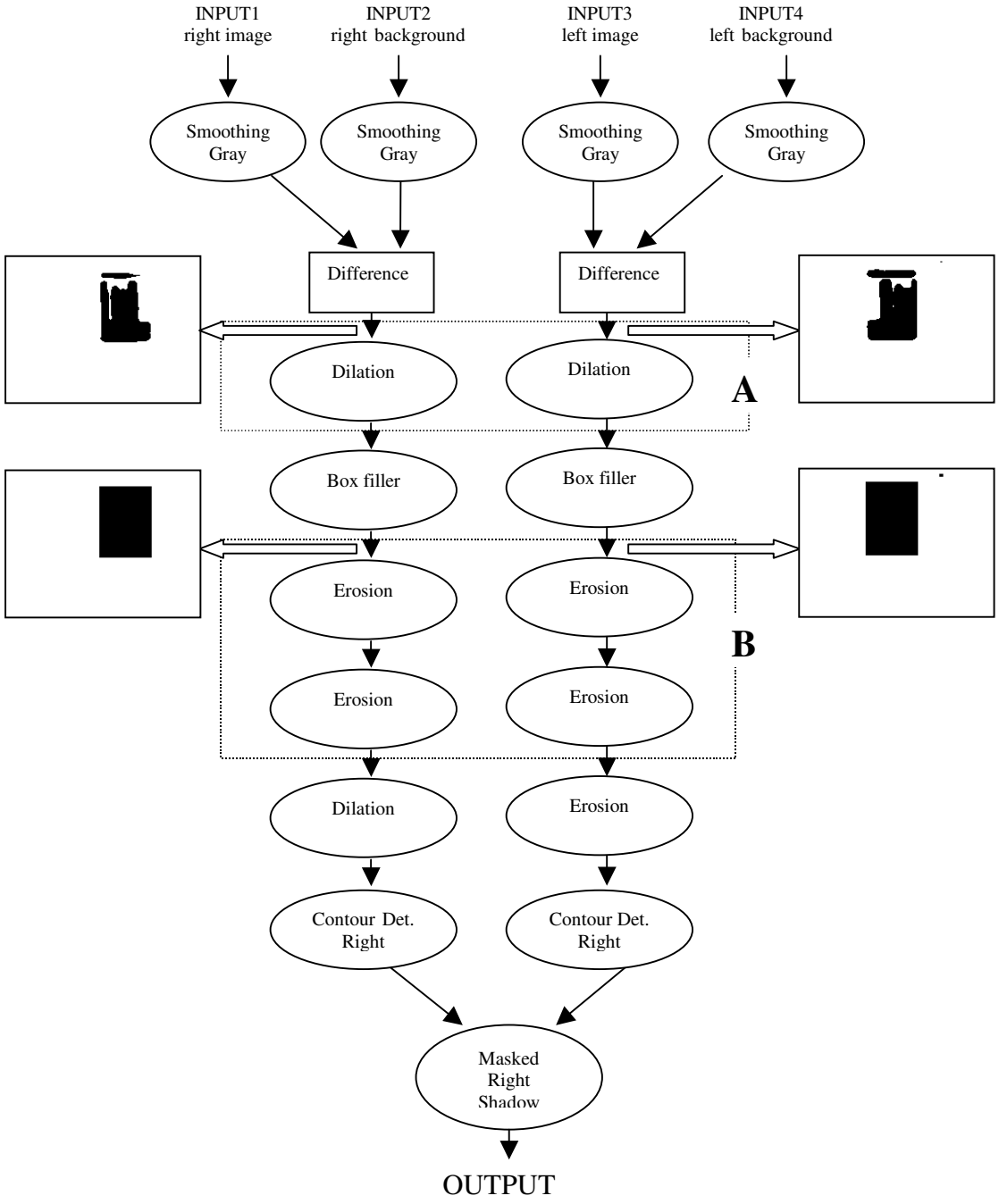


Figura 3.33 Estructura del algoritmo y algunas imágenes parciales

Se pueden realizar modificaciones para distintas situaciones, como aplicar un mayor filtrado de las imágenes de entrada y modificar el umbral utilizado en el bloque *difference* en imágenes ruidosas. El bloque marcado en línea discontinua con la etiqueta *A* señala la función *Dilaton* que se emplea para desframentar el objeto. Dependiendo del caso se pueden poner varias funciones *Dilation* de manera secuencial. En el segundo

bloque marcado en discontinua con la etiqueta *B* se refiere a la cadena de funciones *Erosion* que se utilizan para la eliminación de pequeños objetos resultados de ruido o imprecisión en la obtención de las imágenes. De nuevo se puede variar el número de funciones *Erosion* dependiendo de la situación.

3.4.7 DETECCIÓN DE TRAZOS AISLADOS EN UNA PCB

En este programa se realiza la detección de trozos de pista aislados en una PCB, considerando un trozo aislado como un segmento de pista no conectado con ninguna otra pista ni terminación. La suposición básica del programa es que toda pista estará conectada directamente o a través de alguna otra pista a una terminación, y que toda terminación presenta un agujero de taladro, lo cual posibilita que la CNN identifique fácilmente las terminaciones. La capacidad de detección de fallo del programa se limita por tanto a trozos completamente aislados de pista, no detectando discontinuidades simples que no aíslen trozos de pista. Un aspecto positivo es que presenta invarianza frente a la rotación. Este sencillo programa utiliza las plantillas *Threshold*, *Hole Filler* y C*onnectivity*. En la figura 3.34 se muestra la estructura del programa. La salida del mismo es una imagen que contiene los trazos aislados de la PCB. Se puede ver un ejemplo de aplicación en la figura 3.35.

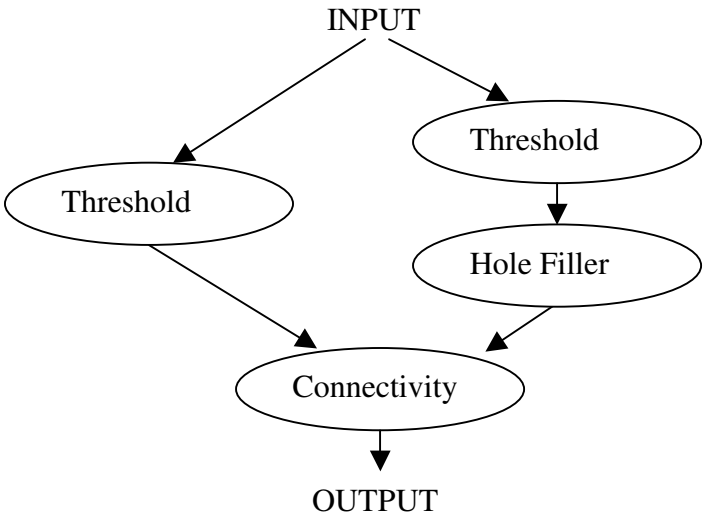


Figura 3.34 Estructura del programa

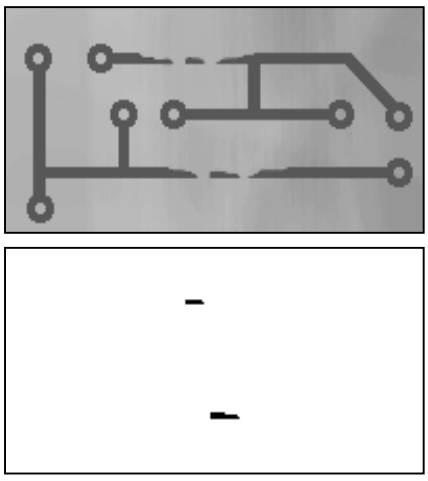


Figura 3.35 Imágenes de entrada y salida

3.5 SIMULACIÓN DEL PROCESAMIENTO DE UNA CNN POR COMPUTADOR.

Un bloque básico común a todos los programas del paquete software es el bloque de simulación de la CNN.

En la simulación se integra la ecuación diferencial que define el comportamiento en la celda i, j. Teniendo en cuenta que el modelo utilizado es el modelo original de Chua-Yang con entorno de vecindad 1, plantilla invariante e interacciones lineales:

$$C\frac{dV_{xij}(t)}{dt} = -R \cdot V_{xij}(t) + \sum_{l=j-1}^{j+1}\sum_{k=i-1}^{i+1} a_{kl}V_{ykl}(t) + \sum_{l=j-1}^{j+1}\sum_{k=i-1}^{i+1} b_{kl}V_{ukl} + I$$

Considerando normalización de parámetros electrónicos ($C=R=1$)tendremos:

$$\frac{dV_{xij}(t)}{dt} = -V_{xij}(t) + \sum_{l=j-1}^{j+1}\sum_{k=i-1}^{i+1} a_{kl}V_{ykl}(t) + \sum_{l=j-1}^{j+1}\sum_{k=i-1}^{i+1} b_{kl}V_{ukl} + I$$

3.5.1 INTEGRACIÓN DE LA ECUACIÓN DIFERENCIAL.

Para integrar la ecuación diferencial se utiliza el método de Runge-Kutta 4. Considerando que el estado de las células adyacentes como constantes, podemos escribir la ecuación de la dinámica como sigue:

$$\frac{dV_{xij}(t)}{dt} = -V_{xij}(t) + a_{ij}V_{yij}(t) + \sum_{\substack{l=j-1\\ \{k,l\}\neq\{i,j\}}}^{j+1}\sum_{k=i-1}^{i+1} a_{kl}V_{ykl}(t) + \sum_{l=j-1}^{j+1}\sum_{k=i-1}^{i+1} b_{kl}V_{ukl} + I$$

Denominando V_{xij} simplemente como *v,* con $V_{yij} = \frac{1}{2}(|v+1| - |v-1|)$, podemos escribir:

$$\frac{dv}{dt} = f(v)$$

Donde los 2 primeros términos de la ecuación completa se consideran variables y los 3 últimos se consideran términos fijos en cada paso de integración.

Dado un paso de integración h, el método de Runge-Kutta 4 estima el estado de la iteración k+1, v_{k+1}, a partir del estado anterior, v_k, y la ecuación de la dinámica $f(v)$ .Para ello aplica la siguiente ecuación:

$$v_{k+1} = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

Con:

- $k_1 = h \cdot f(v_k)$
- $k_2 = h \cdot f(v_k + {}^{k_1}\!/_2)$
- $k_3 = h \cdot f(v_k + {}^{k_2}\!/_2)$
- $k_1 = h \cdot f(v_k + K_3)$

3.5.2 PASO DE INTEGRACIÓN.

Un parámetro muy importante en la integración es la elección del paso de integración. Hay dos formas de plantear la definición del paso de integración, elegir un paso de integración fijo para toda la simulación o elegir un paso de integración variable, que vaya variando en cada iteración de acuerdo con alguna estrategia determinada.

- **Paso fijo.**

Un paso de integración demasiado pequeño da lugar a un integración precisa, pero demasiado lenta, mientras que un paso demasiado grande da lugar a una integración rápida y poco precisa. Hay que tener en cuenta que no estamos haciendo la integración de una sola ecuación diferencial, sino que se están integrando tantas ecuaciones como celdas, y estas celdas interactúan unas con otras, de manera que no se puede permitir una variación muy grande del estado de ninguna en un solo paso. En las CNN es normal que haya momentos en las que los estados varían muy bruscamente, y otros en los que los estados cambien más suavemente. Todo lo dicho anteriormente hacen que la elección del paso fijo no sea un opción acertada en esta aplicación.

- **Paso Variable.**

Lo más adecuado en la simulación de la CNN es elegir un paso que vaya variando dependiendo del comportamiento de la red en cada momento. Se define la máxima variación de estado de la red como la máxima variación de estado de célula en una iteración. El mecanismo elegido para regular el paso es el de procurar que la máxima variación de estado en una determinada iteración se mueva en un rango. Se definen dos parámetros, la *variación máxima deseada* y *la variación máxima límite*, y se actúa de la siguiente manera.

- Paso 0. Se fija un paso para la primera iteración suficientemente pequeño.
- Paso 1.Se simula la evolución de la red con un paso calculado en la iteración anterior.
- Paso 2.Al hacer el cálculo de la evolución del estado se calcula también la máxima variación de estado de la red.
- Paso 3.En caso de que la máxima variación de estado no supere la *variación máxima límite*, especificado por el usuario mediante la definición del parámetro *variation limit,* actualizaremos el estado de la red con el estado calculado. En caso no se actualizará el estado.
- Paso 4. Se calcula el nuevo paso, de manera que la máxima variación de la próxima iteración sea aproximadamente la *variación máxima deseada.* El paso se basa en suponer que el comportamiento de la red va a ser aproximadamente el mismo de una iteración a otra. Suponiendo también que la evolución de la máxima variación va a ser aproximadamente lineal:

$$\frac{h'}{VarDeseada} = \frac{h}{Var} \quad \Rightarrow \quad h' = VarDeseada \cdot \frac{h}{Var}$$

Donde h es el paso de integración actual, h' es el paso que se utilizará en la próxima iteración, Var es la máxima variación de estado en

la red y *VarDeseada* es la máxima variación que se desea tener en la próxima iteración y que el usuario especifica mediante el parámetro *variation predicted*.

- Paso 5. Mientras no se cumpla la condición de parada, se vuelve al paso 1 con el nuevo paso de integración calculado.

Para especificar la condición de parada se define la *variación de parada* a la que en la interfaz de los programas se referirá por el nombre *variation stop*. En realidad no está referida a una variación, sino una estimación de la pendiente de la máxima variación. Se calcula dividiendo la máxima variación por el paso de integración. En caso de que el valor resultante sea menor que el valor de *máxima variación de parada* definido, se considera que la red ha llegado a un estado lo suficientemente cercano al régimen permanente como para estimar que la red ha convergido.

- **Tiempo de simulación**

En cada paso de la integración el paso de integración está representando un tiempo. La suma de todos los pasos de integración de la simulación da lugar al tiempo total de simulación. Hay que tener en cuenta que este valor no es tiempo real, sino un valor normalizado, pues el modelo de la red utilizado está normalizado frente a las magnitudes de los elementos electrónicos de las células de la red neuronal. Al factor de escalado que sirve para parar de convertir el tiempo normalizado calculado en tiempo real en segundos se le denomina τ.

Un valor de τ en una implementación real podría ser 250 ns. Una plantilla de operación lógica, como el OR, tiene tiempos de simulación del orden de 5. El tiempo real de simulación sería el resultado del producto $5\cdot\ \tau$, que es 1.25 μs. A continuación se muestra algunos ejemplos de tiempos de simulación de algunas plantillas típicas, tanto normalizado como real, suponiendo un factor τ de valor 250 ns:

Plantilla	Tiempo normalizado de simulación	Tiempo Real de simulación (τ=250 ns)
Operaciones lógicas	5	1.25 μs
Threshold	5	1.25 μs
Erosion	10	2.5 μs
Concave Location Filler	50	12.5 μs

3.5.3 CONDICIÓN DE CONTORNO

La condición de contorno define qué hacer con los píxeles que quedan en el borde de la pantalla. Se definen tres condiciones de contorno:

- **Fija (Fixed):** Los píxeles del borde del estado permanecen estáticos e iguales al valor del estado inicial.
- **Toroide (Torus):** Los píxeles del borde evolucionan con la dinámica predicha en la ecuación, con la diferencia de que para los vecinos que queden fuera de la imagen se toma el píxel opuesto de la imagen. Es decir, si el vecino queda fuera de la imagen por arriba se toma el pixel extremo

por abajo, y viceversa; y si el vecino queda fuera por la derecha, se toma el pixel extremo por la izquierda, y viceversa.

- **Reflejado(Reflected):** Los píxeles del borde evolucionan con la dinámica predicha en la ecuación, con la diferencia de que para los vecinos que queden fuera de la imagen se toma el pixel vecino en la dirección opuesta. Es decir, si se sale por arriba se toma el pixel vecino por abajo, y viceversa; y si se sale por la derecha, se toma el pixel vecino por la izquierda, y viceversa.

4 OPTIMIZACIÓN BASADA EN TÉCNICAS GENÉTICAS POR COMPUTADOR.

4.1 INTRODUCCIÓN

Las técnicas de optimización genéticas pertenecen a un conjunto de técnicas englobados en el conjunto de algoritmos que se han denominado algoritmos evolutivos, y que realizan una búsqueda de soluciones que se basa en los mecanismos de evolución de los seres vivos. En particular el conjunto de técnicas genéticas aplica el uso de operadores genéticos en la búsqueda de nuevas soluciones. La potencia de dichos mecanismos se han mostrado muy eficientes en la naturaleza de manera evidente e indiscutible. Los primeros desarrolladores de dichas técnicas decidieron que era hora de dejar de envidiar a la naturaleza y empezar a imitarla en nuestro provecho. La ventaja de que disponemos al implementar la búsqueda en un computador es que, así como en la naturaleza las distintas generaciones pueden tardar en sucederse días, o años, dependiendo del tipo de ser vivo en cuestión, en nuestro caso la síntesis de nuevas generaciones es del orden de minutos en el peor de los casos; la búsqueda se realiza mucho más rápidamente.

Antes de nada se debe explicar con mas detalle como se realiza la búsqueda basada en técnicas genéticas. Primero se definen algunos conceptos necesarios.

- **Población**: Conjunto total de individuos con los que se trabaja en cada iteración.
- **Individuo**: Representación de una elemento del espacio de búsqueda.
- **Fitness**: Parámetro que indica la calidad de un individuo que servirá de criterio a la hora de realizar la selección.
- **Selección:** Procedimiento por el cual se escogen los individuos que se utilizarán para dar lugar a la nueva generación basándose en el valor de fitness de cada uno.
- **Codificación**: Forma de representar un elemento del espacio de búsqueda en un conjunto de bits y que da lugar al individuo.
- **Operadores genéticos:** Operadores de búsqueda utilizados en las técnicas genéticas cuyo fundamento es manipular los bits de los individuos de la poblacion original para generar la nueva población. En particular existen dos operadores genéticos distintos:
  - **Operador Cruce:** Combina los bits de dos individuos para explorar nuevas soluciones. Existen distintas formas de forma de combinar los bits que dan lugar a un tipo de búsqueda distinta.
  - **Operador Mutación**: Realiza pequeñas modificaciones en los individuos que dan lugar a distintas soluciones.
- **Evaluación:** Asignación de un valor de fitness a cada individuo, que indica la calidad del mismo y servirá de criterio al procedimiento de selección.

- **Elitismo:** Opción de busqueda en la cual el mejor individuo de una generación pasa directamente a la siguiente generación sin modificación alguna. Esto asegura la monotonía de la convergencia.
- **Cromosoma:** Conjunto de bits correspondientes a un individuo.

El funcionamiento básico de todas las técnicas genéticas, incluyendo las dos que se utilizan aquí, es el mismo. En primer lugar se realiza una evaluación de la población inicial. Posteriormente se van seleccionando por pares individuos de la población, y se aplican los operadores de cruce y mutación con una determinada probabilidad. Haciendo esto repetidamente se dará lugar a una nueva generación, y una vez que se haya completado se vuelve a evaluar y a repetir el procedimiento. En el siguiente diagrama se verá con mayor claridad:

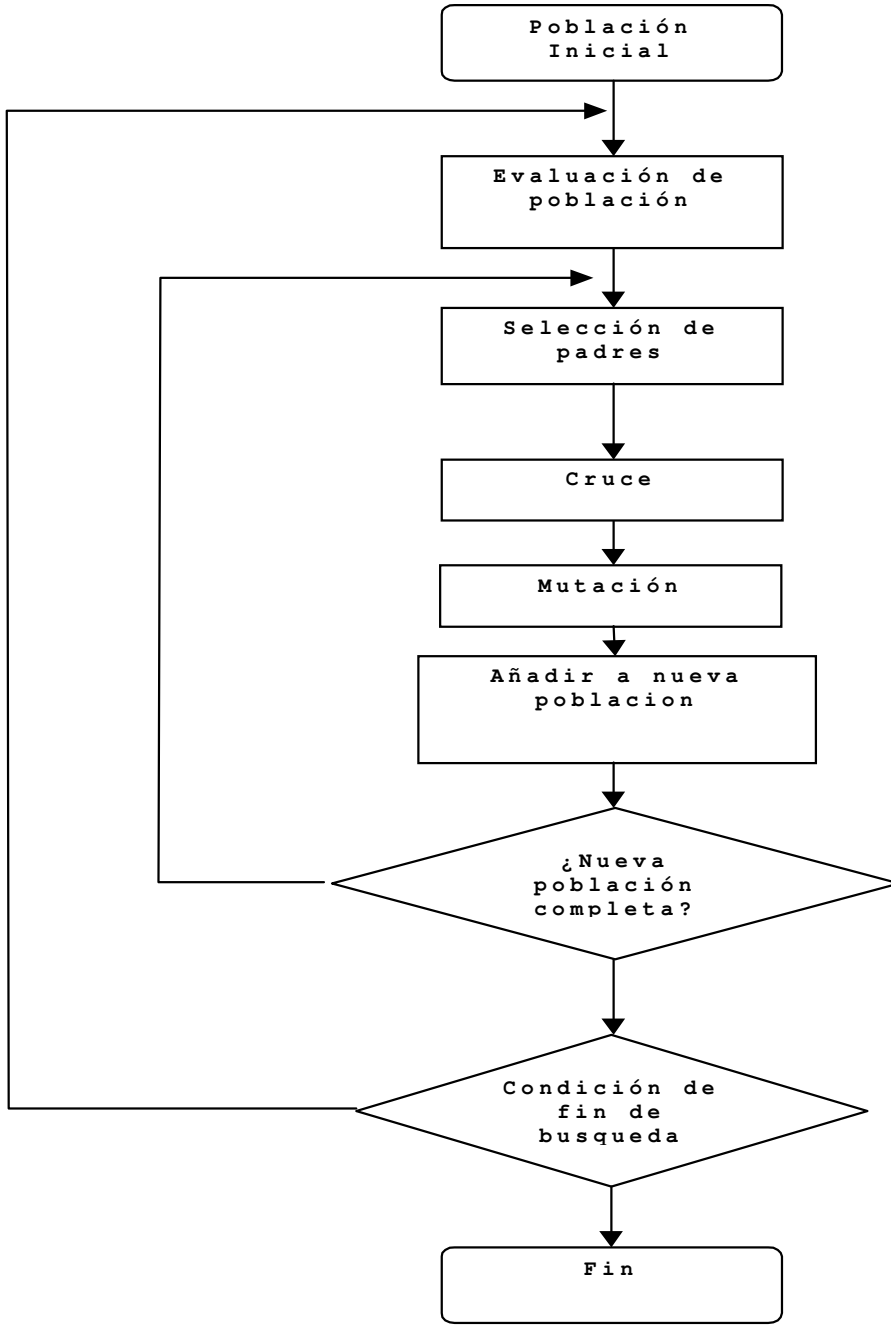


Figura 4.1 Diagrama genérico de una búsqueda genética

Cabe añadir un par de detalles que no se muestran en el esquema. La población inicial se genera aleatoriamente. El cruce y la mutación se aplican con una determinada probabilidad cada uno. La condición de parada podría ser un determinado valor de fitness del mejor individuo, o bien un determinado número de iteraciones. Por simplicidad no se ha indicado en el esquema cómo se realiza el elitismo, simplemente una vez que se haya completado la nueva población se añade el mejor individuo de la población.

Por último un par de comentarios acerca del fitness. Se trata de un parámetro fundamental en la búsqueda para cualquier técnica genética. Como ya se ha comentado es el parámetro que mide la calidad del individuo, y aporta un criterio a la hora de realizar la búsqueda. Aquel individuo que tenga un mayor valor de fitness tendrá más probabilidades para transmitir su carga genética a la siguiente generación de forma total o parcial, dependiendo de la acción de los operadores genéticos. Se entenderá por tanto que se debe diseñar de forma que aquellos individuos que aporten un resultado que se acerque más a lo que estamos buscando tendrán un fitness mejor. Dependiendo la naturaleza de lo que estemos buscando así estimaremos el parecido del resultado.

La elección del rango el valor óptimo del fitness es en principio arbitraria, siempre que se sea coherente. Se dice que un fitness es standard si se asocia el 0 con el mejor valor de fitness ,y se dice que está normalizado cuando permanece siempre entre 0 y 1.

Se debe procurar que la funcion de fitnes sea continua, es decir, que pequeñas mejoras en el resultado obtenido se traduzcan en pequeñas mejoras en el valor del fitness, y grandes mejoras en el resultado produzcan grandes mejoras en el valor del mismo.

Es posible añadir términos adicionales al fitness para que la optimización se realice en varios planos, para lo cual se añaden los valores de los términos que se deseen optimizar con una ponderación adecuada a la importancia relativa de estos. Si se desea que se optimice de forma secuencial debemos ponderar de forma que la aportación de cada término se diferencie en varios órdenes de magnitud. Si queremos que varios parámetros se optimicen simultáneamente se debe hacer una ponderación tal que la aportación de dichos parámetros al fitness total sea del mismo orden. También se puede realizar una ponderación no lineal. Por ejemplo se podría ponderar exponencialmente si resulta indiferente la magnitud de un parámetro en un determinado rango y se quiere penalizar fuertemente los valores de dicho parámetro fuera de ese rango con una cierta gradualidad.

En el presente proyecto se utilizan las dos principales técnicas de búsqueda genética que existen: Algoritmos genéticos y Programación genética.

4.2 ALGORITMOS GENÉTICOS (GA)

4.2.1 INTRODUCCIÓN

El interés en algoritmos de búsqueda heurística basados en los procesos naturales y físicos comenzó cuando [Holland 75] enunció los algoritmos genéticos. Las dos características principales de un GA es, en primer lugar, la representación de los individuos en cadenas binarias de tamaño fijo. Podemos decir que el algoritmo de búsqueda ignora el significado o la estructura de ese conjunto de bits, y lo único que hace es manipular las cadenas seleccionadas combinándolas y modificándolas sin tener

en cuenta el significado o la estructura de la cadena, con la única restricción de que la cadena resultante tenga el tamaño especificado. En realidad cualquier conjunto de bits con tamaño adecuado es una posible solución.

La segunda característica principal el uso en gran medida del cruce como operador de búsqueda. El operador de mutación ocurre con una probabilidad mucho menor y es secundario.

Otro aspecto importante es la naturaleza estocástica del operador de selección, permitiendo que algunos individuos sean elegidos en lugar de otros individuos con mejor fitness, lo cual garantiza la diversidad de la población y evita los óptimos locales. La presión selectiva será un parámetro importante en la selección, de manera que cuanto mayor sea esta, más restrictiva será la selección y mejores serán los individuos que den lugar a la siguiente generación, en promedio. Esto puede parecer bueno, pero puede dar lugar a convergencia prematura y mínimos locales, al perder la diversidad de la población por elegir siempre los mejores. Habrá que ajustar cuidadosamente la presión selectiva para que en cada generación se encamine la búsqueda utilizando buenos individuos sin que haya una pérdida significativa de diversidad en cada iteración.

4.2.2 CODIFICACIÓN:

La codificación consiste en representar los parámetros que se quieren optimizar en una cadena de bits. La codificación se puede realizar como se quiera. La única restricción que imponen los GA es que las cadenas siempre sean de tamaño fijo. Junto con la evaluación del individuo este es el punto dependiente del problema en cuestión. El resto de operadores y procedimientos son muy genéricos y se pueden aplicar a un amplio conjunto de problemas distintos.

Un aspecto fundamental a la hora de realizar la codificación es que el tamaño de la cadena da idea de las dimensiones del espacio de búsqueda. Siempre que sea posible se deberá codificar el mínimo número de parámetros con el mínimo número de bits posible por parámetro.

4.2.3 SELECCIÓN

La selección de los padres que darán lugar a la nueva generación se puede realizar mediante 4 operadores de selección distintos:

4.2.3.1 Ranking

(Parámetros: pmin,pmax)

Operador de selección en el que la probabilidad de que un cromosoma sea elegido es tanto mayor cuanto mayor sea su puesto en una clasificación de los individuos según su fitness. En concreto la probabilidad se calcula en función de dos valores introducidos por el usuario **pmax** y **pmin**, que definen las probabilidades que se asignarán al mejor y peor individuo respectivamente. El resto de valores se obtienen de una interpolación lineal entre estos dos valores en función del puesto del individuo en cuestión. Por ejemplo, un cromosoma en el puesto k de una población de N individuos con pmin, pmax tiene la siguiente probabilidad:

$$P(k) = p_{\min} + \frac{(p_{\min} - p_{\max})(k-1)}{N-1}$$

$$k=1,2,\ldots,N$$

Nota: Se entiende que estas probabilidades son relativas, no absolutas. La probabilidad absoluta de cada individuo sería la fracción del valor anterior por la suma de las probabilidades de todos los individuos de la población:

$$p(k) = \frac{P(k)}{\sum_{k=1}^{N} P(k)}$$

En este método de selección la presión selectiva se ajusta aumentando o disminuyendo la diferencia entre *pmin* y *pmax*.

4.2.3.2 Torneo

(Parámetros: pmin, pmax, N)

Operador de selección que forma un subconjunto de **N** cromosomas, resultante de aplicar el operador selección por Ranking N veces, y selecciona el mejor cromosoma de este subconjunto. Este operador ejerce un presión selectiva adicional a la que ofrece el operador Ranking.

4.2.3.3 Top Percent A y B

(Parámetros: Percent)

Operador que elige aleatoriamente un cromosoma de un subconjunto de la población formado por el porcentaje Percent % de cromosomas de mejor fitness. Hay dos formas de generar la nueva población, que dan lugar a los modos **A** y **B**. En el modo **A** a partir del porcentaje seleccionado se regenera completamente la población, mientras que en el modo **B** este porcentaje promociona automáticamente a la nueva población, y el resto de la nueva población se genera a partir de esta porcentaje.

Mención aparte merece la opción de **Elitismo,** que no es un operador de selección propiamente dicho, sino una opción en la creación de la nueva generación, que consiste en que el cromosoma de mejor fitness de la generación actual pasa a la siguiente generación directamente, sin ningún tipo de cruce o mutación. Esto asegura la monotonía de la evolución del fitness máximo de la población.

4.2.4 MUTACIÓN

La mutación se puede realizar mediante 4 operadores de mutación distintos, los cuales pueden actuar conjuntamente, cada uno de ellos con una probabilidad arbitraria no necesariamente de igual valor:

4.2.4.1 Flip Bit:

En caso de que proceda mutación invierte el bit que corresponda.(de ‘1’ a ‘0’ y viceversa). La probabilidad afecta bit a bit.

4.2.4.2 Gaussian:

(Parámetro: Gaussian noise)

Añade un ruido gaussiano a cada parámetro de la plantilla. Lo que se hace es generar una variable aleatoria gaussiana con desviación típica la indicada por el parámetro Guassian noise. El ruido se suma como valor real del parámetro, no como

valor codificado. Así mismo el parámetro al que se le suma el ruido es entendido como valor real, no como valor codificado.

Para la generación de la variable aleatoria gaussiana se utiliza la función de transformación de **Box-Muller** (1958) , que obtiene dos variables gaussianas , *y1* e *y2*, a partir de dos variables aleatorias uniformes entre 0 y 1, *x1* y *x2*, mediante la aplicación de:

$$y_1 = \sqrt{-2\ln(x_1)}\cos(2\pi x_2)$$
$$y_2 = \sqrt{-2\ln(x_1)}\sin(2\pi x_2)$$

$$x_1 = U(0,1)$$
$$x_2 = U(0,1)$$

Si sólo interesa generar una variable, como es el caso, se toma cualquiera de las dos variables y1 o y2.

En este tipo de mutación se entiende que la probabilidad que se defina afecta independientemente a cada byte del individuo.

4.2.4.3 Uniform

Actúa a nivel de byte. La acción de este tipo de mutación es sustituir el valor del byte en cuestión por otro elegido aleatoriamente entre todo el rango de forma uniforma, es decir, entre 0 y 255 con igual probabilidad para todos los valores. La probabilidad afecta byte a byte.

4.2.4.4 Boundary

Actúa a nivel de byte, y consiste en asignar un valor límite, superior (byte '255') o inferior (byte '0') elegido aleatoriamente. La diferencia con la mutación anterior es que ahora el valor asignado se asigna solo puede ser uno de los dos valores límite, mientras que antes podía ser cualquiera del rango. La probabilidad afecta byte a byte.

4.2.5 CRUCE

En el cruce de los cromosomas podemos encontrar 5 operadores distintos:

4.2.5.1 One Point

Operador que aleatoriamente selecciona un punto de cruce en la cadena de cromosoma a nivel de bit, y forma un nuevo cromosoma a partir de un trozo de cromosoma de cada padre, tomando cada trozo a un lado y otro del punto de cruce.

Como ejemplo considerar los dos siguientes padres en el que el símbolo '|' representa el punto de cruce:

Padre A: 00...11|11...01
Padre B: 01...00|01...01

Hijo: 00...11|01...01

4.2.5.2 Two Point

El comportamiento de esto operador es equivalente al operador anterior, pero en lugar de un punto se eligen dos puntos de cruce para la formación del nuevo hijo, a partir de los trozos de cadenas de bits determinados por estos puntos. La mejor forma de ver esto es con un ejemplo, donde de nuevo representamos el punto de cruce con el símbolo 'l':

Padre A: 00...01|10...01|00...11
Padre B: 11...11|00...00|11...11

Hijo: 11...11|10...01|11...11

4.2.5.3 Uniforme

El cromosoma hijo se obtiene de mezclar ambos cromosomas padre a nivel de bit, tomando alternativamente uno u otro cromosoma con una probabilidad determinada, conocida como razón de mezcla.

4.2.5.4 Aritmética

Parámetro: a.

Operador que actúa a nivel de bytes , en la cada byte de la nueva cadena del cromosoma hijo se obtiene de una media ponderada por un factor **a**, y que podemos expresar con la siguiente ecuación:

$$ByteHijo[i] = a \cdot BytePadreA[i] + (1-a) \cdot BytePadreB[i]$$

Nota: No es un operador genético propiamente dicho, pues no hace una combinación de cromosomas a nivel de bits, pero en algunos casos se a mostrado un mecanismo de búsqueda eficaz.

4.2.5.5 Heurístico

Operador que opera a nivel de bytes, y utiliza el valor del fitness de cada padre para determinar la dirección de la búsqueda. El valor de cada byte se obtiene de la siguiente forma:

$$ByteHijo[i] = ByteMejorPadre[i] + r \cdot (ByteMejorPadre[i] - BytePeorPadre[i])$$

La variabla **r** es una variable aleatoria, de manera que vectorialmente se puede interpretar este cruce como un alejamiento del peor cromosoma desde la posición del mejor cromosoma.

Notar que es posible que el cromosoma resultante se salga del espacio de cromosomas válidos, de manera que habría que repetir el cruce hasta que el cruce resultante sea un cruce válido. Se define un numero de intentos máximo de cruce, y en cada intento la variable aleatoria r es una uniforme cuyo límite inferior es 0, y el límite superior es el valor r obtenido en el anterior intento, 1 para el primer intento. Esto "casi" asegura la formación de una cadena válida en cierto número de intentos. De todas maneras se tiene en cuenta que en el número de intentos definidos no se obtenga un cromosoma válido en cuyo caso se toma directamente el mejor cromosoma como cromosoma hijo.

Nota: No es un operador genético propiamente dicho, pues no hace una combinación de cromosomas a nivel de bits, pero en algunos casos se a mostrado un mecanismo de búsqueda eficaz.

4.2.6 ESQUEMAS Y BLOQUES CONSTRUCTIVOS.

Dada una cierta representación del problema, los GA son capaces de combinar aquellas partes de una solución que son necesarias para formar un óptimo global.

Podemos definir un esquema como una cadena de elementos con tres valores posibles que representamos con ' 1' , ' 0' , ó ' *' , siendo el significado de éste último indiferencia en su valor. Este símbolo ' '* implica proyecciones en distintos subespacios del espacio de búsqueda completo. De hecho los GAs funcionan tomando muestras de estos subespacios. Veamos por qué los esquemas son importantes. Consideremos un esquema con k posiciones fijas. Habrá otros 2^K-1 esquemas con las mismas posiciones fijas que pueden obtenerse considerando todas las permutaciones posibles de ceros y unos en esas k posiciones. En general para k posiciones fijas hay 2^K esquemas distintos que generan una partición de todas esas posibles cadenas. Cada uno de esos conjuntos con k posiciones fijas genera una competición de esquemas, o sea, una competición de supervivencia entre los 2^K esquemas. Puesto que hay 2^l combinaciones posibles de posiciones fijas, habrá 2^{1} competiciones de esquemas distintas. La ejecución del GA genera por lo tanto 2^l competiciones de esquemas simultaneas que el GA intentará resolver, así como localizar el mejor esquema para cada conjunto de posiciones fijas.

Podemos imaginarnos la búsqueda de la cadena óptima por parte del GA como una competición simultánea entre esquemas que incremente el número de sus ejemplos en la población. Si se describe la cadena óptima como la yuxtaposición de esquemas con longitudes de definición cortas y elevados niveles de salud, entonces los ganadores de esas competiciones de esquemas individuales podrían de forma potencial formar la cadena óptima. A esos esquemas con elevados valores de salud medios y reducidas longitudes de definición de les denomina bloques constructivos. La idea de que cadenas con un alto valor de salud pueden ser localizadas tomando muestras de bloques constructivos con alto grado de salud y combinándolos se denomina hipótesis de bloques constructivos, enunciada, al igual que los propios GA, por Holland.

4.3 PROGRAMACIÓN GENÉTICA (GP)

4.3.1 INTRODUCCIÓN

La principal característica de esta técnica es que ahora los individuos son cadenas de tamaño variable. Ahora la cadena de bits correspondientes a cada individuo tiene una estructura que representa un determinado algoritmo, y esto se tiene en cuenta a la hora de aplicar los operadores de cruce y mutación, que se deben adaptar a la estructura de lo que están representando las cadenas de bits. Hay distintas opciones a la hora de elegir una estructura que represente el problema. En lo que sigue se asumirá que la estructura a la que nos referimos es una estructura de árbol, que es la más utilizada y la que en particular se ha escogido en este caso.

El cruce sigue siendo el principal mecanismo de búsqueda en GP**.** [Koza 92], uno de los principales desarrolladores de la GP, volvió a hechar mano de la hipótesis de los bloques constructivos, básicamente equivalente a la hipótesis de Holland para los GAs. En nuestro caso, y para una representación en árbol, un bloque constructivo sería un subárbol, de manera que los individuos que lo contengan ven mejorado su fitness, y así, hay una tendencia a preservar y acumular bloques constructivos en su población, y

formarán cada vez mejores individuos. El uso del cruce hace que el GP progrese más rápidamente que un sistema basado simplemente en la mutación.

Este argumento ha sido tema de debate, y se ha llegado a afirmar que el operador cruce en GP no es más que un operador de macromutación [Banzhaf 98]. De todas maneras, tanto si es un operador que tiende a preservar los bloques constructivos, como si se trata de una macromutación, el operador de cruce se ha mostrado como un mecanismo de busqueda muy potente en múltiples y diversas aplicaciones.

4.3.2 CODIFICACIÓN DEL ÁRBOL

La estructura mas utilizada en muchos y muy diversos tipos de algoritmos es el árbol. Cada uno de los árboles que entraran a formar parte del algoritmo de programación evolutiva será representado en forma de expresión por medio de la notación prefix o prefijo, debida a Lukasiewicz.

Hay dos tipos de nodos en un árbol:

- **Terminales**: son las hojas o nodos terminales del árbol. En nuestro caso son imágenes. Cada terminal será una de las imágenes del juego de imágenes base definido, imágenes que manipulará para intentar aproximarse a un determinado resultado.

- **Funciones**: son los nodos intermedios del árbol. Habrá de dos tipos, binarios y unarios. Los binarios toman dos imágenes de entrada, mientras que los unarios solo toman una. Dan una imagen de salida como resultado.

Las expresiones conforman un conjunto de terminales y de funciones relacionadas de forma coherente. En este apartado vamos a presentar el modo en que se ha utilizado la notación de Lukasiewicz para representar el espacio de algoritmos en el cual vamos a utilizar un algoritmo de optimización evolutiva para encontrar el programa que mejor se adapte al mapeado deseado, y que tipo de propiedades deben cumplir las expresiones para que se la considere coherente.

La conveniencia de esta notación surge en el paso de generación aleatoria de árboles binarios (primer paso de los algoritmos de programación evolutiva) y en las etapas de cruce y mutación genéticas. Los árboles resultantes de estas operaciones deben ser expresiones sintácticamente validas (coherentes), es decir, deben representar algoritmos ejecutables.

Gracias a esta notación, y a un teorema desarrollado por *Rosenbloom*, podemos forzar a que los árboles que entran a formar parte de nuestro algoritmo sean coherentes.

> ***Teorema*** *. Una secuencia S de símbolos y operadores en notación prefijo es una expresión bien formada (coherente) si y solo si:*
>
> ***a.*** *Rango(S) = -1;*
>
> ***b.*** *Rango(cualquier subexpresion a la izquierda de S) ≥ 0;*
>
> *donde la función Rango viene definida por:*
>
> *Rango(operador binario) = 1;*
> *Rango(operador unario) = 0;*
> *Rango(constante) = -1;*
> *Rango(S1 concatenada con S2) = Rank(S1) + Rank(S2).*

Este teorema nos permite reconocer cuando una cadena es o no coherente. El concepto de cadena coherente es importante en el proceso de crecimiento de cadenas

que se producen en el algoritmo evolutivo, así como en el cruce y mutación de los individuos.

Dados un juego de terminales y un juego de funciones definido hay principalmente dos técnicas diferentes para la inicialización de estructuras de árbol, ***full*** y ***grow***.

El método *grow* elige para cado nodo un elemento aleatoriamente del juego de funciones o del juego de terminales, salvo el nodo raíz que siempre es una función, y los nodos de profundidad la máxima permitida, que siempre son terminales. Esto provoca un aspecto irregular del árbol, con ramas de distintas profundidades.

El método *full* por el contrario siempre elige un elemento del juego de funciones, salvo los nodos de profundidad máxima que se elegirían del juego de terminales. Esto provoca que todas las ramas del árbol tengan la profundidad máxima, y tenga un aspecto mucho más regular.

En caso de que la medida de tamaño sea el número de nodos, el crecimiento parará cuando el árbol haya alcanzado el valor del parámetro predefinido.

Debido a que se usa el mismo método para la generación de todos los individuos es posible que se obtenga una población demasiado homogenea, haciendo más dificil la evolución. Para prevenir esto surge la técnica ***ramped-half-and-half***, que consigue una mayor diversidad en la población. Se expone el método mediante un ejemplo. Suponer que se define la profundidad máxima como 6. Se divide la población en 5 grupos de profundidades 2,3,4,5,y 6. Para cada grupo, la mitad de la población se genera mediante el método full, y la otra mitad mediante el método grow.

4.3.3 SELECCIÓN

La selección se realiza de forma equivalente a como se veía en GA, de hecho los operadores son análogos, pero hay un par de ellos no coinciden exactamente. En el programa se han implementado los siguientes métodos de selección:

4.3.3.1 Método nu,lambda.

Parámetros: nu (población),lambda(población extra)

Es equivalente al método topPercent-A de GA.. Un número nu de padres de lugar a un número lambda de hijos, de los cuales se eligen a los nu mejores que serán los padres de la nueva generación

4.3.3.2 Método un+lambda.

Parámetros: nu (población),lambda(población extra)

Es equivalente al método topPercent-B de GA. La diferencia con el método *nu, lambda* es que la elección de los un padres de la próxima generación se hace teniendo en cuenta a los un padres de la nueva generación.

4.3.3.3 Metodo Ranking.

Parámetros:pmax,pmin.

Exactamente igual que el método de GA, la asignación de la probabilidad se asocia según la posición que ocupe en la población total ordenado según el fitness.

4.3.3.4 Método Torneo.

Parámetros:pmax, pmin, N.

Exactamente igual que el método de GA. La selección es fruto de aplicar N veces el operador de selección por Ranking, tomando el mejor de ellos como padre.

4.3.4 OPERADOR DE CRUCE

Se debe tener muy presente la estructura de los que estamos manejando, no hacer una simple combinación de bits que podría provocar inconsistencia en la estructura representada por el conjunto de bits resultante.

Se ha implementado un solo operador de cruce en **GP.** Su funcionamiento es el siguiente. A partir de los dos padres seleccionados y, si procede cruzarlos, se toma uno de ellos y se selecciona uno de los nodos aleatoriamente. En el otro árbol se actúa de igual manera y, una vez que se tienen los dos nodos, se toman los subárboles correspondiente a ambos nodos y se intercambia uno por otro, incluyendo los nodos raíz de ambos subárboles. Se observará de forma más clara en el siguiente ejemplo:

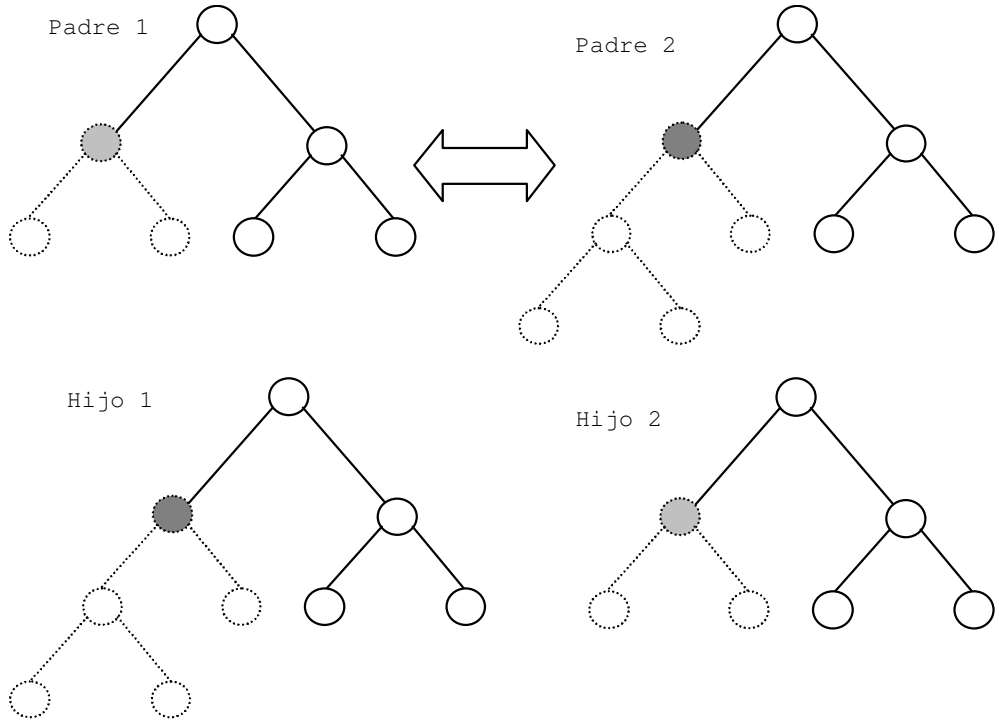


Figura 4.2 Cruce de árbol

4.3.5 MUTACIÓN

Al igual que el operador de cruce, está adaptado a la estructura representada en el cromosoma. Como veremos hay varios tipos de operadores de mutación, y todos ellos introducen pequeñas modificaciones de manera que el cromosoma resultante siga siendo coherente con la estructura de un árbol. Para GP con estructuras basadas en árbol tenemos los siguientes operadores de mutación:

4.3.5.1 Mutación subárbol

Este operador selecciona aleatoriamente un nodo del árbol y hace crecer un nuevo subárbol que substituye al que correspondía al nodo seleccionado. Ver figura 4.3.

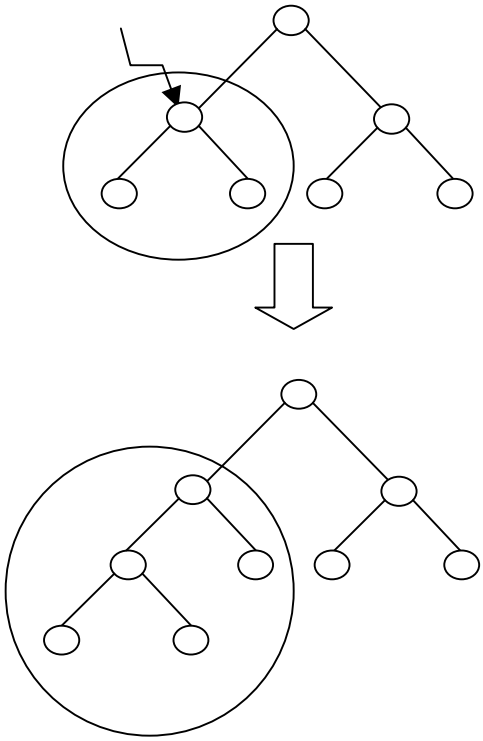


Figura 4.3 Mutación subárbol

4.3.5.2 Mutación Nodo

Selecciona un nodo del arbol y lo sustituye por otro nodo del mismo tipo (terminal o función), elegido aleatoriamente de entre los posibles. Ver figura 4.4.

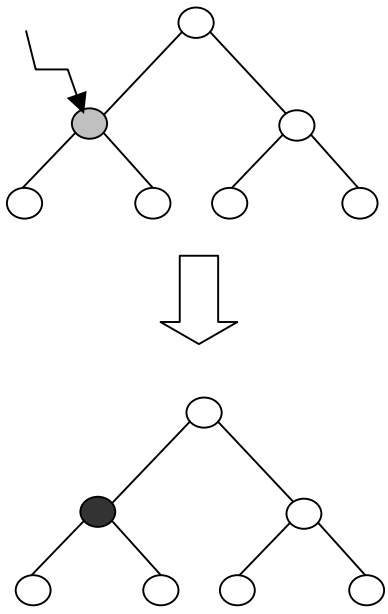
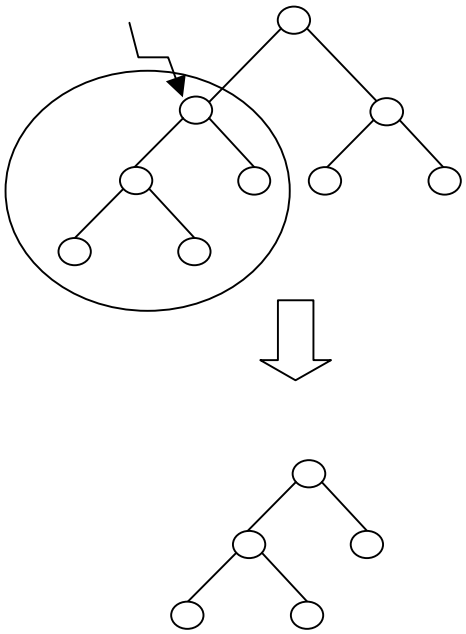


Figura 4.4 .Mutación nodo

4.3.5.3 Mutación Hoist

Se elige aleatoriamente uno de los nodos del árbol, y el subárbol que tiene como nodo raíz a este nodo substituye al árbol completo. . Ver figura 4.5.



4.5 Mutación Hoist

4.3.5.4 Mutación Collapse

Una vez elegido aleatoriamente uno de los nodos del árbol se substituye el subárbol correspondiente por un nodo terminal. Ver figura 4.6.

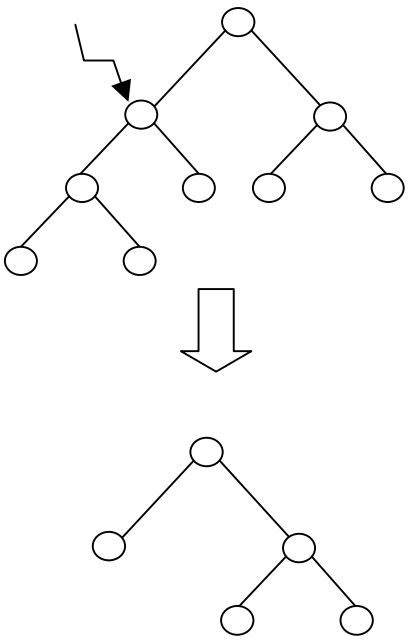


Figura 4.6 Mutación Colapso

4.3.6 INTRONS.

Una característica del GP que se observó desde los primeros ejemplos desarrollados por *Koza* es al aparición de bloques que no tienen efecto sobre el resultado final. Por analogía con la genética biológica se han denominado *introns* que son bloques en la cadena genética que no tienen ninguna aportación al individuo y aparecen de forma natural en las cadenas genéticas de un ser vivo de cualquier especie.

Tienen un efecto positivo y un efecto negativo. El positivo es de protección de los bloques constructivos, pues actúan como colchón frente a los operadores genéticos, que a menudo son bastante destructivos. Pero, por otra parte, tienen un efecto negativo, y consiste en que provocan una tendencia en la población a "hincharse" y, tras un cierto número de iteraciones, la población tiende a tener el tamaño máximo permitido, y esto resulta negativo para la evolución del algoritmo.

Es fácil entender a qué se debe el crecimiento exponencial de los bloques intron en la población. El fitness implica la probabilidad de supervivencia de la carga genética de un individuo, o parte de ella, en una iteración. Pero esto no implica que a lo largo de un cierto número de iteraciones su carga genética tenga más probabilidad que otras, porque hay que tener en cuenta que habrá ciertas estructuras más robustas que otras frente al efecto destructivo de los operadores genéticos, como son las estructuras que contienen bloques "introns". De manera que podemos hablar de un fitness efectivo, en el que se tiene en cuenta este efecto, que será función del fitness absoluto y de la robusted de la estructura, siendo este segundo término mayor en estructuras que contengan bloques "introns".

Una forma de evitar el crecimiento excesivo de la longitud de los individuos es aplicar presión de "parsimony", que consiste en incluir un término en el fitness de manera que se penalice la longitud de los individuos.

5 APLICACIÓN TÉCNICAS GENÉTICAS PARA EL DISEÑO DE SISTEMAS BASADOS EN CNN.

5.1 INTRODUCCIÓN

Las dos técnicas genéticas anteriormente comentadas han sido expuestas desde un punto de vista muy genérico, y tal como han sido presentadas podrían utilizarse para la resolución de una gran variedad de problemas distintos, adaptando cada técnica a las peculiaridades de cada problema, que afectará principalmente a la codificación de los individuos así como a la evaluación de los mismos.

En el presente apartado se trata de comentar estos aspectos específicos para el problema que se trata, el diseño automático de plantillas CNN mediante la técnica GA, y el diseño automático de algoritmos CNN mediante la técnica GP.

5.2 FITNESS

Tanto en el GP como en el GA se ha tomado como criterio de fitness un fitness estandar, es decir, un fitness cuyo óptimo se encuentra en el valor 0. En GP está además normalizado, siempre se encuentra entre 0 y uno. En el GA no está completamente normalizado, puede dar valores algo mayores que 1 debido a términos añadidos, pero normalmente no mucho mayores que 1.

Tanto en el GP como en el GA el cálculo del fitness se basa principalmente en el cálculo de parecido entre la imagen de salida y la imagen deseada. A este término principal se le pueden añadir otros que ponderen algún parámetro que convenga optimizar.

A continuación se comente las particularidades este término principal, y posteriormente trataremos los términos añadidos para cada caso. Dada la simílitud en la evaluación de este término se comenta de forma conjunta para GA y para GP.

5.2.1 TERMINO PRINCIPAL. PARECIDO DE IMÁGENES.

Para evaluar la diferencia de imágenes se tienen varias opciones, alguno no coincidente en GA y GB. A continuación las veremos una a una indicando en cada caso si se aplican en el GA, en el GP o en ambas.

5.2.1.1 Diferencia absoluta de imágenes(GA y GP)

Es un término indicado cuando las salidas sean imágenes binarias. Se recorre la imagen pixel a pixel, y se calcula para cada pixel el valor absoluto de la diferencia entre los valores de los píxeles de las imágenes que se comparan. La normalización de la suma de todos estos términos da como resultado el término principal. Es evidente que el mejor valor sería el cero, que se dará cuando ambas imágenes sean idénticas, y nunca es mayor que 1 al estar normalizado.

Los bordes de la imagen no se tienen en cuenta en esta evaluación.

5.2.1.2 Correlación. Aspectos comunes GP,GA.

Aunque este método se implemente básicamente igual en GA y GP hay alguna diferencia que se comentará después. Ahora se enuncian los aspectos comunes.

Se recorre la imagen pixel a pixel y se calcula el producto de los valores en cada pixel. La suma de estos valores se divide por el número de píxeles y dará la correlación

por definición. Para normalizar este valor se divide por el producto de los modulos de ambas imágenes, entendiendo como módulo el valor cuadrático medio de la imagen. El valor resultante del punto anterior no está completamente normalizado, puesto que se encuentra comprendido entre –1 y 1. Los valores positivos en el valor del punto anterior indican parecido de imagen, mientras que valores cercanos a cero indican imágenes poco parecidas. Los valores negativos indican que la imagen se parece inversamente, es decir que el inverso de una imagen es parecido a la otra imagen. La diferencia entre GP y GA está en el tratamiento de los valores negativos.

5.2.1.3 Correlación GP

No debemos descartar individuos con valores negativos en el parámetro anterior, pues estarán cerca del óptimo en número de funciones. Lo que se hace es el valor absoluto de este valor. Una vez que se tiene el parámetro normalizado entre 0 y 1 se invierte con la función f(x)=1-x, puesto que interesa un fitness standard para ser coherente, que el valor 0 corresponda al mejor y el 1 al peor.

Por último se penaliza ligeramente si ha resultado un parecido inverso, pues no interesa que se impongan a otros individuos que den parecido directo. La penalización depende de que haya un término añadido de penalización de número de funciones del árbol (ver más adelante). Será un valor simbólico muy pequeño en caso de que no se pondere la longitud, y el equivalente de añadir una función más en el árbol multiplicado por 1.1 en el caso de que sí la haya. La razón de elegir precisamente este valor es por que la solución inversión está alejada en una función de la solución directa, y se le asocia un valor algo mayor que la penalización de una función para premiar la solución directa.

5.2.1.4 Correlación GA.

Se saturan los valores negativos a cero se invierte (aplicar $f(x)=1-x$) para que quede un fitness standard.

En GA no pondera de ninguna forma el parecido inverso, pues no tendría sentido hacerlo.

5.2.1.5 Diferencia especial (GP)

Es una optimización que mejora el funcionamiento del GP para imágenes binarias. Se recomienda utilizar especialmente cuando las figuras de la imagen deseada sean pequeñas, y solo se debe aplicar cuando en negro sobre fondo blanco. Suele ocurrir en figuras pequeñas que algoritmos que dan lugar a una salida en la que la figura resultante con unos pocos píxeles coincidentes, pero que nada tiene que ver con la figura en forma, dan un buen resultado de fitness.

Para realizar la medida del parecido se interpreta la imagen deseada de la siguiente manera. Se tienen dos conjuntos de píxeles, los píxeles donde interesa tener píxeles blancos, que se marcarán en la imagen deseada con un píxel blanco extremo(255), y los píxeles donde interesa tener valor negro, que se marcarán en la imagen deseada con un valor distinto del 255 y actuará como ponderador, dando más importancia en los sitios marcados con un valor más próximo al negro extremo (valor 0), y menos cuanto mas alejado este de este valor (hasta el 254).

A la hora de medir la calcular el término resultante se actúa de la siguiente manera. Se llevan dos cuentas, una para el conjunto de píxeles marcados con el valor blanco extremo, y otra para el otro conjunto de píxeles. Para el primer conjunto de píxeles se calculan dos valores, el número de píxeles del conjunto y el sumatorio del

valor absoluto de la diferencia entre el valor del píxel de la salida y el valor blanco (255). Para el segundo conjunto de píxeles se vuelven a calcular dos valores, el número de píxeles del conjunto y el sumatorio del valor absoluto de la diferencia entre el valor del píxel de la salida y el valor negro (0), estando ponderado término de este sumatorio por el valor del píxel de la imagen deseada, de la forma anteriormente comentada.

Una vez que se han calculado estos cuatro valores se procede de la siguiente forma. Para cada conjunto de píxeles se divide el valor obtenido en el sumatorio por el número de píxeles en cada caso, se suman los 2 valores obtenidos en cada caso, y por último se normalizan dividiendo por 255 y por 2. De esta forma se evitan problemas en imágenes con figuras pequeñas sobre fondo blanco, pues los píxeles blancos intervienen en una cuenta separada y tienen en total el mismo peso que los píxeles negros indicativos de las figuras, aunque el número de píxeles de uno y otro conjunto sea muy dispar.

Una característica muy importante de esta forma de calcular el término principal es la menor rigidez en la especificación de la figura deseada, de manera que en lugar de especificar un conjunto concreto de píxeles con la misma importancia, ahora se define de una forma más "borrosa", por así decirlo, la figura que se desea conseguir a la salida, ponderando más aquellos píxeles más oscuros, y menos aquellos más claros.

Resumiendo, en lo que al usuario le interesa, la imagen deseada debe indicar con valor 255 donde se quiere que haya píxeles blancos, y con valor de 0 a 254 donde se quiere que haya píxeles negros, dando más énfasis con el valor 0 y menos con el valor 254.

Es muy importante tener cuidado de que los valores donde se quiere un blanco sea 255 y no 254, pues con el primer valor se indica que se quiere un blanco, y con el segundo se indica que se quiere un negro, pero con poco énfasis.

5.2.1.6 Penalización todo-blanco todo-negro (GP,GA)

En la generación de la población, sobre todo en GA, hay muchas posibilidades de que la imagen de salida sea todo-blanco o todo-negro. En muchos casos la imagen deseada tiene la mayoría de los píxeles blancos, con lo cual resulta que estos individuos son evaluados con un buen fitness, cuando en la mayoría de los casos estarán muy alejados del fitness óptimo. Para evitar esto se inspecciona la imagen de salida del individuo, y en caso de que se detecte que se trate de una imagen todo blanco o todo negro se asigna el valor de fitness 1 a dicho individuo.

5.2.2 TERMINOS AÑADIDOS.

5.2.2.1 Terminos añadidos GA.

Se han definido dos parámetros que se ponderan linealmente y se suman al fitness total.

5.2.2.1.1 Tiempo total de simulación.

Se trata del valor de tiempo de convergencia de la simulación plantilla en red neuronal. Se debe hacer tal que el término que se añada al fitness sea despreciable o bastante menor que el término principal. De no ser así plantillas muy rápidas con resultados arbitrarios podrían dominar en seguida la población y nunca encontraríamos el óptimo.

El término se obtiene de multiplicar el factor de ponderación correspondiente por el valor del tiempo de simulación, y se sumaría directamente al término principal.

5.2.2.1.2 Modulo de la plantilla

Se define como modulo de la plantilla a la suma de las valores absolutos de los parámetros de la plantilla. La introducción de este término se debe a dos razones. Primero, por reducir grados de libertad, pues se comprueba asociado a cada operación hay una familia de posibles plantillas, que pueden moverse en un rango muy grande. Segundo porque se observa que si no se regula esto parece haber una tendencia a que los valores tomen valores grandes positivos y negativos, que parecen compensarse unos con otros pues realizan la misma operación que plantillas con valores pequeños. Para hacer que se tienda a este tipo de plantillas se añade este término. De nuevo la ponderación debe ser tal que no interfiera mucho con el término principal, y su optimización se produzca en un segundo plano.

El término se obtiene de multiplicar el factor de ponderación correspondiente por el módulo de la plantilla. Este valor se obtiene sumando los valores absolutos y dividiendo por el mayor valor posible, de manera que esté normalizado entre 0 y 1. El valor finalmente obtenido se suma directamente al término principal.

5.2.2.1.3 Penalización por tiempo total de simulación excesivo

Se debe poner una cota al tiempo total de simulación por que de no ser así plantillas poco estables o inestables podrían hacer una iteración muy larga o incluso interminable. Para ello se introduce una cota en este tiempo total de manera que si se rebasa se corta la simulación y se le asigna un valor de fitness 1.

5.2.2.2 Términos añadidos GP

5.2.2.2.1 Numero de funciones

Se debe penalizar el número de funciones para evitar que los árboles de la población sean demasiado complejos y lentos.

Para descartar árboles con un número de funciones determinado se utiliza el parámetro **valor de longitud peor**, de manera que si un árbol supera ese número de funciones se le asocia el peor fitness, valor 1, y no se llega a evaluar.

Si no queremos que la penalización se haga de manera tan brusca sino que queremos una penalización gradual se tiene la opción de evaluación de longitud. Para ello se debe activar la casilla correspondiente y poner el parámetro **valor de longitud malo** que indica la longitud máxima permisible sin penalización grave. La exponencial se hace de tal manera que vale 1 en el punto valor de longitud peor y 0.01 en valor de longitud malo, de manera que los valores por debajo de valor de longitud buena apenas se ven afectados por este factor. La optimización se hace principalmente en base al término principal, y en un segundo plano respecto de este término secundario. Entre valor de longitud buena y valor de longitud mala el término se hace apreciable de manera que se penalizan fuertemente longitudes cercanas a valor de longitud mala.

5.3 CODIFICACIÓN

5.3.1 CODIFICACIÓN GA

En nuestro caso los elementos que se representan son los parámetros de las plantillas. La forma de codificarlos es bastante sencilla, cada parámetro se codifica en

un byte, asignando el 0 al menor valor y el 255 al mayor valor. Los valores mayor y menor se han tomado 12 y –5 respectivamente. Debido a la tolerancia de los elementos de las CNN hacen que la precisión en la definición de los parámetros que permiten los 8 bits de cada byte sean más que suficiente. Asignar más bits a cada parámetro no solo innecesario, sino también contraproducente de cara a la eficiencia del GA, pues ampliaría en gran medida el espacio de búsqueda.

La codificación del cromosoma en una cadena de bytes se puede realizar de dos maneras distintas, que el usuario puede configurar:

- Codificación directa:

Cada parámetro de la plantilla se codifica en un byte. Cada cadena ocupa 19 bytes, correspondientes con los 19 parámetros de la plantilla.

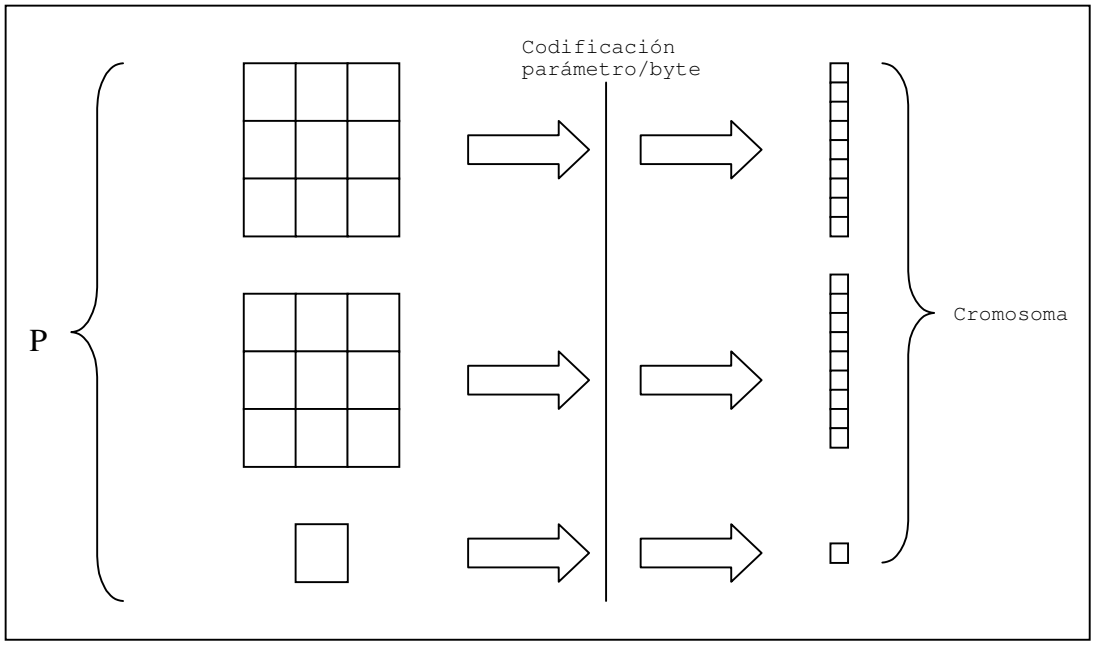


Figura 5.1 Codificación Directa

- Codificación simétrica:

Se reduce el número de parámetros a codificar en 7, al suponer simetría de las matrices A y B. El uso de esta codificación es muy recomendable, porque se reduce en gran medida el espacio de búsqueda, y porque la mayor parte de las plantillas cumplen propiedad de simetría. Se dice que la plantilla es simétrica cuando las matrices A y B son simétricas, tanto de abajo a arriba como de derecha a izquierda, de manera cada matriz queda definida por tres parámetros. Se puede aplicar esta propiedad siempre que se desee un procesado isótropo, en el que no haya direcciones predominantes. De nuevo cada parámetro se codifica con un byte, de manera que cada cromosoma se codificará con una cadena de 7 octetos.

Ambos tipos usan la misma codificación parámetro/byte, en el que intervienen dos variables con valores por defecto que pueden ser redefinidos por el usuario, escalado (de valor 15) y offset (de valor –5). Estos valores mapean los valores de parámetro –5 y 12 en valores de byte 0 y 255 respectivamente. La codificación en un byte se justifican por la tolerancia en los valores de los elementos de una red real, de

manera que aumentar la resolución de la codificación sería un desperdicio. La ecuación utilizada es:

$$ValorByte = \frac{(ValorParametro - offset)}{escalado}$$

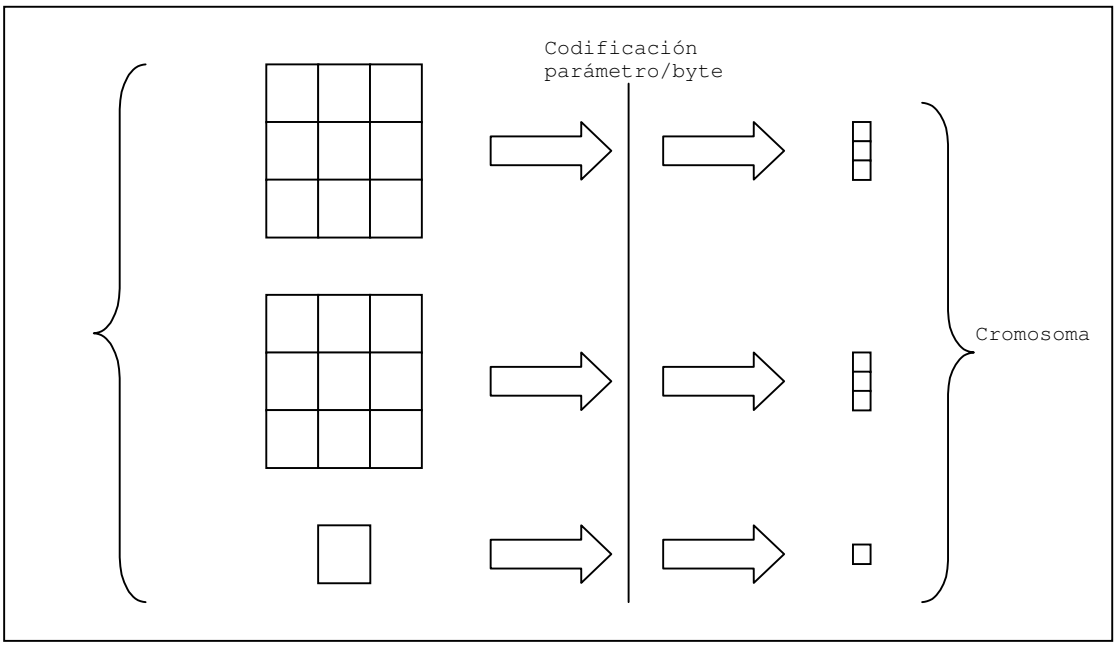


Figura 5.2 Codificación simétrica

5.3.2 CODIFICACIÓN GP

Para representar los individuos debemos realizar un primer paso que consiste en la elección de la estructura de datos más adecuada para codificar los algoritmos de tratamiento de imágenes que se utilizarán en la CNN-UM. En este caso se ha considerado el árbol de plantillas como la estructura mas adecuada para representar el espacio de algoritmos.

Se dijo que un árbol tenía dos tipos de nodos, terminal y función. Para nuestro caso los terminales son imágenes. Cada terminal será una de las imágenes disponibles en el juego de imágenes definido por el usuario para su procesamiento. Por otra parte las funciones representan un determinado procesamiento elemental de la CNN. Cada función estará asociada a una plantilla del juego de plantillas definido por el usuario. Se deja a elección del usuario qué y cuántas plantillas entran en juego, pero debe tener en cuenta que cada vez que se añade una plantilla se está ampliando el espacio de búsqueda, de manera que la búsqueda será tanto más eficiente cuánto menor sea esta número. Asimismo el usuario puede introducir plantillas creadas por él mismo. La salida de las funciones es una imagen que va a la entrada de otra función, excepto la función del nodo raíz, cuya salida es el resultado del algoritmo.

Cada nodo lo vamos a codificar en un byte. Disponemos de un campo de 256 valores, número más que suficiente para codificar todos los tipos de nodo, tanto funciones como terminales. Disponemos de 100 posiciones para codificar los terminales, y de 156 para codificar las funciones. La asignación depende del orden en que se definan los juegos de terminales y de funciones. En el caso de los terminales, al

primero se le asigna el 99 al segundo el 98, y así. De igual manera a las funciones se les asigna el 100 al primero, el 101 al segundo, y así sucesivamente.

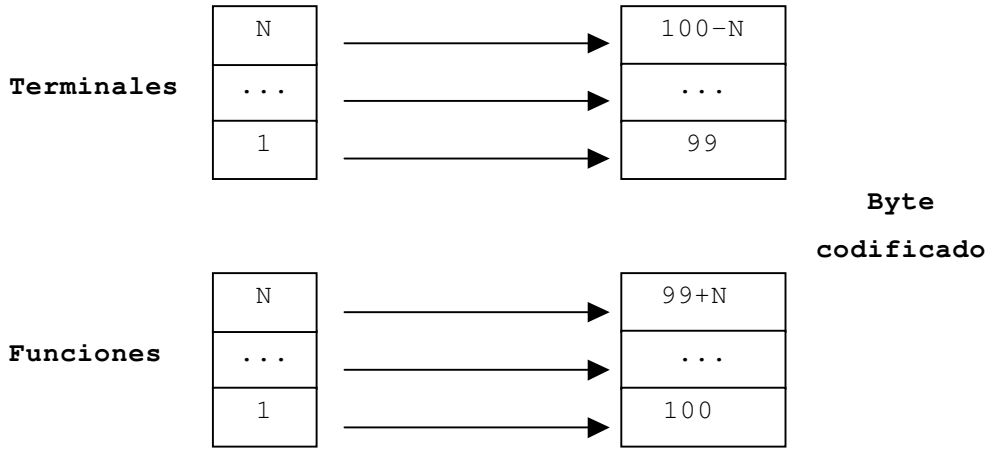


Figura 5.3 Codificación de nodo

En la codificación de árbol se utiliza la notación prefijo para la codificación, y se aplica la regla de Rosembloom al la hora de manipular el árbol. Esto ya fue comentado anteriormente y no se va a insistir en ello. El resultado de aplicar la codificación será una cadena de bytes, en el que cada byte representará un nodo, y éstos nodos aparecen ordenados según la notación prefijo.

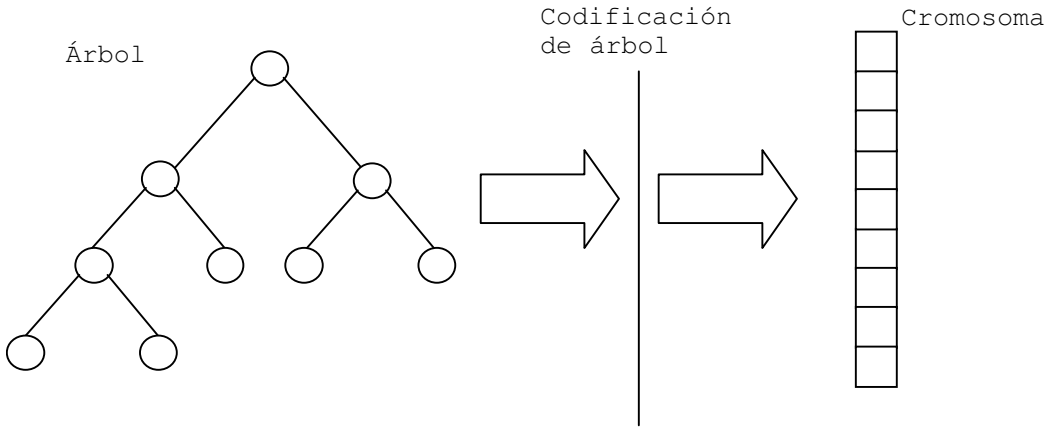


Figura 5.4 Codificación de árbol.

5.3.3 PRESERVACIÓN DE CONCORDANCIA DE TIPOS EN ÁRBOLES

Es muy corriente que en un algoritmo se trate con imágenes tanto binarias como en escala de grises, así como funciones que están diseñadas para tomar a la entrada uno u otro tipo de imagen, y que dan a la salida uno u otro tipo. Nos referimos en este punto a concordancia de tipos en árbol a la propiedad según la cual todas las funciones presentes en un árbol toman a la entrada imágenes con el tipo para el que han sido diseñadas.

La preservación de la concordancia en los árboles de la población es un aspecto muy importante en la aplicación de los operadores genéticos así como en la generación de la población, pues si no se tuviese en cuenta este aspecto, en general tendríamos una mayoría de la población que no mantendría esta concordancia.

Para determinar los tipos de imagen de entrada con los que estamos tratando, se recorren cada imagen de entrada píxel a píxel, de manera que si solamente contiene valores extremos se determinará que se trata de una imagen binaria, mientras que si se observa algún valor intermedio se considerará que se trata de una imagen en grises.

Asimismo se registran los tipos de entrada para los que están diseñados las plantillas, información que está recogida en los ficheros de plantilla. Es por tanto muy importante que se recoja correctamente esta información en el fichero, y toda plantilla que sea introducida por el usuario deberá cumplir este aspecto.

Tanto generación de la población inicial como de las sucesivas poblaciones se mantiene en todo momento la concordancia de los individuos de la población, de manera que la aplicación de los operadores genéticos se realiza de forma controlada, de manera que no den lugar a individuos no concordantes

6 PAQUETE SOFTWARE.

6.1 DESCRIPCIÓN.

En el presente proyecto se ha desarrollado un paquete que consta de cuatro programas, que podemos clasificar dos a dos tanto por su naturaleza (síntesis / análisis) como por las resultados que obtiene y manipula (plantillas / algoritmos). Son los siguientes:

- ***GA Template***: Según la clasificación propuesta anteriormente sería síntesis-plantilla. Trata de obtener automáticamente la plantilla que introducida en la red realice la operación especificada por el usuario mediante el conjunto de imágenes de entrada y de salida deseada, con el mayor parecido posible entre las imágenes de salida resultante y deseada, con opción a tener en cuenta otros parámetros a optimizar, como son el tiempo de convergencia de la plantilla y la magnitud de los valores de la plantilla. Para ello utiliza la técnica de los algoritmos genéticos.

- ***Template Simulator***: Le correspondería el tipo análisis-plantilla. Realiza la simulación de la plantilla que introduzca el usuario, que puede ser una obtenida de forma automática mediante el programa anterior, o bien puede ser introducida directamente por el usuario. De manera que el programa podría ser utilizado para medir la calidad de la plantilla obtenida mediante técnicas genéticas, o bien para que el usuario refine una plantilla simplemente por prueba y error.

- ***GP Algorithm***: Tipo síntesis-algorítmo. El objetivo de este programa es obtener automáticamente un algoritmo que realice un procesado de imagen más complejo que el que se obtiene simplemente con una sola plantilla, mediante la combinación de procesado de varias plantillas. Las plantillas que entran en juego en el programa pertenecerán a un conjunto de plantillas base, que el usuario se encargará de configurar a partir de la biblioteca de plantillas que se ofrece con el programa, la cual puede ser ampliada por el usuario utilizando para ello los dos programas anteriores. El programa se representa con una estructura de árbol.

- ***Algorithm Simulator***: Tipo análisis-algoritmo. Con este programa se puede validar el funcionamiento adecuado del programa obtenido con distintas entradas, así como obtener el tiempo total de procesamiento, suma de los tiempos de procesamiento de las plantillas que utilice el programa. Además de cargar un programa obtenido automáticamente mediante la utilización del programa anterior se permite la edición por parte del usuario para la mejora o el diseño completo de un programa.

En la figura 6.1 se muestra un esquema en el que se pone de manifiesto la interacción de los distintos programas con el usuario y entre ellos. Como se puede observar en el esquema el resultado que se obtiene es, bien una plantilla que realice un procesamiento muy específico que no se encuentre entre las plantillas de la biblioteca proporcionada, o bien un programa, un algoritmo de procesamiento que combine el conjunto de plantillas base que el usuario haya configurado.

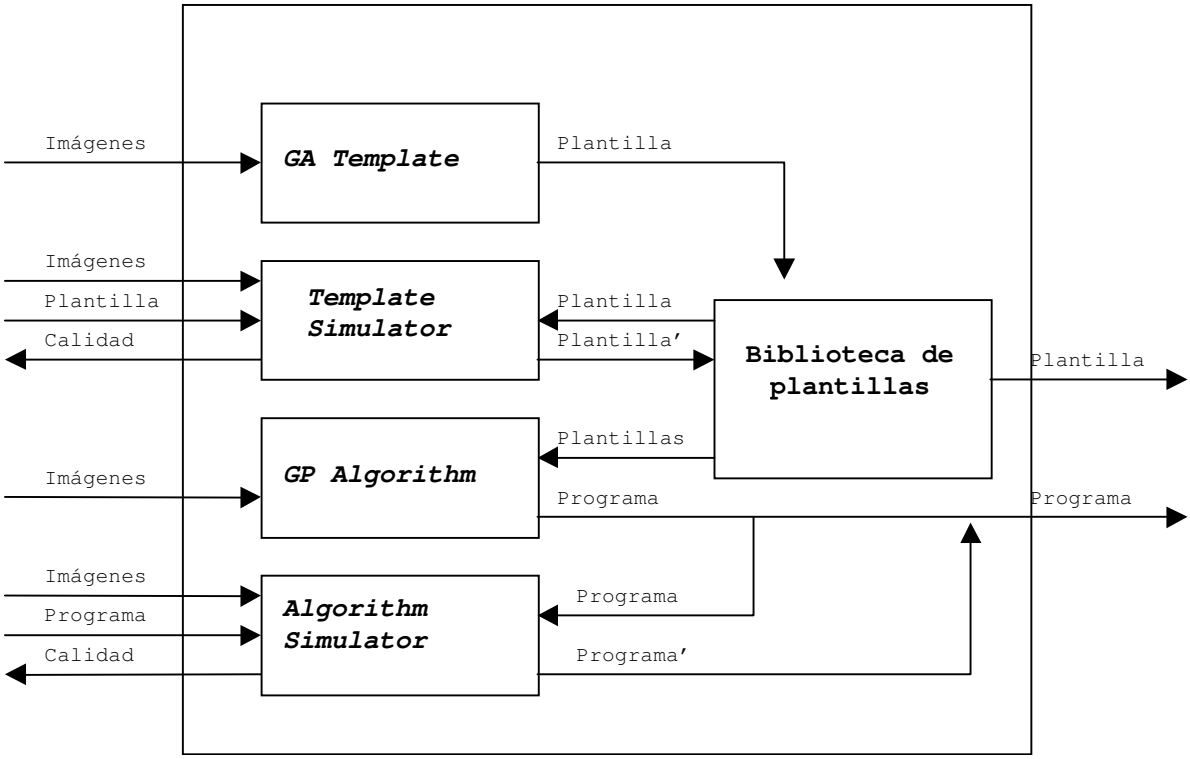


Figura 6.1 Interacción de los programas

6.2 QUÉ INCLUYE EL PAQUETE SOFTWARE

Los ficheros incluidos en el paquete están distribuidos en varios directorios. La estructura de directorios se muestra en la figura 6.2:

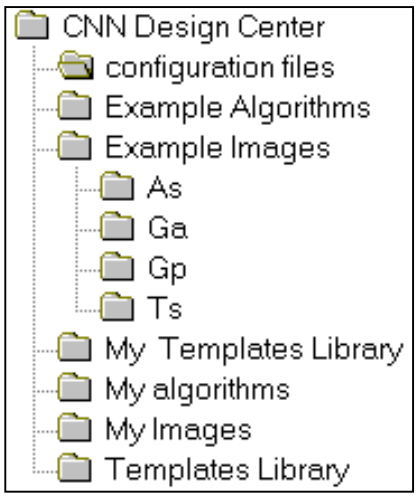


Figura 6.2 Estructura de directorios del paquete

Para un correcto funcionamiento de los ficheros de configuración de funciones incluidos y los algoritmos de ejemplo se debe instalar el paquete en el directorio raíz de la unidad c, en *C:*. En caso contrario se deberán editar tanto los ficheros de

configuración de funciones como los ficheros que contienen los programas, puesto que tanto las plantillas como las imágenes estarán ubicadas en un sitio diferente al supuesto.

El paquete desarrollado incluye los siguientes ficheros:

- Ficheros ejecutables correspondientes a los cuatro programas, que están en el directorio base. Son los siguientes:
  - *Template simulator.exe*
  - *GA Template.exe*
  - *Algorithm simulator.exe*
  - *GP Algorithm.exe*
- Los ficheros tipo *dll* necesarios para su funcionamiento, que están en el directorio base.
- Las plantillas de la biblioteca, que se incluyen en el directorio *templates Library*. En el anexo se comentan detalladamente cada plantilla.
- Las imágenes de ejemplos. Se encuentran en el directorio *Example images*. Las imágenes a su vez se encuentran distribuidas en 4 directorios, *TS*, *GA*, *AS* y *GP*, dependiendo del programa al que correspondan.
- Algunos algoritmos de ejemplo, que se incluyen en el directorio *Example Algorithms*.

Los directorios *My templates library*, *My Images* y *My Algorithms* están vacíos y se incluyen para uso del usuario. En cualquier caso el usuario tiene libertad de almacenar ficheros en los directorios que estime oportuno.

6.3 TIPOS DE FICHEROS INTRODUCIDOS

En los programas incluidos en el paquete se introducen varios tipos de ficheros que a continuación se comentan.

6.3.1 FICHERO DE PLANTILLA (.TEM)

Se utiliza un determinado formato de fichero para el almacenamiento y recuperación de plantillas. Estas plantillas son codificadas en modo texto, de manera que cualquier usuario puede editar la plantilla que requiera sin más que seguir correctamente el formato que se especifica a continuación.

El fichero se divide en distintos campos, delimitados cada uno de ellos mediante etiquetas, que definen de que campo se trata. Se pueden utilizar indistintamente mayúsculas y minúsculas para la declaración de una etiqueta. Una etiqueta es una de las 8 palabras clave correspondientes a los 8 campos distintos que se pueden definir. Estos son:

- **Feedback**: los elementos de este campo son los 9 valores que definen la matriz de realimentación A.
- **Control** : los elementos de este campo son los 9 valores que definen la matriz de control B.
- **Current**: Este campo contiene un elemento, el valor del bias I.

- **Boundary**: Este campo contiene un elemento, un número entero que define la condición de contorno en el procesamiento.
- **Input0**: Mediante este campo definimos el tipo de imagen correspondiente con la primera entrada. Esta entrada corresponde con la imagen entrada de la CNN.
- **Input1**: Mediante este campo definimos el tipo de imagen correspondiente con la segunda entrada. Esta entrada corresponde con la imagen estado inicial de la CNN.
- **Output**: Mediante este campo definimos el tipo de imagen correspondiente con la salida.
- **Implementation**: Para la aceleración del programa GP se incluye este campo en el que mediante un valor entero comprendido entre el 1 y el 12 para la aceleración de 12 plantillas de procesamiento localizado en el píxel, de manera que en caso de ocurrencia de uno de estos valores se realiza un procesamiento equivalente sin simulación de la CNN. El valor 0 es el valor por defecto y sirve para indicar que se trata de una plantilla con implementación en CNN.

Los valores posibles del campo 'Boundary' son los siguientes:

- Valor '0': Condición de contorno tipo fija (fixed).
- Valor '1': Condición de contorno tipo toroide (torus).
- Valor '2': Condición de contorno tipo reflejado (reflected).

Los valores posibles que pueden tomar los dos campos anteriores para definir el tipo de imagen son los siguientes:

- Valor '1': Imagen en escala de grises
- Valor '2': Imagen binaria
- Valor '3': Imagen todo -blanco
- Valor '4': Imagen todo -negro
- Valor '5': Imagen todo -gris neutro
- Valor '6': Imagen espejo
- Valor '7': Imagen todo -blanco, borde negro
- Valor '8': Imagen todo -negro, borde blanco

Los tipos 1 y 2 son entradas que requieren una imagen de entrada del tipo específicado. En caso de que la otra entrada sea también de uno de estos dos tipos tendremos una plantilla binaria. En caso contrario tendremos una plantilla de tipo unario.

Los tipos 3, 4, 5, y 6 no requieren una imagen de entrada, la otra entrada debe necesariamente ser de tipo 1 o 2, y dan lugar a plantillas de tipo unario.

La imagen tipo 6, 'imagen espejo', significa que la entrada en cuestión toma la misma imagen que la otra entrada.

El campo Output solamente puede adoptar dos valores '1' o '2', para indicar sima imagen está en escala de grises, o si se trata de una imagen binaria.

```
%Logic And

feedback:
0 0 0
0 2 0
0 0 0
control:
0 0 0
0 1 0
0 0 0
current:
-1
boundary:
0
input0:
2
input1:
2
output:
2
implementation:
1
```

```
%Halftoning

feedback:
-0.070000   -0.100000 -0.070000
-0.100000    1.100000 -0.100000
-0.070000   -0.100000 -0.070000
control:
0.070000 0.100000 0.070000
0.100000 0.320000 0.100000
0.070000 0.100000 0.070000
current:
0.000000
boundary:
2
input0:
1
input1:
6
output:
2
implementation:
0
```

El campo implementatión tiene trece valores posibles. Estos son:

- 0: Implementación en CNN
- 1: Logic And
- 2: Logic Or
- 3: Logic Inv
- 4: Logic Inv
- 5: Threshold –0.8
- 6: Threshold –0.6
- 7: Threshold –0.4
- 8: Threshold –0.2
- 9: Threshold 0
- 10: Threshold 0.2
- 11: Threshold 0.4
- 12: Threshold 0.6
- 13: Threshold 0.8

Se permite la inclusión de comentarios en el fichero, mediante el uso de el símbolo '*%*', de manera que una línea que comience con este símbolo será considerada línea de comentario. Los caracteres tipo espacio, tabulador y retorno de carro son ignorados.

Como ejemplo se muestra en la figura 6.2 los ficheros correspondientes a dos plantillas de la biblioteca incluida en el paquete, LogicAnd.tem y Halftoning.tem

6.3.2 FICHERO DE CONFIGURACIÓN DE FUNCIONES

Se trata de un fichero de texto con extensión *.cfg* que indica las funciones, que corresponden con plantillas, que se tienen en cuenta en un determinado algoritmo de CNN. Los programas del paquete software que utilizan este fichero son el *Simu Algoritmo* y el *GP Algoritmo.*

Este fichero debe ser editado por el usuario en modo texto de manera que en cada línea aparezcan las distintas plantillas a considerar mediante el nombre del mismo, con la ruta de acceso completa.

6.3.3 FICHEROS DE ALGORITMOS CNN

Tienen extensión *prm* y contienen un programa o algoritmo de CNN. Desde el programa *GP Algorithm* se puede almacenar algún algoritmo correspondiente a cualquiera de los individuos de la población. Desde el programa *Algorithm Simulator* se puede cargar un algoritmo desde un fichero, o bien se puede almacenar alguno que haya sido editado en este programa.

No está en modo texto, de manera que no se recomienda la edición directa del fichero, que podría provocar resultados imprevisibles en su utilización. Tampoco se deben modificar los ficheros correspondientes a imágenes, plantillas o ficheros de configuración de funciones involucrados en algún algoritmo almacenado, pues puede provocar errores en la ejecución del algoritmo.

6.3.4 FICHEROS DE ESTADO

Se trata de ficheros de extensiones *.gps* y *.gas* que permiten realizar la optimización genética en varias sesiones, pararando el proceso para continuarlo en otro momento mediante la recuperación del estado. Esto puede resultar muy práctico debido a la alto coste de tiempo que supone un optimización completa. Estos ficheros se utilizan en los programas del paquete software *GA Plantilla* , al que corresponde el fichero con extensión *.gas* , y *GP Algoritmo* al que corresponde el *.gps* .

Un aspecto importante es que no se trata de un fichero de texto, sino que se almacena directamente en binario, de manera que cualquier intento de edición en modo texto del mismo es totalmente desaconsejable y puede tener resultados imprevisibles.

6.3.5 FICHEROS DE EVOLUCION DE VARIABLE EN EL TIEMPO

En los distintos programas del paquete software se pueden extraer una serie de resultados. Uno de los más representativos son los ficheros que contienen la evolución de una determinada variable en modo texto. Estos ficheros pueden contener una columna, que representa directamente una variable, o dos columnas, que puede representar sendas variables, o bien en cada fila se representa la variable junto al tiempo correspondiente. En cualquier caso la representación gráfica de las variables asociadas a los ficheros es muy sencilla con cualquier programa que lo permita.

7 PROGRAMA *TEMPLATE SIMULATOR*.

7.1 DESCRIPCIÓN

La finalidad de este programa es simular el procesamiento de una determinada plantilla en la CNN emulada, mediante la integración numérica de la ecuación diferencial que define la evolución del estado en cada celda para una determinada plantilla. El usuario podrá utilizar el programa para validar una determinada plantilla obtenida por otro programa, como ayuda en el diseño y afinación, o simplemente como comprobación del procesamiento de una imagen en general.

El usuario debe introducir distintos parámetros que determinan la simulación:

- La plantilla, que se puede introducir manualmente o mediante fichero de plantilla que determinan el procesamiento de la CNN.
- Las imágenes *entrada* y *estado inicial*, que serán introducidas en la CNN.
- Los parámetros de simulación *variación a predecir* y *variación límite*, que especifican como se debe variar el paso de integración (Ver apartado 6.2)
- Las condiciones de parada, que pueden ser por número de pasos de parada, tiempo de integración de parada, o por máxima variación parada.

A la hora de realizar la simulación hoy tres modos de simulación:

- Simulación completa: Se realiza la simulación hasta que se cumpla una condición de parada.
- Simulación paso a paso: Al pulsar el botón se realiza un paso de simulación, y se muestra la salida
- Simulación Ver transitorio: Se hace la simulación mostrando por pantalla el transitorio de la salida hasta convergencia.

Una vez que la simulación ha terminado es el momento de extraer resultados. Los resultados se obtienen en forma de fichero y son tres: Imagen de salida, salida media y parámetro de variación de estado.

- La imagen de salida es salida de la CNN en la última iteración en formato de imagen en mapa de bits (fichero BMP estándar).
- La salida media es la evolución de la media del valor absoluto de la salida en función del tiempo de simulación. Sirve para comprobar cuando la CNN alcanza la imagen final, teniendo en cuenta que los valores de la salida se encuentran entre –1 y 1 será un valor entre 0 y 1. En caso de imagen binaria la salida alcanzará a la imagen final cuando el valor de este parámetro alcance al 1 y se mantenga en este valor. No es un parámetro que sirva para determinar convergencia, puesto que en algunos casos puede alcanzar al uno incluso varias veces antes de que la salida muestre la imagen final. En caso de imagen de grises simplemente permanecerá en un valor entre 0 y 1 cuando se alcance la imagen final.

- En el parámetro de variación de estado se muestra la mayor pendiente de variación de estado en una iteración, en función del tiempo de simulación. Se obtiene sin más que dividir en la variación de estado mayor de cada iteración por el paso de integración. Este parámetro nos permite hacernos una idea de cómo varía el estado de la red en función del tiempo, y se observa que hay fases de mayor variación y fases de menor variación. Sirve como criterio de convergencia. Para más detalle ver el apartado 6.2 sobre la simulación por computador de la CNN.

7.2 IMPLEMENTACIÓN DEL SOFTWARE

El programa ha sido implementado en una estructura multihilo. De no ser así el programa quedaría completamente bloqueado en momentos de procesamiento, que dependiendo del mismo puede implicar una cantidad de tiempo mayor o menor, pudiendo llegar a ser intolerable para un usuario. En los tres modos de simulación se monitoriza perfectamente el estado de la misma en todo momento. En el modo de simulación se puede observar el tiempo total, el nº de paso y el paso de integración en cada iteración. En los modos paso a paso y ver transitorio también se puede ver la imagen de salida parcial.

El hecho de trabajar con hilos hace que se tengan que tratar temas como las condiciones de carrera y la sincronización. Para resolver las condiciones de carrera se utilizan secciones críticas, de manera que haya una coherencia en los datos intercambiados por el hilo principal y el hilo de simulación. Para solucionar la sincronización e utilizan eventos. Para comunicar los hilos se utilizan los mensajes.

En el momento en que se acciona el botón de simulación en cualquiera de los modos se hacen dos cosas, se inicializa la CNN y se lanza el hilo de simulación con el modo correspondiente. El hilo de simulación comienza a hacer iteraciones de simulación hasta que se cumple la condición de parada. Notifica cada iteración y el fin de la simulación mediante el uso de mensajes.

A continuación se muestra un esquema en el que se muestra el funcionamiento básico para el modo de simulación completa. Se puede apreciar que en la comunicación entre los dos hilos hay dos mensajes, el mensaje de fin de iteración y el mensaje de fin de simulación. El hilo principal llamará a una función, según el mensaje recibido. La respuesta al mensaje fin de iteración es tomar los parámetros monitorizados en la simulación (el tiempo total, el nº de paso y el paso de integración). Este paso de información entre los dos hilos está regulado por una sección crítica para que la información sea coherente en todo momento. La respuesta al mensaje fin de simulación es mostrar la imagen de salida resultante e indicar por pantalla que ha finalizado la simulación.

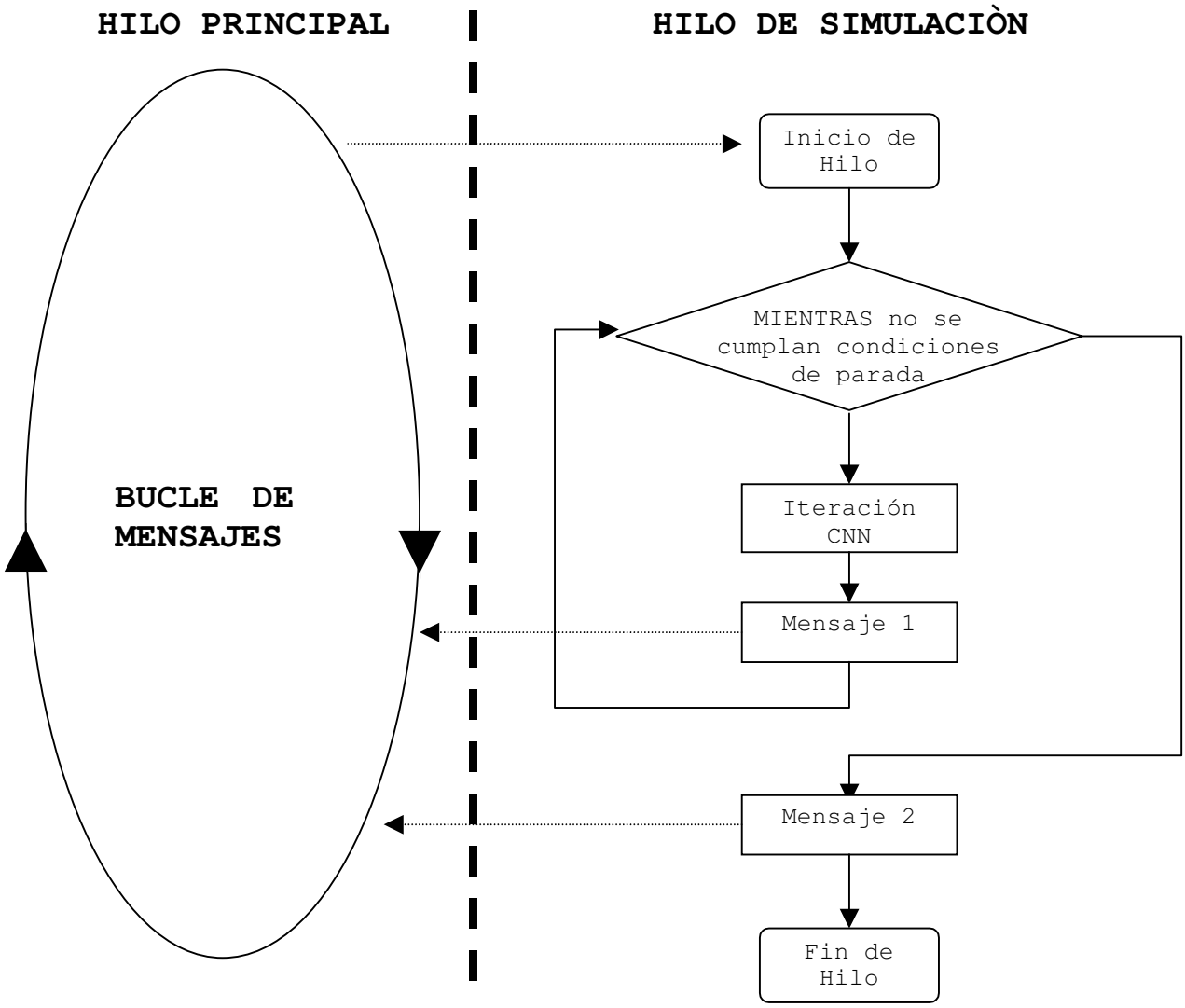


Figura 7.1 Funcionamiento básico en modo simulación completa

A continuación se muestra un esquema que es válido tanto para representar el modo de funcionamiento paso a paso como para el modo ver transitorio. La diferencia con el esquema anterior es que al final de cada iteración el hilo de simulación se bloquea hasta que llegue una notificación por parte del hilo principal. De nuevo se observa dos tipos de mensaje, siendo la respuesta a ambos mensajes básicamente la misma. La diferencia está en la respuesta al mensaje de fin de iteración. Además de mostrar los parámetros monitorizados en la también la imagen de salida. En caso de estar en modo ver transitorio, después de esto se envía una notificación al hilo de simulación para que se desbloquee y siga con la simulación. En caso de estar en modo paso a paso no se desbloquea, y el hilo se queda esperando hasta que se vuelva a accionar el botón de simulación paso a paso, momento en el que se notificará el desbloqueo para que realice una iteración más.

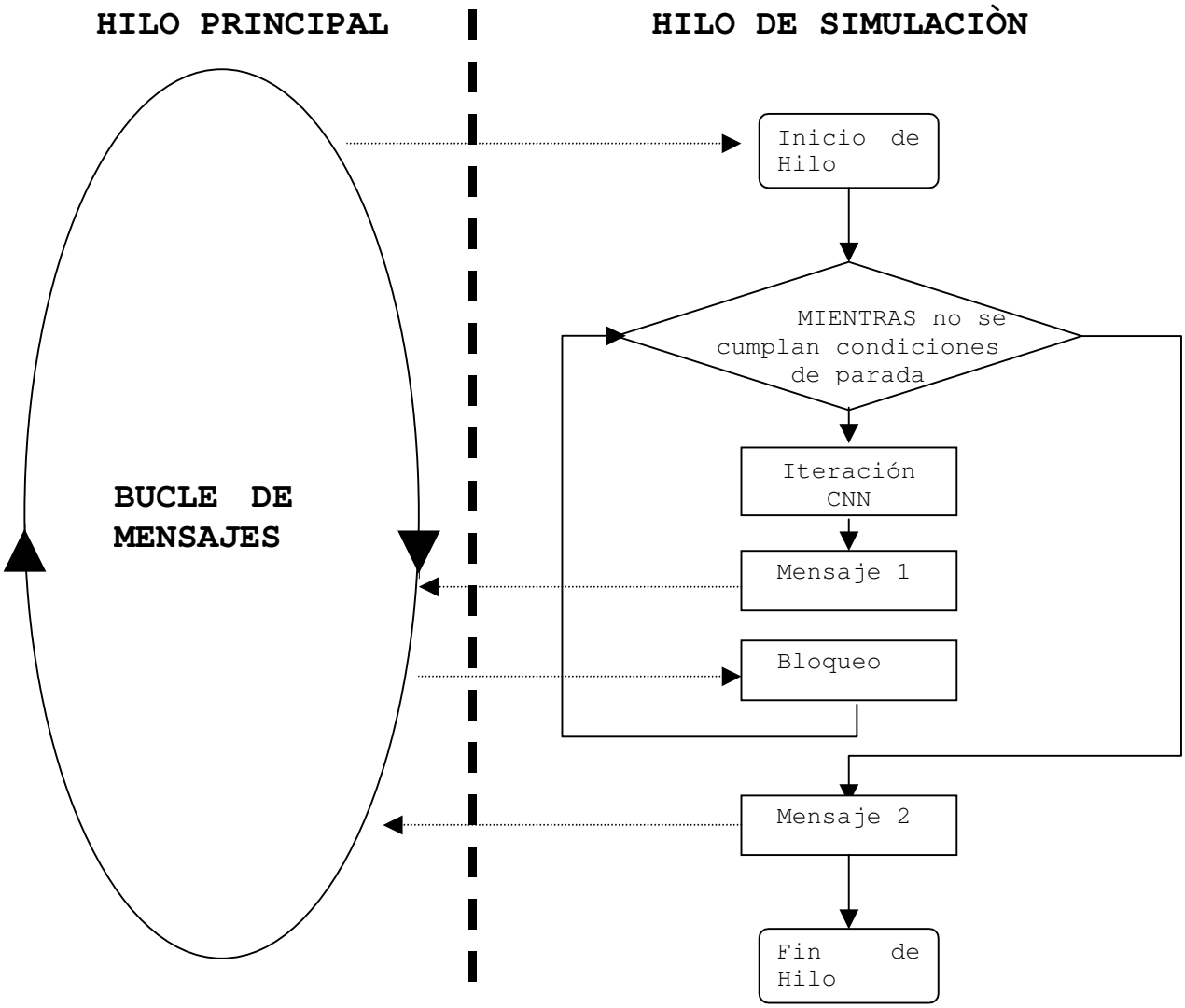


Figura 7.2 Funcionamiento básico en modo *paso a paso* y *ver animación*

7.3 MANUAL DE USUARIO.

Para abrir el programa se debe ejecutar el fichero *Template simulator.exe*. En ese momento se abrirá la ventana principal del programa que se muestra en la figura 7.3.

Todos los controles están a la vista en esta ventana. En el gráfico anterior se ha asociado cada control con una número etiqueta. A continuación se comentan uno por uno cada control, explicando en cada caso su finalidad.,e indicando el número de etiqueta correspondiente a cada control:

- Imágenes

Botón ***Open input image*** (Etiqueta 1): Sirve para indicar el archivo que contiene la imagen que queremos que se introduzca en la CNN como Entrada. Debe ser un archivo BMP en escala de grises y 8 bits de profindidad.

Botón ***Open initial state image*** (Etiqueta 2): Sirve para indicar el archivo que contiene la imagen que queremos que se introduzca en la CNN como Estado Inicial. Debe ser un archivo BMP en escala de grises y 8 bits de profindidad.

Input, Initial State, Output (Etiquetas 3, 4 y 5): Espacio reservado donde se mostrarán las imágenes de entrada, estado inicial y salida de la red.

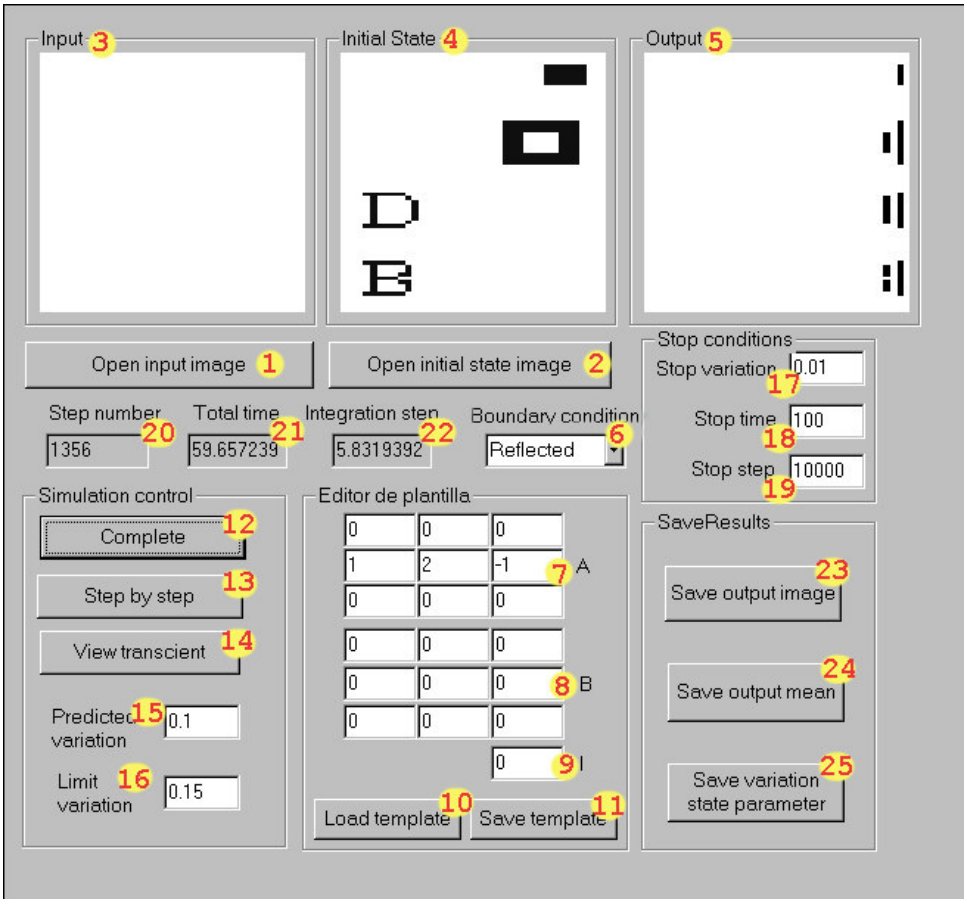


Figura 7.3 Ventana principal del programa

- Configuración de plantilla

Boundary condition (Etiqueta 6, Condición de contorno): Mediante un Combo Box se elige una de las tres posibles condiciones de contorno de la CNN, Fija, Toroide o Refleja.

A, B, I(Etiquetas 7, 8 y 9): En cajas de texto podemos escribir manualmente los valores de los parámetros de la plantilla.

Load Template (Etiqueta 10, Cargar plantilla): Sirve para indicar un fichero que contenga una plantilla, y dicha plantilla se cargará en la CNN.

Save template(Etiqueta 11, grabar plantilla): Sirve para almacenar la plantilla que haya en ese momento en un fichero.

- Opciones de simulación

Botón ***Complete*** (Etiqueta 12, simulación completa): Sirve para indicar que la simulación se realice de forma continuada hasta el fin, sin mostrar ningún tipo de imagen parcial.

Boton ***Step by step*** (Etiqueta 13, paso a paso): Sirve para indicar que se realice un paso de iteración en la simulación.

Botón ***View transcient*** (Etiqueta 14, Ver Transitorio): Sirve para indicar que se realice la simulación hasta el final, pero mostrando las imágenes parciales de la simulación, de manera que se pueda observar el transitorio.

Variation to predict (Etiqueta 15, Variación a predecir): Parámetro que sirve para establecer un criterio en la variación del paso de integración a lo largo de la simulación. Ver apartado 6.2 para más detalle

Limit variation (Etiqueta 16, Variación límite): Parámetro que sirve para establecer un criterio en la variación del paso de integración a lo largo de la simulación. Ver apartado correspondiente para más detalle

- Condiciones de parada.

Stop variation (Etiqueta 17, variación de parada): Establece valor de la variación de estado que determina convergencia. Ver apartado correspondiente para más detalle

Stop time (Etiqueta 18, Tiempo de parada): Fija una cota en tiempo de simulación.

Stop step (Etiqueta 19, paso de parada): Fija el número de pasos de iteración máximo en la simulación.

- Parámetros monitorizados durante la simulación

Step number (Etiqueta 20, número de paso): Indica el número de paso actual de la simulación.

Total time (Etiqueta 21, tiempo total): Indica el tiempo total de simulación hasta ese momento. Recordar que es tiempo normalizado.

Integration step(Etiqueta 22, paso de integración): Indica el paso de integración utilizado en la última iteración.

- Almacenamiento en fichero de resultados:

Botón ***Save input image*** (Etiqueta 23, salvar imagen de salida): Guarda en un fichero BMP el la imagen de salida actual. Esta imagen de salida puede ser final o parcial.

Botón ***Save output mean*** (Etiqueta 24, salvar salida media):Sirve para almacenar en un fichero de texto la evolución de la media del valor absoluto de la salida frente el tiempo de simulación en un fichero de texto, para su observación directa o bien para representar la gráfica correspondiente con algún programa, por ejemplo MatLab.

Botón ***Save state variation parameter*** (Etiqueta 25, salvar variación de estado): Sirve para almacenar en un fichero de texto la evolución de la variación máxima de estado en cada iteración en función del tiempo de simulación. Una vez tenemos el fichero podemos hacer una observación directa, o bien representar mediante algún programa que lo permita, como MatLab.

7.4 EJEMPLOS DE APLICACIÓN

A continuación se muestran varios ejemplos en los que se muestran las capacidades del programa.

7.4.1 RECONSTRUCTOR DE FIGURA.

La plantilla que a continuación se va a simular muestra a la salida las figuras de la entrada seleccionadas en el estado inicial. La forma de seleccionar una figura es que haya algun pixel negro en el estado inicial coincidente con algun pixel de la figura.

La plantilla correspondiente es la siguiente:

$$A=\begin{pmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & 4 & 0.5 \\ 0.5 & 0.5 & 0.5 \end{pmatrix} \quad B=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad I=3$$

Podemos introducir los valores de la plantilla manualmente, o bien cargarlos desde un fichero directamente. El fichero que recoge esta plantilla es el incluido en la biblioteca con nombre ***FigureReconstructor.tem***.

Es concreto se toman las siguientes imágenes como entrada y estado inicial:

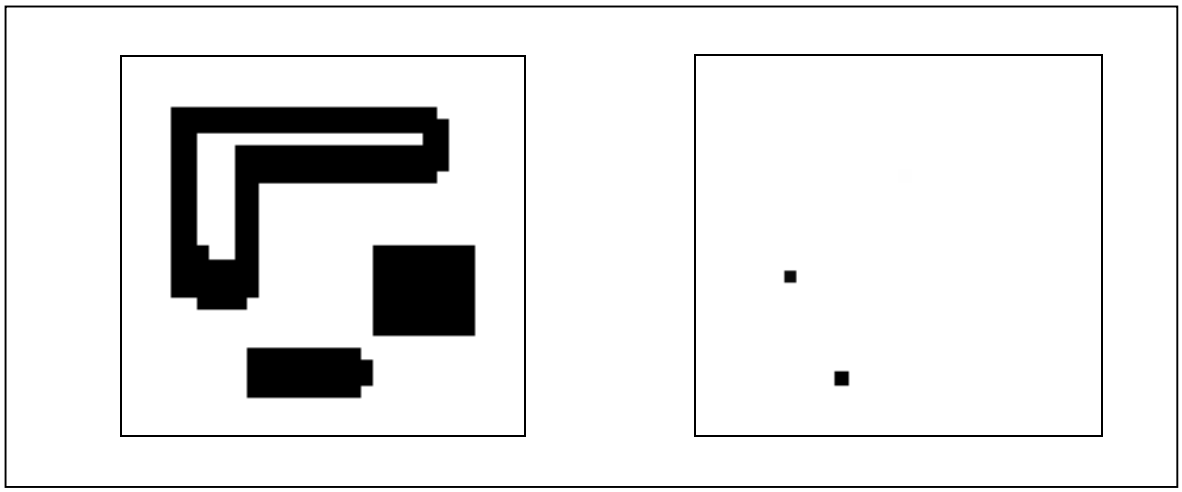


Figura 7.4 Imágenes de entrada y estado inicial

La condición de contorno se selecciona como refleja. Podría ser cualquiera, en este caso no va a influir. Los parámetros de simulación se toman como siguen:

- Variación a predecir: 0.1
- Variación límite: 0.15
- Variación de parada: 0.1
- Tiempo de parada: 200
- Paso de parada: 10000

En este punto estamos en condiciones de comenzar la simulación. Para ver las imágenes del transitorio de la salida accionamos el botón ver transitorio. Se observa una evolución en la salida. Se muestran algunas imágenes parciales en la figura 7.5..

En los indicadores observamos que se ha realizado en 578 pasos de iteración, con un tiempo normalizado de 10.405. Una forma de asegurarrnos de que este valor de tiempo de simulación no es engañoso es observar el paso de integración del último paso. En este caso es de 0.9713, y como este valor es bastante menor que el tiempo totalpodemos considerar que el tiempo anterior es verdadero.

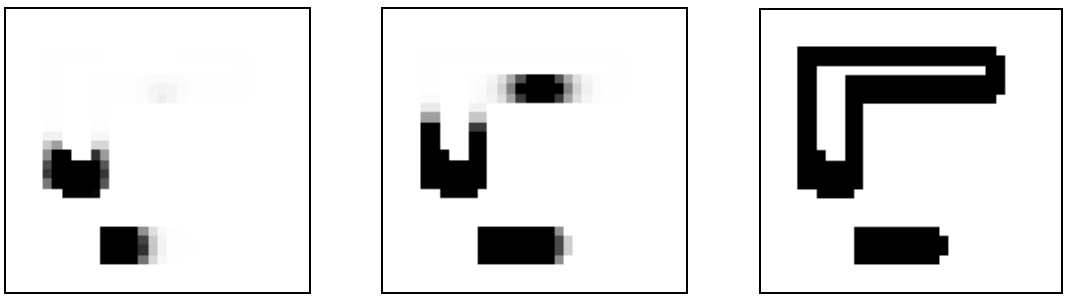


Figura 7.5 Imágenes transitorias en la simulación

Finalmente analizamos la evolución de la media de la magnitud de la salida y la evolución de la pendiente de la variación máxima de estado. Para ello almacenamos en un fichero de texto y abrimos con algún programa que permita la representación de funciones a partir de un fichero de texto, como *MatLab*. Se observan las siguientes gráficas:

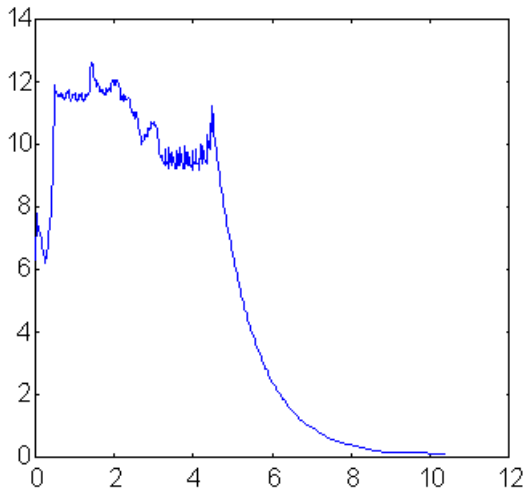


Figura 7.6 Representación gráfica de la evolución de la variación del estado

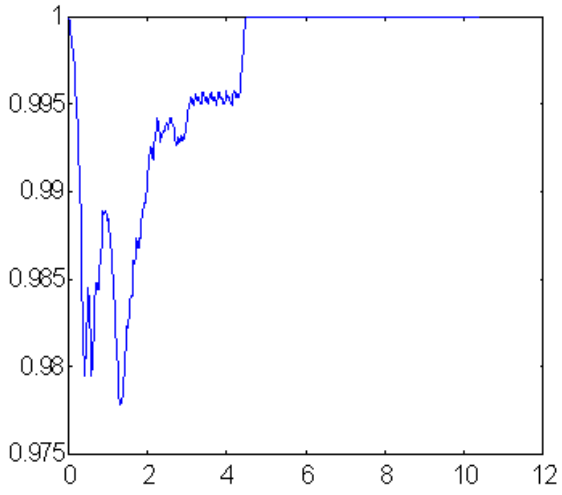


Figura 7.7 Representación gráfica de la evolución de la salida media

Se puede ver como tras un transitorio la salida alcanza su valor final entorno a 5, pero no podemos considerar convergencia en este punto puesto que el estado está aún lejos de la convergencia con variaciones apreciables del estado en esa zona. A la vista de esta gráfica convalidamos que podemos considerar convergencia entorno a 10, que pasado a tiempo real para un valor de $\tau = 250$ ns da un tiempo de procesamiento de 2.5 μs.

Si en la condición de parada hubiéramos puesto un valor de variación mucho menor, por ejemplo 0.01, hubiéramos obtenido un valor de tiempo de simulación de

23.8. Simplemente con la observación del paso de integración en el último paso se observa que se puede tratar de un tiempo engañoso, como así es, pues este valor es 11.3, casi la mitad del tiempo total. Además si observamos la evolución en gráficas de la manera anteriormente comentadas, obtendremos la continuación de las gráficas anteriores, que se observa que ya han alcanzado la convergencia y no tiene por tanto sentido seguir con la simulación tanto tiempo.

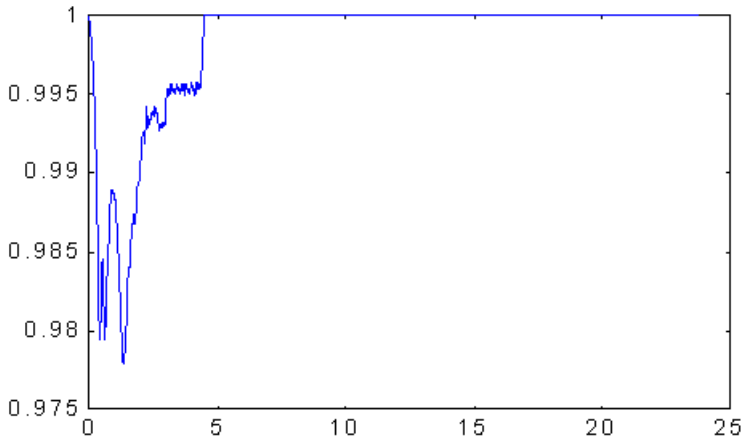


Figura 7.8 Representación gráfica de la evolución de la salida media

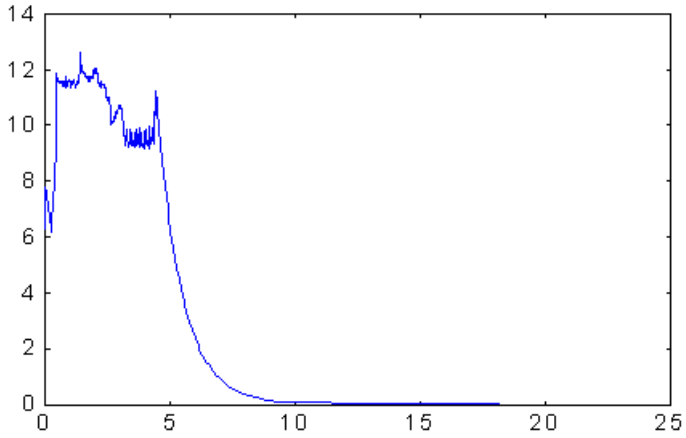


Figura 7.9 Representación gráfica de la evolución de la variación del estado

7.4.2 DETECTOR DE CONECTIVIDAD DE COMPONENTE VERTICAL

La plantilla que a continuación analizamos da lugar a una imagen de salida en la que muestra en número de componentes verticales conectadas de la figura de la entrada. La plantilla correspondiente a este procesamiento es la siguiente:

$$A=\begin{pmatrix} 0 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & -1 & 0 \end{pmatrix} \quad B=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad I=0$$

Estos valores pueden ser introducidos manualmente o cargando el fichero correspondiente.

Se toman las siguientes imágenes como estado inicial y entrada:

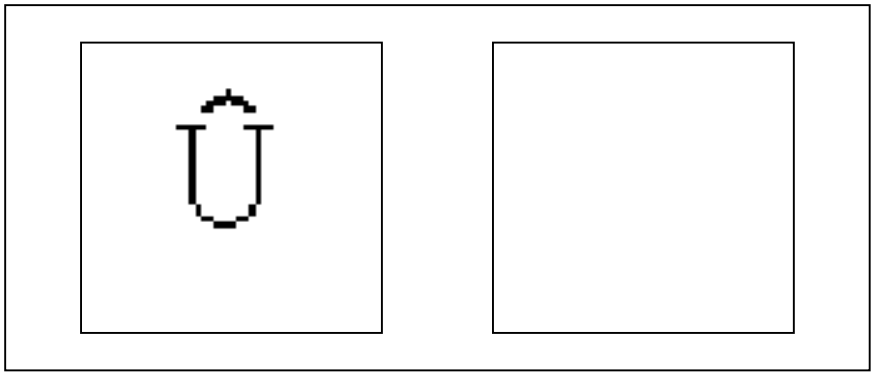


Figure 7.10 Imagen estado inicial y entrada

La imagen a procesar se introduce en el estado inicial, la imagen del estado no influye y podría ser cualquiera.

La condición de contorno se selecciona como refleja. Podría ser de tipo fijo, pero en ningún caso podría ser tipo toroide.

Los parámetros de simulación se toman como siguen:

- Variación a predecir: 0.1
- Variación límite: 0.15
- Las condiciones de parada:
- Variación de parada: 0.01
- Tiempo de parada: 200
- Paso de parada: 10000

En este punto estamos en condiciones de comenzar la simulación. Para ver las imágenes del transitorio de la salida accionamos el botón ver transitorio. Se observa una evolución de la salida, de la que a continuación se muestran algunas imágenes.

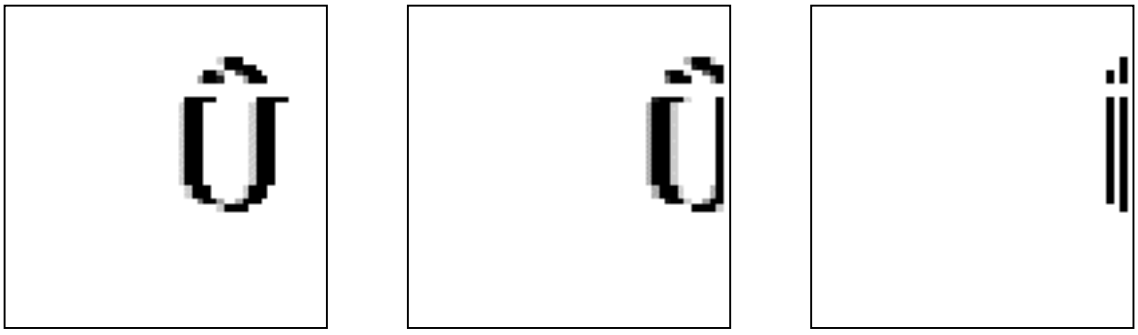


Figura 7.11 Evolución de la imagen de salida

Para la configuración de valores especificada anteriormente se tiene que son necesarios 1204 pasos de iteración, con un tiempo normalizado de 54.3. El valor del último paso es bastante menor que el valor total, de manera que en principio puede este valor ser aceptado como primera aproximación. Este tiempo depende del tamaño de la imagen, y de la distancia entre la/s figura/s y el margen hacia el que se propagan en el procesamiento.

A continuación representamos las gráficas de salida absoluta media y de variación de estado, que nos dan una idea de la evolución de la red.

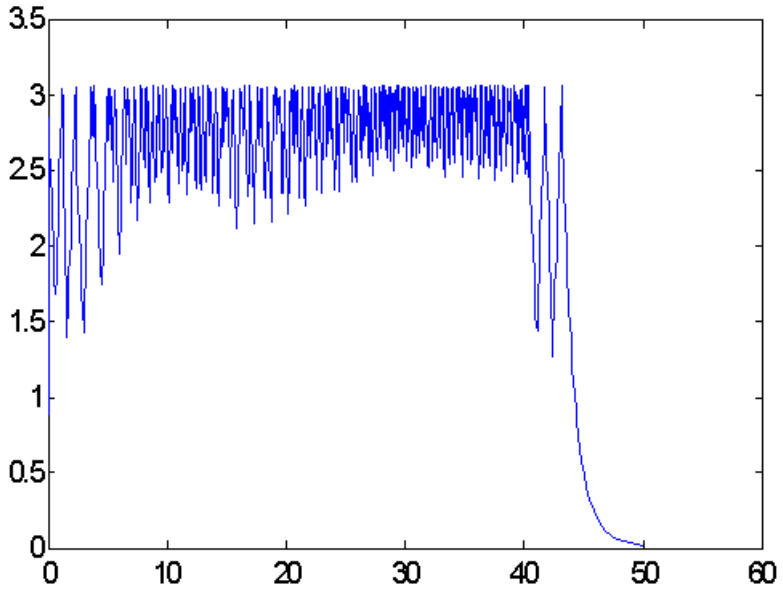


Figura 7.12 Representación gráfica de la evolución de la variación del estado.

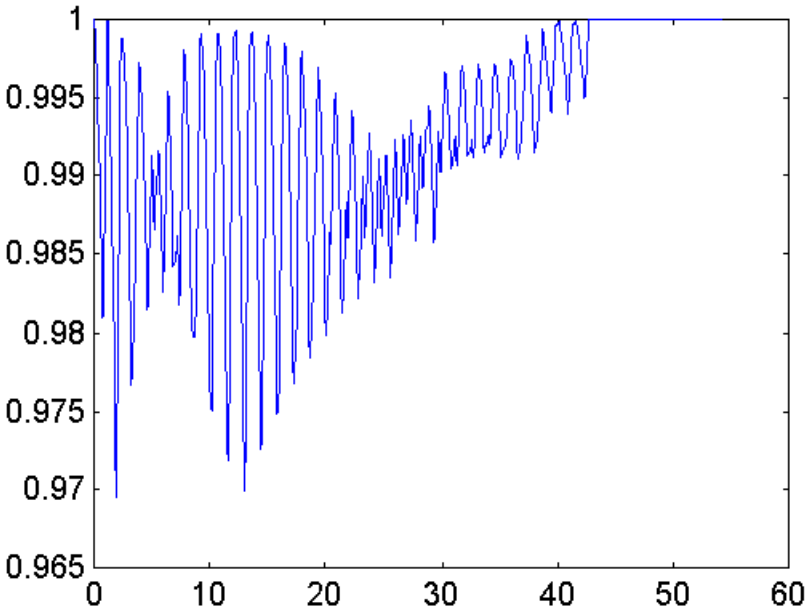


Figura 7.13 Representación gráfica de la salida media

Se observa una naturaleza oscilatoria en la dinámica de esta plantilla. De nuevo se puede ver que un criterio de parada basado en la salida no es aconsejable, pues podría dar lugar a falsa convergencia.

Observando la gráfica podemos determinar que el tiempo en el que se produce la convergencia es algo menor que la primera aproximación, y podemos estimar que la convergencia se produce en un tiempo normailzado entorno a 45, que pasado a tiempo real para un valor de $\tau = 250$ ns da un tiempo de procesamiento de 11.25 μs.

8 PROGRAMA *GA TEMPLATE*

8.1 DESCRIPCIÓN.

El programa realiza una búsqueda de la plantilla que más se aproxime a la operación especificada por las imágenes de entrada y salida deseada. Para ello utiliza la ya comentada técnica genética conocida como técnica de algoritmos genéticos (GA, de Genetic Algorithms).

La búsqueda queda definida con las imágenes correspondientes a la operación buscada y con los parámetros del GA y con los parámetros de la simulación de la CNN.

Las imágenes necesarias son 3:

- **Entrada**: Imagen de entrada de la CNN.
- **Estado Inicial**: Imagen que define el estado inicial de la CNN.
- **Deseada**: Imagen que define la imagen de salida deseada correspondiente a la operación buscada.

Estas imágenes definen un patrón de entrenamiento, y es muy importante que sean representativas de la mayor parte de casos posibles. Cuando se incluyen varias figuras en una imagen se ha de tener en cuenta que el área relativa de cada figura en la imagen deseada define la importancia relativa que queramos asignar a cada una, de manera que un modo razonable de proceder es asignar un área importante a las figuras más genéricas, mientras que a las figuras con peculiaridades se les asignaría un área menor.

Los parámetros propiamente del **GA** definibles por el usuario son los siguientes:

- **Tamaño de la población**: Número de individuos a considerar en cada iteración de la búsqueda.
- **Probabilidades de Mutación**: Se puede asignar probabilidades mayores que cero a varios tipos de mutación, que actuarían simultáneamente en la búsqueda. Se pueden configurar 4 tipos distintos de mutación: FlipBit, Uniforme, Boundary y Gaussiana.
- **Ruido gaussiano:** Solo tiene sentido cuando la mutación gaussiana actúa, y define la desviación típica del ruido que se aplica en caso de ocurrencia de la mutación
- **Tipo de Cruce**: Se consideran 4, One Point, Two Point, Arithmetic, Heuristic y Uniform.
- **Probabilidad de Cruce**: Determina la probabilidad de que dos padres seleccionados sean cruzados para formar un individuo de la nueva generación o no.
- **Factor de cruce aritmético**: Solo tiene sentido cuando el tipo de cruce es el aritmético.
- **Tipo de selección**: Define el método según el cual se seleccionan los individuos que dan lugar a la siguiente generación. Se puede elegir entre 4 métodos distintos de generación, TopPercent-A, TopPercent-B, Rankong y Tournament.

- **Parámetros de selección:**
  - **Percent:** Define el porcentaje para los métodos TopPercent A y B.
  - **ProbMax, ProbMin:** Definen la probabilidad máxima y mínima de los métodos Ranking y Tournament, que se asignarán respectivamente al mejor y peor individuo, asignándose para los intermedios la interpolación lineal entre estos dos valores en función de la posición ocupada en una ordenación de la población según el fitness.
  - **N:** Define el número de contendientes para el metodo Tournament.
- **Codificación de cromosoma:** Define el tipo de codificación a utilizar a la hora de convertir una determinada plantilla en una cadena de bits, y va a dar lugar a una mayor o menor dimensión en el espacio de búsqueda. Hay dos opciones, directa y simétrica.
- **Tipo de evaluación de imagen de salida:** Define el tipo de evaluación a aplicar en la imagen resultante de aplicar una plantilla correspondiente a un individuo, que dará lugar al término principal del fitness. Hay dos posibles, diferencia y correlación.
- **Ponderación de módulo:** Define el factor de ponderación del módulo, parámetro que representa la magnitud media de los parámetros de las plantillas.
- **Ponderación de tiempo:** Define el factor de ponderación del tiempo de simulación en unidades normalizadas. Recordar que este valor está relacionado con el tiempo real que correspondería a dicha plantilla al aplicarla en una CNN real, y no tiene relación directa con el tiempo de procesamiento en el ordenador.
- **Limite de tiempo:** Impone un límite en el tiempo permitido en la simulación. Es importante sobre todo en caso de utilizar codificación directa, pues en la búsqueda pueden aparecer plantillas poco estables o incluso inestables, que dan lugar a un tiempo de procesamiento excesivo, o ilimitado.

En cuanto a los parámetros correspondientes a la simulación de CNN son tres, y definirán criterios para la simulación de la CNN.

- **Variación a Predecir** y **Variación Permitida**: Definen el criterio a seguir en la variación del paso de integración. Cuanto mayor sean más rápida e imprecisa será la simulación, y viceversa. El valor de variación permitida deba ser algo mayor que el valor de variación predicha. En plantillas de procesado binario es posible acelerar enormemente la búsqueda mediante la elección de valores grandes. Se han probado con éxito valores entorno a 1.
- **Variación de Parada:** Define el criterio para considerar la convergencia de la red. Estima en cada iteración la pendiente de la máxima variación de estado y si está por debajo del valor definido para. Un valor que suele dar un buen resultado es 0.1. Para plantillas con evolución muy plana quizás habrá que bajarlo, pero hay que tener en cuenta que un valor demasiado bajo da lugar a un tiempo de simulación engañoso, pudiendo llegar a ser mucho mayor que el real.

Una vez configurara la búsqueda mediante la definición de los elementos anteriormente comentados, la parada de la búsqueda se puede definir por dos criterios distintos: **número de iteraciones** o por **valor de fitness del mejor individuo**. Se parará bien porque el número de iteraciones de la búsqueda sea igual al valor definido de parada, o bien porque el fitness del mejor individuo sea igual o mejor que el valor definido de parada.

8.2 IMPLEMENTACIÓN DEL SOFTWARE

A continuación se describe de forma esquemática la forma en que se ha implementado este programa multihilo. Lo primero que se hace al iniciar la búsqueda es configurarla con los parámetros definidos. A continuación se lanzará un hilo en el que se realizará la búsqueda, y se comunicará con el hilo principal para que el usuario pueda monitorizar el proceso. A continuación se muestra un esquema en el que se enseña básicamente como se realiza la búsqueda.

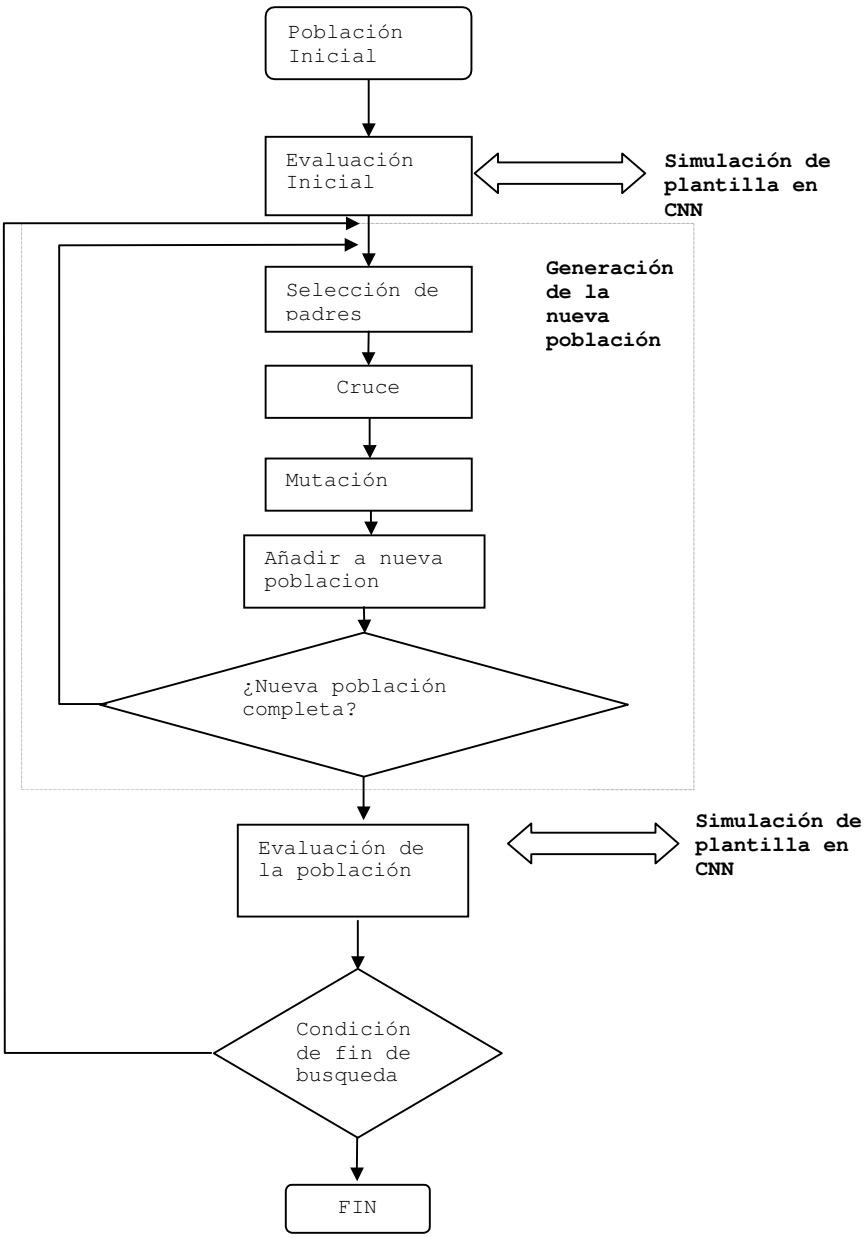


Figura 8.1. Esquema básico de funcionamiento

Como se puede observar en la evaluación de la población se debe realizar la simulación de la CNN con la plantilla correspondiente a cada individuo.

La búsqueda se realiza en un hilo independiente, que se comunica con el hilo principal para permitir que el usuario monitorice la búsqueda. En el siguiente esquema se muestra resumidamente la comunicación entre ambos hilos:

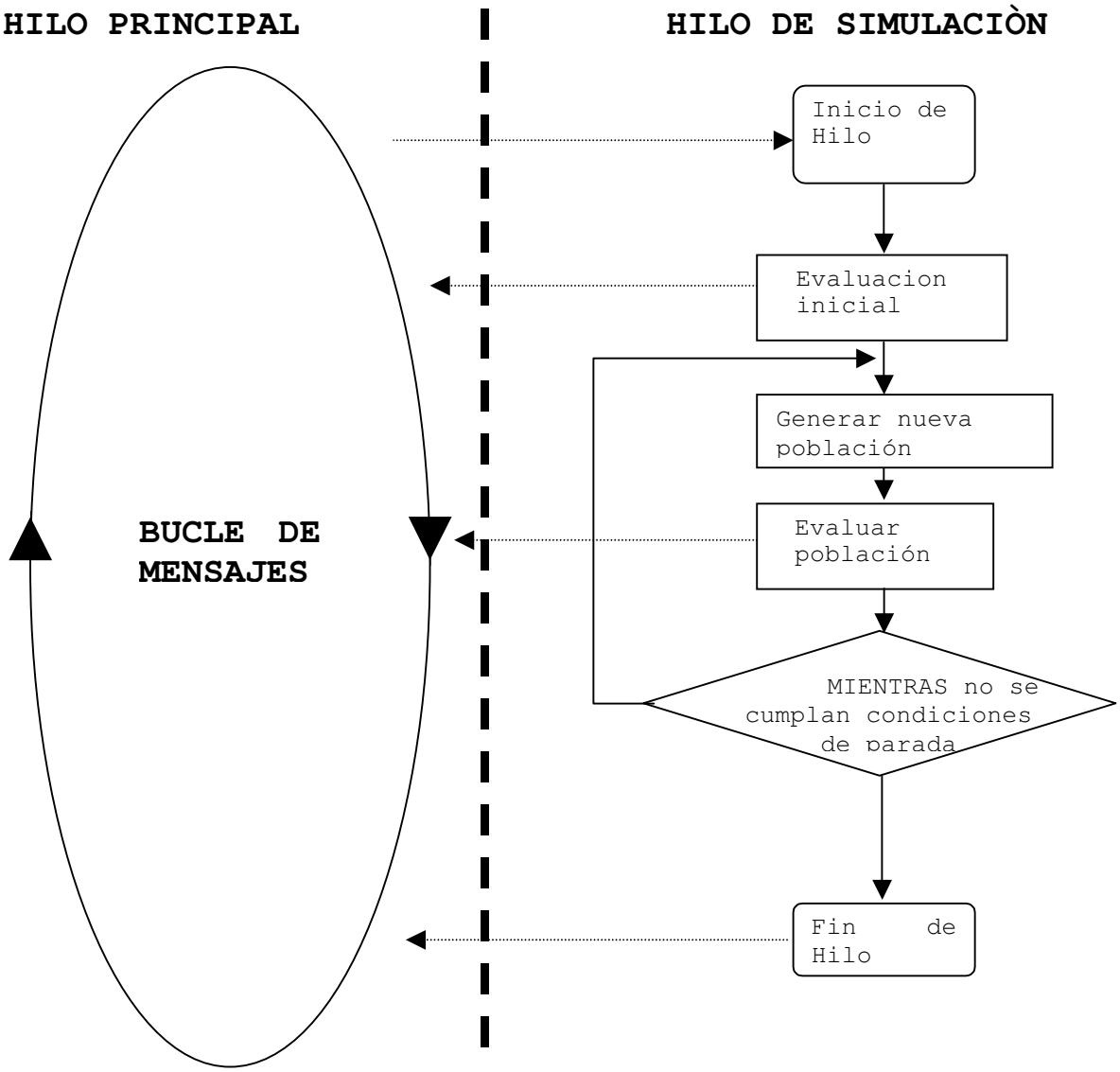


Figura 8.2 Comunicación entre hilos

El hilo principal lanza un hilo para realizar la búsqueda. En la evaluación de la población, tanto en la evaluación inicial como en las sucesivas evaluaciones, cada vez que se evalúa un individuo manda un mensaje al hilo principal, el cual recoge información acerca del fitness del individuo en cuestión. Además también le manda un mensaje en caso de que el individuo sea el mejor de los hasta ahora evaluados. En tal caso el hilo principal también recoge la imagen de salida correspondiente a este individuo, de manera que en todo momento el usuario monitorice el estado de la búsqueda.

Por último se comunica que ha concluido la búsqueda en caso de que se cumpla alguna de las condiciones de parada.

Todo el intercambio de información entre el hilo principal y el hilo secundario se realiza a través de una sección crítica, para mantener la coherencia en la información intercambiada.

Debido a la cantidad de tiempo que puede ser necesaria en una búsqueda determinada se ofrece al usuario la posibilidad de almacenar el estado en un fichero y para recuperarlo posteriormente. Es importante que se haya terminado una iteración completa y la búsqueda haya sido parada tanto para la carga como en la grabación del estado. En el archivo se guardan todos los parámetros de la búsqueda, el número de iteración y toda la población que hubiese en ese momento.

El fichero no está en modo texto, sino que se almacena directamente en el formato original de los datos, de manera que no se recomienda editar este fichero.

8.3 MANUAL DEL USUARIO

La ventana principal del programa es un panel de control en el que se muestra todo lo necesario para la configuración y la monitorización de la búsqueda. En la figura 8.3 se muestra esta ventana.

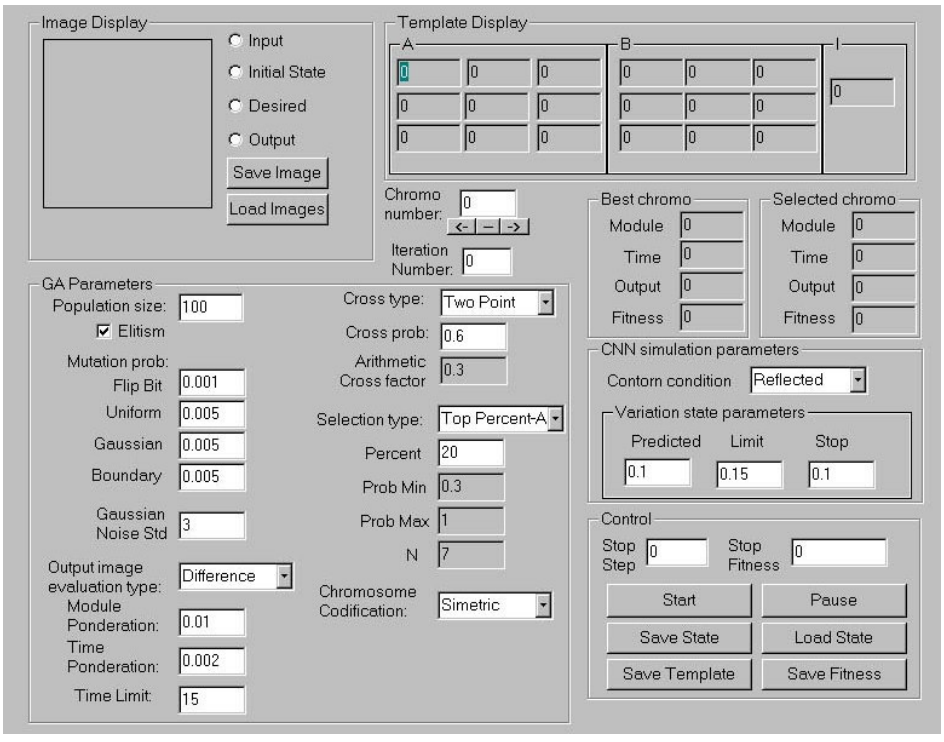


Figura 8.3 Ventana principal

Para configurar la búsqueda hemos de definir distintos elementos, que se puede dividir en imágenes, parámetros del GA y parámetros de simulación de la CNN. A continuación se comenta como se realizan.

- Configuración de las imágenes

Son 3 las imágenes que deben ser configuradas: Entrada, estado inicial y salida deseada. Para cargar cada una de estas imágenes se procede de forma muy parecida. Se acciona el botón de carga de imágenes (Botón *load image*).

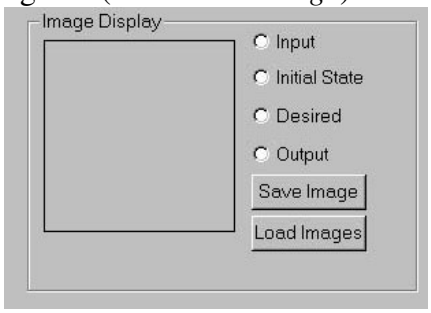


Figura 8.4 Controles para las imágenes

Una vez accionado el botón debemos seleccionar el tipo de imagen que deseamos configurar en el cuadro de diálogo que aparece a continuación:

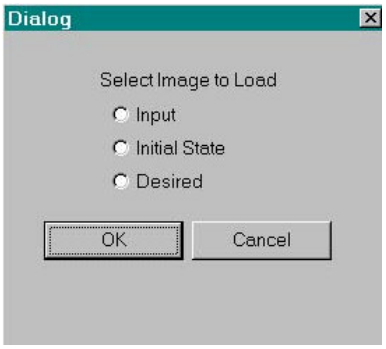


Figura 8.5 Dialogo de selección de imagen

Habrá que elegir entre imagen de entrada (*input*), imagen de estado inicial (*initial state*) o imagen deseada (*desired*).

Por último seleccionamos el fichero tipo BMP que contenga la imagen mediante en el dialogo de apertura de fichero típico.

Realizamos esto para las tres imágenes. Accionando el botón correspondiente podemos comprobar la correcta configuración de las tres imágenes, observando la imagen en el display de imagen (*Image display*).

En cuanto al botón guardar imagen (*Save image*) sirve para almacenar la imagen que se esté mostrando en el display, y su principal uso es almacenar resultados parciales.

- Configuración de los parámetros GA:

Se encuentran todos agrupados en una zona de la ventana. Se configuran sin más que escribir el valor que queramos en una caja de texto, o bien mediante la selección de un elemento de una lista:

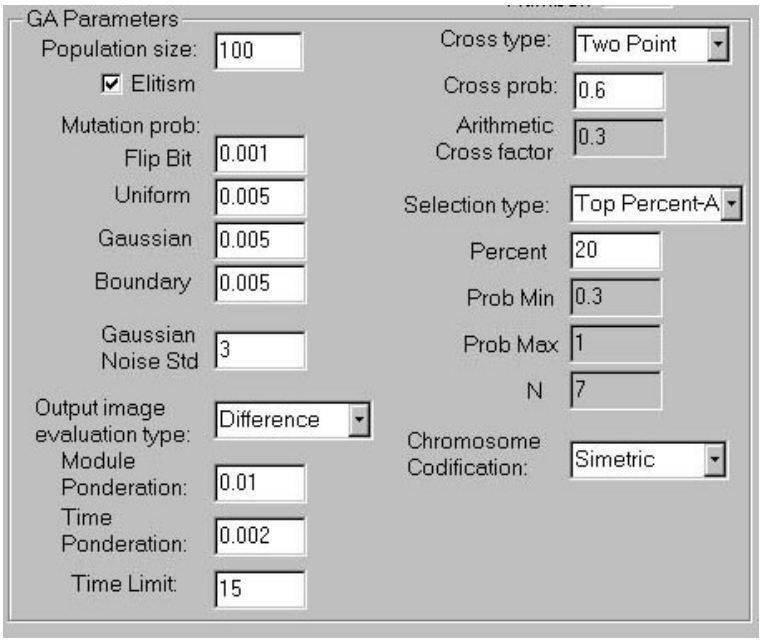


Imagen 8.6 Configuración de los parámetros GA

- Configuración de los parámetros de simulación de la CNN.

La condición de contorno define de que forma se va a tratar los elementos de los bordes en la red. Se selecciona en una lista una de las 3 posibles.

Los otros tres los parámetros sirven para establecer un criterio en la variación del paso de integración para realizar la simulación de la red de forma eficiente, y ya han sido comentados anteriormente. Se trata de variación a predecir, variación límite y variación de parada.

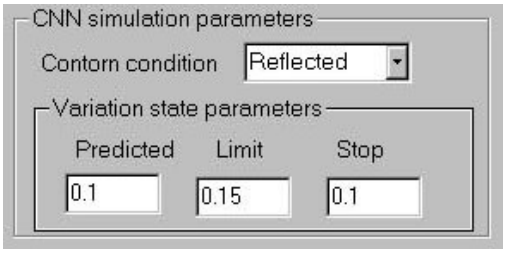


Imagen 8.7 Configuración de los parámetros de simulación CNN.

- Control de la búsqueda

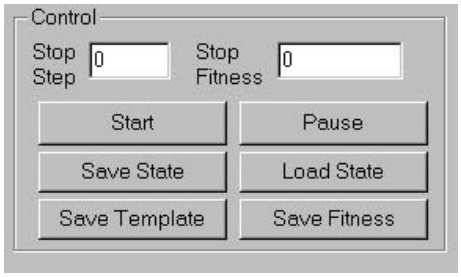


Figura 8.8 Control

Las condiciones de parada de la búsqueda son dos: número de iteraciones y fitness. El primero define el máximo número de iteraciones de la búsqueda, y el segundo hace que la búsqueda pare en caso de que en una iteración el mejor individuo tenga un fitness de valor menor o igual al definido.

Debido al gran coste computacional de la búsqueda puede resultar práctico dividirla en varias sesiones. Para ello se debe almacenar el estado accionando el botón correspondiente (*save state*), y cargarlo cuando se desee recuperarlo para continuar la búsqueda, accionando para ello el botón de carga de estado (*load state*). Tanto para cargar como para salvar el estado la búsqueda debe estar parada. Esto sucede al arrancar el programa, al finalizar la búsqueda y al accionar el botón de pausa y esperar el fin de la iteración.

El botón de arranque (*start*) puede ser accionado cuando la búsqueda esté parada y correctamente configurada, pudiendo servir de inicio de búsqueda o de continuación de una búsqueda interrumpida.

El botón de pausa (*pause*) sirve para parar la búsqueda, bien para almacenar el estado, bien para analizar resultados intermedios, o bien para finalizar la búsqueda.

Es posible almacenar la evolución del fitness a lo largo de las iteraciones de la búsqueda en un fichero de texto, tanto para el fitness del mejor como para el fitness medio de la población de cada iteración. Para ello debemos accionar el botón correspondiente (*save fitness*).A continuación se nos preguntará si queremos almacenar el fitness del mejor, y en caso de afirmación deberemos introducir un nombre para el fichero correspondiente. Después se pregunta si queremos almacenar el fitness medio, y de nuevo en caso afirmativo introducimos un nombre para el fichero.

En ambos casos se almacenará una secuencia de números en modo texto que indican los valores tomados por la variable en cuestión a lo largo de las iteraciones. Posteriormente podemos representar este fichero en una gráfica para el análisis de la evolución de la búsqueda, mediante la utilización de un programa que lo permita.

- Monitorización de la búsqueda y observación de la población.

Durante la búsqueda el programa muestra información de la misma al usuario. Hay dos cuadros de texto en los que se indica el número de iteración y él número de cromosoma que está siendo evaluado en ese momento.

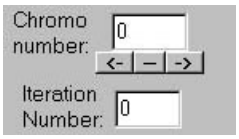


Figura 8.9 Monitorización de la situación de la búsqueda

Tambien se muestran los parámetros principales relecionados con el fitness (fitness total, término principal y términos secundarios), tanto para el individuo actual como para el mejor en la iteración.

Figura 8.10 Monitorización de individuos

Asimismo continuamente se muestra en el display de imagen el mejor individuo de la iteración y sus parámetros principales relativos a la calidad del mismo, de manera que de un vistazo observemos en que estado se encuentra la búsqueda.

Si paramos la búsqueda podemos recorrer la población mediante el uso de los botones y la caja de texto correspondientes, que es la misma caja de texto que se usaba en la monitorización:

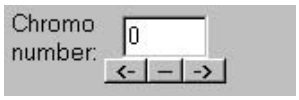


Figura 8.11 Selección de individuo

Para recorrer la población uno a uno se hace uso de los botones flecha, mientras que el botón del medio se utiliza para observar un individuo cualquiera seleccionado mediante la introducción del número de individuo en la caja de texto.

Cada vez que seleccionemos un individuo podremos ver su imagen de salida, sus plantilla correspondiente, sus parámetros principales. La población está ordenada en función del fitness, de manera que el número 0 será el mejor de la población y la mejor solución de la búsqueda en ese momento. Podemos almacenar en la salida de cualquier individuo sin más que seleccionarlo como se acaba de comentar, de manera que se muestre la imagen correspondiente en el display, y guardarla en un fichero bitmap accionando el botón salvar imagen (*save image*).

La observación de los parámetros del los individuos de la población es útil entre otras cosas para diagnosticar un fenómeno indeseable en la búsqueda, como es la convergencia prematura y la homogeneización de la población. Esto ocurrirá cuando se observen patrones excesivamente repetitivos en los parámetros de los individuos de la población. En tal caso la búsqueda carecerá de la potencia de los GA y será muy poco eficiente. Para corregir esto se debe disminuir la selectividad y/o aumentar la probabilidad de mutación.

8.4 EJEMPLOS DE APLICACIÓN.

8.4.1 DETECTOR DE BORDES

Se trata de encontrar la plantilla que realiza un procesamiento de detector de bordes. Para configurar la búsqueda de la plantilla correspondiente al detector de bordes se introducen las siguientes imágenes de entrenamiento, correspondientes a la entrada, el estado inicial y la salida deseada respectivamente:

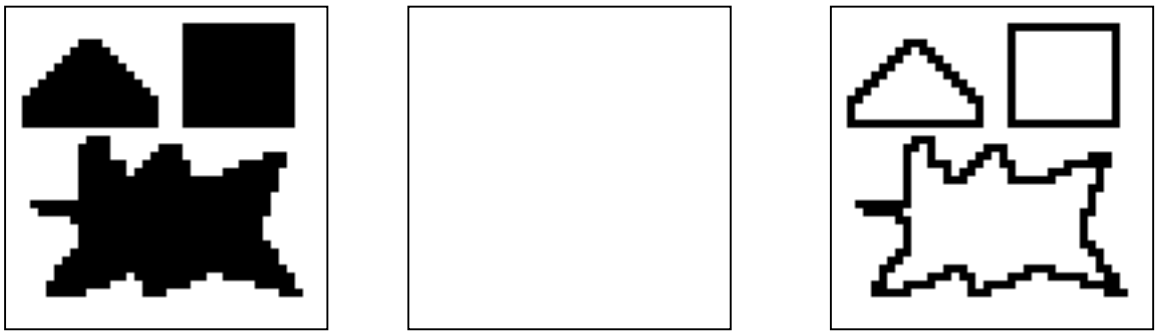


Figura 8.12 Imágenes de entrenamiento

Los restantes parámetros de búsqueda son los siguientes (los parámetros que no se comentan no influyen o están a cero):

- Tamaño de la población: 200
- Probabilidad de Mutación:
  - FlipBit: 0.0015
  - Boundary: 0.005
- Tipo de cruce: Two Point
- Probabilidad de cruce: 0.6
- Tipo de Selección: Ranking.
- Parámetros de selección:
  - Pmin: 0.55
  - Pmax: 1
- Codificación de cromosoma: Simétrica
- Tipo de evaluación de salida: Diferencia
- Ponderación de módulo: NO
- Ponderación de tiempo: 0.0001
- Límite de tiempo: 50

- Condición de contorno: Reflejada
- Parámetros de variación de estado:
  - Predicha:1
  - Limite: 1.5
  - Parada: 0.1
- Condiciones de parada:
  - Paso de parada: 1000
  - Fitness de parada: 0

Una vez configurada completamente la búsqueda con los anteriores parámetros se inicia la búsqueda. En las siguientes gráficas se observa la evolución del fitness medio de la población y del fitness del mejor individuo:

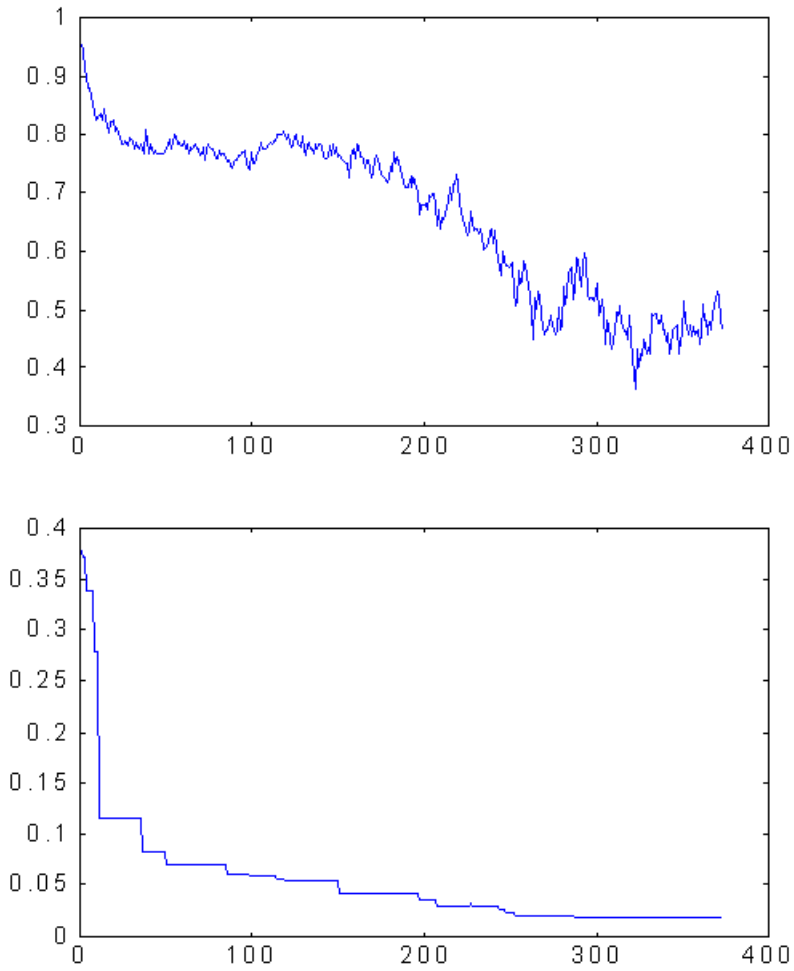


Figura 8.13 Evolución del fitness medio y el fitness mínimo.

Entorno a la iteración 200 se alcanza el óptimo en la imagen de salida, siendo la evolución de fitness que resta debido al término secundario ponderador del tiempo. El resto de las iteraciones la búsqueda intentará optimizar este término secundario. Terminamos la búsqueda cuando se observa que no encuentra mejores individuos y el término secundario intenta afinar un factor de tiempo que es muy bueno (mejor que el de la plantilla de la biblioteca).

A continuación se muestra distintas salidas intermedias de mejores individuos en distintas iteraciones:

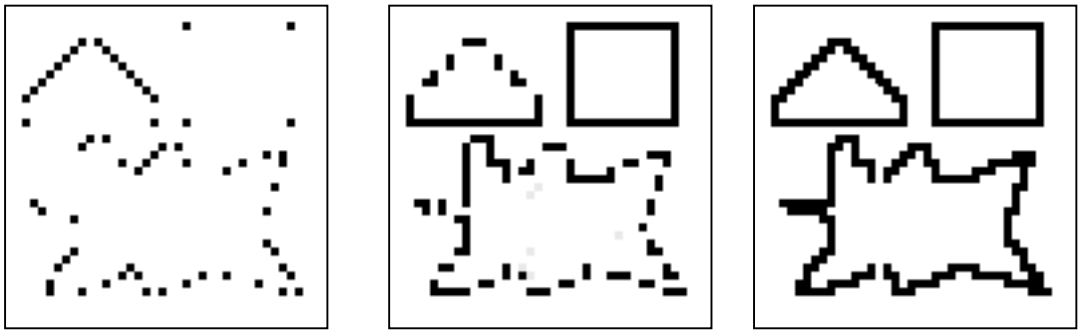


Figura 8.14 Salidas intermedias en la búsqueda

Es importante que se observe continuidad de fitness, lo cual se comprueba en la evolución de las soluciones. Debemos ver que los individuos van refinando poco a poco la solución deseada. Debemos procurar mediante la elección de las imágenes que las soluciones que no nos interesan no sean ponderaras demasiado bien. Una solución que se produce con mucha frecuencia es aquella que muestra directamente la imagen de entrada, y debemos procurar que no se le corresponda un fitness demasiado bueno. Un error que se podría cometer en este ejemplo podría ser introducir figuras con huecos grandes, lo cual provocaría el efecto anteriormente comentado. Para este ejemplo se deben utilizar figuras sin huecos, o con huecos muy pequeños en comparación con el tamaño de la figura.

La plantilla obtenida en la búsqueda tiene los siguientes parámetros:

$$A = \begin{pmatrix} -0.2 & -0.066667 & -0.2 \\ -0.066667 & 6.66667 & -0.066667 \\ -0.2 & -0.066667 & -0.2 \end{pmatrix}$$

$$B = \begin{pmatrix} -1.133333 & -0.800000 & -1.133333 \\ -0.800000 & 10.800000 & -0.800000 \\ -1.133333 & -0.800000 & -1.133333 \end{pmatrix}$$

$$I = 0.533333$$

Si observamos los valores obtenidos vemos que la forma de la matriz B es la típica de un filtro paso de alta, como corresponde a un detector de bordes. Por otra parte si nos fijamos en la matriz A vemos que los el término principal es el predominante,

siendo el resto despreciables frente a este. El hecho de que el procesamiento sea binario da un cierto margen de valores en los que se pueden mover los términos de la matriz sin que varíe el comportamiento, de ahí que los valores que se observan pueden parecer arbitrarios.

En este punto no se puede asegurar que la plantilla que tenemos es la que buscábamos. Debemos validarla con otro ejemplo, para lo cual utilizamos el programa *Template Simulator*. Usamos una imagen arbitraria que sea suficientemente representativa. Las imágenes entrada y estado inicial son las siguientes:

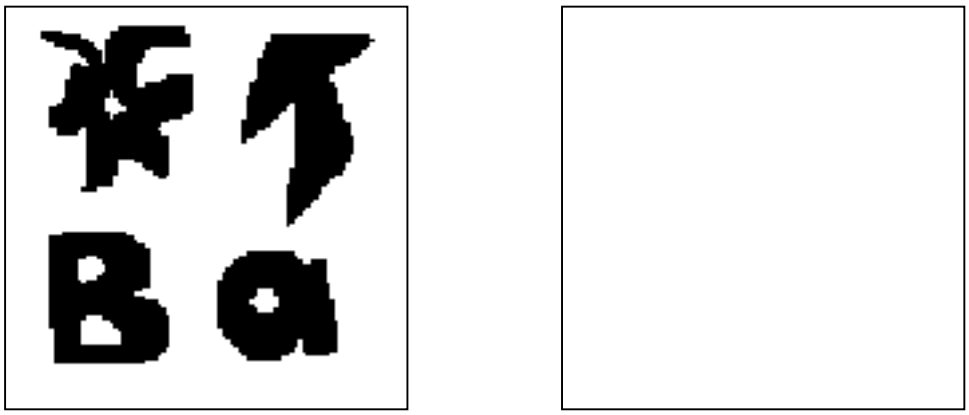


Figura 8.15 Imagenes de entrada y estado inicial para validación

Para introducir la plantilla lo más sencillo es almacenarla en un fichero del programa *GA Template* y cargarla posteriormente en el *Template Simulator*.

Procesando con la plantilla encontrada por el GA se obtiene el siguiente resultado:

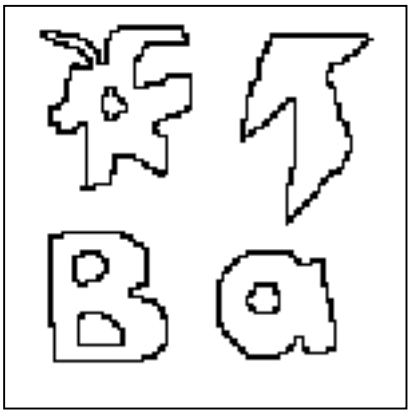


Figura 8.16 Imagen de salida para validación

Con lo cual validamos la plantilla anterior como buen detector de bordes. No hay diferencia en formato de fichero entre esta plantilla y las plantillas de la biblioteca, de manera que podemos guardarla para un posterior uso.

8.4.2 DETECTOR DE COMPONENTES HORIZONTALES CONECTADAS.

El objetivo de la búsqueda GA es encontrar la plantilla que muestra el número de componentes conectadas horizontalmente. Se trata de una plantilla bastante útil en aplicaciones de reconocimiento de caracteres, entre otras. Para configurar la búsqueda de la plantilla correspondiente al detector de componentes horizontales conectadas se introducen las siguientes imágenes de entrenamiento:

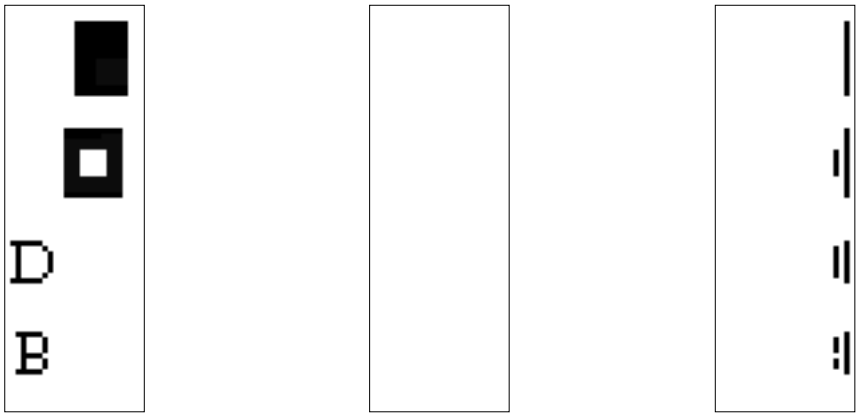


Figura 8.17 Imágenes de entrada, estado inicial y salida deseada para entrenamiento

Los restantes parámetros de búsqueda son los siguientes (los parámetros que no se comentan no influyen o están a cero):

- Tamaño de la población:150
- Probabilidad de Mutación:
  - FlipBit: 0.0015
- Tipo de cruce: Two Point
- Probabilidad de cruce: 0.6
- Tipo de Selección: Ranking.
- Parámetros de selección:
  - Pmin:0.55
  - Pmax:1
- Codificación de cromosoma: Directa
  (No se puede utilizar la codificación simétrica, precisamente este procesamiento se caracteriza por su asimetría)
- Tipo de evaluación de salida: Diferencia
- Ponderación de módulo: NO

- Ponderación de tiempo:0.00001
- Límite de tiempo: 400
  (Debe ser de este orden, pues se trata de una plantilla de transmisión, que se caracteriza por una convergencia más lenta)
- Condición de contorno: Reflejada
  (Valdría la fija, pero en ningún caso se podría utilizar la toroide, pues en con este tipo de plantilla da lugar a un procesamiento inestable)
- Parámetros de variación de estado:
  - Predicha:1
  - Limite:1.5

  (El procesamiento con imágenes binarias puede ser acelerado enormemente con altos valores en los dos parámetros anteriores)
- Parada:0.000001
  (Este tipo de plantilla tiene grandes oscilaciones en la evolución del estado, si elegimos un valor de parada mayor al anterior tendremos falsas convergencias en las simulaciones de las plantillas, y no se llegará a nada en la búsqueda)
- Condiciones de parada:
  - Paso de parada:1000
  - Fitness de parada:0
    (Inalcanzable debido al término secundario)

Una vez configurada completamente la búsqueda con los anteriores parámetros se inicia la búsqueda. En las siguientes gráficas se observa la evolución del fitness medio de la población y del fitness del mejor individuo:

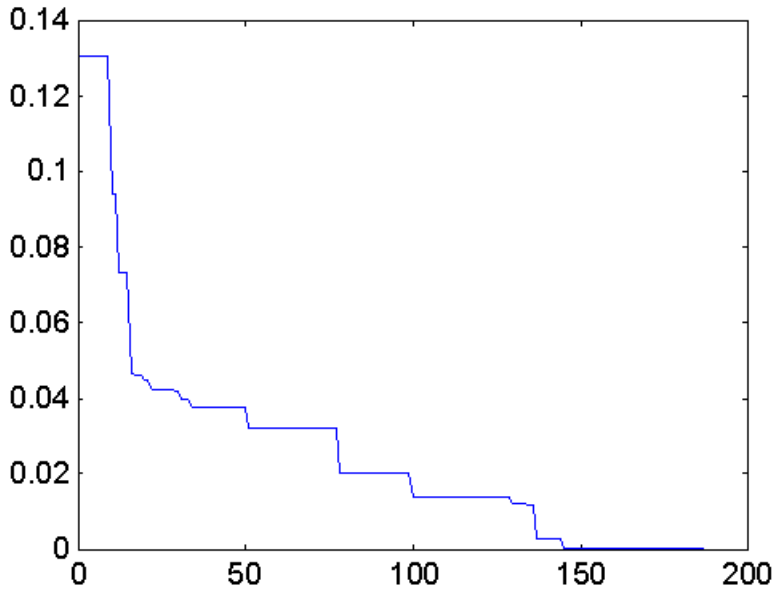


Figura 8.18 Evolución del fitness mínimo.

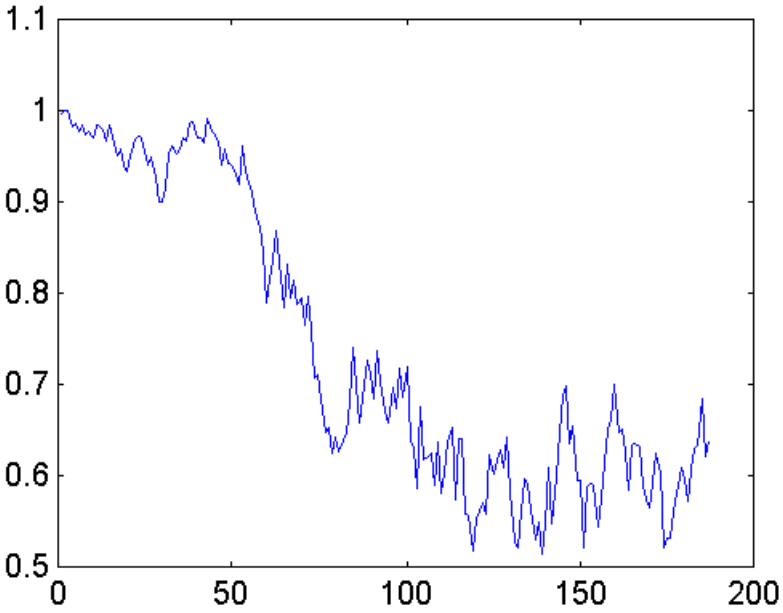


Figura 8.19 Evolución del fitness medio

En a la iteración 145 se alcanza el óptimo en la imagen de salida, siendo el valor de fitness que resta debido al término secundario ponderador del tiempo. El resto de las iteraciones la búsqueda intentará optimizar este término secundario. Detenemos la iteración cuando observamos que la búsqueda deja de evolucionar y se observa un tiempo de procesamiento de la mejor plantilla más que satisfactorio.

A continuación de muestra distintas salidas intermedias de mejores individuos en distintas iteraciones:

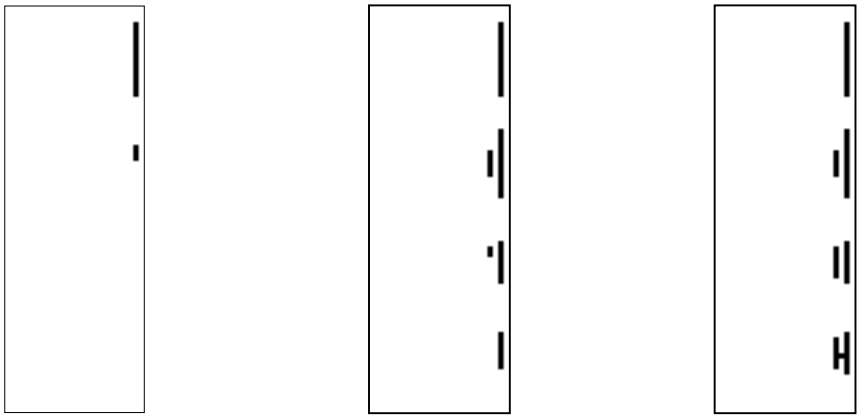


Figura 8.20 Salidas intermedias en la búsqueda

Una dificultad importante de esta plantilla es utilizar imágenes adecuadas, más aún que en otros ejemplos, debido a la naturaleza de este procesamiento. En la realización de este ejemplo se he tenido que modificar varias veces las imágenes de manera que se oriente adecuadamente la búsqueda GA. Si se observa detenidamente el área asignada a cada figura de la entrada se premia mucho a las primeras

aproximaciones a la solución deseada correspondientes a las figuras más simples, de manera que se distingan del resto, y después va premiando cada avance de la búsqueda en figuras más complejas en menor medida.

En este y en cualquier ejemplo se debe tener en cuenta que los primeros individuos que muestren características apropiadas van a ser aproximaciones muy groseras a lo que estamos buscando, de manera que debemos premiar a estas primeras aproximaciones sobre el resto de individuos, y procurar que cada avance se vaya premiando adecuadamente. Esto no es más que lo que ya se comentó como una deseable característica del fitness, la continuidad.

La observación de las imágenes intermedias permite diagnosticar la correcta asignación de fitness, es decir, que se valoren las plantillas que realmente reflejan el procesamiento deseado.

La plantilla obtenida en la búsqueda tiene los siguientes parámetros:

$$A = \begin{pmatrix} 0.26667 & 2.00000 & -2.93333 \\ 8.20000 & 7.53333 & -4.40000 \\ 1.73333 & 1.60000 & -1.46667 \end{pmatrix}$$

$$B = \begin{pmatrix} 3.80000 & 8.26667 & 7.00000 \\ -3.06667 & 10.66667 & -2.53333 \\ -4.53333 & -4.73333 & -2.33333 \end{pmatrix}$$

$$I = 11.53333$$

Se observa una estructura antisimétrica horizontal, como cabría esperar dada la propagación que se realiza hacia la derecha. Además se comprueba que los términos de la fila central tienen mayores magnitudes, de manera que dominan las otras dos filas, como era deseable.

Puede parecer que la matriz B toma unos valores muy extraños, pero hay que tener en cuenta que actúa sobre una imagen plana, de manera que interviene como un término constante en cada celda.

En este punto no se puede asegurar que la plantilla que tenemos es la que buscábamos. Debemos validarla con otro ejemplo, para lo cual utilizamos el programa *Template Simulator*. Usamos una imagen arbitraria que sea suficientemente representativa. Usamos las siguientes imágenes como entrada y como estado inicial:

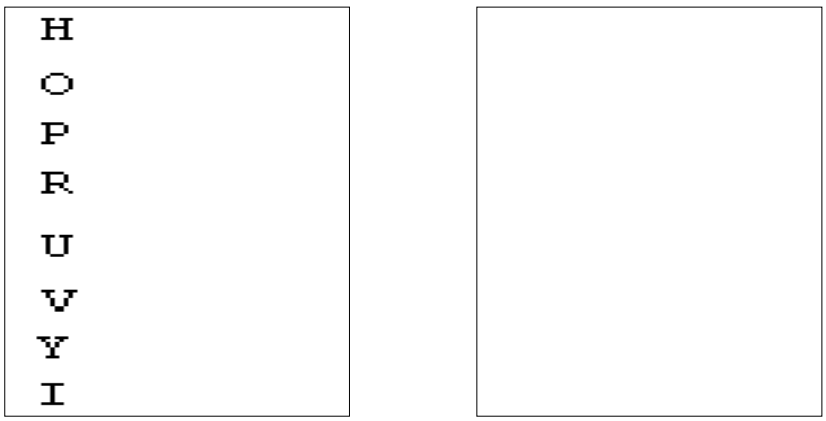


Figura 8.21 Imágenes de entrada y estado inicial para validación

Para introducir la plantilla lo más sencillo es almacenarla en un fichero del programa *GA Template*, y cargarla posteriormente en el *Template Simulator*.

Procesando con la plantilla encontrada por el GA se obtiene el siguiente resultado:

Figura 8.22 Imagen de salida para validación

Con lo cual validamos la plantilla anterior. No hay diferencia en formato de fichero entre esta plantilla y las plantillas de la biblioteca, de manera que podemos guardarla para un posterior uso.

8.4.3 LÓGICA AND.

En este ejemplo se trata de encontrar una de las plantillas básicas para la CNN. Es una plantilla muy sencilla de determinar analíticamente, pero como ejemplo de la capacidad de búsqueda del GA la vamos a utilizar. Hace una función lógica AND píxel a píxel, entendiendo el pixel blanco como '0' y el píxel negro como '1'. Se utiliza las siguiente imagen para orientar la búsqueda:

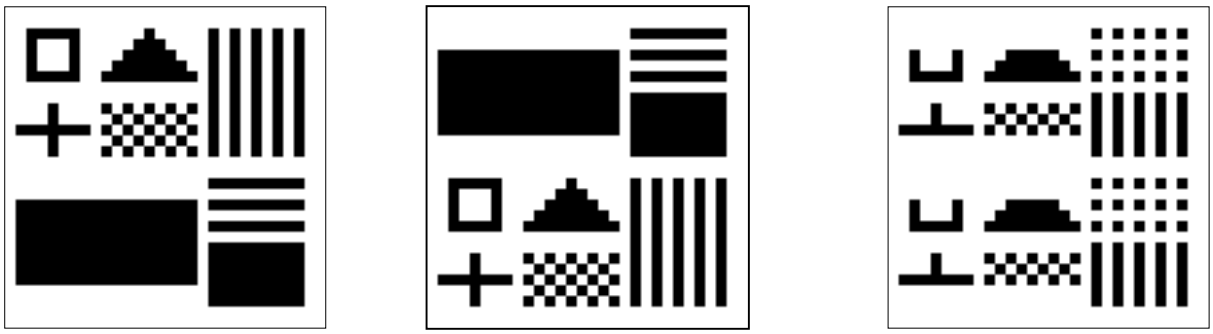


Figura 8.23 Imágenes de entrada, estado inicial y salida deseada para entrenamiento

Los restantes parámetros de búsqueda son los siguientes (los parámetros que no se comentan no influyen o están a cero):

- Tamaño de la población:150
- Probabilidad de Mutación:
  - FlipBit: 0.002
- Tipo de cruce: Two Point
- Probabilidad de cruce: 0.6
- Tipo de Selección: Ranking.
- Parámetros de selección:
  - Pmin:0.55
  - Pmax:1
- Codificación de cromosoma: Simétrica
- Tipo de evaluación de salida: Diferencia
- Ponderación de módulo: NO
- Ponderación de tiempo:0.00001
- Límite de tiempo: 20
  (Este tipo de plantillas es muy rápida)
- Condición de contorno: Reflejada
  (Se podría utilizar cualquiera de las 3)
- Parámetros de variación de estado:
  - Predicha:0.1
  - Limite:0.15
  - Parada:0.1
- Condiciones de parada:
  - Paso de parada:1000
  - Fitness de parada:0
    (Inalcanzable debido al término secundario)

Una vez configurada completamente la búsqueda con los anteriores parámetros se inicia la búsqueda. En las siguientes gráficas se observa la evolución del fitness medio de la población y del fitness del mejor individuo:

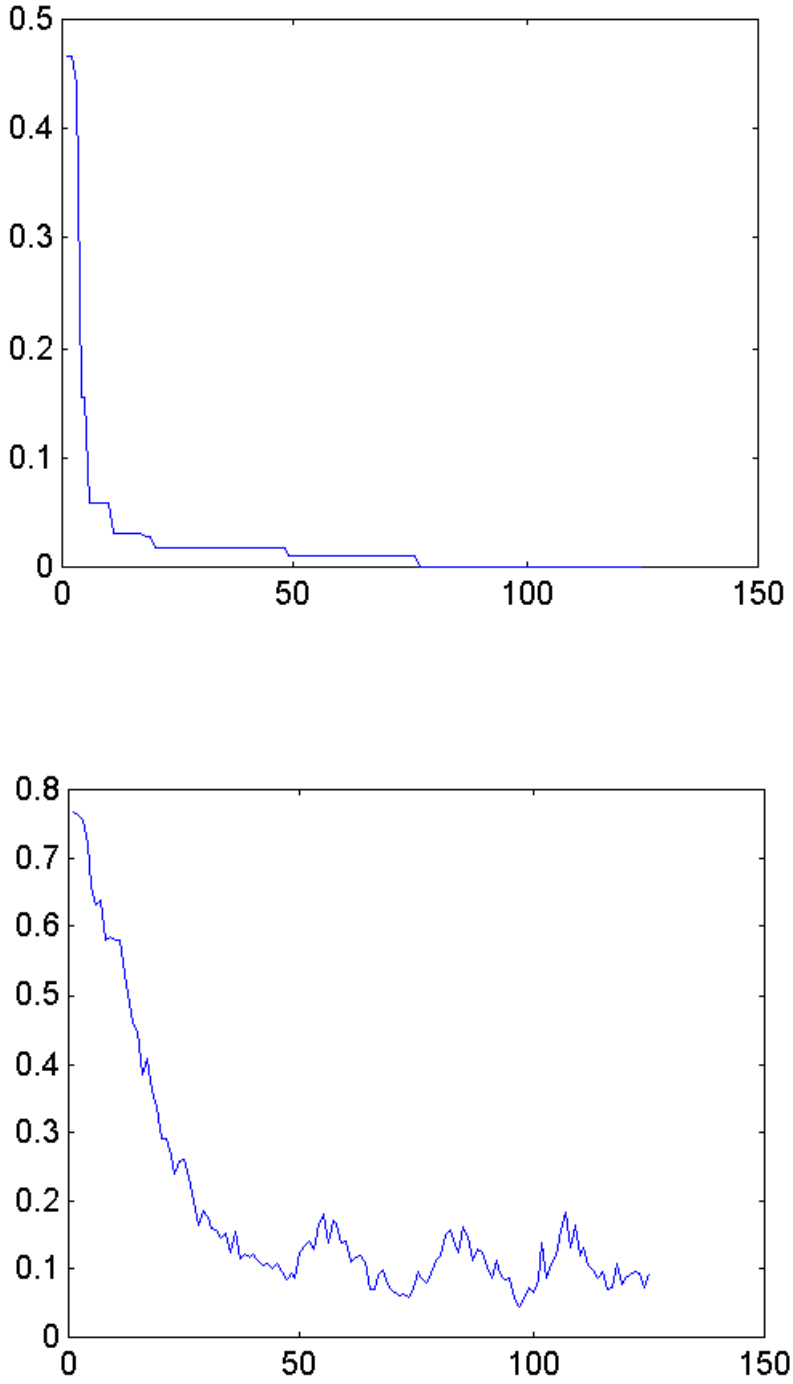


Figura 8.24 Evolución del fitness mínimo y el fitness medio

En a la iteración 68 se alcanza el óptimo en la imagen de salida, siendo el valor de fitness que resta debido al término secundario ponderador del tiempo. El resto de las iteraciones la búsqueda intentará optimizar este término secundario. Detenemos la iteración cuando observamos que la búsqueda deja de evolucionar y se observa un tiempo de procesamiento correcto.

A continuación de muestra distintas salidas intermedias de mejores individuos en distintas iteraciones:

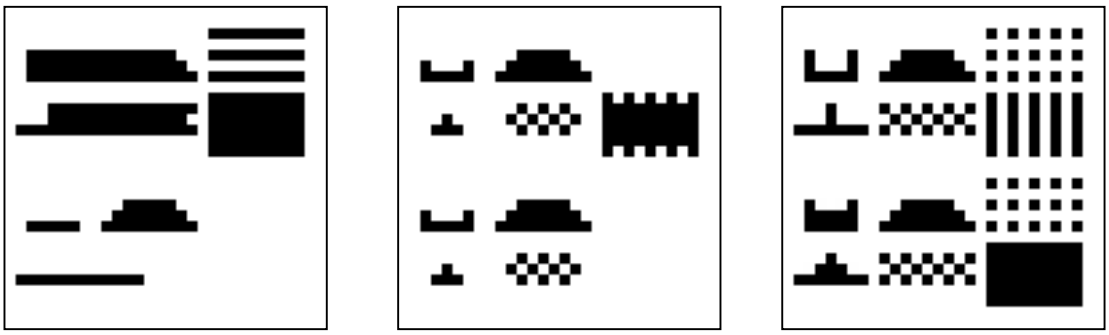


Figura 8.25 Salidas intermedias en la búsqueda

La plantilla obtenida en la búsqueda tiene los siguientes parámetros:

$$A = \begin{pmatrix} -0.20000 & 0.53333 & -0.20000 \\ 0.53333 & 6.93333 & 0.53333 \\ -0.20000 & 0.53333 & -0.20000 \end{pmatrix}$$

$$B = \begin{pmatrix} 0.06667 & -0.80000 & 0.06667 \\ -0.80000 & 8.73333 & 0.80000 \\ 0.06667 & -0.80000 & 0.06667 \end{pmatrix}$$

$$I = -5.00000$$

Si observamos el resultado obtenido vemos que los elementos de vecindad van ponderados por factores despreciables comparados con el término principal. Además se observa una estructura parecida a la plantilla de la biblioteca.

En este punto no se puede asegurar que la plantilla que tenemos es la que buscábamos. Debemos validarla con otro ejemplo, para lo cual utilizamos el programa *Template Simulator*. Usamos una imagen arbitraria que sea suficientemente representativa. Usamos las siguientes imágenes como entrada y como estado inicial:

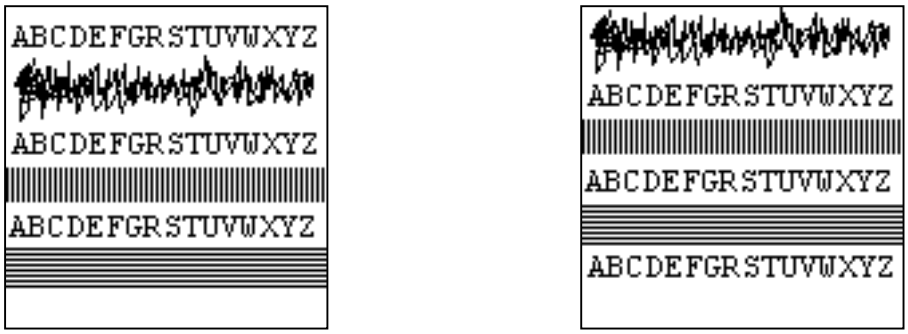


Figura 8.26 Imágenes de entrada y estado inicial para validación.

Para introducir la plantilla lo más sencillo es almacenarla en un fichero del programa *GA Template*, y cargarla posteriormente en el *Template Simulator*.

Procesando con la plantilla encontrada por el GA se obtiene el siguiente resultado:

Figura 8.26 Imagen de salida resultante en la validación

9 PROGRAMA *ALGORITHM SIMULATOR*

9.1 DESCRIPCIÓN.

El programa *Algorithm Simulator* implementa un algoritmo (al que también nos referiremos como programa) de procesamiento de imágenes simulando un sistema basado en CNN. A diferencia del programa *Template Simulator* el procesamiento que se realiza es un procesamiento multiplantilla, una combinación de procesados de imágenes en una estructura de árbol, que representa un algoritmo, y que ya ha sido comentado anteriormente. La utilidad del programa puede ser verificar la validez de un algoritmo obtenido en el programa *GP Algorithm*, la afinación del mismo modificando el algoritmo para optimizar el tiempo de procesamiento, el diseño directo de un algoritmo mediante prueba y error, o simplemente comprobar la capacidad de procesamiento del sistema en un ejemplo determinado.

A continuación se comentan los parámetros configurables por el usuario que determinan la simulación.

- Imágenes de entrada: constituyen las imágenes base del algoritmo para el procesamiento del algoritmo.
- Configuración de funciones: Se especifica el conjunto de funciones base que se aplican en el algoritmo, correspondiendo a cada función una determinada plantilla. Se determina mediante el fichero de configuración de funciones, que contiene el nombre de los ficheros que contienen las plantillas en cuestión. Consultar apartado 6.4 para más información sobre el formato de este y otros ficheros utilizados en el programa.
- Parámetros de simulación de la CNN: Establecen un criterio en la variación del paso de integración y en la convergencia, basado en la variación del estado en las células, que ya ha sido explicado anteriormente. Como siempre son 3:
  - Variación predicha y variación límite. Cuanto mayores sean estos valores mas rápida y ruda será la simulación, y al revés. Se aconseja hacer grandes en plantillas binarias y en las plantillas de procesamiento local a cada celda. Recordar que se debe hacer la *variación límite* algo menor que la *variación predicha*.
  - Variación de parada. Constituye el criterio de convergencia como ya se ha dicho, un valor relativamente grande puede dar como resultado un convergencia prematura, y un valor relativamente pequeño puede dar un criterio de convergencia demasiado restrictiva, y tiempos de procesamiento normalizados 'falsos', mucho mayores que los reales.

En caso de que se edite el programa desde cero será necesaria la configuración de todos los parámetros anteriores. En caso de partir de un programa almacenado en un fichero la configuración será opcional para el caso en que el usuario desee modificar algún aspecto del programa cargado.

En cuanto a los resultados mostrados en el programa en el procesamiento, se muestran los tiempos de procesamiento relativos a cada función, así como el tiempo

total del algoritmo, por supuesto tiempos normalizados de la red neuronal. Tambien se muestra la imagen resultante del procesamiento.

Si se considera oportuno se pueden almacenar algunos resultados en ficheros para posterior análisis o utilización. Es posible almecenar los tiempos de procesamiento anteriormente comentados en un fichero. También se puede almacenar la imagen resultante del procesamiento en un fichero BMP. Por último también se puede almacenar el propio algoritmo de procesamiento.

9.2 IMPLEMENTACIÓN DEL SOFTWARE

De nuevo la programación se ha realizado en una estructura multihilo. Para un algoritmo relativamente simple no sería necesario, pero para mayor complejidad no sería adecuada una programación en un solo hilo, pues el programa se quedaría bloqueado durante un tiempo sin mostrar ningún tipo de información, y sin posibilidad siquiera de minimizar o cerrar la ventana de la aplicación. La comunicación del programa con el usuario durante el procesamiento del algoritmo consiste en mostrar cada vez que se completa una función del algoritmo el tiempo normalizado de procesamiento empleado en dicha función, y finalmente se muestra el tiempo normalizado de procesamiento global del algoritmo, que no es más que la suma de los tiempos empleados por cada función.

El hilo principal, como siempre, es el bucle de mensajes, que como ya se ha dicho, se ocupa básicamente de la interfaz con el usuario. El hilo secundario es el que realiza el procesamiento del algoritmo codificado en un árbol. La comunicación de ambos hilos consiste en informar desde el hilo secundario que se ha realizado el procesamiento correspondiente a una función, y que se ha terminado de procesar el árbol. El paso de información, que consiste en tiempos de procesamiento y el correspondiente número de función se realiza a través de una cola, en el que lógicamente el hilo secundario de ocupa de introducir elementos, y el hilo principal de sacarlos cuado sea informado por el hilo secundario de que hay un nuevo elemento, que es lo equivalente al procesamiento de una nueva función. En la figura 9.1 se muestra un esquema de la comunicación entre los hilos en el que se pone de manifiesto lo comentado anteriormente.

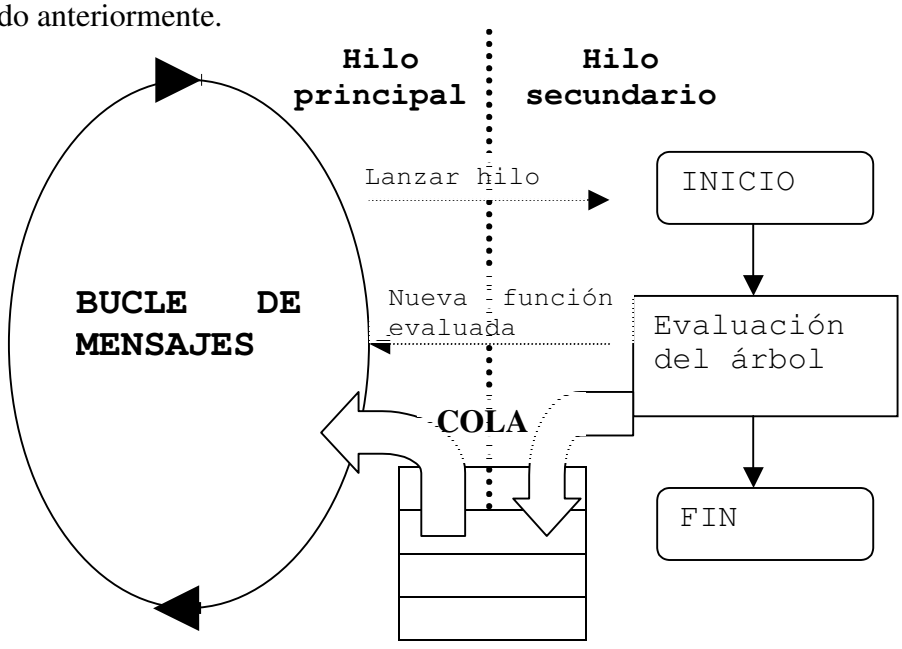


Figura 9.1. Diagrama de la comunicación entre hilos

9.3 MANUAL DE USUARIO

Una vez que se ejecuta el programa se muestra la ventana principal, donde se muestran directamente todos los controles y salidas de programa. En la figura 9.1 se puede ver esta ventana.

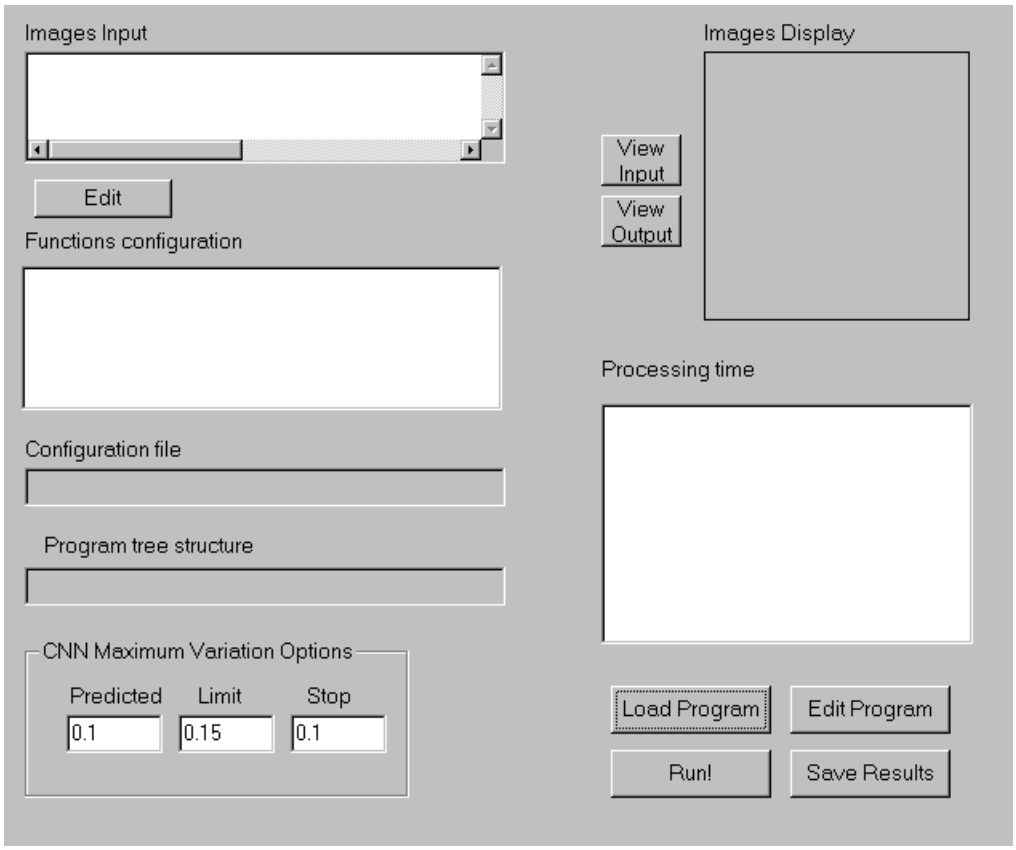


Figura 9.1 Ventana principal del programa

A continuación se comentan uno por uno los elementos que se encuentran en la ventana.

Botón *Load Program*: Sirve para cargar un archivo desde un fichero que haya sido almacenado anteriormente.

Botón *Edit Program*: Sirve para editar un programa, de manera que podemos editar un algoritmo desde cero, o bien podemos modificar algún elemento del algoritmo que en ese momento tengamos cargado. Una vez accionado el botón preguntará qué elemento queremos editar. Primero preguntará si queremos modificar el set de funciones (o se exigirá en caso de que no haya ninguno cargado), y en caso afirmativo se requerirá la introducción del fichero contenedor del set de funciones, que como ya se ha dicho no es más que un fichero de texto que contiene el nombre de los ficheros de plantilla, con su ruta completa, que entran en juego en el algoritmo. Ver apartado 6.3 para más detalle sobre los tipos de fichero utilidados en los distintos programas del paquete.

A continuación se abre el cuadro de diálogo para intoducir imágenes, que son las imágenes base del algoritmo. Se irá preguntando de una en una si se desea añadir una entrada, y tras cada respuesta se debe indicar el nombre del fichero BMP que contiene la imagen.

Por último se pregunta si se desea editar el programa en cuestión, tras cuya respuesta afirmativa se abre una ventana para la edición del programa, como la que se

muestra en la figura 9.2. En la misma escribimos directamente el algoritmo, mediante la notación prefijo del árbol que representa a dicho algoritmo. Cada nodo se codifica de la siguiente forma; en caso de que se trate de un terminal se debe poner el carácter 't' más el número de terminal de qué se trate (recordar que la indexación es basada en 0), y en caso de que se trate de una función se pondría el carácter 'f' más el número de función de que se trate. Por último comentar que se pueden usar indistintamente mayúsculas o minúsculas, y que solo se deben introducir los siguientes caracteres en el cuadro de edición: 'fFtT0123456789 '. Cualquier otro carácter no será válido, incluido los espacios, de manera que hay que escribir los nodos codificados sin ningún tipo de separación .En la figura 9.3 se muestra un ejemplo de codificación.

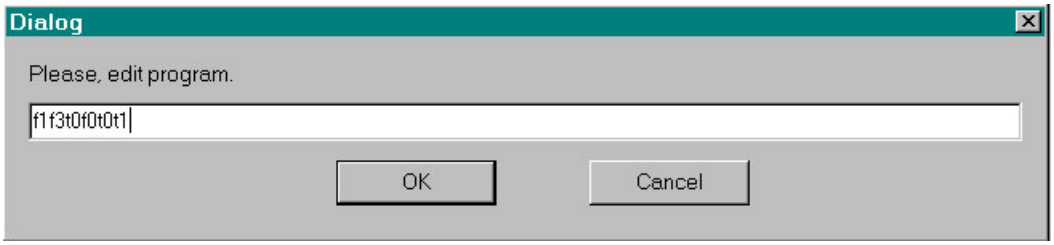


Figura 9.2 Ventana para edición del programa

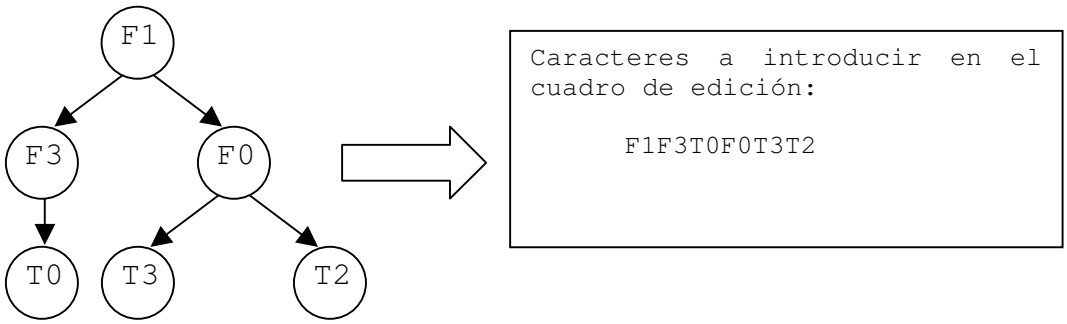


Figura 9.3 Ejemplo de codificación de algoritmo

Como resumen aclaratorio en la figura 9.4 se puede ver un esquema que sintetiza todo el proceso anteriormente comentado.

- **Botón *Run!:*** Sirve para dar la orden de ejecución del algoritmo cargado.
- **Botón *Save Results***: Sirve para almacenar resultados tras la ejecución del algoritmo. Para seleccionar el resultado que se quiere almacenar se abre una ventana de diálogo, mostrada en la figura 9.5. Tras la selección de cualquiera de las opciones se deberá indicar el fichero para el almacenamiento de la información requerida.
- **Botón *View input:*** Accionando este botón se mostrará en el display de imagen la entrada que en ese momento se encuentre seleccionada.

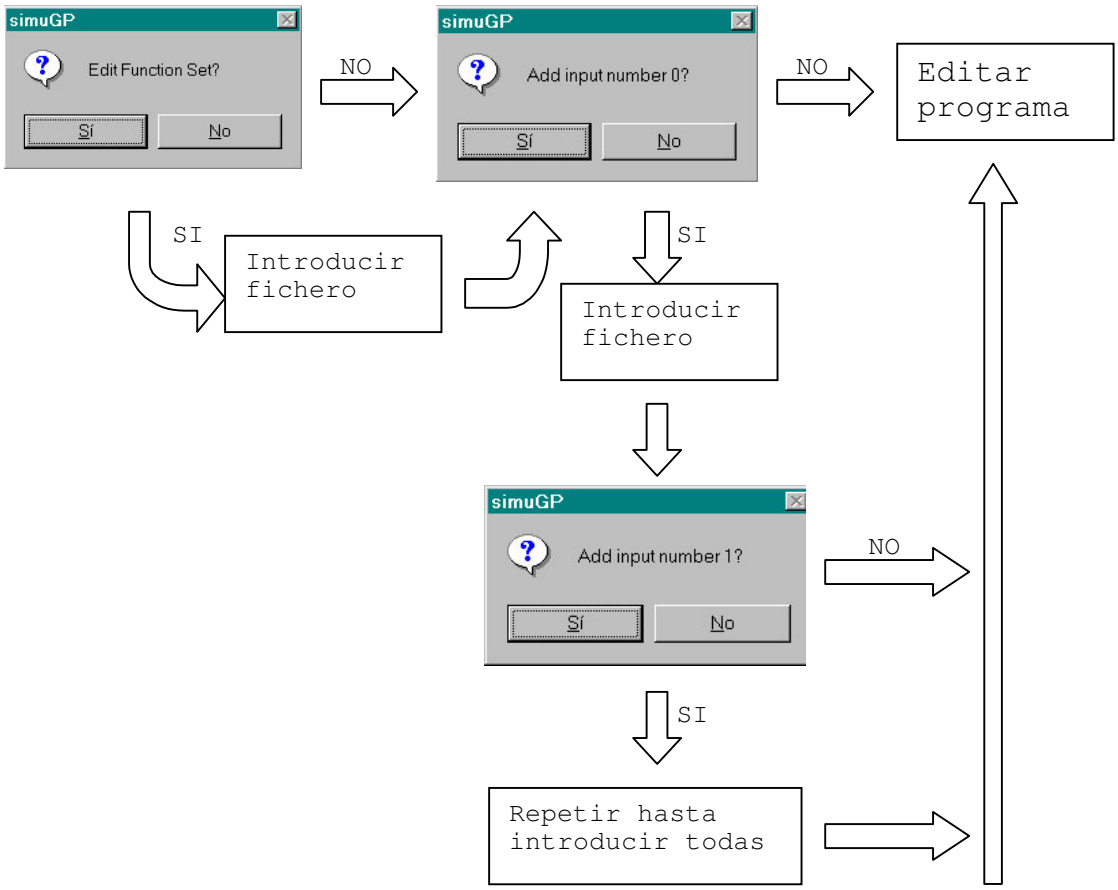


Figura 9.4 Esquema resumen de la edición de programa

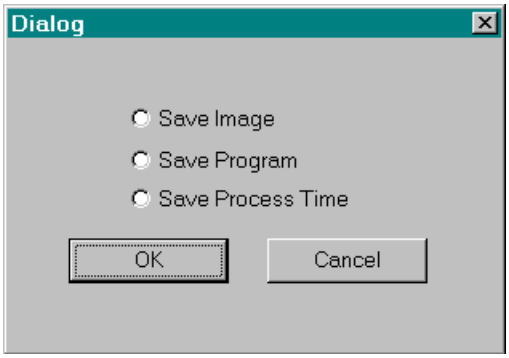


Figura 9.5. Cuadro de diálogo para almacenamiento resultados.

- **Botón *View output:*** Accionando este botón se mostrará en el display de imagen la salida resultante del procesamiento.
- **Imagen *images display:*** Aquí se mostrarán las imágenes seleccionadas por el usuario.

- **Caja de texto *images input:*** En cada línea se mostrará el nombre de cada una de los ficheros que contienen las imágenes introducidas, con su ruta completa.
- **Caja de texto *Functions configuration:*** Se muestran los ficheros correspondientes a cada una de las plantillas introducidas.
- **Caja de texto *Configuration File***: Muestra el nombre del fichero correspondiente a la configuración de las funciones.
- **Caja de texto *Processing time:*** En esta caja se muestran los tiempos de procesamiento normalizados de cada una de las funciones que intervienen en el programa.
- **Caja de texto *Processing time:*** Aquí se muestra el arbol codificado en prefijo, correspondiente al algoritmo en cuestión.
- **Cajas de edición de texto *CNN maximum variations option:*** Son tres cajas en las que se definen los parámetros que rigen la variación del paso de integración en la simulación de cada una de las plantillas:
  - **Caja *predicted*:** Define la variación predicha
  - **Caja *limit***: Define la variación límite.
  - **Caja *stop***: Defiene la variación de parada

9.4 EJEMPLOS DE APLICACIÓN

9.4.1 MEDIDOR DE RUGOSIDAD

En este ejemplo se va a simular el medidor de rugosidad que ya se mencionó como ejemplo de aplicación en visión artificial. Lo primero que hay que hacer es editar el fichero de configuración con las plantillas que se utilizarán, o bien utilizar un fichero de configuración que las contenga. Suponemos que no lo tenemos, de manera que editaríamos el fichero en modo texto, en el que se debería indicar cada uno de los ficheros de plantilla que se van a utilizar, con la ruta completa, y almacenarlo con extensión *.cfg*. En este caso la plantillas que se utilizan son *threshold*, *Concave Location Filler*, *Logic Difference* y *Erosion* todas ellas incluidas en la biblioteca de plantillas del paquete. Para la plantilla de umbralización *thershold* se ha supuesto un umbral 0.

De esta manera el fichero de configuración podría quedar como se muestra en la figura 9.6, donde *[ruta]* habría que sustituirlo por la ruta del directorio donde tengamos las plantillas, y se ha supuesto como unidad la C. Es importante utilizar el carácter '/' para la especificación de la ruta en lugar del carácter '\'.

```
C:/[ruta]/threshold0.tem
C:/[ruta]/concavelocationfiller.tem
C:/[ruta]/logicDifference.tem
C:/[ruta]/Erosion.tem
```

Figura 9.6 Fichero de configuración de funciones

A continuación se realizaría la codificación del programa, para lo cual hay que codificar el árbol que lo representa mediante codificación prefijo. Este árbol se muestra en la figura 9.7.

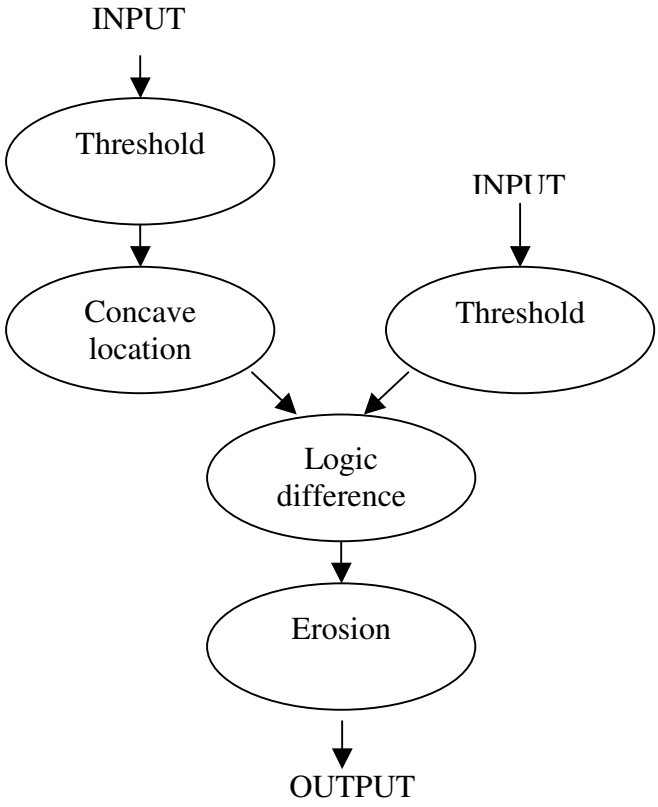


Figura 9.7 Estructura del programa

Utilizando un fichero de configuración que muestre las plantillas en el mismo orden que el fichero de la figura 9.6 la codificación de los distintos nodos sería:

NODO	**NODO CODIFICADO**
Entrada	T0
Threshold0	F0
ConcaveLocationFiller	F1
LogicDifference	F2
Erosion	F3

De manera que la codificación del árbol, y por tanto del programa, utilizando la notación de árbol prefijo, es la siguiente:

F3F2F1F0T0F0T0

Con esto y con la imagen de entrada ya tenemos todo para realizar la simulación. Abrimos el programa *Template Simulator* y pulsamos el botón *Edit program* de la ventana principal. En ese momento se indicará que debemos realizar la configuración de las funciones, y a continuación se debe indicar el fichero *.cfg* que hayamos editado a tal efecto. Después de esto se pedirá la configuración de las imágenes, que en este caso es solo una, e indicaríamos el fichero correspondiente. Por ultimo se abrirá la ventana de

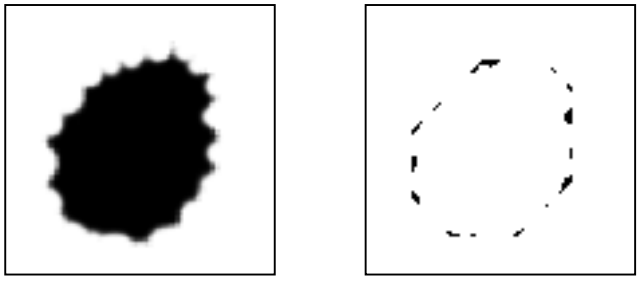
edición del algoritmo. En la caja de texto del cuadro de diálogo se debería introducir el programa codificado.

Tras esto, si todo ha sido realizado correctamente, deben aparecer las funciones con su numeración correspondiente en la caja de texto *Functions configuration*, y el programa codificado en la *Program tree structure.* Tambien podemos ver la imagen introducida, seleccionándola en *Images input* y pulsando el botón *View input*. En caso de que el árbol introducido no sea coherente no aparecerá en *Program tree structure*, y deberá ser nuevamente introducido con la corrección correspondiente.

En caso de que todo sea correcto ajustamos los parámetros de variación de estado en *CNN Simulation options,* que para este caso podrían ser 0.1, 0.15 y 0.01 respectivamente.

En este punto solamente quedaría iniciar la simulación, para lo cual accionaremos el botón *Run!*. Según vaya completando los procesamientos correspondientes a cada una de las plantillas, en *Processing time* irá indicando los tiempos de procesamiento normalizado de cada plantilla, indicando finalmente el tiempo de procesamiento del programa, que será la suma de los tiempos de todas las plantillas. Este tiempo es orientativo, y puede ser mucho mayor al real en caso de que la condición de parada esa muy restrictiva, de manera que para un análisis más preciso del tiempo habría que simular plantilla a plantilla con el programa de simulación de plantillas.

Finalmente se mostrará también la imagen de salida del programa. En este punto se podrán almacenar los resultados que nos interese analizar.



9.8 Imágenes de entrada y salida

También podremos ver los tiempos de procesamiento empleados en cada función. Recordar que estós tiempos son adimensionales, y dependen del diseño electrónico de la CNN. Para este caso se obtiene los siguientes tiempos:

```
Function type 13: 8.305 (tau)
Function type 16: 78.764 (tau)
Function type 3: 7.592 (tau)
Function type 4: 8.678 (tau)
Function type 4: 8.678 (tau)
Total Program Time 112.017 (tau)
```

Para almacenar resultados se debe accionar el botón destinado a tal efecto, *Save result*. Mediante un cuadro de diálogo seleccionaremos si lo que queremos almacenar es la imagen de salida, los tiempos de procesamiento o el programa. La imagen de salida

se almacenará en un fichero BMP, los tiempos de procesamiento en un fichero de texto, y el programa en un fichero que contendrá todos los datos necesarios para poder recuperar el programa en otro momento sin más que seleccionarlo tras accionar el botón *Load Program*, de manera que el usuario puede ir acumulando programas para distintas aplicaciones.

10 PROGRAMA *GP ALGORITHM*

10.1 DESCRIPCIÓN.

En el programa *GP ALGORITHM* se desarrolla la técnica de la programación genética (GP) para la automatización en el diseño de un algoritmo de procesamiento de imágenes en un sistema basado en CNNs. Como se ha dicho los algoritmos que se tratan se limitan al procesamiento de imágenes, y dan como resultado una imagen de salida.

Las plantillas que combina el programa en la búsqueda del algoritmo se consideran fijas, de manera que no se hace ninguna modificación sobre las mismas, la única manipulación que realiza con ellas consiste en la combinación de estas plantillas, las cuales son indicadas por el usuario mediante el fichero de configuración de funciones, que no es más que un fichero de texto en que se indican los nombres de los ficheros que contienen cada una de las plantillas, con su ruta completa. De manera que el usuario puede incluir las plantillas que desee, sin más que editar un fichero de texto, incluyendo plantillas que pueden pertenecer o no a la biblioteca original de pantillas incluidas en el paquete.

A continuación se comentan los distintos elementos que deben ser definidos para realizar la configuración.

Las imágenes a definir son:

- **Entradas**: Son las imágenes base que a manipular en los algoritmos. Constituyen el juego de terminales.
- **Deseada**: Es la salida que se desea conseguir. Interviene a la hora de medir la calidad de los individuos (fitness), según el parecido de la salida del algoritmo correspondiente.

El juego de funciones se define mediante el fichero de configuración de funciones, fichero de texto en el que se indican los nombres de los ficheros que contienen las plantillas en cuestión, con su ruta completa. Es importante diseñar cuidadosamente el número de funciones a incluir, pues un número de funciones excesivo aumenta el espacio de búsqueda y es negativo en la evolución del programa. En caso de que el número de funciones sea muy grande puede ser recomendable dar más importancia a unas plantillas sobre otras, para lo cual simplemente se deben incluir más de una vez en el fichero de configuración de funciones. La importancia relativa que se le de a cada plantilla dependerá del número total de plantillas incluidas y del número de veces que se incluya la plantilla en cuestión en el fichero de configuración.

Los parámetros relativos a la programación genética que el usuario debe definir son los siguientes:

- **Probabilidad de cruce**: Define la probabilidad de que dos padres tomados para la generación de la nueva población sean cruzados o pasen directamente sin cruzarse.
- **Probabilidades de mutación**: Son cuatro valores que definen la probabilidad de que se aplique la mutación en cuestión en cada nuevo individuo generado.
  - Probabilidad de mutación *subárbol*.
  - Probabilidad de mutación *nodo*.

  - Probabilidad de mutación *colapso*.
  - Probabilidad de mutación *hoist.*

- **Tamaño de la población**: Define el número de individuos de la población.
- **Tamaño de la población extra**: Define el número de individuos de la población que intervienen en los métodos de selección (μ,λ) y (μ+λ). En los otros dos métodos de selección se ignora este parámetro.
- **Profundidad de generación**: Define la profundidad máxima en la generación de la población inicial.
- **Profundidad Máxima**: Define la máxima profundidad que debe tener un árbol de la población. No habrá nunca ningún árbol con profundidad mayor que esta.
- **Elitismo**: Casilla de chequeo en la que se indique que se aplique o no el elitismo.
- **Método de selección**: Lista que contiene los cuatro métodos de selección que se pueden aplicar en el programa, de entre los cuales el usuario debe elegir uno.
- **Parámetros de selección**: Son los parámetros necesarios para la correcta configuración de la selección.
  - *Pmin, Pmax*: Parámeros que definen los probabilidades de selección máxima y mínima en el método *Ranking*. En los otro métodos se ignora.
  - N: Parámetro N que define el grado de selectividad en el método de selección *Tournament.*
- **Parámetros del fitness**: Definen la manera de calcular el fitness.
  - Evaluación de longitud: Casilla de chequeo en la que se indica que intervenga en el fitness un término secundario relativo a la longitud del árbol (número de funciones del árbol).
  - Valor de longitud peor: Valor de longitud a partir del cual a un árbol se le asigna el fitness máximo, marcándolo así negativamente.
  - Valor de longitud malo: Valor que sirve para calcular el término ponderador de la evaluación de la longitud. Ver apartado 5 para más detalle.
  - Tipo de evaluación de la imagen de salida: Lista que contiene el tipo de evaluación que se aplica en el cálculo del término principal. Son tres:
    - ¬ Correlación.
    - ¬ Diferencia absoluta.
    - ¬ Diferencia especial.

- **Parámetros de generación**: Sirven para configurar la generación de árboles en el programa.
  - Método de generación de la población inicial. Es posible escogen entre tres métodos:
    - ¬ Método Grow.
    - ¬ Método Full.
    - ¬ Método Ramped.
  - Probabilidad de terminal: Parámetro que sirve en la generación de los árboles para decidir nodo a nodo si tomar una función o un terminal. Interviene no solamente en la generación *Ramped* y *Grow*, sino también en la generación de árboles en la mutación subárbol, de manera que aunque el método seleccionado en la generación sea el método *Full* se debe especificar cuidadosamente este parámetro.

Para una correcta evaluación de los individuos es muy importante definir adecuadamente los parámetros de simulación de la CNN, que son los parámetros de variación de estado, y que determinan el criterio de variación de paso integración como ya se ha comentado en apartados anteriores. Son tres:

- Variación predicha
- Variación límite.
- Variación de parada.

Hay que tener mucho cuidado en el diseño de estos valores, tanto para que la evaluación de los individuos no sea demasiado lenta, como para que la parada no se realice prematuramente.

Por ultimo quedan por comentar los parámetros que definen las condiciones de parada del programa. Parará cuando se cumpla una de las dos condiciones. Son dos:

- Valor de fitness: Parará cuando algún individuo de la población tenga valor del fitnes igual o menor que este.
- Número de iteraciones.

10.2 IMPLEMENTACIÓN DEL SOFTWARE

Como en el resto de programas del paquete se trata de un programa multihilo, que evita que durante el procesamiento la interfaz con el usuario se quede bloqueada y no responda. De nuevo el hilo principal es el bucle de mensajes, cuya principal tarea es tratar la interfaz, mientras que el hilo secundario es lanzado por este hilo principal y es el que realiza todas las tareas de procesamiento.

En la figura 10.1 se muestra un esquema que representa básicamente que como se realiza la búsqueda genética. Tanto en la generación de la población inicial como en la generación de las sucesivas generaciones se cuida que los árboles mantengan coherencia de tipos, en cuanto a que todas las funciones tomen a la entrada con el tipo correspondiente.

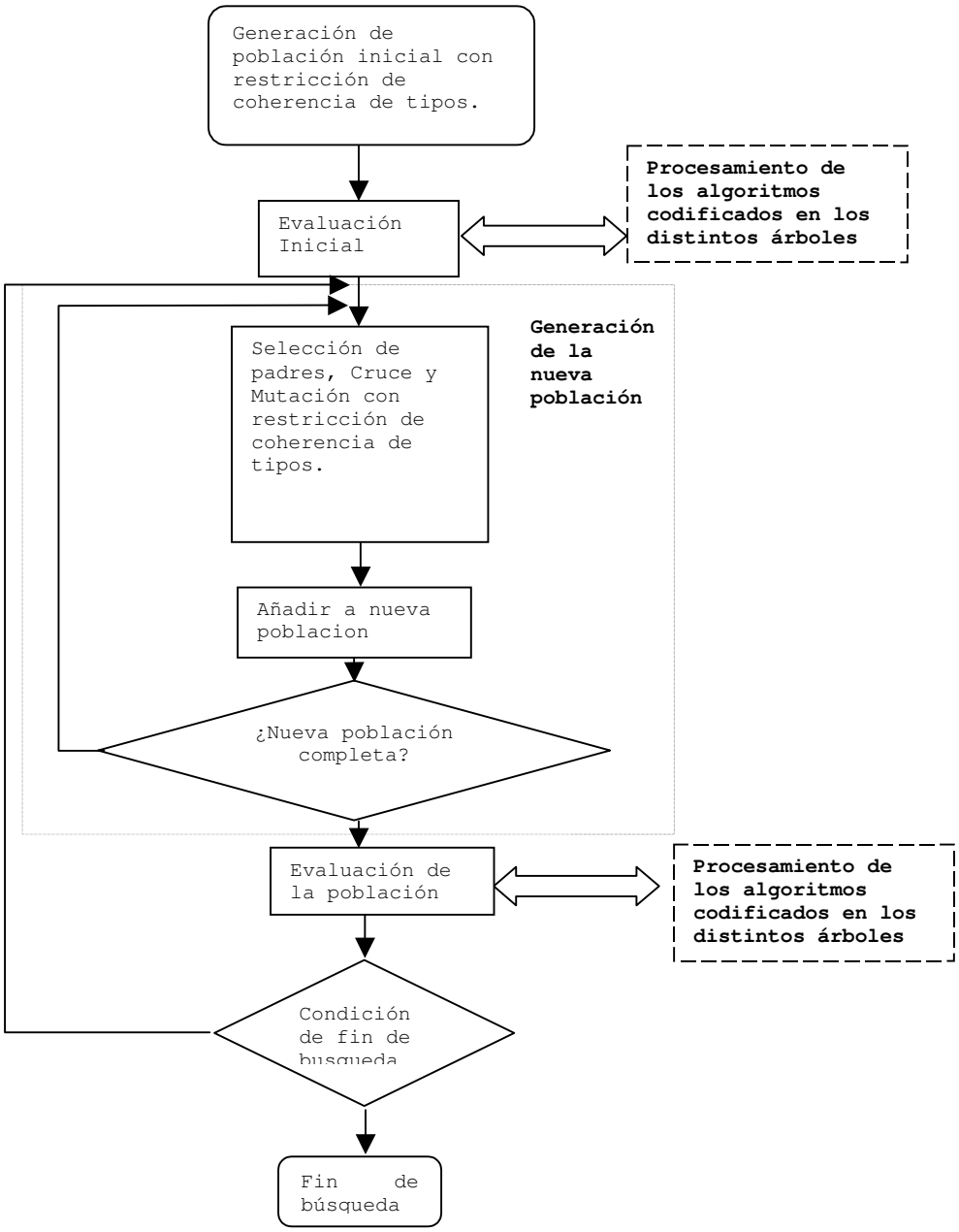


Figura 10.1 Esquema básico de funcionamiento

En la figura 10.2 se muestra la comunicación entre el hilo principal y el hilo que realiza el procesamiento. El hilo principal lanza el hilo de procesamiento cuando el usuario indica que comienza la búsqueda genética. A partir de ese momento la comunicación entre los dos hilos consiste un dos mensajes. Uno es indicar que se ha evaluado un nuevo árbol por parte del hilo de procesamiento, a lo cual el hilo principal responderá mostrando el código del árbol en cuestión y su valor de fitness. El otro es indicar que ha concluido la búsqueda por que se ha cumplido alguna de las condiciones de parada, a lo cual se responderá mostrando por pantalla un mensaje en el que se informa al usuario de este hecho.

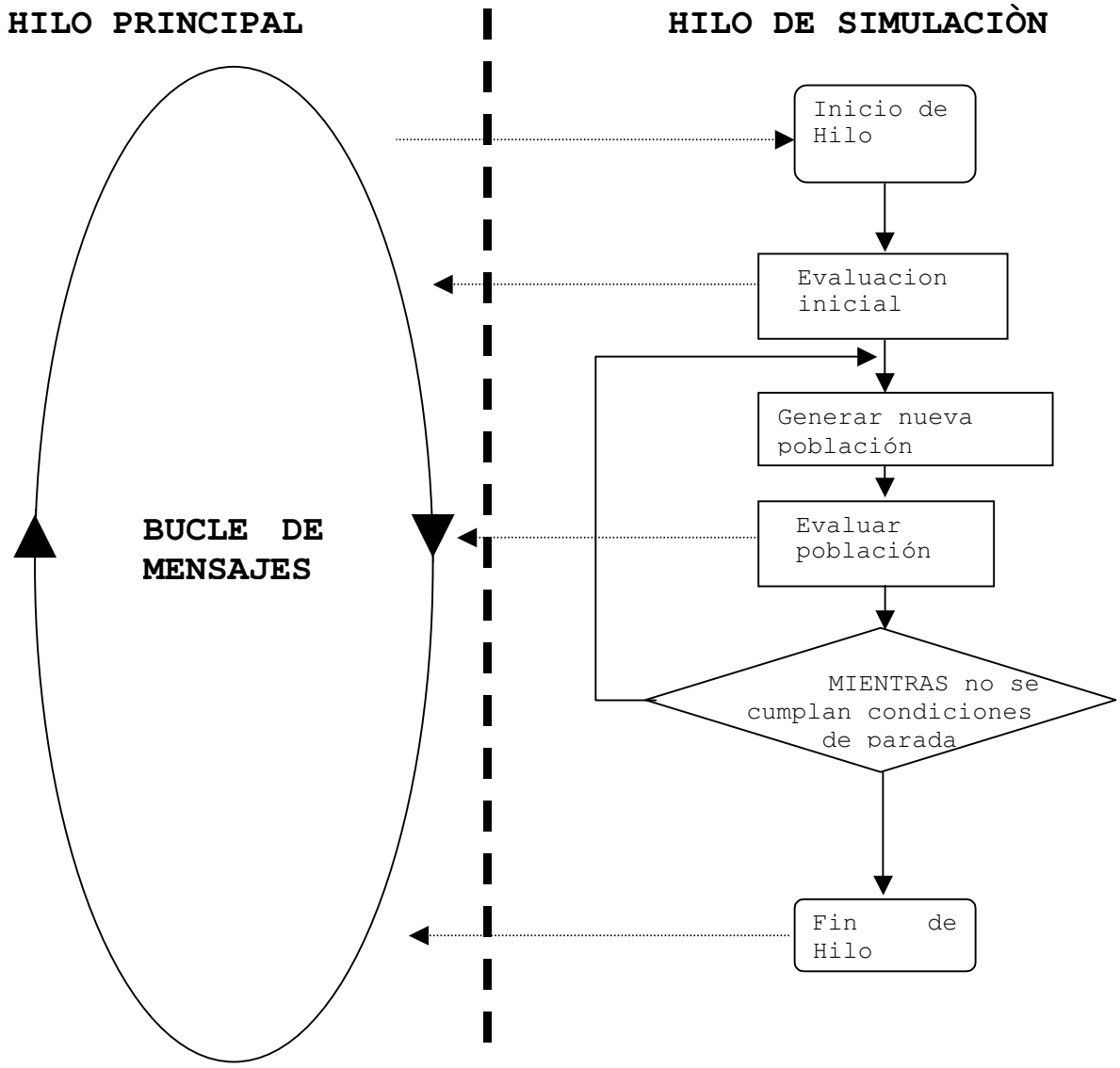


Figura 10.2 Comunicación entre hilos

Todo el intercambio de información entre el hilo principal y el hilo secundario se realiza a través de una sección crítica, para mantener la coherencia en la información intercambiada.

Debido a la cantidad de tiempo que puede ser necesaria en una búsqueda determinada se ofrece al usuario la posibilidad de almacenar el estado en un fichero y recuperarlo posteriormente. Es importante que se haya terminado una iteración completa y la búsqueda haya sido parada tanto para la carga como en la grabación del estado. En el archivo se guardan todos los parámetros de la búsqueda, el número de iteración y toda la población que hubiese en ese momento.

El fichero no está en modo texto, sino que se almacena directamente en el formato original de los datos, de manera que no es recomendable editar este fichero.

10.3 MANUAL DEL USUARIO

La ventana principal del programa (figura 10.3), contiene todos los controles e indicadores necesarios para la correcta configuración y visualización.

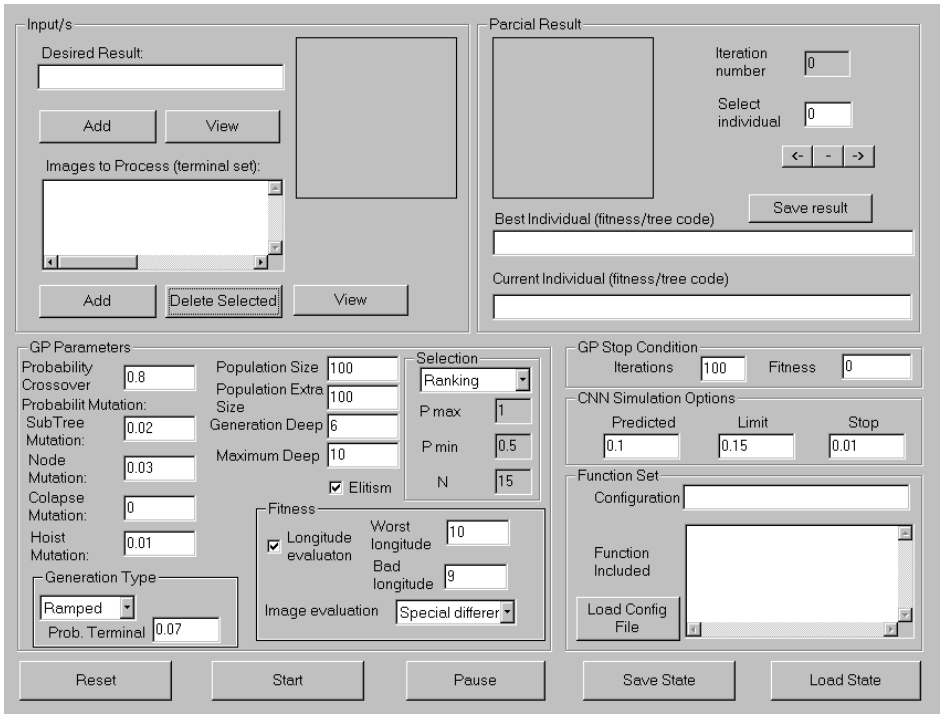


Figura 10.3 Ventana principal

Se puede ver que los distintos elementos de la ventana se muestran agrupados según la manera en que intervengan el el programa. Así, podemos observar las siguientes agrupaciones:

- *Inputs:* Imágenes de entrada.
- *Parcial result*: Muestra resultados parciales durante la búsqueda.
- *GP Parameters*: Los parámetros relativos a la GP.
- *Function Set*: Elementos relativos a la configuración del juego de funciones.
- *CNN simulation parameters*: Parámetros de simulación de la CNN, relativos a la variación de estado.
- *GP stop condition*: Control de la condición de parada de la búsqueda GP.

Vamos a comentar los distintos controles e indicadores en el orden que se debería seguir para realizar un búsqueda GP en general. En primer lugar se debe realizar la configuración de los distintos parámetros e imágenes que intervienen en la búsqueda.

- Configuración de imágenes.

La configuración de las imágenes se realiza de la siguiente manera. En la figura 10.4 se muestra en detalle los elementos de la ventana que intervienen en esta configuración.

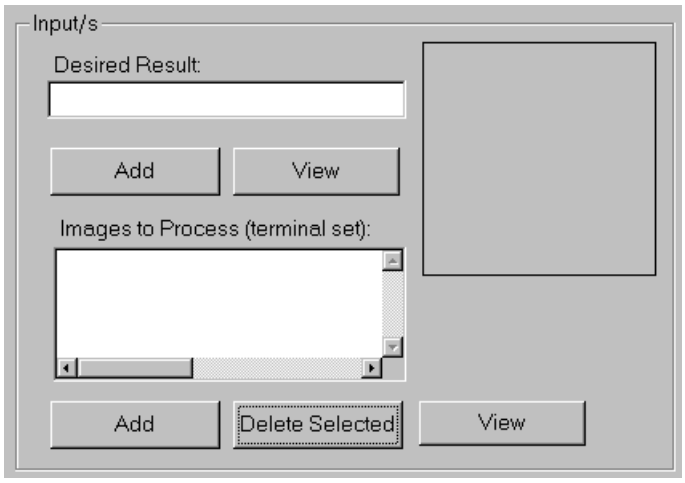


Figura 10.4 Configuración de imágenes

Debemos determinar la imagen deseada, que servirá de criterio para la evaluación de los individuos de la población, y las imágenes que formarán el juego de terminales. Para indicar la imágenes debemos accionar uno de los dos botones *Add*, accionando el botón superior para indicar la imagen deseada, y el inferior para añadir una imagen al juego de terminales. Se pueden añadir tantas imágenes en el juego de terminales como se desee, aunque a efectos prácticos no es recomendable tener un número de imágenes excesivo pues cada imagen aumenta el espacio de búsqueda, lo cual empeora la eficiencia de la misma. Para eliminar alguna imagen del juego de términales se deberá seleccionar en la caja de texto y accionar el botón *delete selected*.

Para visualizar las imágenes cargadas se utilizan los botones *View*, haciendo uso del botón *View* superior para ver la imagen deseada, y del botón inferior para ver la imagen del juego de terminales que en ese momento se encuentre seleccionada.

- Configuración de los parámetros GP.

En la figura 10.5 se muestran los elementos que intervienen en la configuración de los parámentros GP. Son cajas de texto numéricas, casillas de chequeo y cajas de lista de elementos, que permite diseñar completamente la búsqueda GP.

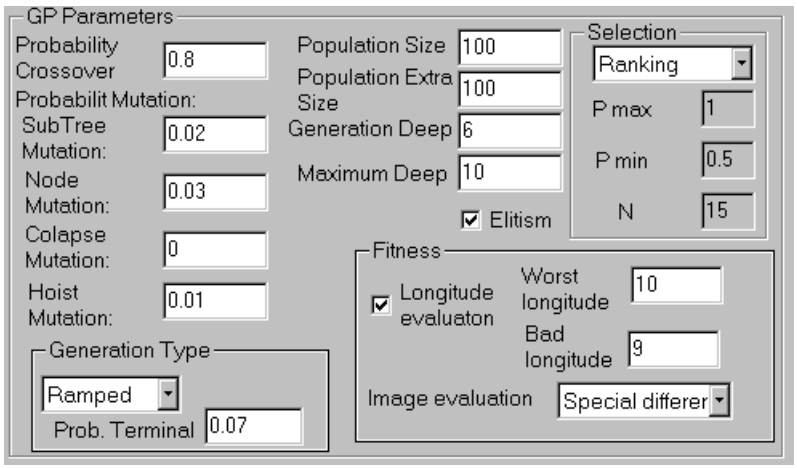


Figura 10.5 Parámetros GP

- Configuración de los parámetros de simulación de la CNN.

Son tres los parámetros que sirven para establecer un criterio en la variación del paso de integración para realizar la simulación de la red de forma eficiente, y ya han sido comentados anteriormente. Se trata de variación a predecir, variación límite y variación de parada.

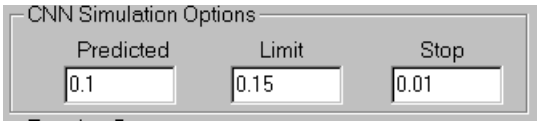


Figura 10.6 Parámetros de simulación de la CNN

- Configuración del juego de funciones.

Para realizar la configuración del juego de funciones se debe accionar el botón *Load config file*, y a continuación especificar el fichero de configuración de funciones. Una vez seleccionado se mostrarán las funciones incluidas en el juego de funciones con la numeración que recibirá cada una en la codificación.

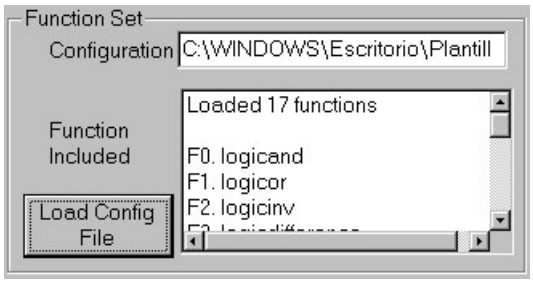


Figura 10.7 Configuración del juego de funciones

- Configuración de las condiciones de parada.

Se pueden definir dos condiciones de parada, que pondrán fin a la búsqueda en caso de que se produzca cualquiera de ellas.

Figura 10.8 Condiciones de parada

Una vez realizada la configuración de la búsqueda GP es el momento de dar comienzo a la búsqueda. Para ello no hay más que accionar el botón *Start* que es encuentra en la parte inferior de la ventana. En ese momento se generará la población inicial, y una vez completada (tardará un tiempo mayor o menor dependiendo del tipo de generación y de la longitud de generación definidos) se comenzará la evaluación de la misma.

Puede ser necesario detener la búsqueda en un momento determinado, para lo cual de deberá accionar el botón *Pause*. Una vez accionado todavía se deberá esperar que se termine de evaluar la población, y hasta que esto no ocurra no se considera que la búsqueda está parada.

En caso de querer abortar una búsqueda para realizar otra se debe usar el botón *Reset*, tras lo cual si se quiere realizar otra búsqueda se debe proceder como si no hubiera habido ninguna.

Para almacenar el estado se usará el botón *Save state*, y especificar el nombre y ubicación del fichero que lo almacene (fichero *gps*). Para recuperar un estado anteriormente almacenado se hará uso del *Load State*, especificando el fichero a continuación. Un aspecto importante que se debe cuidar es que para un correcto almacenamiento y recuperación de estado, **la búsqueda debe encontrarse en un estado de parada**.

En la figura 10.9 se puede ver como se muestra el resultado parcial de la búsqueda en un determinado instante. Cuando se inicia la búsqueda se muestran los distintos individuos (árbol codificado y fitness) así como el mejor individuo de la búsqueda en ese momento (árbol codificado, fitness e imagen de salida). Asimismo se muestran el número de iteración y el número de individuo evaluado, de manera que en cualquier momento se puede observar la marcha de la búsqueda. Para recorrer la población e inspeccionar los resultados de los distintos individuos se hará uso de los botones de recorrido(<- , - , ->), para lo cual se debe haber parado la búsqueda.

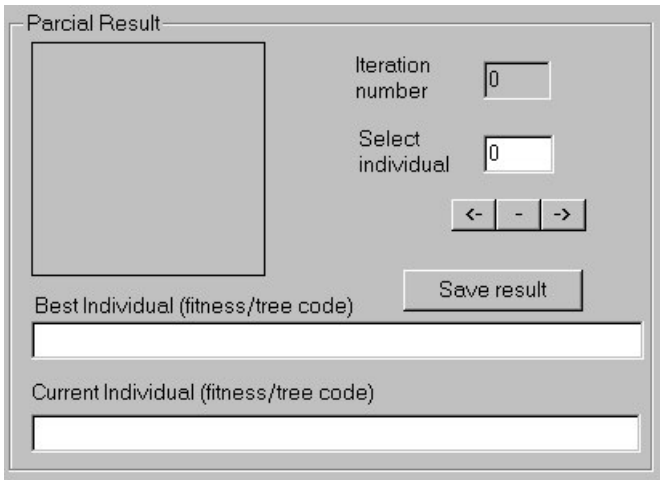


Figura 10.9 Resultado parcial.

Si lo que se quiere es almacenar algún resultado, una vez parada la búsqueda, se debe hacer uso del botón *Save result*. Una vez accionado el botón se abre el cuadro de diálogo que se muestra en la figura 10.10, en el que seleccionaremos entre imagen de salida, fitness e individuo seleccionado. Tras la selección se debe especificar el nombre y ubicación del fichero que almacene el resultado seleccionado. La imagen de salida se almacenará en un fichero BMP. El fitness se almacenará en un fichero con dos columnas de valores, correspondiendo la primera columna al fitness del mejor, y en la segunda el fitness medio. Por último el fichero que almacena el individuo contiene el algoritmo correspondiente, que podrá ser cargado con el programa *Template Simulator* para la simulación del mismo con distintas imágenes.

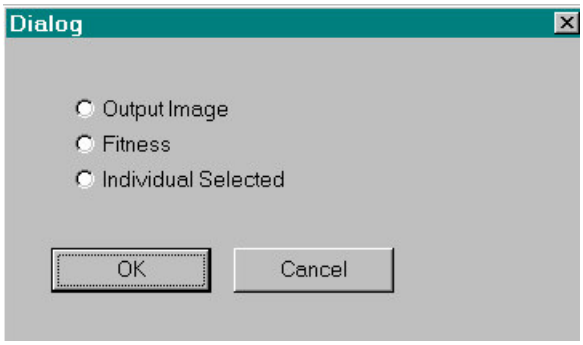


Figura 10.10 Selección de resultado

10.4 EJEMPLOS DE APLICACIÓN

Hay dos cuestiones fundamentales para la aplicación de la programación genética: Las imágenes de entrenamiento y las plantillas a considerar.

Las primeras deben ser tales que definan sin ambigüedades el algoritmo deseado, y además siempre que se pueda se deben introducir soluciones intermedias con un cierto procesado para lo cual nos debemos basar en la intuición y en la experiencia. Otro aspecto importante es que sean lo más pequeña posible, lo cual reducirá enormemente el coste computacional.

Las plantillas por su parte deben ser el menor número posible de estas, pues de esta forma menor será el espacio de búsqueda y más eficiente será la búsqueda. Pero debemos incluir toda plantilla de la cual tengamos el menor indicio que nos pueda ser de utilidad.

En un caso que sea muy complejo se puede intentar descomponer el problema en problemas intermedios, aplicando la programación genética a cada uno de estos problemas.

A continuación se va a aplicar la programación genética para encontrar dos algoritmos ya conocidos, para lo cual se va a proceder como se debería en una situación de desconocimiento de los mismos.

10.4.1 MEDIDOR DE RUGOSIDAD

El este primer ejemplo se va a buscar el algoritmo que realiza la medición de la rugosidad de la superficie de un figura. La imagen de entrada se va a suponer binaria, y va a mostrar la figura en cuestión. A la salida se debe tener una imagen que muestre los puntos donde el contorno tenga oscilaciones.

Empezamos por las imágenes de entrenamiento. Para la ponderación del fitness se va a emplear el método de diferencia especial, de manera que la imagen deseada debe ser en escala de grises, de tal que defina de forma borrosa las zonas donde se desean píxeles negros, que en este caso serán las irregularidades del contorno. Hay que tener en cuenta que el coste computacional es directamente proporcional del área de las imágenes, de manera que hay que procurar en la medida de lo posible minimizar esta área sin restar representatividad del procesamiento buscado. En concreto se han empleado las imágenes de la figura 10.11 como imagen de entrada e imagen deseada.

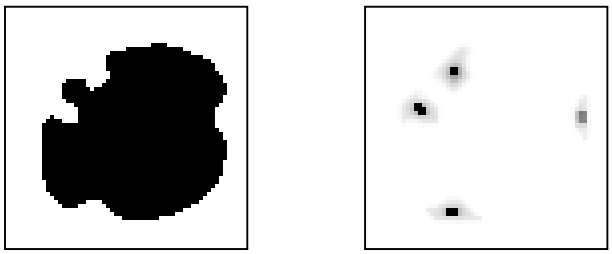


Figura 10.11 Imágenes entrada y deseada

Otro aspecto importante de la búsqueda consiste en definir el conjunto de funciones a considerar en la generación de los algoritmos. Dependiendo del número de funciones que se incluyan se hará el espacio de búsqueda mayor o menor, los cual conllevará una búsqueda más o menos eficiente. En la elección de las plantillas se debe actuar descartando plantillas, bien porque los tipos de imagen de entrada y salida no sean adecuados, bien porque tengan asociado un procesamiento poco relacionado con los que se pretende obtener. Para esto último hay que actuar con mucha precaución, y solamente descartar la plantilla cuando su poca relación con lo que buscamos sea evidente. Para este ejemplo se incluyen las siguientes plantillas de la biblioteca:

- *Logic and*
- *Logic or*
- *Logic inv*
- *Logic difference*
- *Erosion*
- *dilation*
- *Edge Detector*
- *Concave location filler*
- *Box Filler*
- *Box Diagonal Filler*
- *Small Object Remover*
- *Figure Reconstructor*
- *Connectivity*
- *Junction Extractor*
- *Framed Areas Finder*
- *Shadow Left*
- *Shadow Right*
- *Shadow Up*
- *Shadow Down*

- *Shadow LeftDown*
- *Local Southern Detector*
- *Local Northern Detector*
- *Local Eastern Detector*
- *Local Western Detector*

Como ya se ha explicado debemos utilizar un fichero de configuración de funciones, editándolo en modo texto en caso de que no exista, fichero que especificará los distintos ficheros de plantilla a incluir por su nombre y ruta completa, línea a línea. En este caso el fichero debería tener el siguiente aspecto, donde se ha supuesto que la ubicación elegida por el usuario para las plantillas es *CNN design center/Templates Library.*

```
c:/CNN design center/Templates Library/logicand.tem
c:/CNN design center/Templates Library/logicor.tem
c:/CNN design center/Templates Library/logicinv.tem
c:/CNN design center/Templates Library/logicdifference.tem
c:/CNN design center/Templates Library/Erosion.tem
c:/ CNN design center/Templates Library /dilation.tem
c:/ CNN design center/Templates Library /EdgeDetector.tem
c:/ CNN design center/Templates Library /Concavelocationfiller.tem
c:/ CNN design center/Templates Library /BoxFiller.tem
c:/ CNN design center/Templates Library /BoxDiagonalFiller.tem
c:/ CNN design center/Templates Library /SmallObjectRemover.tem
c:/ CNN design center/Templates Library /FigureReconstructor.tem
c:/ CNN design center/Templates Library /Connectivity.tem
c:/ CNN design center/Templates Library /JunctionExtractor.tem
c:/ CNN design center/Templates Library /FramedAreasFinder.tem
c:/ CNN design center/Templates Library /ShadowLeft.tem
c:/ CNN design center/Templates Library /ShadowRight.tem
c:/ CNN design center/Templates Library /ShadowUp.tem
c:/ CNN design center/Templates Library /ShadowDown.tem
c:/ CNN design center/Templates Library /ShadowLeftDown.tem
c:/ CNN design center/Templates Library /LocalSouthernDetector.tem
c:/ CNN design center/Templates Library /LocalNorthernDetector.tem
c:/ CNN design center/Templates Library /LocalEasternDetector.tem
c:/ CNN design center/Templates Library /LocalWesternDetector.tem
```

A continuación debemos configurar los distintos parámetros de programación genética. Los puntos más importantes son un tamaño de población suficientemente grande, y unas tasas de cruce y mutación que eviten la homogeneidad de la población. Durante la búsqueda es recomendable inspeccionar la población y cambiar los parámetros tanto si observa población muy homogénea, o bien una población de fitness muy alto en media. Se recomienda en gran medida, más aún que en GA, utilizar la opción de elitismo dadas las altas tasas de mutación que se manejan. En cualquier caso el usuario tiene la libertad de ajustar este, y el resto de parámetros, como le parezca oportuno. Los parámetros utilizados son los siguientes:

- Tamaño de población: 200
- Probabilidad de cruce: 0.8

- Probabilidades de mutación:
  - Subárbol 0.02
  - Nodo 0.03
  - Colapso 0
  - Hoist 0.01
- Generación:
  - Tipo de generación *Ramped*
  - Probabilidad de terminal 0.07
- Profundidad de generación 6
- Profundidad máxima 10
- Selección:
  - Tipo de selección *Ranking*
  - P max 1
  - P min 0.5
- Fitness
  - Evaluación de imagen *Special Difference*
  - Evaluación de longitud Activado
  - Valor malo de longitud 15
  - Peor valor de longitud 17

Seguidamente definimos los parámetros de simulación de la red, que permiten acelerar la búsqueda, pero con los que hay que tener cuidado. La aceleración de las simulaciones puede provocar la inestabilización de las mismas. Para evitar esto es aconsejable utilizar el simulador de plantillas y probar ajustar los parámetros. Se deben utilizar valores algo menores de los que nos permiten una correcta simulación para las entradas que tratemos, porque un ajuste demasiado al límite puede dar lugar a inestabilidad cuando se trate con imágenes distintas de las que se haya probado. En cualquier caso, si el ajuste no ha sido adecuado no hay más que detener la búsqueda y redefinir los parámetros. Se utilizan los siguientes valores:

- Variación predicha 0.4
- Variación límite 0.5
- Variación de parada 0.001

Por último definimos las condiciones de parada. No hace falta que se cumplan para detener la búsqueda, de manera que se pueden dar unas condiciones suficientemente restrictivas para que la búsqueda no se detenga. Se podrían utilizar los siguientes valores:

- Iteraciones 2000
- Fitness 0

En este punto la búsqueda estaría totalmente configurada, solo quedaría dar comienzo a la misma mediante el botón *Start*. En el momento que lo accionemos el programa irá mostrando los distintos individuos evaluados y el mejor individuo de la búsqueda. La imagen de salida correspondiente al mejor individuo se mostrará en el display de imagen, que al tener activada la opción de elitismo será el mejor que haya habido en la búsqueda hasta ese momento.

El coste computacional de una búsqueda es muy alto, y puede ser necesario el almacenamiento del estado para realizarla en varias sesiones. En el fichero de estado no se almacena la secuencia de fitness, de manera que si queremos hacer un análisis global de la evolución del fitness también debemos almacenar este en otro fichero.

Para parar la búsqueda simplemente debemos utilizar el botón *Pause*. El criterio para detener la búsqueda es que se muestre una solución satisfactoria y que la mejor solución no haya cambiado en un número de iteraciones considerable. En cualquier caso el usuario es libre de detener la búsqueda según su propio criterio. En ese momento debemos esperar a que se complete la evaluación de la población. En el momento en que se complete podemos recorrer las mejores soluciones, que serán las correspondientes a los primeros individuos de la población, pues se estarán ordenados de mejor a peor fitness. En caso de que alguna solución obtenida sea satisfactoria debemos almacenar el individuo y realizar la simulación con otras imágenes para validarla. En caso de que la simulación no dé un buen resultado podemos continuar la búsqueda mediante el botón *Resume*, o realizar otra búsqueda distinta modificando algún parámetro.

En esta búsqueda se encuentran las siguientes imágenes parciales:

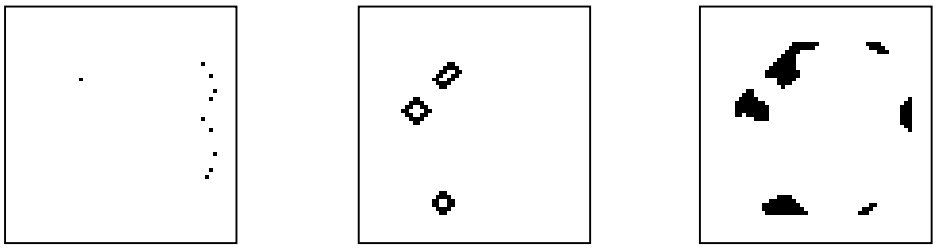


Figura 10.12 Imágenes parciales en la búsqueda

Se detiene la búsqueda en la iteración 396, y se toma como mejor individuo el correspondiente a esta imagen:

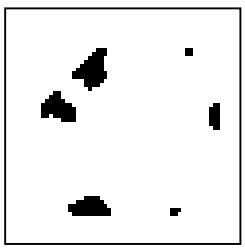


Figura 10.13 Resultado final de la búsqueda

Antes de entras en otras consideraciones validamos este individuo como solución, probando con otros ejemplos. Para ello almacenamos el individuo en un fichero, y lo recuperamos en el programa *Algorithm Simulator*. Editamos las imágenes de entrada para poner otros casos. Se obtienen los siguientes resultados.

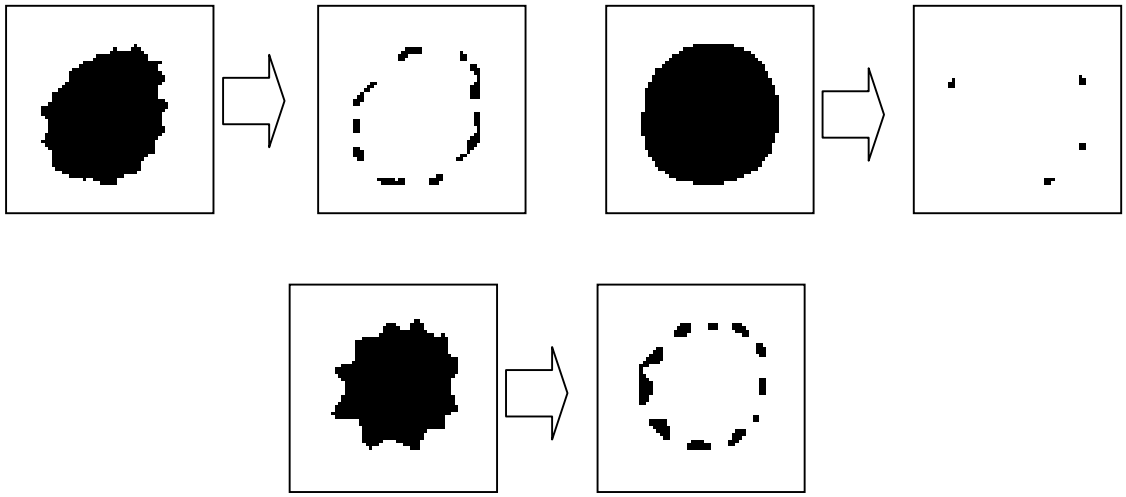


Figura 10.14 Resultados de la aplicación para otros ejemplos

De manera que admitimos la solución como válida. Hay que tener en cuenta que los resultados obtenidos pueden depender de la resolución de la imagen, de manera que para hacer las pruebas se deben utilizar una resolución parecida a la del ejemplo probado.

Podemos representar la evolución del fitness mínimo y del fitness medio de la búsqueda. Para ellos lo almacenamos en un fichero, y lo representamos con un programa adecuado. En el fichero se acumulan los diferentes valores de fitness en dos columnas, la primera corresponde al valor mínimo, y la segunda al valor medio. En caso de haber realizado la búsqueda en varias sesiones debemos concatenar las secuencias de valores para tener una representación global. En la figura 10.15 se muestra la representación gráfica de la evolución del fitness mínimo y del fitness medio. Se puede comprobar que la solución evoluciona de manera bastante más lenta que en GA.

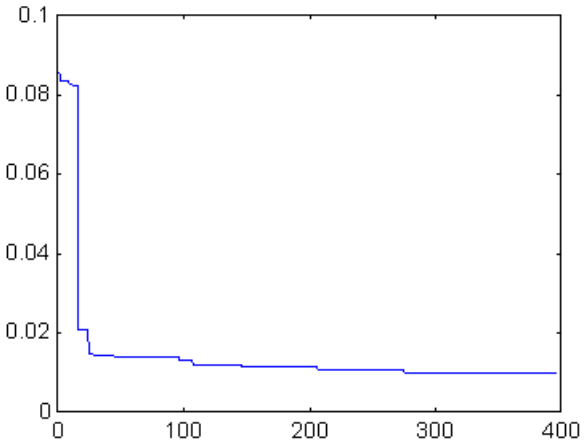


Figura 10.15 Evolución del fitness mínimo

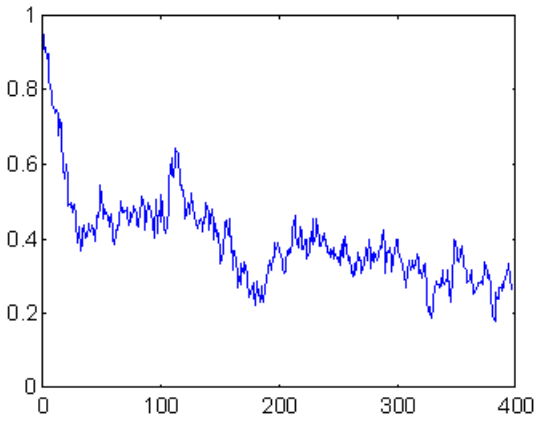


Figura 10.16 Evolución del fitness medio

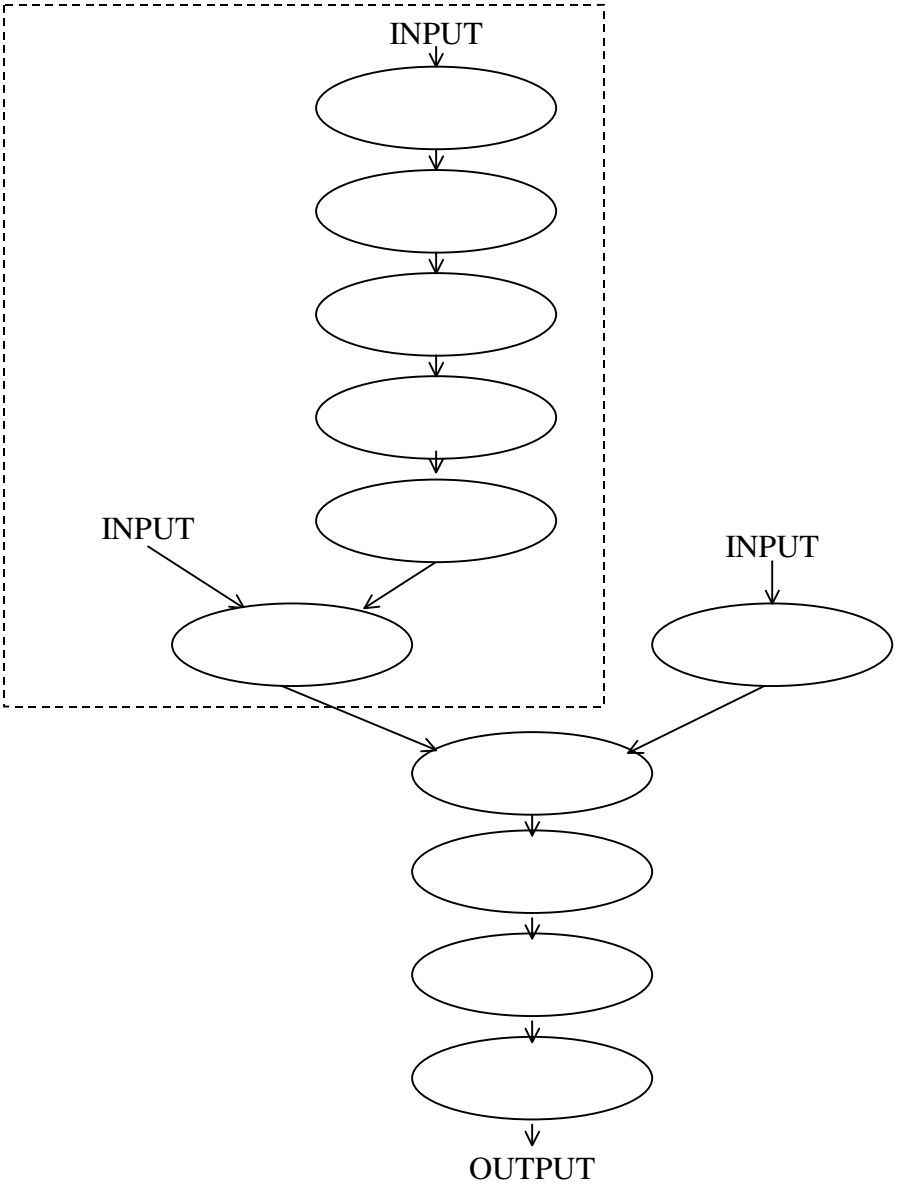


Figura 10.17 Estructura de árbol de la solución

La estructura en árbol del individuo final se muetra en la figura 10.17. En este punto llega el momento del análisis de dicha solución. En el trozo marcado con un recuadro en línea discontinua consiste en un intron, que ya se ha comentado y consiste en una redundancia del individuo que lo protege en los cruces y mutaciones. En este caso todo el bloque marcado equivale a la entrada directamente. Puede extrañar la secuencia final de funciones *Junction Extractor* , pero como se comenta en la descripción de esta plantilla en el anexo, no solo se puede utilizar para extraer los cruces de una figura esqueletizada, sino que también realiza pequeñas erosiones en para una figura en general. Si recordamos la estructura del árbol propuesto para este algoritmo se empleaba función *Erosion* para refinar la solución, pero esta es muy agresiva para una resolución tan baja, y en este caso la función *Junction Extractor* resulta más eficiente. Este es un ejemplo de función que podría ser descartada a priori, pero que en este caso resulta útil.

La conclusión del análisis de la estructura obtenida debe ser eliminar redundancias y extraer la esencia del algoritmo, y sobre esta realizar las adaptaciones en función de la situación en la que se desee aplicar el mismo.

10.4.2 ESTIMADOR DE PROFUNDIDAD

En este segundo ejemplo se va a intentar encontrar el algoritmo que realiza la estimación de profundidad de objetos a partir de la disparidad que se halle entre dos imágenes, izquierda y derecha, de los objetos.

Para la medición de la disparidad se va a suponer que se toma como referencia los puntos más a la derecha de los objetos. Se establece una imagen de entrada lo suficientemente representativa. Se toma una imagen con objetos de tres objetos de formas arbitrarias, con imprecisiones de correspondencia entre los objetos de una y otra imagen, y con dos objetos en la misma horizontal. Se va a establecer como criterio de ponderación de imagen en el fitness el método *difference*, de manera que la imagen deseada va a ser binaria. Esta imagen va a mostrar las disparidades de los diferentes objetos medida a partir del punto más a la derecha y con un ancho aproximado de los objetos. Se toman imágenes con poco área, lo cual acelerará la búsqueda enormemente. En la figura 10.17 se muestran las imágenes entrada y deseada.

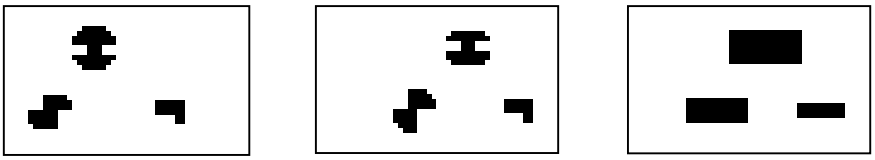


Figura 10.17 Imágenes entrada y deseada

En la elección de las plantillas podemos restringir las plantillas no simétricas de comportamiento direccional a aquellas cuya dirección de propagación sea la horizontal. Con esta restricción, y descartando plantillas incoherentes con el procesamiento del algoritmo deseado, se seleccionan las siguientes plantillas:

- Logic Difference
- Logic Inversion
- Logic And

- Logic Or
- Box Filler
- Box Diagonal Filler
- Concave Location Filler
- Contour Detector Right
- Contour Detector Left
- Masked Shadow Right
- Masked Shadow Left
- Shadow Right
- Shadow Left
- Erosion
- Dilation
- Edge Detector

Para definir este juego de funciones debemos crear un fichero de configuración de funciones, editándolo en modo texto en caso de que no exista, fichero que especificará los distintos ficheros de plantilla a incluir por su nombre y ruta completa, línea a línea. En este caso el fichero debería tener el siguiente aspecto, donde se ha supuesto que la ubicación elegida por el usuario para las plantillas es *CNN design center/Templates Library*:

```
c:/CNN design center/Templates Library/LogicDifference.tem
c:/CNN design center/Templates Library/LogicInv.tem
c:/CNN design center/Templates Library/LogicAnd.tem
c:/CNN design center/Templates Library/LogicOr.tem
c:/CNN design center/Templates Library/BoxFiller.tem
c:/CNN design center/Templates Library/BoxDiagonalFiller.tem
c:/CNN design center/Templates Library/ConcaveLocationFiller.tem
c:/CNN design center/Templates Library/ContourDetectorRight.tem
c:/CNN design center/Templates Library/ContourDetectorLeft.tem
c:/CNN design center/Templates Library/MaskedShadowRight.tem
c:/CNN design center/Templates Library/MaskedShadowLeft.tem
c:/CNN design center/Templates Library/ShadowRight.tem
c:/CNN design center/Templates Library/ShadowLeft.tem
c:/CNN design center/Templates Library/Erosion.tem
c:/CNN design center/Templates Library/Dilation.tem
c:/CNN design center/Templates Library/EdgeDetector.tem
```

En este punto debemos configurar los distintos parámetros de programación genética, de lo cual dependerá en gran medida la eficiencia de la búsqueda. Es recomendable escoger un tamaño de población suficientemente grande, y unas tasas de cruce y mutación que eviten la homogeneidad de la población. Durante la búsqueda es recomendable inspeccionar la población y cambiar los parámetros tanto si se observa población muy homogénea, o bien una población de fitness muy alto en media. Se recomienda utilizar la opción de elitismo dadas las altas tasas de mutación. En cualquier

caso el usuario tiene la libertad de ajustar este, y el resto de parámetros, como le parezca oportuno. Los parámetros utilizados son los siguientes:

- Tamaño de población: 200
- Probabilidad de cruce: 0.8
- Probabilidades de mutación:
  - Subárbol 0.02
  - Nodo 0.03
  - Colapso 0
  - Hoist 0.01
- Generación:
  - Tipo de generación *Ramped*
  - Probabilidad de terminal 0.07
- Profundidad de generación 6
- Profundidad máxima 10
- Selección:
  - Tipo de selección *Ranking*
  - P max 1
  - P min 0.5
- Fitness
  - Evaluación de imagen *Difference*
  - Evaluación de longitud *Activada*
  - Valor malo de longitud 15
  - Peor valor de longitud 17

A continuación definimos los parámetros de simulación de la red que determinarán la velocidad con que se realiza la simulación de la red para cada plantilla del algoritmo procesado. De nuevo se recuerdar que la aceleración de las simulaciones puede provocar la inestabilización de las mismas. Para evitar esto es aconsejable utilizar el simulador de plantillas y probar ajustar los parámetros. Se deben utilizar valores algo menores de los que nos permiten una correcta simulación para las entradas que tratemos, porque un ajuste demasiado al límite puede dar lugar a inestabilidad cuando se trate con imágenes distintas de las que se haya probado. En cualquier caso, si el ajuste no ha sido adecuado no hay más que detener la búsqueda y redefinir los parámetros. Se utilizan los siguientes valores:

- Variación predicha 0.04

- Variación límite 0.05
- Variación de parada 0.001

Por último definimos las condiciones de parada. No hace falta que se cumplan para detener la búsqueda, de manera que se pueden dar unas condiciones suficientemente restrictivas para que la búsqueda no se detenga. Se podrían utilizar los siguientes valores:

- Iteraciones 2000
- Fitness 0

Terminada la configuración de la búsqueda es momento de dar comienzo a la misma mediante el botón *Start*. Durante la búsqueda el programa irá mostrando los distintos individuos evaluados y el mejor individuo de la búsqueda. Se observa que se realiza la evaluación de los individuos con cierta rapidez, debido al poco área que se ha asignado a las imágenes. La imagen de salida correspondiente al mejor individuo se mostrará en el display de imagen, que al tener activada la opción de elitismo será el mejor que haya habido en la búsqueda hasta ese momento.

Puede ser necesario el almacenamiento del estado para realizarla en varias sesiones. En el fichero de estado no se almacena la secuencia de fitness, de manera que si queremos hacer un análisis global de la evolución del fitness también debemos almacenar este en otro fichero.

Con el botón *Pause* podemos detener parcialmente la búsqueda, y reanudarla posteriormente si resulta conveniente activando el mismo botón, que en parada llevará la etiqueta *Resume*. El criterio para detener la búsqueda es que se muestre una solución satisfactoria y que la mejor solución no haya cambiado en un número de iteraciones considerable. En cualquier caso el usuario es libre de detener la búsqueda según su propio criterio. En ese momento debemos esperar a que de complete la evaluación de la población. En el momento en que se complete podemos recorrer las mejores soluciones, que serán las correspondientes a los primeros individuos de la población, pues estarán ordenados de mejor a peor fitness.

En caso de que alguna solución obtenida sea satisfactoria debemos almacenar el individuo y realizar la simulación con otras imágenes para validarla. En caso de que la simulación no dé un buen resultado podemos continuar la búsqueda mediante el botón *Resume*, o realizar otra búsqueda distinta modificando algún parámetro.

En la figura 10.18 se muestran algunas de las imágenes parciales correspondientes a mejores individuos de cada iteración que se van obteniendo.

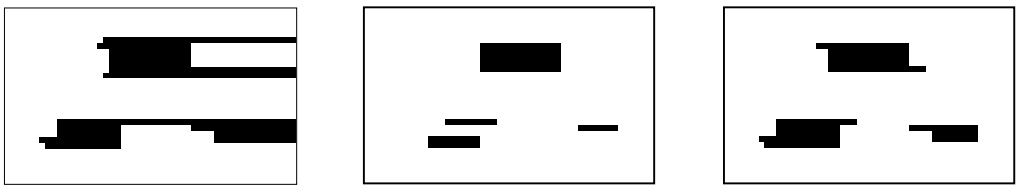


Figura 10.18 Imágenes parciales en la búsqueda.

Detenemos la búsqueda en la iteración 423. En la figura 10.19 se muestra la imagen correspondiente a la salida del algoritmo del mejor individuo de la búsqueda.

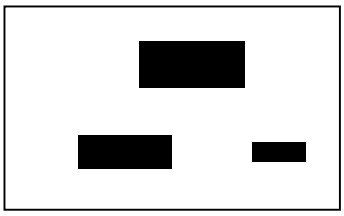


Figura 10.19 Resultado final de la búsqueda

Para admitir la validez del individuo debemos simular el correspondiente algoritmo con distintos ejemplos mediante la utilización del programa *Algorithm Simulator*. Podemos almacenar el algoritmo correspondiente al individuo en un fichero, para abrirlo a continuación en el simulador y editar las imágenes. Se obtienen los resultados mostrados en la figura 10.20.

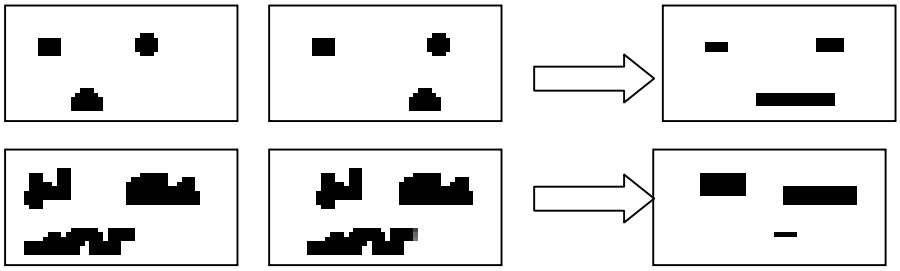


Figura 10.20 Resultados de la aplicación para otros ejemplos

Con estos resultados **no podemos aceptar la solución como válida**. Las imágenes de entrenamiento no han sido suficientemente representativas, y como consecuencia la búsqueda ha estado mal orientada. Hay que repetir la búsqueda con otras imágenes en las que haya una mayor diferencia entre las disparidades de los objetos, y que estos sean de mayor complejidad.

Volvemos a configurar la red cambiando solamente las imágenes de entrenamiento. En la figura 10.21 se muestran las imágenes de entrada e imagen deseada.

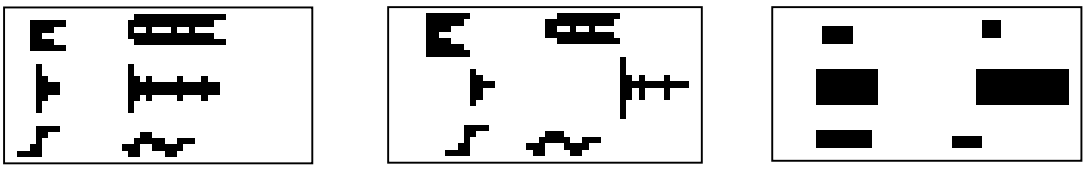


Figura 10.21 Imágenes de entrada derecha, entrada izquierda e imagen deseada

En la iteración 204 se pone fin a la búsqueda. En la figura 10.22 se muestran algunas imágenes parciales en la búsqueda. La imagen de salida del algoritmo resultado de la búsqueda se muestra en la figura 10.23

Se prueba el algoritmo encontrado en otros ejemplos para validar la solución. En este caso se observa que realiza el procesado satisfactoriamente, como se puede observar en la figura 10.23, con lo cual se acepta la solución.

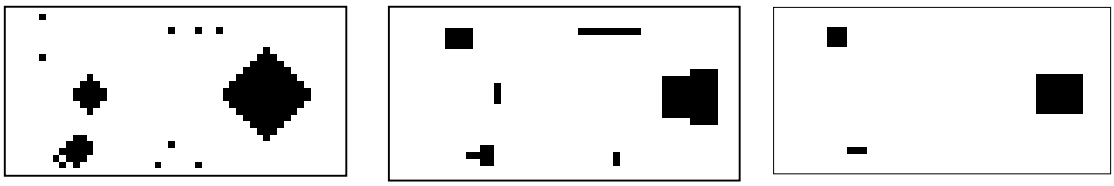


Figura 10.22 Imágenes parciales en al búsqueda

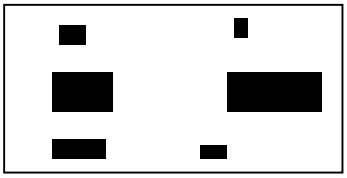


Figura 10.23 Imagen correspondiente a la solución final

Se representan el fitness mínimo y el fintess medio en la figura 10.24 y10.25. Para ello se almacenan en un fichero de texto los distintos valores que se obtienen para cada iteración, y se representan con ayuda de un programa adecuado para ello.

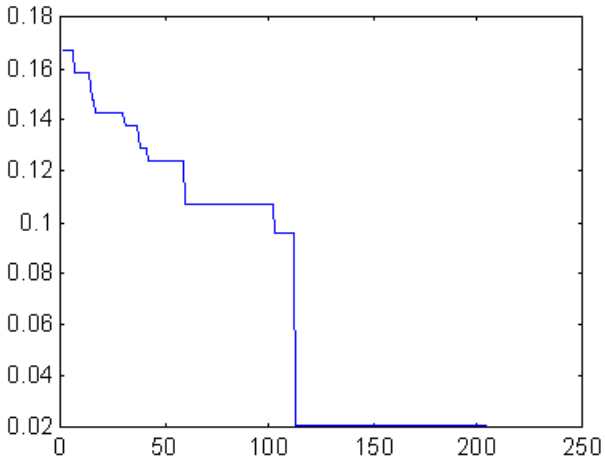


Figura 10.24 Fitness mínimo

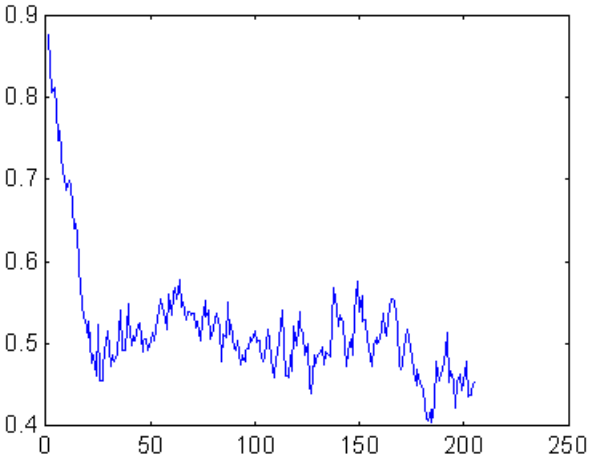


Figura 10.25 Fitness medio

A continuación se pasa a analizar la estructura del algoritmo correspondiente a la solución obtenida. En la figura 10.25 se muestra la estructura del correspondiente árbol. Se comprueba que es prácticamente la misma estructura que en la solución propuesta. La función *Concave Location Filler* es una redundancia del algoritmo, que no tiene ningún efecto puesto que se aplica en una figura que no presenta ninguna concavidad, de manera que podemos considerarlo un intron. Otra diferencia que se observa respecto a la solución que se propuse anteriormente es que no se aplica la función *Dilation*, que realmente puede no ser necesaria en muchos casos, y que la función Masked Shadow se aplica en sentido inverso, pero en cualquier caso el resultado es prácticamente el mismo.

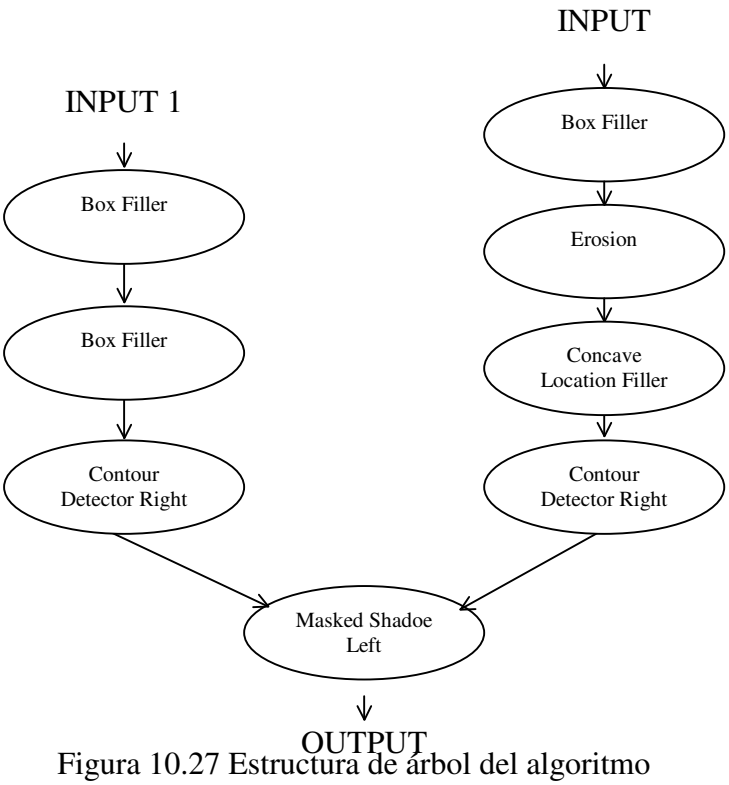


Figura 10.27 Estructura de árbol del algoritmo

11 ANEXO: PLANTILLAS DE LA BIBLIOTECA

En el presente anexo se presentan todas las plantillas incluidas en el paquete. Se comentan las características de cada plantilla y se incorpora un ejemplo para cada plantilla en el que se muestra el procesamiento realizado.

Todas las simulaciones han sido realizadas mediante la utilización del programa incluido en este paquete para la simulación de CNN, *Template Simulator*.

Como resultado se muestra la imagen de salida resultante, así como la evolución de la salida media (en valor absoluto), así como la evolución de la variación del estado. La escala temporal es tiempo normalizado.

11.1 PLANTILLAS LÓGICAS

Denominamos plantillas lógicas a las plantillas que operan de forma local aplicando un operador lógico punto a punto, en el que el blanco se considera '0' y el negro se considera '1'. Estas plantillas son muy simples, y realmente no ponen de manifiesto la capacidad de procesamiento de las CNN, pero son necesarias en muchos algoritmos. Son plantillas que admiten valores altos en la variación de estado predicha y límite para la simulación de la red. Evidentemente su implementación directa de estas plantillas es mucho más eficiente que recurrir a la simulación de la CNN, y en el programa de *GP Algorithm* se utiliza de hecho la implementación directa, lo cual acelera enormemente el procesamiento de los algoritmos.

Los ficheros de las cuatro plantillas lógicas incluidas son:

- *LogicAnd.tem* que contiene la plantilla *Logic And.*
- *LogicOr.tem* que contiene la plantilla *Logic Or.*
- *LogicDifference.tem* que contiene la plantilla *Logic Difference.*
- *LogicInv.tem* que contiene la plantilla *Logic Inversion*.

11.1.1 PLANTILLA *LOGIC AND*

11.1.1.1 Características de la plantilla

- Parámetros:

$$A=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad B=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad I=-1$$

Condición de contorno	Tipo de imagen de entrada	Tipo de imagen de estado inicial	Tipo de imagen de salida
Refleja	*Binaria*	*Binaria*	*Binaria*

- Descripción: Realiza la operación lógica AND entre las imágenes de entrada y estado inicial punto a punto.

11.1.1.2 Ejemplo de aplicación

A continuación se muestra un ejemplo en el que se pone de manifiesto el comportamiento de esta plantilla.

Las imágenes de entrada y estado inicial están contenidas en los ficheros *LogicA.bmp* y *LogicB.bmp* respectivamente.

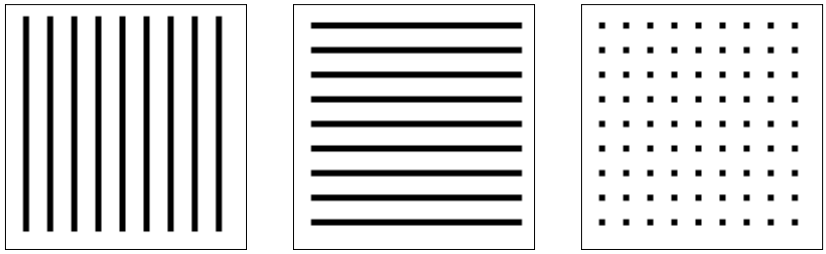


Figura 11.1 Imágenes de entrada, estado inicial y salida

11.1.2 PLANTILLA *LOGIC OR*

11.1.2.1 Características de la plantilla

- Parámetros:

$$A=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad B=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad I=1$$

Condición de contorno	Tipo de imagen de entrada	Tipo de imagen de estado inicial	Tipo de imagen de salida
Refleja	*Binaria*	*Binaria*	*Binaria*

- Descripción: Realiza la operación lógica OR entre las imágenes de entrada y estado inicial punto a punto.

11.1.2.2 Ejemplo de aplicación

A continuación se muestra un ejemplo en el que se pone de manifiesto el comportamiento de esta plantilla.

Las imágenes de entrada y estado inicial están contenidas en los ficheros *LogicA.bmp* y *LogicB.bmp* respectivamente.

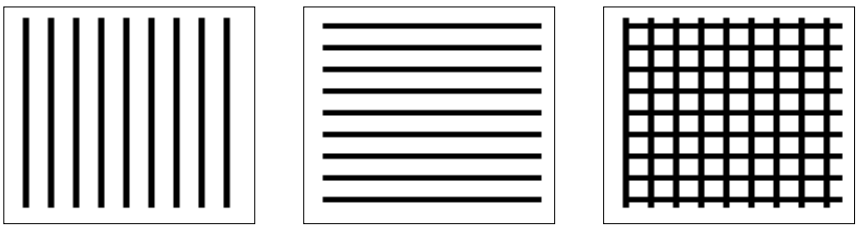


Figura 11.2 Imágenes de entrada, estado inicial y salida

11.1.3 PLANTILLA *LOGIC DIFFERENCE*

11.1.3.1 Características de la plantilla

- Parámetros:

$$A=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad B=\begin{pmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad I=-1$$

Condición de contorno	Tipo de imagen de entrada	Tipo de imagen de estado inicial	Tipo de imagen de salida
Refleja	*Binaria*	*Binaria*	*Binaria*

- Descripción: Realiza la operación lógica de diferencia entre las imágenes de estado inicial y entrada punto a punto. (Estado inicial menos entrada)

11.1.3.2 Ejemplo de aplicación

A continuación se muestra un ejemplo en el que se pone de manifiesto el comportamiento de esta plantilla.

Las imágenes de entrada y estado inicial están contenidas en los ficheros *LogicC.bmp* y *LogicA.bmp* respectivamente.

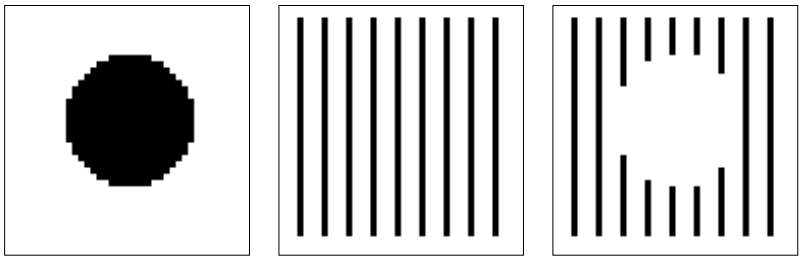


Figura 11.3 Imágenes de entrada, estado inicial y salida.

11.1.4 PLANTILLA *LOGIC INVERSION*

11.1.4.1 Características de la plantilla

- Parámetros:

$$A=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad B=\begin{pmatrix} 0 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad I=0$$

Condición de contorno	Tipo de imagen de entrada	Tipo de imagen de estado inicial	Tipo de imagen de salida
Refleja	*Binaria*	*Todo-gris nulo*	*Binaria*

- Descripción: Realiza la operación lógica Inversión de la imagen de entrada.

11.1.4.2 Ejemplo de aplicación

A continuación se muestra un ejemplo en el que se pone de manifiesto el comportamiento de esta plantilla.

La imagen de entrada está contenida en el fichero *LogicInvInput.bmp*.

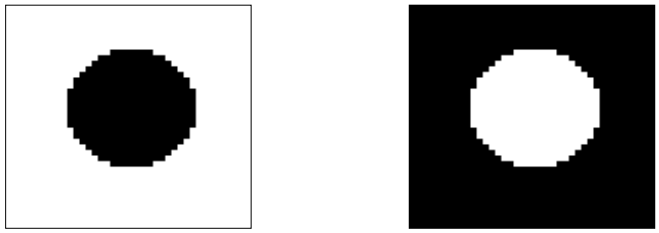


Figura 11.4 Imágenes de entrada y salida.

11.2 PLANTILLAS DE UMBRALIZACIÓN

Estas plantillas transforman una imagen en escala de grises en una imagen binaria. Son muy importantes porque la mayor parte de las plantillas operan a nivel binario, y en caso de tratar imágenes en grises suelen utilizarse estas plantillas.

El procesamiento que se realiza es simplemente una umbralización, es decir, en cada píxel se asigna uno de dos valores posibles, los dos valores extremos del rango considerado. La decisión de uno u otro valor depende del valor del píxel de la entrada y de un parámetro denominado umbral que divide el rango en dos zonas, de manera que se asignará el valor extremo correspondiente a la zona donde se encuentre el valor del píxel de la entrada.

En la biblioteca se han incluido 9 plantillas de umbralización, cada una de ellas con un valor de umbral distinto. Los nombres de la plantilla indican el valor del umbral de la misma. Recordar que en la CNN los valores de cada píxel (salida de cada célula), están comprendidos entre 1 (negro) y –1 (blanco), de manera que los posibles umbrales estan comprendidos en el rango (-1,1). En la biblioteca se recogen los siguientes ficheros que contienen distintas plantillas de umbralización *Threshold*:

- *threshold0.tem,* correspondiente a la plantilla *Threshold* con umbral 0
- *threshold02.tem*, correspondiente a la plantilla *Threshold* con umbral 0.2
- *threshold04.tem*, correspondiente a la plantilla *Threshold* con umbral 0.4
- *threshold06.tem*, correspondiente a la plantilla *Threshold* con umbral 0.6
- *threshold08.tem*, correspondiente a la plantilla *Threshold* con umbral 0.8
- *threshold-02.tem*, correspondiente a la plantilla *Threshold* con umbral -0.2
- *threshold-04.tem*, correspondiente a la plantilla *Threshold* con umbral -0.4
- *threshold-06.tem*, correspondiente a la plantilla *Threshold* con umbral -0.6
- *threshold-08.tem*, correspondiente a la plantilla *Threshold* con umbral -0.8

Se puede comprobar que el usuario puede definir sus propias plantillas de umbralización sin más que modificar el *bias* (parámetro *current* del fichero). Así pues, los parámetros de la plantilla de umbralización genérica es la siguiente:

$$A=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad B=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad I=umbral$$

Dada la similitud de las plantillas solamente se va a analizar la plantilla con umbral igual a 0.

11.2.1 PLANTILLA *THRESHOLD* CON UMBRAL 0

11.2.1.1 Características de la plantilla

- Parámetros:

$$A=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad B=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad I=0$$

Condición de contorno	Tipo de imagen de entrada	Tipo de imagen de estado inicial	Tipo de imagen de salida
Refleja	*Todo-gris nulo*	*En grises*	*Binaria*

- Descripción: Realiza la umbralización de la imagen de entrada con un valor de umbral de 0.

11.2.1.2 Ejemplo de aplicación

A continuación se muestra un ejemplo en el que se pone de manifiesto el comportamiento de esta plantilla. La imagen de entrada está contenida en el fichero *Threshold.bmp*.

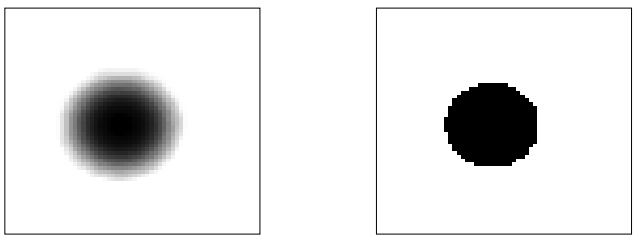


Figura 11.5 Imágenes de entrada y salida.

11.3 PLANTILLAS DETECTOR DE COMPONENTE CONECTADA

La función de estas plantillas es la de detectar componentes conectadas horizontal, vertical o diagonalmente. En los bordes de la imagen se pueden observar el número de componentes conectadas en la dirección en cuestión.

Los ficheros que recogen las plantillas de este apartado son las siguientes:

- *CCDHorizontal.tem* correspondiente a la plantilla *CCD Horizontal*.
- *CCDVertical.tem* correspondiente a la plantilla *CCD Vertical.*
- *CCDDiagonal.tem* correspondiente a la plantilla *CCD Diagonal.*

A continuación se analiza la plantilla *CCD Horizontal*. La *CCD Vertical* y *CCD Diagonal* son completamente equivalentes y se obtienen de la rotación de la matriz *A*.

11.3.1 PLANTILLA *CCD HORIZONTAL*

11.3.1.1 Características de la plantilla

- Parámetros:

$$A=\begin{pmatrix} 0 & 0 & 0 \\ 1 & 2 & -1 \\ 0 & 0 & 0 \end{pmatrix} \quad B=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad I=0$$

Condición de contorno	Tipo de imagen de entrada	Tipo de imagen de estado inicial	Tipo de imagen de salida
Fija	*Todo-gris nulo*	*Binaria*	*Binaria*

- Descripción: Detecta el numero de componentes conectadas horizontales en las figuras de la imagen.

11.3.1.2 Ejemplo de aplicación

A continuación se muestra un ejemplo en el que se pone de manifiesto el comportamiento de esta plantilla.

La imagen de estado inicial está contenida en el fichero *CCDA.bmp*.

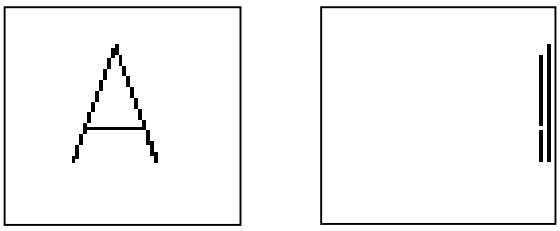


Figura 11.6 Imágenes de estado inicial y salida.

11.4 PLANTILLAS DETECTOR DE COMPONENTE CONECTADA ENMASCARADA

Las plantillas detector de componente conectada enmascarada decir que realizan la misma función que las del apartado anterior con la diferencia de que se puede indicar las posiciones donde se mostrarán las componentes conectadas.

Los ficheros que recogen las plantillas de este apartado son las siguientes:

- *MaskedCCDHorizontal.tem* correspondiente a la plantilla *Masked CCD Horizontal.*

- *MaskedCCDVertical.tem* correspondiente a la plantilla *Masked CCD Vertical.*
- *MaskedCCDDiagonal.tem* correspondiente a la plantilla *Masked CCD Diagonal.*

A continuación se analiza la plantilla *Masked CCD Horizontal*. La *Masked CCD Vertical* y *Masked CCD Diagonal* son completamente equivalentes y se obtienen de la rotación de la matriz *A*.

11.4.1 PLANTILLA *MASKED CCD HORIZONTAL*

11.4.1.1 Características de la plantilla

- Parámetros:

$$A=\begin{pmatrix} 0 & 0 & 0 \\ 1 & 2 & -1 \\ 0 & 0 & 0 \end{pmatrix} \quad B=\begin{pmatrix} 0 & 0 & 0 \\ 0 & -3 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad I=-3$$

Condición de contorno	Tipo de imagen de entrada	Tipo de imagen de estado inicial	Tipo de imagen de salida
Fija	*Binaria*	*Binaria*	*Binaria*

- Descripción: Detecta el numero de componentes conectadas horizontales, con la particularidad de que en la imagen entrada se especifican puntos de bloqueo donde se frena la transmisión del resultado. Mediante la rotación de la matriz A se consiguen el resto de direcciones.

11.4.1.2 Ejemplo de aplicación

A continuación se muestra un ejemplo en el que se pone de manifiesto el comportamiento de esta plantilla.

Las imagenes de entrada y estado inicial están contenidas en los fichero *BB1CCD.bmp* y *BB2CCD.bmp* respectivamente.

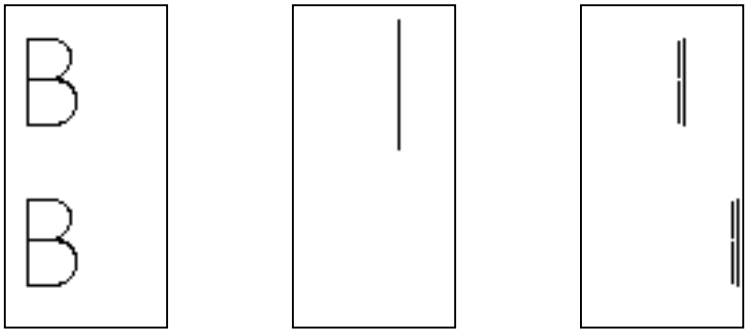


Figura 11.7 Imágenes de estado inicial,entrada y salida.

11.5 PLANTILLAS PROYECCIÓN DE SOMBRA

Como su nombre indica la salida de estas plantillas resulta de proyectar la sombra de la imagen de entrada en una determinada dirección. Hay 8 direcciones posibles , con lo cual hay 8 posibles plantillas de proyección de sombra. La plantilla genérica es la siguiente:

$$A=\begin{pmatrix} x & x & x \\ x & 2 & x \\ x & x & x \end{pmatrix} \quad B=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad I=0$$

Donde con en las *x* se indica la dirección de proyección de sombra, poniendo un 2 en la posición opuesta a la dirección de proyección, y en el resto de posiciones se debe poner un 0.

En la biblioteca se han incluido 5 plantillas, correspondientes a proyección a la izquierda, derecha, arriba , abajo e izquierda-abajo. Para conseguir el resto de direcciones no ha más que rotar la matriz A.

Los ficheros que recogen las plantillas de este apartado son las siguientes:

- *ShadowLeft.tem* correspondiente a la plantilla *Shadow Left.*
- *ShadowRight.tem* correspondiente a la plantilla *Shadow Right.*
- *ShadowDown.tem* correspondiente a la plantilla *Shadow Down.*
- *ShadowUp.tem* correspondiente a la plantilla *Shadow Up.*
- *ShadowLeftDown.tem* correspondiente a la plantilla *Shadow Left Down.*

A continuación se analiza la plantilla *Shadow Left*. Los otras plantillas son completamente equivalentes, con la única variación de la dirección de proyección, que se controla mediante la rotación de la matriz A.

11.5.1 PLANTILLA *SHADOW LEFT*

11.5.1.1 Características de la plantilla

- Parámetros:

$$A=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 2 \\ 0 & 0 & 0 \end{pmatrix} \quad B=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad I=0$$

Condición de contorno	Tipo de imagen de entrada	Tipo de imagen de estado inicial	Tipo de imagen de salida
Refleja	*Todo-negro*	*Binaria*	*Binaria*

- Descripción: Realiza la proyección de la figura de la imagen de entrada hacia la izquierda.

11.5.1.2 Ejemplo de aplicación

A continuación se muestra un ejemplo en el que se pone de manifiesto el comportamiento de esta plantilla.

La imagen de estado inicial está contenida en el fichero *objects.bmp*. Como imagen de entrada se debe utilizar la contenida en el fichero *black100x100.bmp* .

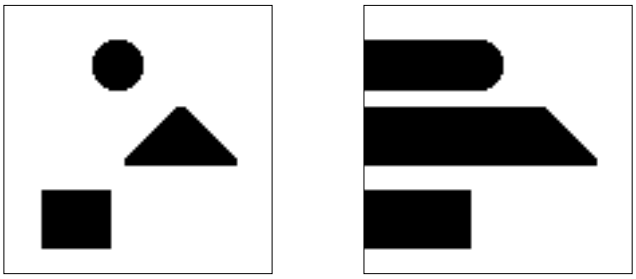


Figura 11.8 Imágenes de estado inicial, y salida.

11.6 PLANTILLAS DE PROYECCIÓN DE SOMBRA ENMASCARADA

El procesamiente realizado por estas plantillas es básicamente el mismo que las del apartado anterior, con la diferencia que en la imagen de entrada se indican zonas de activación mediante píxeles blancos, y zonas de bloqueo con píxeles negros, de manera que la proyección se realiza en las zonas de proyección, y no se realiza en las zonas de bloqueo. De nuevo hay 8 posibles direcciones de proyección, horizontales, verticales y diagonales.

La plantilla genérica es la siguiente:

$$A=\begin{pmatrix} x & x & x \\ x & 1.8 & x \\ x & x & x \end{pmatrix} \quad B=\begin{pmatrix} 0 & 0 & 0 \\ 0 & -1.2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad I=0$$

Donde con en las *x* se indica la dirección de proyección de sombra, poniendo un valor de 1.5 en la posición opuesta a la dirección de proyección, y en el resto de posiciones se debe poner un valor 0.

En la biblioteca se han incluido 5 plantillas, correspondientes a proyección a la izquierda, derecha, arriba, abajo e izquierda-abajo. Para conseguir el resto de direcciones no ha más que rotar la matriz de realimentación *A*.

Los ficheros que recogen las plantillas de este apartado son las siguientes:

- *MaskedShadowRight.tem* correspondiente a la plantilla *Masked Shadow Right.*
- *MaskedShadowLeft.tem* correspondiente a la plantilla *Masked Shadow Left.*
- *MaskedShadowUp.tem* correspondiente a la plantilla *Masked Shadow Up.*
- *MaskedShadowDown.tem* correspondiente a la plantilla *Masked Shadow Down.*
- *MaskedShadowLeftDown.tem* correspondiente a la plantilla *Masked Shadow Left-Down.*

A continuación se analiza la plantilla *Masked Shadow Left*. Los otras plantillas son completamente equivalentes, con la única variación de la dirección de proyección, que se controla mediante la rotación de la matriz A.

11.6.1 PLANTILLA MASKED SHADOW LEFT.

11.6.1.1 Características de la plantilla

- Parámetros:

$$A=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1.8 & 1.5 \\ 0 & 0 & 0 \end{pmatrix} \quad B=\begin{pmatrix} 0 & 0 & 0 \\ 0 & -1.2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad I=0$$

Condición de contorno	Tipo de imagen de entrada	Tipo de imagen de estado inicial	Tipo de imagen de salida
Refleja	*Binaria*	*Binaria*	*Binaria*

- Descripción: Realiza la proyección de la figura de la imagen de estado inicial hacia la izquierda en las zonas marcadas con píxeles blancos. La proyección no se realiza y se bloquea en las zonas de la imagen de entrada marcadas con píxeles negros.

11.6.1.2 Ejemplo de aplicación

A continuación se muestra un ejemplo en el que se pone de manifiesto el comportamiento de esta plantilla.

Las imágenes de estado inicial y entrada están contenidas en los ficheros *Objects.bmp* y *ShadowMask.bmp*.

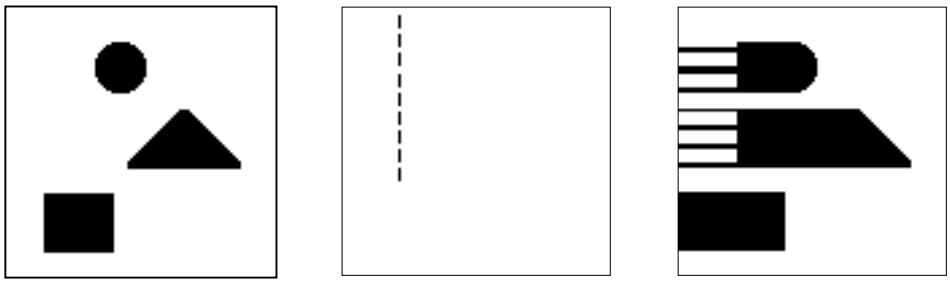


Figura 11.9 Imágenes de estado inicial, entrada y salida.

11.7 PLANTILLAS DE DETECCIÓN DE LÍNEA

Estas plantillas realizan la detección de las líneas que se encuentren en una determinada dirección. Las direcciones posibles son la horizontal, la vertical y las diagonales. Se pueden hacer combinaciones de manera que se eliminen líneas en varias direcciones. La plantilla genérica es la siguiente:

$$A=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad B=\begin{pmatrix} d1 & v & d2 \\ h & 1 & h \\ d2 & v & d1 \end{pmatrix} \quad I=-4$$

Donde *h*, *v*, *d1* y *d2* se asocian a la horizontal, vertical y diagonales respectivamente. Dependiendo de que estos parámetros valgan 0, 1 ó –1 tendremos un comportamiento distinto, que resumimos en el siguiente cuadro:

- Detección de línea horizontal: *h=1*, *v=-1*, *d1=d2=0*.
- Detección de línea vertical: *h=-1*, *v=1*, *d1=d2=0*.
- Detección de línea diagonal NE-SO: *d1=1*, *d2=-1*, *h=v=0*.
- Detección de línea horizontal NO-SE: *d1=-1*, *d2=1*, *h=v=0*.

Los ficheros que recogen las plantillas de este apartado son las siguientes:

- *LineDetectorHorizontal.tem*, correspondiente a la plantilla *Line Detector Horizontal.*
- *LineDetectorVertical.tem*, correspondiente a la plantilla *Line Detector Vertical.*
- *LineDetectorDiagonalNE.tem*, correspondiente a la plantilla *Line Detector Diagonal NE-SO*.
- *LineDetectorDiagonalNO.tem*, correspondiente a la plantilla *Line Detector Diagonal NO-SE*.
- A continuación se analiza la plantilla *Line Detector Horizontal.*

11.7.1 PLANTILLA *LINE DETECTOR HORIZONTAL.*

11.7.1.1 Características de la plantilla

- Parámetros:

$$A=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad B=\begin{pmatrix} 0 & -1 & 0 \\ 1 & 1 & 1 \\ 0 & -1 & 0 \end{pmatrix} \quad I=-4$$

Condición de contorno	Tipo de imagen de entrada	Tipo de imagen de estado inicial	Tipo de imagen de salida
Refleja	*Binaria*	*Todo-blanco*	*Binaria*

- Descripción: Elimina las líneas horizontales en la imagen de entrada.

11.7.1.2 Ejemplo de aplicación

A continuación se muestra un ejemplo en el que se pone de manifiesto el comportamiento de esta plantilla.

La imagen de estado inicial está contenida en el fichero *lines.bmp.* En la imagen de entrada debemos introducir la imagen contenida en el fichero *white100x100.bmp*.

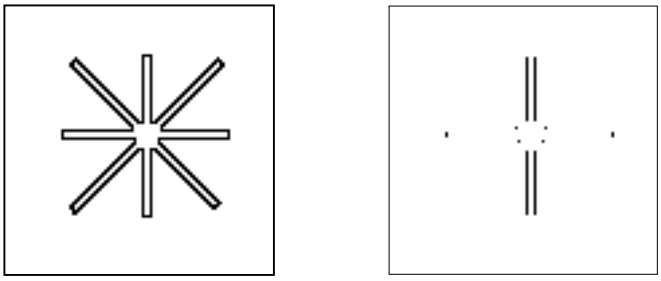


Figura 11.10 Imágenes de entrada y salida.

11.8 PLANTILLAS DE DETECCIÓN DE CONTORNO

Detecta el contorno de la figura de la imagen de entrada en una determinada dirección. En la biblioteca se incluyen detectores en cuatro direcciones, arriba, abajo, derecha e izquierda. Las plantillas incluidas son:

- *ContourDetectorUp.tem*
- *ContourDetectorLeft.tem*
- *ContourDetectorRight.tem*
- *ContourDetectorDown.tem*

A continuación se va a analizar la plantilla de detección de contorno derecho. Se puede obtener fácilmente cualquiera de las otras mediante la rotación de la matriz de control *B*.

11.8.1 PLANTILLA *CONTOUR DETECTOR RIGHT*.

11.8.1.1 Características de la plantilla

- Parámetros:

$$A=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad B=\begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & -1 \\ 0 & 0 & 0 \end{pmatrix} \quad I=-2$$

Condición de contorno	Tipo de imagen de entrada	Tipo de imagen de estado inicial	Tipo de imagen de salida
Refleja	*Binaria*	*Espejo*	*Binaria*

- Descripción: Realiza la proyección de la figura de la imagen de entrada hacia la izquierda.

11.8.1.2 Ejemplo de aplicación

A continuación se muestra un ejemplo en el que se pone de manifiesto el comportamiento de esta plantilla.

La imagen de estado inicial está contenida en el fichero *Objects.bmp*.

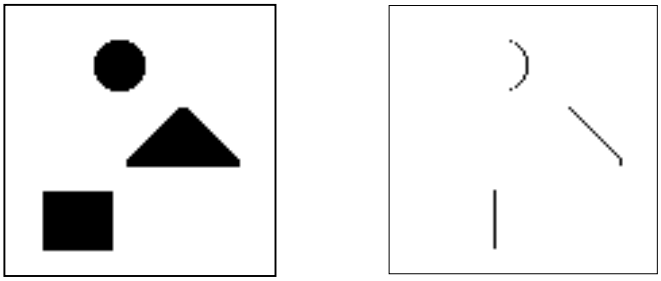


Figura 11.11 Imágenes de estado inicial, y salida.

11.9 PLANTILLAS DE RELLENO

Dada una imagen con una serie de figuras el procesamiento de estas plantillas consiste en mostrar una serie de polígonos en lugar de las figuras, que son los de menor tamaño que contienen dichas figuras. Las características del polígono mostrado depende de la plantilla. La plantilla *Concave Location Filler* muestra polígonos de 8 lados, los cuales deben tener direcciones horizontales, verticales y diagonales. La plantilla *Box Filler* muestra rectángulos, cuyos lados tienen direcciones horizontales y verticales. Y la plantilla *Box Diagonal Filler* también muestra rectángulos, pero con la diferencia de que sus lados tienen direcciones diagonales. Los ficheros que contienen estas tres plantillas son:

- *ConcaveLocationFiller.tem*
- *BoxFiller.tem*
- *BoxDiagonalFiller.tem*

A continuación se analizan cada una de estas tres plantillas

11.9.1 PLANTILLA *CONCAVE LOCATION FILLER*

11.9.1.1 Características de la plantilla

- Parámetros:

$$A=\begin{pmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & 2 & 0.5 \\ 0.5 & 0.5 & 0.5 \end{pmatrix} \quad B=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad I=3.25$$

Condición de contorno	Tipo de imagen de entrada	Tipo de imagen de estado inicial	Tipo de imagen de salida
Fija	*Binaria*	*Espejo*	*Binaria*

- Descripción: Enmarca los objetos de la imagen de estado inicial con polígonos con 8 lados de direcciones horizontal, vertical y diagonales.

11.9.1.2 Ejemplo de aplicación

A continuación se muestra un ejemplo en el que se pone de manifiesto el comportamiento de esta plantilla. La imagen de estado inicial está contenida en el fichero *Objects2.bmp*.

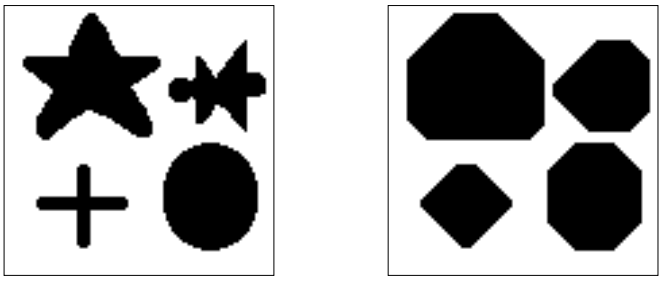


Figura 11.12 Imágenes de estado inicial, y salida.

11.9.2 PLANTILLA *BOX FILLER*

11.9.2.1 Características de la plantilla

- Parámetros:

$$A=\begin{pmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad B=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad I=3.25$$

Condición de contorno	Tipo de imagen de entrada	Tipo de imagen de estado inicial	Tipo de imagen de salida
Fija	*Binaria*	*Espejo*	*Binaria*

- Descripción: Enmarca los objetos de la imagen de estado inicial con rectángulos de lados con direcciones horizontal y vertical.

11.9.2.2 Ejemplo de aplicación

A continuación se muestra un ejemplo en el que se pone de manifiesto el comportamiento de esta plantilla.

La imagen de estado inicial está contenida en el fichero

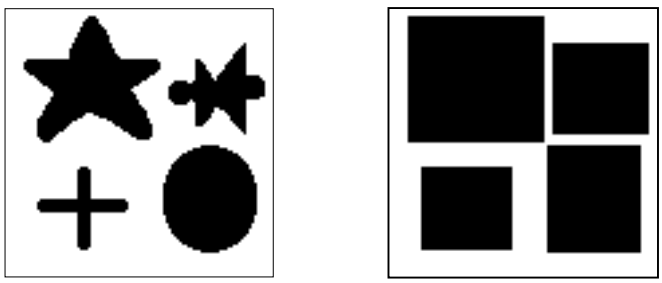


Figura 11.13 Imágenes de estado inicial, y salida.

11.9.3 PLANTILLA *BOX DIAGONAL FILLER*

11.9.3.1 Características de la plantilla

- Parámetros:

$$A=\begin{pmatrix}1&0&1\\0&2&0\\1&0&1\end{pmatrix} \quad B=\begin{pmatrix}0&0&0\\0&2&0\\0&0&0\end{pmatrix} \quad I=3.25$$

Condición de contorno	Tipo de imagen de entrada	Tipo de imagen de estado inicial	Tipo de imagen de salida
Fija	*Binaria*	*Espejo*	*Binaria*

- Descripción: Enmarca los objetos de la imagen de estado inicial con rectángulos de lados con direcciones diagonales.

11.9.3.2 Ejemplo de aplicación

A continuación se muestra un ejemplo en el que se pone de manifiesto el comportamiento de esta plantilla.

La imagen de estado inicial está contenida en el fichero

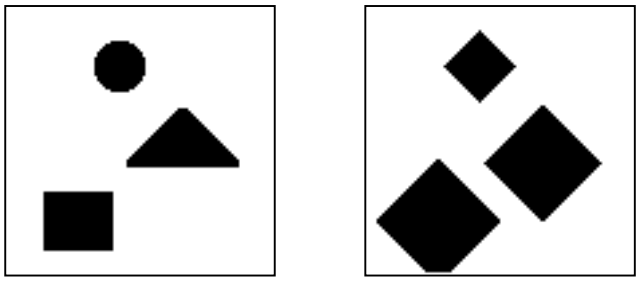


Figura 11.14 Imágenes de estado inicial, y salida.

11.10 PLANTILLAS DIFERENCIA CON UMBRAL

Estas plantillas sirven para estimar diferencias entre dos imágenes en escala de grises píxel a píxel. El resultado es una imagen de salida que en cada píxel muestra un valor blanco o negro dependiendo si el resultado de la diferencia entre píxeles de la imagen de entrada es menor o mayor que el umbral. En la diferencia se suma un umbral para tener un cierto margen de diferencia, y que no cualquier pequeña variación se muestre en la salida. La plantilla de diferencia con umbral genérico es la siguiente:

$$A=\begin{pmatrix}0&0&0\\0&2&0\\0&0&0\end{pmatrix} \quad B=\begin{pmatrix}0&0&0\\0&-1&0\\0&0&0\end{pmatrix} \quad I=Umbral$$

Hay que tener en cuenta que la plantilla no es transitiva, que el resultado de la diferencia puede ser positivo o negativo, y que el signo del umbral es importante. La operación realizada píxel a píxel se puede resumir en:

$$ImagenEstadoInicial - ImagenEntrada + umbral \begin{cases} >0 \ \text{ImagenSalida} \bullet \text{negro} \\ <0 \ \text{ImagenSalida} \bullet \text{blanco} \end{cases}$$

De manera que debido a la naturaleza lineal de la red solamente se pueden detectar las diferencias con un determinado signo. Para determinar diferencias en valor absoluto se deberá realizar la combinación de varias plantillas.

Es recomendable realizar el suavizado de las imágenes antes de realizar la diferencia, para evitar que cualquier imprecisión o ruido provoque diferencias. En la biblioteca se incorporan las siguientes plantillas *Difference Threshold* con diferentes umbrales:

- *DifferenceThreshold0.tem*, correspondiente a la plantilla *Difference Threshold* con umbral 0.
- *DifferenceThreshold01.tem* , correspondiente a la plantilla *Difference Threshold* con umbral 0.1
- *DifferenceThreshold02.tem*, correspondiente a la plantilla *Difference Threshold* con umbral 0.2
- *DifferenceThreshold04.tem*, correspondiente a la plantilla *Difference Threshold* con umbral 0.4
- *DifferenceThreshold06.tem*, correspondiente a la plantilla *Difference Threshold* con umbral 0.6
- *DifferenceThreshold08.tem*, correspondiente a la plantilla *Difference Threshold* con umbral 0.8
- *DifferenceThreshold-01.tem*, correspondiente a la plantilla *Difference Threshold* con umbral -0.1
- *DifferenceThreshold-02.tem*, correspondiente a la plantilla *Difference Threshold* con umbral -0.2
- *DifferenceThreshold-04.tem*, correspondiente a la plantilla *Difference Threshold* con umbral -0.4
- *DifferenceThreshold-06.tem*, correspondiente a la plantilla *Difference Threshold* con umbral -0.6
- *DifferenceThreshold-08.tem* , correspondiente a la plantilla *Difference Threshold* con umbral -0.8

A continuación se analiza la plantilla *Difference Threshold* con umbral –0.1. El resto de plantillas es completamente equivalente.

11.10.1 PLANTILLA DIFFERENCE THRESHOLD CON UMBRAL -0.1

11.10.1.1 Características de la plantilla

- Parámetros:

$$A=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad B=\begin{pmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad I=-0.1$$

Condición de contorno	Tipo de imagen de entrada	Tipo de imagen de estado inicial	Tipo de imagen de salida
Reflected	*En grises*	*En grises*	*Binaria*

- Descripción: Realiza la diferencia píxel a píxel de las imágenes Entrada y Estado Inicial, y en cada uno de ellos da como resultado negro o blanco en función de que esta diferencia sea mayor o menor que 0.1.

11.10.1.2 Ejemplo de aplicación

A continuación se muestra un ejemplo en el que se pone de manifiesto el comportamiento de esta plantilla.

Las imagen de estado inicial y entrada están contenidas en los ficheros *DifferenceA.bmp* y *DifferenceB.bmp* respectimamente.

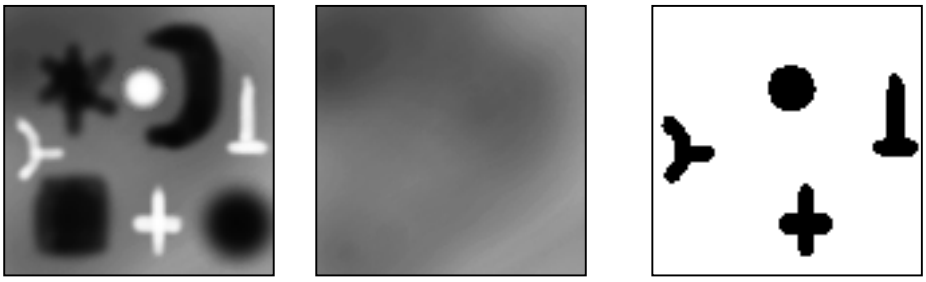


Figura 11.15 Imágenes de entrada, estado inicial y salida.

En caso de invertir el orden de las imágenes, se obtiene el resultado que se muestra en la figura 11.16.

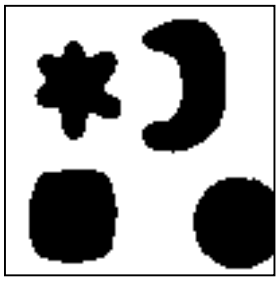


Figura 11.16 Imágenes de salida.

Y en caso de usar como umbral *0.1* en lugar de *–0.1* se obtendrían las imágenes anteriores invertidas.

11.11 PLANTILLAS DETECTORES LOCALES

En estas plantillas se hace una determinada operación en cada píxel y su entorno de vecindad inmediato (radio = 1). Se buscará una determinada morfología local especificada en la matriz de control *B*, de manera que si un píxel de la imagen de entrada verifica dicha morfología, el correspondiente píxel tendrá un valor negro.

La plantilla genérica es la siguiente:

$$A=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad B=\begin{pmatrix} b_1 & b_2 & b_3 \\ b_4 & b_5 & b_6 \\ b_7 & b_8 & b_9 \end{pmatrix} \quad I=x$$

Donde:

$$b_i=\begin{cases} 1, & \text{píxel negro} \\ 0, & \text{indiferente} \\ -1 & \text{píxel blanco} \end{cases} \qquad x = 0.5 - \sum_{i=1}^{9} |b_i|$$

En la biblioteca se incluyen los siguientes ficheros de plantilla:

- *LocalSouthernDetector.tem*, correspondiente a la plantilla *Local Southern Detector*
- *LocalNortherDetector.tem*, correspondiente a la plantilla *Local Northern Detector*
- *LocalEasternDetector.tem*, correspondiente a la plantilla *Local Eastern Detector*
- *LocalWesternDetector.tem*, correspondiente a la plantilla *Local Western Detector*

A continuación se analiza la plantilla *Local Southern Detector.*Las otras tres completamente equivalentes y se obtienen sin más que realizar una rotación en la matriz de control *B.*

11.11.1 PLANTILLA LOCAL SOUTHERN DETECTOR

11.11.1.1 Características de la plantilla

- Parámetros:

$$A=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad B=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & -1 & -1 \end{pmatrix} \quad I=-3.5$$

Condición de contorno	Tipo de imagen de entrada	Tipo de imagen de estado inicial	Tipo de imagen de salida
Reflected	*Binaria*	*Todo-blanco*	*Binaria*

- Descripción: Comprueba la morfología descrita por la matriz *B* píxel a píxel, que en este caso indica píxel central negro y píxeles vecinos inferiores todos blancos, siendo los restantes indiferentes. Se indicará que se ha encontrado dicha morfología mediante un valor negro en el píxel correspondiente de la imagen de salida.

11.11.1.2 Ejemplo de aplicación

A continuación se muestra un ejemplo en el que se pone de manifiesto el comportamiento de esta plantilla.

Las imagen de entrada está contenidas en el fichero *Objects.bmp* . Para la imagen de estado inicial debemos utilizar la contenida en el fichero *White100x100.bmp* .

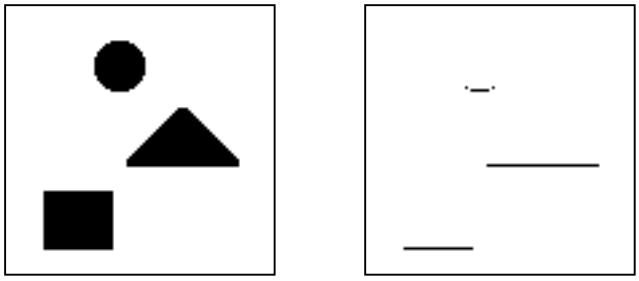


Figura 11.17 Imágenes de entrada y salida.

11.12 PLANTILLA *HOLE FILLER*

11.12.1 CARACTERÍSTICAS DE LA PLANTILLA

- Parámetros:

$$A=\begin{pmatrix} 0 & 1 & 0 \\ 1 & 3 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad B=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad I=-1$$

Condición de contorno	Tipo de imagen de entrada	Tipo de imagen de estado inicial	Tipo de imagen de salida
Fija	*Binaria*	*Todo-blanco con borde negro*	*Binaria*

- Descripción: Rellena los huecos de las figuras de la imagen de entrada.

11.12.2 EJEMPLO DE APLICACIÓN

A continuación se muestra un ejemplo en el que se pone de manifiesto el comportamiento de esta plantilla.

Las imagen de entrada está contenidas en el fichero *HoleFiller.bmp* .Para la imagen de estado inicial debemos utilizar la contenida en el fichero *Black-WhiteBorder.bmp* .

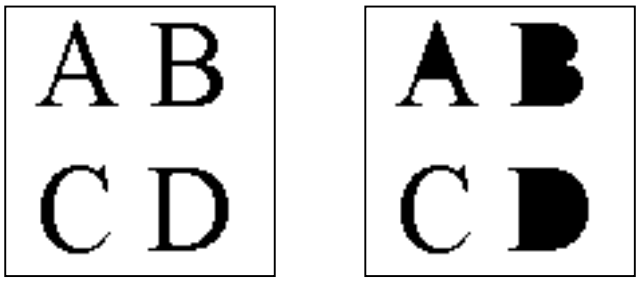


Figura 11.18 Imágenes de entrada y salida.

11.13 PLANTILLA *EROSION*

11.13.1 CARACTERÍSTICAS DE LA PLANTILLA

- Parámetros:

$$A=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad B=\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \quad I=-8.5$$

Condición de contorno	Tipo de imagen de entrada	Tipo de imagen de estado inicial	Tipo de imagen de salida
Refleja	*Binaria*	*Espejo*	*Binaria*

- Descripción: Realiza un erosionado de las figuras de la imagen de entrada, de manera que se muestran a la salida con menor superficie.

11.13.2EJEMPLO DE APLICACIÓN

A continuación se muestra un ejemplo en el que se pone de manifiesto el comportamiento de esta plantilla.

Las imagen de entrada está contenida en el fichero charA.bmp .

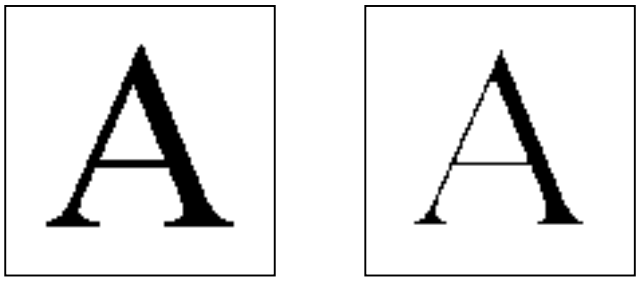


Figura 11.19 Imágenes de entrada, estado inicial y salida.

11.14 PLANTILLA *DITALACION*

11.14.1CARACTERÍSTICAS DE LA PLANTILLA

- Parámetros:

$$A=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad B=\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \quad I=8.5$$

Condición de contorno	Tipo de imagen de entrada	Tipo de imagen de estado inicial	Tipo de imagen de salida
Refleja	*Binaria*	*Espejo*	*Binaria*

- Descripción: Realiza una dilatación de las figuras de la imagen de entrada, de manera que se muestran a la salida con mayor superficie.

11.14.2EJEMPLO DE APLICACIÓN

A continuación se muestra un ejemplo en el que se pone de manifiesto el comportamiento de esta plantilla.

La imagen de entrada está contenida en el fichero *charA.bmp* .

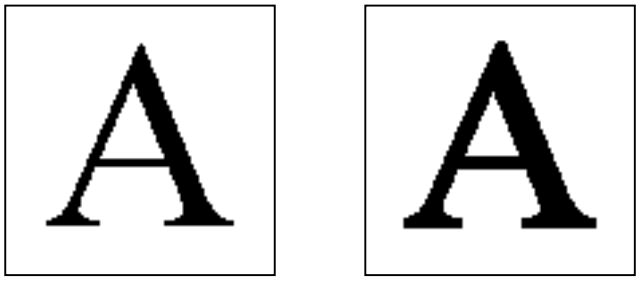


Figura 11.20 Imágenes de entrada y salida.

11.15 PLANTILLA *HALFTONING*.

11.15.1CARACTERÍSTICAS DE LA PLANTILLA

- Parámetros:

$$A=\begin{pmatrix} -0.07 & -0.1 & -0.07 \\ -0.1 & 1.1 & -0.1 \\ -0.07 & -0.1 & -0.07 \end{pmatrix} \quad B=\begin{pmatrix} 0.07 & 0.1 & 0.07 \\ 0.1 & 0.32 & 0.1 \\ 0.07 & 0.1 & 0.07 \end{pmatrix} \quad I=0$$

Condición de contorno	Tipo de imagen de entrada	Tipo de imagen de estado inicial	Tipo de imagen de salida
Refleja	*En grises*	*Espejo*	*Binaria*

- Descripción: Transforma la imagen en grises de la entrada en una imagen binaria de medios tonos.

11.15.2EJEMPLO DE APLICACIÓN

A continuación se muestra un ejemplo en el que se pone de manifiesto el comportamiento de esta plantilla.

La imagen de entrada está contenida en el fichero *Halftoning.bmp* .

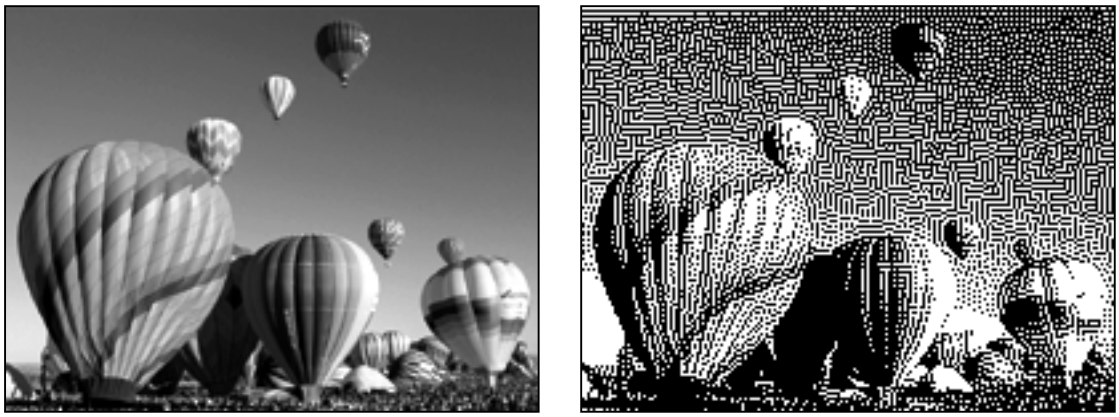


Figura 11.21 Imágenes de entrada y salida.

11.16 PLANTILLA *INVERSE HALFTONING.*

11.16.1CARACTERÍSTICAS DE LA PLANTILLA

- Parámetros:

$$A=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad B=\begin{pmatrix} 0.07 & 0.1 & 0.07 \\ 0.1 & 0.32 & 0.1 \\ 0.07 & 0.1 & 0.07 \end{pmatrix} \quad I=0$$

Condición de contorno	Tipo de imagen de entrada	Tipo de imagen de estado inicial	Tipo de imagen de salida
Refleja	*Binaria*	*Todo-gris neutro*	*Binaria*

- Descripción: Transforma la imagen en medios tonos obtenida por la plantilla *Halftoning* de la entrada en una imagen binaria de medios tonos.

11.16.2EJEMPLO DE APLICACIÓN

A continuación se muestra un ejemplo en el que se pone de manifiesto el comportamiento de esta plantilla.

La imagen de entrada está contenida en el fichero *InverseHalftoning.bmp* .

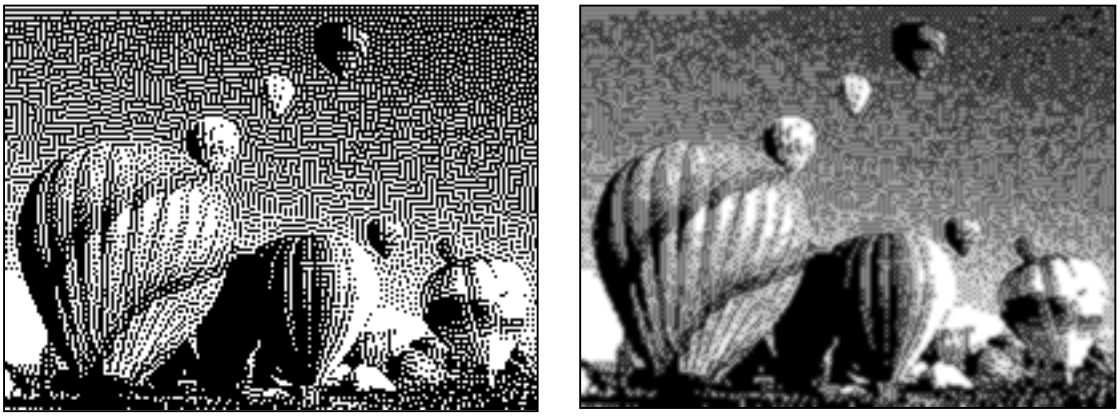


Figura 11.22 Imágenes de entrada y salida.

11.17 PLANTILLA *EDGE DETECTOR*

11.17.1CARACTERÍSTICAS DE LA PLANTILLA

- Parámetros:

$$A=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad B=\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} \quad I=-1$$

Condición de contorno	Tipo de imagen de entrada	Tipo de imagen de estado inicial	Tipo de imagen de salida
Refleja	*Binaria*	*Todo-gris neutro*	*Binaria*

- Descripción: Extrae los bordes de las figuras de la imagen de entrada.

11.17.2EJEMPLO DE APLICACIÓN

A continuación se muestra un ejemplo en el que se pone de manifiesto el comportamiento de esta plantilla.

La imagen de entrada está contenida en el fichero *objects2.bmp* . Para la imagen de estado inicial se debe utilizar la contenida en el fichero *gray100x100.bmp*.

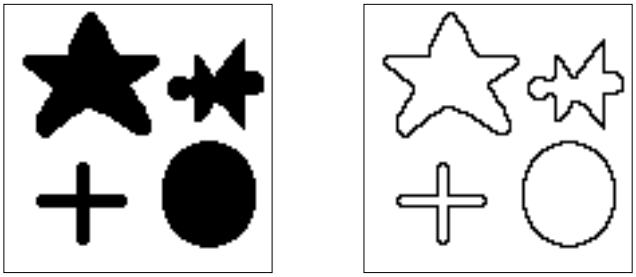


Figura 11.23 Imágenes de entrada y salida.

11.18 PLANTILLA *FIGURE RECONSTRUCTOR*

11.18.1CARACTERÍSTICAS DE LA PLANTILLA

- Parámetros:

$$A=\begin{pmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & 4 & 0.5 \\ 0.5 & 0.5 & 0.5 \end{pmatrix} \quad B=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad I=3$$

Condición de contorno	Tipo de imagen de entrada	Tipo de imagen de estado inicial	Tipo de imagen de salida
Refleja	*Binaria*	*Binaria*	*Binaria*

- Descripción: Extrae los bordes de las figuras de la imagen de entrada.

11.18.2EJEMPLO DE APLICACIÓN

A continuación se muestra un ejemplo en el que se pone de manifiesto el comportamiento de esta plantilla.

La imágenes de estado inicial y entrada están contenidas en los ficheros *FigRecMarks.bmp* y *Objects2.bmp*

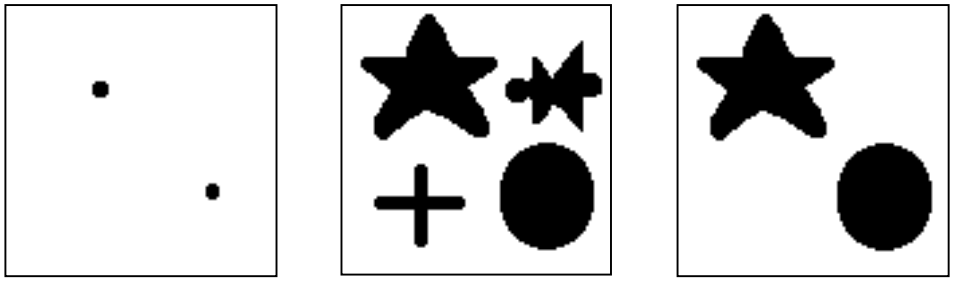


Figura 11.24 Imágenes de estado inicial, entrada y salida.

11.19 PLANTILLA *CONNECTIVITY*

11.19.1CARACTERÍSTICAS DE LA PLANTILLA

- Parámetros:

$$A=\begin{pmatrix} 0 & 0.5 & 0 \\ 0.5 & 3 & 0.5 \\ 0 & 0.5 & 0 \end{pmatrix} \quad B=\begin{pmatrix} 0 & -0.5 & 0 \\ -0.5 & 3 & -0.5 \\ 0 & -0.5 & 0 \end{pmatrix} \quad I=-4.5$$

Condición de contorno	Tipo de imagen de entrada	Tipo de imagen de estado inicial	Tipo de imagen de salida
Refleja	*Binaria*	*Binaria*	*Binaria*

- Descripción: Elimina los objetos marcados en la imagen de entrada. Las imágenes de entrada y estado inicial solamente se diferencian en las marcas, que son píxeles blancos en el estado inicial sobre la figura que se desee eliminar.

11.19.2EJEMPLO DE APLICACIÓN

A continuación se muestra un ejemplo en el que se pone de manifiesto el comportamiento de esta plantilla.

Las imagenes de entrada y estado inicial están contenida en los ficheros *ConectA.bmp* y *ConnectB.bmp* .

Figura 11.25 Imágenes de entrada, estado inicial y salida.

11.20 PLANTILLA *JUNCTION EXTRACTOR*.

11.20.1CARACTERÍSTICAS DE LA PLANTILLA

- Parámetros:

$$A=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad B=\begin{pmatrix} 1 & 1 & 1 \\ 1 & 6 & 1 \\ 1 & 1 & 1 \end{pmatrix} \quad I=-3$$

Condición de contorno	Tipo de imagen de entrada	Tipo de imagen de estado inicial	Tipo de imagen de salida
Refleja	*Binaria*	*Espejo*	*Binaria*

- Descripción: Elimina los píxeles negros que no tengan al menos 3 píxeles vecinos negros. Los píxeles blancos los deja como están. Sirve para extraer los puntos de cruce en una figura esqueletizada. También puede emplearse para realizar pequeñas erosiones en figuras.

11.20.2EJEMPLO DE APLICACIÓN

A continuación se muestra un ejemplo en el que se pone de manifiesto el comportamiento de esta plantilla.

Las imagen de entrada están contenida en el fichero *JunctionExtractor.bmp*. Para el estado inicial se debe utilizar esta misma imagen.

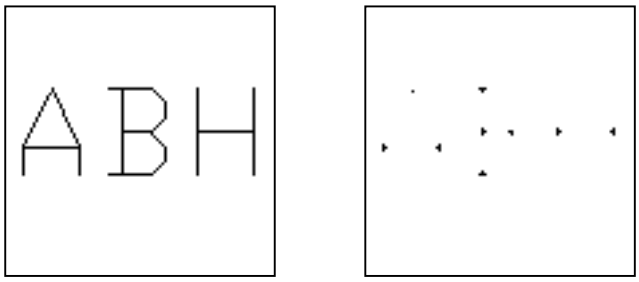


Figura 11.26 Imágenes de entrada, estado inicial y salida.

11.21 PLANTILLA HEAT DIFUSIÓN

11.21.1CARACTERÍSTICAS DE LA PLANTILLA

- Parámetros:

$$A=\begin{pmatrix} 0.1 & 0.15 & 0.1 \\ 0.15 & 0 & 0.15 \\ 0.1 & 0.15 & 0.1 \end{pmatrix} \quad B=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad I=0$$

Condición de contorno	Tipo de imagen de entrada	Tipo de imagen de estado inicial	Tipo de imagen de salida
Refleja	*Todo-gris neutro*	*En grises*	*En grises*

- Descripción: Realiza la operación de difusión de calor sobre la imagen de estado inicial. El resultado es una imagen suavizada con un efecto de desenfoque.

11.21.2EJEMPLO DE APLICACIÓN

A continuación se muestra un ejemplo en el que se pone de manifiesto el comportamiento de esta plantilla.

Las imagen de estado inicial está contenida en el ficheros.

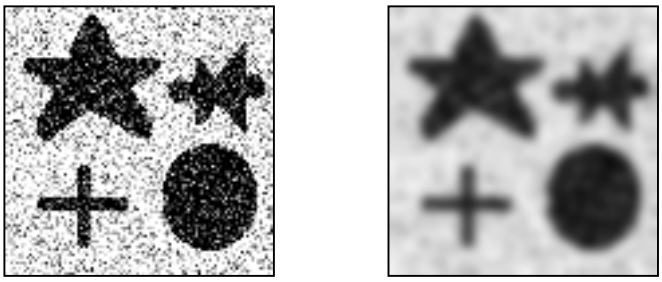


Figura 11.27 Imágenes de estado inicial y salida

11.22 PLANTILLA *BINARY SMOOTHIG*

11.22.1 CARACTERÍSTICAS DE LA PLANTILLA

- Parámetros:

$$A=\begin{pmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad B=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad I=0$$

Condición de contorno	Tipo de imagen de entrada	Tipo de imagen de estado inicial	Tipo de imagen de salida
Refleja	*Todo-gris neutro*	*En grises*	*Binaria*

- Descripción: Realiza una conversión a binario de la imagen de entrada con un suavizado.

11.22.2 EJEMPLO DE APLICACIÓN

A continuación se muestra un ejemplo en el que se pone de manifiesto el comportamiento de esta plantilla.

Las imagen de estado inicial está contenida en el ficheros.

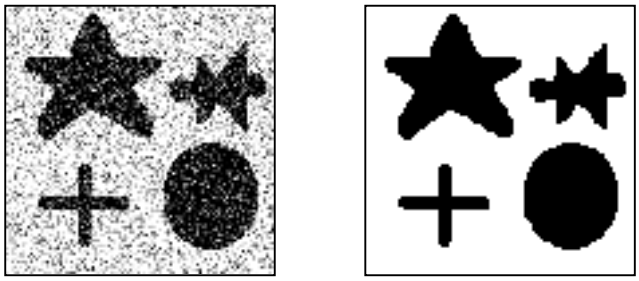


Figura 11.28 Imágenes de estado inicial y salida

11.23 PLANTILLA *GRAY SMOOTHING*

11.23.1CARACTERÍSTICAS DE LA PLANTILLA

- Parámetros:

$$A=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad B=\begin{pmatrix} 0.125 & 0.125 & 0.125 \\ 0.125 & 0.125 & 0.125 \\ 0.125 & 0.125 & 0.125 \end{pmatrix} \quad I=0$$

Condición de contorno	Tipo de imagen de entrada	Tipo de imagen de estado inicial	Tipo de imagen de salida
Refleja	*Todo-gris neutro*	*En grises*	*En grises*

- Descripción: Realiza la operación de difusión de calor sobre la imagen de estado inicial. El resultado es una imagen suavizada con un efecto de desenfoque.

11.23.2EJEMPLO DE APLICACIÓN

A continuación se muestra un ejemplo en el que se pone de manifiesto el comportamiento de esta plantilla.

Las imagen de estado inicial está contenida en el ficheros.

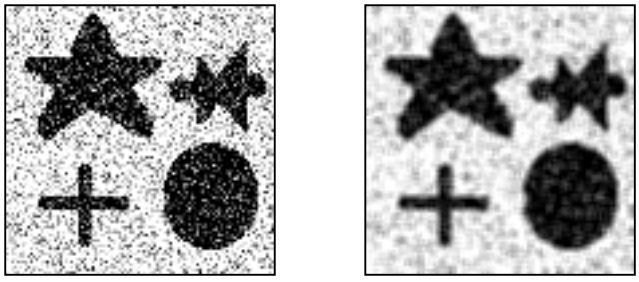


Figura 11.29 Imágenes de entrada y salida

11.24 PLANTILLA *FRAMED AREAS FINDER*

11.24.1CARACTERÍSTICAS DE LA PLANTILLA

- Parámetros:

$$A=\begin{pmatrix} 0 & 1 & 0 \\ 1 & 5 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad B=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad I=-5.25$$

Condición de contorno	Tipo de imagen de entrada	Tipo de imagen de estado inicial	Tipo de imagen de salida
Refleja	*Binaria*	*Binaria*	*Binaria*

- Descripción: Se muestran a la salida las zonas en píxeles negros de la imagen de estado inicial completamente enmarcadas por una curva cerrada de la imagen de entrada. Para que todo funcione bien la curva debe tener un píxel de grosor, y la zona encerrada no debe tener ningún píxel negro.

11.24.2EJEMPLO DE APLICACIÓN

A continuación se muestra un ejemplo en el que se pone de manifiesto el comportamiento de esta plantilla.

Las imágenes de entrada y estado inicial están contenidas en los ficheros .

Figura 11.30 Imágenes de entrada, estado inicial y salida

11.25 PLANTILLA *CONTOUR SMOOTHIG*

11.25.1CARACTERÍSTICAS DE LA PLANTILLA

- Parámetros:

$$A=\begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{pmatrix} \quad B=\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad I=0$$

Condición de contorno	Tipo de imagen de entrada	Tipo de imagen de estado inicial	Tipo de imagen de salida
Refleja	*Binaria*	*Todo-gris neutro*	*Binaria*

- Descripción: Realiza un suavizado del contorno de las figuras de la imagen de estado inicial. Este suavizado se realiza a escala de píxeles, de manera que el suavizado solamente se apreciará a bajas resoluciones.

11.25.2EJEMPLO DE APLICACIÓN

A continuación se muestra un ejemplo en el que se pone de manifiesto el comportamiento de esta plantilla.

La imagen de estado inicial está contenida en el fichero *ContourSmoothing.bmp*. Para la imagen de entrada debemos utilizar el fichero *Gray 100x100.bmp* .

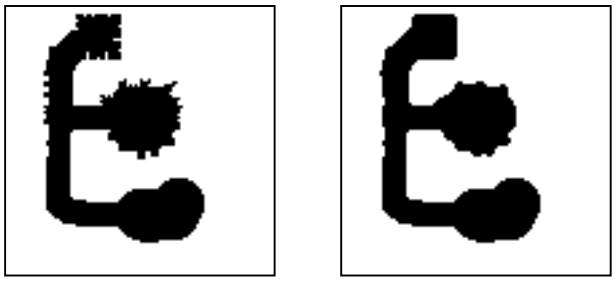


Figura 11.31 Imágenes de estado inicial y salida

12 BIBLIOGRAFÍA

- **[Aizemberg 98];** I. Aizemberg, N. Aizemberg, J. Vandewalle, "Precise Edge Detection: Representation by boolean functions, implementation on the CNN", 1998 Fifth IEEE Int. Workshop on CNNs and their applications proceedings" (IEEE 98 TH8359), pg 301-306 (1998).
- **[Arik 98];** S. Arik, V. Tavsanoglu, "A sufficient condition for the existence of a stable equilibrium point in nonsymmetric Cellular Neural Networks", 1998 Fifth IEEE Int. Workshop on CNNs and their applications proceedings" (IEEE 98 TH8359), pg 74-77 (1998).
- **[Banzhaf 98];** W. Banzhaf, P. Nordin, R. E. Keller and F. D. Francone, "Genetic Programming: An Introduction", Morgan Kaufmann Publishers, Inc. (1998).
- **[Chua 88a];** L.O. Chua, L. Yang, "Cellular neural networks: theory", IEEE Trans. Circuits and Systems, CAS-35, pg. 1257-1272 (1988).
- **[Chua 88b];** L. O. Chua, L. Yang. "Cellular Neural Networks: Applications". IEEE Trans. on Circuits and Systems. Vol. 35, No. 10. pp. 1273-1285 (1998).
- **[Chua 93b];** L.O. Chua, T. Roska, T. Kozek, Á. Zarándy, "The CNN paradigm - A short tutorial", Cellular Neural Networks, ed. Wiley, pg. 2-14 (1993).
- **[Crounse 95];** K. R. Crouse, and L. O. Chua, "Methods for Image Processing and Pattern Formation in the Cellular Neural Netwoks: A tutorial", IEEE CAS-I, Vol.42, no.10 (1995).
- **[Espejo 94];** S. Espejo. "Redes Neuronales Celulares: Modelado y Diseño Monolítico". Tesis doctoral. Universidad de Sevilla, Feb. 1994.
- **[Espejo 96];** S. Espejo, R. Carmona, R. Domínguez-Castro, A. Rodriguez-Vazquez, "A VLSI oriented continuous-time CNN model", Int. Journ. Of Circuit Theory and Applications, vol. 24, pg. 341-356 (1996).
- **[Genç 98];** I. Genç, C. Güzelis, "Threshold class CNNs with Input-dependent Initial State", 1998 Fifth IEEE Int. Workshop on CNNs and their applications proceedings" (IEEE 98 TH8359), pg 130-135 (1998).
- **[Holland 75];** J. Holland, "Adaptation in natural and artificial systems". MIT Press, Cambrige, MA (1975)
- **[Henseler 93];** J. Henseler, P. Braspenning, "Membrain: a cellular neural network model based on a vibrating membrane", Cellular Neural Networks, De. Wiley, pg. 45-58 (1993).
- **[Harrer 93];** H. Harrer, J.A. Nossek, "Discrete-time cellular neural networks", Cellular Neural Networks, ed. Wiley, pg. 15-29 (1993).

- **[Koza 92];** J. R. Koza: "Detailed Description of Genetic Programming". Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, 1992.
- **[Madlenov 98];** V. M. Madlenov, D. Leenaerys, "Estimation of the Basin of attrations of Stabel Equilibrium points in CNNs", 1998 Fifth IEEE Int. Workshop on CNNs and their applications proceedings" (IEEE 98 TH8359), pg 62-67 (1998).
- **[Puffer 98];** F. Puffer, R. Tetzlaff, D. Wolf, "Modeling almost incompressible fluid flow with CNN", 1998 Fifth IEEE Int. Workshop on CNNs and their applications proceedings" (IEEE 98 TH8359), pg 78-82 (1998).
- **[Slot 93];** K. Slot, "Cellular neural network design for solving specific image-processing problems", Cellular Neural Networks, De. Wiley, pg. 191-199 (1993)
- **[Suzuki 93];** H. Suzuki, T. Matsumoto, L.O. Chua, "A CNN Handwritten character recognizer", Cellular Neural Networks, Ed. Wiley, pg. 163-174 (1993)
- **[Vilariño 98];** D.L. Vilariño, D. Cabello, M. Balsi, V.M. Brea, "Image segmentation based on Active Contours using Discrete Time Cellular Neural Networks", 1998 Fifth IEEE Int. Workshop on CNNs and their applications proceedings" (IEEE 98 TH8359), pg 331-336 (1998).