APLICACIÓN DE REDES NEURONALES PARA LA OPTIMIZACION DE LA TRAYECTORIA DE LA HERRAMIENTA EN EL FRESADO DE CAVIDADES COMPLEJAS

EDGAR GALVIS PARRA

FABIO LEONEL GONZALEZ ALMEIDA

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERIAS FISICOMECANICAS

ESCUELA DE INGENIERIA MECANICA

BUCARAMANGA

2012

APLICACIÓN DE REDES NEURONALES PARA LA OPTIMIZACION DE LA TRAYECTORIA DE LA HERRAMIENTA EN EL FRESADO DE CAVIDADES COMPLEJAS

EDGAR GALVIS PARRA
FABIO LEONEL GONZALEZ ALMEIDA

Trabajo de Grado para optar al título de
Ingeniero Mecánico

Director
ISNARDO GONZALEZ JAIMES
Ingeniero Mecánico

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERIAS FISICOMECANICAS
ESCUELA DE INGENIERIA MECANICA
BUCARAMANGA
2012

A Dios, por permitirme llegar a este momento tan especial en mi vida.

A mis padres Pacifico y Lucila†, por su amor comprensión y paciencia.

A mis hermanos, Oscar por su lealtad y compañía en cada etapa del camino recorrido, y Libardo por sus ánimos.

Nathaly, por su amor, apoyo incondicional y desinteresado.

A mis amigos, que nos apoyamos mutuamente en nuestra formación profesional.

FABIO.

A Dios,

A mis padres Guillermo y María Eugenia por su apoyo incondicional.

A mis hermanos, amigos y familiares preocupados por el cumplimiento de esta meta.

Y a todos los que hicieron posible este logro.

EDGAR.

AGRADECIMIENTOS

Los autores expresan sus agradecimientos a:

A la Universidad Industrial de Santander y a la Escuela de Ingeniería Mecánica por la formación dada como profesionales en esta rama de la ciencia.

A Isnardo González Jaimes, Ingeniero Mecánico, director del proyecto y amigo, por su respaldo, confianza y colaboración oportuna.

A todos los amigos que de una u otra forma contribuyeron a nuestra formación integral.

A todos nuestros familiares, que nos motivaron de una u otra forma a culminar este gran trabajo.

CONTENIDO

Pág.

LISTA DE FIGURAS

LISTA DE TABLAS

Pág.

LISTA DE ANEXOS

RESUMEN

TITULO:
APLICACIÓN DE REDES NEURONALES PARA LA OPTIMIZACION DE LA TRAYECTORIA DE LA HERRAMIENTA EN EL FRESADO DE CAVIDADES COMPLEJAS.*

AUTORES:
Edgar Galvis Parra.
Fabio Leonel González Almeida. **

PALABRAS CLAVES:
Fresado de Cavidades, Desbaste, Planeando Trayectoria de Herramienta, Red Neuronal, Mapas Auto-Organizados, CAD/CAM.

DESCRIPCION:
El objetivo de este proyecto es mostrar un nuevo enfoque para la planeación de la trayectoria de la herramienta en el fresado de cavidades complejas para la fase de desbaste. La idea clave es formular el problema de la trayectoria dentro de un TSP (Problema del Vendedor Viajero) así el poderoso método de red neuronal puede ser efectivamente aplicado. Específicamente el método está compuesto de (1) digitalización del área de la cavidad en un número finito de puntos, y (2) un método de red neuronal (llamado SOM-Mapas Auto-Organizados) para encontrar la trayectoria.

El algoritmo del programa se diseño fundamentado en la estructura de la red neuronal de Kohonen, que es típicamente aplicada para el problema del vendedor viajero (TSP).

El resultado es un software de fácil manejo que presenta una eficiente trayectoria de la herramienta (en cuanto a longitud de la trayectoria y retracciones de la herramienta) obtenida para cualquier cavidad formada arbitrariamente con numerosas islas. Los resultados se presentan en un formato claro y de fácil interpretación, que incluye la representación grafica de la trayectoria de la herramienta y longitud de la misma permitiendo guardarlas en cualquier formato de imagen, con independencia del mismo software.

*Proyecto de grado.
**Facultad de Ingenierías Físico-Mecánicas, Escuela de Ingeniería Mecánica, Ing. Isnardo González J.

SUMMARY

TITLE:
APPLICATION OF NEURALS NETWORKS FOR TOOL PATH OPTIMIZATION IN COMPLEX POCKET MILLING.*

AUTHORS:
Edgar Galvis Parra.
Fabio Leonel González Almeida. **

KEY WORDS:
Pocket Milling, Rough Cutting, Tool Path Planning, Neural Network, Self-Organizing Map, CAD/CAM.

DESCRIPTION:
The objective of this project is presents a new approach to rough-cut tool path planning of pocket milling operations. The key idea is to formulate the tool path problem into a TSP (Traveling Salesman Problem) so that the powerful neural network method can be effectively applied. Specifically, the method is composed of (1) digitization of the pocket area into a finite number of tool points, and (2) a neural network method (called a SOM—Self-Organizing Map) for path finding.

The algorithm of the program was designed based in the structure of the net neuronal of Kohonen, which is typically applied for traveling salesman problem (TSP).

The result is software of easy handling that present an efficient tool path (in the sense of path length and tool retraction) can be robustly obtained for any arbitrary shaped pockets with many islands. The results are presented in a clear format and of easy interpretation that includes the graphic representation of the path and length of the tool allowing keeping them in any image format, with independence of the same software.

* Degree Project.
** Physical-mechanical Engineeries Faculty, Mechanical Engineering, Eng. Isnardo González J.

INTRODUCCION

Una de las principales fuentes del desarrollo de los pueblos, es hoy día, el uso y la apropiación que estos hagan de sus recursos tecnológicos. Una herramienta útil para posicionar nuestra industria metalmecánica como una de las más pujantes es el desarrollo de sistemas de información basados en los conceptos de tecnología de grupo, células de fabricación y aplicación de redes neuronales, que sean totalmente adaptados a nuestro medio.

Maquinar cavidades es una de las mayores operaciones de eliminación de material en Control Numérico Computarizado (CNC) principalmente con dados y moldes. La forma de la cavidad varía desde una simple geometría a formas complejas, incluyendo islas, que son típicamente maquinadas en tres etapas: desbaste, corte intermedio y acabado. Debido a que la mayoría del volumen es eliminado en el desbaste, su efecto sobre el tiempo de maquinado es significante, y por lo tanto, la incorporación de una eficiente planeación de la trayectoria de la herramienta tiene que ser uno de los puntos cruciales en los sistemas CAD/CAM contemporáneos.

En este proyecto hacemos referencia a los diferentes tiempos involucrados en la fabricación de piezas (tiempos de pasada, ocupación de maquinas, fabricación, etc.), que son determinantes para el fresado de cavidades complejas, además de exponer los conceptos de maquinas de fresado, sus tipos, características principales y parámetros esenciales. En el fresado de cavidades complejas uno de los problemas más significativos en la optimización de condiciones de mecanizado es el de encontrar una trayectoria adecuada para la remoción de material, por esta razón hacemos un mayor énfasis en la optimización del tiempo principal (tp) aplicando la tecnología de redes neuronales para la solución del problema de la trayectoria de la herramienta.

Desde la aparición de las primeras ideas sobre redes neuronales artificiales a finales de la década de los 50, no se había observado un interés tan grande en su investigación y desarrollo en todo el mundo como en la actualidad. Incluso en nuestro país se aprecia como muchos programas educativos en universidades están incluyendo dentro de sus planes de estudio la tecnología de redes neuronales artificiales. Este auge se debe principalmente a los buenos resultados obtenidos por las redes en campos donde las técnicas convencionales no son aplicables o bien su aplicación no ofrece resultados satisfactorios.

En el tiempo actual donde cada vez se necesita ir más rápido y de una manera optima, el presente trabajo ofrece una gran alternativa en el campo de la industria metalmecánica.

En el presente documento se describe la elaboración de un software que facilita el proceso de planeación de la trayectoria de la herramienta en el fresado de cavidades complejas y reduce la interferencia humana.

1. TEORIA GENERAL DEL FRESADO

1.1 FRESADO[1]

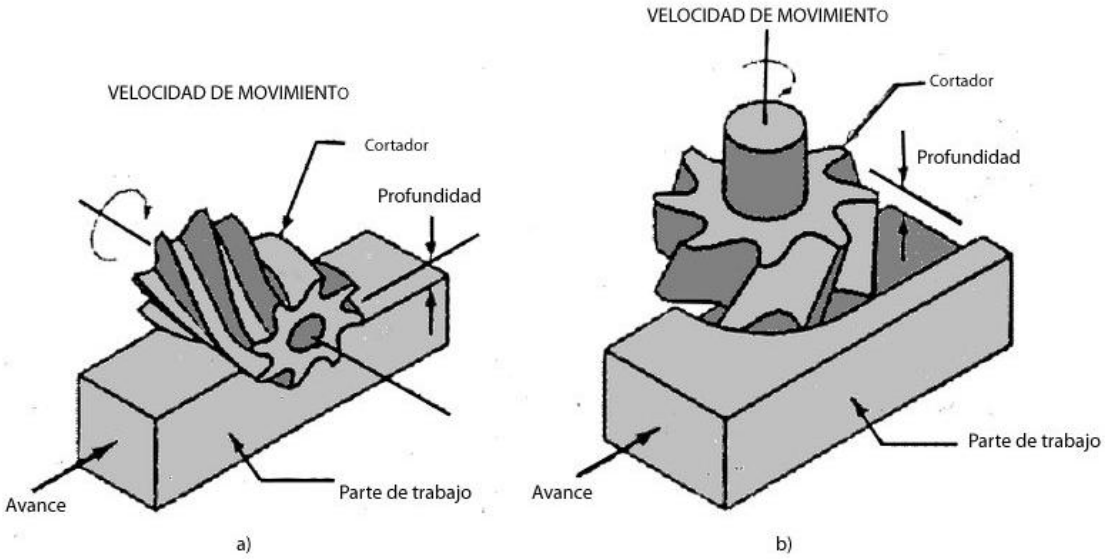
El fresado es una operación de maquinado en la cual se hace pasar una parte de trabajo enfrente de una herramienta cilíndrica rotatoria con múltiples bordes o filos cortantes (en algunos casos raros se usa una herramienta con un solo filo cortante llamado cortador volante). El eje de rotación de la herramienta cortante es perpendicular a la dirección de avance. La orientación entre el eje de la herramienta y la dirección de avance es la característica que distingue al fresado del taladrado. En el taladrado, la herramienta de corte avanza en dirección paralela a su eje de rotación. La herramienta de corte en fresado se llama fresa o cortador para fresadora y los bordes cortantes se llaman dientes. La máquina herramienta que ejecuta tradicionalmente esta operación es una fresadora. La forma geométrica así creada por el fresado es una superficie plana. Se pueden crear otras formas mediante la trayectoria de la herramienta de corte o la forma de dicha herramienta. Debido a la variedad de formas posibles y a sus altas velocidades de producción, el fresado es una de las operaciones de maquinado más versátiles y ampliamente usadas.

El fresado es una operación de corte interrumpido; los dientes de la fresa entran y salen del trabajo durante cada revolución. Esto interrumpe la acción de corte y sujeta los dientes a un ciclo de fuerzas de impacto y choque térmico en cada rotación. El material de la herramienta y la geometría del cortador deben diseñarse para soportar estas condiciones.

1.1.1 Tipos de operaciones de fresado. Hay dos tipos básicos de operaciones de fresado como se muestra en la figura 1: (a) fresado periférico y (b) fresado en las caras.

[1] GROOVER, Mikell. P. Fundamentos de Manufactura Moderna. México: Prentice Hall, 1997.

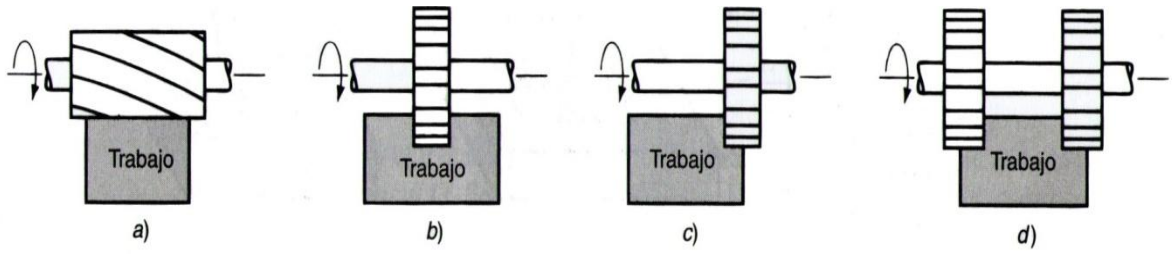
Figura 1. Dos tipos básicos de la operación de fresado a) Fresado periférico y b) Fresado frontal



Fuente: GROOVER, Mikell. P. Fundamentos de Manufactura Moderna. México: Prentice Hall, 1997.

- **Fresado periférico**. En el fresado periférico, el eje de la herramienta es paralelo a la superficie que se está maquinando y la operación se realiza por los bordes de corte en la periferia exterior del cortador. En la figura 2 se muestran varios tipos de fresado periférico: (a) fresado de placa, la forma básica de fresado periférico en la cual el ancho de la fresa se extiende más allá de la pieza de trabajo en ambos lados; (b) ranurado, también llamado fresado de ranuras, en el cual el ancho de la fresa es menor que el ancho de la pieza de trabajo, creando una ranura en el trabajo (cuando la fresa es muy delgada se puede usar esta operación para tallar ranuras angostas o para cortar una parte de trabajo en dos, llamado fresado aserrado); (c) fresado lateral, en la cual la fresa maquina el lado de una pieza de trabajo; y (d) fresado paralelo simultaneo, el cual es el mismo que el fresado natural, excepto que el corte tiene lugar en ambos lados del trabajo.

Figura 2. Fresado periférico: (a) fresado de placa, (b) ranurado, (c) fresado lateral y (d) fresado paralelo simultaneo



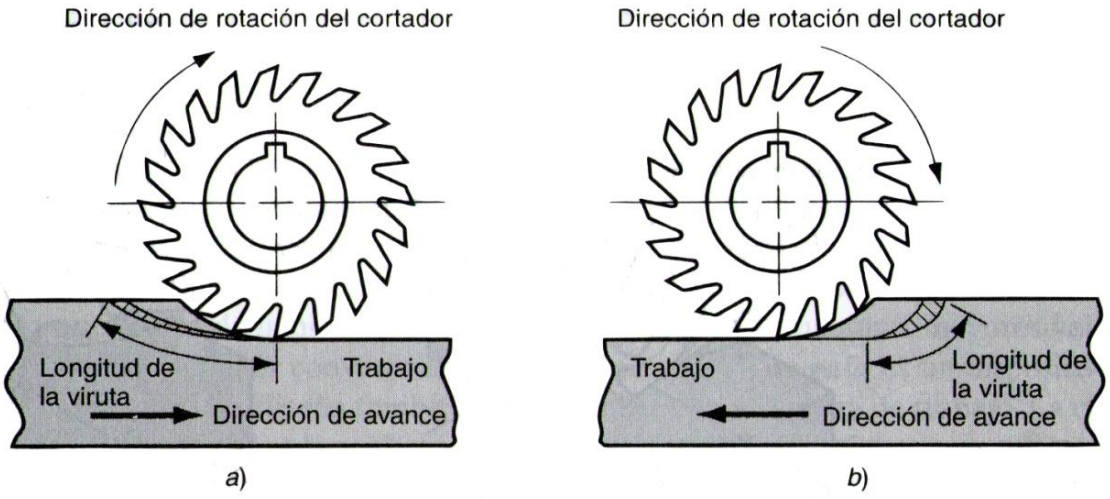
Fuente: Ibíd. Pág. 612.

En el fresado periférico hay dos direcciones opuestas de rotación que puede tener la fresa con respecto al trabajo. Estas direcciones distinguen dos formas de fresado, fresado ascendente y fresado descendente que se ilustra en la figura 3. En el fresado ascendente, también llamado fresado convencional, la dirección del movimiento de los dientes de la fresa es opuesto a la dirección de avance cuando cortan el trabajo. Es decir, cortan ``contra el avance´´. El fresado descendente, también llamado fresado tipo escalamiento, la dirección del movimiento de la fresa es la misma que la dirección de avance cuando los dientes cortan el trabajo. Es un fresado ``con el avance´´.

La geometría relativa de estas dos formas de fresado tiene sus diferencias en las acciones de corte. En el fresado ascendente, la viruta formada por cada diente del cortador comienza muy delgada y aumenta su espesor durante el paso del diente. En el fresado descendente, cada viruta empieza gruesa y se reduce a través del corte. La longitud de una viruta en el fresado descendente es menor que en el fresado ascendente (en la figura la diferencia esta exagerada para mayor comprensión). Esto significa una reducción en el tiempo de trabajo por volumen de material cortado, lo cual tiende a incrementar la vida de la herramienta en el fresado descendente.

La dirección de las fuerzas de corte difiere en el fresado ascendente y descendente. La dirección de la fuerza de corte es tangencial a la periferia de la fresa para los dientes que están enganchados en el trabajo. En el fresado ascendente hay una tendencia a levantar la parte de trabajo al salir los dientes del cortador del material. En el fresado descendente la dirección de la fuerza de corte es hacia abajo, y por esa causa el trabajo se mantiene contra la mesa de la máquina de fresado.

Figura 3. Tipos de fresado (a) Fresado ascendente y (b) fresado descendente



Fuente: Ibíd. Pág. 612.

- **Fresado en las caras o Fresado frontal.** En el fresado frontal, el eje de la pieza es perpendicular a la superficie de trabajo y el maquinado se ejecuta por los bordes o filos cortantes del extremo y la periferia de la fresa. Cuando el diámetro de la fresa es más grande que el ancho de la parte de trabajo, de tal manera que la fresa sobrepasa al trabajo en ambos lados, se denomina fresado frontal convencional, el cual se ilustra en la figura 4(a). De igual manera que en el fresado periférico, también en el fresado frontal existen diversas formas, varias de ellas se ilustran en la figura 4(b): fresado parcial de caras o parcial frontal en la cual la fresa sobrepasa al trabajo solamente en un lado; (c) fresado terminal, en el cual el

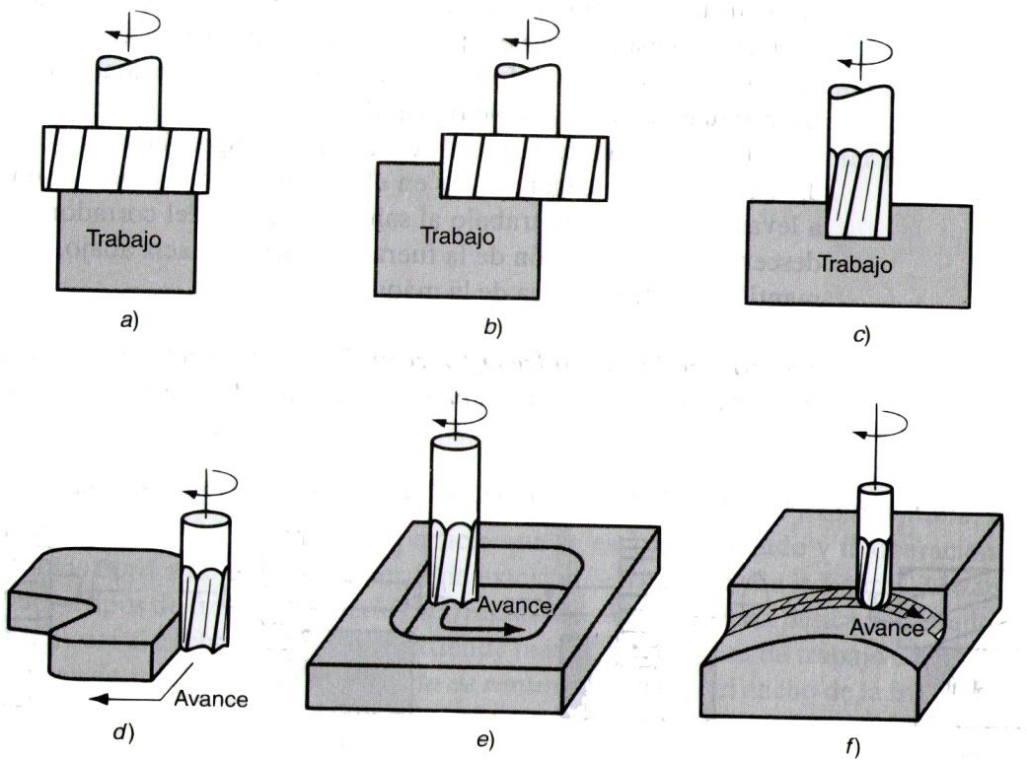
ancho de la fresa es menor que el ancho del trabajo, de manera que se corta una ranura dentro de la parte; (d) el fresado de perfiles es una forma de fresado terminal en el cual se corta una parte plana de la periferia; (e) **fresado de cavidades**, otra forma de fresado terminal usado para fresar cavidades poco profundas en piezas planas; (f) fresado de contorno superficial, en el cual una fresa con punta de bola (en lugar de una fresa cuadrada) se hace avanzar hacia delante y hacia atrás y hacia un lado y otro del trabajo, a lo largo de una trayectoria curvilínea a pequeños intervalos para crear una superficie tridimensional. Se requiere el mismo control básico para maquinar los contornos de moldes y datos en cuyo caso esta operación se llama tallado o contorneo de dados.

1.1.2 Fresas. La clasificación de los cortadores para fresadoras o fresas como se les conoce comúnmente, está muy asociada con las operaciones de fresado que acabamos de describir. Los tipos de fresas incluyen los siguientes:

- **Cortadores cilíndricos o fresas planas.** Estos se usan en el fresado periférico de planchas. Como se indica en la figura 1(a), son fresas cilíndricas con varias filas de dientes. Los bordes cortantes se orientan por lo general en un ángulo de hélice para reducir el impacto de la entrada en el trabajo; estas fresas se llaman cortadores helicoidales.

- **Cortadores formadores o fresas formadoras.** En estos cortadores periféricos, los bordes cortantes tienen un perfil especial que imparten el trabajo. Una aplicación importante está en la fabricación de engranes, en el cual la fresa formadora tiene una forma que corta las ranuras entre los dientes adyacentes de los engranes, formando de esta manera la geometría del diente del engrane.

Figura 4. Fresado frontal: (a) fresado frontal convencional, (b) fresado de frente parcial, (c) fresado terminal, (d) fresado de perfiles, (e) fresado de cavidades y (f) fresado de contorno superficial



Fuente: Ibíd. Pág. 613.

- **Cortadores frontales o fresas frontales.** Estos se diseñan con dientes que cortan tanto lateralmente como en la periferia de la fresa. Las fresas frontales se pueden hacer de acero de alta calidad como se muestra en la figura 1 (b), o se pueden diseñar para usar insertos de carburo cementado.

- **Cortadores para acabado o fresa terminal.** Como se muestra en la figura 4 (c), una fresa terminal se parece a una broca, pero si la observamos con más atención está diseñada para un corte primario con los dientes periféricos más que con su extremo una broca corta solamente en su extremo al penetrar en el trabajo. Las fresas terminales se diseñan con extremos cuadrados, extremos con radio y extremos de bola. Los extremos pueden usarse para fresado frontal, fresado de

perfiles y cavidades, cortar ranuras, grabar, fresar contornos superficiales y tallar dados.

1.1.3 Condiciones de corte en fresado. La velocidad de corte se determina con el diámetro exterior de la fresa. Esta se puede convertir a la velocidad de rotación del husillo usando una fórmula que por ahora debe ser familiar:

$$N = v/(\pi * Do) \quad [rev/min] \qquad (1)$$

El avance f en fresado se determina por lo general como el avance por diente cortante, llamado carga de viruta, y representa el tamaño de la viruta formada por cada filo de corte. Esto se puede convertir a velocidad de avance tomando en cuenta la velocidad del husillo y el número de dientes en la fresa como sigue:

$$fr = N*nt*f \qquad (2)$$

Donde fr = velocidad de avance en pulg/min (mm/min); N = velocidad del husillo en rev/min; nt = numero de dientes de la fresa; y f = carga de viruta en pulg/diente (mm/diente).

La remoción de material en el fresado se determina usando el producto del área de la sección transversal del corte por la velocidad de avance. Por consiguiente, si una operación de fresado de una plancha corta una pieza de trabajo con un ancho w a una profundidad d, la velocidad de remoción de material es

$$MRR = w*d*fr \qquad (3)$$

Esto ignora la entrada inicial de la fresa antes de su enganche completo. La ecuación (3) se puede aplicar al fresado de acabado, fresado lateral, fresado frontal y otras operaciones de fresado, haciendo los ajustes apropiados en el cálculo del área de la sección recta de corte. El tiempo requerido para fresar una pieza de trabajo de longitud L debe tener en cuenta la distancia de aproximación

requerida para enganchar completamente la fresa. Considérese primero el caso del fresado de una plancha (figura 5). Para determinar el tiempo de ejecución de una operación de fresado de la plancha, la distancia de aproximación A para alcanzar la velocidad de corte completo se determina mediante

$$A = (d*(D-d)) \wedge (1/2) \qquad (4)$$

Donde d = profundidad de corte, pulg (mm); D = diámetro de la fresa, pulg (mm). El tiempo para fresar la pieza de trabajo Tm es por tanto

$$Tm = (L+A)/fr \qquad (5)$$

Para el fresado frontal es costumbre dejar para la aproximación la distancia A mas una distancia O, que representa la profundidad de desbaste inicial. Hay dos casos posibles, como se muestra en la figura 6. En ambos casos A = O. El primer caso cuando la fresa se centra sobre la pieza de trabajo rectangular. En la figura 6(a) es evidente que A y O son iguales a la mitad del diámetro del cortador. Esto es:

$$A = O = D/2 \qquad (6)$$

Donde D = diámetro de la fresa, pulg (mm).

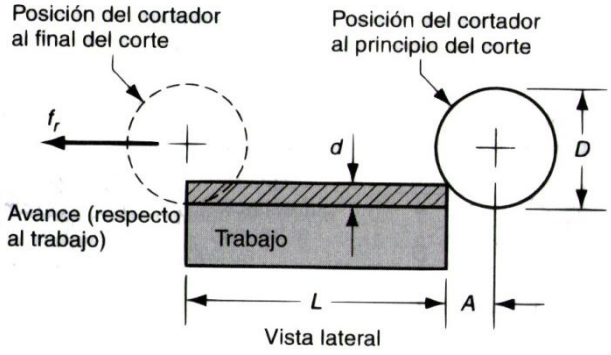
El segundo caso es cuando la fresa sobresale a uno de los lados del trabajo, como se muestra en la figura 6 (b). En este caso, las distancias de aproximación y la distancia adicional están dadas por

$$A = O = (w*(D-w)) \wedge (1/2) \qquad (7)$$

Donde w = ancho del corte, pulg (mm). Por tanto, el tiempo de maquinado en cada caso está dado por:

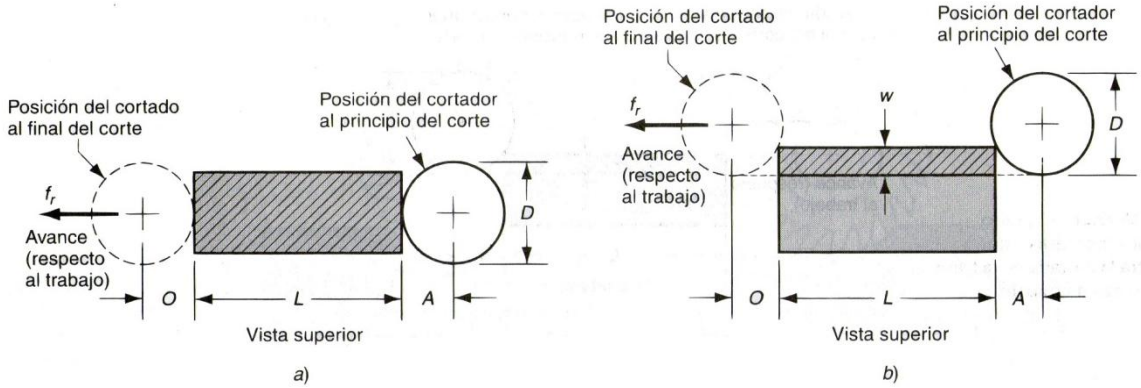
$$Tm = (L+2A)/fr \qquad (8)$$

Figura 5. Fresado de placa mostrando la entrada de la fresa en la pieza de trabajo



Fuente: Ibíd. Pág. 616.

Figura 6. Fresado frontal mostrando las distancias de aproximación y de recorrido adicional para dos casos: (a) cuando el fresador está centrado sobre la pieza de trabajo y (b) cuando el cortador esta desplazado hacia un lado del trabajo



Fuente: Ibíd. Pág. 617.

1.1.4 Maquinas fresadoras. Las maquinas fresadoras deben tener el husillo rotatorio para el cortador y una mesa para sujetar, poner en posición y hacer avanzar la parte de trabajo. Varios diseños de maquinas herramientas satisfacen estos requerimientos. Para empezar, las maquinas fresadoras se pueden clasificar en horizontales y verticales. Una maquina fresadora horizontal tiene un husillo horizontal, y este diseño es adecuado para realizar el fresado periférico (por ejemplo, fresado de planchas, ranurado, y fresado lateral y atravesado) sobre

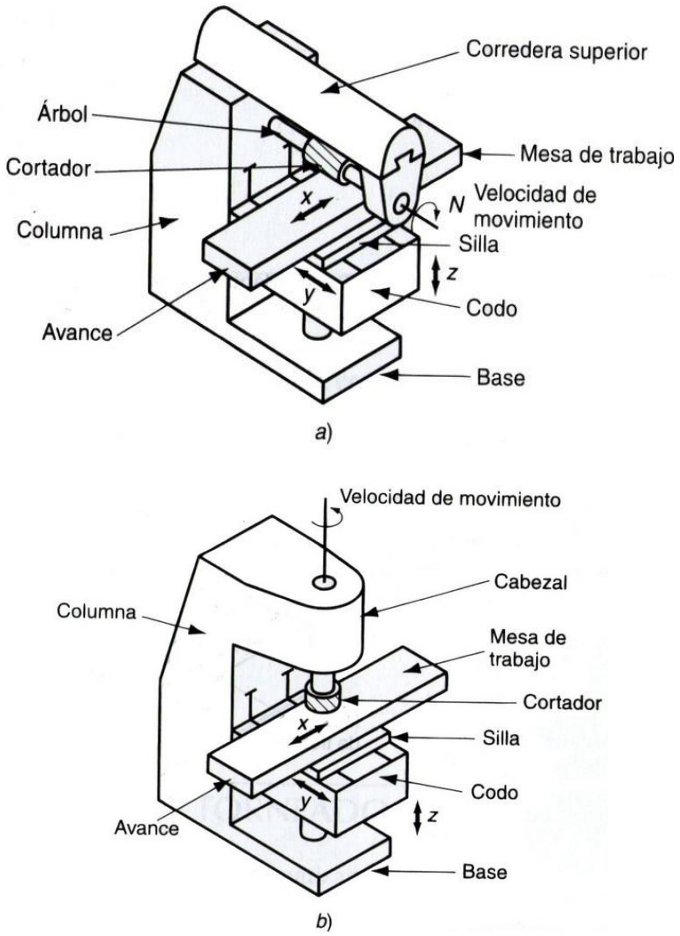
piezas de trabajo que tienen forma aproximadamente cúbica. Una maquina fresadora vertical tiene el husillo vertical, y esta orientación es adecuada para fresado frontal, fresado de acabado, fresado de contorno de superficie y tallado de dados sobre piezas de trabajo relativamente planas.

- **Maquinas fresadoras de rodilla y columna.** La maquina fresadora de rodilla y columna es la máquina herramienta básica para fresado. Deriva su nombre del hecho que sus dos principales componentes son una columna que soporta el husillo y una rodilla (se parece a una rodilla humana) que soporta la mesa de trabajo. Se puede disponer de maquinas horizontales y verticales, como se muestra en la figura 7. En la versión horizontal, un árbol soporta generalmente la fresa. El árbol es básicamente una fresa que sostiene el cortador y se acciona mediante el husillo principal. En las maquinas horizontales se provee un brazo para sostener el árbol. En las maquinas de rodilla y columna verticales los cortadores se pueden montar directamente en el husillo principal.

Una característica de las maquinas fresadoras de rodilla y columna que las hace tan versátiles es la capacidad de la mesa de trabajo para hacer avanzar el trabajo en cualquiera de los tres ejes X, Y o Z. Estas direcciones de los ejes se indican en la figura. La mesa de trabajo se puede mover en la dirección X, la silla se puede mover en la dirección Y, y la rodilla se puede mover verticalmente para lograr el movimiento Z.

- **Fresadora tipo bancada.** Las maquinas fresadoras tipo bancada se diseñan para la producción en masa. Están construidas con mayor rigidez que las maquinas de rodilla y columna, y permiten las velocidades de avance más críticas y las profundidades de corte que se necesitan para las altas velocidades de remoción de material. La construcción característica de las maquinas fresadoras tipo cama se muestra en la figura 8. La mesa de trabajo está montada directamente a la cama de la máquina herramienta en lugar del tipo menos rígido de rodilla y columna.

Figura 7. Dos tipos básicos de maquinas fresadoras de rodilla y columna: (a) horizontal y (b) vertical

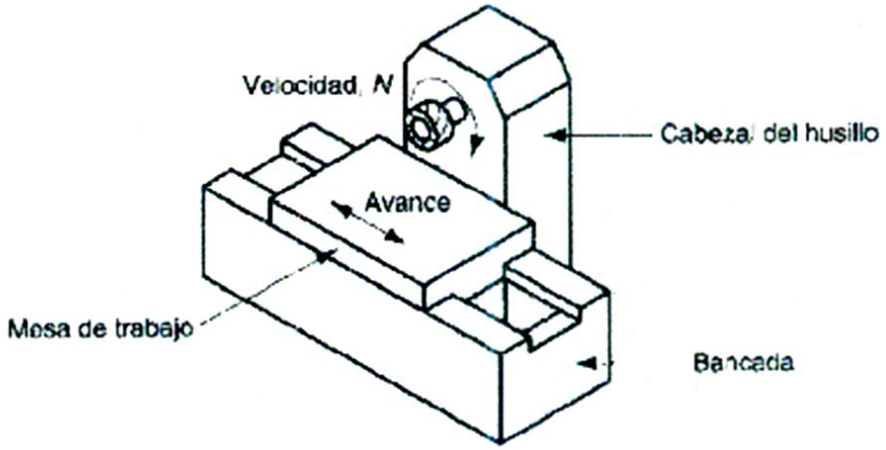


Fuente: Ibíd. Pág. 617.

Esta construcción limita el posible movimiento longitudinal de la mesa para pasar el trabajo por delante de la fresa. La fresa está montada en un cabezal de husillo que puede ajustarse verticalmente a lo largo de la columna de la maquina. Las maquinas de bancada con un solo husillo se llaman maquinas simplex, como se muestra en la figura 8, y están disponibles en modelos verticales u horizontales. Las fresadoras dúplex usan dos cabezales de husillo, los cuales se posicionan por lo general horizontalmente sobre los lados opuestos de la cama para realizar operaciones simultáneas durante un avance del trabajo. Las maquinas triplex

añaden un tercer husillo montado verticalmente sobre la cama para darle mayor capacidad a la maquina.

Figura 8. Maquina fresadora tipo cama simplex de husillo horizontal



Fuente: Ibíd. Pág. 618.

- **Maquinas tipo cepillo.** Las maquinas tipo cepillo forman la categoría más grande de maquinas fresadoras. Su apariencia general y su construcción son las de un cepillo grande; la diferencia es que en lugar del cepillado llevan a cabo el fresado. Por consiguiente, uno o más cabezales de fresado sustituyen a las herramientas de corte de una sola punta que se usan en los cepillos, y el movimiento del trabajo que pasa enfrente de la herramienta es un movimiento de velocidad de avance mas que un movimiento de velocidad de corte. Las fresas tipo cepillo se construyen para maquinar piezas muy grandes. La mesa de trabajo y la cama de la maquina son pesadas y relativamente bajas, casi al ras del piso, y los cabezales fresadores se sostienen sobre una estructura puente que se extiende a través de la mesa.

- **Maquinas fresadoras CNC.** En las maquinas fresadoras CNC la trayectoria de la fresa se controla por datos numéricos en lugar de plantillas físicas. Las maquinas fresadoras CNC están adaptadas especialmente para el

fresado de perfiles, fresado de cavidades, fresado de contorno de superficies y operaciones de tallado de dados, en las que se debe controlar simultáneamente dos o tres ejes de la mesa de trabajo. Normalmente se requiere el operador para cambiar las fresas y cargar y descargar las piezas de trabajo.

A continuación se describen los factores más importantes presentes en el fresado de cavidades complejas para el proceso en el cual esta operación es más utilizada; este proceso es la fabricación de moldes y matrices, también se mencionan los centros de fresado de alta velocidad utilizados en la actualidad.

1.2 ASPECTOS IMPORTANTES EN EL AREA DE FABRICACION DE MOLDES Y MATRICES

1.2.1 Factores que influyen en la maquinabilidad del material.

- **La composición química** del acero es importante. Cuanto mayor sea el contenido de la aleación del acero mayor será su dificultad para mecanizarlo. La maquinabilidad disminuye cuando se incrementa el contenido de carbono.

- **La estructura** del acero también es importante para la maquinabilidad. Las diferentes estructuras incluidas son: forjado, fundido, extrusión, laminado y mecanizado. Las forjas y fundiciones pueden tener también estructuras superficiales de difícil mecanización.

- **La dureza** es uno de los factores principales que afectan a la maquinabilidad. La regla general es que cuanto mayor dureza tenga el acero más difícil será de mecanizar. Puede utilizarse HSS hasta 330-400 HB, HSS + TiN hasta 45 HRC. Hasta 65-70 HRC debe utilizarse metal duro, cerámica, cermet y CBN.

- **Inclusiones no metálicas**, generalmente tienen una mala influencia en la vida de la herramienta. Un ejemplo es el Al2O3 (óxido de aluminio), aunque, siendo cerámica pura, resulta muy abrasivo.

- **Finalmente, las tensiones residuales** pueden causar problemas de maquinabilidad. A menudo es recomendable estabilizar las tensiones después de una operación de desbaste.

1.2.2 Distribución de los costos de producción de un molde o matriz. Aproximadamente, los costos se distribuyen de la siguiente manera:

- Mecanizado 65%
- Material de la pieza 20%
- Tratamiento térmico 5%
- Montaje/ajuste 10%

Lo que muestra claramente la importancia que tiene disponer de una buena maquinabilidad, así como una adecuada y económica solución de mecanizado en la producción de moldes y matrices.[2]

1.2.3 Tipos de operaciones principales y más comunes en la fabricación de moldes y matrices. El proceso de mecanizado debería realizarse al menos en 3 tipos de operaciones: Desbaste, semiacabado y acabado, e incluso algunas veces superacabado. Por supuesto, las operaciones de represado son realizadas después de un semiacabado, como una preparación para el acabado. Es muy importante esforzarse en cada operación para obtener un resultado de mecanizado satisfactorio, lo que conducirá a distribuir correctamente los surcos en el material y preparar la siguiente operación y herramienta (asignación). La vida de

[2] Moldes y matrices [on line]. actualización: 2003-05-07. <www.coromant.sandvik.com>.

la herramienta aumenta y es más predecible en un proceso de mecanizado en el que se realizan los mínimos cambios posibles de la trayectoria y de la carga de la herramienta. Si es posible, las operaciones de acabado deberían llevarse a cabo en máquinas-herramienta específicas. Esto aumentará la exactitud geométrica y la calidad del molde o matriz, consiguiendo tiempos más cortos de pruebas y de montaje.

1.2.4 Tipos de herramientas utilizadas principalmente en este tipo de operaciones.

- **Operaciones de desbaste:** fresas con plaquitas redondas, fresas de punta esférica, fresas de radio grande.

- **Operaciones de semiacabado:** fresas tóricas (fresas con placa redonda en gama de diámetros de 10-25 mm), fresas de punta esférica.

- **Operaciones de acabado:** fresas tóricas, fresas de punta esférica.

- **Operaciones de refresado:** fresas tóricas, fresas de punta esférica, fresas rectas. Es muy importante optimizar el proceso de corte seleccionando las fresas de tamaño adecuado, combinando la calidad y geometría, así como los datos de corte y las estrategias de fresado adecuadas.

1.2.5 Velocidad de corte efectiva (Ve). Siempre es importante basar los cálculos de velocidad de corte efectiva en el diámetro real o efectivo de corte. Si no se hace esto, dará lugar a graves errores de cálculo en el avance, el cual depende de las rpm para una determinada velocidad de corte.
Si cuando se calcula la velocidad de corte se utiliza el valor de diámetro nominal (Dc) de la herramienta, la velocidad de corte efectiva o real será mucho menor si

la profundidad de corte es pequeña. Esto es válido para herramientas como: fresas con placas redondas, fresas de punta esférica. Los niveles de avance serán reducidos y la productividad se verá severamente afectada. Lo más importante es que las condiciones de corte para la herramienta serán inferiores a su capacidad y a la gama recomendada para la aplicación.

Cuando se realiza un mecanizado en 3D el diámetro de corte variará dependiendo de la geometría del molde o matriz. Una solución a este problema es definir secciones/segmentos del molde o matriz con paredes empinadas y aquellas partes que tengan una geometría poco profunda. A partir de ahí, será posible obtener buenos compromisos y resultados, fabricando programas específicos de CAM y datos de corte para cada sección/segmento.

1.2.6 Selección sobre el tipo de fresado. La recomendación principal es: intentar usar el fresado en concordancia lo más posible. Cuando el filo de corte entra a mecanizar en concordancia el grosor de la viruta alcanza su valor máximo. Mientras que con el fresado en contraposición alcanza su valor mínimo. La vida de la herramienta es generalmente más corta en el fresado en contraposición que en el de concordancia, lo cual es debido a que se genera mayor cantidad de calor en contraposición que en concordancia. Cuando el grosor de la viruta en contraposición aumenta de cero hasta el máximo, se genera un exceso de calor puesto que el filo de corte se ve expuesto a una mayor fricción que con en el fresado en concordancia. Las fuerzas radiales son también considerablemente mayores en contraposición, lo cual tiene un efecto perjudicial sobre los rodamientos del husillo.

En el fresado en concordancia el filo de corte está expuesto principalmente a tensiones de compresión, las cuales son más favorables para las propiedades de las herramientas de metal duro o integrales, en comparación con las tensiones que se producen en el fresado en contraposición. Aunque existen algunas consideraciones de excepción. El fresado en contraposición es la primera elección

cuando la operación de fresado se realiza lateralmente con fresas integrales de metal duro, especialmente en materiales templados. Entonces es más fácil obtener una mejor tolerancia en cuanto a plenitud de la pared y también un mejor ángulo de 90 grados en el extremo. En caso de producirse, las marcas de los saltos entre las diferentes pasadas axiales también serán menores. Esto es debido principalmente a la dirección de las fuerzas de corte. Si se está utilizando un filo de corte muy agudo, las fuerzas de corte tenderán a empujar la fresa hacia el material. Un ejemplo en el que se puede aplicar el fresado en contraposición es cuando se dispone de una fresadora manual antigua, con grandes holguras en los husillos, debido a que la presión de corte que ejerce el fresado en contraposición puede usarse para estabilizar la máquina durante el mecanizado.

1.2.7 Posicionamiento de la fresa para obtener el mejor rendimiento. La longitud de corte se ve afectada por la posición de la fresa. A menudo, la vida de la herramienta se ve afectada por la longitud de corte al cual debe someterse el filo de corte. Una fresa que esté posicionada en el centro de la pieza proporcionará una longitud de corte menor, mientras que el arco que describe en el mecanizado será mayor si la fresa se mueve fuera del centro, como se muestra en la figura 6. Teniendo en cuenta cómo se comportan las fuerzas de corte, debe alcanzarse un compromiso en este sentido. En caso de que la fresa sea posicionada en el centro de la pieza, la dirección de las fuerzas radiales de corte variará cuando los filos de las plaquitas salgan y entren del corte. La manipulación del husillo de la máquina también puede aumentar las vibraciones y dirigirlas hacia la plaquita.

Moviendo la fresa fuera del centro se obtendrá una dirección más constante y favorable de las fuerzas de corte. Cuanto mayor sea el voladizo, más difícil resultará contrarrestar todos los posibles riesgos debido a las vibraciones.

1.2.8 Eliminación de vibraciones en un proceso de maquinado. Cuando existe riesgo de vibraciones, la acción básica a realizar es reducir las fuerzas de corte. Esto puede hacerse utilizando las herramientas correctas, métodos y datos de corte.

Se recomienda seguir las propuestas comprobadas y relacionadas seguidamente:[3]

- Elija las fresas de paso grande y diferencial.
- Utilice plaquitas con geometrías positivas y ligeras.
- Utilice la fresa más pequeña posible. Esto es particularmente importante cuando se fresa con adaptadores antivibratorios.
- Utilice plaquitas de corte con pequeño filo de corte redondeado (ER). Pasar de un recubrimiento grueso a uno fino. Si es necesario, utilizar plaquitas sin recubrir. Utilizar calidades tenaces con substrato de grano fino.
- Utilizar un gran avance por diente. Reducir la velocidad de corte y mantener el avance de la mesa (=gran avance/diente). O mantener la velocidad de rotación e incrementar el avance de la mesa (y avance/diente). ¡No reducir el avance por diente!
- Reducir las profundidades de corte radial y axial.
- Seleccionar una sujeción estable de herramienta. Utilizar el adaptador de mayor tamaño posible para obtener la mejor estabilidad. Utilizar extensiones cónicas para una mayor rigidez.

Con voladizos largos, utilizar adaptadores antivibratorios, en combinación con fresas de paso grande y diferencial. Posicionar la fresa lo más próxima posible al adaptador antivibratorio.

Posicionar la fresa fuera del centro de la pieza.

Si se utiliza la fresa con un número de dientes par - se recomienda retirar una de cada dos plaquitas.

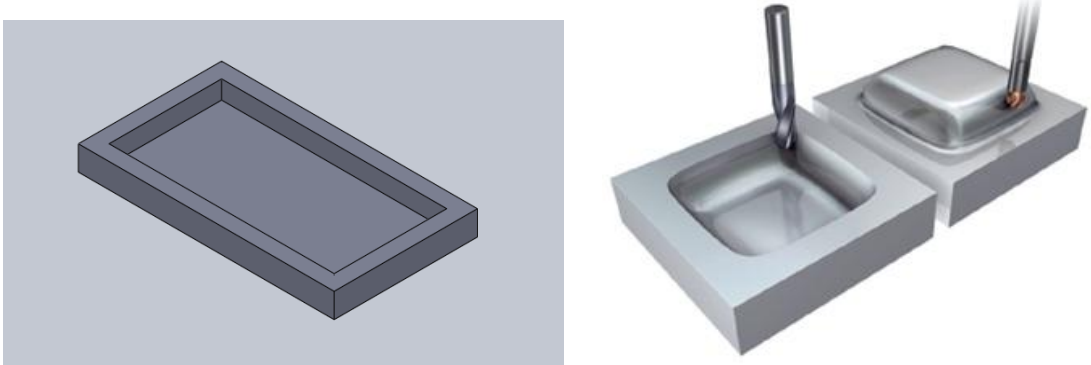
[3] Moldes y matrices [on line]. actualización: 2003-05-07. <www.coromant.sandvik.com>.

1.3 FRESADO DE CAVIDADES

1.3.1 Cavidad. Se llama cavidad o boca al espacio físico en donde el material retirado por la fresa conformará la pieza. Maquinar cavidades es una de las mayores operaciones de eliminación de material en Control Numérico Computarizado (CNC) principalmente con dados y moldes. La forma de la cavidad varía desde una simple geometría a formas complejas, incluyendo islas, que son típicamente maquinadas en tres etapas: desbaste, corte intermedio y acabado. Debido a que la mayoría del volumen es eliminado en el desbaste, su efecto sobre el tiempo de maquinado es significante, y por lo tanto, la incorporación de una eficiente planeación de la trayectoria de la herramienta tiene que ser uno de los puntos cruciales en los sistemas CAD/CAM contemporáneos. En el fresado de cavidades el espesor de la viruta arrancada es constante. En este procedimiento únicamente trabajan las extremidades de los dientes.

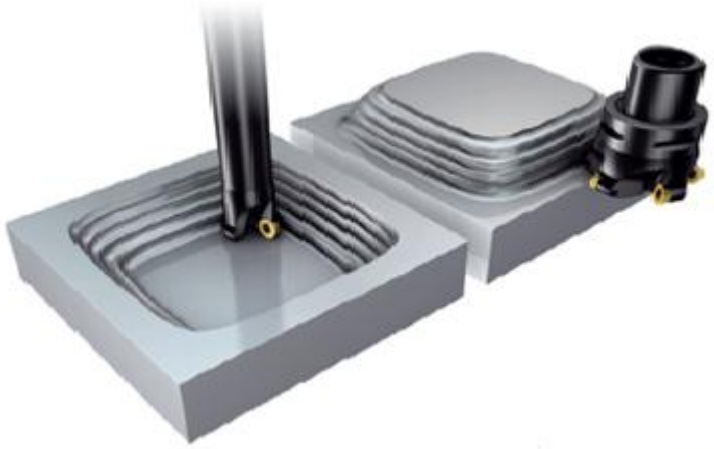
1.3.2 Cavidad simple. Una cavidad simple puede llamarse a aquella cavidad donde no existan islas, o la forma geométrica de ésta sea fácil de representar, y por tanto de mecanizar; por ejemplo un círculo o un cuadrado. Operaciones de fresado como: fresado periférico, planeado, son otros ejemplos de fresado de geometría simple.

Figura 9. Cavidades: a) Cavidad simple b) Proceso de cajeado de una superficie



a) b)

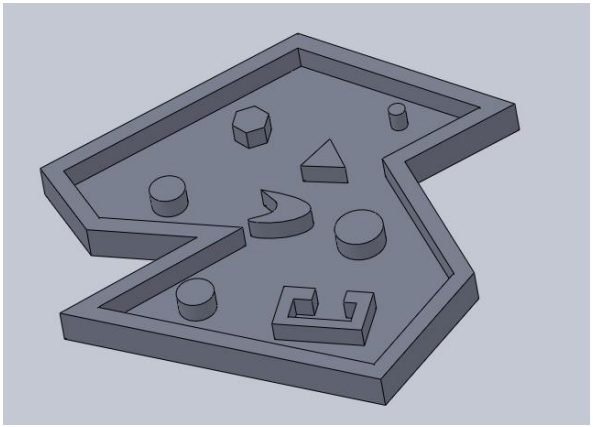
Figura 10. Planeado



Fuente: Catalogo electrónico Sandvik. http://www.sandvik.coromant.com

1.3.3 Cavidad compleja. Cuando la superficie de trabajo tiene una o varias islas y además la forma geométrica de éstas no es fácil de dibujar, se puede decir que es una cavidad compleja.

Figura 11. Cavidad compleja, ocho islas de figuras geométricas y superficie de fresado irregular



Fuente: Autor. Dibujo SolidWorks 2010.

En el fresado de cavidades el espesor de la viruta arrancada es constante. En este procedimiento únicamente trabajan las extremidades de los dientes. La superficie mecanizada en el fresado de cavidades tiene mejor aspecto que en otros procedimientos de fresado, ya que en ella no queda traza alguna de la forma de la fresa, sino únicamente las rayas o surcos dejados por los dientes, que son arcos de cicloides alargados.

2. COMPUTACION NEURONAL

2.1 ANTECEDENTES

El cerebro humano es el dispositivo de cálculo más complejo conocido por el hombre. La capacidad del cerebro para pensar, recordar y solucionar problemas ha inspirado a los diseñadores, ingenieros y programadores de computadores a crear sistemas de computación ``mas inteligentes´´.

A pesar de que muchas tareas resultan especialmente adecuadas para ser resueltas mediante técnicas convencionales hay otras aplicaciones como distinguir entre objetos similares, reconocer imágenes visuales complejas o aprender a partir de la experiencia en lugar de repetir indefinidamente un conjunto explicito de instrucciones, para las cuales no se ha logrado una solución admisible.

En un elevado porcentaje estos problemas no son irresolubles; lo que sucede es que son difíciles de resolver en términos de algoritmos secuenciales. Para resolver este tipo de problemas se han tomado ciertas características de la fisiología del cerebro como base para nuevos modelos de procesamiento. Estas técnicas han recibido el nombre de **sistemas neuronales artificiales,** o simplemente **redes neuronales.**

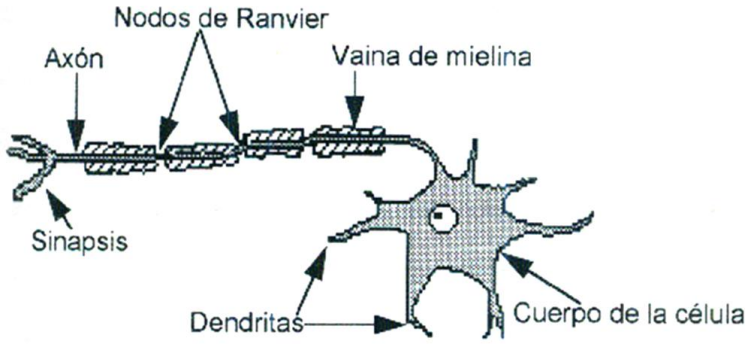
En las siguientes secciones se describen aspectos de los sistemas neuronales biológicos y su funcionamiento. Además se presenta una síntesis de los sistemas neuronales artificiales, describiendo la forma en que procesan información y las principales características que los distinguen de otras ramas de la inteligencia artificial.

2.2 NEUROFISIOLOGIA ELEMENTAL

2.2.1 La neurona individual. La neurona es la unidad celular fundamental del sistema nervioso y, en particular, del cerebro. Cada neurona es una unidad de microproceso la cual recibe y combina señales provenientes de muchas otras

neuronas a través de estructuras de entrada llamadas **dendritas.** Si la señal combinada es suficientemente fuerte la neurona se activa y produce una señal de salida que es transmitida a lo largo de un componente único llamado **axón.** El axón de muchas neuronas está rodeado por una membrana que se denomina **vaina de mielina.** Los **nodos de Ranvier** interrumpen periódicamente la vaina de mielina a lo largo del axón. La figura 12 representa los principales componentes de una célula nerviosa típica perteneciente al sistema nervioso central.

Figura 12. Partes de una neurona



Fuente: MANTILLA CORREDOR, Mario A. y MURILLO FUENTES, William F. Aplicación de la Tecnología de Redes Neuronales a la Fase de Diagnostico del Mantenimiento Predictivo de Equipo Industrial Bucaramanga, 1997, 100P. Trabajo de grado (Ingeniero de Sistemas). Universidad Industrial de Santander. Facultad de Ciencias Físico-Mecánicas. Escuela de Ingeniería de Sistemas e Informática.

La membrana de la neurona separa el plasma intracelular del fluido intersticial que se encuentra fuera de ella. En estos fluidos se encuentran iones positivos de sodio y potasio así como iones negativos de cloro y orgánicos, los cuales producen un equilibrio en el cual hay más iones de sodio y cloro fuera de la célula, y más iones orgánicos y de potasio dentro de ella, produciendo una diferencia de potencial a través de la membrana de la célula de unos 70 a 100 milivoltios, denominado **potencial de reposo** de la célula.

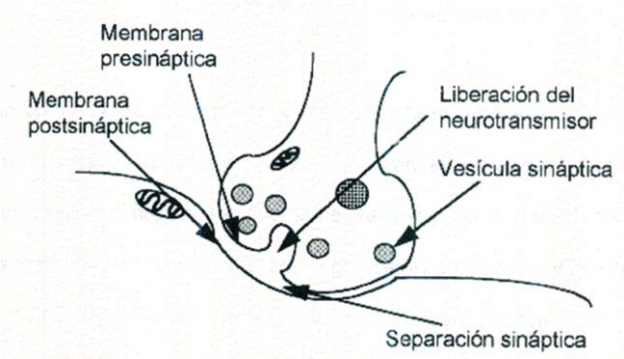
Las entradas excitatorias que llegan a la célula reducen la diferencia de potencial que existe entre los dos lados de la membrana celular, y las inhibitorias lo aumentan. La despolarización resultante en el montículo del axón permite un fuerte flujo entrante de iones sodio positivo, contribuyendo aun más a la despolarización. Este efecto autogenerado da lugar al **potencial de acción.** Este potencial viaja a largo del axón en forma discontinua, de un nodo a otro y va produciendo la despolarización de cada uno de los nodos de ranvier. Una vez que el potencial de acción ha pasado por un cierto punto, ese punto no puede volver a ser excitado durante 1 milisegundo, que es el tiempo que tarda en volver a su potencial de reposo. Este periodo refractario limita la frecuencia de transmisión de los impulsos nerviosos a unos 1000 por segundo.

2.2.2 Unión Sináptica. Las neuronas se interconectan a través de conexiones llamadas uniones sinápticas o simplemente **sinapsis.** La comunicación entre las neuronas es de naturaleza química, pero tiene un efecto eléctrico que puede ser cuantificado. Esta comunicación tiene lugar como resultado de la liberación de unas sustancias llamadas **neurotransmisores** por parte de la célula presináptica y ser absorbida por la célula postsinaptica. Cuando el potencial de acción llega a la membrana presináptica se produce un flujo entrante de iones de calcio. Estos iones dan lugar a que las vesículas que contienen los neurotransmisores se fundan con la membrana sináptica, liberando así sus neurotransmisores en la **separación sináptica.**

Los neurotransmisores se difunden a través de la unión y se unen a la membrana postsinaptica en ciertos lugares llamados receptores, producen así cambios de permeabilidad para esta membrana y permiten la entrada de especies iónicas. Si el flujo entrante de especies iónicas es positivo, se disminuye la diferencia de potencial en este punto de la membrana y por consiguiente el potencial de reposo. Este efecto es inhibitorio. Estos dos efectos son locales, actúan tan solo a lo largo de una pequeña distancia hacia el interior de la célula y se suman en el

montículo del axón. Si la suma es mayor que un cierto valor umbral se genera un potencial de acción. La figura 13 muestra esta actividad.

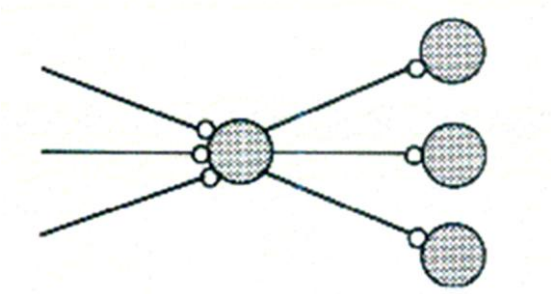
Figura 13. Unión Sináptica



Fuente: Ibíd. Pág. 7.

2.2.3 Circuitos neuronales y computación. Los conceptos de convergencia y divergencia parecen ser el fundamento de toda la actividad del sistema nervioso central, y forman la base de la mayoría de los modelos de redes neuronales artificiales conocidos. La figura 14 muestra estos dos principios básicos: cada neurona envía impulsos a muchas otras neuronas (divergencia) y recibe impulsos procedentes de muchas otras neuronas (convergencia).

Figura 14. Convergencia y divergencia neuronal



Fuente: Ibíd. Pág. 8.

Teniendo una idea de la forma en que operan las neuronas individuales, y del modo en que están conectadas entre sí, surge el interrogante de cómo se

combinan estos conceptos relativamente sencillos para dar al cerebro sus enormes capacidades. El primer intento significativo por explicar esta pregunta se hizo a través del trabajo del neurobiólogo Warren McCulloch y el estadístico Walter Pitts, quienes fueron los primeros de tratar al cerebro como un organismo computacional.

La teoría de McCulloch-Pitts se basa en cinco principios:

1. La actividad de una neurona es un proceso todo-nada, es decir, las neuronas son binarias: o bien están activadas o bien están desactivadas.
2. Es preciso que un número fijo de sinapsis (>1) sean excitadas dentro de un periodo de adición latente para que se excite una neurona.
3. El único retraso significativo dentro del sistema nervioso es el retardo sináptico.
4. La actividad de cualquier sinapsis inhibitoria impide por completo la excitación de la neurona en ese momento.
5. La estructura de la red de interconexiones no cambia con el transcurso del tiempo.
6. La teoría de McCulloch-Pitts ha ayudado a dar forma a los pensamientos de muchas personas que han tenido importancia en el desarrollo de las ciencias de la computación en la actualidad, con su conclusión de que cualquier función lógica puede ser modelada en base a agrupaciones de neuronas interconectadas convenientemente y que se obtiene una gran potencia de cálculo cuando se interconectan adecuadamente y se colocan dentro del sistema nervioso.

2.3 REDES NEURONALES ARTIFICIALES

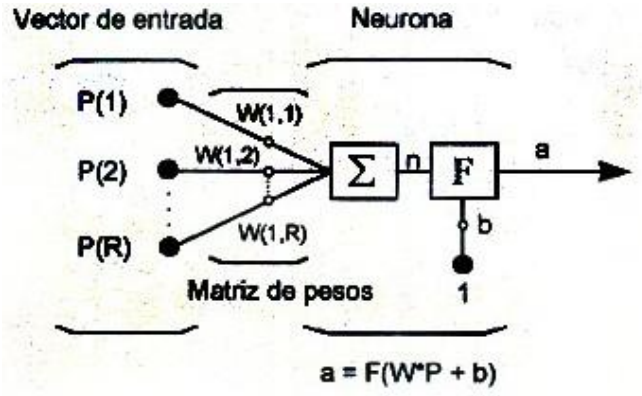
Una red de neuronas artificiales (comúnmente llamada red neuronal) es un sistema de procesamiento de datos formado por una gran cantidad de elementos de procesamiento individuales altamente interconectados en una arquitectura inspirada en la estructura del cerebro.

2.3.1 Elementos de procesamiento (EP). Los elementos individuales de cálculo que forman la mayoría de los modelos de sistemas neuronales artificiales no suelen denominarse neuronas artificiales; lo más frecuente es darles el nombre de **nodos, unidades** o **elementos de procesamiento (EPs).**

Un elemento de procesamiento tiene muchas conexiones de entrada cada una con un **peso** o **intensidad de conexión.** La suma de los productos de cada valor de entrada por su correspondiente peso de conexión se denomina **nivel de actividad.** Este nivel de actividad es transformado por una **función de transferencia**, para producir el valor de salida del elemento de procesamiento.

Algunos elementos de procesamiento tienen un parámetro adicional llamado **bias** o **termino de tendencia,** el cual se comporta como el peso de una conexión que tiene un valor de entrada igual a 1 o -1. Esta estructura se muestra en la figura 15 con la notación matricial correspondiente.

Figura 15. Modelo de un elemento de procesamiento



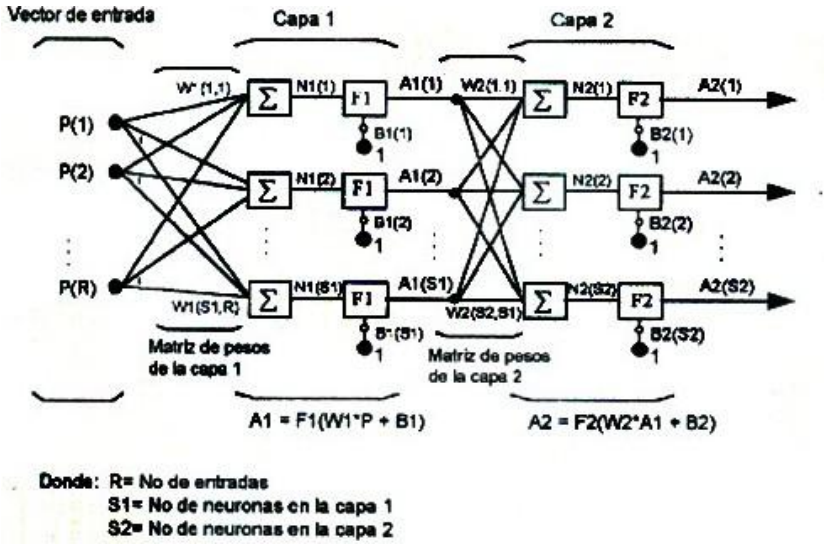
Fuente: Ibíd. Pág. 11.

Por si mismo este modelo de un elemento de procesamiento no es muy interesante; el efecto interesante resulta de la forma en que estos pueden ser interconectados para formar redes neuronales.

2.3.2 Arquitectura de una red neuronal. Una red neuronal es un conjunto de elementos de procesamiento distribuidos en forma de **capas** o **niveles,** las cuales

a su vez, pueden estar conectadas parcial o totalmente. Hay dos capas con conexiones al exterior de la red: una **capa de entrada** donde se le presentan los datos a la red, y una **capa de salida** que contiene la respuesta de la red a una entrada dada. Las capas intermedias se denominan **capas ocultas** y sirven para realizar transformaciones a los datos de entrada buscando obtener la salida deseada. La figura 16 esquematiza en forma básica esta arquitectura con la notación matricial correspondiente.

Figura 16. Arquitectura básica de una red neuronal artificial



Fuente: Ibíd. Pág. 12.

En la gran mayoría de los problemas prácticos no hay razón para usar más de una capa oculta. Aquellos problemas que requieren dos capas ocultas raramente se encuentran en situaciones de la vida real.

2.3.3 Etapas en la operación de las redes neuronales artificiales. Las fases principales en la operación de una red neuronal son: el entrenamiento, la validación y la producción.

- **Entrenamiento**. Es el proceso de adaptación o modificación de los pesos de conexión en respuesta a la información presentada en la capa de entrada y

opcionalmente en la capa de salida. El tipo de entrenamiento más común se denomina **entrenamiento supervisado,** el cual utiliza un conjunto de datos de entrenamiento formado por pares de vectores entrada-salida. Para cada ejemplo se compara la salida obtenida por la red con la salida que se espera obtener. Una vez se ha procesado todo el conjunto de ejemplos de entrenamiento se procede a actualizar los pesos que conectan las capas de neuronas en la red de acuerdo a una **regla de aprendizaje.** La regla de aprendizaje busca reducir la medida de error en los resultados de la red.

Los datos seleccionados para formar parte del conjunto de entrenamiento deben cubrir todo el espacio del fenómeno analizado. Desafortunadamente no hay definiciones sobre la cantidad de ejemplos a utilizar o la forma de seleccionarlos. Al igual que muchos aspectos de las redes neuronales la experiencia suele ser la mejor muestra.

La red neuronal no extrapola bien y si solo se usan datos relativos a una parte del espacio total de entrada-salida durante el entrenamiento, cuando se evalué con datos que no estén dentro de dicho rango las respuestas no serán confiables y no mostraran en realidad el comportamiento del fenómeno.

La fase de entrenamiento termina cuando el nivel de error en los resultados puede considerarse aceptable de acuerdo a la naturaleza del problema.

- **Validación**. En esta etapa se evalúa el desempeño de la red una vez ha terminado el proceso de entrenamiento. El procedimiento usual consiste en separar el conjunto de casos conocidos en dos subconjuntos. Uno es el conjunto de entrenamiento, el cual es usado para entrenar la red. El otro es el conjunto de validación, el cual se usa para probar la red entrenada.

El propósito de la fase de validación, es proporcionar un indicio de los resultados que pueden esperarse de la red cuando se usa en una población general. Si el desempeño de la red durante la validación no es satisfactorio, debe considerarse que hay información importante en el conjunto de validación que la red no ha aprendido, debido a que el conjunto de entrenamiento fue mal diseñado. En este

caso de deben combinar los conjuntos de entrenamiento y de validación en un gran conjunto de entrenamiento, reentrenar la red y recoger un nuevo conjunto de validación para volver a probar la red.
No debe subestimar la importancia de la validación, en muchos aspectos, una apropiada validación es más importante que el mismo entrenamiento.

- **Producción**. Durante esta etapa la red utiliza los valores de pesos de conexión resultantes del entrenamiento para producir una respuesta en función de los valores de entrada presentados.

2.4 CARACTERISTICAS DE LAS REDES NEURONALES ARTIFICIALES.

2.4.1 Aprendizaje a partir de ejemplos. A diferencia de los sistemas expertos tradicionales, donde el conocimiento esta explicito en forma de reglas, las redes neuronales generan su propio conocimiento a partir de los ejemplos que se les presentan. Solo se requiere tener un conjunto de ejemplos representativos de un fenómeno para entrenar la red, tratando que esta se adapte de tal forma que genere unas salidas cuando se le presenten determinadas entradas.

2.4.2 Memoria distribuida. Una característica importante de las redes neuronales es la forma en que almacenan la información. La memoria en las redes neuronales, al igual que en los sistemas biológicos, es distribuida. Los pesos de conexiones son las unidades de memoria de una red neuronal y sus valores representan el estado actual de conocimiento de la red.

2.4.3 Tolerancia al ruido y fallas. La tolerancia a fallas se refiere al hecho de que si algún elemento de procesamiento es destruido, deshabilitado o modificado, o los pesos de conexión son ligeramente alterados, entonces el comportamiento de la red solo se afecta levemente debido a que la información no está ubicada en

un solo lugar sino que se encuentra distribuida a través de toda la red. Una red neuronal también responderá a entradas con ruido. Si algunas entradas han sido dañadas por ruido y sus características originales se han perdido, la red neuronal producirá una salida basándose en aquellas que aun se conserven en buen estado o que posean la menor cantidad de ruido. Esta es una ventaja de la tecnología neuronal con respecto a las soluciones algorítmicas tradicionales.

2.5 REDES NEURONALES CON CONEXIONES LATERALES Y AUTORRECURRENTES.

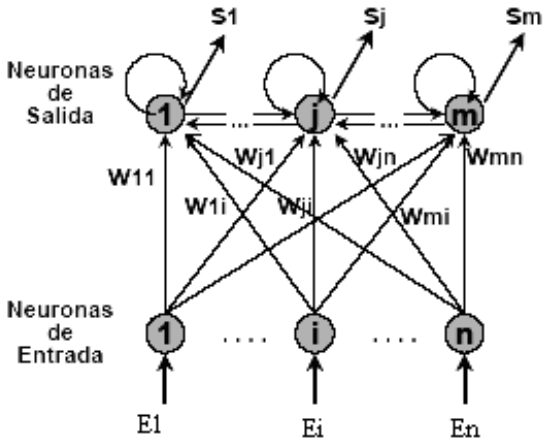
2.5.1 El modelo de kohonen. Existen evidencias que demuestran que en el cerebro hay neuronas que se organizan en muchas zonas, de forma que las informaciones captadas del entorno a través de los órganos sensoriales se representan internamente en forma de *mapas bidimensionales*. Por ejemplo, en el sistema visual se han detectado *mapas* del espacio visual en zonas del córtex (capa externa del cerebro). También en el sistema auditivo se detecta una organización según la frecuencia a la que cada neurona alcanza la mayor respuesta (organización *tonotópica*).

Aunque en gran medida esta organización neuronal esta predeterminada genéticamente, es probable que parte de ella se origine mediante el aprendizaje. Esto sugiere, por tanto, que el cerebro podría poseer la capacidad inherente de formar *mapas topológicos* de las informaciones recibidas del exterior. De hecho, esta teoría podría explicar su poder de operar con elementos semánticos: algunas áreas del cerebro simplemente podrían crear y ordenar neuronas especializadas o grupos con características de alto nivel y sus combinaciones. Se trataría, en definitiva, de construir mapas espaciales para atributos y características.

A partir de estas ideas, T. KOHONEN presentó un sistema con un comportamiento semejante. Se trataba de un modelo de red neuronal con capacidad para formar

mapas de características de manera similar a como ocurre en el cerebro. El objetivo de Kohonen era *demostrar que un estimulo externo* (información de entrada) *por sí solo, suponiendo una estructura propia y una descripción funcional del comportamiento de la red, era suficiente para forzar la formación de los mapas.* Este modelo tiene dos variantes, denominadas LVQ (*learning vector quantization)* y TPM (*topology-preserving map)* o SOM (*self-organizing map)*. Ambas se basan en el principio de formación de mapas topológicos para establecer características comunes entre la información (vectores) de entrada a la red, aunque difieren en las dimensiones de éstos, siendo de una sola dimensión en el caso de LVQ, y bidimensional, e incluso tridimensional, en la red TPM.

Figura 17. Arquitectura de la red LVQ (learning vector quantization)



HILERA, José R. y MARTINEZ Víctor J. Redes Neuronales Artificiales. Fundamentos, Modelos y Aplicaciones. ALFAOMEGA. 2000. Pág. 253-276.

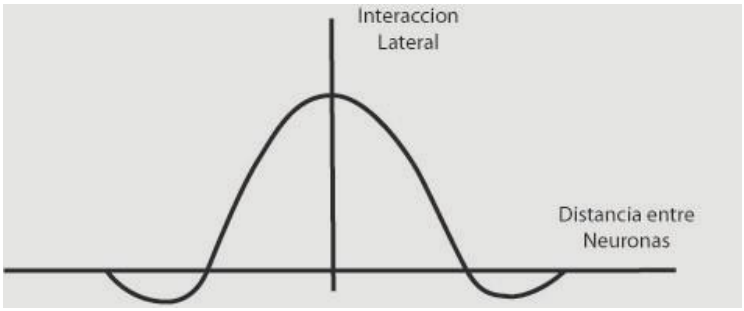
2.5.2 Arquitectura de una red de kohonen. La arquitectura de la versión original (LVQ) del modelo de Kohonen (fig. 17) se trata de una red de dos capas con N neuronas de entrada y M de salidas. Cada una de las N neuronas de entrada se conecta a las M de salida a través de conexiones hacia delante (*feedforward*).

Entre las neuronas de la capa de salida, puede decirse que existen conexiones laterales de inhibición (peso negativo) implícitas, pues aunque no estén

conectadas, cada una de estas neuronas va a tener cierta influencia sobre sus vecinas. El valor que se asigne a los pesos de las conexiones *feedforward* entre la capa de entrada y de salida (w_{ji}) durante el proceso de aprendizaje de la red va a depender precisamente de esta interacción lateral.

La influencia que una neurona ejerce sobre las demás es función de la distancia entre ellas, siendo muy pequeña cuando están muy alejadas. Es frecuente que dicha influencia tenga la forma de un *sombrero mejicano*, como se muestra en la figura 20. Esta afirmación tiene una base biológica, ya que existen evidencias fisiológicas de interconexiones laterales de este tipo entre las neuronas del sistema nervioso central de los animales. Así, se ha podido comprobar que en determinados primates se producen interacciones laterales de tipo excitatorio entre neuronas próximas en un radio de 50 a 100 micras, de tipo inhibitorio en una corona circular de 150 a 400 micras de anchura alrededor del circulo anterior, y de tipo excitatorio muy débil, prácticamente nulo, desde ese punto hasta una distancia de varios centímetros.

Figura 18. Interacción entre neuronas de la capa de salida

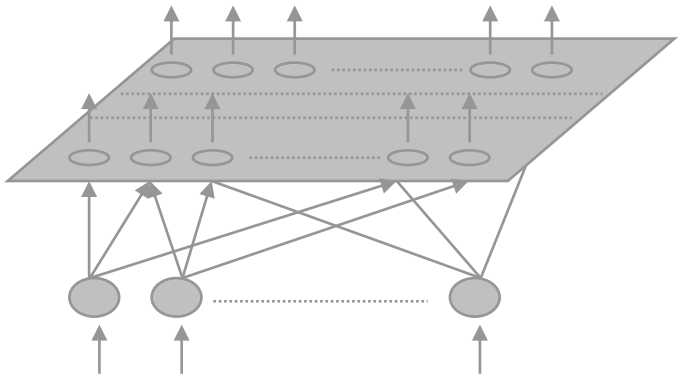


Fuente: Ibíd. Pág. 255.

Por otra parte, la versión del modelo denominado TPM (*topology preserving map)* trata de establecer una correspondencia entre los datos de entrada y un espacio bidimensional de salida, creando *mapas topológicos* de dos dimensiones, de tal forma que ante datos de entrada con características comunes se deben activar neuronas situadas en zonas próximas de la capa de salida.

Por esta razón, la representación habitual de esta red suele ser la mostrada en la figura 19, donde las M neuronas de salida se disponen de forma bidimensional para representar precisamente los mapas de características.

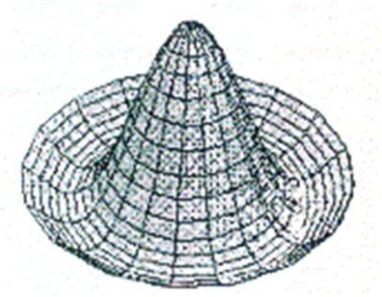
Figura 19. Arquitectura de la red TPM (*topology preserving map)* de Kohonen



Fuente: Ibíd. Pág. 256.

La interacción lateral entre neuronas de la capa de salida sigue existiendo, aunque ahora hay que entender la distancia como una zona bidimensional que existe alrededor de cada neurona. Esta zona puede ser circular (fig. 20), cuadrada, hexagonal o cualquier otro polígono regular centrado en dicha neurona.

Figura 20. Interacción circular entre neuronas de la capa de salida



Fuente: Ibíd. Pág. 256.

2.5.3 Funcionamiento de una red de kohonen. El funcionamiento de esta red es relativamente simple. Cuando se presenta a la entrada una información $\mathbf{E_K}$ **=**

($e_1^{(k)}$,….., $e_N^{(k)}$), cada una de las M neuronas de la capa de salida la recibe a través de las conexiones *feedforward* con pesos ***w_{ji}***.

También estas neuronas reciben las correspondientes entradas debidas a las conexiones laterales con el resto de las neuronas de salida y cuya influencia dependerá de la distancia a la que se encuentren.

Así, la salida generada por una neurona de salida *j* ante un vector de entrada E_K sería:

$$Sj\,(t+1) = f\left(\sum_{i=1}^{N} w_{ji}\, e_i^{(k)} + \sum_{i=1}^{N} Int_{pj}\, s_p(t)\right) \qquad (9)$$

Donde Int_{pj} es una función del tipo sombrero mejicano (fig. 16 y 18) que representa la influencia lateral de la neurona *p* sobre la neurona *j.* La función de activación de las neuronas de salida (f) será del tipo continuo, lineal o sigmoidal, ya que esta red trabaja con valores reales.

Es evidente que se trata de una red de tipo competitivo, ya que al presentar una entrada E_K la red evoluciona hasta una situación estable en la que se activa una neurona de salida, la vencedora. Por ello, la formulación matemática de su funcionamiento puede simplificarse mediante la siguiente expresión, que representa cual de las M neuronas se activaría al introducir dicha información E_K:

$$S_j = \begin{cases} 1 \;\; MIN\,|E_K - W_J| = MIN\left(\sqrt{\left(\sum (e_i^{(k)} - w_{ij})^2\right)}\right) \\ 0 \;\; resto \end{cases} \qquad (10)$$

Donde $|E_K-W_J|$ es una medida (por ejemplo, distancia euclídea) de la diferencia entre el vector de entrada $E_K = (e_1^{(k)},\ldots, e_N^{(k)})$ y el vector de los pesos $W_J = (w_{j1},\ldots, w_{JN})$ de las conexiones entre cada una de las neuronas de entrada y la neurona de salida j. En estos pesos se registran los datos almacenados en la red durante el proceso de aprendizaje. En la fase de funcionamiento, lo que se pretende es

encontrar el dato *aprendido* más parecido al de entrada para, en consecuencia, averiguar que neurona se activará y, sobre todo, en que zona del espacio bidimensional de salida se encuentra.

Lo que hace la red de Kohonen, en definitiva, es realizar una tarea de clasificación, ya que la neurona de salida activada ante una entrada representa la clase a la que pertenece dicha información de entrada. Además, como ante otra entrada parecida se activa la misma neurona de salida, u otra cercana a la anterior, debido a la semejanza entre las clases, se garantiza que las *neuronas topológicamente próximas sean sensibles a entradas físicamente similares.* Por esta causa, la red es especialmente útil para establecer relaciones, desconocidas previamente, entre conjuntos de datos.

2.5.4 Aprendizaje. El aprendizaje en el modelo de Kohonen es de tipo OFF LINE, por lo que se distingue una etapa de aprendizaje y otra de funcionamiento. En la etapa de aprendizaje se fijan los valores de los pesos da las conexiones (*feedforward)* entre la capa de entrada y la de salida.

Esta red utiliza un *aprendizaje no supervisado de tipo competitivo.* Las neuronas de la capa de salida compiten por activarse y solo una de ellas permanece activa ante una determinada información de entrada a la red. Los pesos de las conexiones se ajustan en función de la neurona que haya resultado vencedora.

Durante la etapa de entrenamiento, se presenta a la red un conjunto de informaciones de entrada (*vectores de entrenamiento)* para que esta establezca, en función de la semejanza entre los datos, las diferentes categorías (una por neurona de salida) que servirán durante la fase de funcionamiento para realizar clasificaciones de nuevos datos que se presenten a la red. Los valores finales de los pesos de las conexiones entre cada neurona de la capa de salida con las de entrada se corresponderán con los valores de los componentes del vector de aprendizaje que consigue activar la neurona correspondiente. En el caso de existir más patrones de entrenamiento que neuronas de salida, más de uno deberá

asociarse con la misma neurona; es decir, pertenecerá a la misma clase. En tal caso, los pesos se obtienen como un promedio de dichos patrones.

En este modelo, el aprendizaje no concluye después de presentarle una vez los patrones de entrada, sino que habrá que repetir el proceso varias veces para refinar el mapa topológico de salida, de tal forma que cuantas más veces se presenten los datos, tanto más se reducirán las zonas de neuronas que se deben activar ante entradas parecidas, consiguiendo que la red pueda realizar una clasificación mas selectiva.

El algoritmo de aprendizaje utilizado para establecer los valores de los pesos de las conexiones entre las N neuronas de entrada y las M de salida es el siguiente:

En primer lugar, se inicializan los pesos (w_{ji}) con valores aleatorios pequeños y se fija la zona inicial de vecindad entre las neuronas de salida.

A continuación se presenta a la red una información de entrada (la que debe *aprender)* en forma de vector $E_K = (e_1^{(k)},\ldots, e_N^{(k)})$, cuyas componentes $e_i^{(k)}$ serán valores continuos.

Puesto que se trata de un aprendizaje competitivo, se determina la neurona vencedora de la capa de salida. Esta será aquella j cuyo vector de pesos W_J (vector cuyas componentes son los valores de los pesos de las conexiones entre esa neurona y cada una de las neuronas de la capa de entrada) sea el mas parecido a la información de entrada E_K (patrón o vector de entrada).

Para ello, se calculan las distancias o diferencias entre ambos vectores, considerando una por una todas las neuronas de salida. Suele utilizarse la distancia euclídea o la siguiente expresión, que es similar a aquella, pero eliminando la raíz cuadrada:

$$d_j = \sum (e_i^{(k)} - w_{ij})^2 \qquad 1 \leq j \leq M \qquad (11)$$

Siendo:

$e_i^{(k)}$: Componente i-ésimo del vector k-ésimo de entrada.

w_{ji}: Peso de la conexión entre la neurona i de la capa de entrada y la neurona j de la capa de salida.

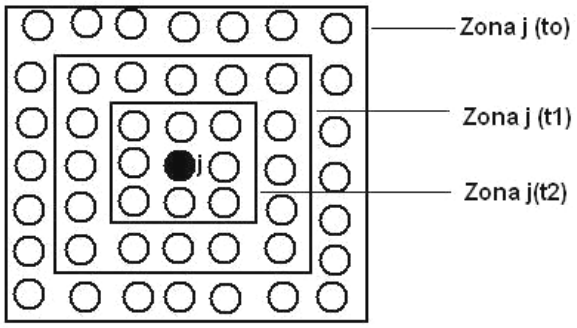
Una vez localizada la neurona vencedora (j^*), se actualizan los pesos de las conexiones entre las neuronas de entrada y dicha neurona, así como los de las conexiones entre las de entrada y las neuronas vecinas de la vencedora. En realidad, lo que se consigue con esto es asociar la información de entrada con una cierta zona de la capa de salida.

$$Wji\ (t+1) = w_{ji}\ (t) + \alpha\ (t)\ [\ e_i^{(k)} - w^*_{j\,i}\ (t)\ ]\ \text{ para } j \in \text{Zona}^*_j(t) \qquad (12)$$

Zona $^*_j$(t) es la zona de vecindad alrededor de la neurona vencedora j^* en la que se encuentran las neuronas cuyos pesos son actualizados. El tamaño de esta zona se puede reducir en cada iteración del proceso de ajuste de los pesos, con lo que el conjunto de neuronas que pueden considerarse vecinas cada vez es menor (fig. 19). Sin embargo, en la práctica es habitual considerar una zona fija en todo el proceso de entrenamiento de la red.

El término α (t) es un parámetro de ganancia o coeficiente de aprendizaje, con un valor entre 0 y 1, decrece con el numero de iteraciones (t) del proceso de entrenamiento. De tal forma que cuando se ha presentado un gran número de veces todo el juego de patrones de aprendizaje ($500 \leq t \leq 10000$) su valor es prácticamente nulo, con lo que la modificación de los pesos es insignificante.

Figura 21. Posible evolución de la zona de vecindad



Fuente: Ibíd. Pág. 261.

Suele utilizarse alguna de las siguientes expresiones:

$$\alpha(t) = 1/t \qquad \alpha(t) = \alpha_1 (1 - (t\,\alpha_2)) \qquad (13)$$

Siendo α_1 un valor de 0.1 ó 0.2 y α_2 un valor próximo al número total de iteraciones del aprendizaje. Suele tomarse un valor $\alpha_2 = 10000$.

El proceso se debe repetir, volviendo a presentar solo el juego de patrones de aprendizaje E_1, E_2,…, un mínimo de 500 veces ($t \geq 500$).

2.5.5 aplicaciones de la red de Kohonen. El modelo de Kohonen es uno de los más útiles en computación neuronal, a pesar de sus limitaciones en cuanto a la duración del proceso de aprendizaje y a la imposibilidad de aprender nuevos datos sin tener que volver a repetir completamente el proceso de aprendizaje con todos los patrones. Esta utilidad se debe a su capacidad para establecer clases o categorías de datos sin supervisión.

Como aplicaciones, destacan las relacionadas con el reconocimiento de patrones (voz, texto, imágenes, señales, etc.), codificación de datos, comprensión de imágenes y resolución de problemas de optimización, como el del *viajante.*

También se ha utilizado este modelo en robótica, comprobándose su utilidad en el diseño de sistemas para controlar el movimiento de un brazo mecánico en un espacio tridimensional. En dichos sistemas, se utiliza una red de Kohonen para aprender las magnitudes tensoriales necesarias para moverse en un entorno real, considerando los efectos del desgaste que pueden alterar la dinámica del brazo con el transcurso del tiempo.

A continuación se mencionan cuatro campos de aplicación práctica para este tipo de redes.

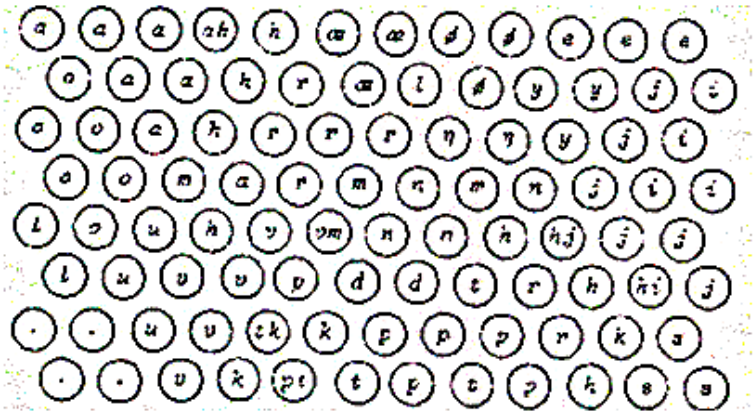
- **Reconocimiento de voz**. Una de este modelo es el conocido *mecanógrafo fonético (phonetic typewriter).* Se trata por un prototipo ideado por el propio Kohonen para convertir, en tiempo real, el habla en texto escrito a partir de un

vocabulario limitado con una precisión del 92% y de forma independiente de la persona que hable y del ruido ambiente, con el inconveniente de requerir un proceso de aprendizaje relativamente largo.

La red utilizada tiene 15 neuronas de entrada, a través de las cuales recibe las componentes en frecuencia que caracterizan a los diferentes fonemas que componen las palabras, obtenidos mediante un micrófono y un preprocesador analógico y digital. Se entrena la red para que ante fonemas parecidos active neuronas de salida próximas, creando finalmente lo que el autor denomina un *mapa fonotópico.* En la figura 22 se muestra un ejemplo de mapa para el idioma finés (el de Kohonen, su autor), donde se indican los fonemas ante los cuales han aprendido a responder cada una de las neuronas de salida.

Después de aprendizaje, durante la fase de funcionamiento, cuando se pronuncia una palabra, esta se divide en fonemas que se presentan secuencialmente a la red, la cual responde activando en orden las adecuadas neuronas en la capa de salida, recorriendo un camino en el mapa que se corresponde con la secuencia de entrada, dando lugar a una trascripción fonética de la palabra.

Figura 22. Ejemplo de mapa fonotópico



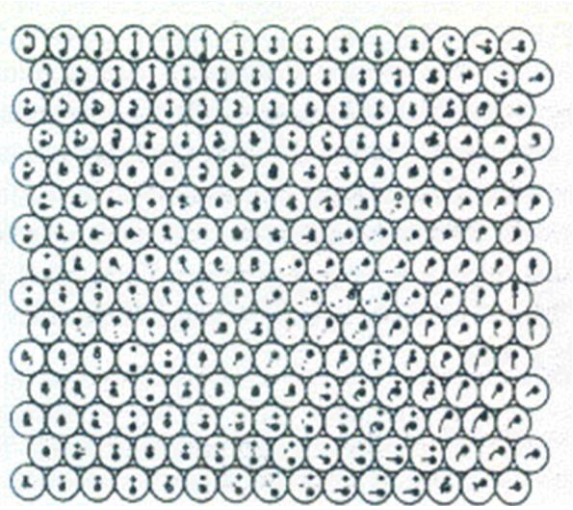
Fuente: Ibíd. Pág. 268.

- **Reconocimiento de texto manuscrito**. Esta aplicación es semejante a la anterior pero considerando *mapas grafotópicos*, en lugar de *fonotópicos*, para

convertir un texto manuscrito en texto estándar con caracteres codificados en ASCII. En este caso, la red utilizada recibe como entrada la representación paramétrica de un *stroke* o porción de trazo limitado entre dos puntos sucesivos de escritura manuscrita, algo semejante a los fonemas en el habla. Esta representación puede consistir en las coordenadas cartesianas de puntos equidistantes a lo largo del trazo de una palabra.

La red de Kohonen se entrena con diferentes *strokes*, generando un mapa grafotópico como el indicado en la figura 23, correspondiente al entrenamiento de una red con 15x15 = 225 neuronas de salida y 10 de entrada (10 valores por *stroke)*, utilizando un conjunto de patrones de aprendizaje de 200 palabras. En la figura se muestran los diferentes *strokes* asociados a las 225 neuronas de salida. El círculo pequeño en el interior de una neurona indica el punto inicial del *strokes*, y la cruz identifica los ejes cartesianos.

Figura 23. Ejemplo de mapa grafotópico



Fuente: Ibíd. Pág. 269.

Después del aprendizaje, durante la fase de funcionamiento, cuando se analiza una palabra manuscrita, ésta se divide previamente en *strokes* que se presentan secuencialmente a la red, la cual responde activando en orden las adecuadas

neuronas de la capa de salida, recorriendo un camino en el mapa que se corresponde con la secuencia de entrada, dando lugar a una trascripción grafológica de la palabra.

También existe la posibilidad de adaptar el sistema al idioma concreto con que se trabaje. Para ello, se puede añadir una segunda red a la salida de la anterior que detecte largas secuencias de *strokes* de una longitud dada, de esta forma se pueden detectar caracteres completos e incluso palabras frecuentes del lenguaje, como artículos o terminaciones de palabras (…mente, …ísimo, etc.).

- **Codificación de datos**. El modelo de Kohonen se ha utilizado en aplicaciones relacionadas con la codificación de la información como mecanismos para diseñar códigos que compriman o reduzcan el tamaño da los datos (*data quantization)* a almacenar o a transmitir por un sistema de comunicación.

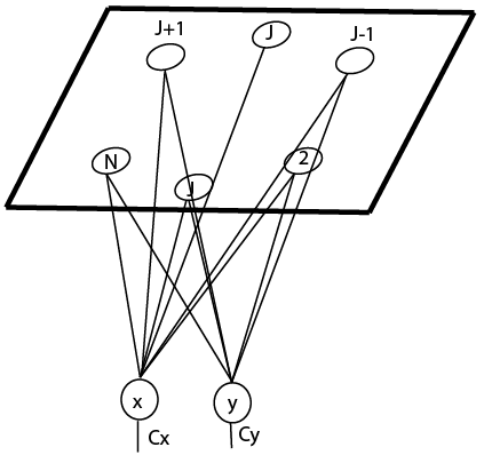
En este sentido, cabe destacar su utilidad en la comprensión de imágenes en color. Esta red se utiliza en la conversión de imágenes originales con píxeles de 2^{24} = 16.777.216 posibles colores diferentes (el color de un píxel se codifica con 24 bits) a imágenes para monitores VGA con píxeles de 2^8 = 256 colores posibles (8 bits por píxel). En esta aplicación se utiliza la red de Kohonen para establecer los 256 colores más frecuentes presentes en las imágenes originales. Para ello, previamente se divide cada imagen en otras tres de tipo monocolor que representen las versiones en rojo, verde y azul, que combinadas dan lugar a la imagen original. Cada píxel de estas versiones se codifica con 24/3 = 8 bits, por lo que existen 256 posibles grados de intensidad de rojo, verde y azul.

Se utiliza una red con tres neuronas de entrada que reciben los valores correspondientes a cada color básico (rojo, verde y azul) de cada una de los píxeles de la imagen. En la capa de salida se disponen 256 neuronas en representación de los 256 colores más frecuentes que debe averiguar la red. Después del entrenamiento, en los pesos de las tres conexiones de cada neurona de salida con las tres de entrada se ha obtenido la comunicación de rojo, verde y

azul para cada uno de los 256 colores más frecuentes. En realidad, puede decirse que se ha obtenido un mapa *cromato-tópico* de la imagen *aprendida* por la red.
En definitiva, lo que se hace es utilizar una red de Kohonen para averiguar, al reducir una imagen a 256 colores, que píxeles de la imagen original deberían representarse con el mismo color, lo cual no es sino una nueva forma de codificar la información utilizando un menor número de bits por píxel (8 en lugar de 24 en este ejemplo), y para conocer la combinación de rojo, verde y azul para ese color.

- **Resolución de problemas de optimización**. El modelo de Kohonen se ha utilizado también para resolver el problema de optimización conocido como *problema del viajante* o del *vendedor viajero,* estableciendo el camino mas corto entre una serie de ciudades sin pasar dos veces por la misma.

Figura 24. Red de Kohonen para resolver el problema del viajante



Fuente: Ibíd. Pág. 272.

La red que puede solucionar este problema tiene tantas neuronas de salida como ciudades del recorrido, y dos neuronas de entrada para las coordenadas en el plano (x, y) de cada ciudad (figura. 24). La distribución de las neuronas de salida es circular, estableciendo para cada neurona (j) una zona de vecindad que incluye las neuronas anterior (j-1) y posterior (j+1).

El proceso a seguir para resolver el problema, consiste en entrenar a la red para que *aprenda* la situación geográfica de las N ciudades del recorrido, adaptando los pesos mediante el algoritmo de Kohonen. Cuando el entrenamiento ha concluido, al existir tantas neuronas de salida como ciudades, deberán coincidir los vectores de pesos de la red $\mathbf{W_j = [\ w_{jx}, w_{jy}\ ]}$ con las coordenadas de las N ciudades ($C_1 = [\ c_x^{(1)}, c_y^{(1)}\ ]$,…., $C_N = [\ c_x^{(N)}, c_y^{(N)}\ ]$). Entonces el problema ha sido resuelto, siendo la solución el recorrido resultante de unir los puntos (ciudades) correspondientes a los pesos de las neuronas adyacentes de la capa de salida.

Este mecanismo puede entenderse mejor a través de un ejemplo, con cinco ciudades distribuidas como ilustra la figura 25. En este caso, en lugar de distancias, se trabajara con las coordenadas de las ciudades en el plano X-Y, cuyos valores son:

$$C_1 = [1,9] \quad C_2 = [6,5] \quad C_3 = [9,7] \quad C_4 = [2,1] \quad C_5 = [7,2]$$

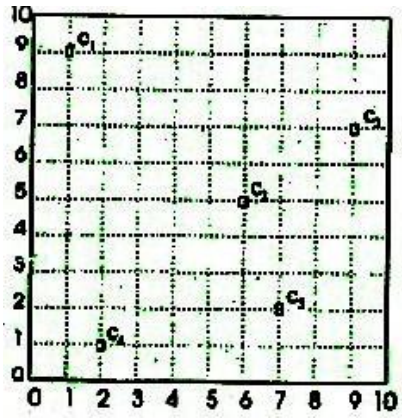
A partir de las coordenadas, es directo el cálculo de las distancias entre las ciudades. Suponiendo que se está trabajando con valores expresados en kilómetros, las distancias serian las indicadas en la tabla 1.

Tabla 1. Distancia (en Km.) entre las cinco ciudades del ejemplo

	C_1	C_2	C_3	C_4	C_5
C_1	0.00	6.40	8.25	8.06	9.22
C_2	6.40	0.00	3.60	5.66	3.16
C_3	8.25	3.60	0.00	9.22	5.38
C_4	8.06	5.66	9.22	0.00	5.10
C_5	9.22	3.16	5.38	5.10	0.00

Fuente: Ibíd. Pág. 273.

Figura 25. Situación de las 5 ciudades del ejemplo



Fuente: Ibíd. Pág. 274.

Puede comprobarse que el recorrido más corto, corresponde con la secuencia C_1-C_3-C_2-C_5-C_4-C_1, con una distancia de valor:

Distancia mínima = $d_{13}+d_{32}+d_{25}+d_{54}+d_{41}$ = 8.25+3.60+3.16+5.10+8.06

= 28.17 Km.

Se puede utilizar una red de Kohonen para obtener una solución del problema. En este caso, se trataría de una red con dos neuronas de entrada para las coordenadas X e Y de cada ciudad, y 5 neuronas de salida para disponer de 5 vectores (W_1,…., W_5) de pesos W_J = [w_{jx} , w_{jy}], uno por ciudad, ya que cuando la red *aprenda* las posiciones de las ciudades, el valor de los pesos coincidirá con dichas posiciones, siendo el orden del recorrido el de los pesos: primero se pasara por la ciudad almacenada en W_1, después por la que indique W_2 y finalmente por la de W_5. Por tanto, la solución del problema consiste en averiguar el reparto de ciudades por pesos para conocer la secuencia del recorrido.

3. DESARROLLO DE LA APLICACIÓN AL FRESADO DE CAVIDADES

3.1 EL PROBLEMA DEL SECUENCIAMIENTO EN OPERACIONES DE MECANIZADO

Programar la secuencia de operaciones de mecanizado que se deben realizar para fabricar una determinada pieza es un problema sumamente complejo, pues si tenemos en cuenta que para fabricar una pieza se deben realizar ``n´´ operaciones de mecanizado, entonces se sabe que se puede elaborar esta pieza de n! formas diferentes. Dentro de este conjunto de posibilidades, se encuentra que los tiempos envueltos en la ejecución del proceso pueden ser muy diferentes, dependiendo de la posibilidad escogida.[4]

Los altos tiempos de mecanizado generan costos más elevados, lo que se traduce en pérdida de competitividad en el mercado, lo cual puede llevar a la compañía a la quiebra, problema que se refleja en el resto de la sociedad, pues quienes dependen directa o indirectamente de estas compañías ingresaran a formar parte del grupo de los desempleados.

Para comprender las posibles causas de la gran diferencia que se presenta en los tiempos de mecanizado, dependiendo de la secuencia de operaciones seleccionada, se deben tener claros los diferentes tiempos que se generan en el proceso productivo. Son estos:

- **Tprb = Tiempo de preparación básico de la tarea:** es el tiempo necesario para la ejecución de todos los trabajos indispensables para el inicio de la tarea,

[4] ANAYA BARRERA, Renan y GALVIZ MUÑOZ, José David. Optimización del Secuenciamiento del Mecanizado de Piezas en una Máquina Herramienta. Bucaramanga, 2001, 192P. Trabajo de grado (Ingeniero Mecánico e Ingeniero de Sistemas).Universidad Industrial de Santander. Facultad de Ciencias Físico-Mecánicas. Escuela de Ingeniería Mecánica. Escuela de Ingeniería de Sistemas e Informática.

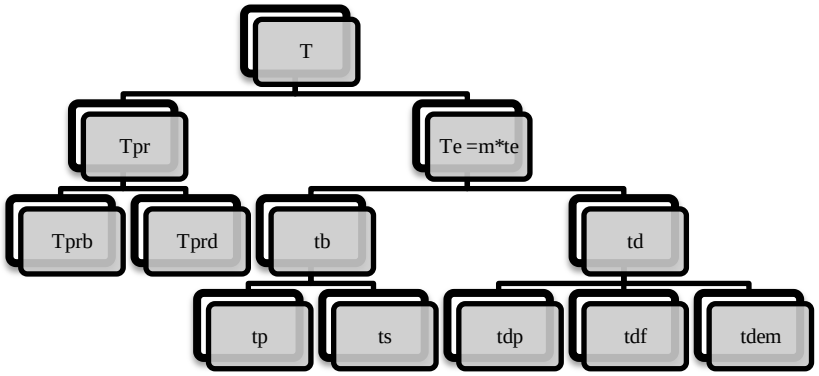
así como, para la recolocación de la maquina y del local de trabajo, después de concluida la tarea, en las condiciones iniciales. Incluye, entre otros, los siguientes trabajos: obtención del material, herramientas, dispositivos, accesorios, plantillas, diseños y especificaciones; montajes de las herramientas, accesorios y dispositivos, ajuste de las velocidades de corte y del avance, ejecución de piezas de prueba, llenado de formularios y fichas de producción; desmontaje de herramienta, accesorios y dispositivos, limpieza de la maquina.

- **Tprd = Tiempo de preparación distribuido** de la tarea: es el tiempo de preparación requerido en razón de factores ocasionales tales como: esclarecimiento de dudas, cambio de material defectuoso, remoción de fallas en la maquina, en los dispositivos y accesorios, ajuste de la calidad del filo o de los ángulos de la herramienta.
- **tp = Tiempo principal:** Es el tiempo en que ocurre efectiva remoción de material, haciendo que una pieza se aproxime a la forma final deseada.
- **ts = Tiempo secundario de ejecución:** Es el tiempo para la realización de todos los trabajos que se repiten regularmente para cada pieza, tales como: coger una pieza y llevarla hasta la maquina, sujetar la pieza, alistar la maquina, aproximar la herramienta, poner el avance, cambiar el sentido de rotación, parar la maquina, inspeccionar, medir y retirar la pieza da la maquina, etc.
- **td = Tiempo distribuido de ejecución de la pieza = tdp + tdf + tdem**, es una medida, por pieza, de los tiempos accidentales que ocurre en una fase de ejecución del lote solicitado.
- **tdp = Tiempo distribuido debido al personal,** tales como, recepción de salarios, necesidades fisiológicas, accidentes, atrasos, tratamiento médico, fatiga mental o física, etc.
- **tdf = Tiempo distribuido debido a la herramienta:** Remoción, reafilación y remontaje de la herramienta.

- **tdem = Tiempo distribuido debido al equipo y al material,** tales como, remecanizado de piezas fuera de medida, rearrume de las piezas, eliminación de fallas y defectos en el equipamiento y en los accesorios, remoción de virutas, etc.

Estos tiempos y su relación se observaran en la figura 26.

Figura 26. Componentes del tiempo global



Fuente: STEMMER, Caspar E. Herramientas de corte I. 3ª Edición. Florianópolis: Editora DAUFSC, 1993.

Para tratar de dar solución al problema del secuenciamiento, sin la utilización de un computador, se tendría que realizar una serie de cálculos complejos y tediosos con cada una de las posibilidades. Para facilitar esta labor, se hace sumamente necesario la utilización de un software que permitan procesar grandes volúmenes de información, de manera confiable y precisa, para seleccionar de forma rápida el secuenciamiento optimo para la fabricación de la pieza o para la reducción de los tiempos anteriormente mencionados individualmente y por lo tanto del tiempo global, para conseguir finalmente llegar a volúmenes de fabricación más altos y a reducir los tiempos muertos en cada una de las maquinas.

En el sector metalmecánico, la disminución de los tiempos está relacionado directamente con la disminución de los costos de producción. Realizar la producción utilizando un secuenciamiento diferente al optimo aumentaría los costos de fabricación y por ende coloca en una posición desfavorable a la

compañía, frente a su competencia en el mercado, por tanto, se hace necesaria la implementación de un software que de una manera sencilla y confiable, proporcione el secuenciamiento optimo o la optimización de los tiempos involucrados en la fabricación de una pieza determinada, sujeto a las maquinas herramientas existentes en la empresa y que sea capaz de dar respuesta en tiempo real a una perturbación (falla o perdida) causada por la herramienta en la planificación original, cuando se mecaniza una pieza en la maquina.

En los procesos de fabricación en los cuales se tenga que hacer una operación de fresado de cavidades, la dificultad más importante de resolver es la determinación de una trayectoria optima para llevar a cabo la remoción de material. Por esta razón, a continuación se describe un método para la solución de este problema teniendo en cuenta la aplicación de redes neuronales.

3.2 PLANEACION DE LA TRAYECTORIA DE LA HERRAMIENTA EN EL FRESADO DE CAVIDADES

Este proyecto de grado muestra un nuevo enfoque para la planeación de la trayectoria de la herramienta en operaciones de fresado de cavidades. La idea clave es formular el problema de la trayectoria dentro de un TSP (traveling salesman problem) así este poderoso método puede ser efectivamente aplicado. Específicamente, el método está compuesto de: (1) digitalización del área de la cavidad en un número finito de puntos, (2) un método de red neuronal (llamado SOM-mapas auto-organizados) para encontrar la trayectoria.

Por el procedimiento de red neuronal, una eficiente trayectoria de la herramienta (en cuanto a longitud de la ruta y retracciones de la herramienta) podrá ser obtenida para cualquier cavidad formada arbitrariamente con algunas islas.

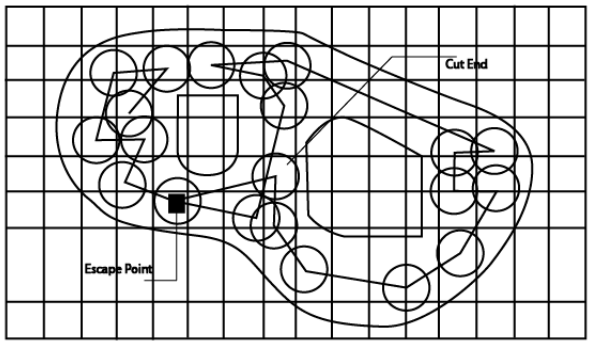
La validez y el poder del algoritmo está demostrado a través de simulación en computador.

3.2.1 Modelando TSP y solución aproximada. Se considera inicialmente la geometría de la cavidad que incluye dos islas, mostrada en la figura 27. El plano

es dividido por una rejilla con un número finito de celdas. Sobre la rejilla en el plano, la trayectoria de la herramienta puede ser planeada manualmente especificando los puntos por donde debe pasar la herramienta en secuencia. El método de ``editar la trayectoria al azar´´ es usado en los CAPS (sistemas de programación automático conversacional), este método de enseñanza manual puede tratar cualquier forma de cavidad compleja, el cual es un camino practico para trabajar con este tipo de geometrías. En este método, sin embargo, el funcionamiento de la trayectoria depende de la habilidad del programador. Además, cuando la geometría es muy complicada, no es fácil especificar todos los puntos detallados sin modificar los límites de la cavidad y de las islas.

Como una alternativa, el problema puede ser formulado dentro de un TSP (problema del vendedor viajero) descomponiendo el área a remover en un grupo de puntos en la rejilla para ser visitados por la herramienta. Solucionando el problema del TSP se puede obtener una trayectoria con la distancia mínima del recorrido pero esta solución se puede convertir en un serio problema para los algoritmos convencionales. Por el método de redes neuronales, el TSP ha sido solucionado satisfactoriamente y resultados de investigaciones anteriores demuestran que el procedimiento da una solución óptima rápidamente, incluso si el número de ciudades aumenta significativamente.

Figura 27. Planeación de la trayectoria al azar



Fuente: **SUK-HWAN, Suh y YANG-SOO, Shin.** Neural network modeling for tool path planning of the rough cut in complex pocket milling. JOURNAL OF MANUFACTURING SYSTEMS. 1996. Pág.295-304.

El problema del vendedor viajero TSP es un problema de optimización combinatoria (NP-hard), lo cual sugiere que no hay algoritmos eficientes que resuelvan este problema de forma exacta y si hay algoritmos exactos para su uso, estos solamente pueden usarse en problemas de tamaño pequeño o medio para resolverlos. La complejidad reside en la enorme cantidad de posibles caminos, muchos de ellos de longitud similar. Para N ciudades, el numero de rutas alternativas es de N!/2N. De esta forma, para 5 ciudades existen 12 caminos posibles; para 10, el número de rutas es de 181.440, y para 50, del orden de $3x10^{62}$.

El problema del vendedor viajero (TSP) ha sido tratado por redes neuronales del tipo Hopfield, Kohonen, Guilty y la red elástica propuesta por Durbin y Willshaw. Hopfield utiliza *n* x *n* neuronas para representar un tour de *n* ciudades. Las *n* x *n* neuronas, son agrupadas en *n* grupos de *n* neuronas. Cada grupo de n neuronas, es utilizado para representar la posición en el tour de una ciudad particular. Los resultados encontrados por Hopfield son inferiores a 30 ciudades, con gran esfuerzo computacional.

La red elástica propuesta por Durbin y Willshaw utiliza el enfoque anterior para la resolución del TSP y para 30 ciudades, requieren 70 neuronas. La red elástica, adaptativamente forma un tour inicialmente pequeño, situado en el centro del plano que gradualmente y no uniformemente es resituado.

La red Guilty, puede ser considerada como un sistema neural híbrido, en la cual los conceptos propios de las RNA más representativos son: el Aprendizaje Competitivo y la aplicación de la Vecindad Topológica de Kohonen. Es del tipo adaptativa, entrenada con la presentación de los datos de entrada en forma reiterativa, aplicando en esta etapa de aprendizaje la autoorganización.

A continuación se describe la red neuronal utilizada en el trabajo de grado por la capacidad de trabajar con un número mayor de ciudades y sus grandes ventajas sobre las demás redes neuronales.

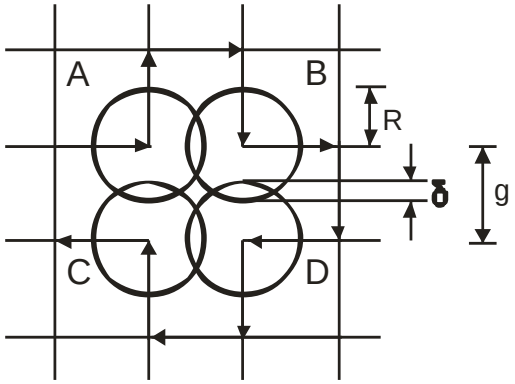
Las Redes Adaptativas (Kohonen), poseen la característica de autoorganización, en contraste a la red de Hopfield no es entrenada con los datos del problema sino configurada con éstos, permitiendo alcanzar un estado estable. La propiedad básica de esta red, es construir un mapeo desde un conjunto con cierto orden topológico a un continuo o discreto conjunto topológico que conserve las propiedades del conjunto inicial. Utilizar este concepto en la resolución del TSP, conlleva a tomar un arreglo unidimensional de neuronas y mapear las neuronas sobre las ciudades, tal que dos posiciones vecinas en el arreglo correspondan a dos ciudades vecinas en el tour.

Específicamente, la solución aproximada es: (1) generar los puntos de la herramienta, (2) solucionar el problema del TSP, (3) modificación parcial de los segmentos de la trayectoria que sobresalen en las islas y/o las esquinas agudas. Detalles de cada procedimiento se dan a continuación.

3.2.2 Generando puntos de la herramienta. Para un tamaño y distancia de sobreposición de la herramienta dados, los puntos de la herramienta pueden ser generados primero formando un mapa cuadriculado, después identificando el área interna y removiendo los puntos seleccionados cercanos al límite de la cavidad y las islas.

R y **δ** son el radio de la herramienta y la distancia de sobreposición. Entonces el intervalo del mapa cuadriculado puede ser definido como **$g = 2R - \delta$.** Determinando la distancia de sobreposición de la herramienta considerando que: (1) el número de puntos de la herramienta (y por lo tanto la longitud de la trayectoria de la herramienta) aumentan al aumentar la distancia de sobreposición, también (2) cuando la distancia de sobreposición disminuye puede existir área sin cortar como muestra la figura 28. Mas precisamente, hay área sin cortar si el intervalo del mapa cuadriculado es mayor que $\sqrt{2} * R$ en el caso en que todos los puntos internos (A, B, C, D) no estén conectados. Basándose en esto, la distancia de sobreposición **$\delta \geq (2-\sqrt{2})R$,** o **$g \leq \sqrt{2}\,R$.**

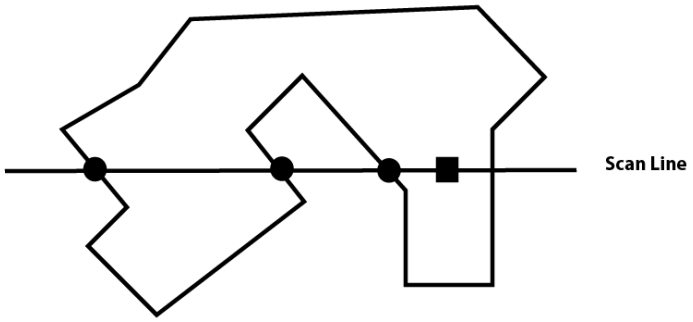
Figura 28. Área sin cortar



Fuente: Ibíd. Pág. 297.

- **Identificando puntos internos.** Para determinar los puntos de la herramienta en el mapa cuadriculado, cada punto en la cuadricula debe ser verificado para ver si pertenece al área removible (área interna de la cavidad). La prueba puede hacerse dibujando una línea de verificación pasando a través de los puntos de la cuadricula, después contando el número de puntos de intersección con la cavidad a la izquierda del punto; es decir, el número impar significa que el punto de la rejilla esta dentro del área a remover (figura 29).

Figura 29. Identificación de puntos internos por la línea de verificación



Fuente: Ibíd. Pág. 297.

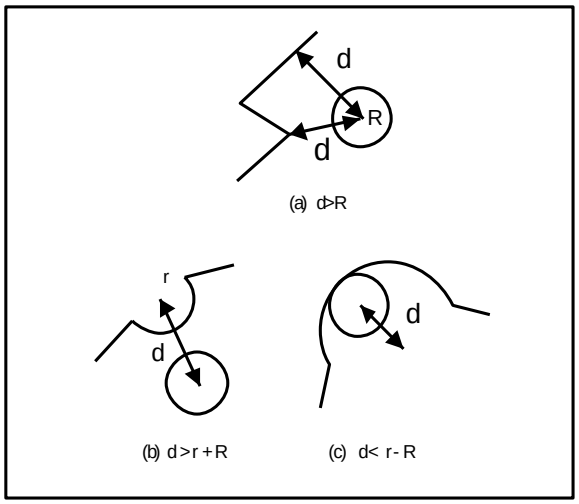
- **Remoción de puntos límites.** Para determinar los puntos de la herramienta, los puntos internos que causan sobrecorte pueden ser removidos. La distancia entre el centro de la herramienta y el límite de la cavidad puede ser

mayor o igual que la suma del radio de la herramienta (R) y la tolerancia del corte final (ε). En otras palabras, un punto interno se convierte en punto de la herramienta, si:

$$|l\text{-}p| \geq R + \varepsilon \qquad (14)$$

Donde **l** es el punto interno y **p** es la cavidad o límite de la isla. La figura 30 ilustra el criterio de distancia para varios limites de cavidades, asumiendo ε = 0.

Figura 30. Criterio de distancia para la remoción de puntos limites



Fuente: Ibíd. Pág. 298.

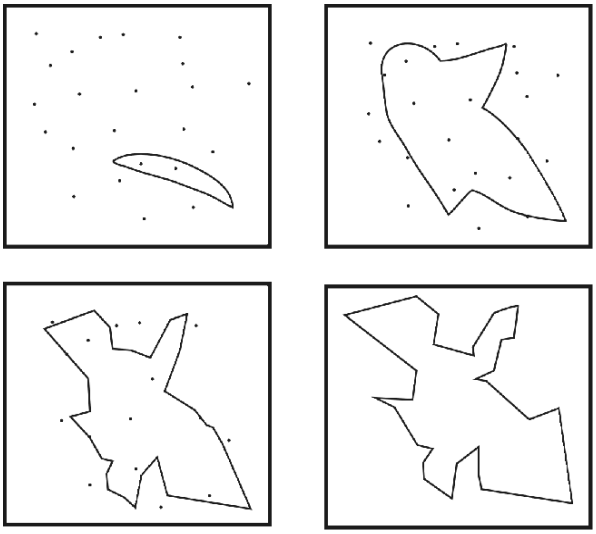
3.2.3 Generación de la trayectoria por un SOM modificado.

- **Método SOM básico.** El método SOM presentado inicialmente[5] comienza por un anillo inicial (o banda elástica) compuesta de un sistema de nodos en el plano, seguida por una evolución iterativa tal que todos los puntos de ciudades

[5] **SUK-HWAN, Suh y YANG-SOO, Shin.** Neural network modeling for tool path planning of the rough cut in complex pocket milling. JOURNAL OF MANUFACTURING SYSTEMS. 1996. Pág.295-304.

son dibujados dentro del cuadrilátero. Si todos los puntos de ciudades son dibujados dentro del cuadrilátero, la conexión de estos puntos ciudades produce una solución óptima al TSP (en el sentido de la distancia). La figura 31 muestra los conceptos tomados de SUK-HWAN, Suh y YANG-SOO, Shin.

Figura 31. Proceso de evolución del SOM



Fuente: Ibíd. Pág. 298.

Inicialmente se considera una Red con una capa de entrada formada por dos neuronas, una para cada coordenada de las ciudades, y una capa de salida formada por un vector con tantas neuronas como ciudades incluya el problema, con lo que se establece una vecindad lineal en dicha capa. El vector se 'cierra', de forma que la neurona de la última posición se considera vecina de la neurona de la primera posición. La única misión de la capa de entrada es pasar las coordenadas de las ciudades a las neuronas de la siguiente capa, no realizando ningún tipo de procesamiento. El algoritmo para entrenar la Red es el siguiente:

- **Algoritmo 1**

1º- Poner el contador de iteraciones a 0.

2º- Poner el contador de patrones presentados a 0.

3º- Seleccionar una ciudad del conjunto de ciudades y presentarla a la Red.

4º- Determinar la neurona ganadora para esa ciudad.

5º- Actualizar los pesos de la ganadora y su vecindad.

6º- Incrementar en 1 el contador de patrones presentados.

7º- Si el número de patrones presentados es igual al número de ciudades, incrementar en 1 el contador de iteraciones. Si no, ir al paso 3º.

8º- Determinar la neurona ganadora para cada ciudad. Si toda ciudad tiene una neurona a distancia menor que el error prefijado, acabar.

9º- Actualizar la tasa de aprendizaje y el radio de vecindad.

10º- Si el número de iteraciones no es el máximo permitido, ir al paso 2º.

Antes de aplicar el algoritmo de entrenamiento, se iniciarán los pesos con valores aleatorios entre 0 y 1, el número máximo de iteraciones del algoritmo, el error máximo permitido para el ajuste entre las ciudades y sus neuronas asociadas, la tasa de aprendizaje y el radio de vecindad. Las ciudades se pueden ir tomando por rotación dentro del conjunto de ciudades o se pueden elegir de forma aleatoria. La neurona ganadora para una ciudad será la que esté a menor distancia, es decir, la que tenga un vector de pesos más próximo a las coordenadas de la ciudad. La medida de distancia considerada es la Euclídea.

$$d\ (wij,\ x) = (\sum\ (wij_K - X_K\ )^2\ )^{1/2} \qquad (15)$$

La zona de vecindad alrededor de la neurona vencedora en la que se encuentran las neuronas cuyos pesos son actualizados se puede reducir en cada iteración del proceso de ajuste de pesos, con lo que el conjunto de neuronas que pueden considerarse vecinas cada vez es menor. sin embargo, en la práctica es habitual considerar una zona fija en todo el proceso de entrenamiento de la red.

La expresión aplicada para actualizar los pesos es la indicada por

$$W_i(t+1) = W_i(t) + \alpha(t)\ (X_k - W_{ij\ k}) \qquad (16)$$

La tasa de aprendizaje, α (t), se actualiza de acuerdo con la expresión:

$$\alpha\ (t) = \alpha_0 + (\alpha_f - \alpha_0)\ t\ /\ t\ \alpha \qquad (17)$$

α_0 : tasa de aprendizaje inicial (< 1)

α_f : tasa de aprendizaje final (»0.01)

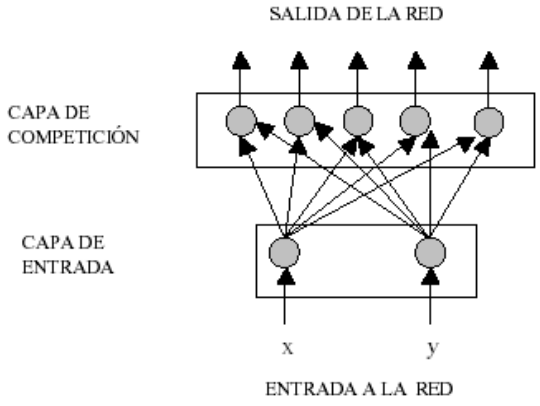
t α: máximo número de iteraciones hasta llegar a α_f

- **Red autoasociativa.** En este apartado se describe con detalle la Red diseñada en base a los resultados de todas las características analizadas en el apartado anterior. La solución propuesta utiliza un vector de neuronas de tamaño mayor o igual que el número de ciudades del problema. Cuando una neurona gane para varias ciudades, se añadirá una nueva neurona a su lado. Para evitar que el número de neuronas crezca sin límite, se eliminarán las neuronas no ganadoras. Como en el algoritmo básico analizado, se considerarán una tasa de aprendizaje decreciente con el tiempo.
- Inicialmente se pueden insertar/eliminar numerosas neuronas en cada bucle del algoritmo, pero se observa que a medida que avanzan las iteraciones cada vez es menor el número de operaciones de inserción/eliminación. Si al aproximarnos al número máximo de iteraciones permitidas para el algoritmo, se observa que aún oscila demasiado el número de neuronas del mapa, debería repetirse el proceso con un número mayor de iteraciones. El objetivo es proporcionar suficientes iteraciones, de modo que al iniciarse el ajuste fino del mapa, la mayor parte, si no son todas las neuronas, sean ya ganadoras para una y sólo una ciudad. Se presentará iterativamente el conjunto de ciudades, de forma que los pesos de las neuronas se aproximen a las coordenadas de las ciudades. Al finalizar el proceso, el número de neuronas debe coincidir con el de ciudades (habrá una neurona asociada a cada ciudad). Para determinar el camino a seguir, se irá de la ciudad asociada a la neurona i a la asociada a la i+1, para i=1, 2,…N, recorriendo todo el vector de neuronas.

Para cerrar el camino, el último tramo vendrá dado por la ruta que une la ciudad asociada a la neurona N con la asociada a la neurona 1. La distancia del camino vendrá dada por la suma de las distancias entre los sucesivos pares de ciudades del camino.

- **Arquitectura de la red.** El problema considerado se define como un conjunto de pares de valores (x, y), que representan las coordenadas de las ciudades. Se utiliza un Mapa Autoorganizativo formado por las dos capas ya descritas: la capa de entrada y la capa de competición. La **capa de entrada** está formada por dos neuronas, cada una de las cuales recibe una de las coordenadas de la ciudad presentada como entrada, (x, y). La salida de esta capa será la entrada de la capa de competición. Su única misión es la de distribuir las entradas a todas las neuronas de la siguiente capa. La capa de entrada será igual para todos los problemas. La **capa de competición** inicialmente puede tener un número variable de neuronas. Se puede partir de un número de neuronas igual al de ciudades del problema. Como para encontrar una solución se necesitan como mínimo tantas neuronas como ciudades, no se considera la posibilidad de partir de un número de neuronas menor que el de ciudades. Cada una de estas neuronas recibirá dos entradas, que corresponden a las salidas de la capa anterior y son, por tanto, las coordenadas de la ciudad presentada como entrada a la Red. Durante el entrenamiento se podrán añadir o eliminar neuronas de esta capa.

Figura 32. Red



Fuente: PÉREZ DELGADO, Mª Luisa y MARTÍN MARTÍN, Quintín. Diseño de una Red Neuronal Autoasociativa para Resolver el Problema de Viajante del Comercio (Traveling Salesman Problem, tsp). 27 Congreso Nacional de Estadística e Investigación Operativa Lleida, 8-11 de abril de 2003.

- **Método de selección de ciudades.** Se pueden utilizar dos métodos diferentes para seleccionar la ciudad que se tomará como entrada:

Rotación: en cada iteración de entrenamiento se empieza tomando la primera ciudad de la lista de ciudades y se van tomando en orden, hasta llegar a la última de la lista.

Aleatorio: se selecciona aleatoriamente una de las ciudades de la lista. Cuando se ha presentado un número de ciudades igual a N, concluye la iteración. Hay que tener en cuenta que utilizando éste método no se garantiza que se presenten todas las ciudades en cada una de las iteraciones del algoritmo. Sin embargo, está comprobado que la selección aleatoria genera buenos resultados.

- **Métrica de distancia**. Se pueden aplicar varias métricas para determinar la neurona ganadora para una ciudad. Se calculará la distancia entre las coordenadas de la ciudad y los pesos de todas las neuronas, tomando como ganadora la que de un valor menor. La medida de distancia considerada en este trabajo es la Euclídea.

La distancia entre la ciudad Ck=(Xk, Yk) y la neurona j, cuyo vector de pesos es Wj=(Wj 1,Wj 2), será:

$$d(Ck, \mathbf{w_j}) = ((Xk - Wj1)^2 + (Yk - Wj2)^2)^{1/2} \qquad (18)$$

- **Adición y eliminación de neuronas.** Como ya se comentó, uno de los inconvenientes del algoritmo básico de Kohonen es que existe la posibilidad de que haya neuronas que no ganen para ninguna ciudad, mientras que otras ganen para varias. Para evitar este problema, habría que llevar las neuronas no ganadoras junto a las ganadoras múltiples.

Una de las posibilidades estudiadas consiste en añadir una nueva neurona al mapa siempre que se produzca una victoria múltiple de alguna neurona (es decir, duplicar la neurona siempre que se presente una ciudad y gane una neurona que ya ha ganado para una ciudad diferente).

El algoritmo utilizado para insertar y eliminar neuronas es el siguiente:

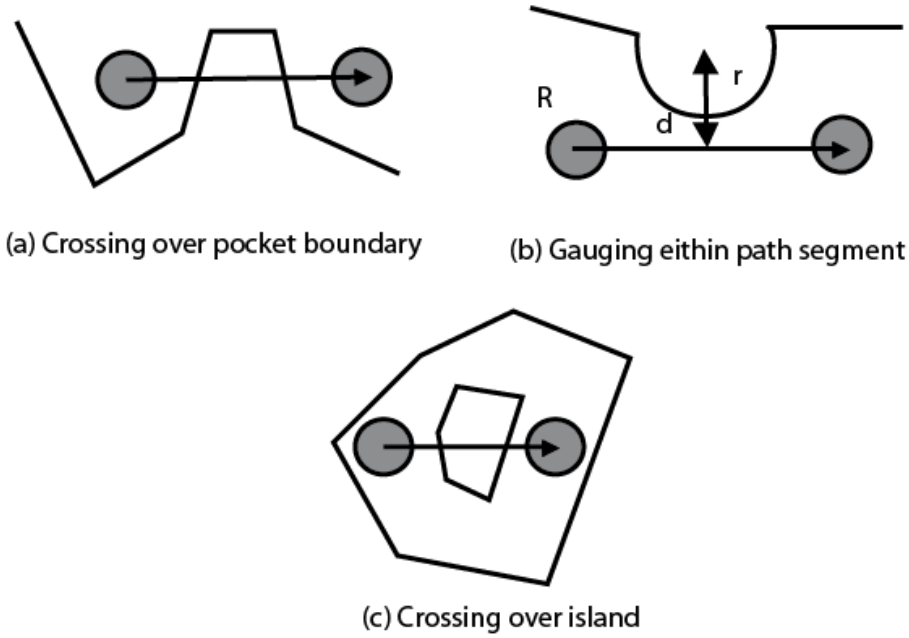
- **Algoritmo 2**

1º- Se inicia el contador de neuronas a 1 (k=1)
2º- Si el contador de victorias de la neurona nk es mayor que 1, Si el contador de victorias de n_k-1 es mayor que el de n_k+1, insertar una neurona (n*) entre n_k y n_{k-1}, cuyos pesos serán:
$w\ n^*_j = (0{,}04\ w_{k-1\ j}) + (0{,}95\ w_{kj}) + (0{,}01\ \sigma_j)$ para j=1, 2
si no, insertarla entre nk y nk+1, siendo sus pesos:
$w\ n^*_j = (0{,}04\ w_{k+1\ j}) + (0{,}95\ w_{kj}) + (0{,}01\ \sigma_j)$ para j=1, 2
donde σ_j es un valor aleatorio entre 0 y 1.
El contador de última victoria de n* se hace igual al número de la iteración actual del algoritmo de entrenamiento.
Los pesos de n_k se hacen iguales al 99% de sus pesos actuales.
3º- Si el contador de victorias de la neurona n_k es 0, no resultó ganadora para ninguna ciudad al presentar todo el conjunto de ciudades y su última victoria no tuvo lugar en las 3 últimas iteraciones, eliminarla.
4º- Se incrementa el contador (k=k+1)
5º- Si k es mayor que el número de neuronas, acabar. Si no, volver al paso 2º.

3.2.4 Modificación de la trayectoria. Como la trayectoria de la red neuronal es generada basada en la conexión de puntos inteligentes, esto no garantiza una trayectoria optima entre los puntos de la herramienta en el mecanizado de cavidades. Este problema puede ser clasificado en dos partes: (1) la trayectoria no es factible, y (2) la trayectoria es indeseable. Así, el segmento de la trayectoria que posee estas complicaciones necesita ser modificado.
El primer problema puede ser detectado por el cálculo de la distancia entre la trayectoria de la herramienta y los límites de la cavidad. Si la distancia es menor o

igual al radio de la herramienta, los movimientos de ``retraer la herramienta ´´ y ``bajar la herramienta ´´ son realizados en los puntos de inicio y finalización del segmento. En este caso la operación de perforado debe ser precedida por un punto donde la herramienta baje (esto es hecho automáticamente por el programa). El contorno de la cavidad o la isla está compuesto de líneas y arcos, los esquemas para la corrección de la trayectoria se muestran en la figura 33.

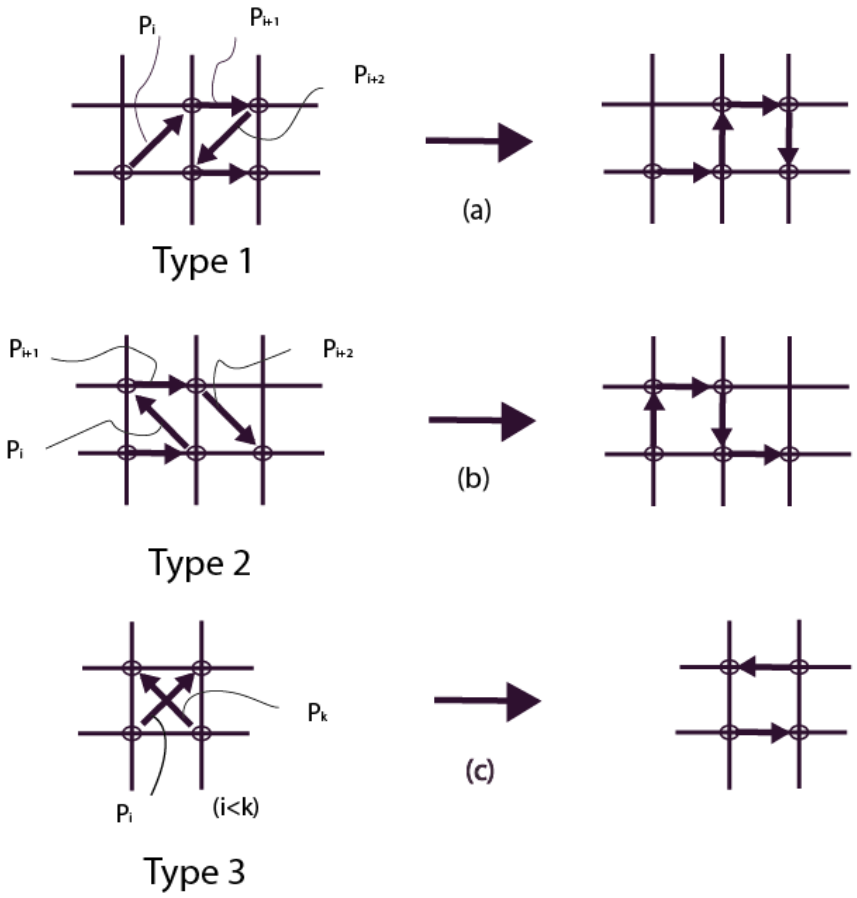
Figura 33. Modificación de segmentos de la trayectoria no factibles



Fuente: **SUK-HWAN, Suh y YANG-SOO, Shin.** Neural network modeling for tool path planning of the rough cut in complex pocket milling. JOURNAL OF MANUFACTURING SYSTEMS. 1996.

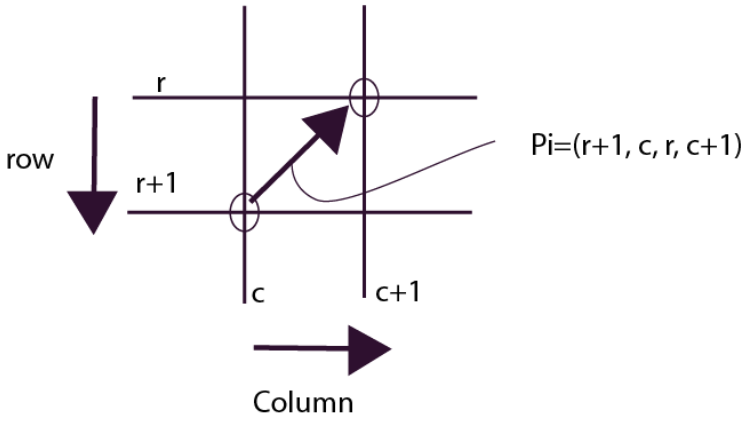
El segundo problema ocurre cuando la trayectoria es diagonalmente conectada (figuras 34a y 34b) o cruzadas (figura 34c). El algoritmo implementado detecta los tres tipos y los modifica como se muestra en la figura 34. Por la modificación, la longitud de la trayectoria es reducida de g (2+2√2) a 4g para tipos (1 y 2) y de 2g√2 a 2g para tipo 3.

Figura 34. Modificación de segmentos de trayectoria indeseables



Fuente: Ibíd. Pág., 300.

Figura 35. Segmento de la trayectoria sobre el mapa cuadriculado



Fuente: Ibíd. Pág., 300.

Al tomar $\mathbf{P_{i\,=}\,(S_{i,1},\ S_{i,2},\ E_{i,1},\ E_{i,2})}$ como el segmento de la trayectoria sobre el mapa cuadriculado cono muestra la figura 35. Entonces, el algoritmo para modificación de la ruta para tipos 1 y 2 es como sigue:

- Algoritmo 3 – Modificación de la trayectoria para tipos 1 y 2

1. Lectura P_i, i Є [1: n], y dejar i = 0.
2. i = i+1.
3. Si i > L, parar, donde L = n+1 si la trayectoria total P es cerrada

   n-2 de otro modo
4. Verificar si $\mathbf{P_i}$ es una trayectoria diagonal:

   Si $|S_{i,k} – E_{i,k}| = 1$, donde k Є [1:2], ir a paso 5.

   De otro modo, ir a paso 2.
5. Verificar si $\mathbf{P_{i+2}}$ es una ruta diagonal y paralela a $\mathbf{P_{i.}}$ (de ahora en adelante, si **P** es cerrada, **índice segmento = MOD (numero de segmento, n**))

   Si $\mathbf{(|S_{i,k} – E_{i+2,k}| = 1)}$ y $\mathbf{(|E_{i,k} – S_{i+2,k}| = 1),}$ donde **k Є [1:2],** entonces ir a Paso De otro modo ir a paso 2.
6. Modelo MATCHING:

   Si $(\sum_k |S_{i,k} – E_{i+1,k}| = 3)$ y $(\sum_k |E_{i+1,k} – E_{i+3,k}| = 1)$, entonces ir a paso 7

   Todo lo demás si $(\sum_k |S_{i,k} – E_{i+1,k}| = 1)$ y $(\sum_k |S_{i+1,k} – S_{i-1,k}| = 1)$,

   Entonces ir a paso 8, todo lo demás ir a paso 2.
7. Modificación para tipo 1:

   $P_{i\,=}\,(S_{i,1},\ S_{i,2},\ E_{i+2,1},\ E_{i+2,2})$,

   $P_{i+1\,=}\,(S_{i+3,1},\ S_{i+3,2},\ E_{i,1},\ E_{i,2})$,

   $P_{i+2\,=}\,(S_{i,1},\ S_{i,2},\ E_{i,1},\ E_{i,2})$,

   $P_{i+3\,=}\,(S_{i+2,1},\ S_{i+2,2},\ E_{i+3,1},\ E_{i+3,2})$.

   Ir a paso 2.
8. Modificación para tipo 2:

   $P_{i-1\,=}\,(S_{i-1,1},\ S_{i-1,2},\ E_{i,1},\ E_{i,2})$,

   $P_{i\,=}\,(S_{i+1,1},\ S_{i+1,2},\ E_{i+1,1},\ E_{i+1,2})$,

$P_{i+1} = (S_{i+2,1}, S_{i+2,2}, E_{i-1,1}, E_{i-1,2})$,

$P_{i+2} = (S_{i,1}, S_{i,2}, E_{i+2,1}, E_{i+2,2})$.

Ir a paso 2.

Similarmente, el algoritmo para modificación de la trayectoria tipo 3 puede ser resumido como sigue:

- Algoritmo 4 – Modificación de la trayectoria para tipo 3.

1. Lectura P_i, i Є [1: n], y dejar i = 0.
2. i = i+1.
3. Si i > n, parar.
4. Verificar si P_i es una trayectoria diagonal:

   Si $|S_{i,k} – E_{i,k}| = 1$, donde k Є [1:2], ir a paso 5.

   De otro modo, ir a paso 2.
5. Hallando P_k, k > i+1, que es diagonal y cruza P_i.

   Para k = i+2,….., N{ si $P_k = (S_{i,1}, E_{i,2}, E_{i,1}, S_{i,2})$ o $P_k = (E_{i,1}, S_{i,2}, S_{i,1}, E_{i,2})$,

   Ir a paso 6}

   Ir a paso 2.

6. Modificando para tipo 3:

   $P_i = (S_{i,1}, S_{i,2}, S_{k,1}, S_{k,2})$, $P_k = (E_{i,1}, E_{i,2}, E_{k,1}, E_{k,2})$

   Ir a paso 2.

El estudio de redes neuronales con una descripción detallada de su funcionamiento y mecanismo de aprendizaje permite realizar la emulación de estas redes para resolver diferentes problemas mediante la codificación (utilizando un lenguaje de programación cualquiera) de los algoritmos que se presentan a lo largo del trabajo de grado. Esta codificación se realizo en MATLAB por las razones que mencionaremos a continuación.

A menudo nos hacemos preguntas como “¿qué es lo que hace que Matlab sea tan bueno?” y “¿por qué debería elegir Matlab en lugar de la herramienta X?”. La respuesta es la productividad. La utilización de Matlab es simplemente la forma más productiva que se ha encontrado para crear aplicaciones para Windows. Se puede condensar la productividad de esta herramienta de diseño software en un pentágono que refleja los cinco atributos más importantes:

- La calidad del entorno de desarrollo visual.
- La velocidad del compilador frente a la eficiencia del código compilado.
- La potencia del lenguaje de programación frente a su complejidad.
- La flexibilidad y la escalabilidad de la arquitectura de la base de datos.
- Los métodos de diseño y de utilización recomendados por el entorno.

Aunque probablemente habrá otros muchos factores, como temas de utilización, soporte de otros productos y demás, se ha llegado a la conclusión que este modelo simplista explica con bastante precisión por que elegimos Matlab.

Finalmente, para comprobar la funcionalidad de los algoritmos neuronales aplicados en el Software, se llevara a cabo una comparación con un programa utilizado actualmente en el fresado de cavidades en este caso MASTERCAM.

4. DESCRIPCION Y FUNCIONAMIENTO DEL SOFTWARE

El software para la optimización de trayectorias de fresado ”NEURAL NET OPTRASOFT” es una herramienta basada en redes neuronales que permite obtener una posible trayectoria optima para el fresado de cavidades complejas por medio de algoritmos y estructuras computacionales basadas en el aprendizaje y comparación de parámetros de distancia según la arquitectura de la red neuronal, utilizando una interfaz practica y funcional que permite la recolección, edición y visualización de información en cada proceso por medio de pantallas, facilitando la interacción entre el usuario, el motor de la red y el funcionamiento del software como se detalla a continuación en los numerales de Descripción del software y Funcionamiento del Software.

4.1 DESCRIPCION DEL PROCESO EN MATLAB

4.1.1 Digitalización de imágenes. En Matlab una imagen a escala de grises es representada por medio de una matriz bidimensional de m x n elementos en donde n representa el numero de píxeles de ancho y m el numero de píxeles de largo y tiene un solo plano, cada elemento de la matriz de la imagen tiene un valor de 0 (negro) a 255 (blanco). Por otro lado una imagen de color RGB (la más usada para la visión computacional) es representada por matriz tridimensional m x n x p, donde m y n tienen la misma significación que para el caso de las imágenes de escala de grises mientras p representa el plano 1, 2 o 3, que para RGB puede ser R=1 para el rojo, G=2 para el verde y B=3 para el azul.

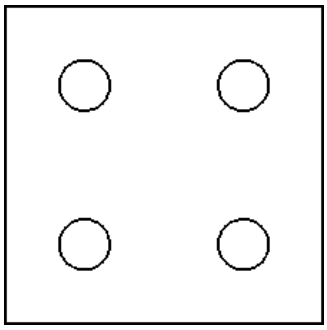
- **Lectura de imágenes y digitalización de puntos.** Para leer imágenes contenidas en un archivo en MATLAB se utiliza la función imread que tiene la siguiente sintaxis:

>> imread(‘nombre_del_archivo”);

Para utilizar una imagen (Ver figura 36) para su procesamiento en MATLAB se almacena en una variable:

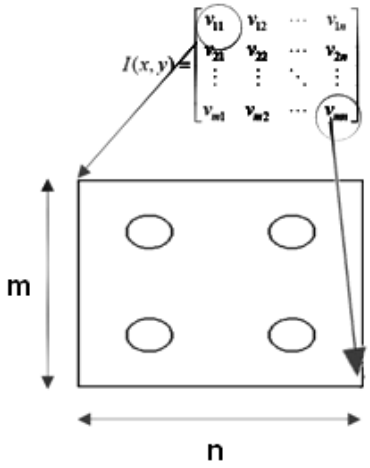
>> imagen = imread(‘cavidad.jpg”);

Figura 36. Imagen de la cavidad en formato de gráficos BMP



Con ello el archivo de imagen se convierte a niveles de intensidad en el intervalo de [0, 255] (Ver figura 37) que quedan almacenados en la variable de MATLAB llamada imagen.

Figura 37. Identificación de coordenadas en Matlab



Para graficar la matriz de pixeles (Ver figura 38) se utiliza el comando *imshow*:

>> imshow(imagen);

Figura 38. Representación en MATLAB de la imagen de la cavidad

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
0	0	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
0	0	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
0	0	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
0	0	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
0	0	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
0	0	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
0	0	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
0	0	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
0	0	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
0	0	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
0	0	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
0	0	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
0	0	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
0	0	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
0	0	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255

- **Acceso a los valores de la imagen.** El acceso a pixeles es una operación necesaria para el procesamiento de la imagen, en MATLAB se realiza indicando la posición del pixel que se requiere conocer y MATLAB retorna su valor, por ejemplo para saber el valor del pixel ubicado en la coordenada (10,10) se accede a la posición de la matriz que representa la imagen de la siguiente manera (Ver figura 38):

```
>> pixel = imagen(10,10);
```

Esto retornará para el ejemplo, el valor de pixel = 255, ya que la posición x = 10, y= 10, corresponde a una zona blanca de la imagen

Para un tamaño y distancia de sobreposición de la herramienta dados, los puntos de la herramienta pueden ser generados primero formando un mapa cuadriculado, después identificando el área interna y removiendo los puntos seleccionados cercanos al límite de la cavidad y las islas.

Para determinar los puntos de la herramienta en el mapa cuadriculado, cada punto en la cuadricula debe ser verificado para ver si pertenece al área removible (área

interna de la cavidad). La prueba puede hacerse dibujando una línea de verificación pasando a través de los puntos de la cuadricula, después contando el número de puntos de intersección con la cavidad a la izquierda del punto; es decir, el número impar significa que el punto de la rejilla esta dentro del área a remover.

Para generar la rejilla de puntos y elegir los puntos que pertenecen a la cavidad se sigue el siguiente algoritmo:

De acuerdo a los parámetros de la herramienta se genera un conjunto de puntos equidistante (cuadricula).

Se realiza el procesamiento de la imagen y se identifican las islas, obteniendo las coordenadas de los pixeles que pertenecen a sus contornos.

Se comprueba cuales puntos de la cuadricula pertenecen al área fuera de los contornos.

Si el punto está fuera del contorno:

- Se determina la distancia respecto a los bordes de la imagen.
- Si la distancia es mayor que el radio de la fresa para ese punto, se almacena como punto de la rejilla. De lo contrario se descarta.

Programación de digitalización en Matlab

```
inicio construirRejilla
leer radioFresa, distanciaRadios;
leer mapa_cavidad;
res <-  resolución del mapa;
rejilla <- generarPuntosEquidistantes();
puntos_cavidad <- determinar_Puntos_Fuera_de_Islas();
bordes <-   intersección(mapa_cavidad,  puntos_cavidad);
Para cada punto i que pertenece a rejilla
        Si  rejilla(i) pertenece a puntos_cavidad
                Para cada j que pertenece a
```

```
                    dist <-  min( distanciaEuclideana(rejilla(i), bordes(j)) )
                    Si  dist>=radioFresa
                            Remover(rejilla(i));
            Termina j
Termina i
Termina construirRejilla
```

4.1.2 Procesamiento de la imagen. Para determinar la geometría de la cavidad se aplican operaciones morfológicas y de segmentación sobre la imagen con el fin de determinar:

Coordenadas de contorno de la cavidad.

Coordenadas del contorno de las islas.

Para ello se aplica el concepto de imágenes binarias y operaciones basadas en objetos.

- **Imágenes binarias.** Una imagen binaria es una imagen en la cual cada píxel puede tener solo uno de dos valores posibles 1 o 0. Como es lógico suponer una imagen en esas condiciones es mucho más fácil encontrar y distinguir características estructurales. En visión computacional el trabajo con imágenes binarias es muy importante ya sea para realizar segmentación por intensidad de la imagen, para generar algoritmos de reconstrucción o reconocer estructuras.

La forma más común de generar imágenes binarias es mediante la utilización del valor umbral de una imagen a escala de grises; es decir se elige un valor limite (o bien un intervalo) a partir del cual todos los valores de intensidades mayores serán codificados como 1 mientras que los que estén por debajo serán codificados a cero. En Matlab este tipo de operaciones se realizan de la siguiente forma. Por ejemplo si de la imagen sample quisiera realizarse este tipo de operación de tal forma que los píxeles blancos (cuyo valor es 255) sean considerados como 1 y los que son negros o cero que se mantengan como ceros, esto se escribiría en línea de comandos como:

> >imagenBinaria=(imagenCavidad==255);

- **Operaciones basadas en objetos.** En una imagen binaria, puede definirse un objeto como un conjunto de píxeles conectados con valor 1. Para muchas operaciones la distinción de objetos depende la convención utilizada de conectividad, que es la forma en la que se considera si dos píxel tienen relación como para considerar que forman parte del mismo objeto. La conectividad puede ser de dos tipos, de conexión-4[6] ó bien conexión-8[7]. En la conectividad conexión-8 se dice que el píxel pertenece al mismo objeto si existe un píxel de valor uno en las posiciones 1, 2, 3, 4, 5, 6, 7 y 8. Por su parte la conectividad conexión-4 considera que son dos objetos diferentes relaciona solo a los píxel 2, 4,5 y 7.

La función bwlabel realiza un etiquetado de los componentes existentes en la imagen binaria, la cual puede ser considerada como una forma de averiguar cuántos elementos (considerando como elementos agrupaciones de unos bajo alguno de los dos criterios de conectividad) están presentes en la imagen. La función tiene el siguiente formato:

>> ImageR=bwlabel(ImageS, conectividad);

Donde ImagenR es la imagen resultado que contiene los elementos etiquetados con el número correspondiente al objeto, ImagenS es la imagen binaria que se desea encontrar el número de objetos y conectividad puede ser 4 o 8 (Ver figura 40).

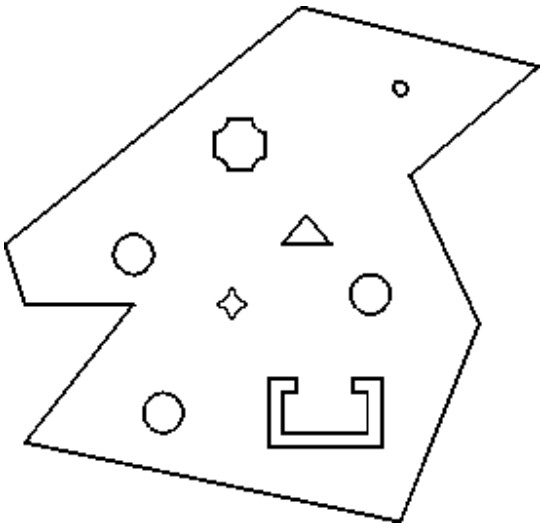
La imagen resultado asigna a cada píxel perteneciente a un determinado objeto según su conectividad la etiqueta perteneciente al número de objeto, mientras que los píxeles en cero no tienen efecto en la operación.

Por ejemplo, para determinar los objetos en la cavidad mostrados en la figura 39.

[6] Se consideran pixeles conectados en direcciones perpendiculares (arriba, abajo, derecha e izquierda).

[7] Se consideran también los pixeles vecinos diagonales.

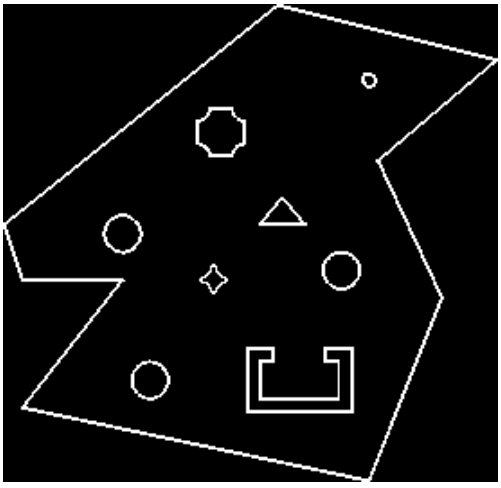
Figura 39. Imagen a procesar



Fuente: autores.

```
>>im = imread('imagenes\cavidad_4_4.bmp'); %Lectura de la imagen
>>notIm = ~im(:,:,1); % Se invierten los colores para poder segmentar
```

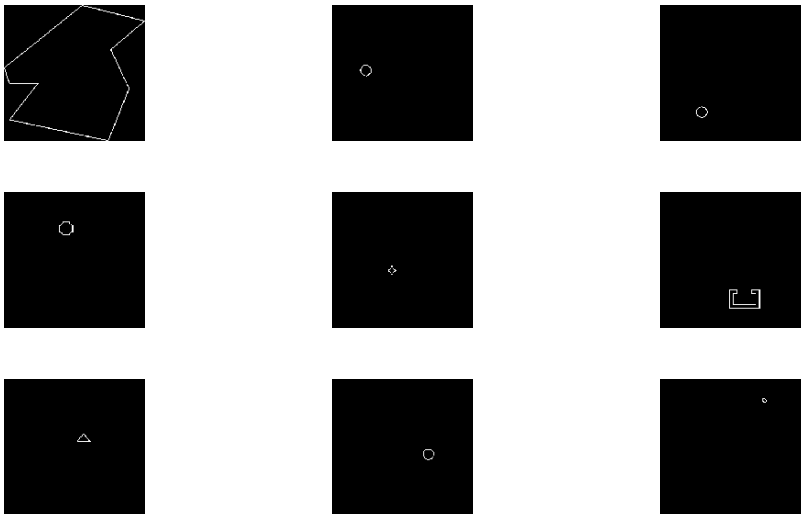
Figura 40. Invertir colores para segmentar la imagen



Fuente: autores.

```
>>[I,L] = bwlabel(notIM); %Se realiza segmentación para obtener los objetos
```

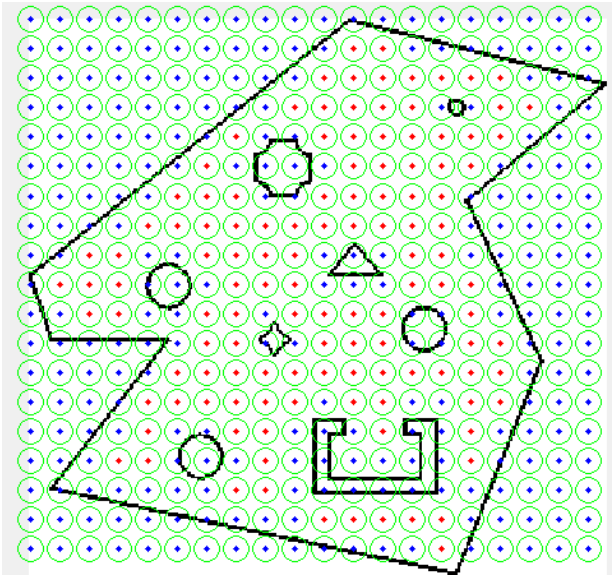
Figura 41. Segmentación para obtener los objetos



Fuente: autores.

Se obtienen las coordenadas de cada isla y del contorno de la cavidad por separado. Con esta información se genera la rejilla de puntos uniforme, que corresponden a cada punto (ver figura 42).

Figura 42. Mapa de la cavidad y rejilla de puntos



Fuente: autores.

En la imagen de la cavidad, los puntos en color rojo cumplen las siguientes condiciones:

1. Están por dentro del borde más externo y están fuera de las islas.
2. Ninguna distancia de un punto rojo a un borde es menor que el radio de la herramienta determinado en los parámetros de entrada.

Para determinar si un punto pertenece a la cavidad o no, se aplica el siguiente algoritmo:

1. Para cada punto Pi que pertenece a la rejilla:
2. Si Pi pertenece al conjunto de puntos dentro de la cavidad:

2.1 Si Pi pertenece a algún conjunto de coordenadas de las islas

2.2 Se descarta el punto y pasa a la siguiente iteración

2.3 De lo contrario se comprueba si el punto está dentro del borde externo de la cavidad

2.4 Si el punto está dentro de la cavidad pero fuera de las islas se pasa como punto aceptado

2.5 Se itera nuevamente hasta evaluar todos los puntos de la rejilla.

3. Para cada punto Qi que pertenece al conjunto de aceptados en el paso 2:

3.1 Se determina la distancia euclídea[8] de Qi a todos los puntos del conjunto conformado por los puntos del borde exterior de la cavidad y de los bordes de las islas.

3.2 Si la distancia euclídea es mayor que el radio de la herramienta se almacena el punto de la rejilla definitivo.

3.3 De lo contrario se descarta el punto y se itera el siguiente Qi

[8] La distancia euclidiana o euclídea es la distancia "ordinaria" (que se mediría con una regla de acero) entre dos puntos de un espacio euclídeo, la cual se deduce a partir del teorema de Pitágoras.

4.2 DESCRIPCIÓN DEL SOFTWARE.

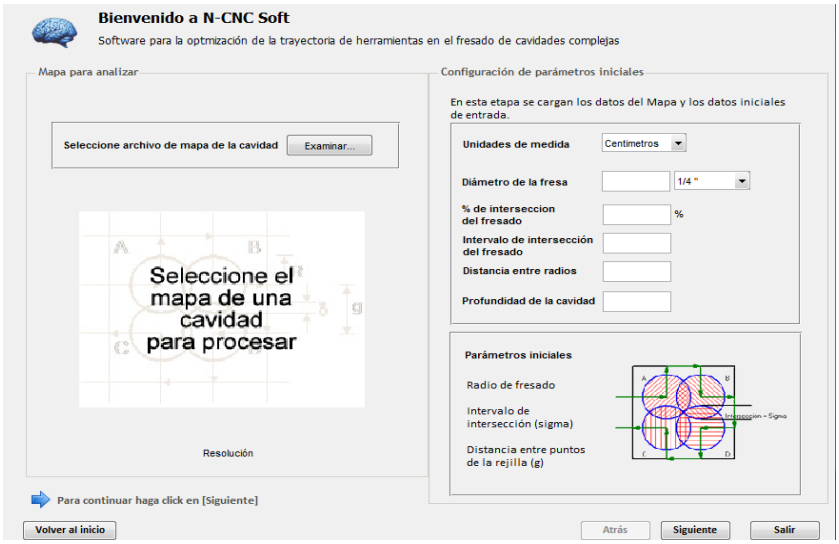
El software NEURAL NET OPTRASOFT Versión 1.0 utiliza una interfaz que presenta una Pantalla de Presentación Splash, una pantalla para cada uno de los cuatro pasos del proceso, y dentro de cada paso su respectivo subproceso, los resultados dan una imagen en 3 dimensiones y un archivo con los puntos de la trayectoria.

4.2.1 Pantalla De Presentación Inicial o Splash. Esta pantalla (Ver Figura 43) se activa en la ejecución inicial del software y constituye la presentación inicial del proyecto, tiene un tiempo de visualización temporal de segundos, terminado este tiempo se visualiza la pantalla principal.

Figura 43. Pantalla de Presentación inicial o Splash

4.2.2 Pantalla Step 1. En el paso 1 (Ver figura 44) se hace la adquisición de los parámetros principales para el proceso. Se divide en dos subprocesos, la adquisición del mapa de datos o imagen a analizar, y la configuración de los parámetros iniciales del mecanizado (Ver anexo 1).

Figura 44. Pantalla de Step 1

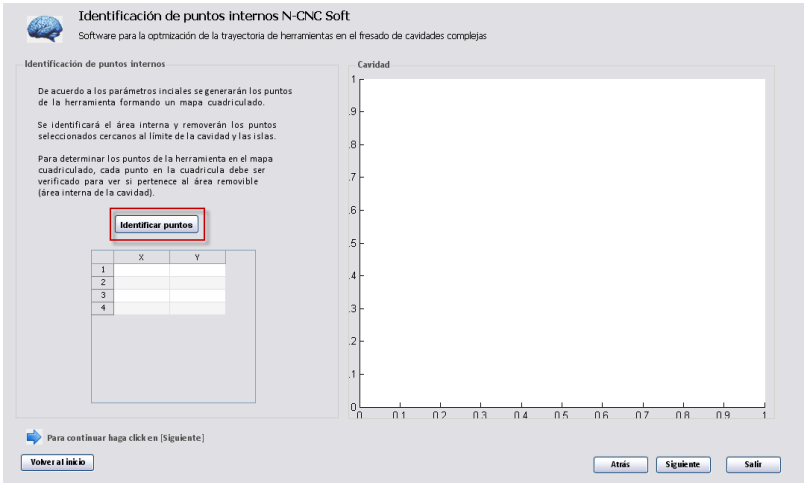


4.2.3 Pantalla Step 2. De acuerdo a los parámetros iniciales se generarán los puntos de la herramienta formando un mapa cuadriculado.

Se identificará el área interna y removerán los puntos seleccionados cercanos al límite de la cavidad y las islas.

Para determinar los puntos de la herramienta en el mapa cuadriculado, cada punto en la cuadricula debe ser verificado para ver si pertenece al área removible, interna de la cavidad (Ver anexo 2).

Figura 45. Pantalla de Step 2

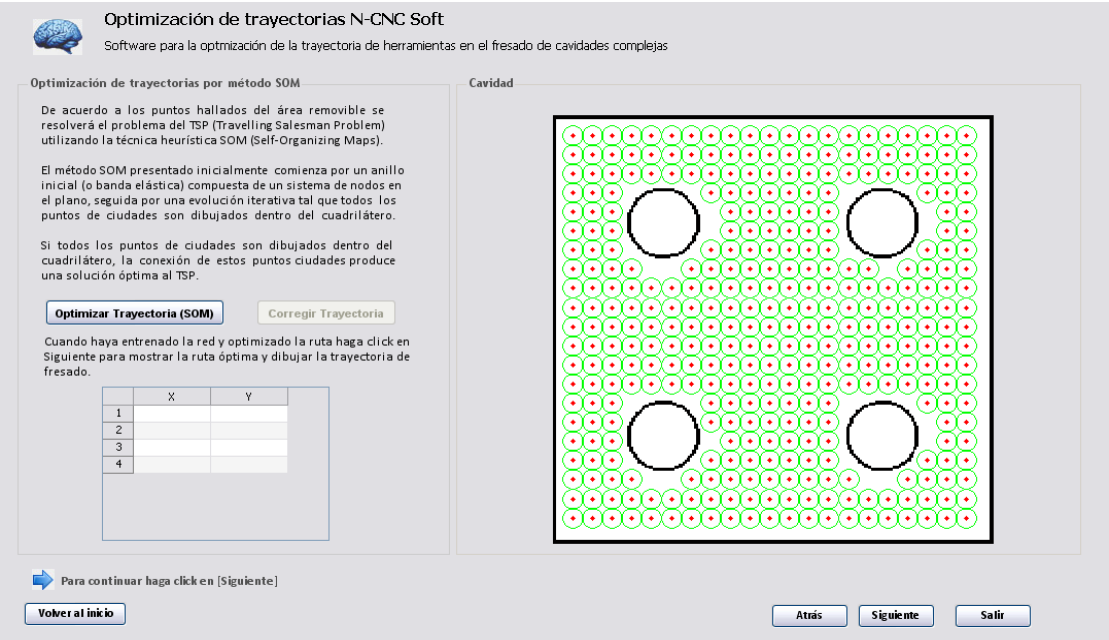


4.2.4 Pantalla Step 3. De acuerdo a los puntos hallados del área removible se resolverá el problema del TSP (Travelling Salesman Problem) utilizando la técnica heurística SOM (Self-Organizing Maps) (Ver anexo 3).

El método SOM presentado inicialmente comienza por un anillo inicial (banda elástica) compuesta de un sistema de nodos en el plano, seguida por una evolución iterativa tal que todos los puntos de ciudades (equivalente a ciudades en el algoritmo teórico) son dibujados dentro del cuadrilátero (Ver figura 46).

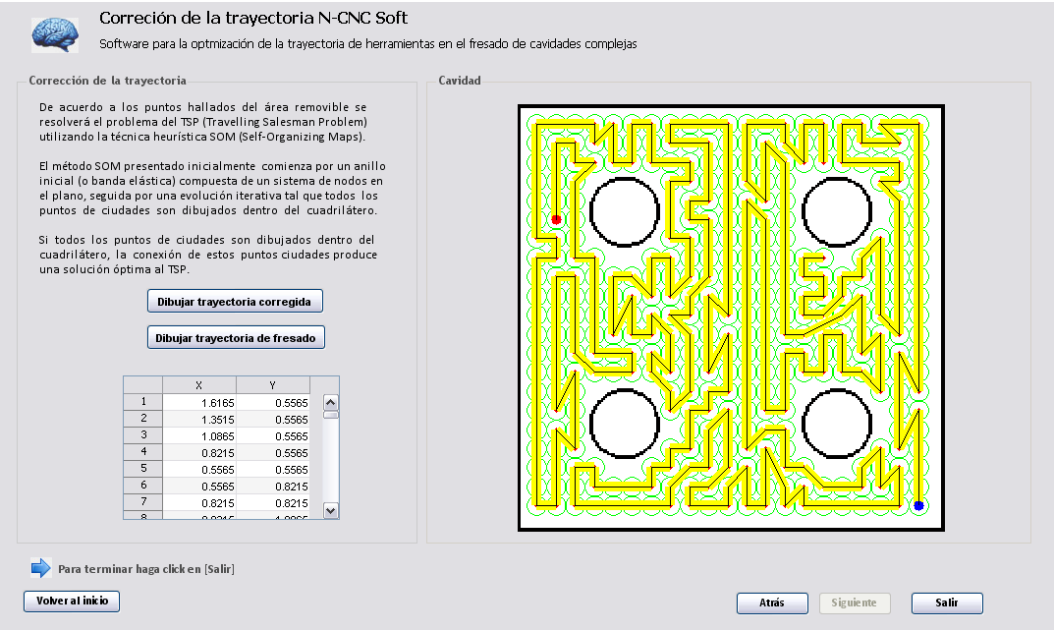
Si todos los puntos son dibujados dentro del cuadrilátero, la conexión de estos puntos produce una solución óptima al TSP.

Figura 46. Pantalla de Step 3



4.2.5 Pantalla Step 4. Se dibujan la trayectoria corregida y la trayectoria de la herramienta (Ver figura 47 y anexo 4).

Figura 47. Pantalla de Step 4



4.2.6 Pantalla Resultados. La pantalla imprime los datos, permite guardar tanto la imagen o cavidad (Ver Figura 48) en cualquier momento del proceso de ejecución como la trayectoria en 3D y las coordenadas de los puntos maquinables de la cavidad (Ver Figura 49).

Figura 48. Pantalla de Imprimir Datos Imagen o Cavidad

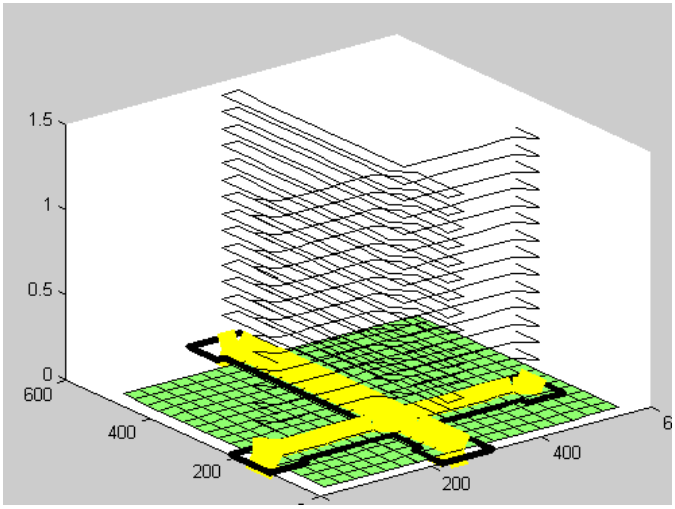
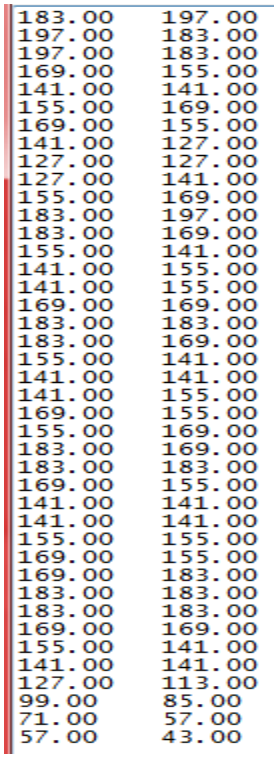


Figura 49. Pantalla de Imprimir Coordenadas de puntos maquinables



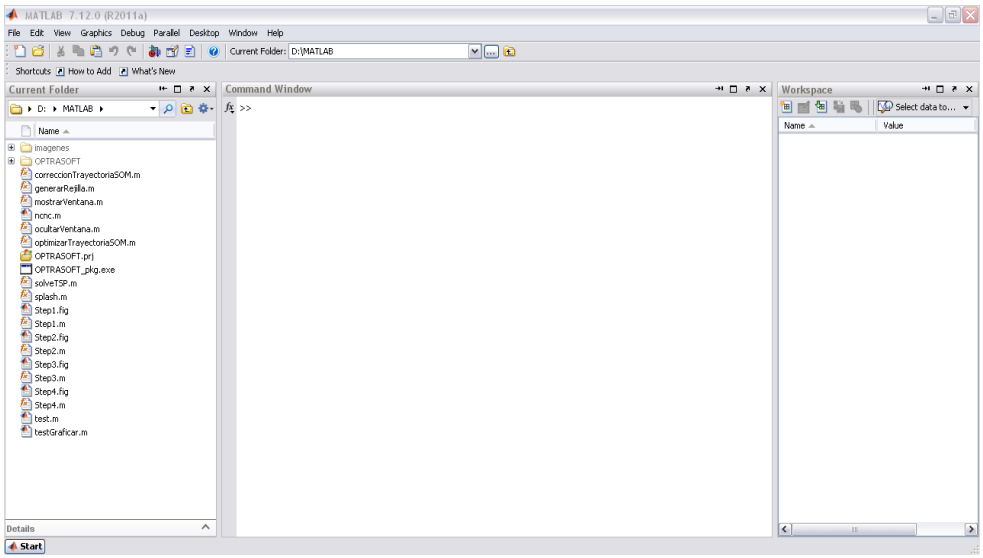
183.00	197.00
197.00	183.00
197.00	183.00
169.00	155.00
141.00	141.00
155.00	169.00
169.00	155.00
141.00	127.00
127.00	127.00
127.00	141.00
155.00	169.00
183.00	197.00
183.00	169.00
155.00	141.00
141.00	155.00
141.00	155.00
169.00	169.00
183.00	183.00
183.00	169.00
155.00	141.00
141.00	141.00
141.00	155.00
169.00	155.00
155.00	169.00
183.00	169.00
183.00	183.00
169.00	155.00
141.00	141.00
141.00	141.00
155.00	155.00
169.00	155.00
169.00	183.00
183.00	183.00
183.00	183.00
169.00	169.00
155.00	141.00
141.00	141.00
127.00	113.00
99.00	85.00
71.00	57.00
57.00	43.00

4.3 FUNCIONAMIENTO DEL SOFTWARE.

OPTRASOFT está desarrollado en MATLAB y permite la interacción a través de interfaces gráficas de usuario. Consiste en un conjunto de interfaces que guiarán al usuario a través de un conjunto de pasos secuenciales para el procesamiento de una cavidad.

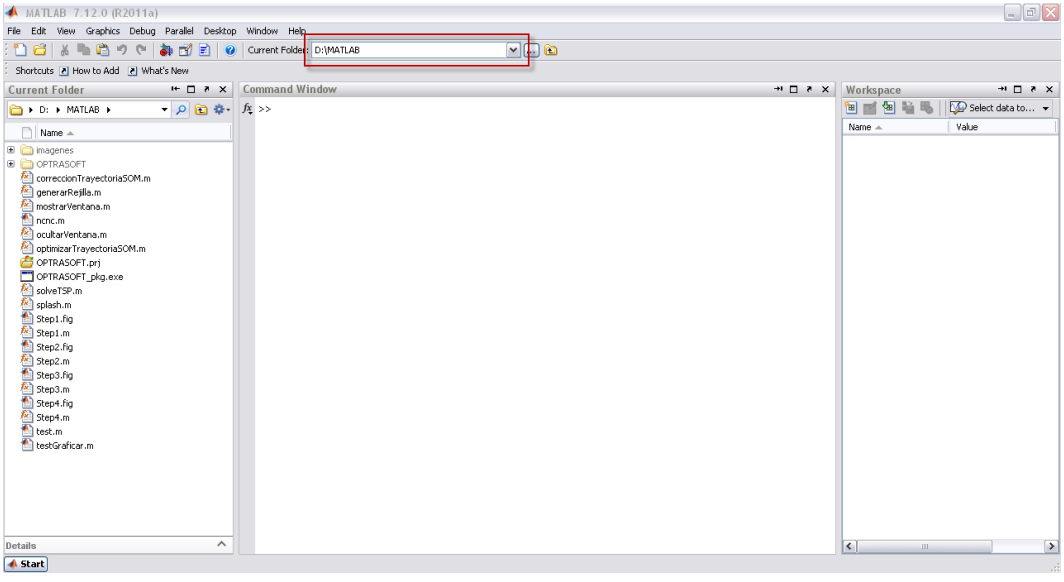
4.3.1 Inicio y ejecución del software. Para ejecutar NEURAL NET OPTRASOFT, es necesario primero abrir el entorno de MATLAB (Ver figura 50):

Figura 50. Plataforma de Matlab



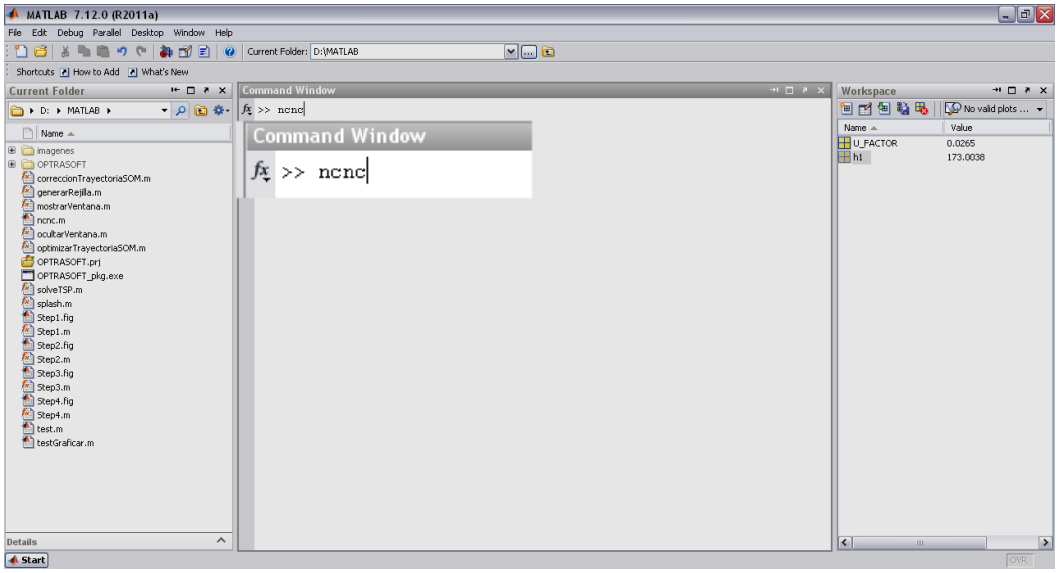
Luego posicionarse en el directorio donde se halla instalado NEURAL NET OPTRASOFT (Ver figura 51).

Figura 51. Ubicación del directorio de Neural Net Optrasoft



Y escribir en la línea de comandos el comando ncnc, así MATLAB para iniciar la ejecución del programa (Ver figura 52).

Figura 52. Comando de iniciacion del programa



- Código del programa

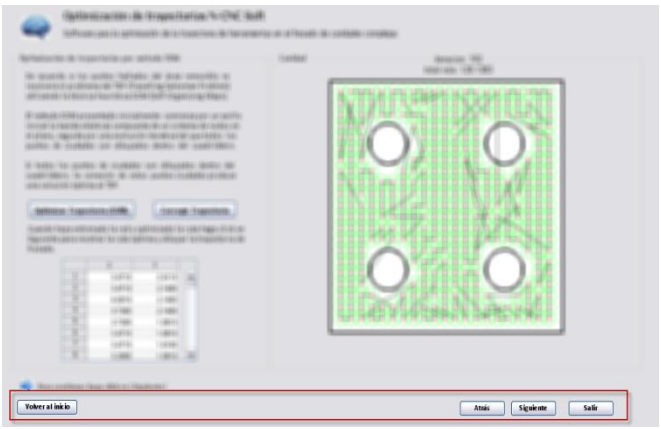
```
clear all;
clearvars;
clc; cla;
close all;
splash('imagenes/splash.bmp', 1800);
h1 = Step1;
```

En seguida aparecerá la pantalla de bienvenida al programa (ver figura 43), y el paso uno en el proceso (Ver figura 44).

En todas las ventanas del programa, en la zona inferior habrá un panel de botones de herramientas (Ver figura 53).

Figura 53. Cinta de opciones: a) botones especiales b) Pantalla de parámetros iniciales

a)



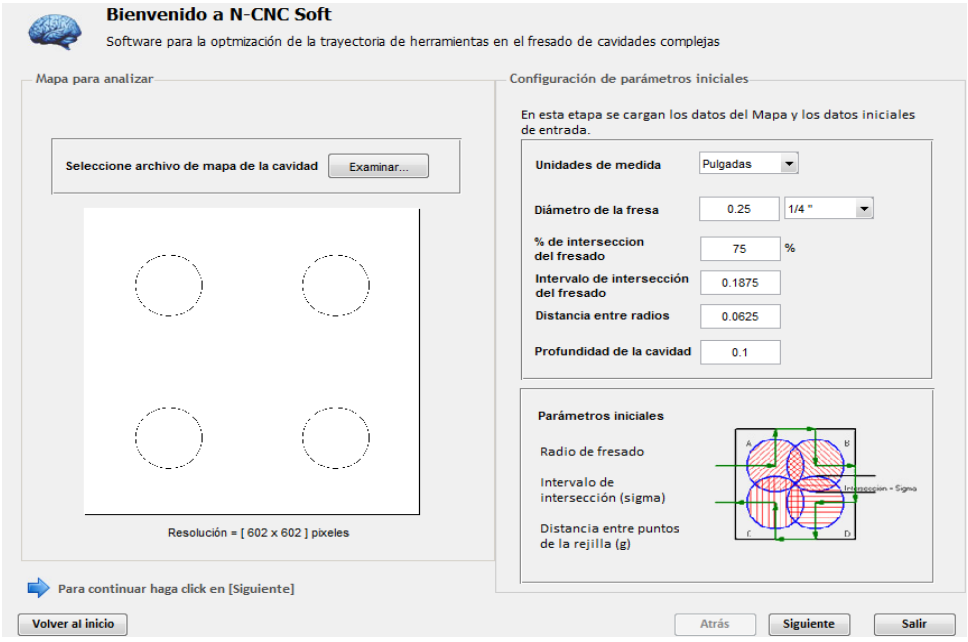
b)

Estas funciones permiten volver a iniciar el programa, Ir al paso anterior, Ir al paso siguiente y Salir del programa, respectivamente (Ver figura 53.a)

4.3.2 Selección de la cavidad a procesar y parámetros de la herramienta.

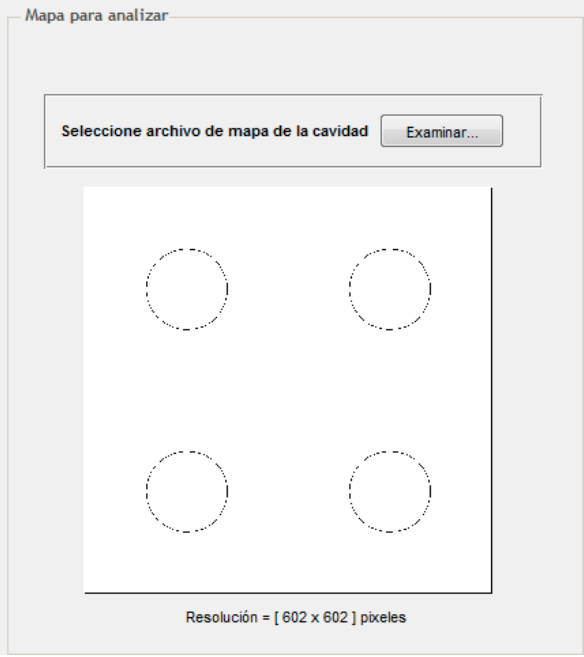
Automáticamente el programa nos lleva a través de una serie de pasos donde se desarrolla el procesamiento de la cavidad (ver figura 54) el cual se divide en dos subprocesos.

Figura 54. Pantalla de procesamiento de la cavidad



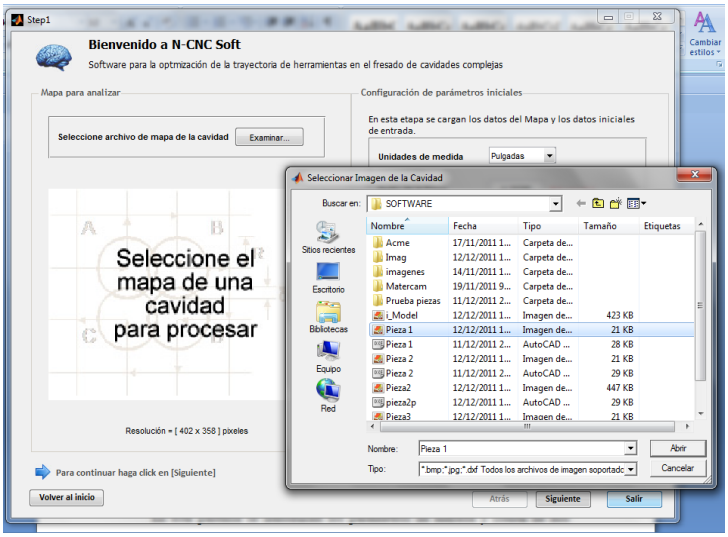
- **Mapa para analizar.** Espacio en la que se cargan el mapa de bits o imagen de la cavidad a simular (Ver figura 55), haciendo clic en el botón Examinar se despliega una pantalla de exploración (Ver Figura 56) donde se selecciona el archivo y se presiona el botón abrir, inmediatamente la imagen es cargada en la zona de visualización de la pantalla mapa de datos, cerrando la pantalla de exploración y visualizando la pantalla de datos de entrada iniciales, y una pantalla individual donde se maneja la imagen completa (Ver figura 57) si no es la imagen indicada se puede cargar otra, realizando el mismo procedimiento, una vez cargada se identifican los valores o atributos de la imagen, información que es utilizada posteriormente por la red como son la longitud máxima en la coordenada *X* y la longitud máxima en la coordenada *Y*.

Figura 55. Pantalla de mapa para analizar



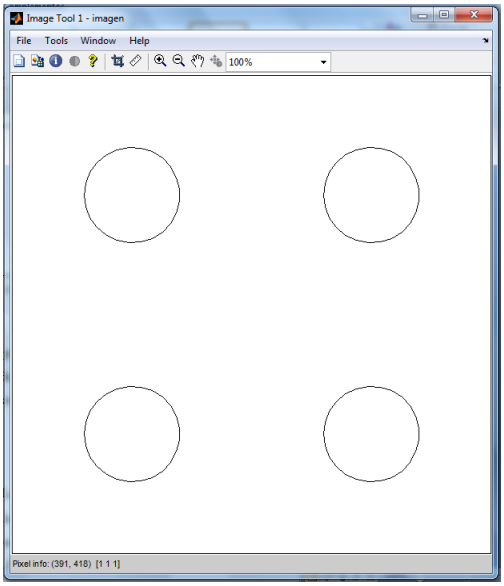
En la interfaz mostrada en la figura 55, permite seleccionar archivo de mapa de la cavidad; el programa permite introducir mapas de cavidad en formato BMP, JPG o DXF (Drawing Exchange Format). Para seleccionar el archivo haga clic en el botón examinar. El resultado es el mostrado en la figura 56.

Figura 56. Pantalla de Exploración Datos Mapa



El software permite calcular y verificar las proporciones de la imagen en las unidades píxeles.

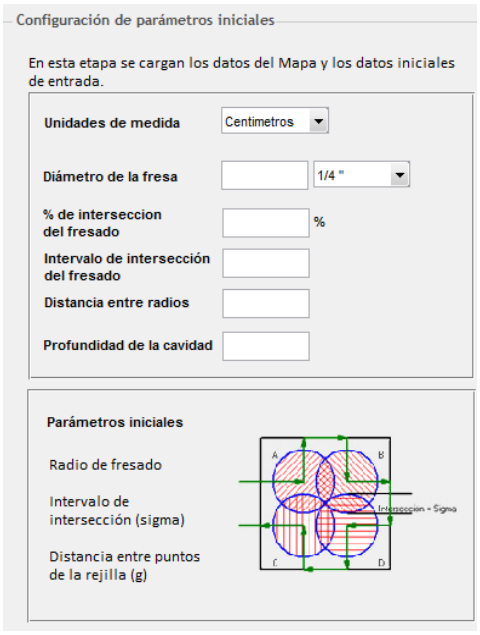
Figura 57. Pantalla de la Imagen



- **Datos De Entrada Iniciales.** La pantalla de configuración de parámetros iniciales (Ver figura 58) permite elegir las unidades a utilizar en la simulación por la red (Ver Figura 59), aunque internamente se procesan los datos en píxeles; las

unidades posibles son Pulgadas (in), Centímetros (Cm) y Píxeles, una vez seleccionadas se capturan los datos en los cajones de texto del radio de la fresa y porcentaje de intersección calculando el intervalo de intersección de fresado (Sigma), y la distancia entre centros de la rejilla (g), para conocer el recorrido total. Los elementos de este formulario son:

Figura 58. Pantalla de Configuración de parámetros iniciales

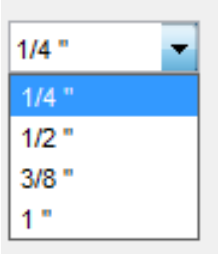


- **Unidades de medida:** Se escogen las unidades de medida para la simulación. (Ver figura 59).

Figura 59. Opciones de unidades

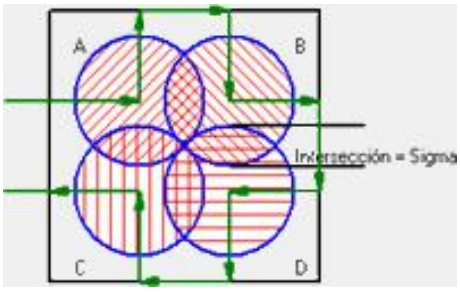
- **Diámetro de la fresa:** Escoge el Diámetro de la herramienta en la cinta de opciones, estandarizada con las brocas comúnmente usadas en la industria: ¼, ½, 3/8, y 1 pulgada (Ver figura 60).

Figura 60. Opciones de Diametro de la fresa



- **Porcentaje de intersección del fresado:** Es el porcentaje de superposición entre dos radios de fresa, y da el parámetro para calcular el intervalo y distancia entre centros.
- **Intervalo de intersección del fresado:** Es la distancia de superposición entre dos radios de fresa, debe ser menor que el doble del radio de la fresa.
- **Distancia entre centros:** Es la distancia a la que se asignarán los puntos en la rejilla de maquinado que se detallará en pasos posteriores, definido por el porcentaje de intersección (Ver figura 61).

Figura 61. Parametros de la fresa

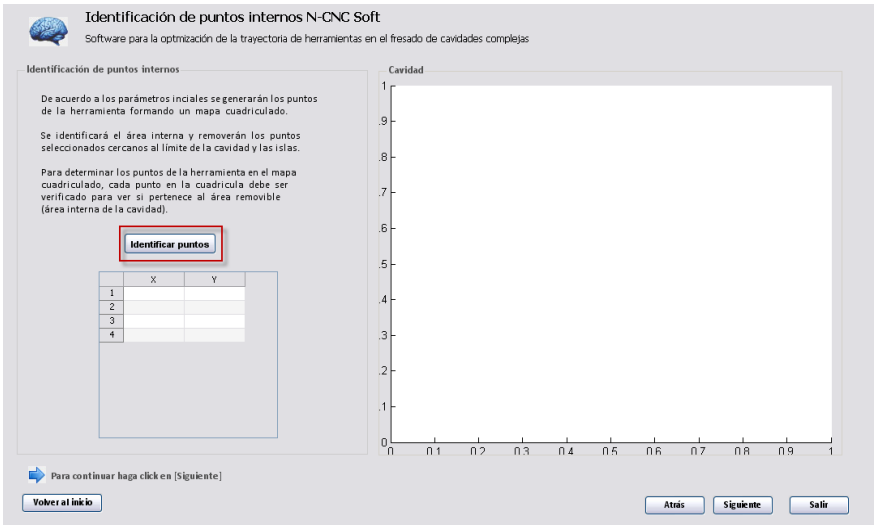


- **Profundidad de la cavidad:** Distancia en profundidad a la que se hará el fresado de la pieza.

La secuencia continua haciendo clic en el botón Siguiente.

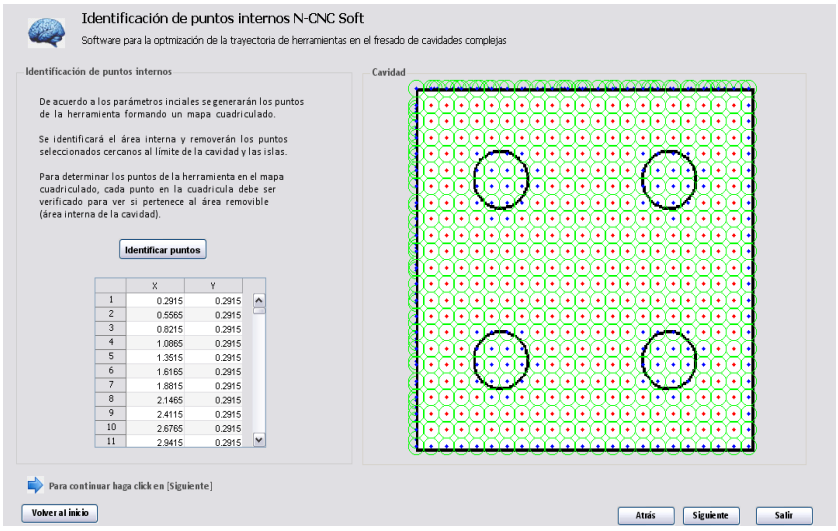
4.3.3 Creación de la rejilla uniforme de puntos. Una vez cargados los datos iniciales y la imagen, se procede a calcular los puntos de rejilla de la cavidad, haciendo clic en el botón “Identificar puntos”. (Ver figura 62).

Figura 62. Pantalla de identificacion de puntos internos



Seguidamente aparece n en la imagen, los puntos de rejilla en la zona grafica de la pantalla Principal, y también la tabla de resultados en la que aparecen las coordenadas de los puntos iniciales de la red, los cuales se ubican en el centroide de la imagen (Ver Figura 63).

Figura 63. Regilla y coodenadas de puntos

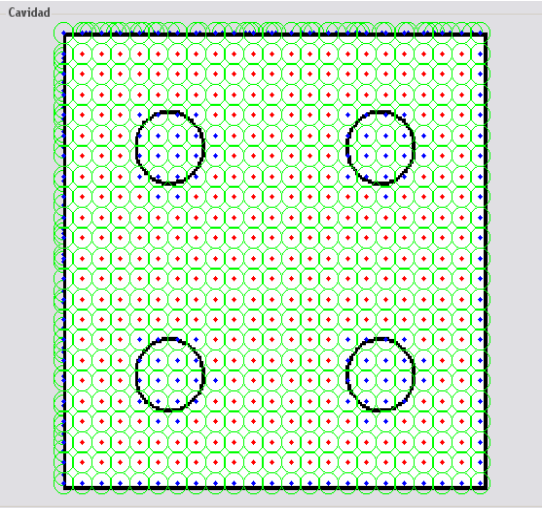


El software generará una rejilla uniforme de puntos y comprobará para cada punto de la rejilla si pertenece a la cavidad o no y presentará el resultado en una imagen independiente (Ver figura 64).

Estos resultados se interpretan de acuerdo a las siguientes convenciones:

- Punto que pertenece al área removible en rojo (maquinable)
- Punto que pertenece al área no removible en azul (interior de las islas, bordes de la cavidad, o área externa a la cavidad).

Figura 64. Rejilla de puntos



El programa también muestra las coordenadas de la rejilla de los puntos admisibles para el procesamiento de la cavidad, en las unidades seleccionadas en los parámetros de entrada del programa (Ver figura 65).

Figura 65. Coordenadas de puntos

	X	Y
1	0.2915	0.2915
2	0.5565	0.2915
3	0.8215	0.2915
4	1.0865	0.2915
5	1.3515	0.2915
6	1.6165	0.2915
7	1.8815	0.2915
8	2.1465	0.2915
9	2.4115	0.2915
10	2.6765	0.2915
11	2.9415	0.2915

4.3.4 Optimización y corrección de la trayectoria. Al iniciar este paso hay una ventana con el mapa de la cavidad y los puntos admisibles para el maquinado (Ver figura 66).

Para trazar una trayectoria óptima el programa debe resolver el problema del TSP (travelling salesman problem), a través del método de mapas autoorganizables conformando un anillo o banda elástica que a través de una red neuronal encuentra la trayectoria óptima para ese conjunto de puntos. La trayectoria óptima es la distancia mínima en la que la herramienta pasa por todos los puntos sin devolverse (Ver figura 67).

Una vez calculados y ubicados los datos iniciales, se inicia el proceso de aprendizaje off-line (no supervisado), haciendo clic en el botón “optimizar trayectorias”, se inicia el proceso de funcionamiento de la red descrito anteriormente (Ver figura 68).

Figura 66. Ventana con el mapa de la cavidad y puntos admisibles para el maquinado

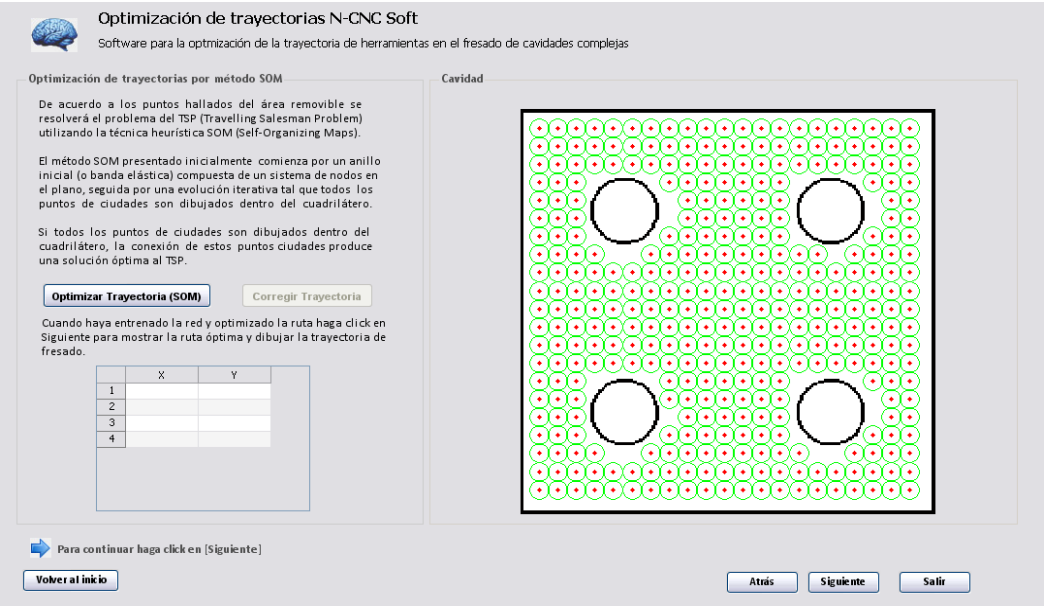
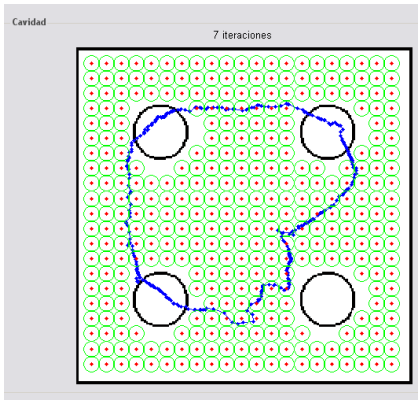
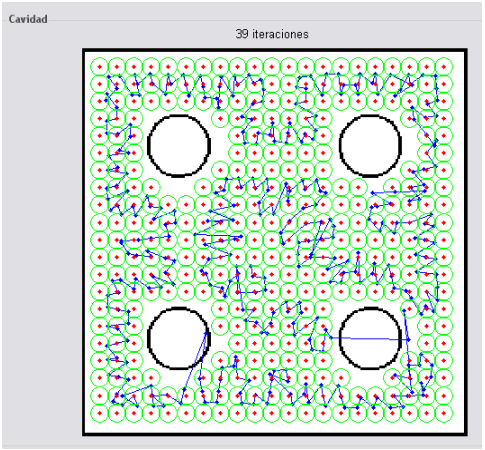


Figura 67. Optimización de la trayectoria

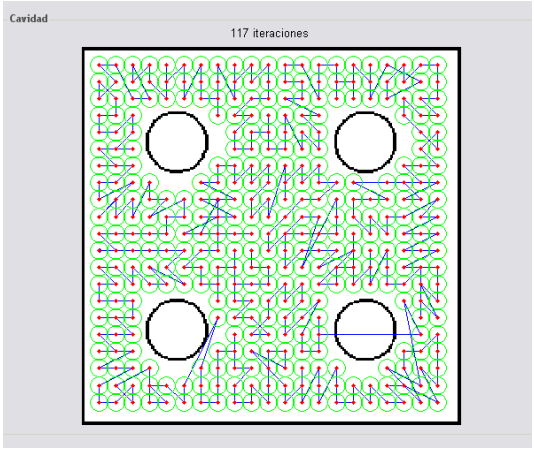
a) Anillo Inicial



b) Iteraciones de la trayectoria



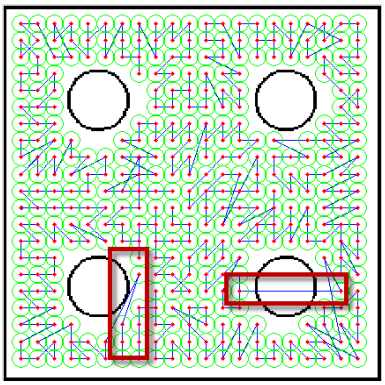
c) Anillo final



El software determina el número de iteraciones que debe realizar y se detiene automáticamente al completar el proceso, al terminar se activará automáticamente “Corregir Trayectoria”.

Finalizado el proceso de Optimizar la trayectoria es necesario corregir ciertas rutas que son indeseables en la trayectoria trazada (Ver figura 68).

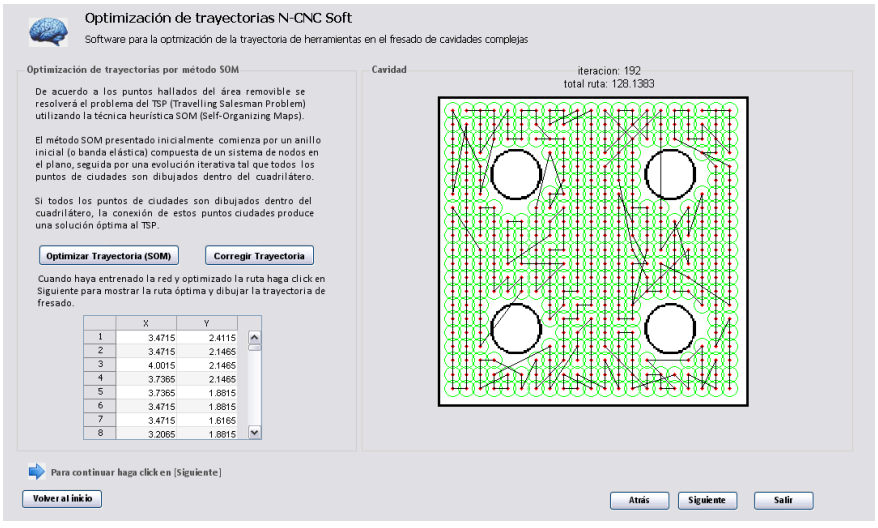
Figura 68. Errores en la trayectoria



Fuente: autores.

Para corregir estas trayectorias se hace clic en el botón “Corregir Trayectoria”; el software iterará a través de todas las rutas y corregirá las indeseables. (Ver figura 69).

Figura 69. Correccion de la trayectoria



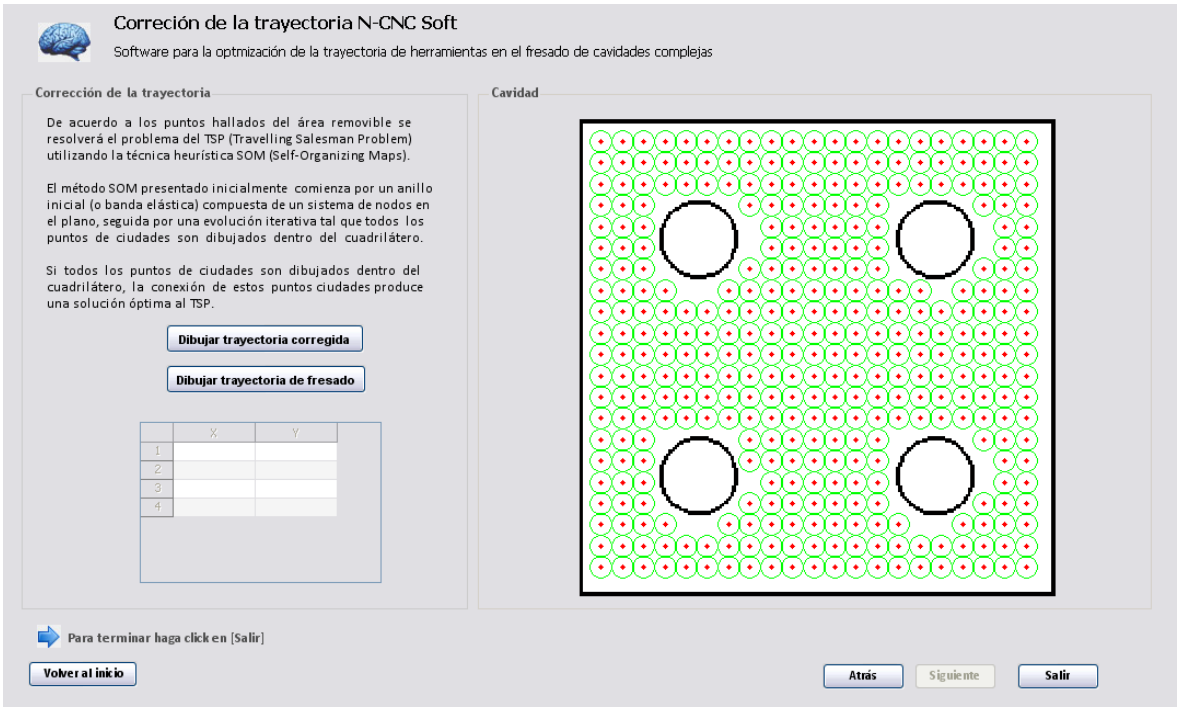
Al finalizar, se actualizará la tabla de coordenadas de los puntos de la rejilla en el orden en que deben ser visitados para que la distancia sea óptima (Ver figura 70).

Figura 70. Coordenadas corregidas

	X	Y
1	1.6165	0.5565
2	1.3515	0.5565
3	1.0865	0.5565
4	0.8215	0.5565
5	0.5565	0.5565
6	0.5565	0.8215
7	0.8215	0.8215
8	0.8215	1.0865

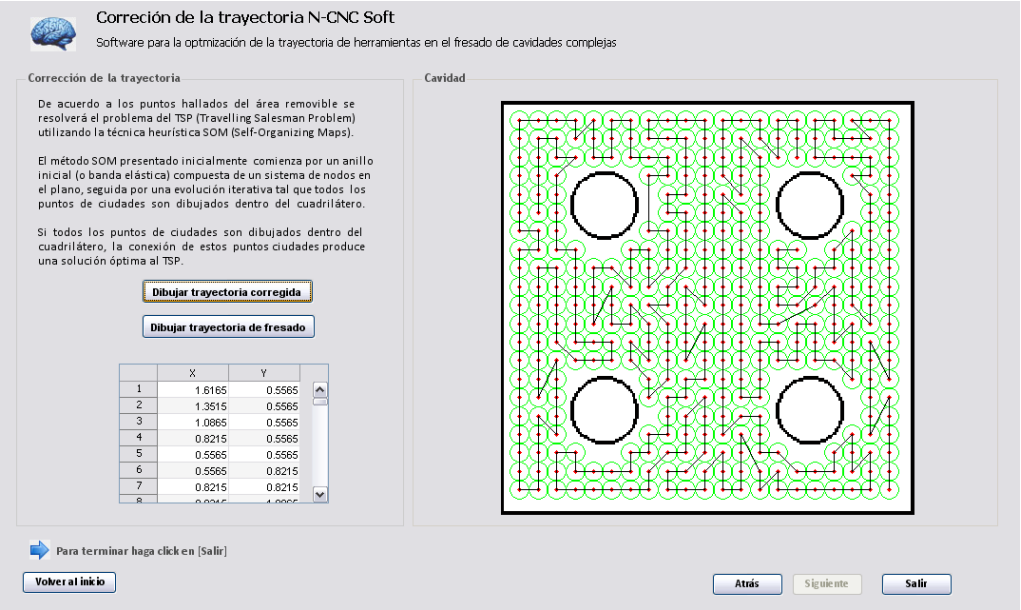
4.2.5 Pasos finales. Corrección de la trayectoria (Ver figura 71).

Figura 71. Pantalla de resultados



Haga clic en el botón “Dibujar trayectoria corregida” para volver a ver la trayectoria corregida (Ver figura 72).

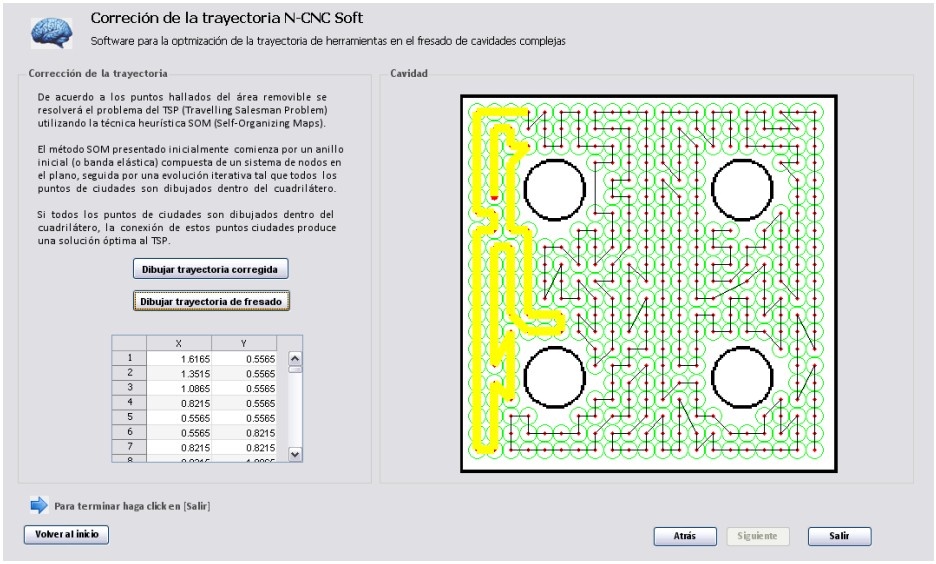
Figura 72. Dibujar trayectoria corregida



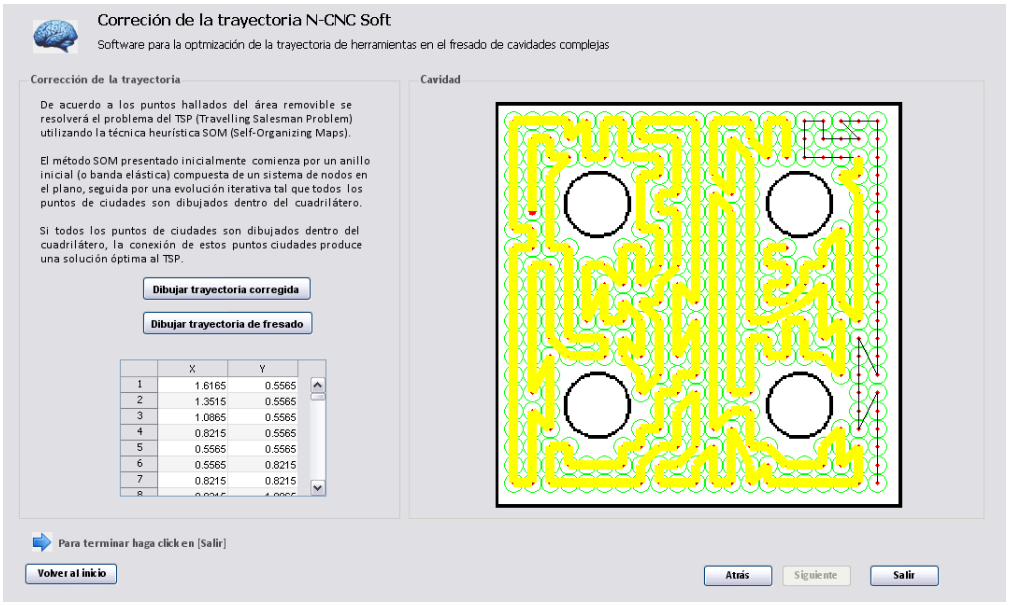
A continuación se hace clic en el botón “Dibujar trayectoria de fresado” donde se puede observar el proceso de fresado cuya finalidad es mostrar la zona desbastada. Hasta este paso concluye el proceso de obtención de la trayectoria (Ver Figura 73).

Figura 73. Trayectoria de la fresa:

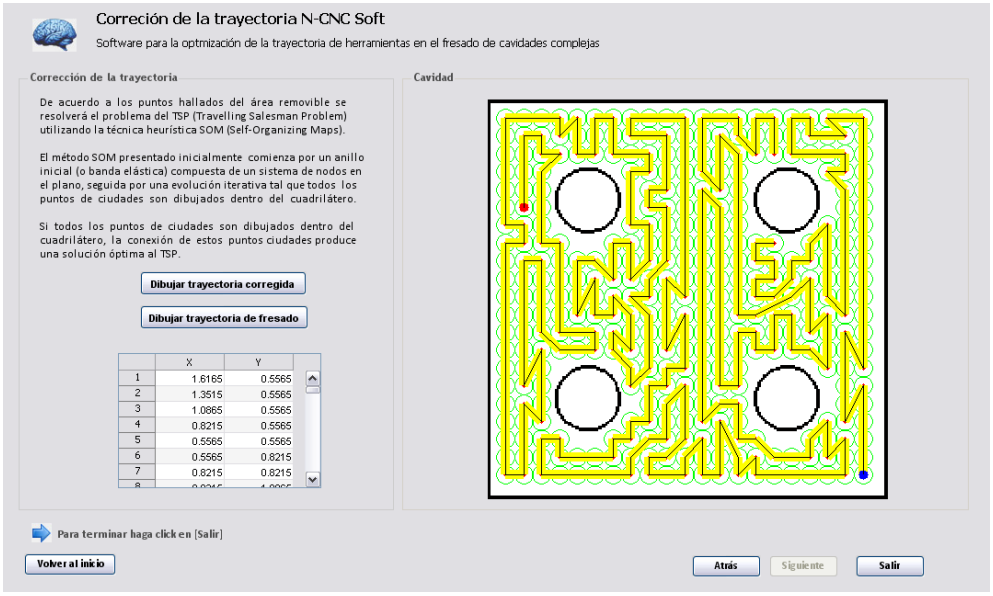
a) Fase Inicial



b) Fase intermedia



c) Fase final



Finalmente el software mostrará utilizando el parámetro de profundidad de la cavidad, un modelo de la pieza y la trayectoria de fresado (Ver figura 74) y las coordenadas de los puntos.

Figura 74. Trayectoria en 3 dimensiones

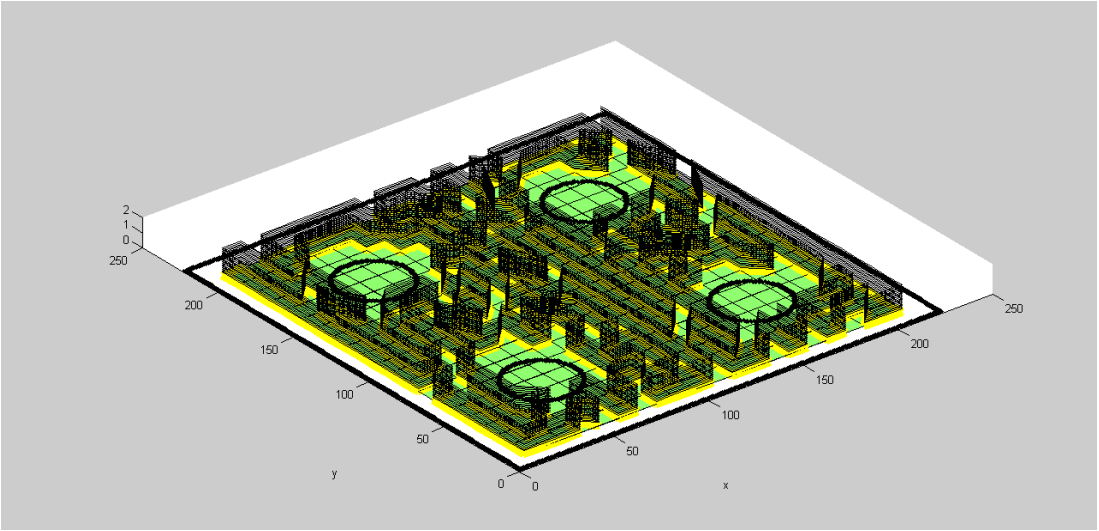


Figura 75. Coordenadas de los puntos maquinables

183.00	197.00
197.00	183.00
197.00	183.00
169.00	155.00
141.00	141.00
155.00	169.00
169.00	155.00
141.00	127.00
127.00	127.00
127.00	141.00
155.00	169.00
183.00	197.00
183.00	169.00
155.00	141.00
141.00	155.00
141.00	155.00
169.00	169.00
183.00	183.00
183.00	169.00
155.00	141.00
141.00	141.00
141.00	155.00
169.00	155.00
155.00	169.00
183.00	169.00
183.00	183.00
169.00	155.00
141.00	141.00
141.00	141.00
155.00	155.00
169.00	155.00
169.00	183.00
183.00	183.00
183.00	183.00
169.00	169.00
155.00	141.00
141.00	141.00
127.00	113.00
99.00	85.00
71.00	57.00
57.00	43.00

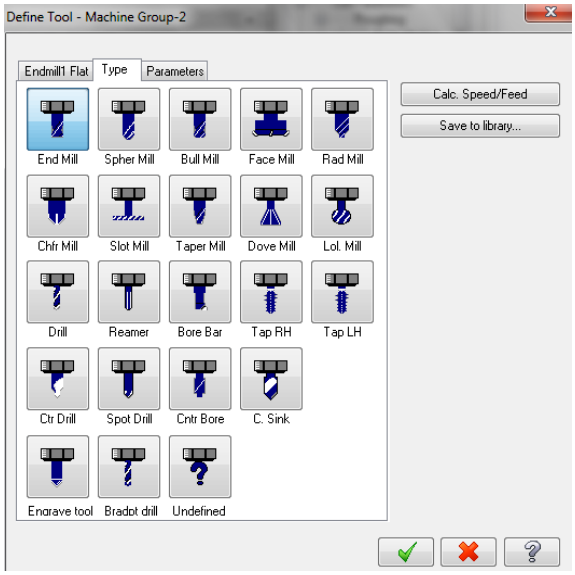
5. TRAYECTORIAS DESCRITAS POR MASTERCAM Y NEURALNET OPTRASOFT.

Se compraró la trayectoria de la herramienta generada por el software con la tecnología existente en este caso MASTERCAM; se tomara como parámetro de comparación la longitud utilizada por MASTERCAM para simular su trayectoria y la longitud utilizada en la trayectoria generada por el software NEURALNET OPTRASOFT bajo las mismas condiciones de corte.

5.1 TRAYECTORIAS GENERADAS POR MASTERCAM.

El procedimiento utilizado por MASTERCAM inicia con la selección de la herramienta utilizada en el mecanizado y los parámetros de corte (Ver figura 76).

Figura 76. Selección de la herramienta



A continuación se elige el tipo de cavidad, distancia de retracción de la herramienta, profundidad y dirección de maquinado (ver figura 77 y 78).

Figura 77. Retracción, profundidad y avance de maquinado

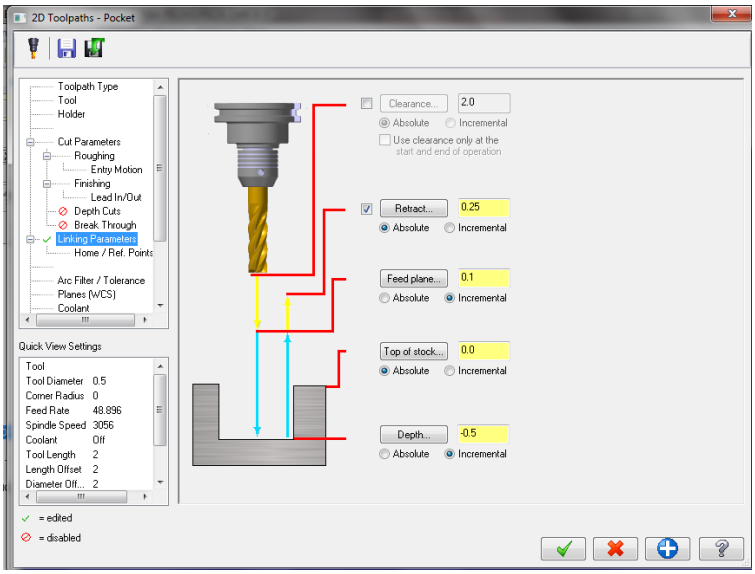
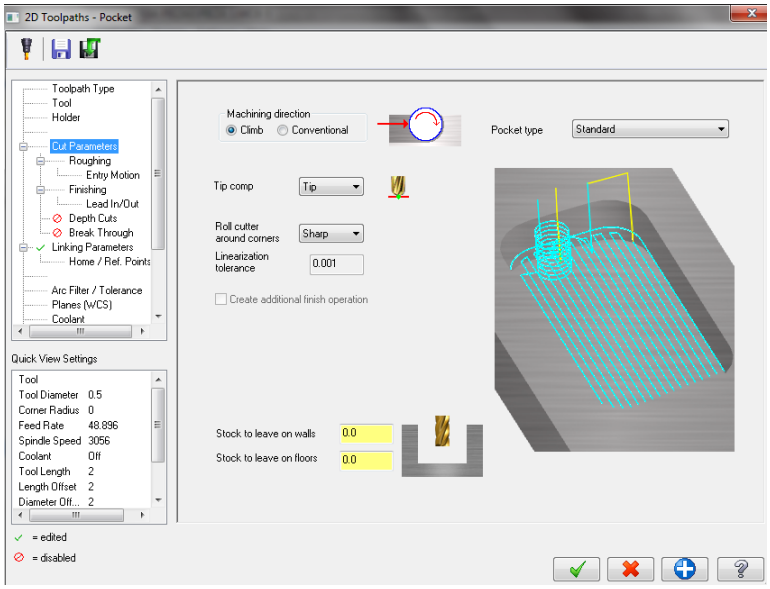
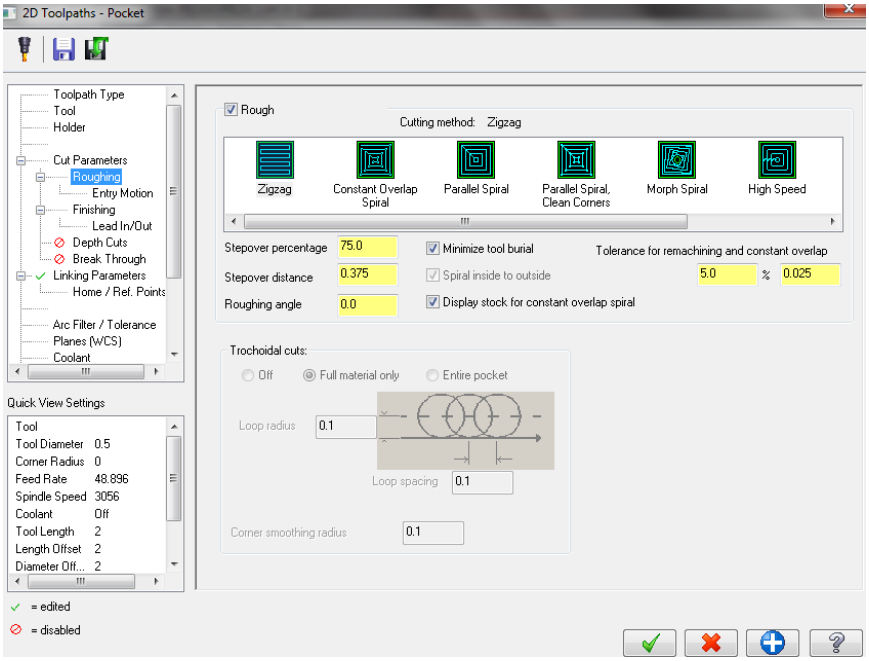


Figura 78. Dirección de maquinado de la herramienta



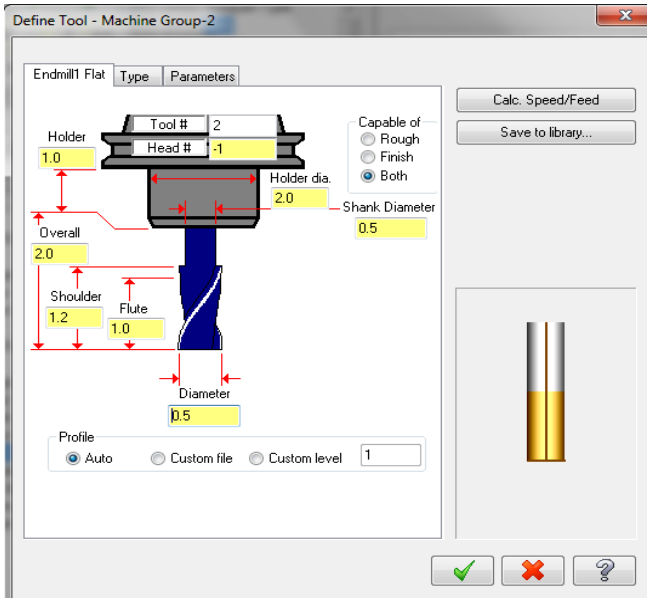
MASTERCAM utiliza 8 trayectorias predeterminadas para realizar el maquinado que son: zig-zag, constant overlap spiral, parallel spiral, parallel spiral clean corner, morph spiral, high speed, one way y trae spiral. Se selecciona en la ventana la trayectoria con la cual se desea trabajar (ver figura 79).

Figura 79. Selección de trayectoria



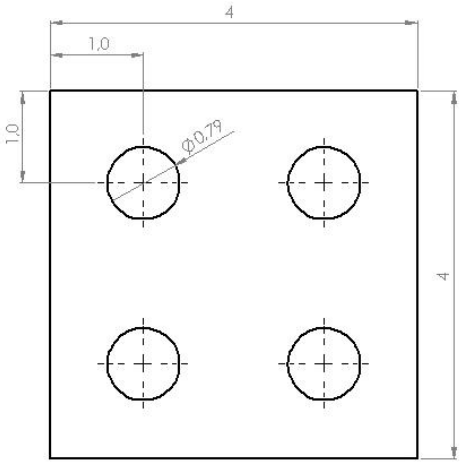
En la siguiente figura se muestran las dimensiones de la herramienta seleccionada en la librería de Mastercam.

Figura 80. Dimensiones de la herramienta



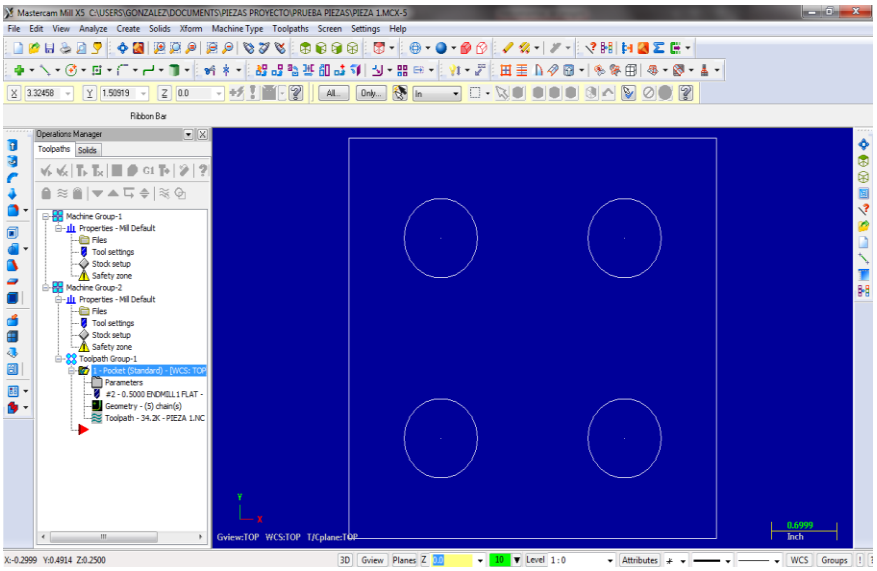
5.1.1 Descripción del procedimiento utilizado por Mastercam para obtener la longitud recorrida por la herramienta para la cavidad 1. Una vez seleccionada la herramienta, parámetros de corte y tipo de trayectoria se describen las dimensiones de la cavidad en el editor de dibujo de Mastercam (ver figura 81 y 82).

Figura 81. Plano de la cavidad 1 (dimensiones en pulgadas)



Fuente: autores

Figura 82. Geometría de la cavidad 1 (diámetro de la fresa 0.5 in)



A continuación se muestra una de las trayectorias utilizadas por Mastercam llamada Zig-Zag, la longitud total recorrida (ver figuras 83 y 84).

Figura 83. Trayectoria Zig-Zag descrita por Mastercam

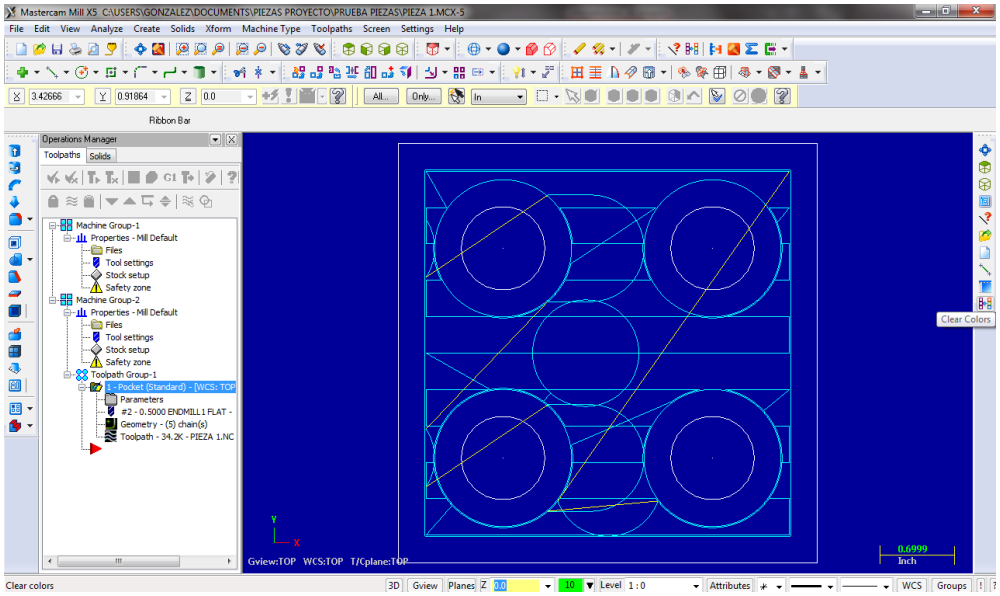
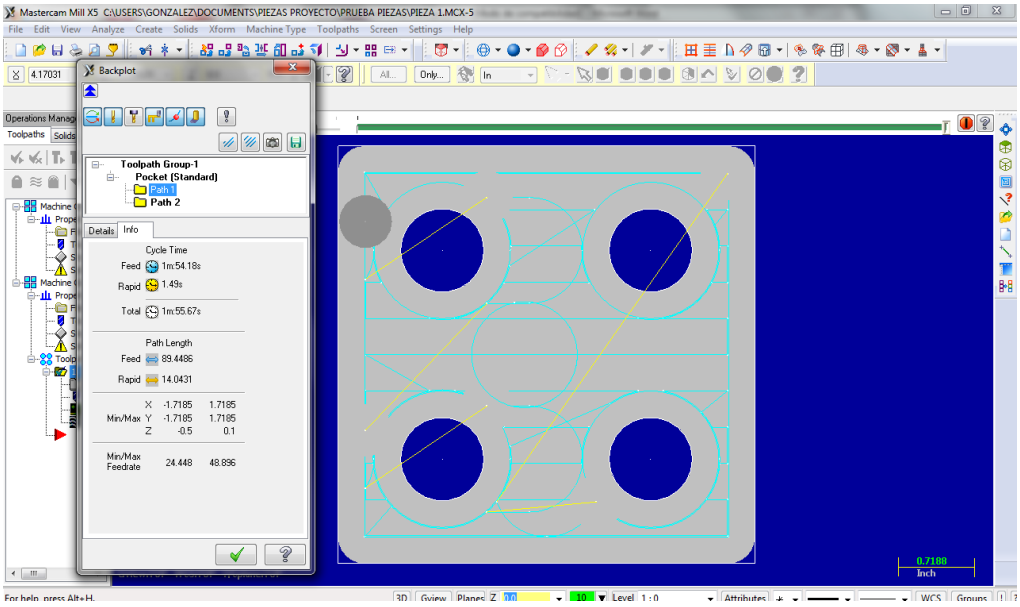


Figura 84. Tiempo de simulación estimado por Mastercam



El proceso para la obtención de la longitud de la trayectoria es el mismo para todas las cavidades con las cuales se hizo la comparación.

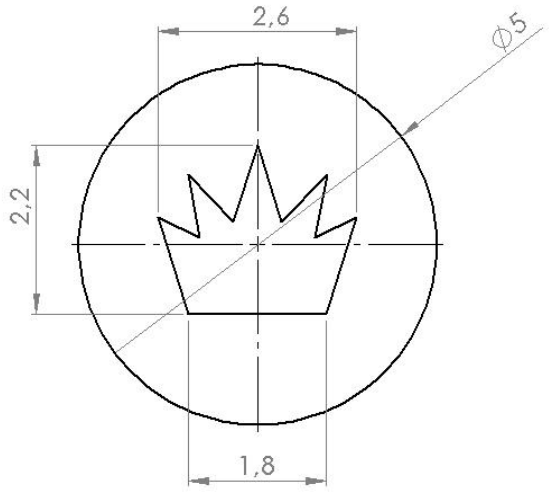
Tabla 2. Trayectorias de la cavidad 1 en Mastercam

Cavidad	*P**	*Diametro de la fresa*	*Zigzag*	*Constant overlap spiral*	*Parallel spiral*	*Parallel spiral cleancorner*	*Morph spiral*	*High speed*	*One Way*	*True spiral*
1	40	3/8	138,80	222,76	159,76	171,81	286,27	375,52	278,68	216,16
		1/2	116,25	157,77	137,74	148,46	230,50	239,65	220,60	176,33
	50	3/8	126,25	208,49	131,80	139,03	246,46	383,45	236,64	187,45
		1/2	108,22	140,48	132,05	139,50	200,62	274,60	184,83	154,90
	60	3/8	120,18	195,69	143,53	152,00	220,62	309,37	217,80	169,29
		1/2	105,76	139,53	107,20	111,19	180,95	215,15	168,99	142,01
	70	3/8	112,81	206,33	136,81	142,70	201,91	307,94	198,36	150,87
		1/2	103,49	138,23	102,94	108,78	166,07	236,57	161,47	129,14
	80	3/8	109,39	211,63	132,68	137,80	182,40	385,02	177,39	146,63
		1/2	97,70	137,03	101,89	106,36	156,65	231,37	146,02	119,31
	90	3/8	106,65	183,26	128,36	132,19	172,46	338,00	170,57	136,23
		1/2	96,58	136,51	99,82	103,09	146,98	252,11	134,47	121,50

*P=porcentaje de sobreposición (Stepover)

5.1.2 Descripción del procedimiento utilizado por Mastercam para obtener la longitud recorrida por la herramienta para la cavidad 2.

Figura 85. Plano de la cavidad 2 (dimensiones en pulgadas)



Fuente: autores.

Figura 86. Geometría de la cavidad 2 (diámetro de la fresa 0.5 in)

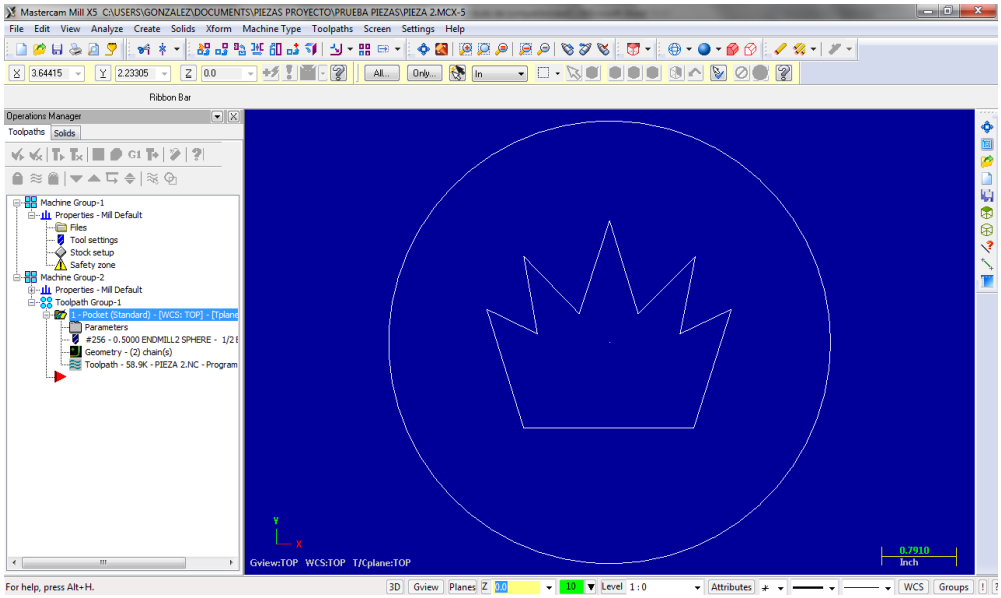


Figura 87. Trayectoria Morph Spiral descrita por Mastercam

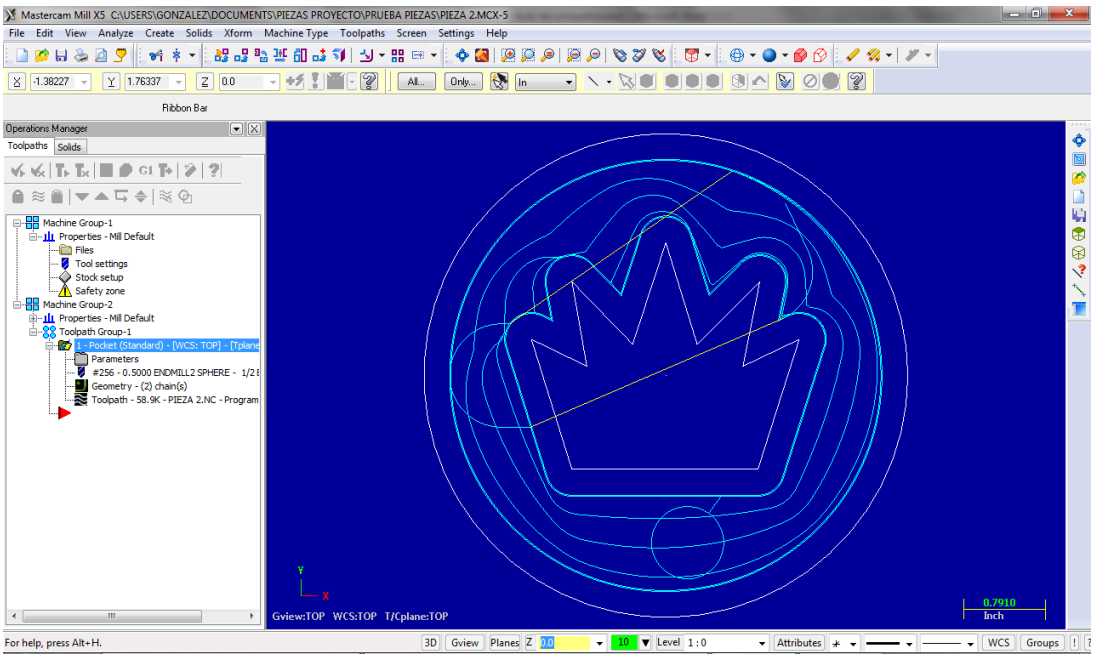


Figura 88. Tiempo de simulación estimado por Mastercam.

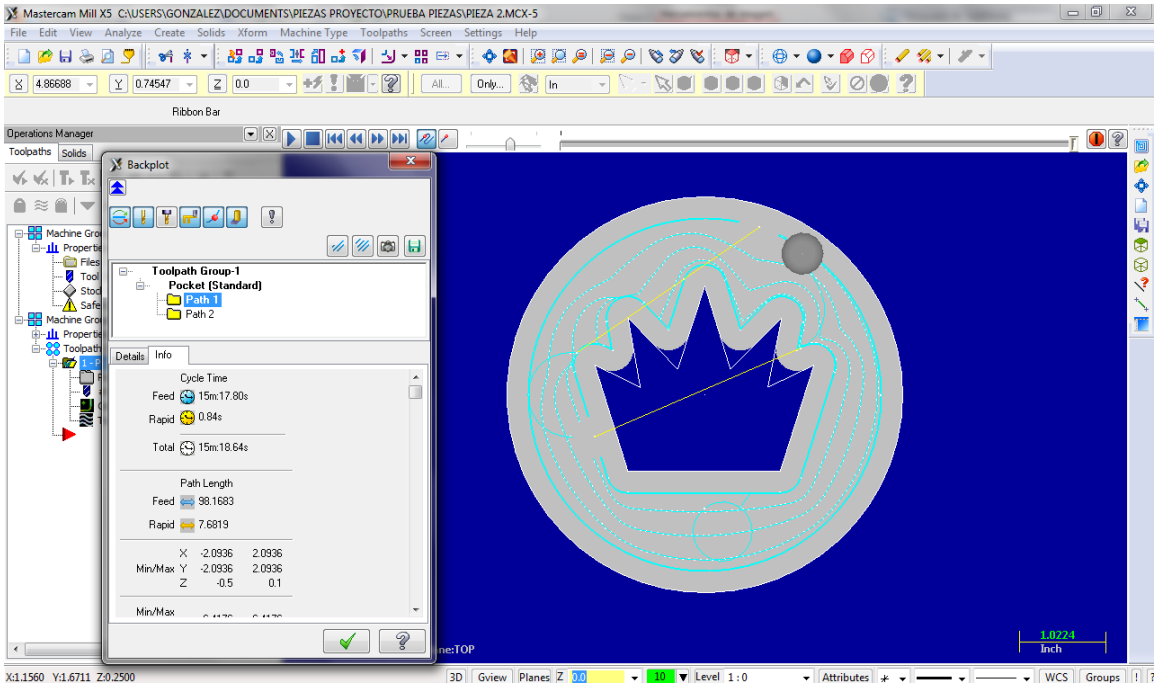


Tabla 3. Trayectorias de la cavidad 2 en Mastercam

Cavidad	*P**	*Diametro de la fresa*	*Zigzag*	*Constant overlap spiral*	*Parallel spiral*	*Parallel spiral cleancorner*	*Morph spiral*	*High speed*	*One Way*	*True spiral*
2	40	1/4	216,30	344,56	239,75	251,48	367,98	544,51	359,83	280,01
		3/8	136,09	173,51	135,92	140,21	174,98	360,69	268,28	141,28
		1/2	107,54	147,05	150,58	153,47	125,86	282,72	182,70	114,97
		1	54,00	54,55	55,65	56,86	81,64	68,09	81,91	100,35
	50	1/4	166,88	198,87	208,46	215,18	209,87	443,30	339,74	170,40
		3/8	122,86	147,99	169,56	182,51	152,28	338,90	216,93	122,55
		1/2	99,79	111,37	122,13	125,36	124,87	244,48	163,93	104,11
		1	50,68	54,56	56,54	56,54	56,54	68,09	68,72	89,90
	60	1/4	150,05	200,60	193,39	197,81	186,53	389,25	282,49	150,21
		3/8	112,23	152,41	160,76	164,70	140,09	337,67	192,12	107,75
		1/2	95,52	112,26	138,51	142,92	102,93	276,13	145,96	96,00
		1	51,32	54,56	56,53	56,54	56,54	68,09	66,39	87,00

*P=porcentaje de sobreposición (Stepover)

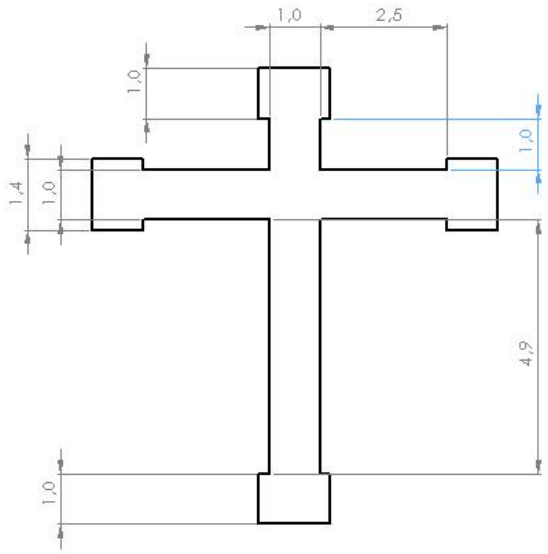
Tabla 3. (Continuación)

Cavidad	*%*	*Diametro de la fresa*	*Zigzag*	*Constant overlap spiral*	*Parallel spiral*	*Parallel spiral cleancorner*	*Morph spiral*	*High speed*	*One Way*	*True spiral*
2	70	1/4	140,73	176,17	180,20	183,08	171,51	389,46	259,05	133,69
		3/8	106,53	127,05	148,93	150,48	128,90	307,05	172,36	104,70
		1/2	89,79	99,63	102,40	102,40	106,39	245,88	127,93	89,51
		1	49,81	54,56	56,54	56,54	56,54	68,09	58,80	75,63
	80	1/4	128,77	176,28	166,82	169,68	158,04	342,65	227,07	128,32
		3/8	101,53	135,85	110,37	113,05	121,18	271,80	159,74	100,41
		1/2	93,12	102,94	100,62	100,62	91,44	231,85	126,64	88,91
		1	47,88	54,56	56,54	56,54	56,54	68,09	54,59	74,71
	90	1/4	123,97	157,06	151,37	153,35	148,78	356,37	205,24	120,87
		3/8	99,08	145,79	142,65	147,00	116,72	255,48	143,70	92,46
		1/2	86,47	103,05	91,05	91,05	99,42	208,15	117,07	84,55
		1	47,58	54,56	54,56	54,56	54,56	68,09	54,59	71,19

*P=porcentaje de sobreposición (Stepover)

5.1.3 Descripción del procedimiento utilizado por Mastercam para obtener la longitud recorrida por la herramienta **para la cavidad 3.**

Figura 89. Plano de la cavidad 3 (Dimensiones en pulgadas)



Fuente: autores.

Figura 90. Geometría de la cavidad 3 (diámetro de la fresa 0.5 in)

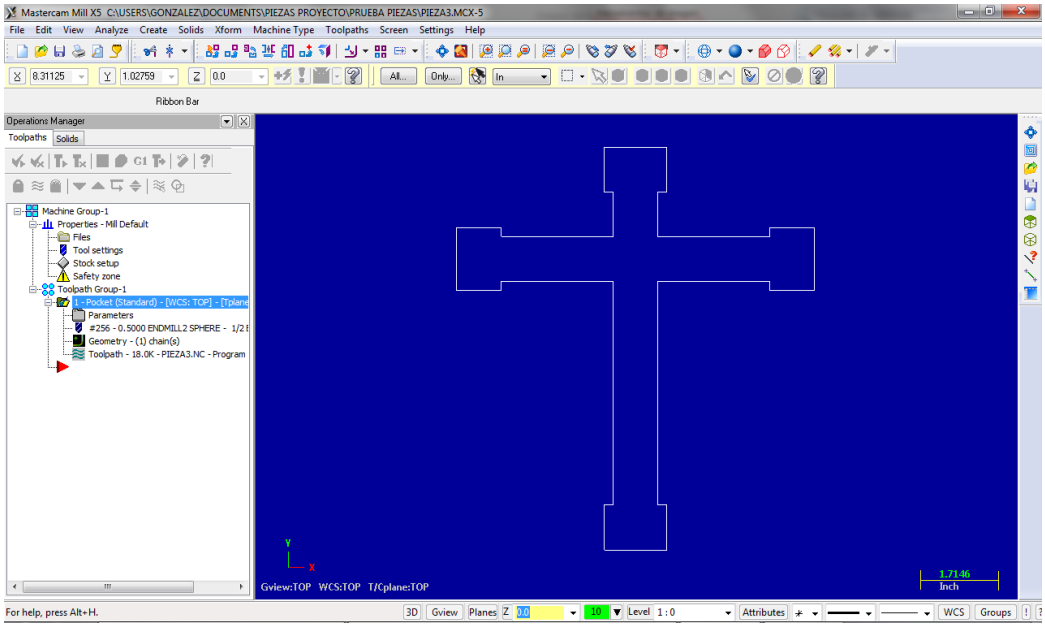


Figura 91. Trayectoria Zig-Zag descrita por Mastercam

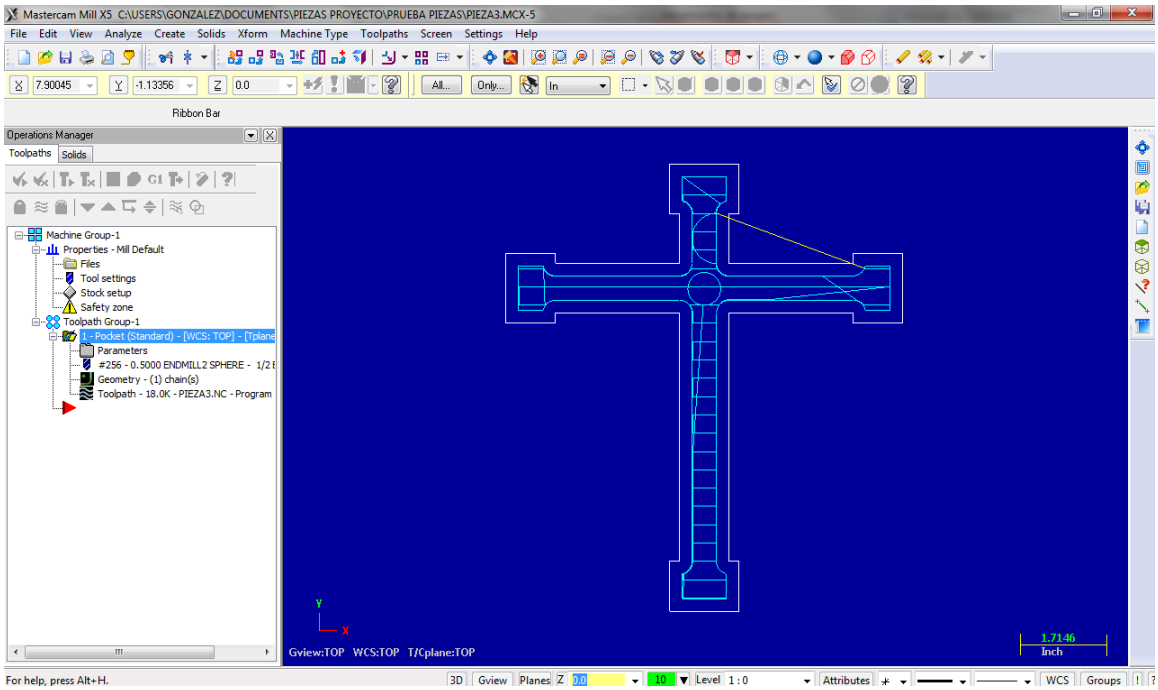


Figura 92. Tiempo de simulación estimado por Mastercam

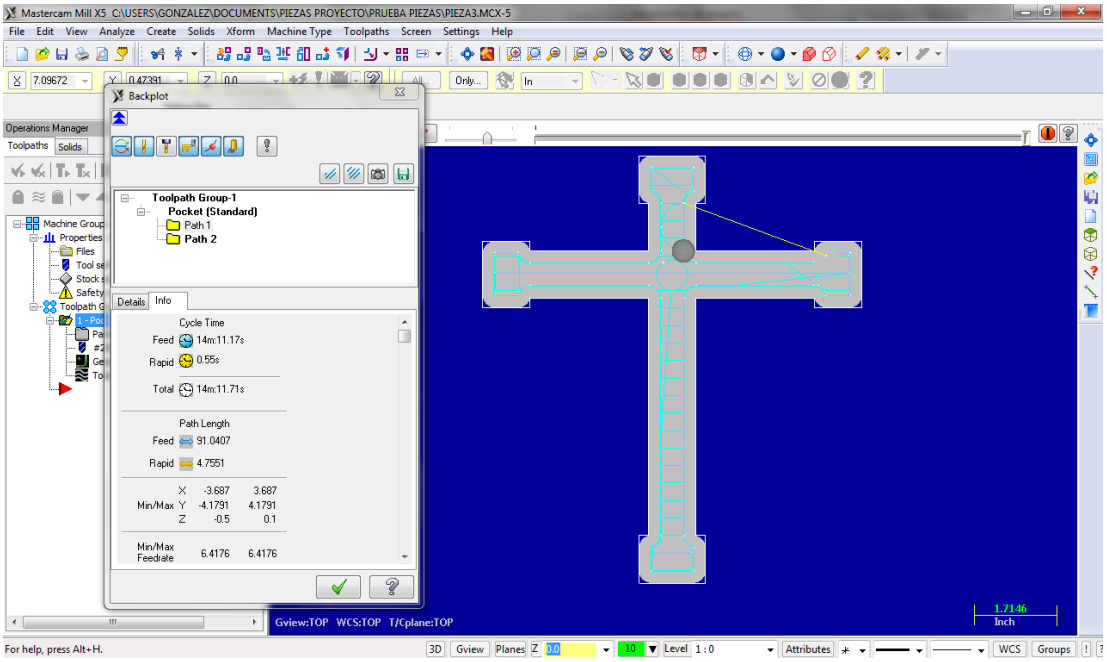


Tabla 4. Trayectorias de la cavidad 3 en Mastercam

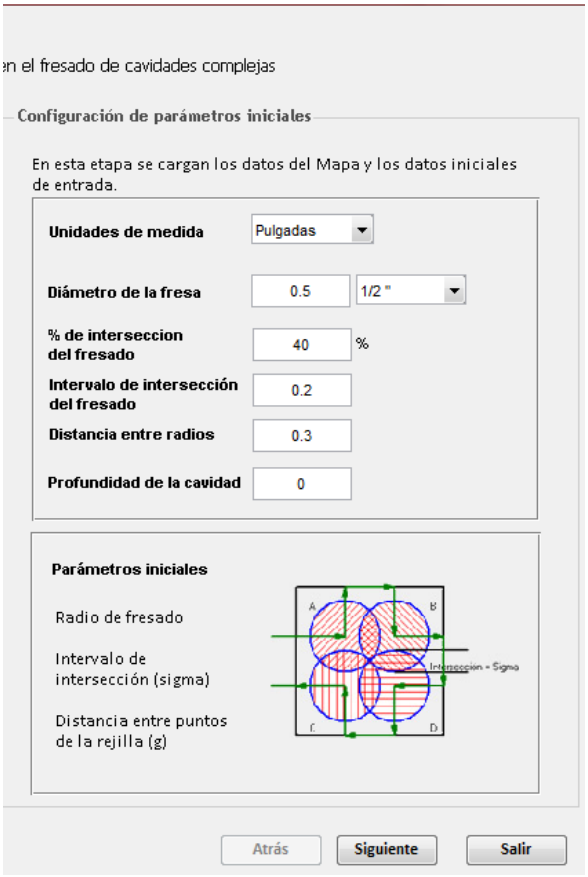
Cavidad	*P**	*Diametro de la fresa*	*Zigzag*	*Constant overlap spiral*	*Parallel spiral*	*Parallel spiral cleancorner*	*Morph spiral*	*High speed*	*One Way*	*True spiral*
3	40	1/4	193,98	209,83	208,72	222,12	1135,92	598,12	446,46	725,61
		3/8	145,49	155,72	151,95	179,28	757,06	456,31	298,44	482,98
	50	1/4	171,73	189,76	178,47	191,69	925,09	580,79	373,19	606,72
		3/8	129,20	143,84	142,96	152,97	621,93	463,56	247,91	410,83
	60	1/4	157,03	175,64	170,73	178,78	785,17	663,05	326,66	515,66
		3/8	124,02	141,65	111,89	115,07	533,27	448,87	224,72	346,50
	70	1/4	142,03	172,15	144,90	148,69	685,72	652,10	282,19	454,79
		3/8	113,56	124,10	140,70	148,95	468,88	414,76	197,77	316,07
	80	1/4	135,96	157,52	141,90	147,95	606,31	702,38	259,95	409,38
		3/8	117,13	123,71	115,51	130,27	418,81	296,77	194,53	289,27
	90	1/4	137,94	156,84	138,02	143,82	552,72	712,49	255,82	376,08
		3/8	101,94	122,66	112,54	118,18	378,64	291,83	162,61	259,58

*P=porcentaje de sobreposición (Stepover)

5.2 TRAYECTORIAS GENERADAS POR NEURALNET OPTRASOFT.

El procedimiento utilizado por NEURAL NET OPTRASOFT inicia con la selección del diámetro de la herramienta utilizada en el mecanizado y los parámetros de corte (Ver figura 93).

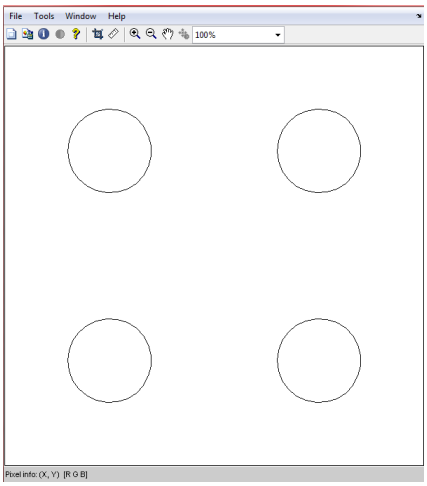
Figura 93. Selección del diámetro de la herramienta



En MASTERCAM se utilizan 8 trayectorias predeterminadas para realizar el maquinado que son: zig-zag, constant overlap spiral, parallel spiral, parallel spiral clean corner, morph spiral, high speed, one way y trae spiral. Parametro que no es necesario en NEURAL NET OPTRASOFT puesto que el software está diseñado para que escoja la trayectoria no dependiendo de la forma, sino de la distancia total recorrida más corta.

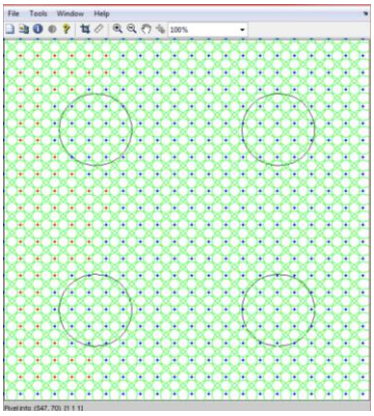
5.2.1 Descripción del procedimiento utilizado por NEURAL NET OPTRASOFT para obtener la longitud recorrida por la herramienta para la cavidad 1. Una vez seleccionada la herramienta y los parámetros de corte, el software calcula y verifica las proporciones de la imagen en píxeles, y la interpreta en una pantalla de imagen de Matlab (Ver figura 94).

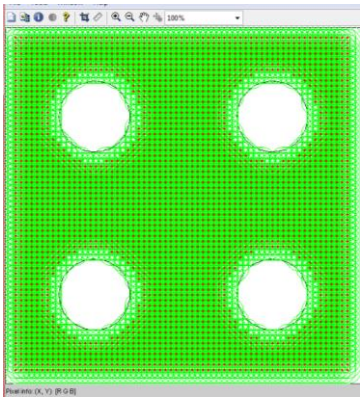
Figura 94. Reconocimiento de la Imagen de la cavidad 1



Con los parámetros iniciales, se crea una rejilla de puntos, evaluando si pertenecen o no al área de la pieza a maquinar (Ver figura 95), los puntos rojos están dentro y permanecen mientras los azules se eliminan.

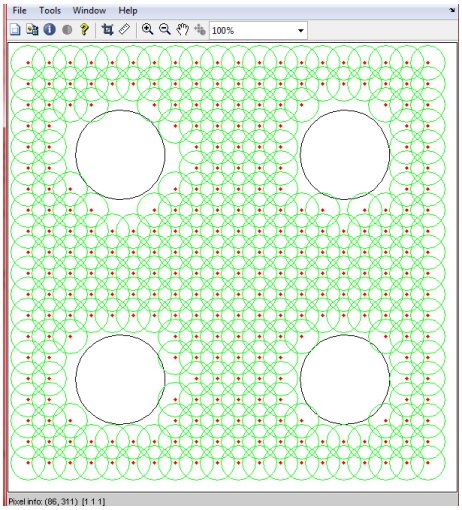
Figura 95. Selección de puntos pertenecientes a la cavidad 1:





Se eliminan los puntos descartados (Ver figura 96).

Figura 96. Puntos maquinables en la cavidad 1



Una vez identificados los puntos, el software itera calculando posibles rutas, extendiendo una banda elástica compuesta por el mismo número de puntos ubicados dentro de la cavidad (Ver figura 97), y busca una trayectoria hasta encontrar la de menor recorrido (Ver figura 98).

Figura 97. Banda elástica en la cavidad 1

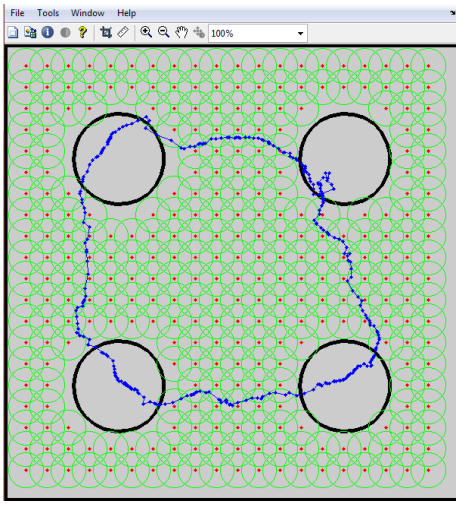
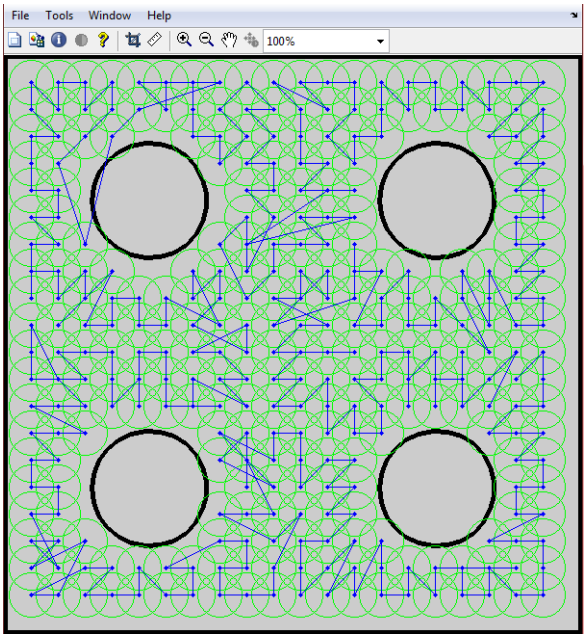


Figura 98. Iteraciones de las trayectorias de la cavidad 1



Finalizado el proceso de Optimizar la trayectoria es necesario corregir ciertas rutas que son indeseables en la trayectoria trazada (Ver figura 99 Y 100).

Figura 99. Corrección de la optimización en la cavidad 1

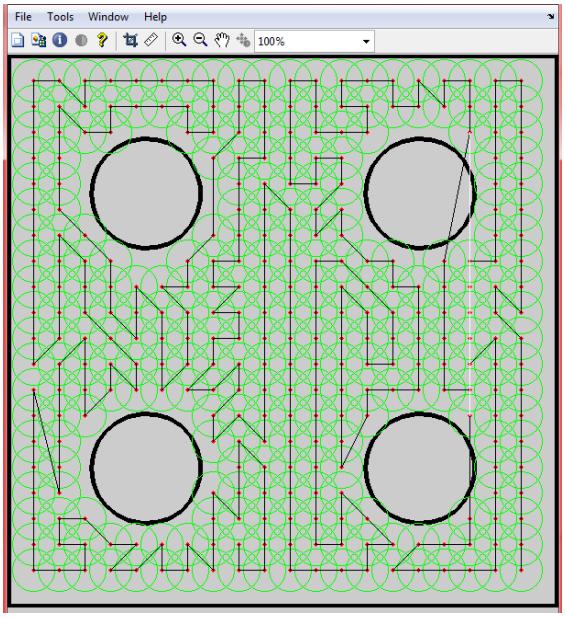
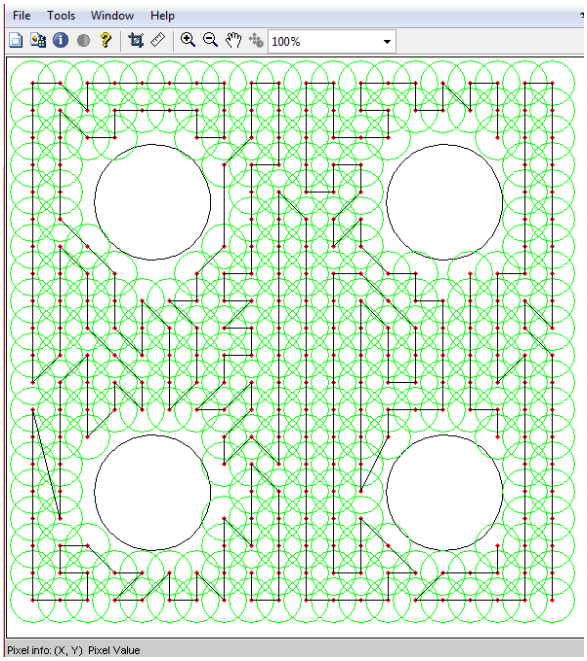


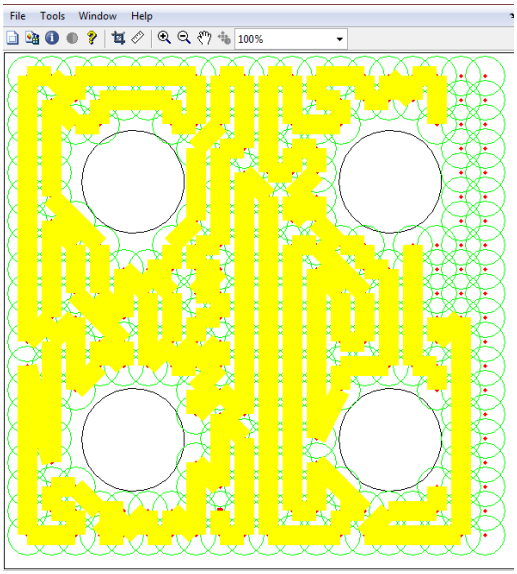
Figura 100. Trayectoria corregida de la cavidad 1



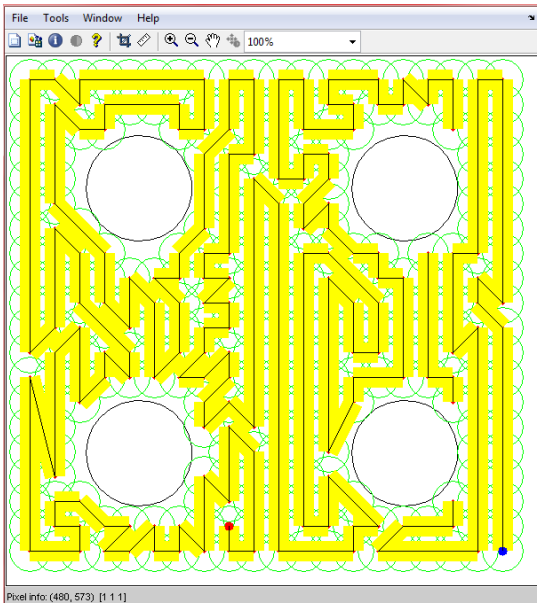
Una vez corregida la trayectoria, se dibuja el recorrido de la herramienta (Ver figura 101)

Figura 101. Recorrido de la herramienta en la cavidad 1

Inicial



Final



Y obtenemos el recorrido en 3 dimensiones (Ver figura 102) y las coordenadas de los puntos maquinables (Ver figura 103).

Figura 102. Trayectoria en 3 dimensiones de la cavidad 1

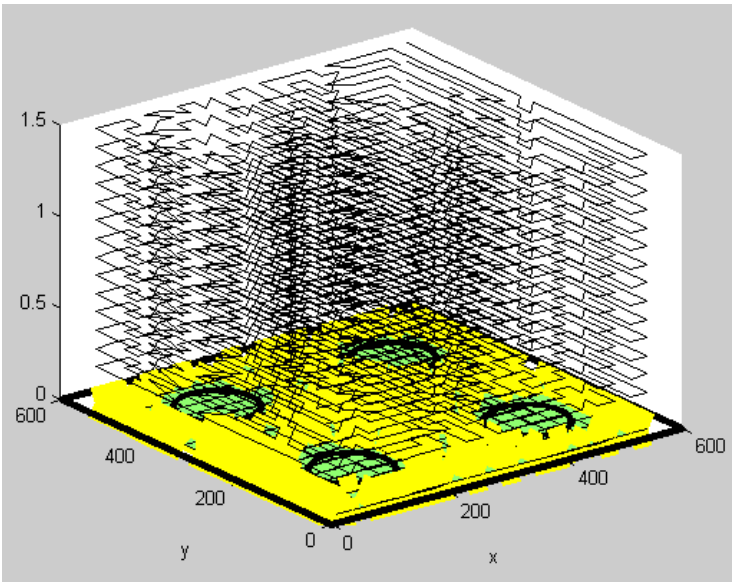


Figura 103. Coordenadas de los puntos maquinables de la cavidad 1

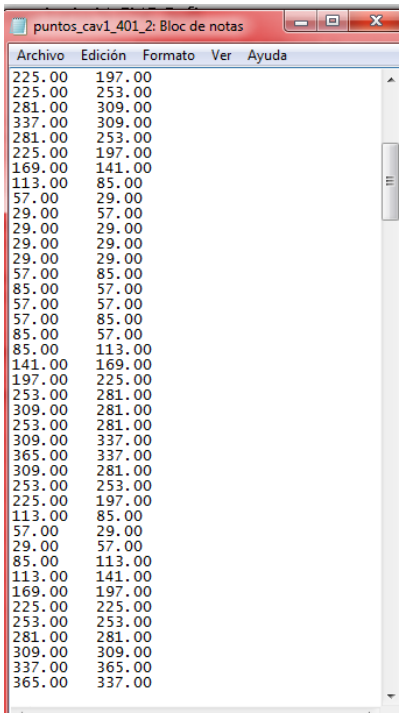


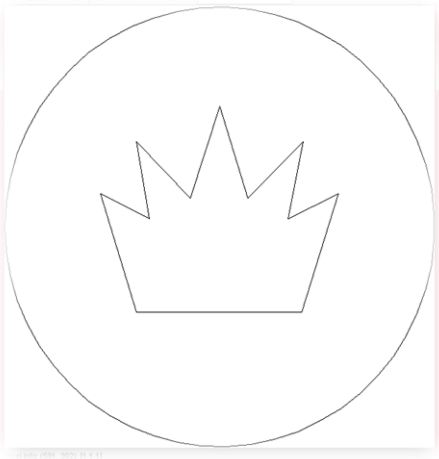
Tabla 5. Longitud de las trayectorias de la cavidad 1 en Neural Net Optrasoft

Cavidad	*%*	*Diametro de la fresa*	*Neural Net Optrasoft*
1	40	3/8	222,60
		1/2	147,80
	50	3/8	177,52
		1/2	129,20
	60	3/8	149,24
		1/2	96,00
	70	3/8	122,56
		1/2	82,63
	80	3/8	112,92
		1/2	76,80
	90	3/8	92,37
		1/2	58,23

5.2.2 Descripción del procedimiento utilizado por NEURAL NET OPTRASOFT para obtener la longitud recorrida por la herramienta para la cavidad 2. Una

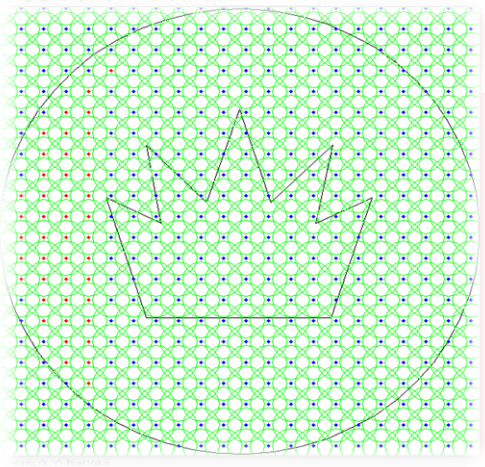
vez seleccionada la herramienta y los parámetros de corte, el software calcula y verifica las proporciones de la imagen en píxeles, y la interpreta en una pantalla de imagen de Matlab (Ver figura 104).

Figura 104 Reconocimiento de la Imagen de la cavidad 2



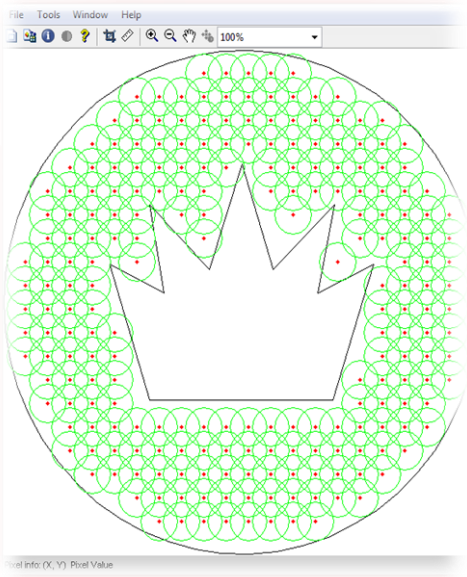
Con los parámetros iniciales, se crea una rejilla de puntos, evaluando si pertenecen o no al área de la pieza a maquinar (Ver figura 105), los puntos rojos están dentro y permanecen mientras los azules se eliminan.

Figura 105. Selección de puntos pertenecientes a la cavidad 2



Se eliminan los puntos descartados (Ver figura 106).

Figura 106. Puntos maquinables en la cavidad 2



Una vez identificados los puntos, el software itera calculando posibles rutas, extendiendo una banda elástica compuesta por el mismo número de puntos ubicados dentro de la cavidad (Ver figura 107), y busca una trayectoria hasta encontrar la de menor recorrido (Ver figura 108).

Figura 107. Banda elástica en la cavidad 2

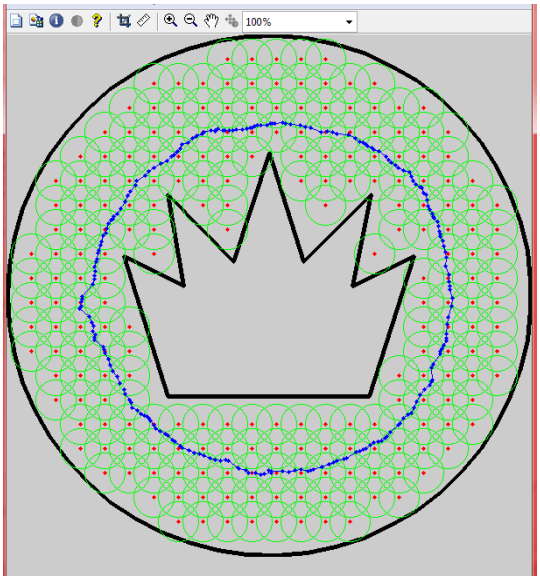
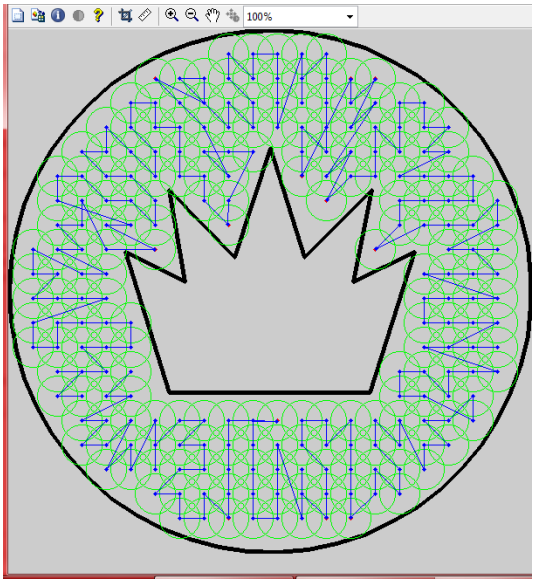


Figura 108. Iteraciones de las trayectorias de la cavidad 2



Finalizado el proceso de Optimizar la trayectoria es necesario corregir ciertas rutas que son indeseables en la trayectoria trazada (Ver figura 109 Y 110).

Figura 109. Corrección de la optimización en la cavidad 2

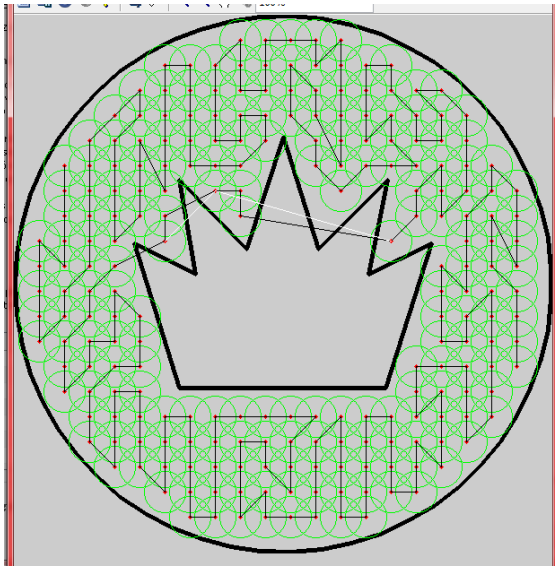
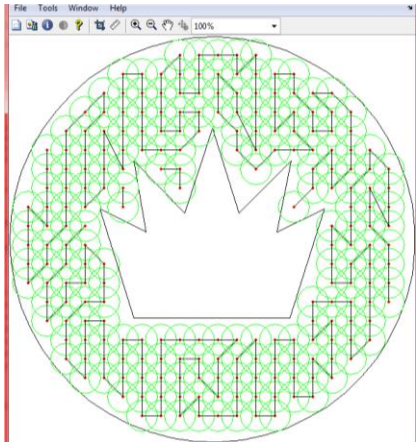


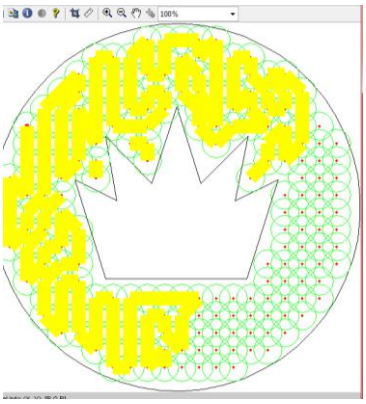
Figura 110. Trayectoria corregida de la cavidad 2



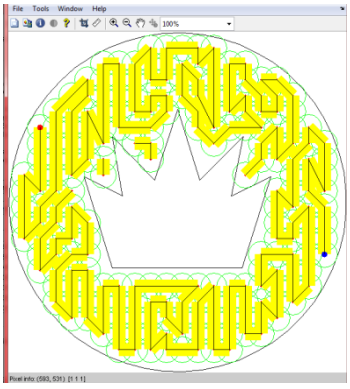
Una vez corregida la trayectoria, se dibuja el recorrido de la herramienta (Ver figura 101)

Figura 111. Recorrido de la herramienta en la cavidad 2

a) Inicial



b) Final



Y obtenemos el recorrido en 3 dimensiones (Ver figura 112) y las coordenadas de los puntos maquinables (Ver figura 113).

Figura 112. Trayectoria en 3 dimensiones de la cavidad 2

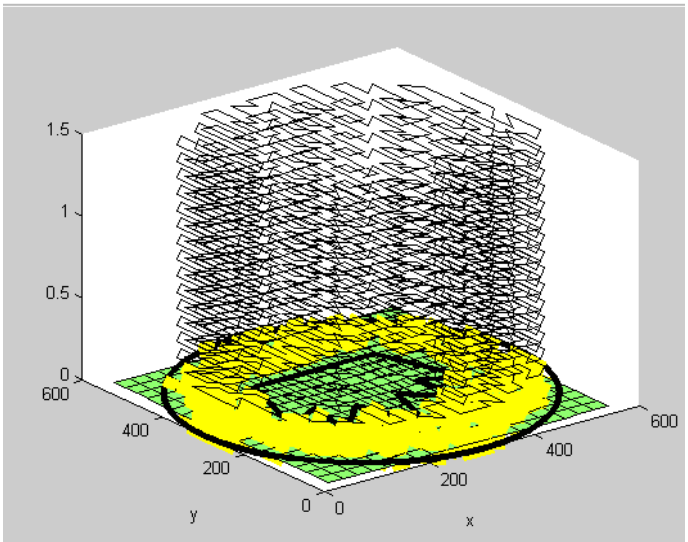
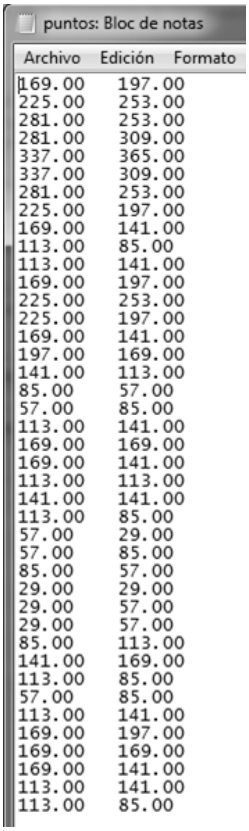


Figura 113. Coordenadas de la trayectoria del maquinado de la cavidad 2


puntos: Bloc de notas

Archivo Edición Formato

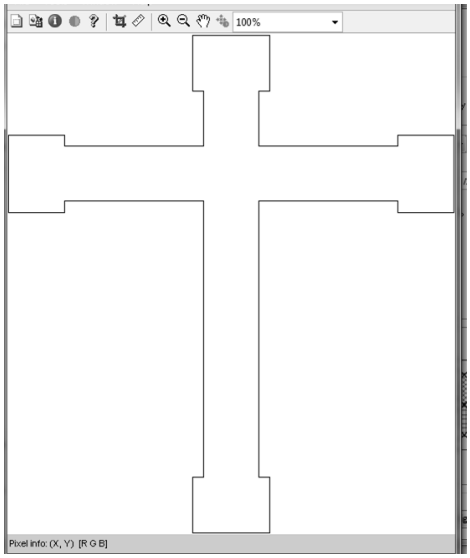
169.00	197.00
225.00	253.00
281.00	253.00
281.00	309.00
337.00	365.00
337.00	309.00
281.00	253.00
225.00	197.00
169.00	141.00
113.00	85.00
113.00	141.00
169.00	197.00
225.00	253.00
225.00	197.00
169.00	141.00
197.00	169.00
141.00	113.00
85.00	57.00
57.00	85.00
113.00	141.00
169.00	169.00
169.00	141.00
113.00	113.00
141.00	141.00
113.00	85.00
57.00	29.00
57.00	85.00
85.00	57.00
29.00	29.00
29.00	57.00
29.00	57.00
85.00	113.00
141.00	169.00
113.00	85.00
57.00	85.00
113.00	141.00
169.00	197.00
169.00	169.00
169.00	141.00
113.00	141.00
113.00	85.00

Tabla 6. Longitud de las trayectorias de la cavidad 2 en Neural Net Optrasoft

Cavidad	*%*	*Diametro de la fresa*	*Neural Net Optrasoft*
2	40	1/4	
		3/8	156,25
		1/2	104,12
		1	25,97
	50	1/4	217,74
		3/8	127,14
		1/2	88,13
		1	25,09
	60	1/4	166,70
		3/8	101,39
		1/2	70,07
		1	16,79
	70	1/4	147,88
		3/8	88,56
		1/2	59,82
		1	19,44
	80	1/4	123,13
		3/8	77,42
		1/2	53,93
		1	15,11
	90	1/4	112,51
		3/8	69,20
		1/2	48,79
		1	14,56

5.2.3 Descripción del procedimiento utilizado por NEURAL NET OPTRASOFT para obtener la longitud recorrida por la herramienta para la cavidad 3. Una vez seleccionada la herramienta y los parámetros de corte, el software calcula y verifica las proporciones de la imagen en píxeles, y la interpreta en una pantalla de imagen de Matlab (Ver figura 114).

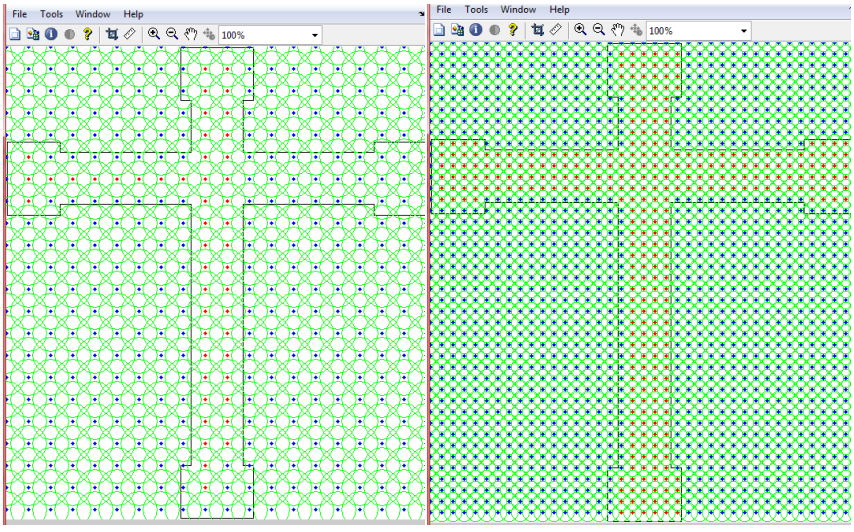
Figura 114. Reconocimiento de la Imagen de la cavidad 3



Con los parámetros iniciales, se crea una rejilla de puntos, evaluando si pertenecen o no al área de la pieza a maquinar (Ver figura 115), los puntos rojos están dentro y permanecen mientras los azules se eliminan.

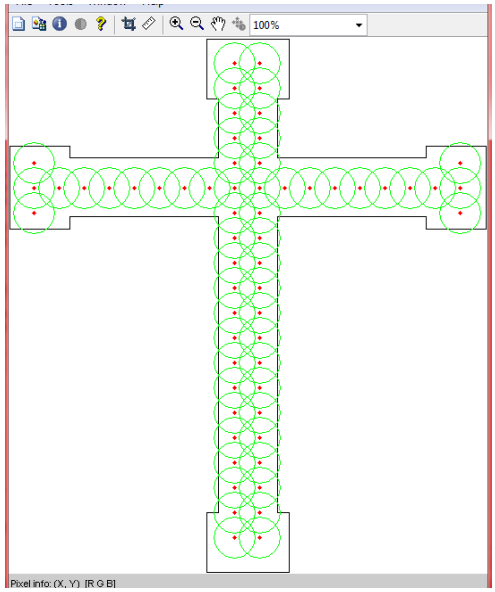
Figura 115. Selección de puntos pertenecientes a la cavidad

a) Superposición de 40 % b) Superposición de 70 %



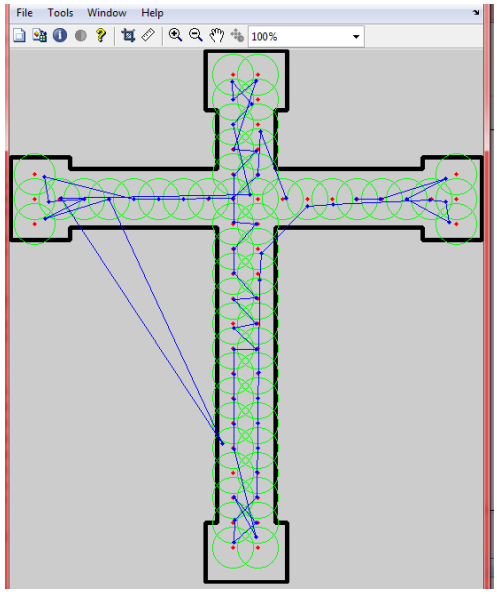
Se eliminan los puntos descartados (Ver figura 116).

Figura 116. Puntos maquinables en la cavidad 2



Una vez identificados los puntos, el software itera calculando posibles rutas, extendiendo una banda elástica compuesta por el mismo número de puntos ubicados dentro de la cavidad, y busca una trayectoria hasta encontrar la de menor recorrido (Ver figura 117).

Figura 117. Iteraciones de las trayectorias de la cavidad 3



Finalizado el proceso de Optimizar la trayectoria es necesario corregir ciertas rutas que son indeseables en la trayectoria trazada (Ver figura 118 y 119).

Figura 118. Corrección de la optimización en la cavidad 3

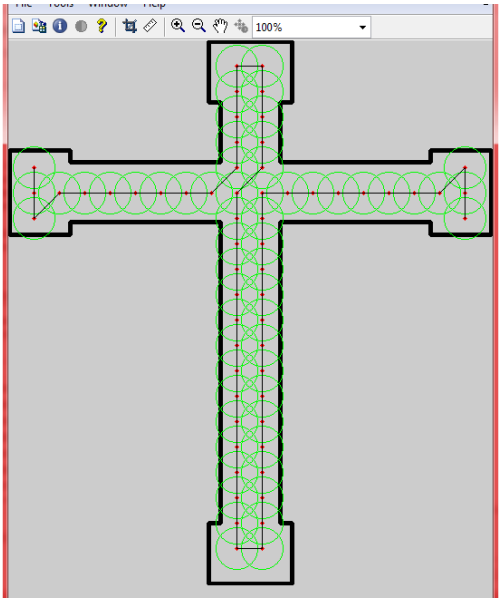
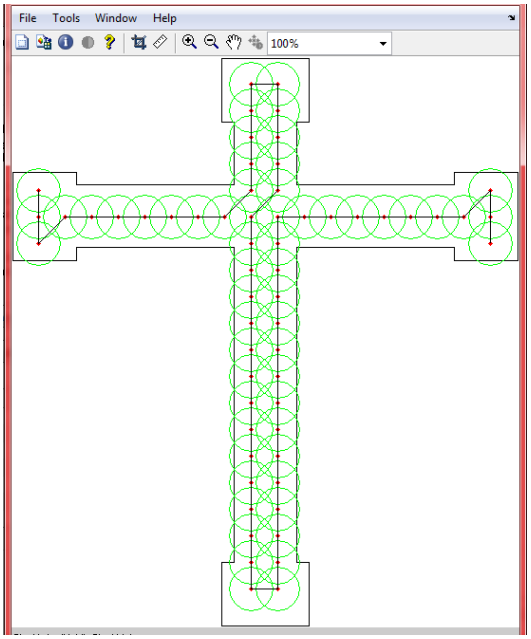
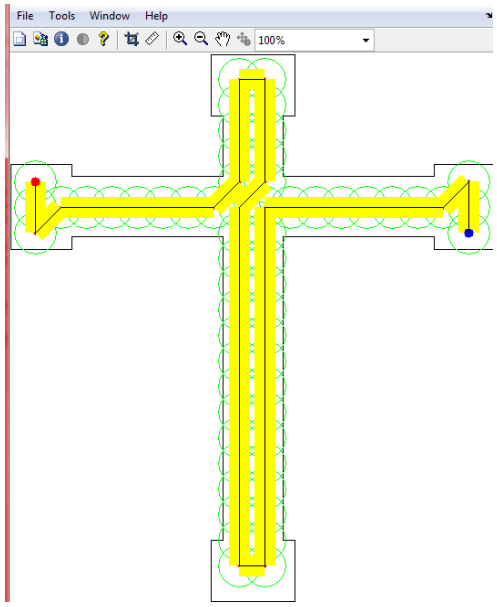


Figura 119. Trayectoria corregida de la cavidad 3



Una vez corregida la trayectoria, se dibuja el recorrido de la herramienta (Ver figura 120).

Figura 120. Recorrido de la herramienta en la cavidad 3



Y obtenemos el recorrido en 3 dimensiones (Ver figura 121) y las coordenadas de los puntos maquinables (Ver figura 122).

Figura 121. Trayectoria en 3 dimensiones de la cavidad 2

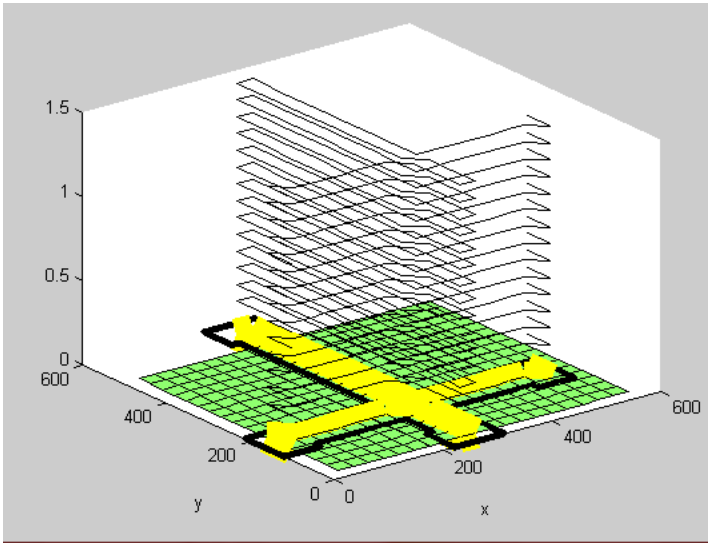


Figura 122. Coordenadas de la trayectoria del maquinado de la cavidad 2

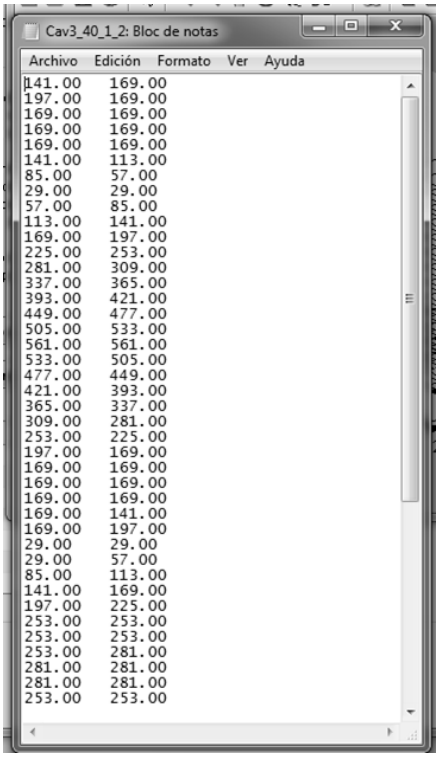


Tabla 7. Longitud de las trayectorias de la cavidad 3 en Neural Net Optrasotf

Cavidad	*%*	*Diametro de la fresa*	*Neural Net Optrasoft*
3	40	1/4	80,63
		3/8	40,16
	50	1/4	67,22
		3/8	37,17
	60	1/4	49,86
		3/8	28,59
	70	1/4	50,49
		3/8	24,13
	80	1/4	39,46
		3/8	19,05
	90	1/4	35,58
		3/8	17,77

6. ANALISIS DE LOS RESULTADOS

6.1 RESULTADOS OBTENIDOS

Estos resultados son los obtenidos por las ocho trayectorias encontradas en la librería de Mastercam X5 aplicadas a las tres cavidades de prueba escogidas y los resultados por la simulación de la trayectoria de Neural Net Optrasoft en Mastercam X5.

Tabla 8. Longitudes de las trayectorias (pulgadas)

Cavidad	*%*	*Diametro de la fresa*	*Zigzag*	*Constant overlap spiral*	*Parallel spiral*	*Parallel spiral cleancorner*	*Morph spiral*	*High speed*	*One Way*	*True spiral*	*Neural Net Optrasoft*
1	40	3/8	138,80	222,76	159,76	171,81	286,27	375,52	278,68	216,16	222,60
		1/2	116,25	157,77	137,74	148,46	230,50	239,65	220,60	176,33	147,80
	50	3/8	126,25	208,49	131,80	139,03	246,46	383,45	236,64	187,45	177,52
		1/2	108,22	140,48	132,05	139,50	200,62	274,60	184,83	154,90	129,20
	60	3/8	120,18	195,69	143,53	152,00	220,62	309,37	217,80	169,29	149,24
		1/2	105,76	139,53	107,20	111,19	180,95	215,15	168,99	142,01	96,00
	70	3/8	112,81	206,33	136,81	142,70	201,91	307,94	198,36	150,87	122,56
		1/2	103,49	138,23	102,94	108,78	166,07	236,57	161,47	129,14	82,63
	80	3/8	109,39	211,63	132,68	137,80	182,40	385,02	177,39	146,63	112,92
		1/2	97,70	137,03	101,89	106,36	156,65	231,37	146,02	119,31	76,80
	90	3/8	106,65	183,26	128,36	132,19	172,46	338,00	170,57	136,23	92,37
		1/2	96,58	136,51	99,82	103,09	146,98	252,11	134,47	121,50	58,23
2	40	1/4	216,30	344,56	239,75	251,48	367,98	544,51	359,83	280,01	
		3/8	136,09	173,51	135,92	140,21	174,98	360,69	268,28	141,28	156,25
		1/2	107,54	147,05	150,58	153,47	125,86	282,72	182,70	114,97	104,12
		1	54,00	54,55	55,65	56,86	81,64	68,09	81,91	100,35	25,97
	50	1/4	166,88	198,87	208,46	215,18	209,87	443,30	339,74	170,40	217,74
		3/8	122,86	147,99	169,56	182,51	152,28	338,90	216,93	122,55	127,14
		1/2	99,79	111,37	122,13	125,36	124,87	244,48	163,93	104,11	88,13
		1	50,68	54,56	56,54	56,54	56,54	68,09	68,72	89,90	25,09
	60	1/4	150,05	200,60	193,39	197,81	186,53	389,25	282,49	150,21	166,70
		3/8	112,23	152,41	160,76	164,70	140,09	337,67	192,12	107,75	101,39
		1/2	95,52	112,26	138,51	142,92	102,93	276,13	145,96	96,00	70,07
		1	51,32	54,56	56,53	56,54	56,54	68,09	66,39	87,00	16,79

Tabla 8. (Continuación)

Cavidad	*%*	*Diametro de la fresa*	*Zigzag*	*Constant overlap spiral*	*Parallel spiral*	*Parallel spiral cleancorner*	*Morph spiral*	*High speed*	*One Way*	*True spiral*	*Neural Net Optrasoft*
2	70	1/4	140,73	176,17	180,20	183,08	171,51	389,46	259,05	133,69	147,88
		3/8	106,53	127,05	148,93	150,48	128,90	307,05	172,36	104,70	88,56
		1/2	89,79	99,63	102,40	102,40	106,39	245,88	127,93	89,51	59,82
		1	49,81	54,56	56,54	56,54	56,54	68,09	58,80	75,63	19,44
	80	1/4	128,77	176,28	166,82	169,68	158,04	342,65	227,07	128,32	123,13
		3/8	101,53	135,85	110,37	113,05	121,18	271,80	159,74	100,41	77,42
		1/2	93,12	102,94	100,62	100,62	91,44	231,85	126,64	88,91	53,93
		1	47,88	54,56	56,54	56,54	56,54	68,09	54,59	74,71	15,11
	90	1/4	123,97	157,06	151,37	153,35	148,78	356,37	205,24	120,87	112,51
		3/8	99,08	145,79	142,65	147,00	116,72	255,48	143,70	92,46	69,20
		1/2	86,47	103,05	91,05	91,05	99,42	208,15	117,07	84,55	48,79
		1	47,58	54,56	54,56	54,56	54,56	68,09	54,59	71,19	14,56
3	40	1/4	193,98	209,83	208,72	222,12	1135,92	598,12	446,46	725,61	80,63
		3/8	145,49	155,72	151,95	179,28	757,06	456,31	298,44	482,98	40,16
	50	1/4	171,73	189,76	178,47	191,69	925,09	580,79	373,19	606,72	67,22
		3/8	129,20	143,84	142,96	152,97	621,93	463,56	247,91	410,83	37,17
	60	1/4	157,03	175,64	170,73	178,78	785,17	663,05	326,66	515,66	49,86
		3/8	124,02	141,65	111,89	115,07	533,27	448,87	224,72	346,50	28,59
	70	1/4	142,03	172,15	144,90	148,69	685,72	652,10	282,19	454,79	50,49
		3/8	113,56	124,10	140,70	148,95	468,88	414,76	197,77	316,07	24,13
	80	1/4	135,96	157,52	141,90	147,95	606,31	702,38	259,95	409,38	39,46
		3/8	117,13	123,71	115,51	130,27	418,81	296,77	194,53	289,27	19,05
	90	1/4	137,94	156,84	138,02	143,82	552,72	712,49	255,82	376,08	35,58
		3/8	101,94	122,66	112,54	118,18	378,64	291,83	162,61	259,58	17,77

6.2 ANALISIS ESTADISTICO DE LOS RESULTADOS

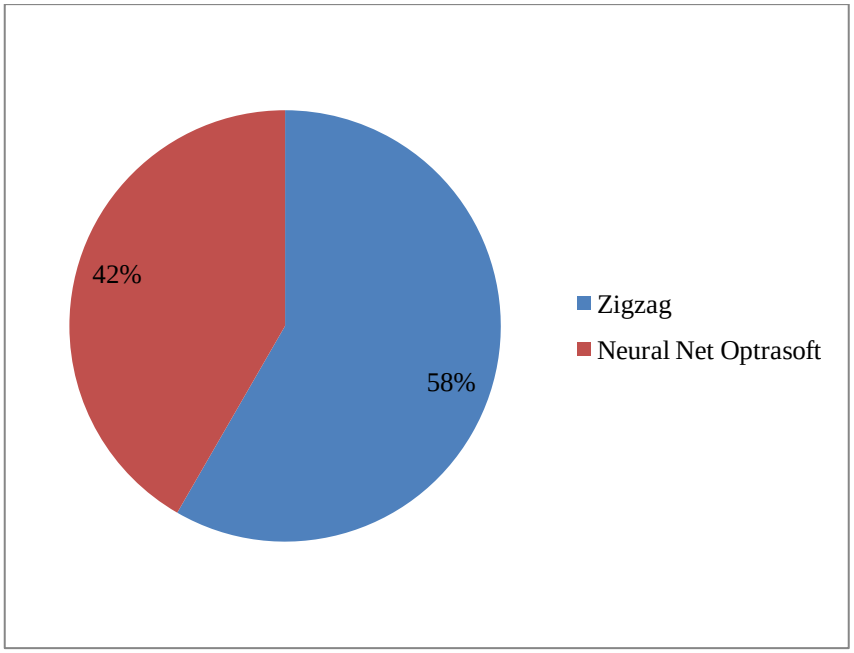
6.2.1 Análisis de la cavidad 1

En la cavidad No 1 se hicieron pruebas con dos diámetros diferentes de la herramienta, y seis porcentajes de sobreposición, para un total de 12 pruebas, de las cuales se obtuvieron los siguientes resultados:

Tabla 9. Tabla de frecuencia de la mejor trayectoria en la cavidad 1

Trayectoria	Frecuencia
Zigzag	7
Neural Net Optrasoft	5

Figura 123. Desempeño de las trayectorias en la cavidad No. 1

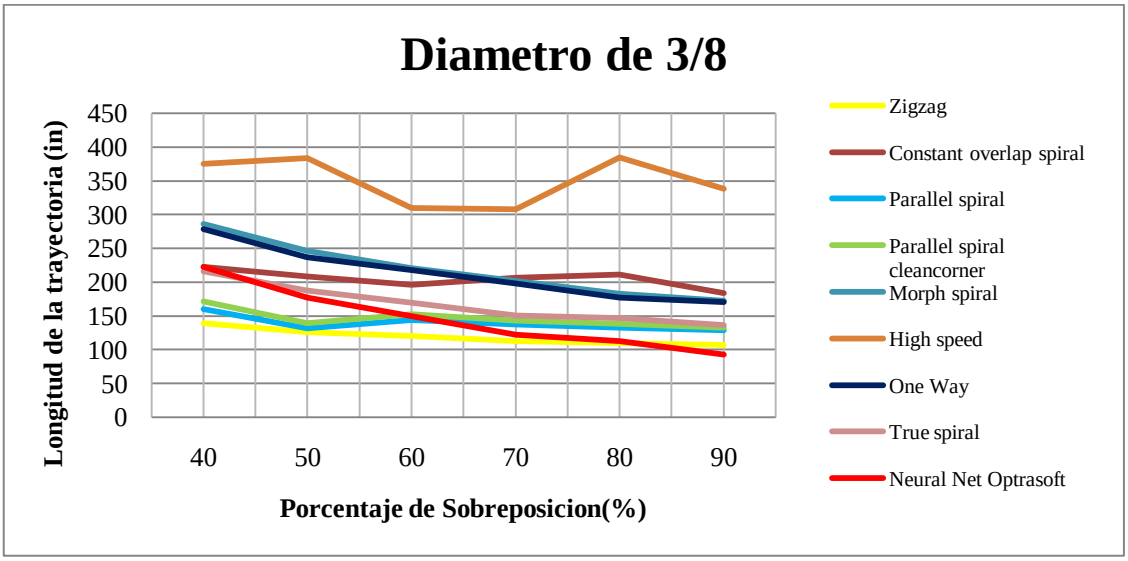


Fuente: autores.

El 42% de las pruebas demostraron que Neural Net Optrasoft obtuvo mejores desempeños con respecto a los ocho tipos de trayectorias usados por Mastercam X5, superado solamente por la trayectoria Zigzag que demostró ser la mejor trayectoria para este tipo de cavidad.

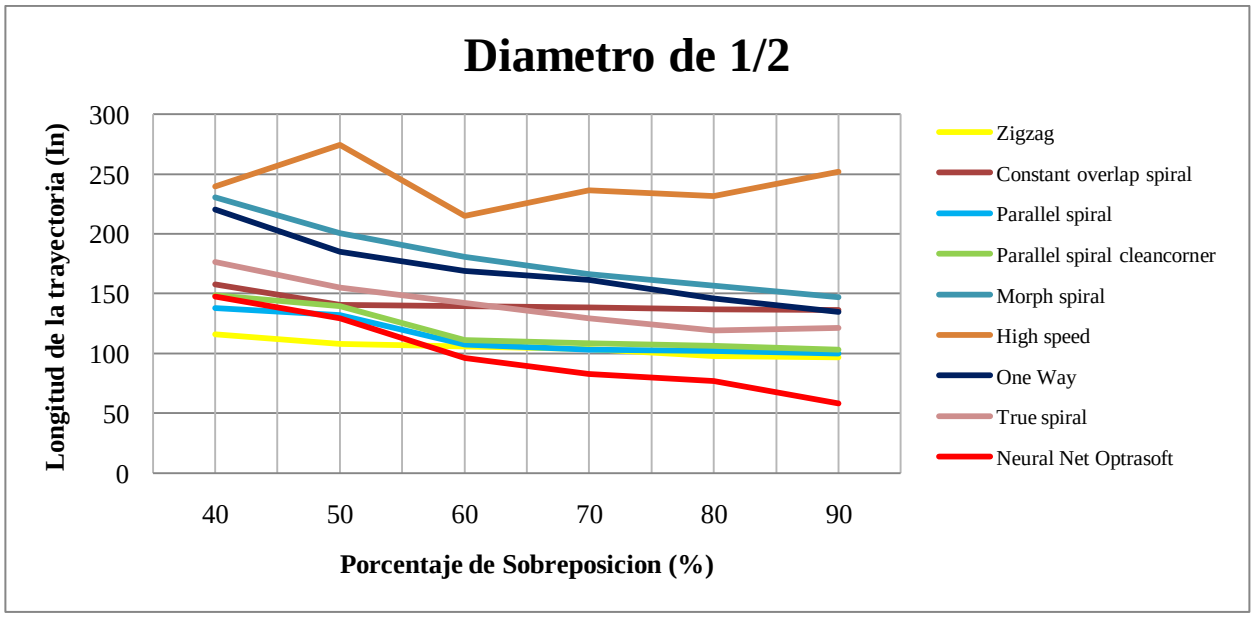
Midiendo el desempeño individual con un diámetro de 3/8” y ½” conocemos la influencia del porcentaje de sobreposición.

Figura 124. Trayectorias de la cavidad 1 con diámetro de 3/8”



Fuente: autores.

Figura 125. Trayectorias de la cavidad 1 con diámetro ½“



Fuente: autores.

A mayor diámetro y mayor porcentaje de sobreposición, la longitud de la trayectoria se hace menor, aumentando el desempeño del maquinado. En

cavidades simples es mejor utilizar la trayectoria de zigzag para diámetros pequeños y Neural Net Optrasoft para diámetros grandes.

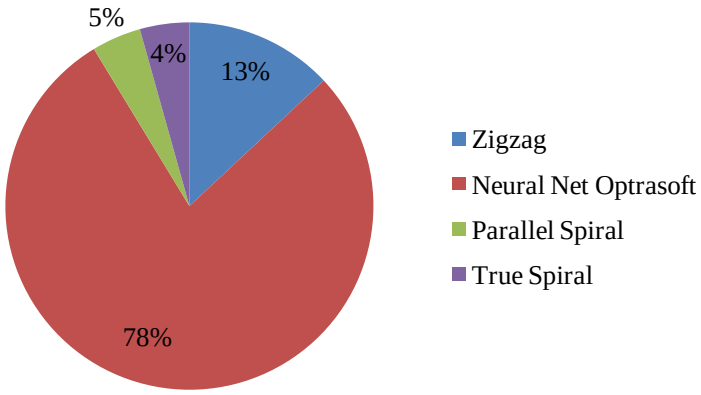
6.2.2 Análisis de la cavidad 2

En la cavidad No 2 se hicieron las pruebas con los cuatro diámetros comunes en el fresado, y seis porcentajes de sobreposición, para un total de 24 pruebas, de las cuales se obtuvieron los siguientes resultados:

Tabla 10. Tabla de frecuencia de la mejor trayectoria en la cavidad 2

Trayectoria	Frecuencia
Zigzag	3
Neural Net Optrasoft	18
Parallel Spiral	1
True Spiral	1

Figura 126. Desempeño de las trayectorias en la cavidad No. 2

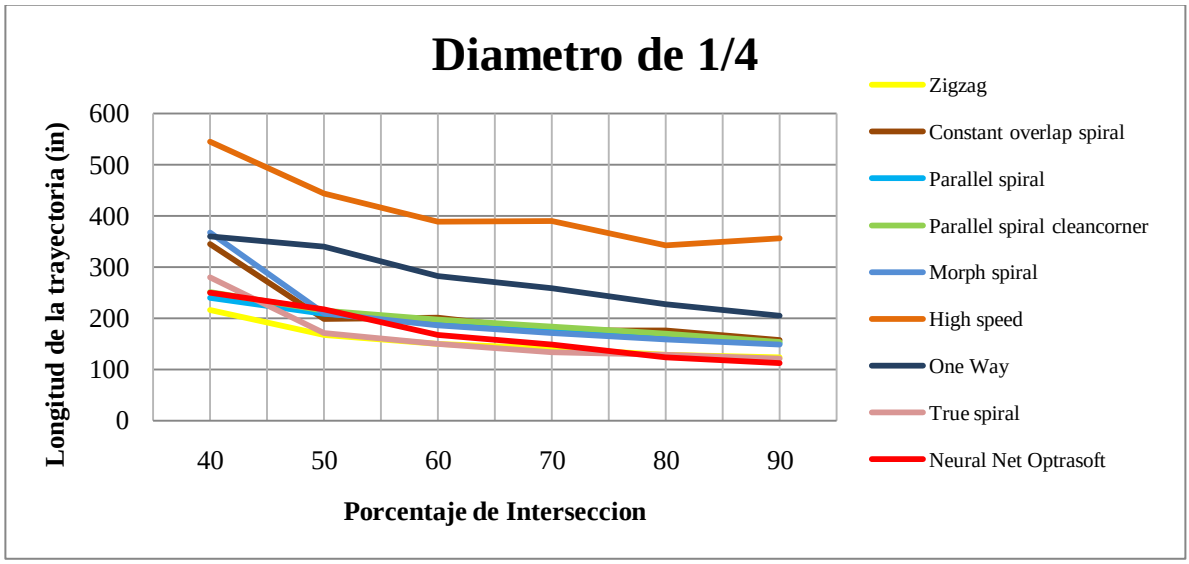


Fuente: autores.

El 78% de las pruebas demostraron que Neural Net Optrasoft obtuvo mejores desempeños en cuanto a longitud de trayectorias con respecto a los ocho tipos de trayectorias usados por Mastercam X5, seguido de la trayectoria Zigzag pero no muy significante.

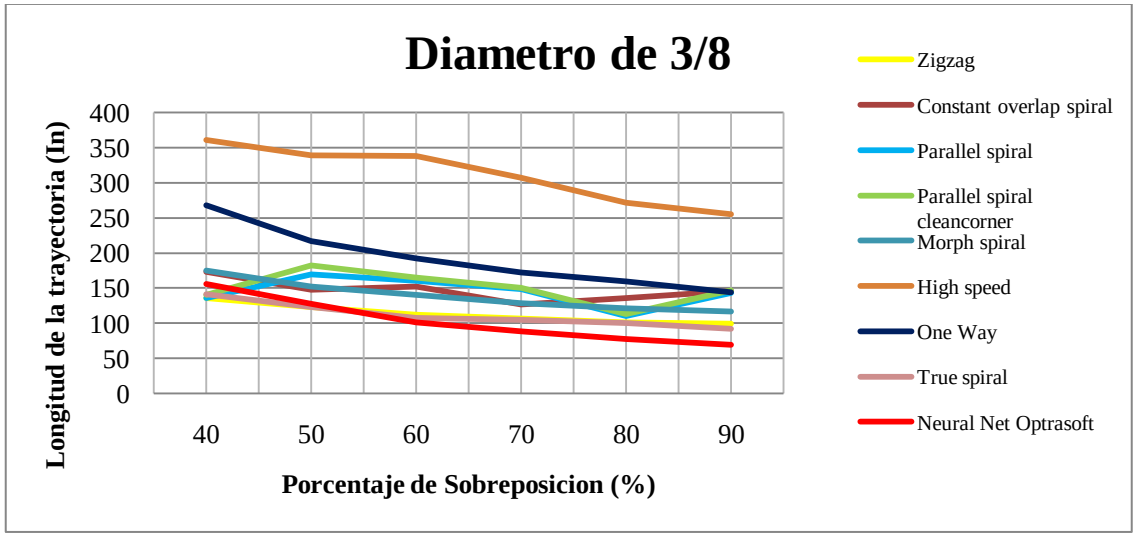
Midiendo el desempeño individual con un diámetro de ¼”, 3/8”, ½”, y 1” conocemos la influencia del porcentaje de sobreposición.

Figura 127. Trayectorias de la cavidad 2 con diámetro de 1/4”



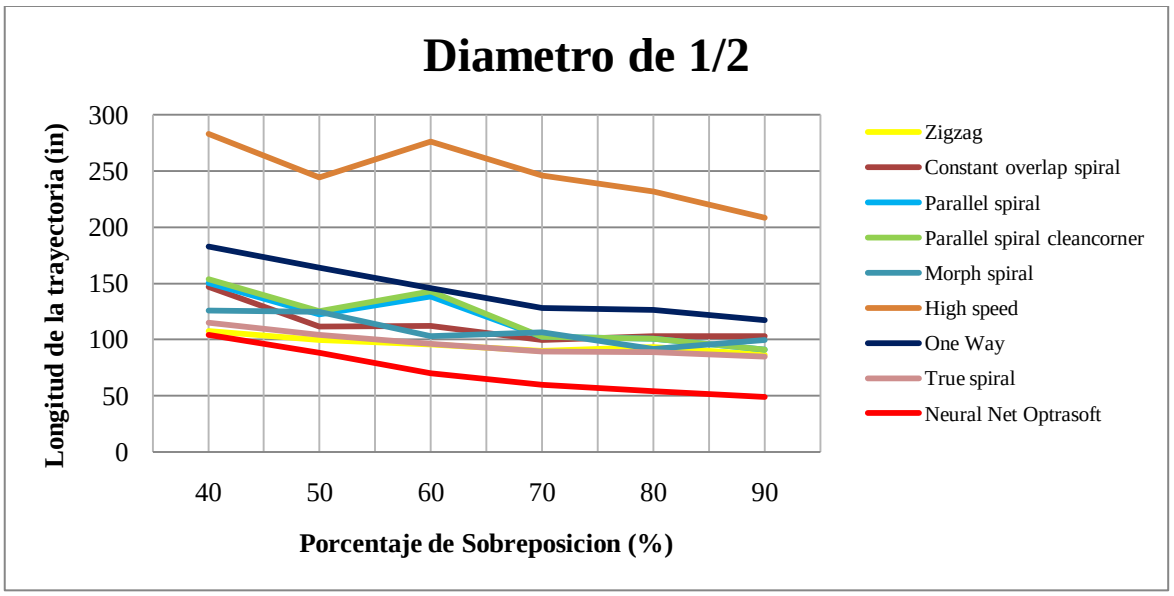
Fuente: autores.

Figura 128. Trayectorias de la cavidad 2 con diámetro de 3/8”



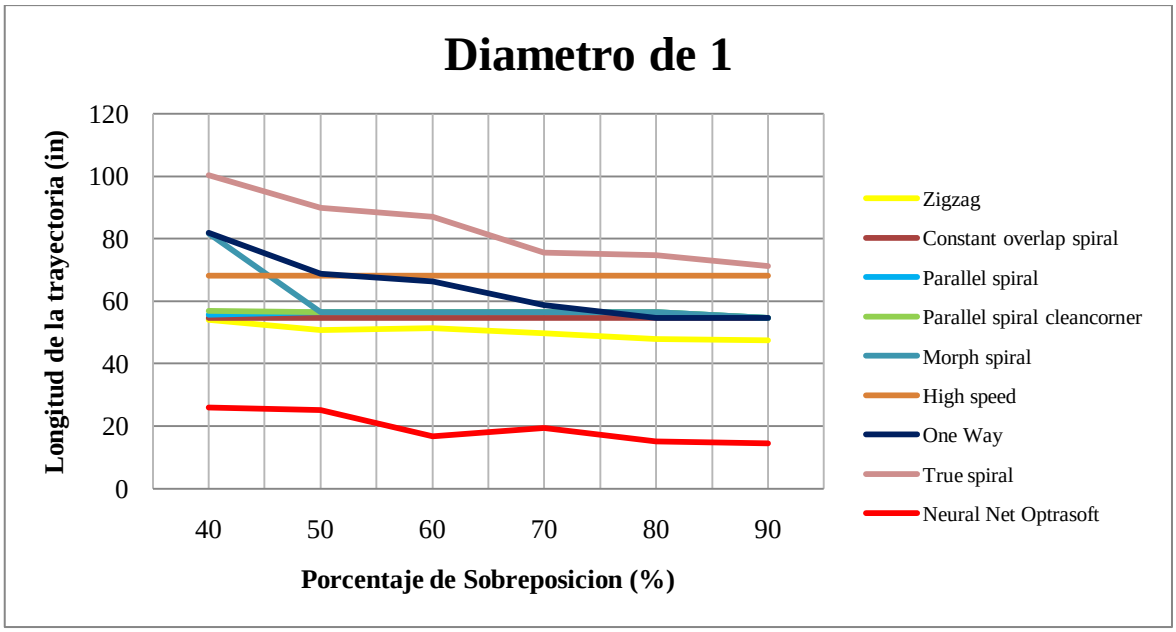
Fuente: autores.

Figura 129. Trayectorias de la cavidad 2 con diámetro de ½”



Fuente: autores.

Figura 130. Trayectorias de la cavidad 2 con diámetro de 1



Fuente: autores.

A mayor diámetro y mayor porcentaje de sobreposición, la longitud de la trayectoria se hace menor, aumentando el desempeño del maquinado. En

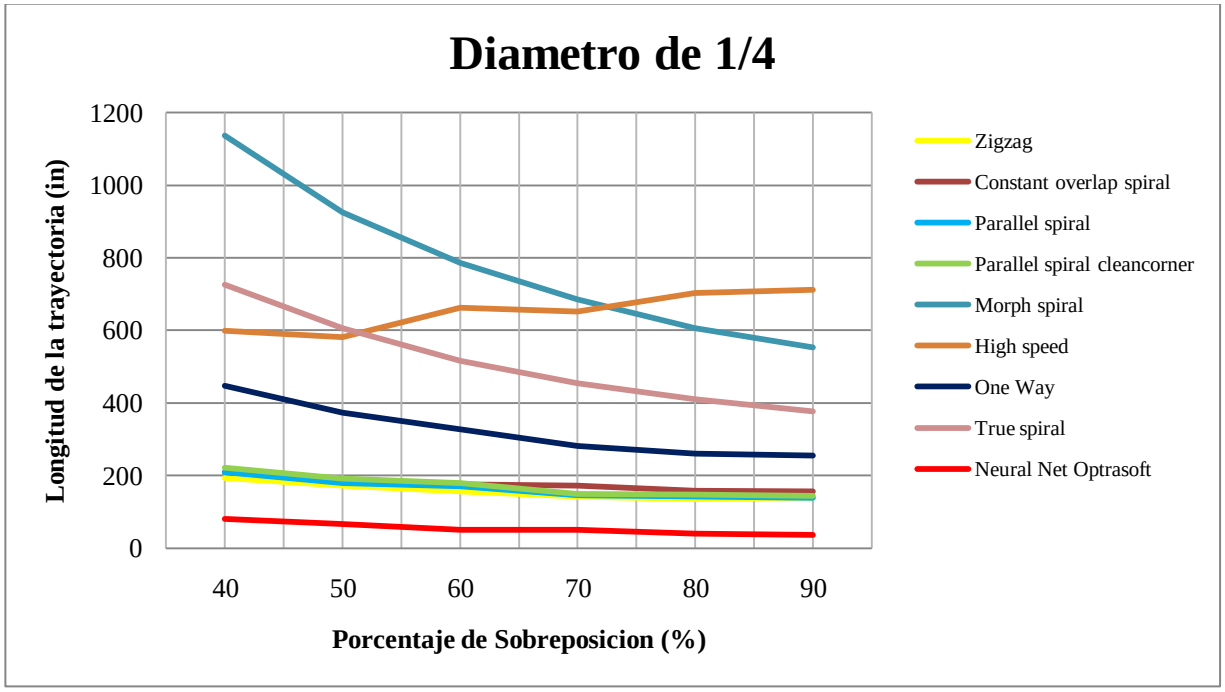
cavidades con islas complejas, Neural Net Optrasoft demostró un mejor desempeño en casi todas las condiciones, dando el mejor resultado en el diámetro más grande, y dando su máximo desempeño en porcentajes de sobreposición altos.

6.2.3 Análisis de la cavidad 3.

En la cavidad No 3 se hicieron pruebas con dos diámetros diferentes, y seis porcentajes de sobreposición, para un total de 12 pruebas, de las cuales se comprobó una eficiencia del 100% en este tipo de cavidad, superando por márgenes considerablemente altos los ocho tipos de trayectorias de Mastercam X5.

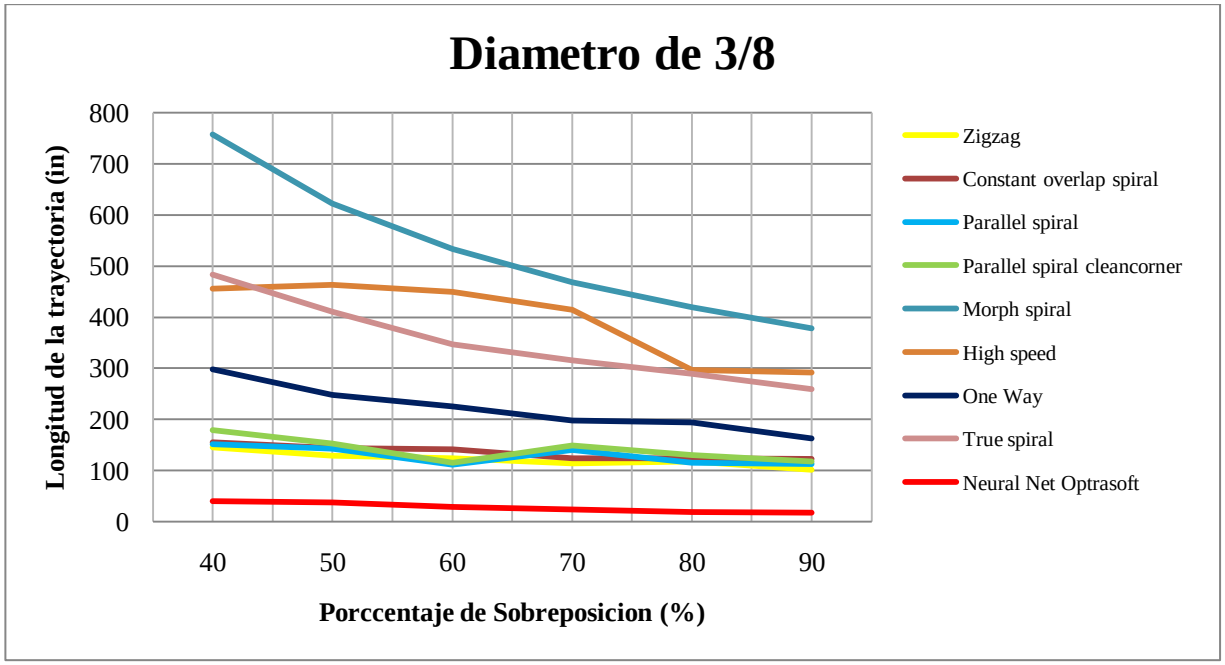
Midiendo el desempeño individual con diámetro de ¼" y 3/8" conocemos la influencia del porcentaje de sobreposición para este tipo de cavidad.

Figura 131. Trayectorias de la cavidad 3 con diámetro de 1/4"



Fuente: autores.

Figura 132. Trayectorias de la cavidad 3 con diámetro de 3/8”



Fuente: autores.

En cavidades complejas, Neural Net Optrasoft demostró el mejor desempeño en todas las condiciones, superando todo tipo de trayectorias en el software Mastercam X5, y una tendencia más uniforme con las variaciones de porcentaje.

La tendencia casi constante en todo tipo de cavidades de las trayectorias demuestra que a mayor diámetro y mayor porcentaje de sobreposición aumenta el desempeño del maquinado lo que implica una longitud de trayectoria menor, ahorrando tiempo y herramienta.

Al aumentar el diámetro de la herramienta y aumentar el porcentaje de sobreposición se cubre mayor área de maquinado con una sola pasada, esto beneficia tanto en Mastercam como Neural Net Optrasoft que se traduce en una disminución de la longitud de la trayectoria, la diferencia radica en que el software que se diseño está basado en el principio de no tener que volver a pasar por el mismo lugar dos veces, haciendo más evidente la ventaja al disminuir el área a maquinar con cada avance y optimizar ese espacio que queda.

Dependiendo del tipo de cavidad el software Neural Net Optrasoft mostro mejores desempeños, en cavidades complejas, con formas irregulares, mientras Mastercam X5 tiene mejores desempeños en cavidades simples, con formas regulares, debido a que mientras Neural Net Optrasoft analiza la cavidad como una serie de puntos y traza su recorrido de forma eficiente, Mastercam X5 se limita a trazar su recorrido con una forma preestablecida, sin importar si hay o no una trayectoria mejor.

7. CONCLUSIONES

Se cumplió el objetivo del proyecto "Aplicación de redes neuronales para la optimización de la trayectoria de la herramienta en el fresado de cavidades complejas" que fue el desarrollo de un software que mediante la digitalización de un número finito de puntos dentro del área de la cavidad, permite identificar la trayectoria optima de la herramienta en superficies de dos dimensiones, implementado un modelo fundamentado en redes Neuronales (Red de Kohonen) para la solución de trayectorias más cortas para el fresado de diferentes tipos de cavidades, teniendo en cuenta el diámetro de la herramienta, el porcentaje de sobreposición y restricciones propias de cada cavidad.

Al finalizar el proyecto se han alcanzado los objetivos planteados al inicio, elaborando una herramienta informática que facilita la planeación y optimización de la trayectoria de la herramienta en el fresado de cavidades complejas, contribuyendo al desarrollo industrial en las empresas del sector metalmecánico y por medio de ella fomentando el conocimiento y uso de técnicas modernas de computación, para procesos metalmecánicos dentro de la industria general.

En el aspecto económico el proyecto ofrece resultados que sirven de apoyo en la implementación de un nuevo modelo de trayectoria que tiene gran influencia en los tiempos de maquinado y la vida útil de las herramientas dando lugar a ahorros de tiempo, costos de la herramienta y mayor productividad, al obtener trayectorias más cortas haciendo buen uso del espacio, obedeciendo a principios de eficiencia.

En el aspecto de ingeniería, el proyecto demuestra una vez más que de la teoría a la práctica solo está el Ingeniero, quien con el aprovechamiento de los conceptos teóricos adquiridos en su formación profesional lleva a la industria que lo rodea soluciones a problemas reales soportadas en la ciencia para el desarrollo.

Del desarrollo del proyecto se concluye que:

- Al aumentar el diámetro de la herramienta y aumentar el porcentaje de sobreposición se cubre mayor área de maquinado con una sola pasada, esto beneficia tanto en Mastercam como Neural Net Optrasoft que se traduce en una disminución de la longitud de la trayectoria, la diferencia radica en que el software que se diseño está basado en el principio de no tener que volver a pasar por el mismo lugar dos veces, haciendo más evidente la ventaja al disminuir el área a maquinar con cada avance y optimizar ese espacio que queda.
- Dependiendo del tipo de cavidad el software Neural Net Optrasoft mostro mejores desempeños, en cavidades complejas, con formas irregulares, mientras Mastercam X5 tiene mejores desempeños en cavidades simples, con formas regulares, debido a que mientras Neural Net Optrasoft analiza la cavidad como una serie de puntos y traza su recorrido de forma eficiente, Mastercam X5 se limita a trazar su recorrido con una forma preestablecida, sin importar si hay o no una trayectoria mejor.
- Comparando la trayectoria de la herramienta generada por el software con la tecnología existente en este caso MASTERCAM, tomando como parámetro de comparación la longitud de la trayectoria generada por cada uno de los ocho tipos de trayectorias y comparándola con la longitud utilizada por Neural Net Optrasoft, se confirma la eficiencia de este nuevo tipo de trayectoria implementado en el nuevo software.

BIBLIOGRAFIA

- ANGENIOL, B. VAUBOIS, G. y TELIER, J. Self-organizing feature maps and the traveling salesman problem. Neural Networks 1. 1988.

- BRALLA, James G. Manual de diseño de producto para manufactura: Guía práctica para producción a bajo costo. México: McGraw-Hill, 1993.

- CASTELINO, Kenneth; D´SOUZA, Roshan y WRIGHT, Paul K. Tool-path optimization for minimizing airtime during machining. University of California at Berkeley. 2001.

- BLACK, C. Stewart, CHILES, Vic, LISSAMAN, A. J., MARTIN, S. J., Principios de ingeniería de manufactura. México: CECSA. 1999.

- GROOVER, Mikell P. Fundamentos de manufactura moderna, materiales, procesos y sistemas. México: Prentice Hall. 1997.

- HILERA, José R. y MARTÍNEZ, Víctor J. Redes Neuronales Artificiales: Fundamentos, Modelos y Aplicaciones. Madrid: Alfaomega. 2000.

- HOPFIELD, J. J. y TANK, D. W. Neural computation of decisions in optimization problem. Biological cybernetics Vol. 52 (1985); p 141-152.

- KOHONEN, Teuvo. Self organization and associative memory. New York: Springer-Verlag. 1989.

- MARTELLOTI, M. An analysis of the milling process. Transactions of ASME. 1945.

- PARK, S.C. y CHOI, B.K. Tool-path planning for direction-parallel area milling. Comput Aided Des Vol 32 (2000); p 17 –25.

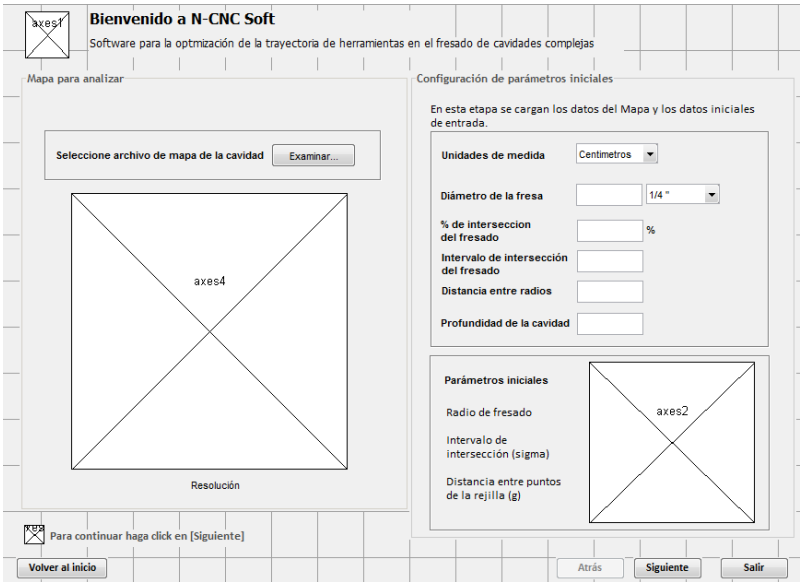
- RAMACHANDRAN, Prabhy. Feasible offset region based tool path planning for face milling. CODEF. 2003.

- SUK-HWAN, Suh y YANG-SOO, Shin. Neural network modeling for tool path planning of the rough cut in complex pocket milling. Journal of Manufacturing Systems. 1996.

ANEXOS

Anexo A. CODIGO DE PROGRAMACION PASO 1

Figura 133. Archivo Step1.fig



Fuente: autores.

Código Step1.m

```
function varargout = Step1(varargin)
% End initialization code - DO NOT EDIT

% --- Executes just before Step1 is made visible.
function Step1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.

% Choose default command line output for Step1
handles.output = hObject;
scrsz = get(0,'ScreenSize');
pos_act = get(gcf, 'Position');
xr = scrsz(3) - pos_act(3);
xp = round(xr/2);
yr = scrsz(4) - pos_act(4);
yp = round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);

% Update handles structure
axes(handles.axes1);
img = imread('imagenes/3620-fresadora-8-mm.jpg');
imshow(img);
```

```
axes(handles.axes4);
fig = imread('imagenes/default.jpg');
imshow(fig);
axes(handles.axes3);
fig = imread('imagenes/next.jpg');
imshow(fig);
axes(handles.axes2);
fig = imread('imagenes/param_fresa.jpg');
imshow(fig);
assignin('base','U_FACTOR',0.0265);
set(handles.txtRadioFresa,'string','');
set(handles.txtSigma,'string','');
set(handles.txtG,'string','');
set(handles.porcentaje,'string','');
set(handles.txtProfundidad,'string','');
set(handles.popupmenu1,'value',1);

guidata(hObject, handles);
% UIWAIT makes Step1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = Step1_OutputFcn(hObject, eventdata, handles)

% --- Executes on button press in pushbuttonBack.
function pushbuttonBack_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonBack (see GCBO)

% --- Executes on button press in pushbuttonNext.
function pushbuttonNext_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonNext (see GCBO)
radioFresa = str2double(get(handles.txtRadioFresa,'string'));
sigma = str2double(get(handles.txtSigma,'string'));
g = str2double(get(handles.txtG,'string'));
prof = str2double(get(handles.txtProfundidad,'string'));
U_FACTOR = evalin('base','U_FACTOR');
radioFresa = fix(radioFresa/U_FACTOR);
sigma = fix(sigma/U_FACTOR);
g = fix(g/U_FACTOR);

assignin('base','radioFresa',radioFresa/2);
assignin('base','sigma',sigma);
assignin('base','g',g);
assignin('base','profundidad',prof);
ocultarVentana('h1');
```

```
h2 = Step2;
assignin('base','h2',h2);

% --- Executes on button press in pushbuttonExit.
function pushbuttonExit_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonExit (see GCBO)
choice = questdlg('¿Realmente desea salir?','Salir','Si','No','No');
switch choice
    case 'Si'
    clear all;
    clc;
    close;
end


% --- Executes on button press in pushbuttonHome.
function pushbuttonHome_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonHome (see GCBO)

% --- Executes during object creation, after setting all properties.
function pushbuttonExit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to pushbuttonExit (see GCBO)

% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu1 contents as
cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu1
U_FACTOR_0 = evalin('base','U_FACTOR');
v = get(hObject,'Value');
switch v
    case 1
    U_FACTOR = 0.0265; %Pixeles a centimetros
    set(handles.sizeBroca,'visible', 'on');
    case 2
    U_FACTOR = 0.01043; %Pixeles a pulgadas
    set(handles.sizeBroca,'visible', 'on');
    case 3
    U_FACTOR = 1; %Pixeles a pixeles
    set(handles.sizeBroca,'visible', 'off');
end
U_FACTOR_F = U_FACTOR;
```

```
if U_FACTOR_0 ~= U_FACTOR_F

   if(length(get(handles.txtRadioFresa,'string'))>0)
      txt                                                                    =
str2num(get(handles.txtRadioFresa,'string'))*U_FACTOR_F/U_FACTOR_0;
      set(handles.txtRadioFresa,'string',txt);
    end
   if(length(get(handles.txtSigma,'string'))>0 )
      txt = str2num(get(handles.txtSigma,'string'))*U_FACTOR_F/U_FACTOR_0;
      set(handles.txtSigma,'string',txt);
   end
   if(length(get(handles.txtG,'string'))>0 )
      txt = str2num(get(handles.txtG,'string'))*U_FACTOR_F/U_FACTOR_0;
      set(handles.txtG,'string',txt);
   end
   if(length(get(handles.txtProfundidad,'string'))>0 )
      txt                                                                    =
str2num(get(handles.txtProfundidad,'string'))*U_FACTOR_F/U_FACTOR_0;
      set(handles.txtProfundidad,'string',txt);
   end
end
set(handles.text9,'string',      sprintf('%3.1f      <      r      <      %3.1f',6*U_FACTOR,
101*U_FACTOR));
assignin('base','U_FACTOR',U_FACTOR);

% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)

function txtRadioFresa_Callback(hObject, eventdata, handles)
% hObject    handle to txtRadioFresa (see GCBO)

% Hints: get(hObject,'String') returns contents of txtRadioFresa as text
%          str2double(get(hObject,'String')) returns contents of txtRadioFresa as a
double
r =  str2double(get(handles.txtRadioFresa,'string'));
sigma = (2-1.414)*r;
set(handles.text10,'string',sprintf('> %3.3f',sigma));

% --- Executes during object creation, after setting all properties.
function txtRadioFresa_CreateFcn(hObject, eventdata, handles)
% hObject    handle to txtRadioFresa (see GCBO)

function txtSigma_Callback(hObject, eventdata, handles)
% hObject    handle to txtSigma (see GCBO)
```

```
% Hints: get(hObject,'String') returns contents of txtSigma as text
%        str2double(get(hObject,'String')) returns contents of txtSigma as a double
r =  str2double(get(handles.txtRadioFresa,'string'));
sigma = str2double(get(handles.txtSigma,'string'));
g = 1.414*r;
set(handles.text11,'string',sprintf('< %3.3f',g));

% --- Executes during object creation, after setting all properties.
function txtSigma_CreateFcn(hObject, eventdata, handles)
% hObject    handle to txtSigma (see GCBO)

function txtG_Callback(hObject, eventdata, handles)
% hObject    handle to txtG (see GCBO)

% --- Executes during object creation, after setting all properties.
function txtG_CreateFcn(hObject, eventdata, handles)
% hObject    handle to txtG (see GCBO)

% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)

% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
[fileImagen, pathImagen] = uigetfile(...
   {'*.bmp;*.jpg;*.dxf', '*.bmp;*.jpg;*.dxf Todos los archivos de imagen soportados';
...
   '*.bmp','*.bmp Archivo de mapa de bits'; ...
   '*.jpg','*.jpg Archivo JPEG'; ...
   '*.dxf','*.dxf CAD Drawing Exchange File'}, ...
   'Seleccionar Imagen de la Cavidad');
[pathstr, name, ext] = fileparts( [pathImagen fileImagen] );
if( strcmpi(ext,'.dxf') )
   arg = [ '.\Acme\AcmeCADConverter.exe /r /e /ls /p 1 /f 1 ' "" fullfile(pathImagen,
fileImagen) "" ];
   convertir = dos(arg);
   if convertir==0
      fileImagen = [name '.bmp'];
      im = double(imread(fullfile(pathImagen, fileImagen),'bmp'));
      im = im(:,:,1)>0;
      imt = ones(size(im)+20);
      imt(10:size(im,1)+9,10:size(im,2)+9)=im;
      im = imt;
```

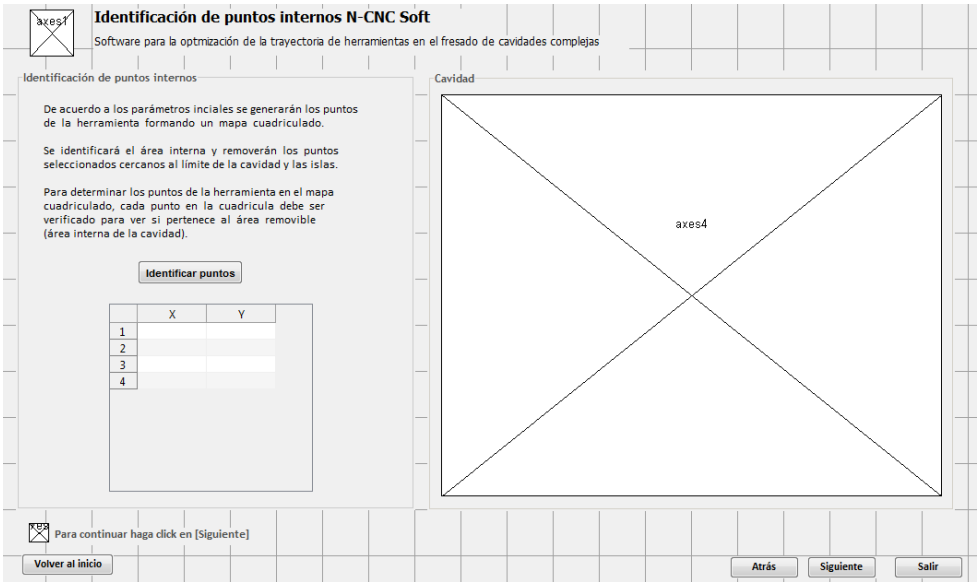
```
        borde = edge(im,'canny');
        [X,Y] = find(borde==1);
        minx = min(X);     miny = min(Y);
        maxx = max(X);     maxy = max(Y);
        im = im(minx:maxx,miny:maxy);
        im = im*255;
        imagen(:,:,1) = im;
        imagen(:,:,2) = im;
        imagen(:,:,3) = im;
        imwrite(imagen,'i_Model.bmp','bmp');
        fileImagen = 'i_Model.bmp';
        imagen = double(imread(fullfile(pathImagen, fileImagen),'bmp'));
    end
else
    if( strcmpi(ext,'.bmp') )
        imagen = double(imread(fullfile(pathImagen, fileImagen),'bmp'));
    else
        imagen = double(imread(fullfile(pathImagen, fileImagen),'jpg'));
    end
end

assignin('base','imagen',imagen);
axes(handles.axes4);
imshow(imagen);
set(handles.txtRes,'string',sprintf('Resolución = [ %d x %d ] pixeles',...
    size(imagen,1),size(imagen,2)));

if (size(imagen,1)>500 || size(imagen,2)> 500)
    assignin('base','isHighRes', 1);
    imtool(imagen);
else
    assignin('base','isHighRes', 0);
end

% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)

% --- Executes during object creation, after setting all properties.
function axes4_CreateFcn(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.
function axes2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes2 (see GCBO)
```

```
% --- Executes during object creation, after setting all properties.
function axes3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes3 (see GCBO)

% --- Executes during object creation, after setting all properties.
function txtRes_CreateFcn(hObject, eventdata, handles)
% hObject    handle to txtRes (see GCBO)

function txtProfundidad_Callback(hObject, eventdata, handles)
% hObject    handle to txtProfundidad (see GCBO)
% Hints: get(hObject,'String') returns contents of txtProfundidad as text
%        str2double(get(hObject,'String')) returns contents of txtProfundidad as a
double

% --- Executes during object creation, after setting all properties.
function txtProfundidad_CreateFcn(hObject, eventdata, handles)

% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton8 (see GCBO)

% --- Executes on selection change in sizeBroca.
function sizeBroca_Callback(hObject, eventdata, handles)
% hObject    handle to sizeBroca (see GCBO)

% Hints: contents = cellstr(get(hObject,'String')) returns sizeBroca contents as cell
array
%        contents{get(hObject,'Value')} returns selected item from sizeBroca
v = get(hObject,'Value');
w = get(handles.popupmenu1,'Value');
if w==1
   U_FACTOR = 2.54075;
else
   U_FACTOR = 1;
end
switch v
   case 1 %1/4
   set(handles.txtRadioFresa,'string',0.25*U_FACTOR);
   case 2 %1/2
   set(handles.txtRadioFresa,'string', 0.5*U_FACTOR);
   case 3 %3/8
   set(handles.txtRadioFresa,'string', (3/8)*U_FACTOR);
   case 4 % 1
   set(handles.txtRadioFresa,'string', 1*U_FACTOR);
```

```
end
set(handles.porcentaje,'string','');
set(handles.txtSigma,'string','');
set(handles.txtG,'string','');

 % --- Executes during object creation, after setting all properties.
function sizeBroca_CreateFcn(hObject, eventdata, handles)
% hObject    handle to sizeBroca (see GCBO)

function porcentaje_Callback(hObject, eventdata, handles)
% hObject    handle to porcentaje (see GCBO)

% Hints: get(hObject,'String') returns contents of porcentaje as text
%        str2double(get(hObject,'String')) returns contents of porcentaje as a double
r =  str2double(get(handles.txtRadioFresa,'string'));
porcentaje = str2double(get(handles.porcentaje,'string'));
set(handles.txtSigma,'string',r*porcentaje/100);
set(handles.txtG,'string',r-(r*porcentaje/100));

% --- Executes during object creation, after setting all properties.
function porcentaje_CreateFcn(hObject, eventdata, handles)
% hObject    handle to porcentaje (see GCBO)
```

Anexo B. CODIGO DE PROGRAMACION PASO 2

Figura 134. Archivo Step2.fig



Fuente: autores.

Código Step2.m

```
function varargout = Step2(varargin)
% End initialization code - DO NOT EDIT

 % --- Executes just before Step2 is made visible.
function Step2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.

% Choose default command line output for Step2
handles.output = hObject;
scrsz = get(0,'ScreenSize');
pos_act = get(gcf, 'Position');
xr = scrsz(3) - pos_act(3);
xp = round(xr/2);
yr = scrsz(4) - pos_act(4);
yp = round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);

% Update handles structure

axes(handles.axes1);
img = imread('imagenes/3620-fresadora-8-mm.jpg');
```

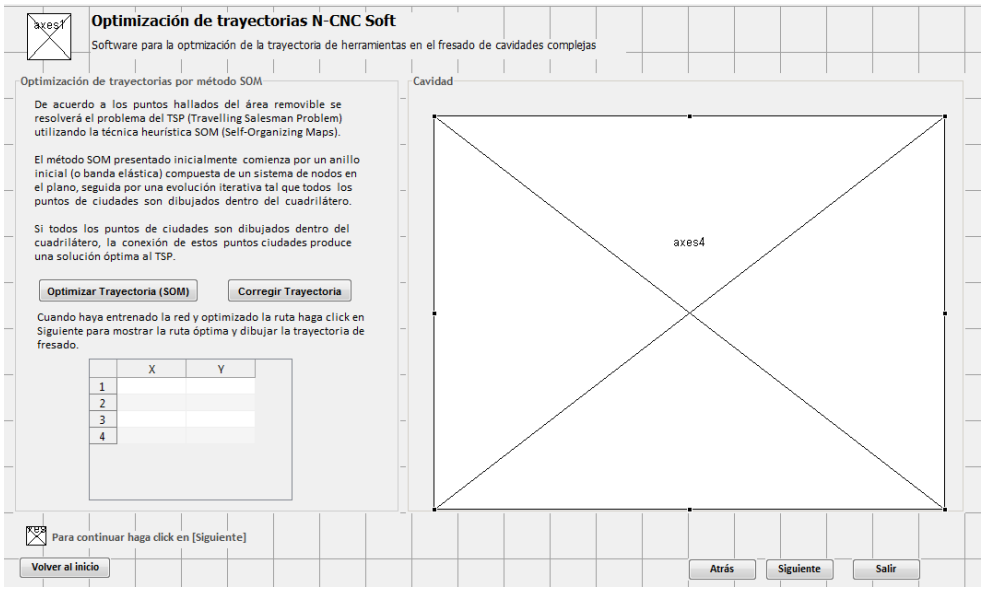
```
imshow(img);
axes(handles.axes3);
fig = imread('imagenes/next.jpg');
imshow(fig);
axes(handles.axes4); cla;
if(evalin('base','isHighRes')==0)
    imshow(evalin('base', 'imagen'));
else
    set(handles.axes4,'visible','off');
end
guidata(hObject, handles);
% UIWAIT makes Step2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = Step2_OutputFcn(hObject, eventdata, handles)

% --- Executes on button press in pushbuttonBack.
function pushbuttonBack_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonBack (see GCBO)
ocultarVentana('h2');
mostrarVentana('h1');

% --- Executes on button press in pushbuttonNext.
function pushbuttonNext_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonNext (see GCBO)
ocultarVentana('h2');
h3 = Step3;
assignin('base','h3',h3);

% --- Executes on button press in pushbuttonExit.
function pushbuttonExit_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonExit (see GCBO)
choice = questdlg('¿Realmente desea salir?','Salir','Si','No','No');
switch choice
    case 'Si'
    clear all;
    clc;
    close;
end

% --- Executes on button press in pushbuttonHome.
function pushbuttonHome_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonHome (see GCBO)
tructure with handles and user data (see GUIDATA)
```

```
ocultarVentana('h2');
h1 = Step1;

% --- Executes during object creation, after setting all properties.
function pushbuttonExit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to pushbuttonExit (see GCBO)


% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)

% Hint: place code in OpeningFcn to populate axes1

% --- Executes during object creation, after setting all properties.
function axes4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes4 (see GCBO)

% Hint: place code in OpeningFcn to populate axes4

% --- Executes during object creation, after setting all properties.
function axes3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes3 (see GCBO)

% Hint: place code in OpeningFcn to populate axes3

% --- Executes during object creation, after setting all properties.
function txtRes_CreateFcn(hObject, eventdata, handles)
% hObject    handle to txtRes (see GCBO)

% --- Executes on button press in pushbuttonIdentificar.
function pushbuttonIdentificar_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonIdentificar (see GCBO)
radioFresa = evalin('base','radioFresa');
g = evalin('base','g');
imagen = evalin('base','imagen');
axes(handles.axes4); cla;
isHighRes = evalin('base', 'isHighRes');
if isHighRes == 1
   imtool(imagen);
   generarRejillaBig(radioFresa,g,imagen, imgca);
else
   generarRejilla(radioFresa,g,imagen);
end
```

```
U_FACTOR = evalin('base','U_FACTOR');
p = evalin('base','p')*U_FACTOR;
set(handles.uitable1,'data',p);

% --- Executes during object creation, after setting all properties.
function uitable1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to uitable1 (see GCBO)
```

Anexo C. CODIGO DE PROGRAMACION PASO 3

Figura 135. Archivo Step3.fig



Fuente: autores.

Código Step3.m

```
function varargout = Step3(varargin)
% End initialization code - DO NOT EDIT

% --- Executes just before Step3 is made visible.
function Step3_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.

% Choose default command line output for Step3
handles.output = hObject;
scrsz = get(0,'ScreenSize');
pos_act = get(gcf, 'Position');
xr = scrsz(3) - pos_act(3);
xp = round(xr/2);
yr = scrsz(4) - pos_act(4);
yp = round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);

% Update handles structure

axes(handles.axes1);
img = imread('imagenes/3620-fresadora-8-mm.jpg');
```

```
imshow(img);

axes(handles.axes3);
fig = imread('imagenes/next.jpg');
imshow(fig);

imtool close all;
axes(handles.axes4); cla;
set(handles.axes4, 'visible','off');
imagen = evalin('base','imagen');
radioFresa = evalin('base', 'radioFresa');
fig = evalin('base','rejilla');
isHighRes = evalin('base','isHighRes');
if isHighRes == 0
    cla;
    set(handles.axes4, 'visible','on');
    imshow(imagen(:,:,1)); hold on;
    plot(fig(:,2),fig(:,1),'r.');
    plot(fig(:,2),fig(:,1),'go','MarkerSize',12*fix(2*radioFresa)/16);
else
    imtool(imagen);
    assignin('base', 'hIma', imgca);
    hold(imgca,'on');
    plot(imgca, fig(:,2),fig(:,1),'go','MarkerSize',12*fix(2*radioFresa)/16);
    plot(imgca, fig(:,2),fig(:,1),'r.');
end

guidata(hObject, handles);
% UIWAIT makes Step3 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = Step3_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);


% --- Executes on button press in pushbuttonBack.
function pushbuttonBack_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonBack (see GCBO)
ocultarVentana('h3');
mostrarVentana('h2');

% --- Executes on button press in pushbuttonNext.
function pushbuttonNext_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonNext (see GCBO)
```

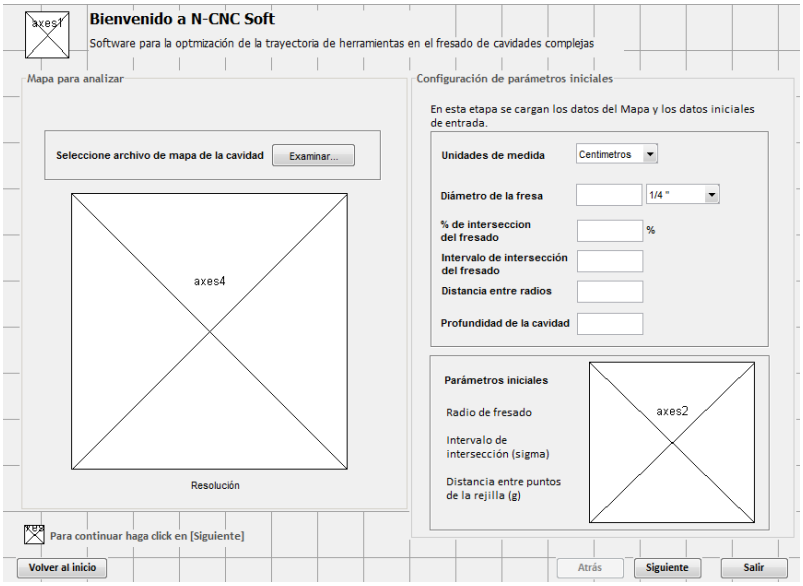
```
ocultarVentana('h3');
h4 = Step4;
assignin('base','h4',h4);

% --- Executes on button press in pushbuttonExit.
function pushbuttonExit_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonExit (see GCBO)
choice = questdlg('¿Realmente desea salir?','Salir','Si','No','No');
switch choice
    case 'Si'
    clear all;
    clc;
    close;
end

% --- Executes on button press in pushbuttonHome.
function pushbuttonHome_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonHome (see GCBO)
ocultarVentana('h3');
h1 = Step1;

% --- Executes during object creation, after setting all properties.
function pushbuttonExit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to pushbuttonExit (see GCBO)

% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)

% Hint: place code in OpeningFcn to populate axes1

% --- Executes during object creation, after setting all properties.
function axes4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes4 (see GCBO)

% --- Executes during object creation, after setting all properties.
function axes3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes3 (see GCBO)
% Hint: place code in OpeningFcn to populate axes3

% --- Executes during object creation, after setting all properties.
function txtRes_CreateFcn(hObject, eventdata, handles)
% hObject    handle to txtRes (see GCBO)

% --- Executes on button press in pushbuttonIdentificar.
```

```
function pushbuttonIdentificar_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonIdentificar (see GCBO)
p = evalin('base','p');
U_FACTOR = evalin('base','U_FACTOR');
imagenOriginal = evalin('base','imagen');
radioFresa = evalin('base','radioFresa');
imtool close all;
axes(handles.axes4);
hA = gca;
if (evalin('base', 'isHighRes')==0)
  [ruta, indices] = solveTSP(p, true, gca, imagenOriginal, radioFresa, 1);
  hI = hA;
else
  imtool(imagenOriginal);
  hI = imgca;
  [ruta, indices] = solveTSPBig(p, true, hI, hA, imagenOriginal, radioFresa, 1);
end
assignin('base','ruta', ruta);
assignin('base','indices', indices);
set(handles.uitable1,'data',ruta*U_FACTOR);
set(handles.uitable1,'Enable','on');
set(handles.pushbuttonIdentificar,'Enable','on');
mask = correccionTrayectoriaSOM(ruta,imagenOriginal,size(ruta,1),hI);
assignin('base','mask', mask);

    % --- Executes on button press in pushbuttonEntrenar.
function pushbuttonEntrenar_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonEntrenar (see GCBO)
axes(handles.axes4);
hA = gca;
axis off;
p = evalin('base','p');
U_FACTOR = evalin('base', 'U_FACTOR');
radioFresa = evalin('base','radioFresa');
imagenOriginal = evalin('base','imagen');
set(handles.uitable1,'data',p*U_FACTOR);
set(handles.uitable1,'Enable','on');
if evalin('base','isHighRes') == 1
    w                                                                                                    =
optimizarTrayectoriaSOMBig(p,radioFresa,(imagenOriginal(:,:,1)==255),evalin('bas
e','hIma'), hA);
else
    w =  optimizarTrayectoriaSOM(p,radioFresa,(imagenOriginal(:,:,1)==255));
end
assignin('base','w',w);
```

```
set(handles.uitable1,'data',w*U_FACTOR);
set(handles.uitable1,'Enable','on');
set(handles.pushbuttonIdentificar,'Enable','on');

% --- Executes during object creation, after setting all properties.
function uitable1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to uitable1 (see GCBO)
```

Anexo D. CODIGO DE PROGRAMACION PASO 4

Figura 136. Archivo Step4.fig



Código Step4.m

```
function varargout = Step4(varargin)
% End initialization code - DO NOT EDIT

 % --- Executes just before Step4 is made visible.
function Step4_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.

% Choose default command line output for Step4
handles.output = hObject;
scrsz = get(0,'ScreenSize');
pos_act = get(gcf, 'Position');
xr = scrsz(3) - pos_act(3);
xp = round(xr/2);
yr = scrsz(4) - pos_act(4);
yp = round(yr/2);
set(gcf,'Position',[xp yp pos_act(3) pos_act(4)]);

% Update handles structure
axes(handles.axes1);
img = imread('imagenes/3620-fresadora-8-mm.jpg');
imshow(img);
imtool close all;
```

```
axes(handles.axes4); cla;
set(handles.axes4, 'visible', 'off');
if(evalin('base','isHighRes')==0)
    fig = evalin('base','imagen');
    imshow(fig(:,:,1)); hold on;
    radioFresa = evalin('base', 'radioFresa');
    fig = evalin('base','rejilla');
    plot(fig(:,2),fig(:,1),'r.');
    plot(fig(:,2),fig(:,1),'go','MarkerSize',12*fix(2*radioFresa)/16);
end

axes(handles.axes3);
fig = imread('imagenes/next.jpg');
imshow(fig);

guidata(hObject, handles);
% UIWAIT makes Step4 wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = Step4_OutputFcn(hObject, eventdata, handles)

% --- Executes on button press in pushbuttonBack.
function pushbuttonBack_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonBack (see GCBO)
ocultarVentana('h4');
mostrarVentana('h3');

% --- Executes on button press in pushbuttonNext.
function pushbuttonNext_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonNext (see GCBO)
figure;
grid on;
ruta = evalin('base', 'ruta');
indice = evalin('base', 'indices');
radioFresa = evalin('base', 'radioFresa');
imagen = evalin('base','imagen');
prof = evalin('base','profundidad');
mask = evalin('base', 'mask');
testGraficar(imagen, ruta, indice, mask, radioFresa,prof);
fid = fopen('puntos.txt', 'w');
fprintf(fid, '%.2f \t %.2f \n\r\n', ruta);
fclose(fid);dos('start puntos.txt');

 % --- Executes on button press in pushbuttonExit.
function pushbuttonExit_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to pushbuttonExit (see GCBO)
choice = questdlg('¿Realmente desea salir?','Salir','Si','No','No');
switch choice
    case 'Si'
    clear all;
    clc;
    close;
end



% --- Executes on button press in pushbuttonHome.
function pushbuttonHome_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonHome (see GCBO)
ocultarVentana('h4');
h1 = Step1;

% --- Executes during object creation, after setting all properties.
function pushbuttonExit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to pushbuttonExit (see GCBO)

% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% Hint: place code in OpeningFcn to populate axes1

 % --- Executes during object creation, after setting all properties.
function axes4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes4 (see GCBO)
% Hint: place code in OpeningFcn to populate axes4

% --- Executes during object creation, after setting all properties.
function axes3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes3 (see GCBO)
% Hint: place code in OpeningFcn to populate axes3

 % --- Executes during object creation, after setting all properties.
function txtRes_CreateFcn(hObject, eventdata, handles)
% hObject    handle to txtRes (see GCBO)

% --- Executes on button press in pushbuttonIdentificar.
function pushbuttonIdentificar_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonIdentificar (see GCBO)
axes(handles.axes4);
if (evalin('base', 'isHighRes') ==0)
```

```
    hI = gca;
else
    imtool(evalin('base', 'imagen'));
    hI = imgca;
    hold(hI,'on');
    radioFresa = evalin('base', 'radioFresa');
    fig = evalin('base','rejilla');
    plot(hI,fig(:,2),fig(:,1),'r.');
    plot(hI,fig(:,2),fig(:,1),'go','MarkerSize',12*fix(2*radioFresa)/16);
end
ruta = evalin('base','ruta');
indices = evalin('base','indices');
radioFresa = evalin('base','radioFresa');
mask = evalin('base','mask');
hold(hI,'on');
plot(hI,ruta(1,2), ruta(1,1), 'r.', 'MarkerSize',20);
for i=1:size(indices,1)-1
    if(mask(i)==1)
    hold(hI,'on');
    P1 = [ ruta(i,1) ruta(i+1,1) ];
    P2 = [ ruta(i,2) ruta(i+1,2) ];
    plot(hI, P2,P1,'y-', 'LineWidth',12*fix(radioFresa)/16);
    if(rem(i,4)==0)
        pause(0.05);
    end
    end
end
plot(hI,ruta(1,2), ruta(1,1), 'r.', 'MarkerSize',25);
plot(hI,ruta(end,2), ruta(end,1), 'b.', 'MarkerSize',25);
for i=1:size(indices,1)-1
    if(mask(i)==1)
        hold(hI,'on');
        P1 = [ ruta(i,1) ruta(i+1,1) ];
        P2 = [ ruta(i,2) ruta(i+1,2) ];
        plot(hI,P2,P1,'k-');
    end
end

 % --- Executes on button press in pushbuttonEntrenar.
function pushbuttonEntrenar_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonEntrenar (see GCBO)
p = evalin('base','p');
imagenOriginal = evalin('base','imagen');
radioFresa = evalin('base','radioFresa');
axes(handles.axes4);
```

```
if (evalin('base','isHighRes')==0)
    hI = gca;
else
    imtool(evalin('base', 'imagen'));
    hI = imgca;
    hold(hI,'on');
    radioFresa = evalin('base', 'radioFresa');
    fig = evalin('base','rejilla');
    plot(hI,fig(:,2),fig(:,1),'r.');
    plot(hI,fig(:,2),fig(:,1),'go','MarkerSize',12*fix(2*radioFresa)/16);
end
ruta= evalin('base','ruta');
mask= evalin('base','mask');
indices = evalin('base','indices');
for i=1:size(indices,1)-1
    if(mask(i)==1)
    hold(hI,'on');
    P1 = [ ruta(i,1) ruta(i+1,1) ];
    P2 = [ ruta(i,2) ruta(i+1,2) ];
    plot(hI, P2,P1,'k-');
    end
end

U_FACTOR = evalin('base','U_FACTOR');
set(handles.uitable1,'data',ruta*U_FACTOR);
set(handles.uitable1,'Enable','on');
set(handles.pushbuttonIdentificar,'Enable','on');
```