Universidad Michoacana
de San Nicolás de Hidalgo

División de Estudios de Posgrado de la Facultad de Ingeniería Eléctrica

DESARROLLO DE UNA MÁQUINA DE CONTROL NUMÉRICO COMPUTARIZADO Y SU APLICACIÓN PARA LA MANUFACTURA DE PIEZAS A PARTIR DE DIBUJOS

TESIS

Que para obtener el grado de

Maestro en Ciencias en Ingeniería Eléctrica

Presenta:

Mario Antonio Santana Gómez

Director de tesis:

Dr. Leonardo Romero Muñoz

Morelia, Michoacán
Agosto de 2014

DESARROLLO DE UNA MÁQUINA DE CONTROL NUMÉRICO COMPUTARIZADO Y SU APLICACIÓN PARA LA MANUFACTURA DE PIEZAS A PARTIR DE DIBUJOS

Los Miembros del Jurado de Examen de Grado aprueban
la **Tesis de Maestría en Ciencias en Ingeniería Eléctrica** de ***Mario Antonio Santana Gómez***

Dr. Juan José Flores Romero
Presidente del Jurado

Dr. Leonardo Romero Muñoz
Director de Tesis

Dr. José Antonio Camarena Ibarrola
Vocal

Dr. Félix Calderón Solorio
Vocal

Dr. Ignacio Juárez Campos
Revisor Externo

Dr. J. Aurelio Medina Rios
Jefe de la División de Estudios de Posgrado
de la Facultad de Ingeniería Eléctrica. UMSNH
(Por reconocimiento de firmas).

UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO
Junio 2014

Agradecimientos

Deseo dar las gracias a nuestro Creador, todo viene de Dios.

Quiero expresar mi gratitud al Dr. Leonardo Romero por su constante guía, consejos y apoyo. Por su asombrosa conciencia de que lo que hacemos no es para obtener un título o cumplir un requisito, sino que, lo que hacemos es para que le sirva a otras personas. Gracias por todo...

Gracias a mis padres, María de Jesús Gómez y Ángel Santana, un agradecimiento infinito por su amor y apoyo incondicional. Gracias por darme desde muy pequeño una gran visión que me motiva a llegar a ser lo mejor que puedo ser.

A mis queridos hermanos Ángel, José de Jesús, Ramiro, Julia, Aurora, Aurelia, Alfredo y Elizabeth; mi relación con ustedes es muy importante. Gracias a Elizabeth con quien he convivido más, por darme herramientas para elevar mi vuelo.

Gracias hermanos y hermanas por su apoyo en toda mi vida.

Agradezco a Cuauhtémoc Gómez y Alejandro Santana, mis primos más cercanos, por su constante apoyo y amistad.

¡Gracias a toda mi familia!

Valoro el apoyo que recibí por parte de Moisés García, Antonio Ramos, Roberto Tapia, Marco Antonio Pacheco, Elisa López, Benjamín Magaña, Cuitláhuac Huerta, Vicente Sierra y Armando Arroyo. Gracias.

Quiero dar las gracias a la Universidad Michoacana y al CONACYT por financiar este trabajo.

Mil gracias a todos.

Resumen

Las máquinas de Control Numérico Computarizado (CNC) comerciales son muy caras y hacen que los usuarios se vuelvan dependientes de los fabricantes ya que usan hardware y protocolos propietarios; y el software disponible normalmente es cerrado y adaptado específicamente a la máquina en cuestión; lo cual limita desarrollos de nuevas aplicaciones por parte de sus usuarios.

Por tanto las empresas que decidan adquirir una máquina CNC comercial realizarán una inversión inicial muy costosa, tendrán que contratar un servicio de mantenimiento especializado, para esa máquina específica, que sólo lo ofrece el fabricante.

Estos problemas nos motivaron a desarrollar maquinas CNC de tres y cuatro grados de libertad, de bajo costo y con arquitectura abierta que pueden aplicarse en la fabricación de piezas de madera, como guitarras, muebles, figuras, etc., así como en otras posibles aplicaciones que usen materiales suaves.

Tomamos como reto generar la programación de las máquinas CNC a partir de un dibujo. Los programas disponibles para esta tarea: image–to–gcode, Potrace, Illustrator y pstoedit, presentaron ciertas deficiencias. Para corregirlas, se desarrolló una nueva aplicación llamada trace2Gcode que genera código G (el lenguaje de programación para las máquinas CNC) a partir de una imagen, partiendo de los trazos en un dibujo.

Adicionalmente, se implementó una aplicación para permitir la entrada mediante la fotografía tomada por una cámara digital, corrigiendo la distorsión proyectiva y dejando como salida un archivo listo para la aplicación trace2Gcode.

En los resultados trace2Gcode logra sobresalir de Potrace e Illustrator en número de líneas de código y en tiempo. Se logra ofrecer a las personas, por ejemplo artesanos que no tienen una preparación especial para aprovechar las máquinas CNC, un medio sencillo y fácil de programar máquinas CNC para que hagan lo que ellos desean.

Palabras clave: máquinas de Control Numérico Computarizado (CNC), código G, aplicación para la manufactura, mecanizado a partir de dibujos.

Abstract

The Commercial machines Computer Numerical Control (CNC) are very expensive and make users become dependent on manufacturers and using hardware and proprietary protocols; and software available is normally closed and specifically tailored to the machine in question; limiting development of new applications by users.

Therefore companies that decide to purchase a commercial CNC machine will be an expensive initial investment, will have to hire a maintenance service specialized for that specific machine, which only offers the manufacturer.

These problems motivated us to develop machines CNC three and four degrees of freedom, low cost and open architecture that can be applied in the manufacture of pieces of wood, guitars, furniture, figures, etc., as well as other possible applications using soft materials.

We take the challenge to generate the programming of CNC machines from a drawing. Programs available for this task: image-to-gcode, Potrace, Illustrator and pstoedit presented certain deficiencies. To correct, a new application called G trace2Gcode generated code (the programming language for CNC machines) from an image, starting with the strokes in a drawing was developed.

Additionally, an application was implemented to allow entry through photographs taken by a digital camera, editing and leaving the projective distortion output a trace2Gcode ready for application file.

The results achieved trace2Gcode overhang and Illustrator Potrace number of lines of code and time. It manages to offer people, for example artisans who have special training to take advantage of CNC machines, a simple and easy way to program CNC machines to do what they want.

Keywords: machines Computer Numerical Control (CNC), G code, application for manufacturing, machining from drawings.

Contenido

Lista de Figuras

Lista de Tablas

Publicaciones

Leonardo Romero, Mario Santana. Maquina de control numérico computarizado de 4 grados de libertad para realizar trabajos sobre madera y otros materiales suaves. Concurso de prototipos de desarrollo tecnológico en la 20va. Semana Nacional de Ciencia y Tecnología. 2013.

Leonardo Romero, M. S., Moises García. Development of a computer numerically controlled router machine with 4 degrees of freedom using an open architecture. Power, Electronics and Computing (ROPEC), 2013 IEEE International Autumn Meeting on. 2013.

Capítulo 1

Introducción

Los robots se han utilizado con éxito para hacer varias tareas, incluyendo: la industria automotriz, la industria eléctrica, la industria electrónica, productos de metal y muchos otros.

Proponemos colaborar en la introducción y aprovechamiento de los Robots manipuladores en México y particularmente en Michoacán, desarrollando una máquina de Control Numérico Computarizado (CNC) de bajo costo y de arquitectura abierta que pueda aplicarse en la fabricación de piezas de madera, como guitarras, muebles, figuras, etc. Consideramos que se tiene un potencial muy grande como lograr el incremento de la productividad de los fabricantes de guitarras de Paracho, Mich., así como en otras posibles aplicaciones que usen materiales suaves como en las fábricas de muebles de madera, zapatos, cárteles publicitarios, tarjetas de circuitos impresos (PCB) o procesos con materiales delicados que requieren mayor precisión como el vidrio, etc.

El diseño e implementación de máquinas CNC es un trabajo que involucra las áreas de ingeniería eléctrica, electrónica, mecánica y computación, entre otras. Las máquinas CNC desarrolladas son susceptibles de comercializarse y aplicarse en gran parte de las empresas del ramo de la madera.

1.1. Definición del Problema

Muchas de las tareas que realizan los talleres que trabajan con madera o materiales suaves, pueden automatizarse con una máquina CNC de 3 Grados De Libertad (GDL): uno para desplazamientos en el eje X, otro para desplazamientos en el eje Y, otro para desplazamientos en el eje Z. Con este tipo de máquinas se puede realizar con la precisión necesaria tareas tales como: corte de precisión en un plano siguiendo trayectorias predefinidas, perforación, desbaste, tallado de figuras, entre otras.

Las máquinas CNC comerciales son muy caras y hacen que los usuarios se vuelvan dependientes de los fabricantes ya que usan hardware y protocolos propietarios; y el software disponible es cerrado y adaptado específicamente a la máquina en cuestión. Las máquinas CNC comerciales son normalmente cerradas, lo cual limita desarrollos de nuevas aplicaciones por parte de sus usuarios, la expansión de dichas máquinas obliga a que sea realizada por el fabricante.

Por tanto los productores que decidan adquirir una máquina CNC comercial realizarán una inversión inicial muy costosa, tendrán que contratar un servicio de mantenimiento especializado, para esa máquina específica, que sólo lo ofrece el fabricante, contar con ingenieros de diseño que usen software de Manufactura Asistida por Computadora (CAM), Ingeniería Asistida por Computadora (CAE) y Diseño Asistido por Computadora (CAD). En caso de que se dañe un dispositivo sólo se podrá adquirir con el mismo fabricante y las posibles actualizaciones y mejoras que surjan se podrán implementar hasta que el fabricante haga las adecuaciones. Con todo ello, se tiene una dependencia del fabricante, un alto costo de inversión y un uso complejo.

1.2. Justificación

A nivel mundial las máquinas CNC son la base de muchos procesos de fabricación. La mayoría de los robots industriales se utilizan en Asia (en particular, China, Japón y Corea) y Europa. Mientras en países avanzados ya utilizan robots manipuladores en las empresas, en México solamente las grandes empresas pueden acceder y aprovechar es-

ta nueva tecnología, principalmente por el costo inicial y de mantenimiento [International Federation of Robotics, 2013].

Como un ejemplo de lo anterior, en la Figura 1.1 se muestra un robot puliendo una guitarra en una Fábrica de Guitarras en los Estados Unidos y en la Figura 1.2 a un artesano de Paracho Michoacán lijando una parte de la guitarra. Observamos el contraste, por un lado la producción de guitarras con robots y por el otro la fabricación de guitarras sin ninguna herramienta tecnológica avanzada.

Figura 1.1: Fabricación de guitarras en Estados Unidos.

Al trabajar con materiales suaves como madera, plástico y aluminio, la mayoría de los robots industriales cuentan con capacidades que sobrepasan los requerimientos en precisión y robustez. Estas máquinas de automatización comerciales tienen un costo

Figura 1.2: Artesano michoacano lijando una parte de una guitarra.

elevado, poco accesible para los fabricantes que usan materiales suaves. Sin embargo, la tecnología es la base del crecimiento económico en el corto y principalmente en el largo plazo [Vergara Reyes, 2009], toda fábrica que no se modernice y use las nuevas tecnologías quedará sin posibilidades de ser rentable y mantenerse en el mercado.

Otro inconveniente con el que se enfrentan los fabricantes que usan materiales suaves, es que al querer incorporar tecnología en sus procesos, el precio del software es excesivo; además, para usarlo se requieren conocimientos y habilidades que sobrepasan el perfil de un obrero.

Sin embargo, existe una exigencia que ya es común en todo proceso de fabricación, los tiempos reducidos de entrega. Las máquinas programadas satisfacen esta necesidad ya que pueden trabajar todo el tiempo sin descanso a fin de cumplir cualquier

requerimiento [Horath, 1993].

En el uso de un sistema CNC encontramos las siguientes ventajas [Steve F. Krar, 2001]:

- Mayor precisión y mejor calidad de productos.
- Mayor uniformidad en los productos producidos.
- Un operario puede operar varias máquinas a la vez.
- Fácil procesamiento de productos de apariencia complicada.
- Flexibilidad para el cambio en el diseño y en modelos en un tiempo corto.
- Fácil control de calidad.
- Es posible satisfacer pedidos urgentes.
- Se reduce la fatiga del operador.
- Mayor seguridad en las labores.
- Aumento del tiempo de trabajo por maquinaria.

Los sistemas CNC no se han adaptado y expandido a los pequeños y medianos fabricantes en México, por su cerrada estructura, y debería ser cambiada disminuyendo la dependencia de software y hardware especial. Los sistemas más funcionales tienen una estructura modular y capacidad de reconstrucción rápida en software y hardware para satisfacer el rápido desarrollo de la tecnología, el mercado y la estructura de la organización en el sistema de producción.

Una arquitectura abierta debe tener la capacidad de integrar piezas de equipos de varios fabricantes diferentes y obtener soluciones de control con varias interfaces de aplicaciones programables, manteniendo el mismo rendimiento a menor costo. Si bien hay algunos esfuerzos para definir una arquitectura abierta, para este proyecto se tomó el camino OSEC (Open System Environment for Controller). Arquitectura que se basa en una microcomputadora personal IBM-PC estándar para controlar el equipo de fabricación [Asato et al., 2002].

1.3. Objetivo de la tesis

Desarrollar dos máquinas de CNC, una de 3 Grados De Libertad (GDL) y otra de 4 GDL; y una herramienta de software que permita manejar máquinas CNC de manera fácil y simple, incluso para operadores sin ningún conocimiento en programas para Diseño Asistido por Computadora (CAD, por sus siglas en inglés).

En particular se plantearon los siguientes objetivos:

1. Mostrar la importancia de las máquinas CNC.
2. Desarrollar dos máquinas CNC con 3 y 4 GDL basadas en arquitectura abierta tanto en hardware como en software, para trabajar con materiales suaves.
3. Desarrollar una aplicación de mecanizado a partir de imágenes con trazos en 2 dimensiones. La entrada será un dibujo conteniendo los trazos a realizar y la salida sera u programa en código G (un lenguaje estándar para programar máquinas CNC) que puede ser ejecutado en la máquina CNC.
4. Que este trabajo sirva como una herramienta en el inicio de la investigación e innovación de esta área.

1.4. Organización del documento

En el capítulo 2 se aborda de manera general las dos máquinas CNC, nos introduciremos en los sistemas de control numérico. El capítulo 3 expone la creación de una nueva aplicación que genera código G a partir de una imagen, veremos los programas que existen para resolver este reto, describiremos el proceso de la nueva aplicación y sus parámetros. En el capítulo 4 se presentan los resultados experimentales para la nueva aplicación y un análisis de los resultados. El capítulo 5 contiene las conclusiones y se enumeran los trabajos futuros.

En los apéndices A y B se presenta a detalle el desarrollo de las máquinas CNC de 4 y 3 GDL, sus componentes, el hardware y software utilizado. El apéndice C se describen los fundamentos de la programación en código G de las máquinas CNC, en particular

las instrucciones para realizar tareas de mecanizado en las máquinas de control numérico desarrolladas, incluyendo algunos ejemplos.

En el apéndice D se describe con detalle el método de mínimo cuadrados totales para estimar una línea a partir de un conjunto de puntos.

En el apéndice E se describe los conceptos relacionados con los métodos de mínimos cuadrados y completos, reducido y modificado para estimar una circunferencia a partir de un conjunto de puntos.

Finalmente en el apéndice F se describe los archivos que integran el código fuente de la nueva aplicación desarrollada.

Capítulo 2

Máquinas de Control Numérico Computarizado de tres y cuatro grados de libertad

Los robots industriales se han utilizado con éxito para hacer varias tareas, incluyendo: la industria automotriz, la industria eléctrica / electrónica, productos de metal y muchos otros [International Federation of Robotics, 2012].

La incorporación de tecnología es una variable fundamental para el aumento del crecimiento y desarrollo económico. Es así, que aumentará constantemente en los próximos años la fabricación de robot industriales y la inserción de ellos en los centros de producción, como lo estima la organización International Federation of Robotics (IFR) [International Federation of Robotics, 2012].

En el Posgrado de la Facultad de Ingeniería Eléctrica de la UMSNH se integraron dos Máquinas de Control Numérico Computarizado (CNC), la primera con 4 Grados De Libertad (GDL) y la segunda con 3 Grados De Libertad (GDL), por parte de un servidor y mi asesor de tesis. Se pretende incursionar en el desarrollo de máquinas CNC con el objetivo de apoyar los procesos de maquinado en fábricas del estado donde producen piezas que no requieren extrema precisión y usan materiales suaves, como son las fábricas de muebles de madera, guitarras, zapatos, carteles publicitarios, tarjetas de circuitos

impresos (PCB), etc.

La máquina CNC de 4 GDL se encuentra en el Laboratorio de Computación de la División de Estudios de Posgrado de la Facultad de Ingeniería Eléctrica y la máquina CNC de 3 GDL se encuentra en el Laboratorio de Ingeniería Eléctrica de la División de Estudios de Posgrado de la Facultad de Ingeniería Eléctrica, ambas en el edificio Omega 2 de Ciudad Universitaria en Morelia, Michoacán.

2.1. ¿Qué es una máquina CNC?

Básicamente, es una máquina basada en una computadora, que toma una serie de instrucciones y los convierte en movimiento. Una máquina CNC tiene una herramienta de trabajo (punta de corte, router, cortador de plasma, láser, etc.) que hará el mecanizado sobre la pieza hasta que se produzca el resultado deseado.

En la actualidad, las máquinas CNC han llegado a ser la base de muchos procesos industriales. Las máquinas CNC incluyen robots, líneas de producción y todas aquellas máquinas que son controladas por dispositivos digitales. Pueden variar en tamaño, características, aptitud para determinado trabajo, etc., pero tienen un común denominador: sus ejes principales son los ejes X e Y [Smid, 2008].

Gracias a estas máquinas se logra satisfacer las exigencias de producción, a través de la automatización de procesos manufactura de las materias primas en los diversos objetos que usamos a diario como: ropa, zapatos, automóviles, aviones, mesas, sillas, puertas, etc.

Las máquinas CNC eliminan la posibilidad de errores humanos en la manufactura ya que la operación es controlada por un programa. Los siguientes puntos resumen cómo trabaja una máquina CNC [Ramírez, 2008].

- El programa es introducido al sistema mediante algún dispositivo de almacenamiento o directamente en la computadora.
- El programa es interpretado y se convierten los datos a señales de control, en la manera que los motores encargados del desplazamiento puedan entender.

- Los mecanismos de la máquina convierten las señales de control a los movimientos requeridos.

2.2. Desarrollo de una máquina CNC de cuatro grados de libertad

Diseñamos una máquina de CNC con cuatro grados de libertad usando componentes de la marca Longs Motor. La Tabla 2.1 muestra las características de la máquina. Mientras que el apéndice A contiene los detalles de construcción de esta máquina.

Eje	Desplazamiento	Resolución	Velocidad
X	16”	0.0005”	1”/s
Y	8”	0.0005”	1”/s
Z	2.5”	0.0005”	1”/s
A	–	0.45^o	$1800^o/s$

Tabla 2.1: Especificaciones de la máquina CNC de 4GDL.

2.2.1. Descripción general de la mesa CNC

Sobre una mesa de acero colocamos toda la infraestructura de nuestra máquina de CNC, una placa que permite usar mordazas para la sujeción de la pieza de trabajo. Como herramienta de corte, un router de la marca Dewalt, modelo DWP611, que gira a una velocidad de 30,000 rev/min; puede usar brocas, cortadoras y fresas de alta velocidad; la herramienta será desplazada por tornillos sinfín, que son movidos por motores de pasos. Además añadimos una aspiradora que se encargará de absorber los materiales derivados de la pieza de trabajo cuando el router esté funcionando.

Esta máquina puede hacer trabajos tipo torno. En la Figura 2.1 se aprecia la máquina y se señalan los cuatro motores que hacen desplazar la herramienta en los cuatro ejes, estos ejes los nombramos convencionalmente X, Y, Z y A.

Los tres ejes principales de la máquina son X, Y y Z. Los dos primeros se asocian al movimiento en el plano horizontal de la mesa de trabajo, mientras que el tercero es el

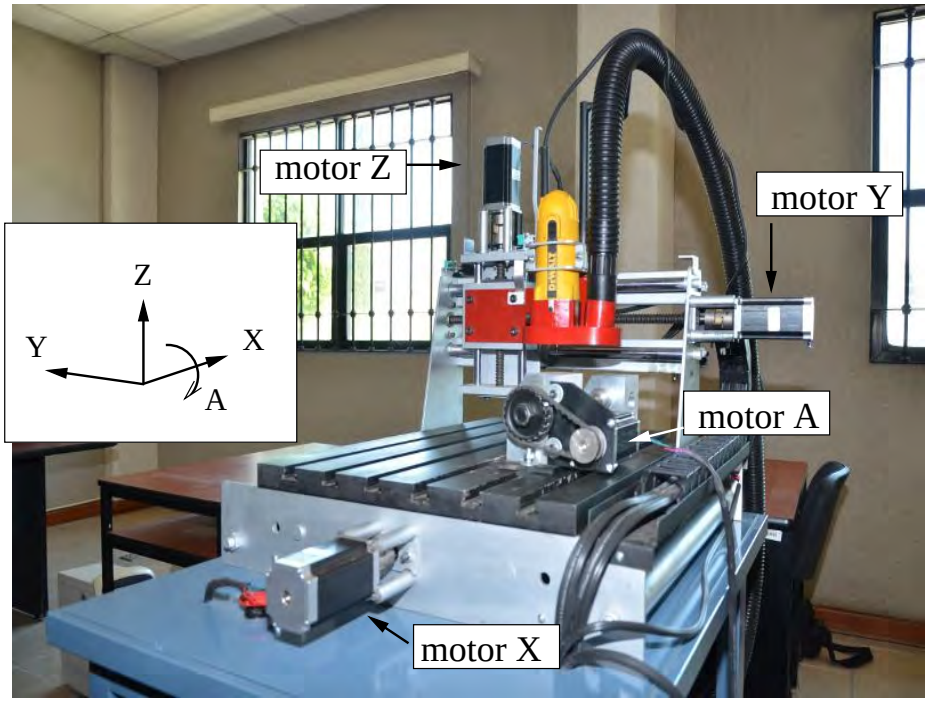


Figura 2.1: Máquina CNC de cuatro grados de libertad.

desplazamiento vertical de la herramienta de la máquina.

El cuarto eje, llamado A, permite hacer trabajos tipo torno, haciendo girar (teniendo como eje de giro el eje X) la pieza de trabajo contra la herramienta cortante y a medida que la herramienta cortante se mueve longitudinal y transversalmente respecto al eje de la pieza de trabajo, se generará la forma deseada de la pieza de trabajo.

En la Figura 2.2 se muestran los dos elementos que sostienen la pieza de trabajo para utilizar el cuarto grado de libertad A. En el lado izquierdo es el acoplamiento del motor y en el lado derecho es un soporte giratorio que permite sostener la pieza de trabajo. También en esta figura se aprecian las magnitudes para la pieza de trabajo en este eje: 14 pulgadas de longitud máxima y 5 pulgadas de diámetro máximo.

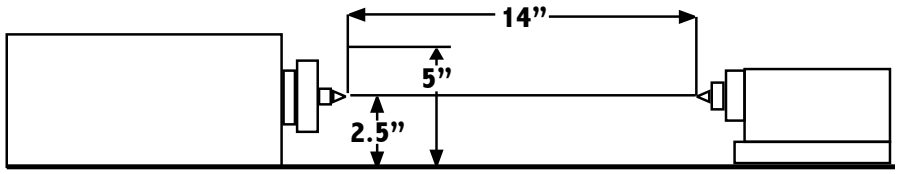


Figura 2.2: Dimensiones para el cuarto eje.

2.3. Desarrollo de una máquina CNC de tres grados de libertad

También diseñamos una segunda máquina de Control Numérico Computarizado (CNC) de sólo tres grados de libertad. Esta segunda máquina la desarrollamos usando dispositivos de la marca Probotix. La Tabla 2.2 muestra las características de la máquina. Mientras que el apéndice B contiene los detalles de construcción de esta máquina.

Eje	Desplazamiento	Resolución	Velocidad
X	16”	0.0005”	1”/s
Y	8”	0.0005”	1”/s
Z	2.5”	0.0005”	1”/s

Tabla 2.2: Especificaciones de la máquina CNC de 3GDL.

2.3.1. Descripción general de la mesa CNC

Esta mesa de CNC es muy similar a la de cuatro grados de libertad, cuenta con su mesa de acero, una placa que permite colocar mordazas para la sujeción de la pieza de trabajo, cuenta con tres motores de pasos, como herramienta de trabajo un router de la marca Dewalt, modelo DWP611 y a su lado una aspiradora.

En la Figura 2.3 se aprecia la máquina y se señalan los tres motores que hacen desplazar la herramienta en los tres ejes, estos ejes los nombramos convencionalmente X, Y y Z.

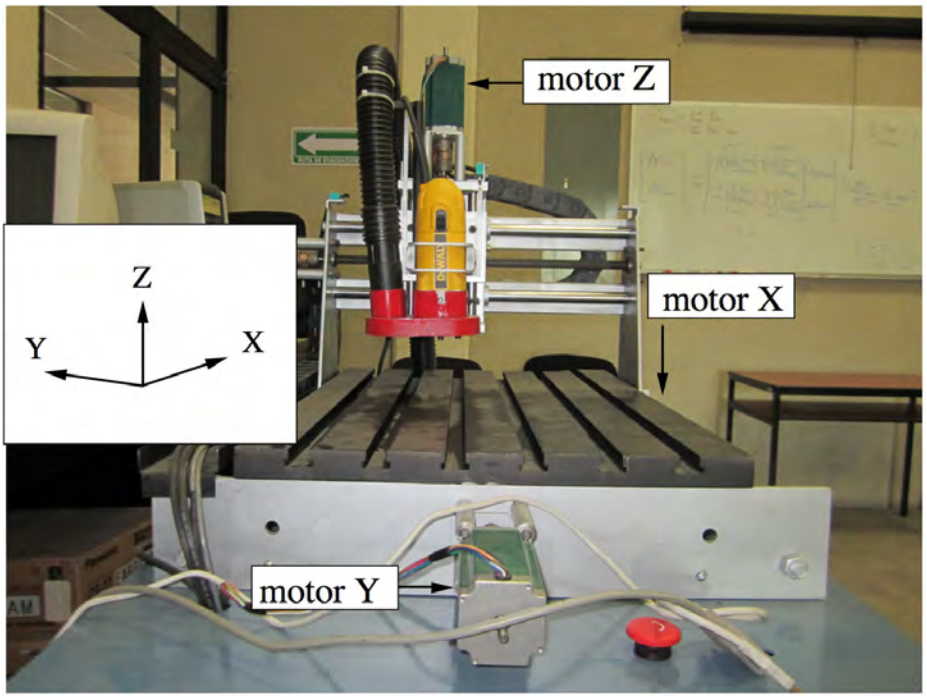


Figura 2.3: Los tres grados de libertad.

2.4. Comparación de las mesas de 3 y 4 GDL

Inicialmente se integró la mesa de 3GDL y después la de 4GDL. Los dispositivos de control de motores de la mesa de 3GDL, de la marca Probotix son poco robustos, siendo necesario limitar la corriente de los motores para no sobrecalentar los componentes eléctricos.

La mesa de 4GDL superó este problema de calentamiento al utilizar controladores más robustos para los motores de pasos. Adicionalmente se diseño con un cuarto grado de libertad para expandir el tipo de trabajos realizados por la máquina.

En ambos casos, la parte mecanica de estas máquinas fue construida por Rafael Cardiel Cervantes, una persona muy habil en cuestiones mecánicas, que tiene su taller en Paracho, Michoacán.

Capítulo 3

Diseño de una nueva aplicación: trace2Gcode

La metodología tradicional para el trabajo con máquinas CNC tiene la estructura mostrada en la Figura 3.1, primero se usa un software CAD (Diseño Asistido por Computadora) para el diseño en alto nivel de la pieza a manufacturar, en seguida se convierte el diseño CAD a código G con un software CAM (Manufactura Asistida por Computadora) y un software CNC que convierte el código en movimientos de la máquina [Smid, 2008]; estos pasos son realizados por expertos, son programas de alto nivel que requieren un entrenamiento especial previo. Ejemplos de programas comerciales son: AutoCAD, SolidWorks, Mastercam, entre otros.

Figura 3.1: Proceso de trabajo tradicional con máquinas CNC.

Platicando con artesanos que trabajan con herramientas para madera, expresaban su interés en que un dibujo en una hoja de papel pudiera llevarse automáticamente a una máquina, sin tener que realizar el aprendizaje previo de un lenguaje de programación o de un programa de cómputo especializado.

En este capítulo presentamos la construcción de un software que permite hacer el proceso de diseño simple, partiendo de una fotografía de un dibujo y entregando como salida un programa escrito en lenguaje de código G para ejecutarlo en la máquina CNC.

Con esta aplicación obtendremos el código G necesario para la programación de máquinas CNC, a partir de un dibujo de trazos realizado por una persona, sin la necesidad de usar paquetes de software especializados y costosos; evitando también el tiempo de aprendizaje y familiarización con los paquetes CAD/CAM. Esta aplicación está pensada en particular para que pueda ser usada por artesanos que no tienen conocimiento previos de CAD/CAM o que incluso se les dificulta el utilizar la computadora.

3.1. Programas disponibles

Antes de abordar el desarrollo de la nueva aplicación, revisamos los programas disponibles que pudrían realizar la tarea propuesta: convertir los trazos de una imagen a un programa en código G.

Encontramos los siguientes programas: image–to–gcode, Potrace, Illustrator y pstoedit:

1. image-to-gcode: este programa libre entrega como resultado movimientos de la máquina CNC en caminos de izquierda a derecha y de arriba a abajo, similar a la operación de una impresora (y por lo tanto se necesita mucho más tiempo para terminar la tarea), en la Figura 3.2 se muestra la interfaz de este programa y se notan las trayectorias tipo impresora.

2. Potrace: al vectorizar una imagen este programa libre, hace un recorrido del contorno de cada trazo con sólo líneas y genera múltiples vectores por cada trazo; estos problemas se pueden observar en la Figura 3.4 en el acercamiento, donde se ven múltiples líneas, esta imagen es el resultado de tomar como imagen de entrada mostrada en la Figura 3.3b. La salida de este programa es un archivo gráfico vectorial.

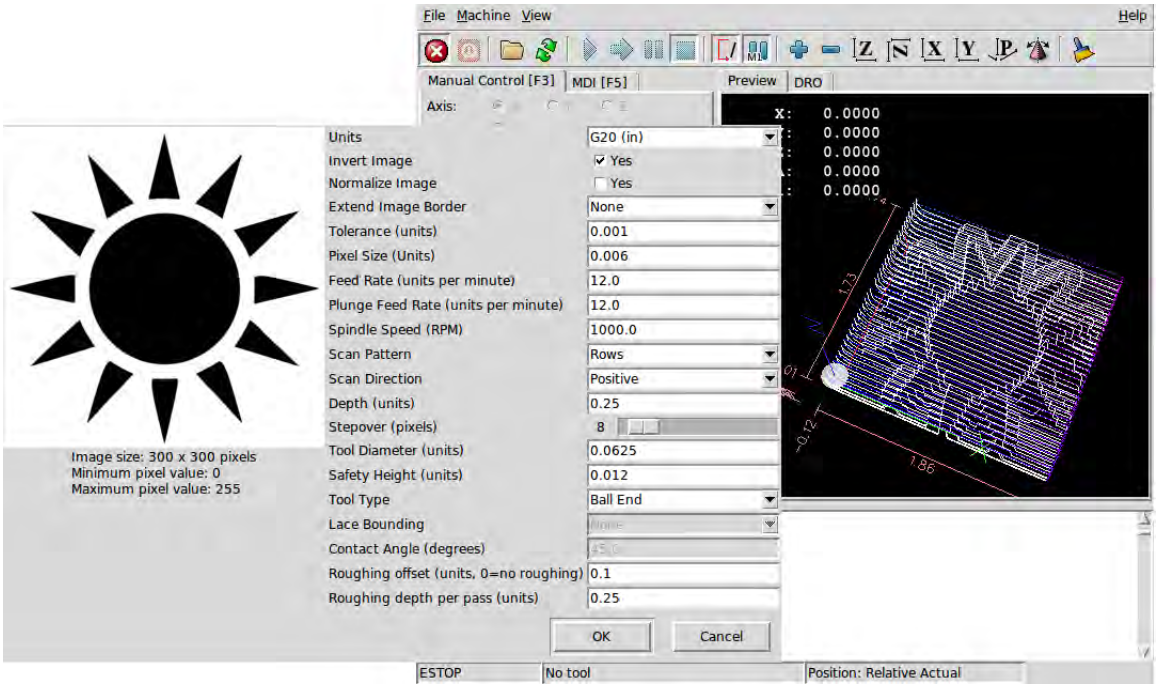


Figura 3.2: image–to–gcode es una aplicación para convertir imágenes a código G.

3. Illustrator: es un programa comercial y tiene problemas similares al programa Potrace (ver Figura 3.5), genera contornos de los trazos, no usa círculos y crea múltiples vectores por trazo. En la Figura 3.5 en el acercamiento, se ven múltiples líneas, esta imagen es el resultado de tomar como imagen de entrada la Figura 3.3b. La salida de este programa es también un archivo gráfico vectorial.

4. pstoedit: este programa tiene como entrada un archivo gráfico vectorial y permite generar como salida un archivo en código G. Lo utilizamos para traducir los archivos de Potrace e Illustrator a código G. Sin embargo, no usa las funciones de código G para trayectorias de arcos y círculos, solamente trazos lineales. Además los desplazamientos no son optimizados para minimizar el movimiento entre trazos, lo cual resulta en una pérdida de tiempo, gasto de energía y desgaste de la máquina CNC.

Estos programas ofrecen una solución parcial al problema de traducir trazos contenidos en una imagen, a movimientos en una máquina de control numérico.

Es de resaltar que aún para trazos delgados, se obtienen contornos y por lo

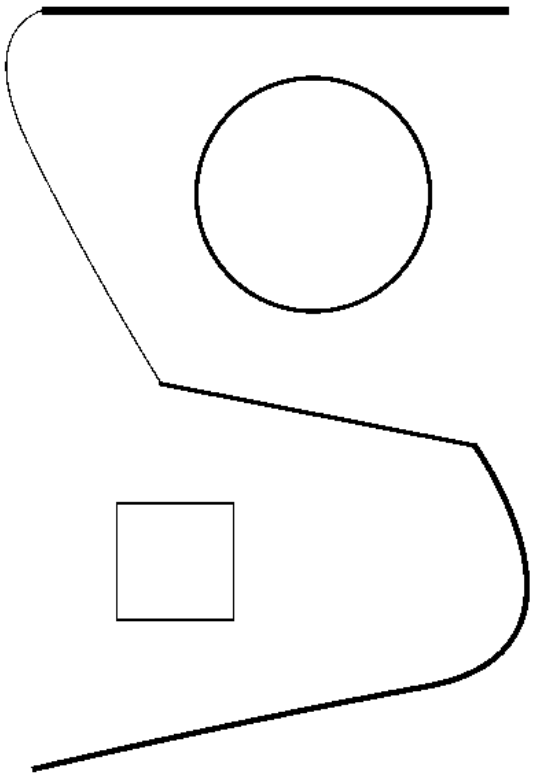


Figura 3.3: Imagen de entrada binarizada.

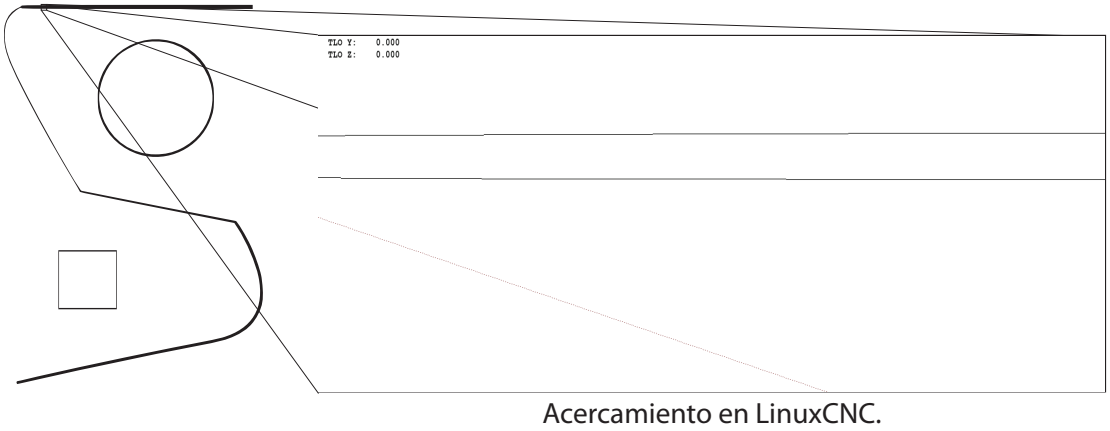


Figura 3.4: Resultado de Potrace.

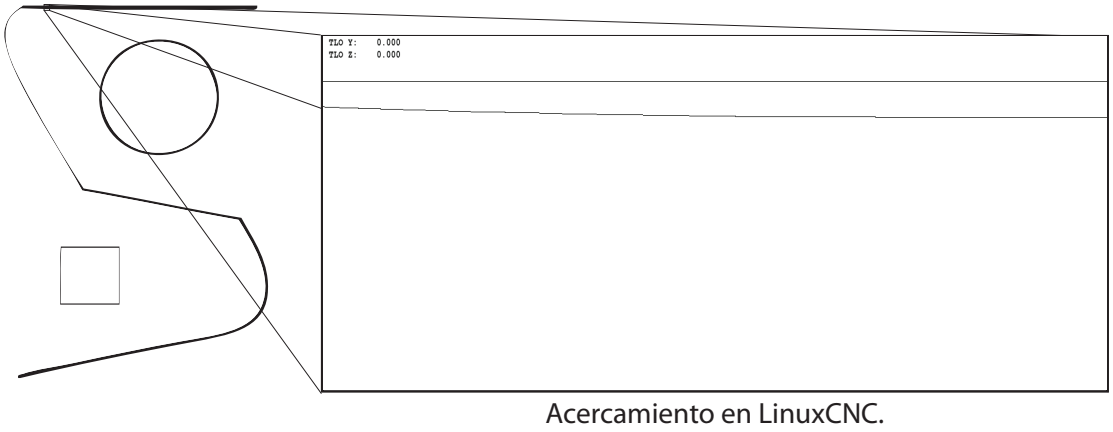


Figura 3.5: Resultado de Illustrator.

tanto se generan dos trazos. Este problema de Potrace se agrava en Illustrator donde, en ocasiones, se generan más de dos vectores por trazo. Adicionalmente, Potrace e Illustrator solamente utilizan segmentos de línea, no círculos o arcos.

En las siguientes secciones se desarrolla una aplicación con el objetivo de subsanar los problemas anteriormente mencionados.

3.2. Aplicación desarrollada

El objetivo de la nueva aplicación es resolver los problemas encontrados en los programas comerciales y libres disponibles, por lo tanto ahora los trazos delgados van a generar una sola línea; y se incluirán las funciones de círculos y segmentos de círculo. Se espera que funcione mejor que los programas disponibles. Esta nueva aplicación la llamamos trace2Gcode.

En esta sección presentamos la construcción de esta nueva aplicación especializada en generar la programación de una máquina CNC a partir de una imagen. Contiene dos programas: imageCNC y trace2Gcode.

El proceso a seguir es el siguiente:

Programa imageCNC:

1. La entrada es una fotografía del dibujo que contiene los trazos a seguir.
2. Este programa remueve la distorsión proyectiva introducida por el uso de una cámara y deja como salida una imagen binarizada.

Programa trace2Gcode:

3. Buscar trazos en la imagen binarizada de entrada.
4. En cada trazo, determinar qué píxeles pueden formar una línea mediante el algoritmo de seguimiento de línea [Leonardo Romero, 2013].
5. Aplicar el algoritmo de revisión hacia atrás para solucionar posibles errores del paso anterior [Leonardo Romero, 2013].
6. De las líneas obtenidas determinara qué líneas son mejor representadas por un círculo o arco.
7. Transformar las líneas, arcos y círculos a código G.
8. Ordenar las secuencias de movimientos, reduciendo la longitud total de trayectoria de la herramienta de trabajo.
9. La salida es un archivo en código G que puede ser interpretado por LinuxCNC para realizar la tarea deseada.

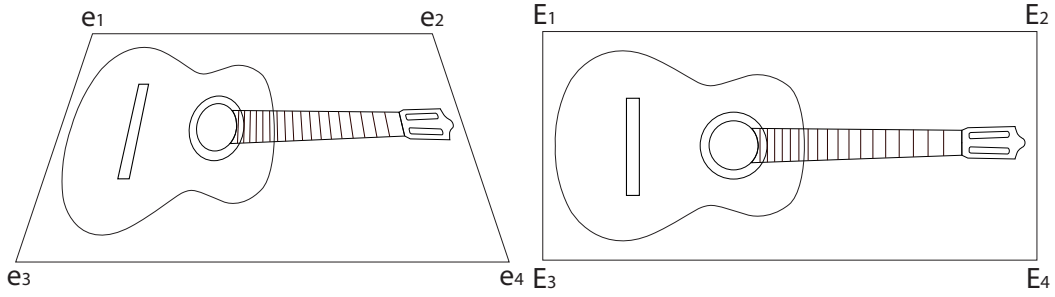
Un usuario podría saltarse el paso 1 y 2 y solamente utilizar el programa trace2Gcode con una imagen binarizada, generada por algún programa de computación (v. gr. Paint), o generada por un escaner de página completa (que no introduce la distorsión proyectiva de la cámara).

3.3. Corrección de la distorsión proyectiva

Para ilustrar el propósito de la aplicación imageCNC la Figura 3.6a muestra un ejemplo de la fotografía tomada al dibujo de la Figura 3.6b, dónde la parte inferior estaba

más cerca de la cámara y la parte superior estaba más alejada de la cámara; ahí observamos que se genera una distorsión proyectiva dependiendo del ángulo con que se tome la fotografía. El propósito de imageCNC es quitar la distorsión proyectiva, que el trapecio (Figura 3.6a) vuelva a ser un rectángulo (Figura 3.6b).

El propósito de la aplicación imageCNC es que a un dibujo, enmarcado por un rectángulo cuyas dimensiones son conocidas, con la suposición de que las esquinas del marco son los puntos más cercanos a la esquina correspondiente, le debe corregir la distorsión proyectiva ocasionada por la adquisición del dibujo con una cámara.



(a) Dibujo (b) Corrección

Figura 3.6: Corrección de la distorsión proyectiva.

La corrección la hacemos al tomar los puntos de las esquinas (e_1, e_2, e_3 y e_4) del marco y asociar las esquinas (E_1, E_2, E_3 y E_4) de la imagen deseada, mediante los siguientes pasos:

1. Aplicamos el detector de Harris [Harris y Stephens, 1988] para encontrar las esquinas del dibujo, usando la función getCorners de OpenCV [Laganiere, 2011].

2. Ubicamos las esquinas correspondientes al marco del dibujo. Así la esquina e_1 es la más cercana (utilizando distancia Euclidiana) a la esquina E_1, la e_2 a la E_2 y así sucesivamente. Con esto formamos los pares de correspondencia de puntos (esquinas del marco y esquinas deseadas).

3. Con esos pares de correspondencia de esquinas hacemos una transformación homográfica, para corregir la distorsión proyectiva del dibujo de entrada, utilizando la

función findHomography de OpenCV [Laganiere, 2011]. El resultado es una imagen como la mostrada en la Figura 3.6b.

4. La imagen corregida se binarizada, utilizando como referencia un umbral dado por el usuario.

Se asume que la foto es tomada con buena iluminación, que los trazos son contrastantes respecto al fondo. Por ejemplo, los trazos y el marco son de color negro sobre un fondo blanco.

Por convención la imagen de salida contendrá los trazos en color negro sobre un fondo blanco.

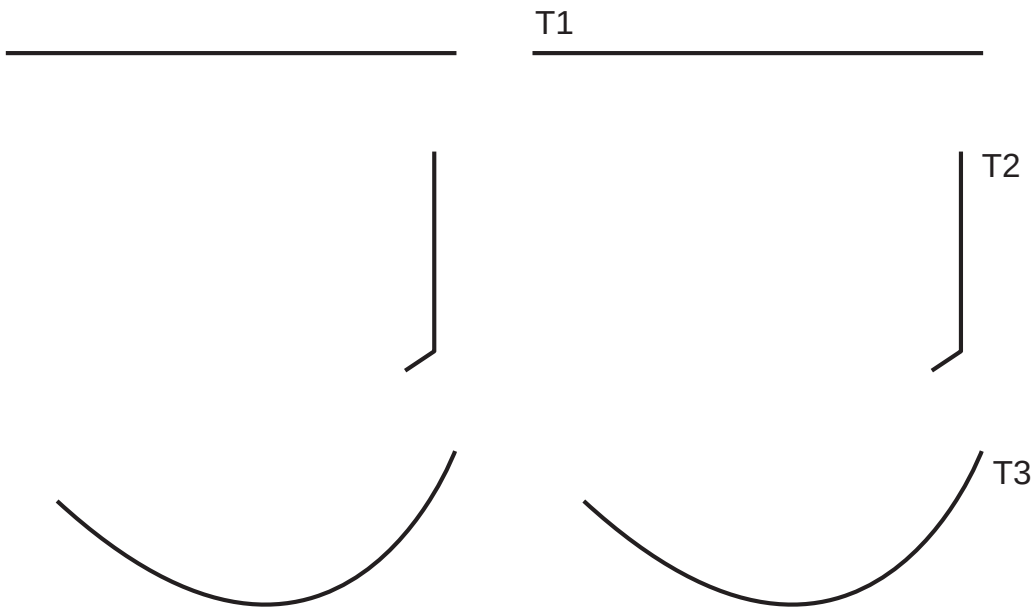
3.4. Búsqueda de trazos

En este paso encontramos los trazos contenidos en la imagen. Por ejemplo, la imagen mostrada en la Figura 3.7a contiene tres trazos, los cuales se muestran en la Figura 3.7b. Si consideramos un trazo como una secuencia de píxeles negros adyacentes, podemos asociar los píxeles de la imagen que corresponden a cada uno de los trazos, mediante una lista de píxeles de cada trazo.

Comenzamos por visitar cada píxel de la imagen, de izquierda a derecha, de arriba a abajo, hasta encontrar un píxel negro, a partir de ese píxel hacemos una búsqueda en profundidad sobre los píxeles vecinos aún no visitados; así vamos encontrando todos los píxeles negros conectados y los agregamos a la lista de píxeles del trazo, en caso de que sólo encuentre píxeles blancos vecinos, cerramos el trazo y continuamos con el barrido de la imagen buscando otro píxel negro. Al final únicamente guardamos una lista de trazos y cada trazo con su lista de píxeles que lo componen.

La búsqueda en profundidad la hacemos en el orden de prioridad que se ilustra en la Figura 3.8, dando prioridad a encontrar trazos horizontales, después verticales y al final diagonales.

Para facilitar la búsqueda de líneas o círculos en los trazos es conveniente que los píxeles se ordenen considerando un cierto nivel de “diámetro” de la búsqueda. Podemos



(a) Imagen de entrada (b) Trazos detectados

Figura 3.7: Trazos encontrados en una imagen de entrada.

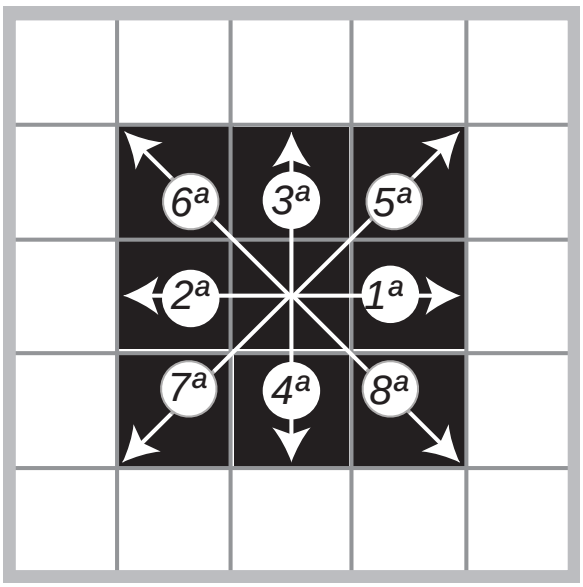


Figura 3.8: Prioridad en la búsqueda.

imaginar que la búsqueda se hace con una “lupa” de determinado diámetro, de manera que los trazos gruesos (de dos o más píxeles) se vean todos al mismo tiempo si son vistos por la “lupa”.

El algoritmo “busca trazos” implementa esta idea, y tiene como parámetro el nivel de búsqueda, con el que se definen dos tipos de píxeles, unos que están en la ventana de observación y otros que están en la ventana de búsqueda. En la Figura 3.9 se ilustran las ventanas para los niveles 1, 2 y 3. La ventana de observación puede ser de un solo píxel (nivel 1), de 3x3 píxeles (nivel 2), de 5x5 píxeles (nivel 3), etc. En la ventana de observación los píxeles que sean negros serán añadidos directamente al trazo. La búsqueda en profundidad se hace en los píxeles de la ventana de búsqueda. El efecto es que el recorrido de los trazos lo podemos hacer considerando un tamaño de ventana de observación pequeño, mediano o grande.

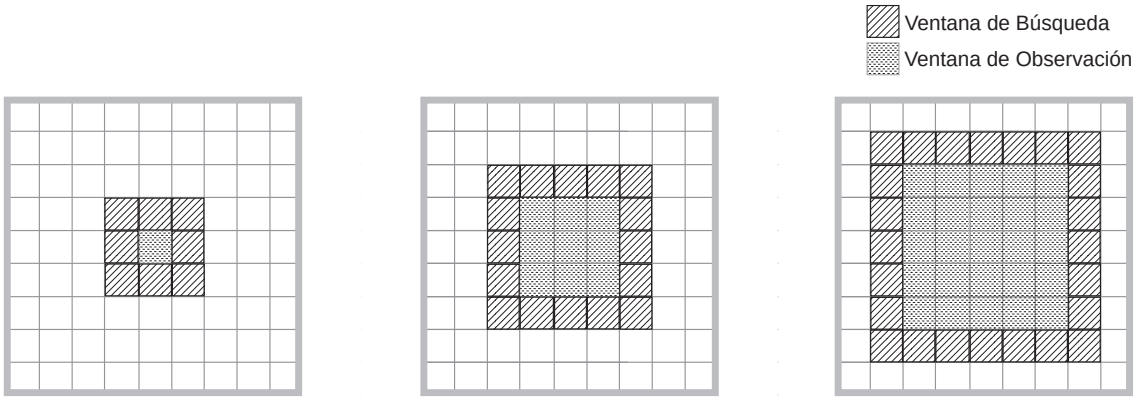


Figura 3.9: Ventanas de observación y de búsqueda.

Con ello tenemos un control del recorrido de los píxeles. El beneficio es que con un nivel grande, píxeles adyacentes físicamente están también adyacentes en el ordenamiento de la lista de píxeles del trazo; facilitando la tarea siguiente, de encontrar líneas o círculos. Ésto tiene relevancia si consideramos el diámetro de la herramienta de trabajo de la máquina CNC.

Como ejemplo del efecto del nivel de búsqueda en el trazo de la Figura 3.10, para un nivel 1, los píxeles son visitados en el orden de la secuencia numérica: $1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12$;

como se aprecia en la Figura 3.11a. Para un nivel 2, se sigue la secuencia numérica: $1, 1', 2, 2', 3, 3', 3'', 3''', 4, 4', 4'', 4'''$; como se aprecia en la Figura 3.11b, el píxel $3'$ y $3''$ están adyacentes en el orden de la lista y también físicamente muy cercanos, mientras que en un nivel 1 el píxel 6 y 12 están físicamente muy cerca, pero en la secuencia muy alejados. En los pasos siguientes, los algoritmos de construcción de líneas van a aprovechar esta secuencia, ellos procesan los píxeles que están cercanos para formar líneas.

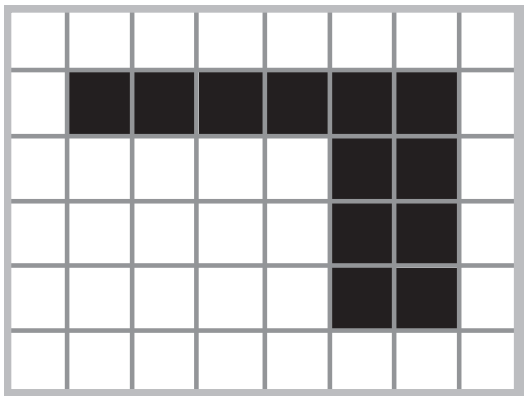
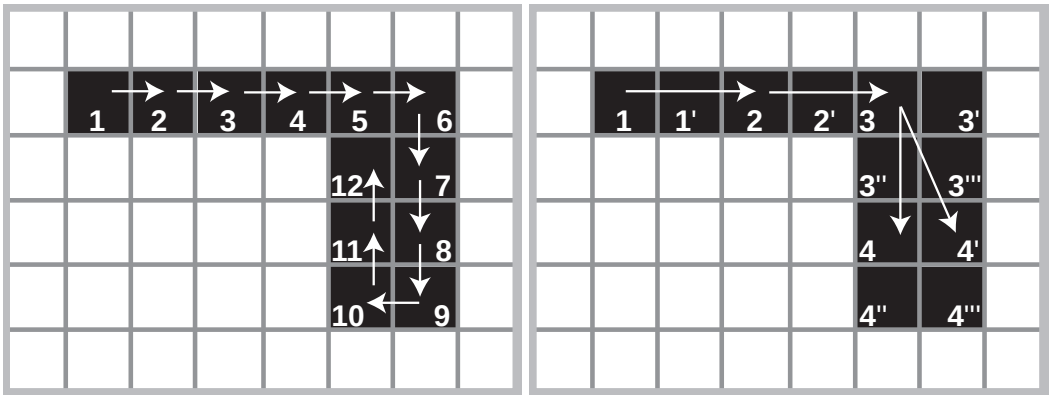


Figura 3.10: Trazo de entrada para probar el efecto de diferentes niveles de búsqueda.



(a) Nivel 1 (b) Nivel 2

Figura 3.11: Recorrido de un trazo con diferentes niveles.

3.5. Seguimiento de línea

La descripción de los algoritmos de *seguimiento de línea* y *revisión hacia atrás* es basada en la revisión existente de algoritmos de extracción de líneas de los artículos [Fernandez et al., 2010], [Borges y Aldon, 2004], [Nguyen et al., 2007] y [Leonardo Romero, 2013].

El objetivo es encontrar la secuencia de líneas contenidas en un trazo. Si se trata de un trazo recto probablemente se encontrará una sola línea, si se trata de un trazo con una curvatura suave, probablemente encontrará una secuencia de muchas líneas, que en otro paso posterior se verá si es más conveniente remplazar las líneas por un arco de círculo.

El algoritmo de seguimiento extendido de línea elegido [Leonardo Romero, 2013] es muy rápido y simple. El método se basa en el cálculo de una línea θ que se adapte a una secuencia de puntos. El algoritmo inicia uniendo los dos primeros puntos con una línea, si éstos están próximos entre sí. Para considerar agregar un nuevo punto a la línea, digamos z_n, se calcula la distancia ortogonal $d_\perp(z_n, \theta)$ del punto z_n a la línea θ en cuestión. Si la distancia es mayor que un valor de referencia fijo T_{max}, se inicia una nueva línea. De lo contrario, la línea θ se calcula con todos los puntos, incluyendo z_n.

Una línea es representada en su forma polar:

$$\theta_j = \langle r_j, \phi_j \rangle \tag{3.1}$$

donde r_j y ϕ_j son la distancia de la normal y el ángulo de inclinación de la normal, respectivamente. La Figura 3.12 ilustra esta representación donde la normal es el segmento de recta más corto entre la línea y el origen del sistema de referencia. Los puntos $z = \langle x, y \rangle$ que están en la línea $\theta_j = \langle r_j, \phi_j \rangle$ satisfacen la ecuación:

$$r_j = x \cos \phi_j + y \sin \phi_j \tag{3.2}$$

La distancia ortogonal de un punto $z_i = \langle x_i, y_i \rangle$ a la línea θ_j está dada por:

$$d_\perp(z_i, \theta_j) = r_j - x_i \cos \phi_j - y_i \sin \phi_j. \tag{3.3}$$

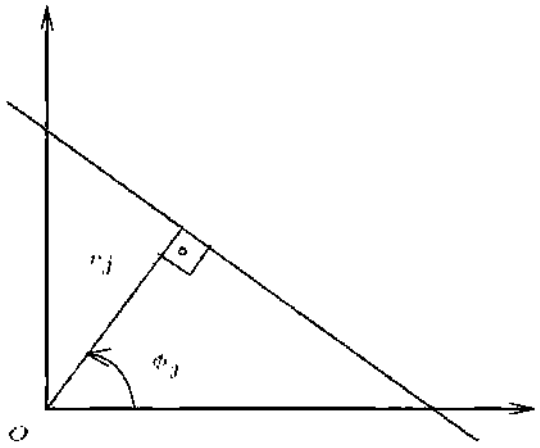


Figura 3.12: Representación de una línea en su forma polar.

Esta ecuación puede ser fácilmente obtenida considerando la línea $r_i = x_i \cos\phi_i + y_i \sin\phi_i$; dicha línea pasa por el punto z_i y es paralela a la línea θ_j cuando $\phi_i = \phi_j$; en cuyo caso, $d_\perp(z_i, \theta_j) = r_j - r_i$.

Asumiendo que la mejor linea θ con parámetros $\langle r, \phi \rangle$ minimiza la suma de los cuadrados de las distancias perpendiculares de los puntos a la línea; si la línea tiene n puntos $\langle x_i, y_i \rangle$ $(i = 1, \ldots, n)$, la solución está dada por [Arras y Siegwart, 1997]:

$$\begin{bmatrix} r \\ \phi \end{bmatrix} = \begin{bmatrix} \overline{x}\cos\phi + \overline{y}\sin\phi \\ \frac{1}{2}\arctan\frac{-2s_{xy}}{s_{yy}-s_{xx}} \end{bmatrix} \tag{3.4}$$

donde

$$\begin{aligned} \overline{x} &= \frac{1}{n}\sum_{i=1}^{n} x_i, & \overline{y} &= \frac{1}{n}\sum_{i=1}^{n} y_i, \\ s_{xx} &= \sum_{i=1}^{n}(x_i - \overline{x})^2, & s_{yy} &= \sum_{i=1}^{n}(y_i - \overline{y})^2, \\ s_{xy} &= \sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y}). \end{aligned}$$

El apéndice D contiene la derivación completa del método de minimos cuadrados totales para encontrar la mejor línea dada por un conjunto de puntos.

El Algoritmo 1, calcula la secuencia de líneas, así como los puntos que forman cada línea, a partir de la secuencia de puntos de un trazo.

Algoritmo 1 Seguimiento de línea

Entrada: Una secuencia Z de puntos $(z_1, \ldots, z_N)$ y T_{max} la distancia máxima a la que puede estar un punto de la línea para ser incluido

Salida: Una secuencia Θ de líneas $(\theta_1, \ldots, \theta_M)$ y una secuencia L de listas de puntos $(L_1, \ldots, L_M)$ donde L_i contiene los puntos de la línea θ_i

$i \leftarrow 1$, $j \leftarrow 1$, $l \leftarrow 1$, $\Theta \leftarrow \{\}$, $L \leftarrow \{\}$
while $j < N - 2$ **do**
  $\theta_l \leftarrow$ Mejor línea con $(z_i, \ldots, z_{j+1})$ { Ecuación 3.4 }
  $T \leftarrow d(z_{j+2}, \theta_l)$ {Ecuación 3.3 }
  if $T > T_{max}$ **then**
    Agregar línea θ_l a Θ
    Agregar los puntos $(z_i, \ldots, z_j + 1$ a la lista L_i
    $l \leftarrow l + 1$
    $i \leftarrow j + 2$
    $j \leftarrow i$
  else
    $j \leftarrow j + 1$
  end if
end while
return Θ y L

3.6. Revisión hacia atrás

Cuando el algoritmo de seguimiento de línea está añadiendo nuevos puntos a una línea, es posible que algunos de los puntos finales (antes de alcanzar el valor de

T_{max}) realmente pertenezcan a la línea siguiente. Esta situación se muestra en la Figura 3.13, donde hay dos líneas (una horizontal y una vertical), pero la línea horizontal incluye puntos verticales, por lo que se mueve ligeramente hacia arriba la línea calculada por estos puntos, debido al punto erróneo añadido (el punto de la esquina).

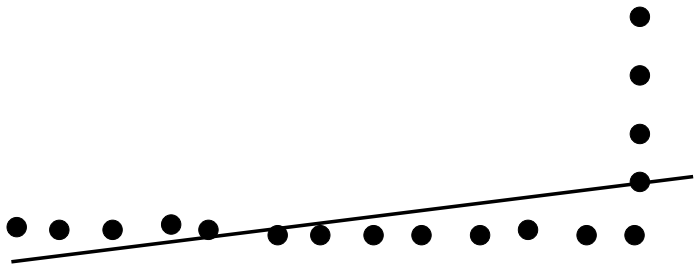


Figura 3.13: Algoritmo de seguimiento de línea no calculó con precisión la línea.

El Algoritmo 2, de revisión hacia atrás, decide si es mejor asociar los últimos puntos de línea θ_l a la línea θ_{l+1} y considera dos casos:

1. La línea θ_l tiene 3 o más puntos. En este caso, el último punto de θ_l se asocia a la línea más cercana: θ_l o θ_{l+1}. Si el punto está más cercano a θ_{l+1}, la línea θ_{l+1} se vuelve a calcular con el nuevo punto. Este paso se repite hasta que un punto del final de θ_l realmente pertenece a la línea θ_l. De esta manera, este paso elimina las asociaciones erróneas del algoritmo de seguimiento de línea.

2. La línea θ_l tiene 2 puntos. En este caso, si la distancia desde el segundo punto de θ_l a la línea θ_{l+1} es menor o igual a T_{max}, ese punto es asociado a la línea θ_{l+1}. En ese caso, la misma revisión es ejecutada con el primer punto de θ_l. Al final, en la línea de θ_l puede haber dos, uno o cero puntos.

3.7. Encontrando círculos y segmentos de círculo

Una vez que tenemos representados los puntos como líneas, también es posible que esos puntos puedan ser mejor representados por segmentos de círculos o círculos completos.

Algoritmo 2 Revisión hacia atrás

Entrada: Una secuencia Z de puntos $(z_1, \ldots, z_N)$, T_{max}, Θ $(\theta_1, \ldots, \theta_M)$ e I

Salida: Θ e I

for $i = 1, \ldots, M-1$ **do**
  $\langle s_1, e_1 \rangle \leftarrow I(i)$ {índices de la línea θ_i}
  $\langle s_2, e_2 \rangle \leftarrow I(i+1)$ {índices de la línea θ_{i+1}}
  $d_2 \leftarrow d(z_{e_1}, \theta_{i+1})$
  if $d_2 < T_{max}$ **then**
    repeat
      $d_2 \leftarrow d(z_{e_1}, \theta_{i+1})$
      if $e_1 - s_1 + 1 > 2$ **then**
        $\theta_i' \leftarrow$ la mejor línea con $(z_{s_1}, \ldots, z_{e_1-1})$
        $d_1 \leftarrow d(z_{e_1}, \theta_i')$ { θ_i' no incluye z_{e_1} }
      else
        $d_1 = T_{max}$
      end if
      if $d_2 < d_1$ **then**
        $e_1 \leftarrow e_1 - 1$, $s_2 \leftarrow s_2 - 1$ {ajusta índices}
        Actualiza índices de las líneas θ_i y θ_{i+1} en I
        recálcula líneas θ_i y θ_{i+1} y actualiza Θ
      else
        break **repeat**
      end if
    until $s_1 = e_1$
  end if
end for
Elimina de Θ e I, las líneas sin puntos
return Θ e I

En general, se toma dos líneas, que están a una distancia menor o igual a T_{max}; con los puntos de esas líneas se calcula el círculo que mejor se ajuste a los puntos.

Comparamos el error del círculo y el error de las líneas. El error del círculo E_c es la suma de los cuadrados de distancias de los puntos a la circunferencia calculada. El error de línea E_l es la suma de los cuadrados de distancias de los puntos a sus líneas correspondientes.

Si $E_c < E_l$ es preferible representar los puntos con la circunferencia calculada.

Si dos líneas adyacentes se fusionan en una circunferencia se prueba si las líneas que siguen también están mejor representadas por la misma circunferencia.

Para determinar el centro y el radio de una circunferencia dados los puntos probamos dos métodos.

En el primero, encontramos la solución del sistema de ecuaciones formado por las normales a las líneas; la solución es el punto de intersección de las normales, lo cual nos indica que existen posibilidades que esas líneas formen parte de un círculo (ver Figura 3.14), sino existe solución, las líneas son paralelas y descartamos la opción de que puedan ser parte de un círculo.

En caso de que exista solución, esa solución es el centro del círculo y su radio es el promedio de las distancias del centro del círculo a cada uno de los puntos que contienen las líneas.

El segundo método que probamos es el llamado Modified Least-Squares (MLS) [Umbach y Jones, 2003], que calcula el centro (C_x, C_y) y radio (r) con las siguientes fórmulas cerradas:

$$C_x = \frac{DC - BE}{AC - B^2} \tag{3.5}$$

$$C_y = \frac{AE - BD}{AC - B^2} \tag{3.6}$$

$$r = \sum_{i=1}^{n} \sqrt{(x_i - C_x)^2 + (y_i - C_y)^2}/n \tag{3.7}$$

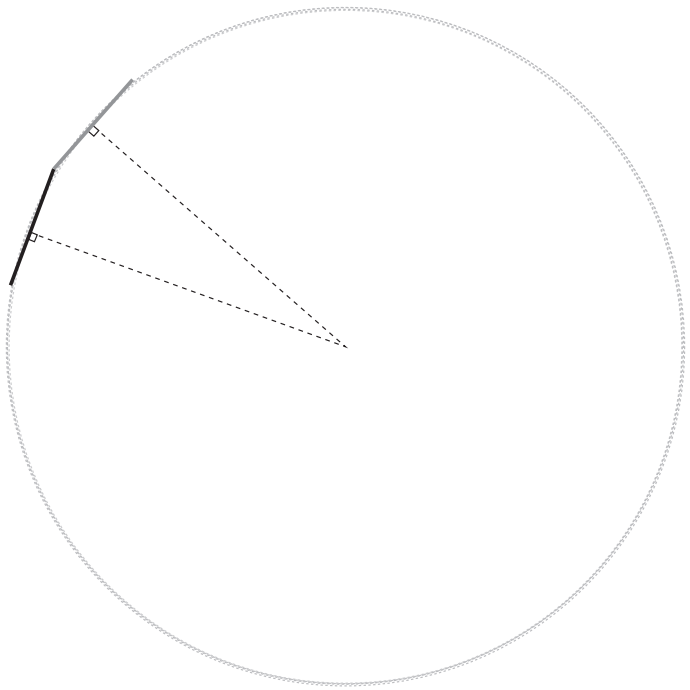


Figura 3.14: Punto de intersección de las líneas normales.

donde

$$A = n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2 \tag{3.8}$$

$$B = n\sum_{i=1}^{n} x_i y_i - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right) \tag{3.9}$$

$$C = n\sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2 \tag{3.10}$$

$$D = 0.5\left\{n\sum_{i=1}^{n} x_i y_i^2 - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i^2\right) + n\sum_{i=1}^{n} x_i^3 - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} x_i^2\right)\right\} \tag{3.11}$$

$$E = 0.5\left\{n\sum_{i=1}^{n} y_i x_i^2 - \left(\sum_{i=1}^{n} y_i\right)\left(\sum_{i=1}^{n} x_i^2\right) + n\sum_{i=1}^{n} y_i^3 - \left(\sum_{i=1}^{n} y_i\right)\left(\sum_{i=1}^{n} y_i^2\right)\right\} \tag{3.12}$$

Con el método Modified Least-Squares se obtuvieron mejores resultados en exactitud del cálculo del centro y radio de la mejor circunferencia que se ajusta a los puntos. En el apéndice E se describe el método MLS con mayor detalle.

3.8. Reordenamiento de trazos

En el código G todos los trazos tiene dos puntos extremos, de manera que un trazo en código G es representado por un punto de inicio y un punto final. Cuando termina un trazo la máquina CNC, la herramienta de trabajo se levanta y se posiciona en el inicio del siguiente trazo.

Para desplazamientos en línea recta usamos la instrucción G01 y para desplazamientos con la herramienta de trabajo levantada usamos G00. El lector puede consultar el apéndice C para un tutorial de la programación en código G.

En el caso de arcos usamos el formato con coordenadas del centro G02 y G03, respectivamente dependiendo del trazo. En circunferencias completas, el punto de inicio y de fin, puede ser cualquier punto de la circunferencia.

El reordenamiento de los trazos en código G la realizamos de la siguiente manera:

1. El primer trazo t_i es el que tiene uno de sus extremos más cerca al origen del sistema de referencia.

2. El siguiente trazo t_{i+1} es el más cercano al punto final de t_i.

3. El paso anterior se repite, de modo que el siguiente trazo es el más cercano al anterior.

El objetivo es buscar que la máquina CNC no tenga que hacer desplazamientos innecesarios. Con lo cual logramos un ahorro de tiempo, energía y vida útil de la máquina CNC. También se mejora el desempeño, si el punto final del trazo y el inicio del siguiente están muy cercanos (por ejemplo menor a T_{max}) se evita subir y bajar la herramienta para iniciar el trazo siguiente.

3.9. Parámetros

La aplicación imageCNC tiene como parámetros:

1. Nombre del archivo de imagen de entrada en formato JPG.
2. El ancho del marco en centímetros (un número real).
3. El alto del marco en centímetros (un número real).
4. Nombre del archivo de la imagen binaria de salida generada en formato JPG.
5. El umbral de binarización (un entero entre 0 y 255).
6. Un umbral para eliminar los restos del marco en la imagen (un entero entre 0 y el número máximo de píxeles).

La aplicación se ejecuta con la instrucción siguiente:

./imageCNC [entrada.jpg] [ancho] [alto] [salida.jpg] [umbral] [ancho_marco]

Por ejemplo, para un dibujo en un archivo llamado dibujo_entrada.jpg, con un marco de 20.0 centímetros de ancho y 15.0 centímetros de alto, un archivo de salida llamado dibujo_binarizado.jpg, con un umbral de binarización de 128, un umbral de 10 para eliminar los restos del marco, la instrucción es la siguiente:

./imageCNC dibujo_entrada.jpg 20.0 15.0 dibujo_binarizado.jpg 128 10

La aplicación trace2Gcode tiene los parámetros:

1. El nombre del archivo de entrada en formato JPG (imagen binaria).
2. El umbral para el algoritmo de seguimiento de línea y revisión hacia atrás (umbral T).
3. El nivel para la búsqueda de trazos (nivel N).
4. La anchura en centímetros que representa la imagen. La altura se ajusta automáticamente conservando la relación de aspecto.

5. La profundidad con que se hará el tallado o corte (en milímetros). Se debe dar un número negativo.

6. La altura de los movimientos seguros (con la herramienta levantada en milímetros). Un número positivo sube la herramienta.

7. La velocidad del desplazamiento de la herramienta en milímetros por minuto.

Esta aplicación se ejecuta con la instrucción:

./trace2Gcode [entrada.jpg] [umbral T] [nivel N] [ancho] [profundidad] [altura] [velocidad]

Por ejemplo, para un archivo de entrada llamado dibujo_binarizado.jpg, con un umbral $T = 3.5$, un nivel $N = 3.0$, con un marco de anchura de 20.0 centímetros, para una profundidad de tallado o corte a 2.0 milímetros, con movimientos seguros a 10.0 milímetros de altura y a una velocidad de 100 milímetros por minuto, se utiliza la siguiente instrucción:

./trace2Gcode [dibujo_binarizado.jpg] 3.5 3.0 20.0 -2.0 10.0 100

El resultado será un archivo en código con el nombre gcode.ngc.

Capítulo 4

Resultados experimentales

En este capítulo presentamos los resultados experimentales de trace2Gcode con imageCNC, las aplicaciones diseñadas en el capítulo anterior.

Partiendo de una fotografía del dibujo con los trazos a mecanizar, la aplicación imageCNC corrige la distorsión proyectiva del dibujo y lo deja en formato binario, justo como lo necesita traceGcode; el programa trace2Gcode trata de obtener líneas, después ve si esas líneas son mejor representadas con circunferencias y finalmente convierte los trazos a código G, que se puede pasar al programa LinuxCNC para su ejecución en la máquina CNC.

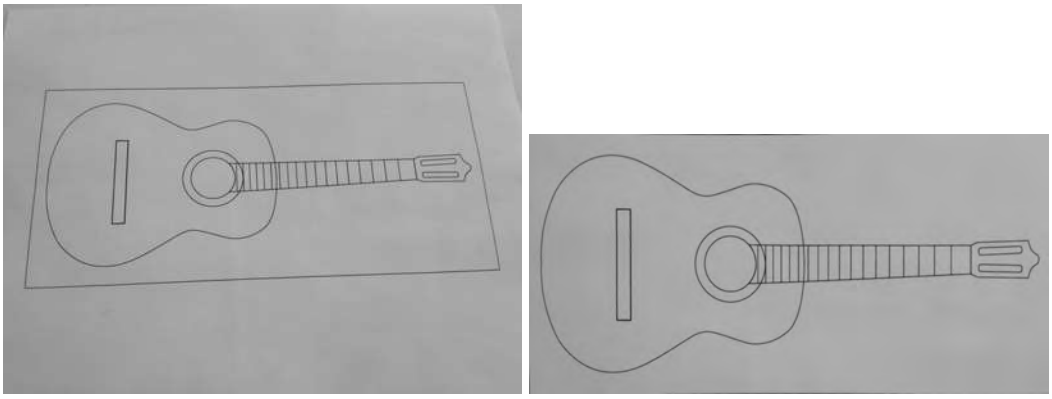
Realizamos una comparación del programa desarrollado con respecto de Potrace e Illustrator, y observamos su desempeño. También lo comparamos contra un dibujo hecho con un mínimo de trazos, el dibujo de la base del robot móvil mostrada en la Figura C.12 en el apéndice C.

Finalmente hacemos una comparación de los resultados obtenidos en relación al tiempo, número de líneas de código G y el número de funciones circulares generadas.

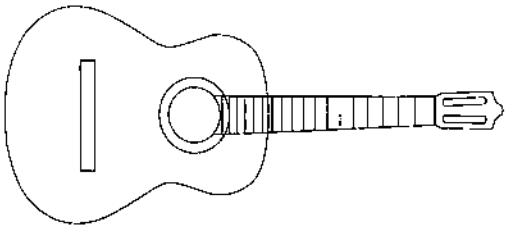
4.1. Caso de prueba 1: Un dibujo de una guitarra

De acuerdo al planteamiento del problema es necesario adquirir el dibujo desde cualquier medio, para ello usaremos el programa imageCNC para la corrección de

la distorsión proyectiva. La Figura 4.1 ilustra la imagen del archivo en formato JPG que capturó una cámara de un teléfono celular y la corrección de la distorsión proyectiva.



(a) Dibujo capturado (b) Dibujo corregido



(c) Binarización

Figura 4.1: Corrección de la distorsión proyectiva.

Al programar imageCNC hay que indicarle el archivo de entrada, el ancho y alto del marco en centímetros, el archivo de salida, el umbral de binarización (puede tomar valores entre 0 y 255) y valor para eliminar los restos del marco en la imagen. Para el dibujo de la Figura 4.1a con un marco de 10 centímetros de ancho y 5 centímetros de alto, usamos un umbral de binarización de 101 y un valor de 21 para eliminar los restos del marco. El siguiente comando da el resultado buscado:

./imageCNC entrada.jpg 10.0 5.0 salida.jpg 101 21

El resultado se ilustra en la Figura 4.1c, donde se observa que la guitarra no tiene distorsión proyectiva.

Una vez que contamos con el dibujo ejecutamos la aplicación trace2Gcode. Ini-

ciamos sintonizando los 2 parámetros más importantes, el umbral de los algoritmos Line Tracking y Back tracking (T) y el nivel de búsqueda de trazos (N), encargados de determinar la forma del mecanizado de los trazos.

A manera de ejemplo usamos la Figura 4.1c, fijamos $N = 4.0$ y variamos T, para ver los efectos de T.

La Figura 4.2 es el resultado de un $T = 1.5$, donde se nota que tiene trazas de distancias cortas.

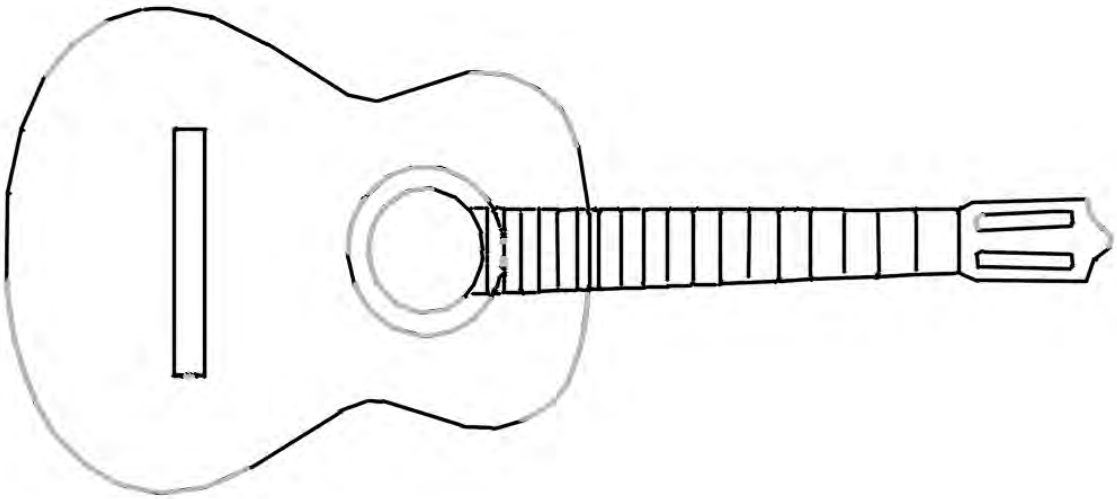


Figura 4.2: T $= 1.5$, N $= 4.0$, Líneas $= 251$, Arcos $= 25$.

La Figura 4.3 es el resultado de un $T = 2.0$, se nota que tiene trazos con mayor distancia y por tanto el número de trazos disminuye, comparado con $T = 1.5$.

La Figura 4.4 es el resultado de un $T = 5.0$, se nota que aumenta la distancia de los trazos y por tanto el número de trazos disminuye, comparado con $T = 2.0$. Esto hace el mecanizado de la guitarra con escalones.

El parámetro N sirve para indicar el tamaño de la ventana de búsqueda para encontrar trazos. Para un dibujo con trazos gruesos, un N grande, asegurará que píxeles adyacentes físicamente estén también adyacentes en el ordenamiento y con ello facilitará una mejor construcción de líneas.

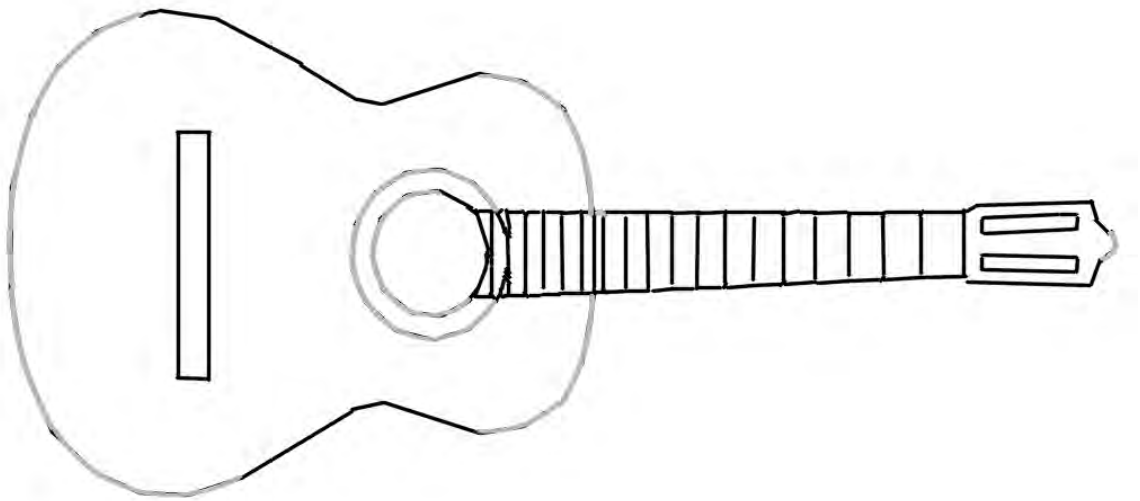


Figura 4.3: T $= 2.0$, N $= 4.0$, Líneas $= 172$, Arcos $= 17$.

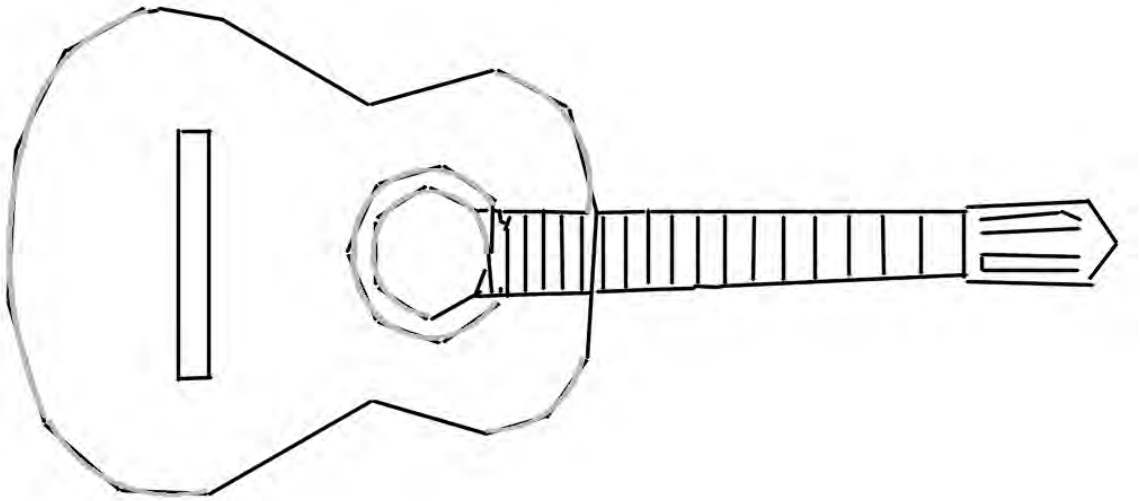


Figura 4.4: T $= 5.0$, N $= 4.0$, Líneas $= 81$, Arcos $= 9$.

La Figura 4.5 ilustra el resultado físico del mejor desempeño de trace2Gcode, que se alcanzo al usar los parámetros de T $= 2$ y N $= 4$. Se nota que se logra obtener el

mecanizado de la guitarra, sin embargo son notorios los escalones entre trazos.

Figura 4.5: Resultado físico de trace2Gcode con T$= 2.0$ y N $= 4.0$.

Con el mismo dibujo de entrada (Figura 4.1c) usamos los programas Potrace e Illustrator, junto con pstoedit. La Figura 4.6 ilustra de cerca los resultados de trace2Gcode, Potrace e Illustrator, donde se observa que trace2Gcode evita los trazos dobles pero los otros dos programas generan un mecanizado más cercano a la forma de la guitarra de entrada.

En la Tabla 4.1 se muestra la comparación en tiempo de maquinado, número de líneas de código G y el número de trazos circulares generados, para el dibujo con un ancho de 30 centímetros, a una velocidad de corte de 256 milímetros por minuto (10.0790 pulgadas por minuto), bajando la herramienta de corte 8 milímetros (0.3149 pulgadas) y con movimientos seguros a 10 milímetros (0.3937 pulgadas). Se aprecia que trace2Gcode

logra sobresalir de Potrace e Illustrator en número de líneas de código y en tiempo, al no generara trazos dobles, usar trazos circulares y minimizar los desplazamientos entre trazos.

Programa	Tiempo	Líneas de Código	Trazos circulares
Potrace	16.1 min	1267	0
Illustrator	35.2 min	2431	0
trace2Gcode	9.4 min	518	17

Tabla 4.1: Resultados con trace2Gcode y los programas Illustrator y Potrace, para un dibujo de una guitarra.

4.2. Caso de prueba 2: Un dibujo de una base de un robot móvil

Teniendo como entrada el dibujo de la Figura 4.7 usamos trace2Gcode, Illustrator y Potrace.

La Figura 4.8 ilustra el resultado de trace2Gcode, se observa un mecanizado escalonado.

La Figura 4.10 ilustra el resultado de Illustrator, se observa un mecanizado escalonado y trazos dobles.

La Figura 4.9 ilustra el resultado de Potrace, se tiene trazos dobles pero un mecanizado más cercano al ideal.

La Figura 4.11 ilustra el resultado de un código hecho a mano (código G presentado en la sección C.4 del Apéndice C en la página 103).

En la Tabla 4.2 se muestra la comparación en tiempo, número de líneas de código y el número de trazos circulares generados, para el dibujo a una velocidad de corte de 256 mm por minuto (10.079 pulgadas por minuto), bajando la herramienta de corte 8 milímetros (0.3149 pulgadas) y con movimientos seguros a 10 milímetros (0.3937 pulgadas). Se aprecia que trace2Gcode logra sobresalir de Potrace e Illustrator en número de líneas de código y en tiempo, al no generara trazos dobles, usar trazos circulares y minimizar los desplazamientos entre trazos.

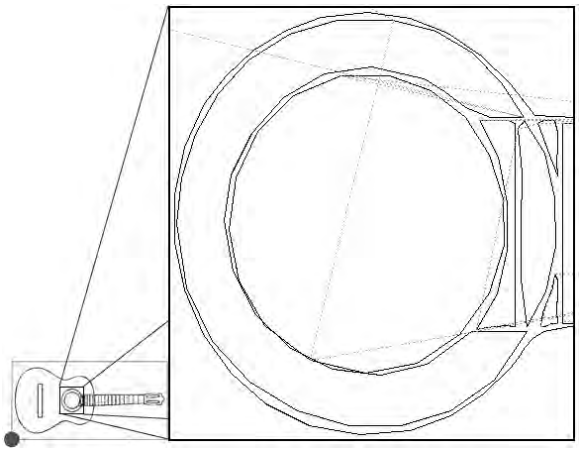
Programa	Tiempo	Líneas de Código	Trazos circulares
Potrace	7.9 min	373	0
Illustrator	18.6 min	1233	0
trace2Gcode	4.0 min	98	11
Código manual	4.1 min	37	2

Tabla 4.2: Resultados con trace2Gcode, Potrace, Illustrator y el código ideal para un dibujo de un robot móvil.

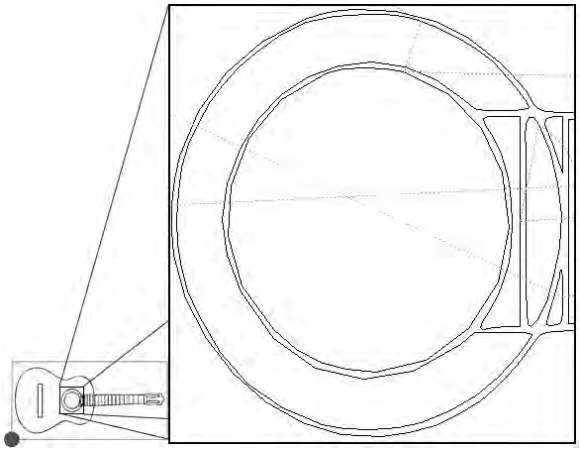
El programa trace2Gcode aún no logra representar el dibujo perfectamente, la sintonización de los parámetros T y N tienen diferente efecto sobre cada una de las circunferencias y es más fácil hacer la sintonización para representar mejor el círculo grande, ya que no es igual la discretización en un círculo grande y en un círculo pequeño. En el círculo grande la mayor parte es representada por arcos, mientras que en el círculo pequeño la mayor parte es representado por líneas; la resolución de la imagen es importante.

4.3. Análisis de resultados

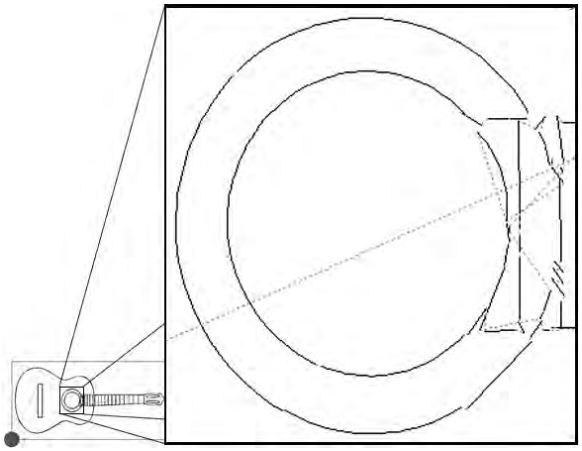
- ImageCNC logra que el usuario pueda mecanizar sus dibujos físicamente, sin necesidad de usar software especializado.
- Trace2Gcode genera movimientos sencillos, mientras los otros hacen movimientos dobles.
- Trace2Gcode incluye circunferencias.
- El tiempo de maquinado para trace2Gcode es significativamente menor que para los otros programas.



(a) Illustrator



(b) Potrace



(c) trace2Gcode

Figura 4.6: Resultados de los tres programas.

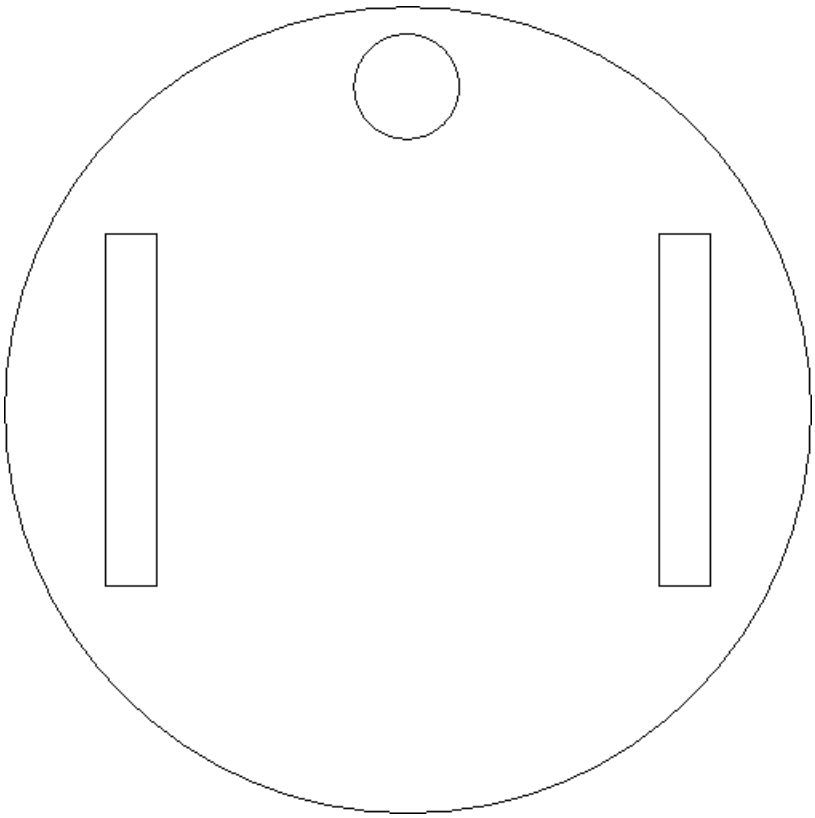


Figura 4.7: Dibujo de una base para un robot móvil.

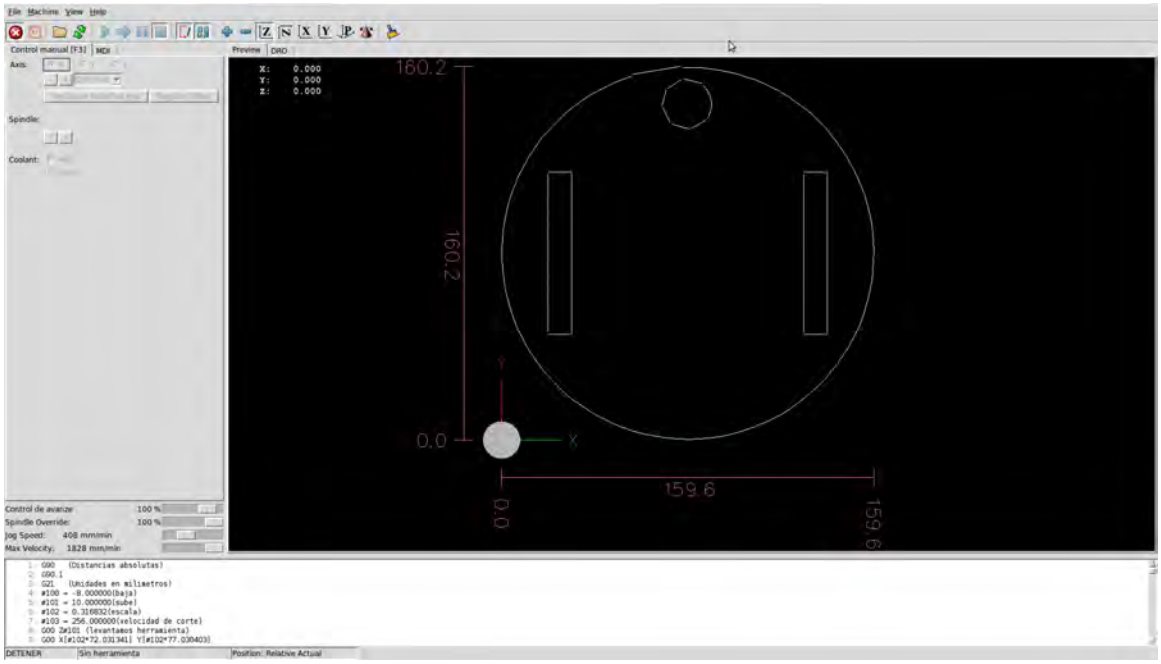


Figura 4.8: Resultado en LinuxCNC de trace2Gcode con la base del robot móvil.

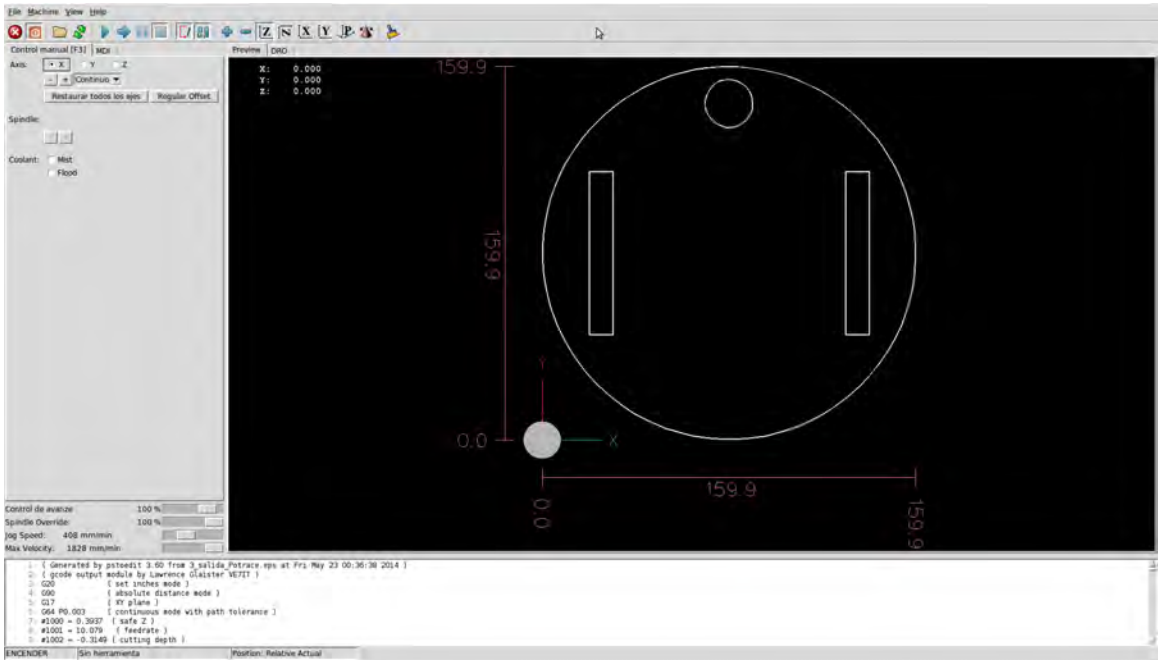


Figura 4.9: Resultado en LinuxCNC de Potrace con la base del robot móvil.

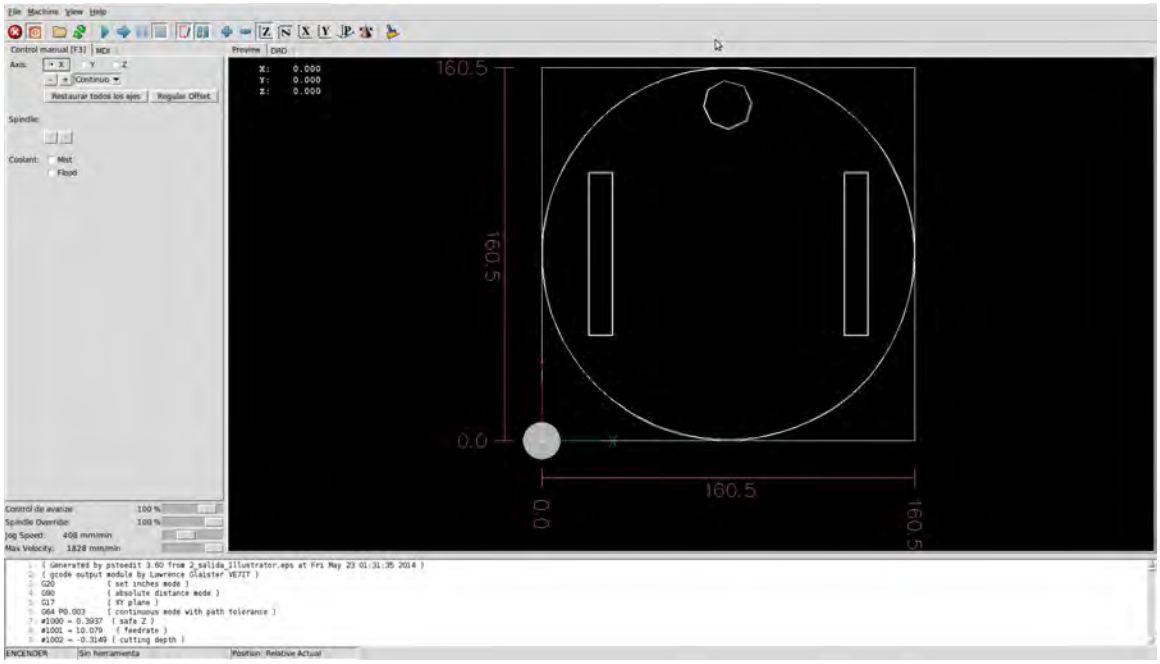


Figura 4.10: Resultado en LinuxCNC de Illustrator con la base del robot móvil.

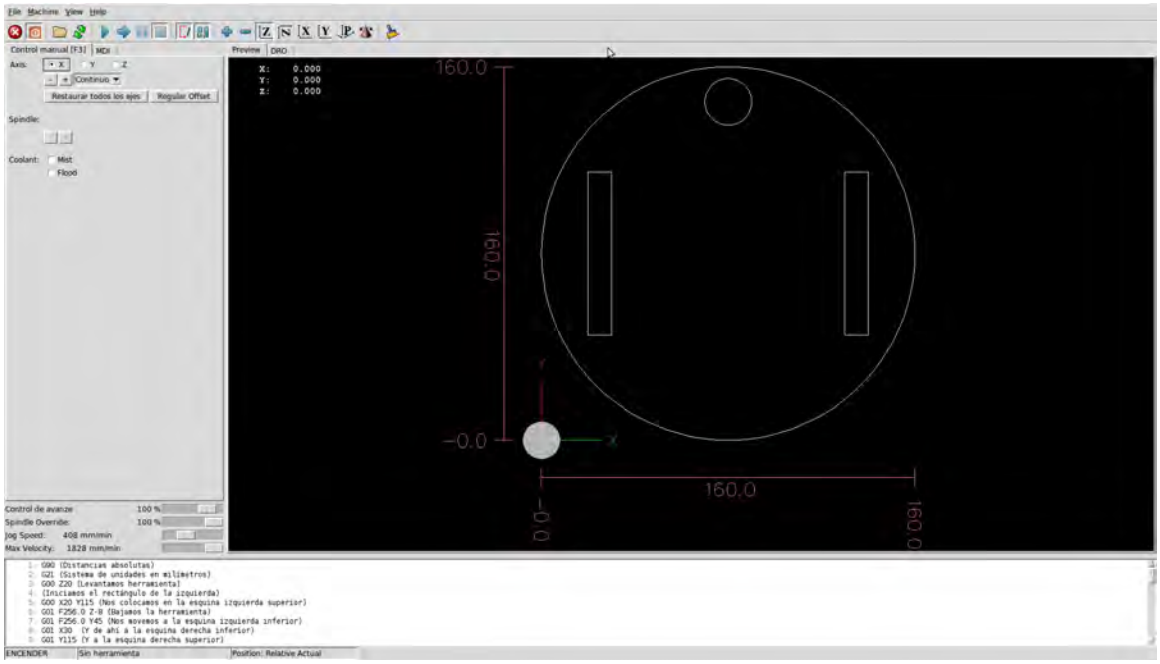


Figura 4.11: Resultado ideal de la base del robot móvil.

Capítulo 5

Conclusiones y trabajos futuros

A continuación se presentan las conclusiones del desarrollo de este trabajo, resaltando los logros obtenidos y señalando los posibles trabajos futuros que amplían el alcance de esta tesis.

5.1. Conclusiones generales

Se consiguió el objetivo perseguido en este trabajo de tesis que fue el desarrollo de dos máquinas CNC para materiales suaves y una aplicación para mecanizado a partir de imágenes. La tesis aporta los siguientes beneficios:

- **Máquinas CNC de bajo costo.** Las máquinas CNC desarrolladas, comparadas con las máquinas comerciales, presentan una inversión inicial muy pequeña (menos de la mitad de un modelo comercial de características similares). Este punto es muy importante, porque las máquinas CNC comerciales tienen costos superiores a los 12,000.00 dólares, las cuales, en muchas ocasiones, no son accesibles a las micro, pequeñas y medianas empresas.

- **Arquitectura abierta.** Los motores, controladores y el dispositivo de comunicación son estándares, permitiendo la incorporación de varios fabricantes y un mantenimiento simple al no depender de un único proveedor.

- **Máquina CNC basada en una Computadora Personal.** A nivel internacional una tendencia hacia una arquitectura abierta de máquinas CNC, es utilizar como procesador principal una computadora personal. En este desarrollo seguimos esta tendencia, para no depender de hardware especializado o ligado a algún proveedor. Esto permite actualizar el equipo a muy bajo costo y seguir aprovechando los avances tan vertiginosos que se están dando en las computadoras personales.

- **Basado en LinuxCNC.** Usamos un sistema operativo libre, basado en el sistema operativo Linux, el cual es un sistema robusto, con disposición de código fuente, muy utilizado en sistemas de medio y alto desempeño (por ejemplo en servidores de red) y muy popular entre programadores y desarrolladores en todo el mundo.

- **Uso de software libre preexistente.** Realizamos una búsqueda de las aplicaciones libres para el propósito de la máquina CNC, las evaluamos y las probamos en tareas de mecanizado.

- **Desarrollo de software especializado.** Desarrollamos una aplicación para mecanizado en 2 dimensiones, a partir de dibujos en papel (que sería de fácil manejo por un artesano), que tiene un mejor desempeño en tiempo que los programas existentes para la misma tarea.

- **Capacidad de expansión.** Lo aprendido al desarrollar esta máquina permite fácilmente escalar el proyecto a otras necesidades. Por ejemplo para realizar tareas sobre áreas de trabajo mayores, como las requeridas en la fabricación de muebles.

- **Posibilidad de explotación comercial.** Teniendo como ventajas la plataforma abierta (en hardware y software), un bajo costo en hardware y nulo en software, buscamos que esta tecnología sea aprovechada para que sean comercializadas este tipo de máquinas y sobre todo usadas en mejorar la competitividad de los talleres y fabricación de guitarras en Paracho, en primera instancia. En otras palabras, buscamos que la investigación realizada en la Universidad Michoacana tenga receptores de la sociedad que puedan beneficiarse de ella y promover con ello el desarrollo del estado.

- **Máquina CNC para trabajos académicos.** La máquina CNC se utilizó para hacer las bases de robots usados en talleres impartidos en el 2013 y 2014 en la Facultad de Ingeniería Eléctrica y para participar en competencias de Robots.

5.2. Trabajos futuros

Respecto a las líneas de trabajo futuro que pueden dar continuidad a la presente investigación, tenemos las siguientes:

- Sintonización automática de los parametros T y N.
- Implementación de métodos para afinar más la aproximación del centro y radio de las circunferencias.
- Creación de algoritmos probabilistas para representar círculos y líneas.
- Creación de un editor de código G donde el usuario pueda decidir la secuencia en que se realizarán los trazos en la máquina CNC.
- Optimización del orden de los trazos dependiendo si es grabado o corte. Si la máquina CNC no sujeta la pieza por succión, para los cortes, habría que hacer primero los trazos internos y al final los trazos del contorno exterior.
- Corrección de la adquisición de las imágenes para casos donde se presenten sombras o imágenes poco contrastantes.
- Uso de imágenes en tono de grises en lugar de imágenes binarias en todos los algoritmos para tener más información. Una idea es considerar el tono de gris de cada píxel como un peso asociado a dicha posición.

Apéndice A

Máquina CNC de cuatro grados de libertad

A.1. Descripción del hardware

A.1.1. Motores de pasos

Los movimientos en cada eje son generados por motores de pasos con torque de 424 oz.in (3 N.m), tipo NEMA 23, marca Longs Motor, modelo 23HS9430. En la Figura A.1 se presenta su vista exterior y su diagrama eléctrico. La Tabla A.1 nos presenta sus principales características.

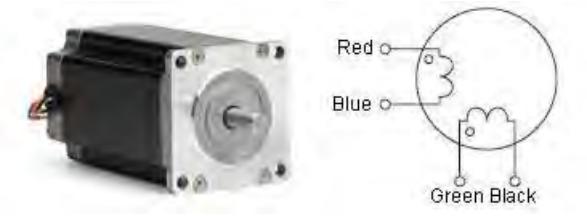


Figura A.1: Motor Nema 23 [Instruments, 2012].

b

Torque	425 oz.in
Ángulo de paso	$1.8°$
Pasos por revolución	200
Precisión angular	$\pm 3\,\%$
Fases	2
Temperatura de operación	-20 a $40°$C
Temparatura ambiente nominal	$40°$C
Carga por Eje (20,000 Horas a 1,500 rpm)	
Radial	20 lb (9.1 kg)
Empuje	6 lb (2.7 kg)
Tracción	50 lb (22.7 kg)

Tabla A.1: Características de los motores de pasos NEMA 23 [Motor, 2013].

A.1.2. Controladores de los motores

El manejo de cada motor lo hacemos mediante un controlador de motores de pasos de la marca Longs Motor, modelo DM542A, de dos fases con corrientes menores a cuatro ampers. La Figura A.2 muestra el controlador físicamente. La Tabla A.2 tiene las características del controlador: voltaje de entrada, Corriente de entrada, Corriente de salida, Consumo de energía, Temperatura de operación, entre otras.

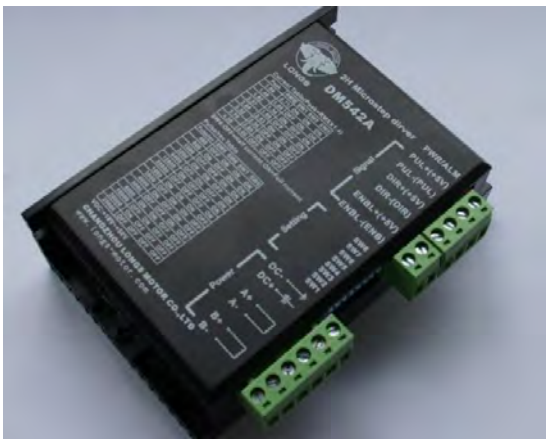


Figura A.2: Controlador de motor de pasos DM542A [Motor, 2013].

El controlador cuenta con dos bloques de terminales, “Power” y “Signal”, y un bloque de ajustes “Setting”, que contiene interruptores de configuración. En el bloque de

Voltaje de entrada	$24 - 50$ VDC
Corriente de entrada	< 4 A
Corriente de salida	1.0 A a 4.2 A
Consumo	80 W
Temperatura	-10 a $45°$
Humedad	No condensación, No gotas de agua
Gas	Prohibido gases combustibles y polvos conductores
Peso	200 G

Tabla A.2: Características del controlador DM542 [Motor, 2013].

"Power" están las conexiones para las fases del motor y la alimentación del propio controlador, la Figura A.3 ilustra el orden en que se encuentran. Mientras que en el bloque "Signal" están las terminales que permitirán el manejo de los motores, la dirección de giro, dar un paso de giro y habilitar el controlador. En la Figura A.4 se ilustra el acomodo de estas terminales. La Figura A.5 ilustra el orden y los tiempos adecuados de las señales, los pulsos de paso ("PUL") deben ser mayores de $2.5\mu S$ y con un tiempo entre pulsos mayor a $2.5\mu S$, sin sobrepasar la frecuencia máxima de 200KHz

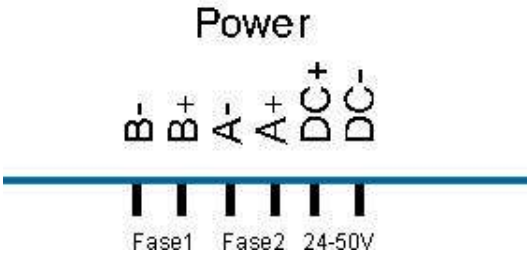


Figura A.3: Terminales del bloque "Power".

Existen dos maneras de manejar los motores, ya sea por pulsos positivos ($>$3.5 V) o negativos ($<$0.5 V). Cuando se usan pulsos positivos las terminales PUL$-$, DIR$-$ y ENBL$-$ deben ser conectadas a GND, mientras que cuando se utilizan pulsos negativos las terminales PUL$+$, DIR$+$ y ENBL$+$ deben ser conectados a VCC; en cada caso las terminales opuestas serán las de control.

El bloque "Setting" tiene los interruptores que nos servirán para determinar la resolución de los micropasos y la corriente para los motores. Los tres primeros interruptores

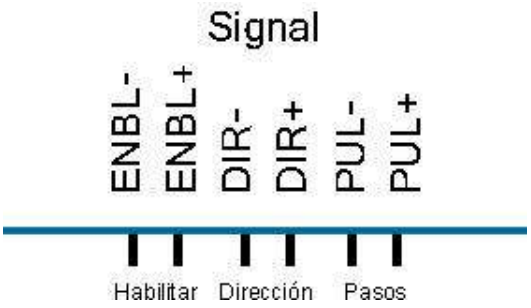


Figura A.4: Terminales del bloque “Signal”.

(SW1, SW2 y SW3) son los encargados de determinar la corriente con que funcionarán los motores, las posibles combinaciones de posiciones de los interruptores determinan la corriente (véase la Tabla A.3).

Corriente Pico	Corriente RMS	SW1	SW2	SW3
1.00A	0.71A	ON	ON	ON
1.46A	1.04A	OFF	ON	ON
1.91A	1.36A	ON	OFF	ON
2.37A	1.69A	OFF	OFF	ON
2.84A	2.03A	ON	ON	OFF
3.31A	2.36A	OFF	ON	OFF
3.76A	2.69A	ON	OFF	OFF
4.20A	3.00A	OFF	OFF	OFF

Tabla A.3: Configuración de la corriente que proporciona el controlador a los motores [Motor, 2013].

El interruptor SW4 etiquetado como “Standstill Current Setting” al colocarlo en ON su función es mantener la corriente que está seleccionada por los primeros interruptores y al colocarlo en OFF hará que, cuando los motores estén sin movimiento, la corriente disminuya en un 50 % . Por último, la combinación de estados de los interruptores SW5, SW6, SW7 Y SW8 determinan la resolución de los micropasos mediante la Tabla A.4.

A.1.3. Interfaz del puerto paralelo

Para comunicarnos con los controladores de los motores lo hacemos mediante la interfaz tipo X220926, de la marca Longs Motor; el puerto paralelo de una computadora

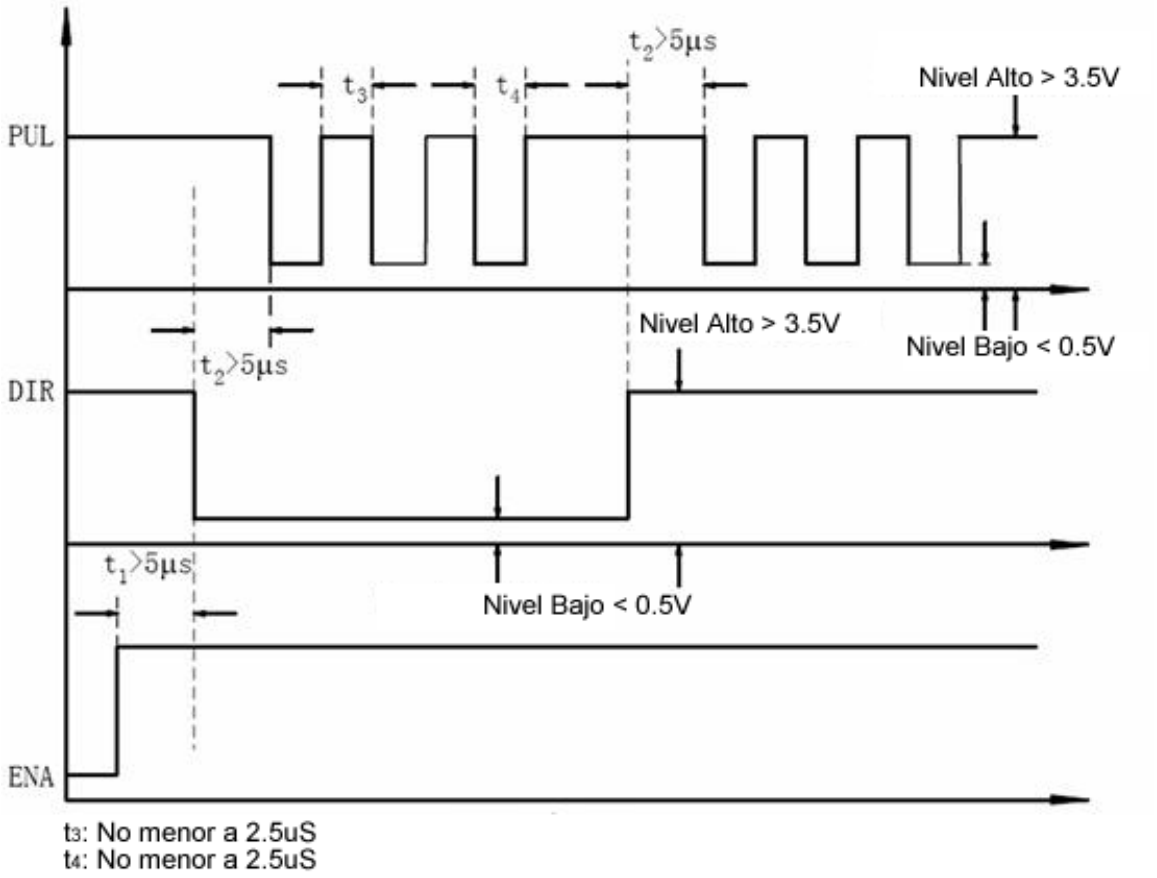


Figura A.5: Descripción de las señales de control [Motor, 2013].

será encargado de enviar las señales necesarias para manejar la máquina de CNC. A través del puerto paralelo, la computadora puede tener hasta 12 señales digitales de salida y 5 señales digitales de entrada. En la Figura A.6 se muestra la interfaz físicamente y en la Tabla A.5 están sus características. Esta tarjeta se alimenta con 5 volts, los cuales los obtenemos del puerto USB de la computadora.

La interfaz protege la computadora personal de posibles daños por picos o cortocircuitos en los controladores. Hace un aislamiento por medio de optoacopladores, transmitiendo la información mediante luz, sin que exista una conexión eléctrica. Permite alimentación independiente para la sección a controlar. En caso de que sólo usemos una fuente de voltaje, colocamos los jumpers JN1 y JN2 que unirán las tierras y la alimentación en todo el dispositivo. La Figura A.7 nos muestra la ubicación de los jumpers.

Micropaso	Paso/Rev	SW5	SW6	SW7	SW8
2	400	OFF	ON	ON	ON
4	800	ON	OFF	ON	ON
8	1600	OFF	OFF	ON	ON
16	3200	ON	ON	OFF	ON
32	6400	OFF	ON	OFF	ON
64	12800	ON	OFF	OFF	ON
128	25600	OFF	OFF	OFF	ON
5	1000	ON	ON	ON	OFF
10	2000	OFF	ON	ON	OFF
20	4000	ON	OFF	ON	OFF
25	5000	OFF	OFF	ON	OFF
40	8000	ON	ON	OFF	OFF
50	10000	OFF	ON	OFF	OFF
100	20000	ON	OFF	OFF	OFF
125	25000	OFF	OFF	OFF	OFF

Tabla A.4: Selección de la resolución de los micropasos [Motor, 2013].

Voltaje de alimentación	5 VDC
DB25 Terminales de salida	P1, P2, P3, P4, P5, P6, P7, P8, P9, P14, P16, P17
DB25 Terminales de entrada	P10, P11, P12, P13, P15
DB25 Terminal GND	P18-P25
Construcción	Optoacopladores

Tabla A.5: Descripción de la interfaz [Motor, 2013].

A.1.4. Fuente de potencia

La alimentación es proporcionada por una fuente conmutada de 350 Watts, de la marca Longs Motor, modelo S-350-24, que proporciona 24VDC/14.6A. Sus características se encuentran en la Tabla A.6.

A.2. Conexiones

Unimos mediante cables de distintos colores cada uno de los dispositivos que conforman nuestra máquina de CNC. La FiguraA.8 ilustra las conexiones, los motores son conectados a los controladores, los controladores se comunican con la interfaz del puerto

Figura A.6: Interfaz del tipo paralelo tipo X220926.

Voltaje DC	24 V
Tolerancia	$\pm 1\,\%$
Rango de corriente	0 a 14.6 A
Rizado&Ruido Max.	150 mVpp
Regulación lineal	$\pm 0.5\,\%$
Potencia	350.4 W
Eficiencia	$83\,\%$
Rango de voltaje	4.5A/115V 2.5A/230V
Temperatura de trabajo	-20 a $85°$C
Peso	0.9 Kg

Tabla A.6: Características de la fuente de alimentación [Motor, 2013].

paralelo y a la interfaz están conectados interruptores. A continuación se describe con detalle las conexiones.

En la interfaz del puerto paralelo colocamos los jumpers (JN1 y JN2) para compartir en la etapa de control y de potencia la fuente de 5 V, que se obtiene del conector USB. En cada controlador de los motores de pasos ajustamos el bloque “Setting” mediante los interruptores. Como los motores NEMA 23 que estamos usando tienen una corriente nominal de 3 A, optamos por trabajar con corrientes pico de 2.84 A y corrientes RMS de 2.03 A, colocando en cada bloque “Setting” los interruptores SW1 y SW2 en ON, mientras que SW3 en OFF.

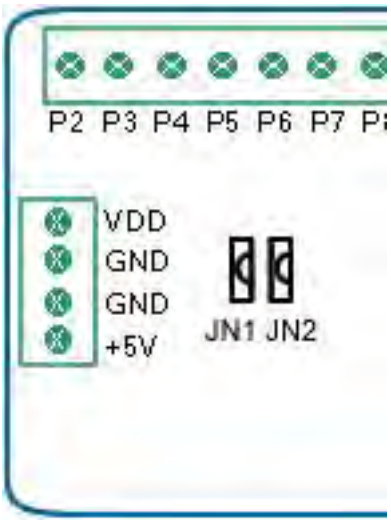


Figura A.7: Localización de los jumpers que permiten una alimentación compartida.

Posicionamos en OFF el interruptor SW4 etiquetado como “Standstill Current Setting” para disminuir la corriente de los motores en un cincuenta por ciento cuando no exista movimiento. En nuestro caso los motores no requerirán ejercer un torque elevado cuando no se estén moviendo y por tanto ahorramos energía.

Determinamos la resolución de los micropasos, sabiendo que entre menos pasos tengamos mayor será el torque; le damos prioridad al mayor torque, seleccionando la configuración de 2 micropasos (400 pasos en una revolución) en los interruptores del cinco al ocho. La tabla A.7 muestra la configuración de los controladores de los motores de pasos, la cual es la misma para los cuatro motores.

El conector USB nos permite obtener los 5 volts requeridos para alimentar la interfaz del puerto paralelo.

SW1	SW2	SW3	SW4	SW5	SW6	SW7	SW8
ON	ON	OFF	OFF	OFF	ON	ON	ON
Corriente pico de 2.84A			Disminuir corriente cuando no exista movimiento	Micropasos de 2 (400 pasos/rev)			

Tabla A.7: Selección de los parámetros del controlador.

En el bloque de “Signal” de cada controlador de los motores de pasos usamos

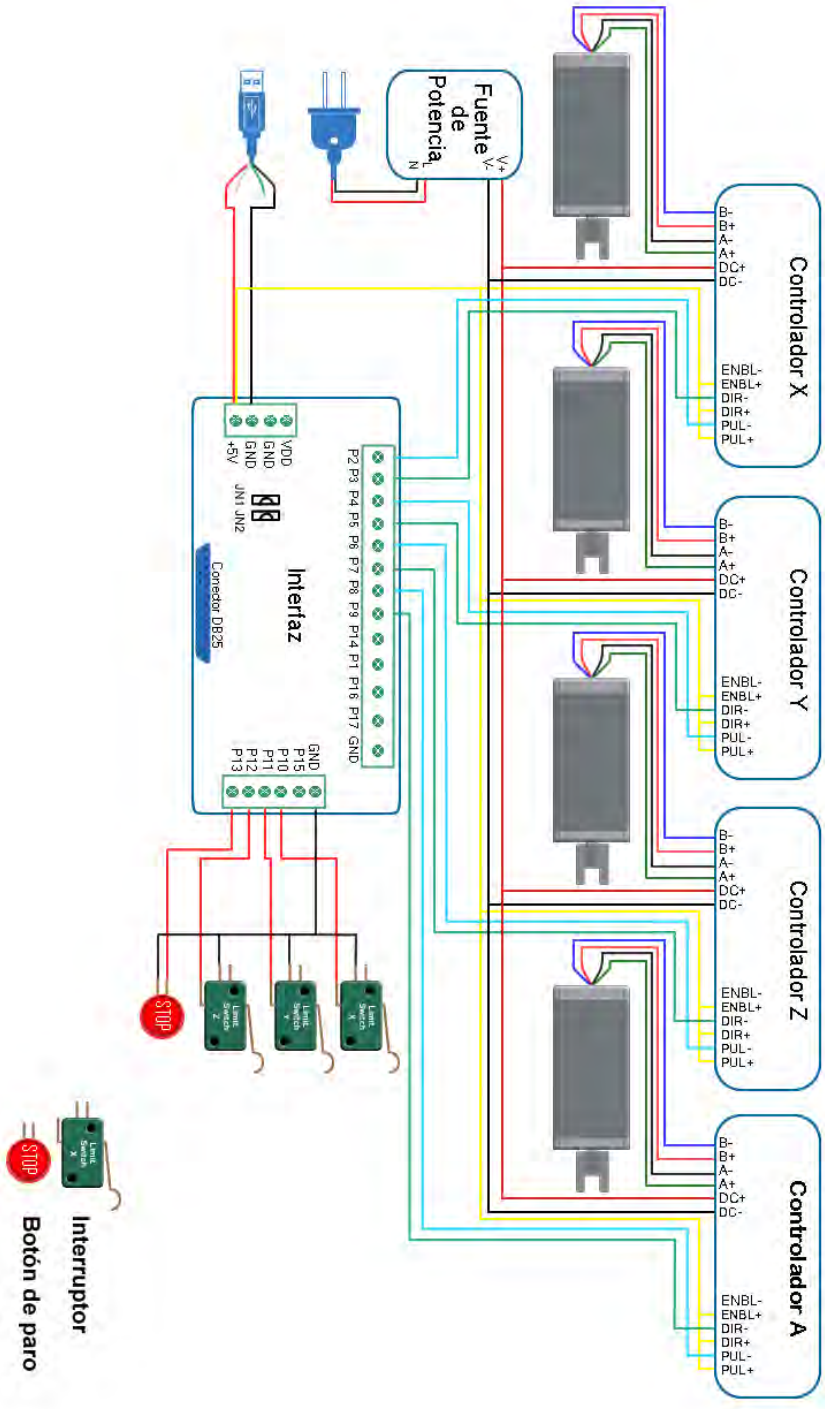


Figura A.8: Conexión de los dispositivos en la máquina de Control Numérico Computarizado.

pulsos negativos (voltajes menores a 0.5 V) para manejar los motores, por ello las terminales ENBL$+$, PUL$+$ y DIR$+$ los mantenemos energizados. La terminal ENBL$-$ se deja sin conectar para siempre contar con los motores habilitados. Las señales que llevamos a la interfaz para poder controlarlas son PUL$-$ y DIR$-$.

Adicionalmente colocamos un botón de paro Normalmente Cerrado (NC) para emergencias y tres interruptores de desplazamiento límite Normalmente Abiertos (NA), uno para el eje X, otro para el eje Y y otro para el eje Z. Estas señales las conectamos a la interfaz del puerto paralelo en las terminales de entrada.

A.3. Software

Para poner en marcha la máquina de CNC y controlarla usamos software libre. En una computadora personal, instalamos la distribución del sistema operativo en tiempo real denominado LinuxCNC; es un software para el control computarizado de máquinas, robots u otros dispositivos de automatización, puede controlar servomotores, motores paso, relés, fresadoras, tornos, cortadoras de plasma y otros dispositivos relacionados. LinuxCNC es software libre publicado bajo los términos de GNU GPLv2 [LinuxCNC, 2013].

LinuxCNC es un descendiente del Enhanced Machine Controller (EMC), creado por el National Institute of Standards and Technology (NIST), que es una agencia del Departamento de Comercio del gobierno de los Estados Unidos.

Para iniciar a trabajar con LinuxCNC utilizamos un programa asistente “Wizard” para crear una configuración de la mesa CNC donde se definen los parámetros de nuestra máquina. Al ejecutar el programa, nos mostrará la ventana de la Figura A.9 y enseguida la ventana de la Figura A.10 donde le indicaremos que crearemos una nueva configuración.

Enseguida, como se muestra en la Figura A.11, elegimos un nombre para la configuración de nuestra máquina, los ejes a usar y las unidades de medida, en nuestro caso le nombramos DEPFIE_MesaCNC4GDL, con los cuatro ejes XYZA y usando pulgadas.

A continuación especificamos las características del controlador de motores de pasos. Los controladores DM542A requieren un tiempo para los pulsos de paso (“Step time”) mayor de $2.5\mu S(2500\eta S)$ y un tiempo entre pulsos mayor a $2.5\mu S(2500\eta S)$, sin so-

Figura A.9: El asistente “Wizard” de LinuxCNC.

brepasar la frecuencia máxima de 200KHz.

Elegimos un tiempo de $5000\eta S$ tanto para el pulso de un paso como para el tiempo entre pulsos (“Step Space”).

Enseguida, mientras que el tiempo de un pulso de dirección (“Direction Hold”) sea mayor $5\mu S(5000\eta S)$ y el tiempo que debe haber entre el último pulso de paso y un cambio de dirección (“Direction Setup”) debe ser mayor a $5\mu S(5000\eta S)$, elegimos $5500\eta S$ para ambos parámetros, como lo vemos en la Figura A.12.

Mantenemos la dirección del puerto paralelo 0x378 y para “Base Period Maximum Jitter” corremos “Test Base Period Jitter”, que es la “Latency Test”, una prueba que mide el intervalo de tiempo entre una solicitud y la respuesta del hardware.

Ejecutamos varios programas mientras se ejecuta la prueba y así, determinar los peores casos de tiempos de respuesta de la computadora.

En la Figura A.13 se muestra esta prueba en una computadora con procesador

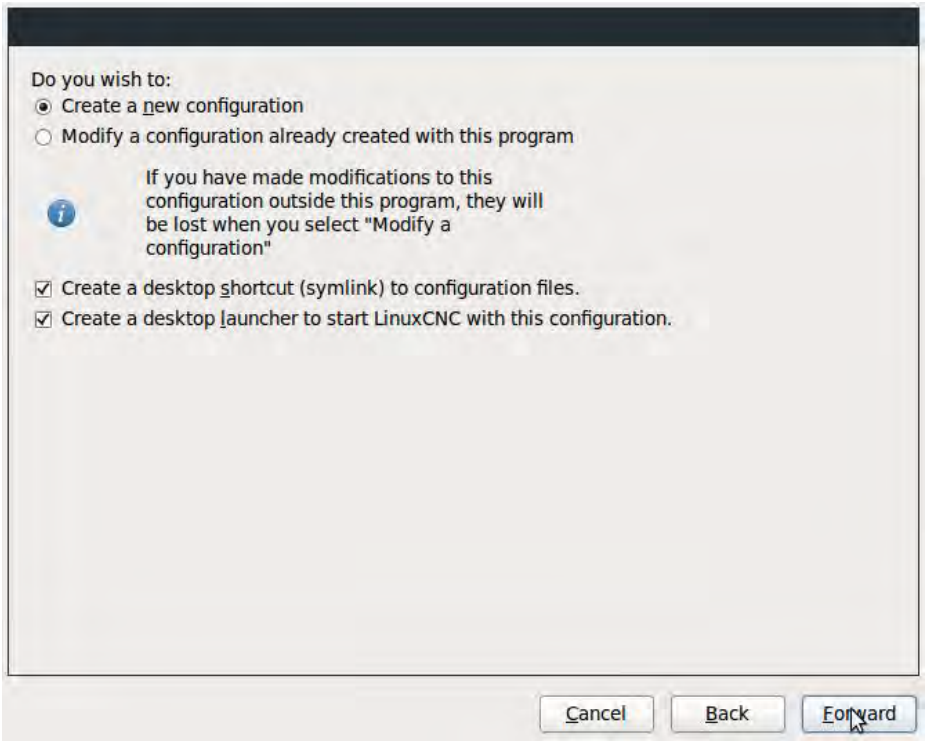


Figura A.10: Creación de una nueva configuración.

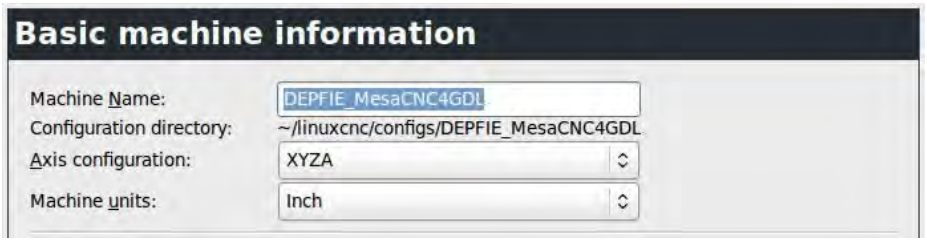


Figura A.11: Primera parte de la información básica de la máquina.

de la marca Intel, modelo Pentium 4 a 3Ghz, con 38GB en disco duro y 1Gb de memoria RAM. De las mediciones realizadas nos interesa el parámetro “Max Jitter” que en nuestro caso es de $25194\eta S$, éste será el valor de “Base Period Maximum Jitter”, lo colocamos como en la Figura A.14.

Las configuraciones avanzadas no serán utilizadas, dejamos intacta la ventana de la Figura A.15 y avanzamos.

Enseguida indicamos cómo es usada cada terminal del puerto paralelo, según las

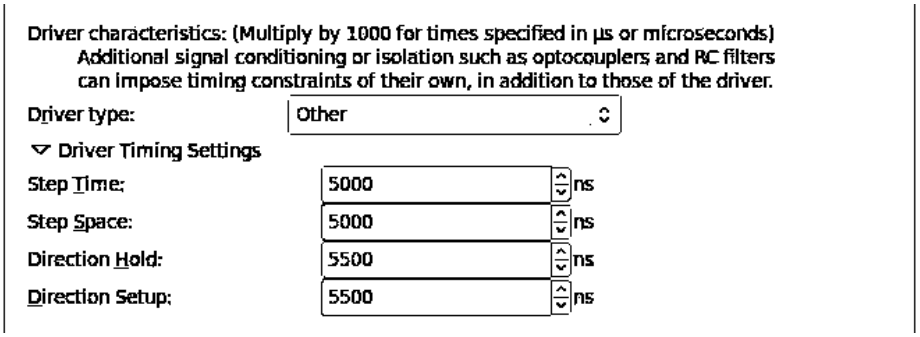


Figura A.12: Selección de los tiempos con que funcionará la máquina.

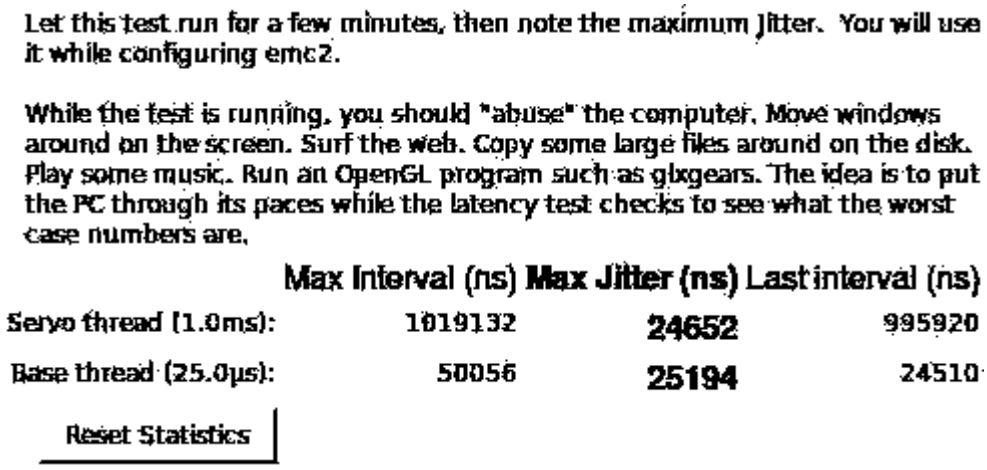


Figura A.13: Ejecución del “Test Base Period Jitter”.

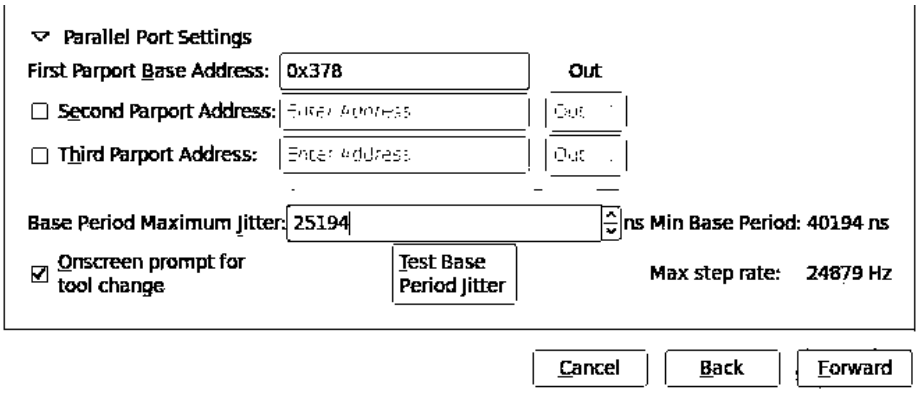


Figura A.14: Última parte de la información básica de la máquina.

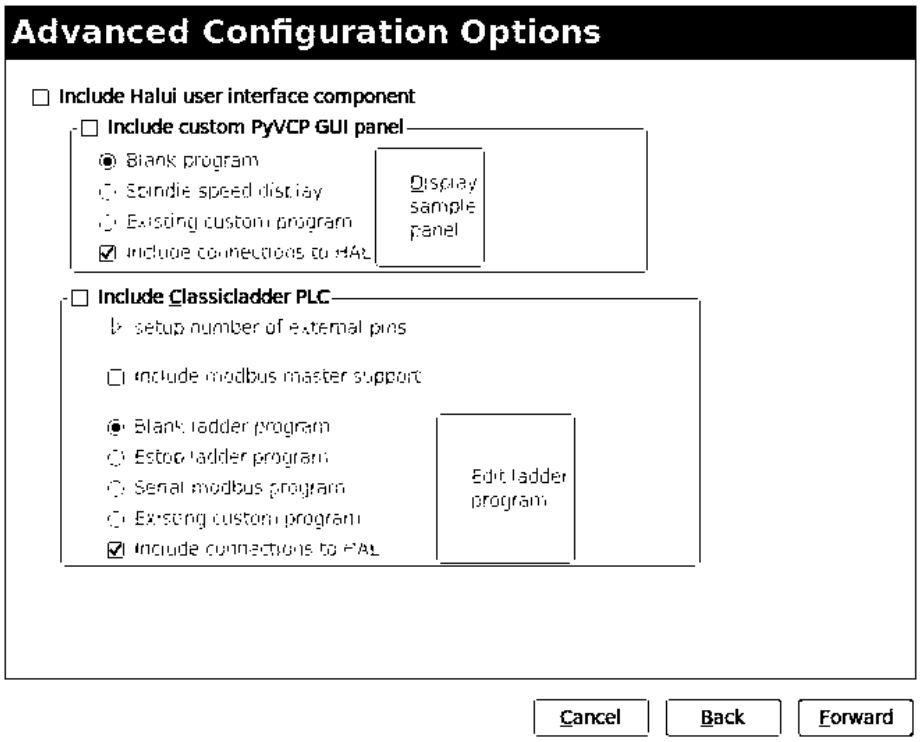


Figura A.15: Plantilla a mantener intacta.

conexiones de la interfaz y los colocamos como inversos ya que usamos pulsos negativos (voltajes menores a 0.5V); las terminales de dirección las mantenemos sin invertir para lograr la disposición típica de ejes, como se muestra en la Figura A.16. La ventana con los valores de la configuración de las terminales la apreciamos en la Figura A.17.

A continuación configuramos los parámetros de cada eje. Seleccionamos los parámetros para manejar los motores, una resolución de 2 micropasos para 200 pasos en una revolución; los tornillos sinfín dan 5 giros para lograr una pulgada de desplazamiento de la herramienta, con ello llenamos el valor del parámetro “Leadscrew Pitch”. Asimismo como no contamos con ningún acoplamiento en los ejes X, Y y Z el parámetro “Pulley teeth” lo determinamos como de 1.0:1.0. En la Figura A.18 vemos estos parámetros.

La velocidad y aceleración máxima con que funcionarán los motores es $1in/seg$ y $2in/seg^2$ respectivamente, de tal forma que se pueda obtener un torque alto, movimientos suaves y estables y se evite que los motores patinen. El origen es el cero y el desplaza-

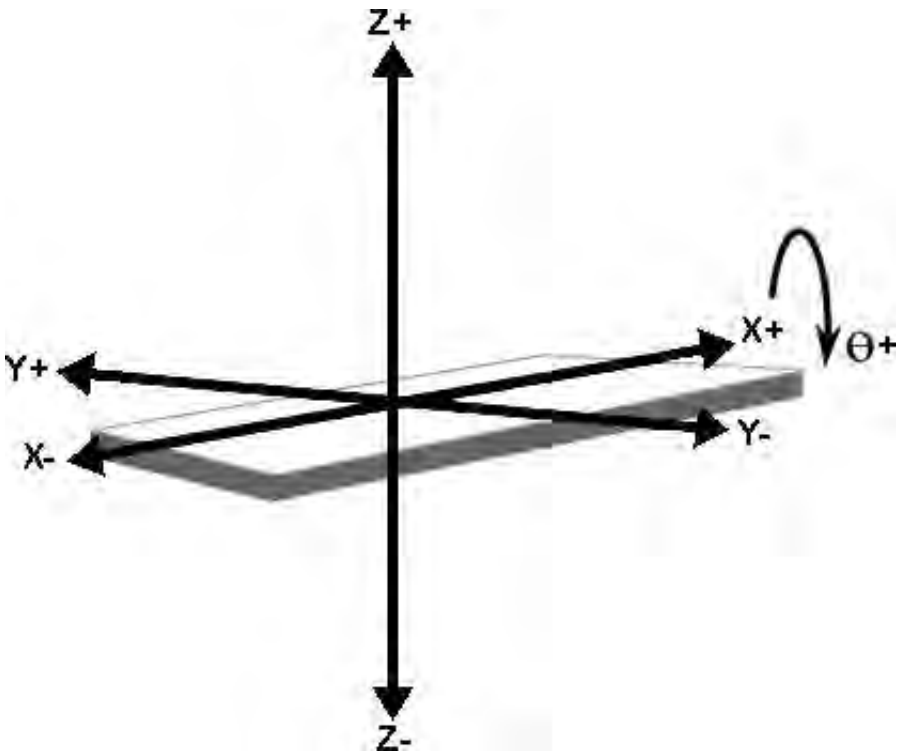


Figura A.16: Representación de la disposición típica de los ejes.

miento que tendrá cada eje según las dimensiones de nuestra máquina es en X de -8 a 8in, en Y de -5 a 5in y en Z de -1 a 1in. Éste es el espacio de trabajo para que la máquina de CNC cuente con todos los cuadrantes. En la Figura A.19 se muestra la configuración del eje X, en la Figura A.20 la del eje Y y en la Figura A.21 la del eje Z.

Finalmente en la Figura A.22 se ilustra la configuración para el cuarto eje, donde se indica que el motor tiene un acoplamiento, un reductor de 2:1, una máxima velocidad de $1800.0°/seg$ y aceleración de $10°/seg^2$.

Una vez creada la configuración de nuestra máquina, iniciamos LinuxCNC con ella, cargamos el archivo prueba_inicial.ngc y activamos la comunicación con la máquina, como se muestra en la Figura A.23.

Para ejecutar instrucciones es necesario “homear”, llevar la máquina a la posición de origen de todos los motores, para ello posicionamos la herramienta, en cada eje en el centro de su región de desplazamiento y seleccionamos esa posición como posición de

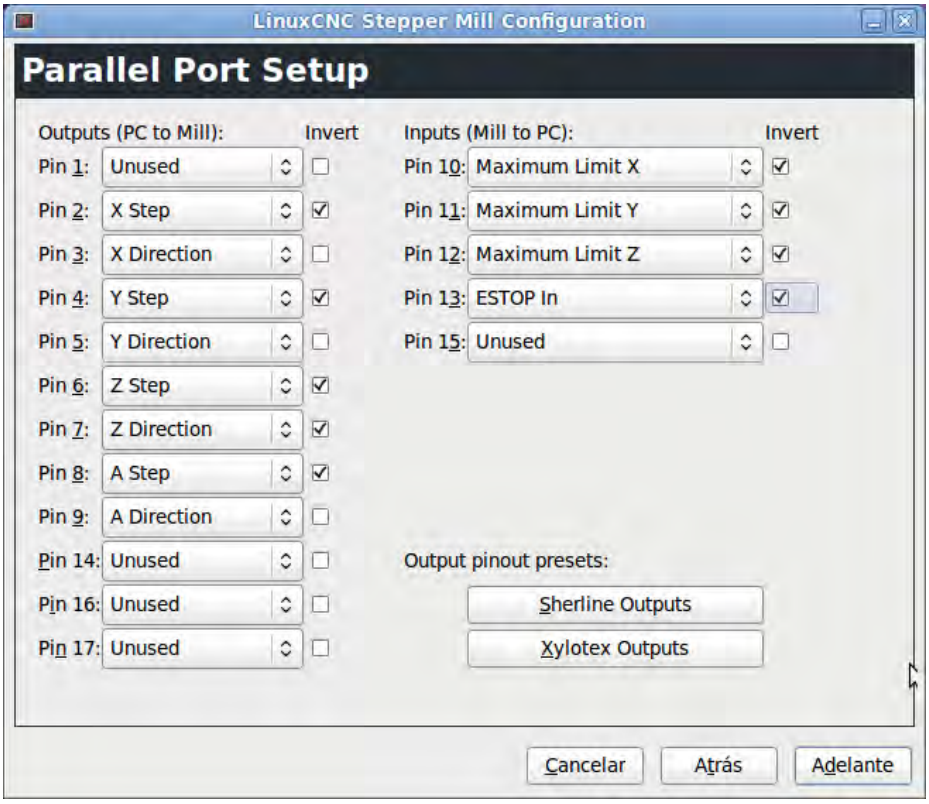


Figura A.17: Configuración de las terminales a manejar.

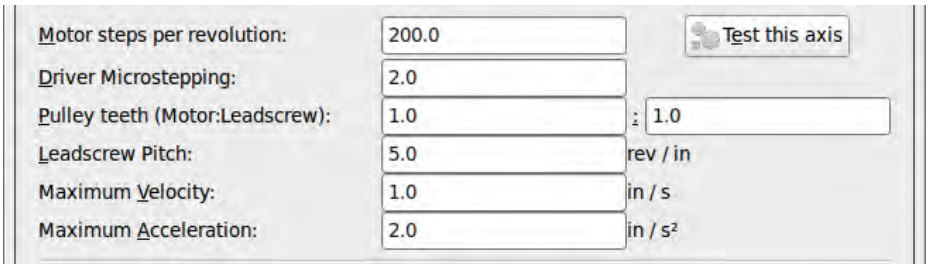


Figura A.18: Configuración para los ejes principales X, Y y Z.

origen, lo vemos en la Figura A.24. Al mover la herramienta en cada eje debe coincidir con el sentido típico de ejes, como en la Figura 2.1.

El archivo prueba_inicial.ngc contiene el siguiente programa en código G y su objetivo es probar que las articulaciones de la máquina estén correctamente configuradas.

G90 (Distancias absolutas)

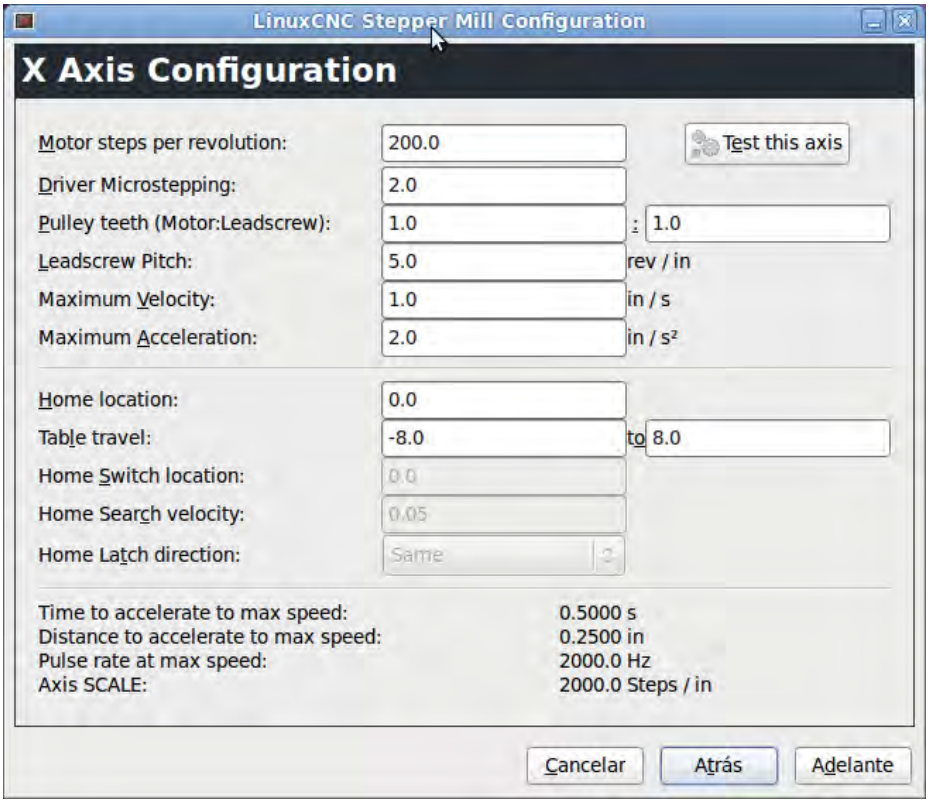


Figura A.19: Configuración del eje X.

G20 (Unidades en pulgadas)

G01 X1 F8 (Movimiento en X una pulgada a $8in/min$)

G01 Y1 (Movimiento en Y una pulgada)

G01 Z1 (Movimiento en Z una pulgada)

G01 A360 F1080 (Movimiento en A una vuelta a $1080°/min$)

G00 X0 Y0 Z0 A0 (Movimiento rápido al origen de los ejes XYZA)

M30 (Fin del programa)

La Figura A.25 ilustra la ejecución del programa, la máquina tiene que mover la herramienta una pulgada en el eje X, en seguida una pulgada en el eje Y, a continuación una pulgada en el eje Z, todos estos movimiento a una velocidad de $8in/min$, después el eje A girar una vuelta a una velocidad de $1080°/min$. Una vez hecho este recorrido la herramienta regresará a la posición de origen a la máxima velocidad de la máquina.

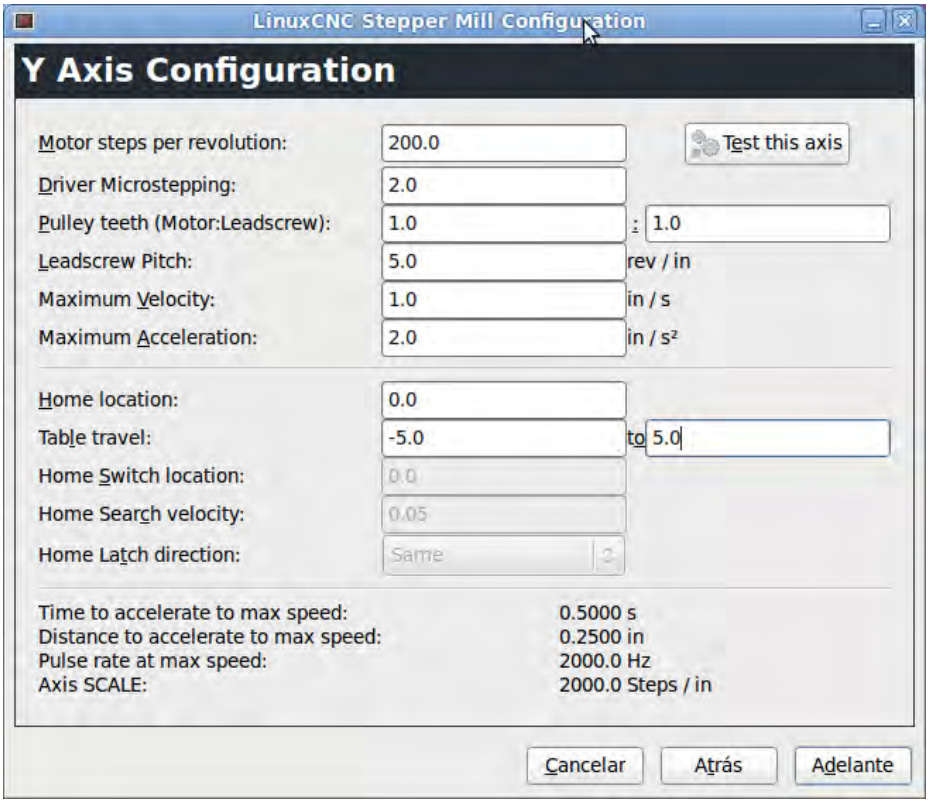


Figura A.20: Configuración del eje Y.

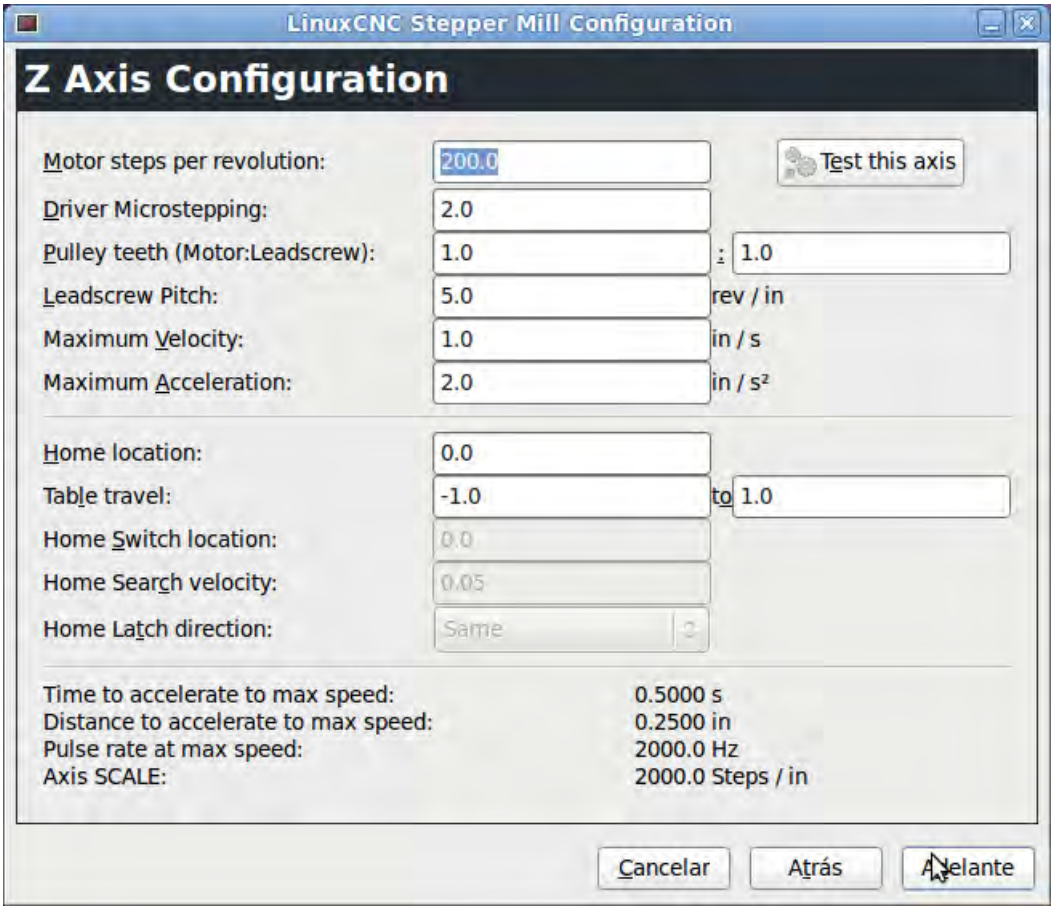


Figura A.21: Configuración del eje Z.

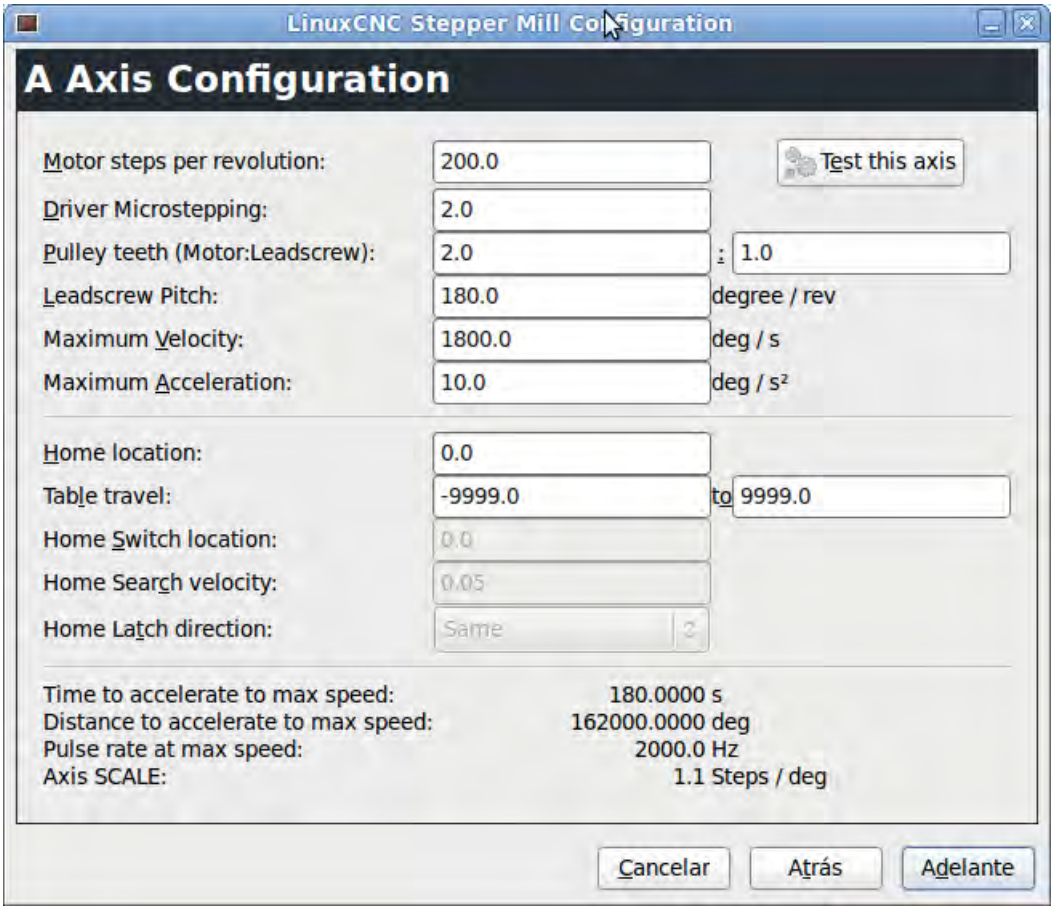


Figura A.22: Configuración del eje A.

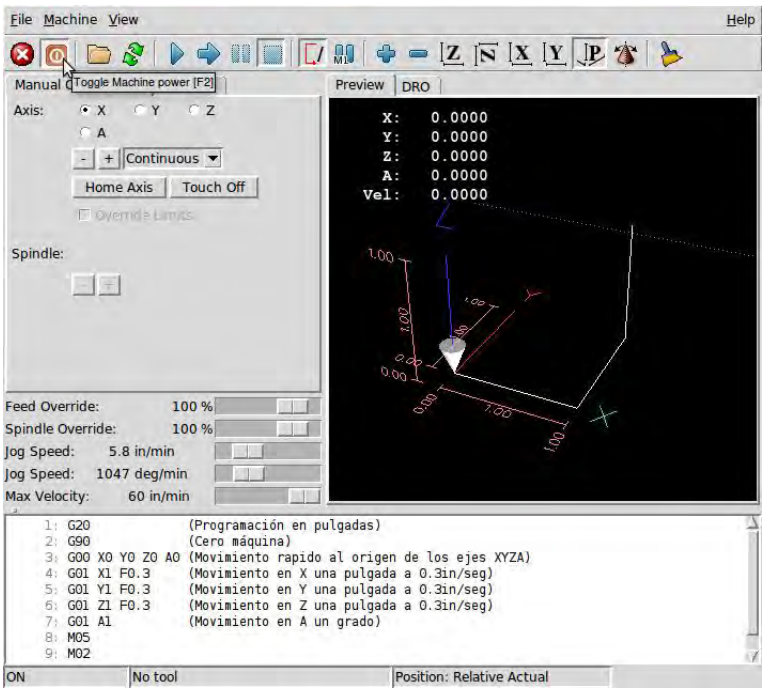


Figura A.23: LinuxCNC con el programa prueba_inicial.ngc y activando la comunicación con la máquina.

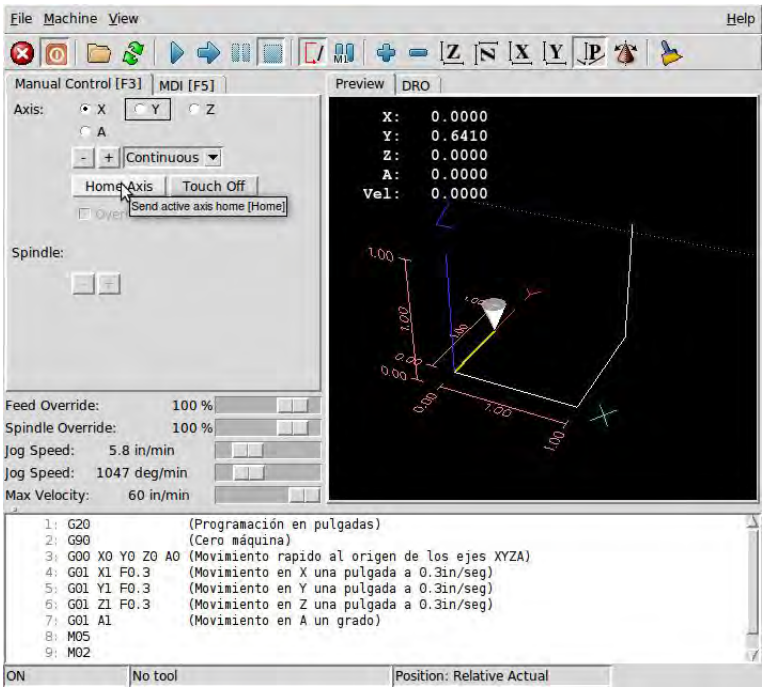


Figura A.24: Determinando el origen de cada eje.

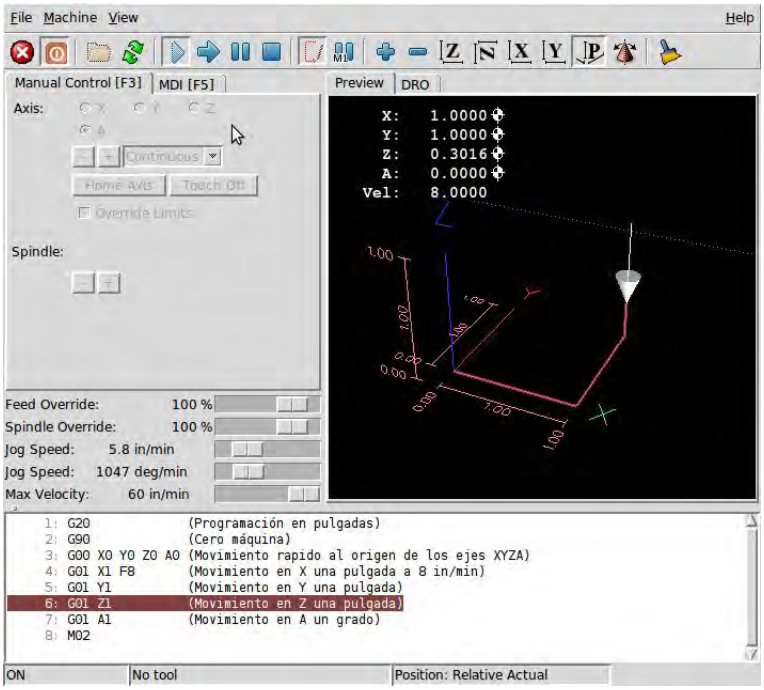


Figura A.25: Ejecución del programa en código G.

Apéndice B

Máquina CNC de tres grados de libertad

B.1. Descripción del hardware

B.1.1. Motores de pasos

Cada movimiento de la máquina es generado por motores de pasos con torque de 400 Oz In, tipo NEMA 23 de la marca Probotix, modelo HT23-400-8. En la figura B.1 está su vista exterior y su composición interna. En la Tabla A.1 nos presenta sus principales características.

Figura B.1: Motor de pasos tipo NEMA 23 de la marca Probotix [Probotix, 2013].

B.1.2. Controladores de los motores

Cada uno de los tres motores de la máquina está al mando de un controlador de la marca Probotix, modelo ProboStepVX, controlador de motor de pasos, unipolar y para corrientes menores de 3A. La Figura B.2 muestra al controlador físicamente. En Tabla B.1 están las características de voltaje, corriente, potencia y temperatura de trabajo.

Figura B.2: Controlador de motor de pasos modelo ProboStepVX [Probotix, 2013].

Voltaje de entrada	12 a 44VDC
Corriente de entrada	0.5 a 3A
Corriente de salida	$<$3A
Potencia nominal de salida	132W
Temperatura	Necesita disipador externo

Tabla B.1: Características del controlador ProboStepVX [Probotix, 2013].

Este controlador cuenta con tres bloques de conectores “Stepper Motor”, “PBX Header” y “Power Input”. En el primer bloque están las conexiones para las fases del motor, en la Figura B.3 las podemos apreciar.

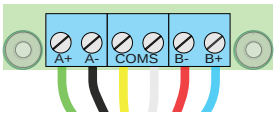


Figura B.3: Terminales del bloque “Stepper Motor” [Probotix, 2013].

En el bloque “PBX Header” se encuentran las terminales que recibirán las señales de control etiquetadas como “Direction” y “Step”, que servirán para indicar la dirección y mover el motor un paso, respectivamente. El controlador funciona con lógica negativa. En la Figura B.4 se ilustra el diagrama del conector de este bloque.

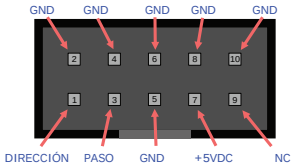


Figura B.4: Diagrama de las terminales del bloque “PBX Header” [Probotix, 2013].

El ancho mínimo del pulso para “Step” debe ser de $5\mu S$ y su frecuencia máxima de 40 kHz.

Finalmente en el bloque “Power Input” están las terminales para alimentar el controlador, GND y VCC.

Cada controlador cuenta con un potenciómetro que modifica un voltaje de referencia etiquetado como “VREF Trimmer”, el cual determina la corriente que alimentará al motor, mediante la Tabla B.2.

Voltaje de referencia	Corriente de salida
0.07V	0.5A
0.15V	1A
0.22V	1.5A
0.29V	2A
0.36V	2.5A
0.44V	3A

Tabla B.2: Configuración de la corriente que proporciona el controlador a la salida [Probotix, 2013].

También cuenta con una sección de interruptores para seleccionar micropasos, consultando la Tabla B.3. El modo fijo 8 proporciona el 70 % de corriente en ambas fases y sus movimientos son suaves; mientras que el modo fijo F proporciona el 100 % de corriente entre las fases y proporciona un poco más de par.

La localización del potenciómetro y los interruptores se muestra en la Figura B.5.

SW1	SW2	SW3	Paso
ON	ON	ON	Completo (modo fijo 8)
OFF	ON	ON	Completo (modo fijo F)
ON	OFF	ON	Medio
OFF	OFF	ON	Medio (modo fijo F)
ON	ON	OFF	Un cuarto
OFF	ON	OFF	Un octavo
ON	OFF	OFF	Un dieciseisavo
OFF	OFF	OFF	“Sleep”

Tabla B.3: Selección de la resolución de los pasos del controlador [Probotix, 2013].

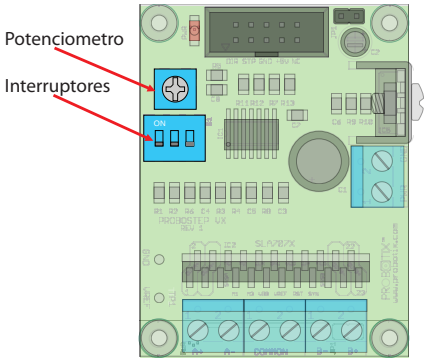


Figura B.5: Potenciómetro e interruptores en el controlador [Probotix, 2013].

B.1.3. Interfaz del puerto paralelo

El puerto paralelo de una computadora personal será el encargado de enviar las señales para manejar la máquina CNC. La comunicación con la máquina y la computadora se hace mediante la interfaz PBX-RF, de la marca Probotix; la cual permite enlazar las 17 señales disponibles del puerto paralelo, 12 señales digitales de salida y 5 señales digitales de entrada, de una computadora personal con la máquina de CNC. La Figura B.6 muestra la interfaz físicamente y en la Tabla B.4 están sus características.

La interfaz protege la computadora personal contra posibles daños por picos o cortocircuitos en los controladores. La manera en que realiza el aislamiento es muy similar a la técnica que usa la interfaz de la máquina de cuatro grados de libertad, sólo que ahora esta interfaz usa chips de radiofrecuencia para el aislamiento, parecido a aisladores ópticos, salvo que aquí se utilizan ondas de radio para enviar señales a través del plano

Figura B.6: Interfaz PBX-RF [Probotix, 2013].

Voltaje de alimentación	5 VDC
DB25 Terminales de salida	P1, P2, P3, P4, P5, P6, P7, P8, P9, P14, P16, P17
DB25 Terminales de entrada	P10, P11, P12, P13, P15
DB25 Terminal GND	P18-P25
Construcción	Acopladores RF

Tabla B.4: Descripción de la interfaz PBX-RF [Probotix, 2013].

de aislamiento en lugar de luz.

Las señales en el puerto paralelo serán las mostradas en la Tabla B.5.

La interfaz del puerto paralelo permite el empleo de tres fuentes de alimentación independientes; para su propia operación, para los voltajes de puerto paralelo y para los voltajes del controlador.

La alimentación de la interfaz es suministrada por medio del puerto USB de la computadora.

En caso de usar sólo una fuente y compartir la alimentación propia con todo el dispositivo desconectamos el jumper JP6.

Existen siete jumpers en la interfaz, la Figura B.7 muestra su localización y en la Tabla B.6 se lista cada uno de ellos con su respectiva función.

Terminal	Señal
1	Habilitar A
2	Paso X
3	Dirección X
4	Paso Y
5	Dirección Y
6	Paso Z
7	Dirección Z
8	Paso A
9	Dirección A
10	Botón Paro
11	Limite Z
12	Limite Y
13	Limite X
14	Habilitar X
15	Entrada Auxiliar
16	Habilitar Y
17	Habilitar Z
18-25	GND

Tabla B.5: Descripción de las señales del puerto paralelo.

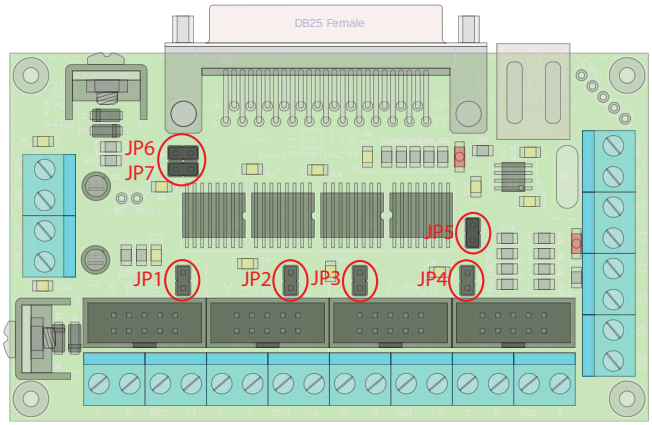


Figura B.7: Localización de los siete jumpers en la interfaz [Probotix, 2013].

B.1.4. Fuente de Potencia

La fuente de alimentación que se encargará de la energía de los motores y los controladores, es una fuente conmutada de 150W. En la Tabla B.7 están sus característi-

Jumper	Función
JP1	Alimentar controlador X
JP2	Alimentar controlador Y
JP3	Alimentar controlador Z
JP4	Alimentar controlador A
JP5	Habilitar entradas Pull-Ups
JP6	Usar regulador de voltaje en aislamiento
JP7	Habilitar entradas

Tabla B.6: Jumpers en la interfaz PBX-RF [Probotix, 2013].

cas.

Voltaje DC	24 V
Máxima carga	0 a 6.5A
Potencia	150W
Voltaje de entrada	115VCA/220VCA
Protección	Auto-recuperción %
Temperatura de trabajo	-20 a 80°C
Peso	0.73 Kg

Tabla B.7: Características de la fuente de alimentación [Probotix, 2013].

B.2. Conexiones

Conectamos cada uno de los dispositivos mediante cables de colores y cable IDC de 10 terminales, como se ilustra en la Figura B.8.

La interfaz está alimentada usando el puerto USB de la computadora personal y la energía para las señales de los controladores la adquirimos de la fuente conmutada de 150W. Colocamos todos los jumper (JP1, JP2, JP3, JP4, JP5, JP6 y JP7) en ON para tener habilitados los controladores, contar con entradas tipo Pull-Ups, habilitar el regulador de voltaje para las señales de los controladores y habilitar las entradas. La ubicación de los jumpers se muestra en la Figura B.7.

En cada controlador ajustamos a 0.44V el “VREF Trimmer” mediante el potenciómetro, para tener una corriente de 3A para los motores. El bloque de interruptores los

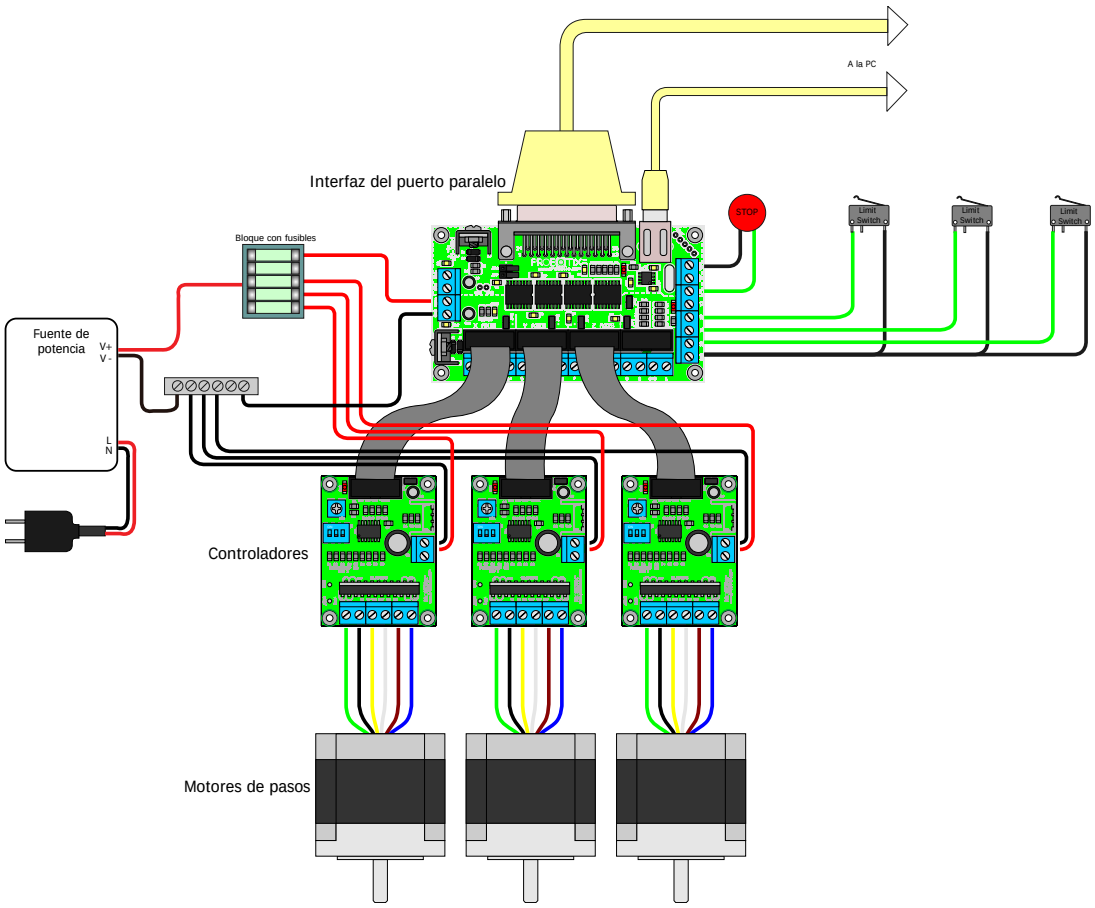


Figura B.8: Conexión de los dispositivos [Probotix, 2013].

colocamos como la Tabla B.8, de tal manera que los pasos se conviertan en medios pasos, para tener 400 pasos en una revolución.

Interruptor	Estado
SW1	On
SW2	Off
SW3	On

Tabla B.8: Posición de los interruptores en la interfaz.

También agregamos un botón de paro NC (Normalmente Cerrado) y tres interruptores limite NA(Normalmente Abiertos) como entradas a la interfaz.

B.3. Software

El manejo de esta máquina es muy similar a la máquina de cuatro grados de libertad, usamos el software libre LinuxCNC. A continuación describimos parte de su configuración.

Cambiaremos la sección de “Driver Timing Settings” en base a los tiempos en que funciona el controlador ProboStepVX. El ancho de pulso para “Step” debe ser $\geq 5000\eta S$ y el tiempo entre pulsos (“Step Space’) que genere una frecuencia menor a 40KHz. Elegimos “Step” de $13000\eta S$ y “Step Space’ de $13000\eta S$, para así tener una frecuencia de 38.4KHz.

Para el pulso de dirección (“Direction Hold”) y para el tiempo entre el último pulso de paso y un cambio de dirección (“Direction Setup”) usamos $5000\eta S$. Como se aprecia en la Figura B.9.

La configuración en que se usará el puerto paralelo es en base a la Tabla B.5, como se muestra en la Figura B.10. Las terminales son invertidos ya que los controladores funcionan con lógica negativa.

Los parámetros de cada eje son los mismos que usamos para la máquina de cuatro grados. Ejecutamos el mismo archivo de prueba y verificamos que se realicen las instrucciones del programa con éxito.

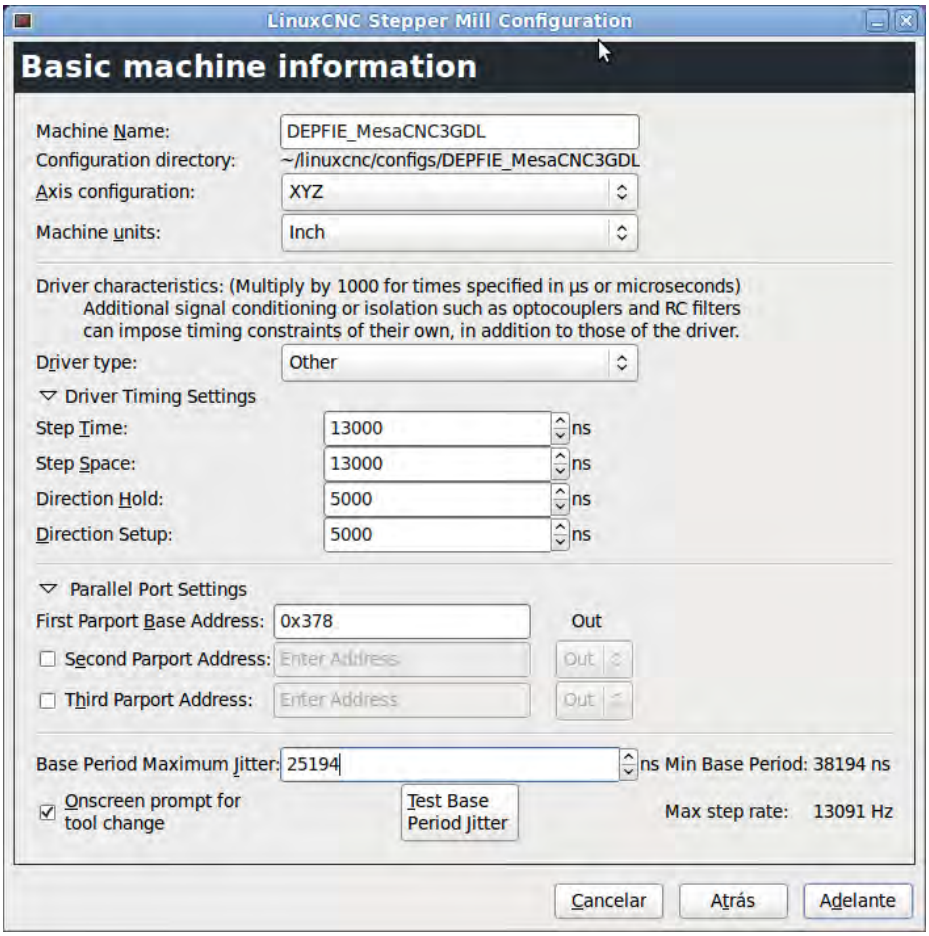


Figura B.9: Configuración de los tiempos para las señales de control.

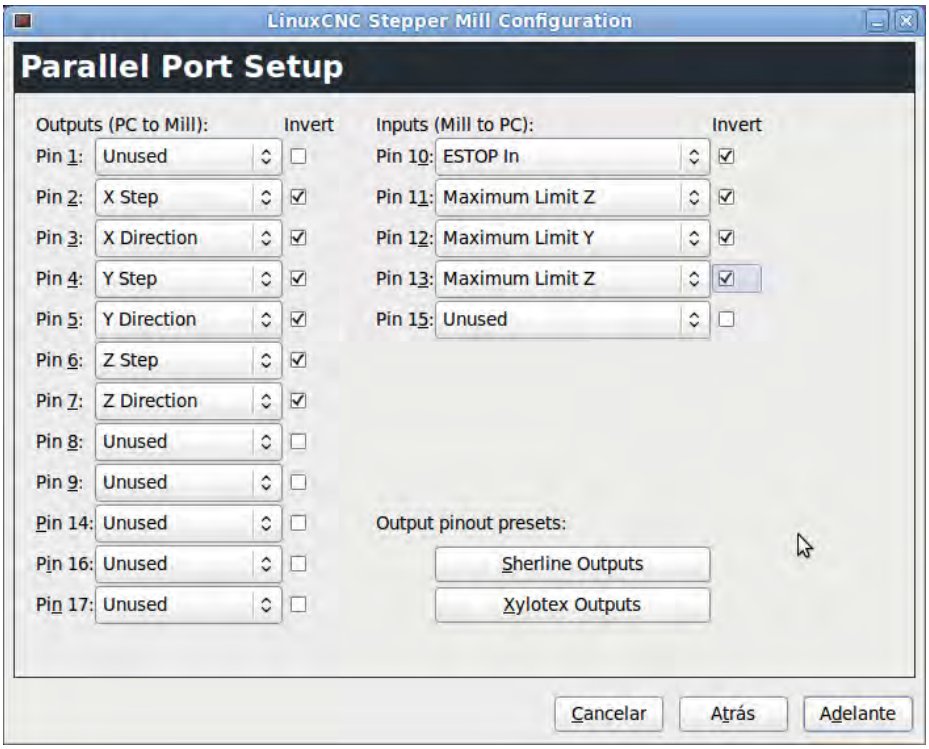


Figura B.10: Configuración de las terminales a manejar.

Apéndice C

Programación en código G

Hoy en día en que las computadoras son cada vez son pequeñas y económicas y con ello el uso del control numérico computarizado se puede extender a todo tipo de mecanismos como tornos, fresadoras, rectificadores, electroerosionadores (arco eléctrico), etc. Y en diversos procesos de fabricación como torneado, corte, perforación, sujeción, medición (por coordenadas), soldadura, entre otros.

A pesar de que las máquinas de CNC son utilizadas alrededor del mundo, existen variaciones en su programación aún entre las fabricadas por la misma empresa, y esto es debido a los diferentes estándares que se utilizan, entre los que se encuentran el ISO 6983 (International Standarization Organization) y el EIA RS274 (Electronic Industries Association).

En este capítulo vamos a explorar los fundamentos de la programación CNC, ¿Cómo es la programación en las máquinas CNC?, ¿Qué tipo de movimientos puede realizar las máquinas? y ¿Qué funciones existen para generar esos movimientos?; para tener un conocimiento básico de estas máquinas sofisticadas e involucrarnos en el uso de esta tecnología, nos centraremos en el estandar EIA RS274 que es con el que funcionan nuestras máquinas de CNC, al usar el software LinuxCNC.

C.1. Estructura del programa

Un programa CNC se compone de una secuencia de bloques de código [EMCO, 2003]. En el campo de CNC cada línea de programa es llamada bloque. En esta terminología, un bloque se define simplemente como una sola instrucción procesada por el sistema CNC.

Un bloque se define con la letra “N” seguida de un número, por ejemplo N20. Donde “N” indica “bloque” y “20” es el número correspondiente de bloque.

Se debe definir un bloque cuando se utiliza como destino de referencias o saltos. En otros casos es opcional el definir bloques. El Código C.1 es un ejemplo simple, tiene tres bloques N20, N30 y N40, los primeros dos realizan movimientos lineales y el último bloque indica el fin del programa (las funciones G01 y M30 se explicaran más adelante). El resultado de la ejecución de este programa se presenta en la Figura C.1.

Código C.1: Simple ejemplo con bloques.

```
N20  G01  X4  Y2  F101
N30  G01  X8  Y0
N40  M30
```

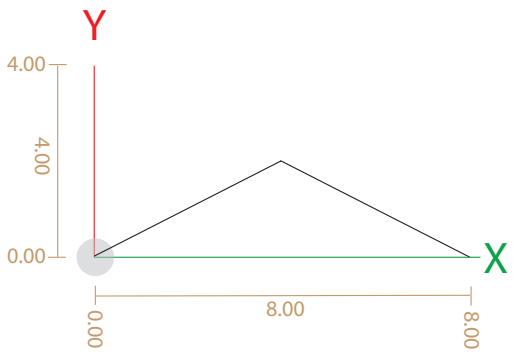


Figura C.1: Simple ejemplo con bloques.

La numeración de los bloques puede ser con números arbitrarios, lo habitual es programar con números múltiplos de 5 ó 10 para facilitar la inserción de futuras líneas y en orden ascendente [Félix Ledo Pernas, 2006].

El programa proporciona la información necesaria para realizar el mecanizado de la pieza deseada. Cada bloque puede tener las letras “N”, “G”, “F”, “S”, “M” y las que identifican a los ejes (ver Tabla C.1), acompañadas de un valor numérico. El formato del valor numérico incluye, además de los dígitos “0” a “9”, el punto decimal “.” y los signos “+”, “-” para determinar el sentido del movimiento [AUTOMATION, 2010].

La programación admite espacios entre letras, números y signo, así como prescindir del signo si fuera positivo. Así, “X10 Y20”, “X 10 Y 20”, “X10Y20” son la misma instrucción.

Caracter	Significado
N	Número de bloque
G	Funciones de recorrido
X, Y, Z	Nombre de ejes principales
A, B, C	Nombre de ejes rotativos
R	Radio de un arco
I, J, K	Coordenadas del centro de un arco o círculo
F	Velocidad de avance
( )	Los comentarios del programa van entre paréntesis
M	Funciones auxiliares

Tabla C.1: Caracteres que forman parte del lenguaje de programación.

Las funciones o comandos G se utilizan para informar a la máquina CNC de la geometría y condiciones de trabajo, como por ejemplo, la forma de las trayectorias, sistemas de referencia absoluta y relativa, unidades de trabajo, etc.

Cada eje está definido por una letra; esa letra seguida de un valor numérico hace referencia a una posición en el eje.

El avance de los ejes se representa mediante la letra “F” seguida del valor de la velocidad de avance deseada.

Se puede incluir información a modo de comentario mediante paréntesis “(” y “)”. Los comentarios pueden ir en cualquier parte y existir más de un comentario en el mismo bloque. El siguiente bloque es un ejemplo del uso de comentarios.

N10 G01 X1 (Comentario) Y2 (más comentario)

Finalmente las funciones auxiliares M permiten controlar elementos de la máquina como el manejo del cabezal, cambio de herramienta, uso de refrigerante, etc. En nuestro caso sólo nos servirán para indicar el fin del programa. El Código C.2 ilustra un programa completo con lo anteriormente dicho.

Código C.2: Un programa completo.

```
N10 G00 X0 Y0 Z0 (Bloque que lleva la herramienta al origen)
N20 G00 X1 Y1    (Coloca la herramienta en la posición [1,1] del plano XY en
    unidades de pulgada)
N30 M30          (Termina el programa)
```

C.2. Descripción de las funciones G básicas

C.2.1. Distancias absolutas (G90) o incrementales (G91) en desplazamientos lineales

Antes de usar cualquier instrucción de movimiento es necesario indicar el tipo de distancia que usaremos en el desplazamiento lineal, pueden ser distancias absolutas o incrementales.

Las distancias absolutas hacen referencia a las coordenadas de origen de la máquina. Por ejemplo, un movimiento rápido con magnitud de 8 en el eje X, desplazará la herramienta a la posición 8 del eje X, sin importar donde se encuentre la herramienta. Esta forma de trabajo la determinamos con la función G90 para desplazamientos lineales.

En el modo de distancias incrementales, suma la magnitud del movimiento a la posición actual. Por ejemplo, si la herramienta se encuentra en la posición 2 del eje X y hacemos un movimiento rápido con magnitud de 8 en el eje X, desplazará la herramienta a la posición 10 del eje X. Este modo de programación lo indicamos con G91 para desplazamientos lineales.

El modo de distancias sigue vigente en todas las siguientes instrucciones mientras no se cambie.

C.2.2. Distancias absolutas (G90.1) o incrementales (G91.1) en círculos y segmentos de círculos

En forma similar a los desplazamientos lineales, para el caso de funciones relacionadas con círculos y segmentos de círculos es necesario, antes de usarlas, definir si se utilizan distancias absolutas o incrementales.

Para programar con distancias absolutas lo hacemos mediante la función G90.1 y para distancias incrementales con G91.1.

El modo de distancias sigue vigente en todas las siguientes instrucciones mientras no se cambie.

C.2.3. Programación en pulgadas (G20) o en milímetros (G21)

Otro parámetro que también debemos considerar antes de realizar cualquier movimiento es el sistema de unidades con que trabajaremos. Los desplazamientos y el avance de los ejes se pueden definir en el sistema métrico (milímetros) o en el sistema inglés (pulgadas). El sistema de unidades se puede seleccionar desde el programa mediante las funciones:

G20 (Selección de las unidades en pulgadas)
G21 (Selección de las unidades en milímetros)

Si no se programa ninguna de estas funciones, utiliza pulgadas por omisión. A partir de la ejecución de una de las funciones, la máquina CNC asume dicho sistema de unidades para los bloques siguientes. El Código C.3 es un ejemplo de programación en pulgadas. El resultado de la ejecución de este programa se presenta en la Figura C.2.

Código C.3: Un ejemplo sencillo utilizando pulgadas como unidad de trabajo.

```
G90 (Programamos en distancias absolutas)
G20 (Pulgadas como sistema de unidades)
G01 X2 Y1 F508 (Movimiento lineal con un avance de 508 pulgadas/minuto)
M30 (Fin del programa)
```

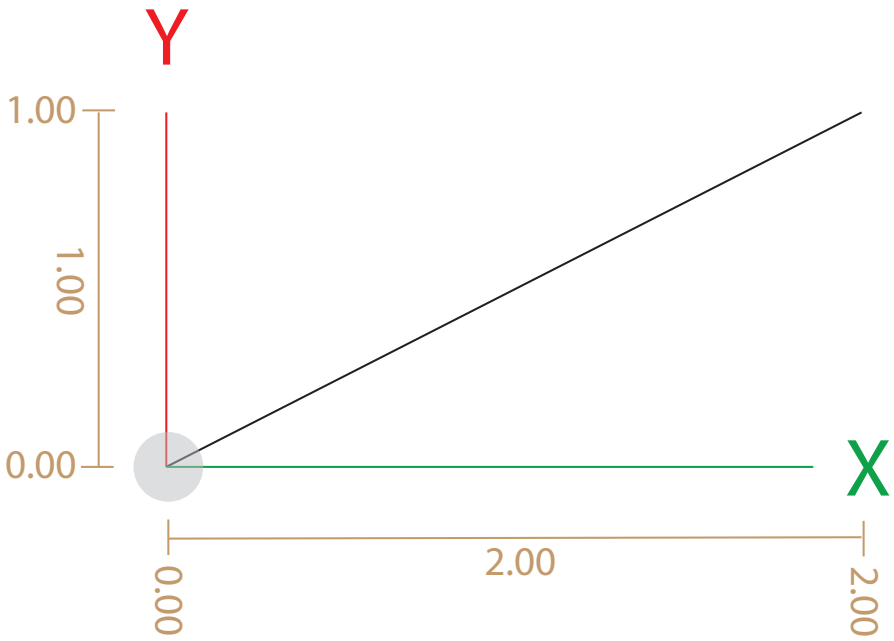


Figura C.2: Un ejemplo sencillo utilizando pulgadas como unidad de trabajo.

En los ejes rotativos las unidades de programación son grados por lo que no les afecta el cambio entre milímetros y pulgadas. El Código C.4 es un programa donde el eje A gira 200 grados. En LinuxCNC no se logra apreciar el efecto, la Figura C.3 ilustra la ejecución de este programa.

Código C.4: Las unidades para el eje rotativo siempre son grados.

```
G90 (Programación en distancias absolutas)
G20 (Pulgadas como sistema de unidades)
G01 A200 F360 (Mueve el eje A 200 grados con un avance de 360 grados/minuto)
M30 (Fin del programa)
```

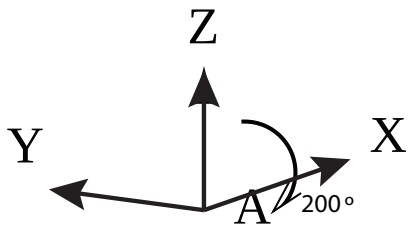


Figura C.3: Las unidades para el eje rotativo siempre son grados.

C.2.4. Movimiento lineal rápido (G00)

Esta función desplaza la herramienta a la máxima velocidad de la máquina, desde la posición actual a un punto especifico. Independientemente del número de ejes que se desplacen, la trayectoria resultante es siempre una línea recta. El formato de la función es el siguiente:

G00 X$< Num >$ Y$< Num >$ Z$< Num >$
Donde $< Num >$ es un número real.

El movimiento se realiza definiendo las coordenadas del punto final en los diferentes ejes. No es necesario programar todos los ejes, sólo aquellos que se desean desplazar. Esta función permanece activa hasta que se programa una nueva función. El código C.5 es un programa que muestra un ejemplo de la función G00, la herramienta de corte de la máquina CNC está en las coordenadas iniciales (0,0,0) y ejecutando un movimiento rápido la llevamos con la máxima velocidad al punto donde realizará una perforación. La Figura C.4 ilustra la ejecución de este programa. Cuando se inicia un programa con movimientos rápidos, LinuxCNC no dibuja su recorrido ya que esta función es pensada para movimientos sin cortes en la pieza de trabajo. En otros casos LinuxCNC dibuja este movimiento con una linea discontinua.

Código C.5: Un ejemplo de uso de movimiento rápido G00.

```
G90 (Programación en distancias absolutas)
G20 (Pulgadas como sistema de unidades)
G00 Z1 (Movimiento rápido, levantamos herramienta una pulgada)
G00 X0.5 Y0.5 (Movimiento rápido)
G01 Z-0.5 F10 (Avance con velocidad programada, baja herramienta)
M30 (Fin del programa)
```

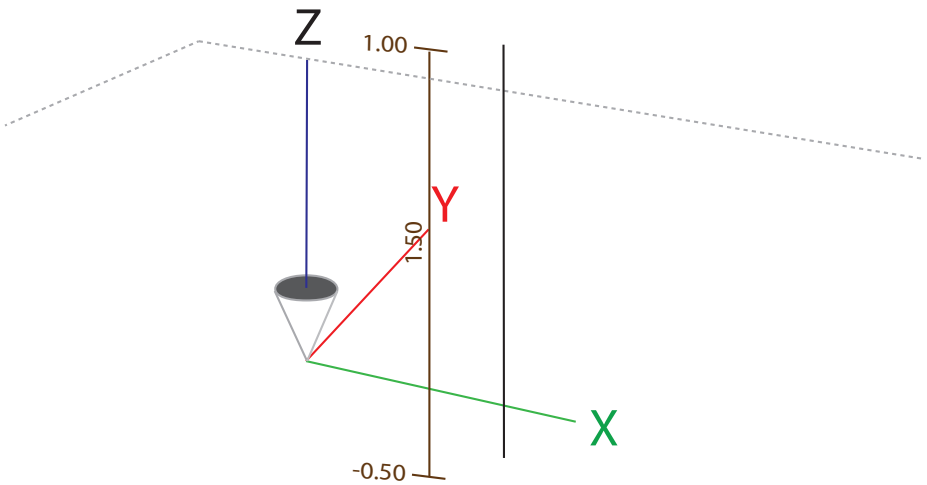


Figura C.4: Un ejemplo de uso de movimiento rápido G00.

También la función G00 puede programarse como G0. Esta función es usada comúnmente cuando es necesario hacer un posicionamiento rápido.

C.2.5. Movimiento lineal con velocidad programada (G01)

Esta función desplaza la herramienta con una velocidad de avance programada con el parámetro F, desde la posición actual a un punto especifico. El resultado es una trayectoria rectilínea. El formato de la función es el siguiente:

G01 X$< Num >$ Y$< Num >$ Z$< Num >$ A$< Num >$ F$< Num >$

El movimiento se realiza definiendo las coordenadas del punto final en los diferentes ejes y el parámetro de velocidad F. No es necesario programar todos ejes, sólo

aquellos que se desean desplazar.

El avance F programado permanece activo hasta que se programa un nuevo valor, por tanto, no es necesario definirlo en cada bloque. Las unidades de este parámetro se definen de acuerdo al sistema de unidades con que esté funcionando la máquina CNC por minuto (por ejemplo pulgadas/min).

Esta función también permanece activa hasta que se programa una nueva función.

También la función G01 puede escribirse como G1. Esta función es usada para el mecanizado de la pieza deseada. El Código C.6 es un ejemplo de movimientos lineales en los ejes X, Y y Z. La Figura C.5 ilustra la ejecución del programa. Y un ejemplo para el eje rotatorio es el Código C.4. La Figura C.3 ilustra la ejecución de este programa.

Código C.6: Ejecución de movimientos lineales.

```
G90 (Programación en distancias absolutas)
G20 (Pulgadas como sistema de unidades)
G00 Z2 (Levanta herramienta rápidamente)
G00 X1 Y2 (Movimiento rápido a P1)
G01 Z-1 F225 (Baja herramienta lentamente)
G01 X14 Y10 F450 (Trazo L1)
Y12 (Trazo L2, sigue el efecto de G01)
X10 Y10 (Trazo L3)
X1 (Trazo L4)
Y2 (Trazo L5)
G00 Z2 (Levanta herramienta rápidamente)
G00 X0 Y0 (Regresa al origen)
M30 (Fin del programa)
```

C.2.6. Movimiento circular en sentido horario (G02)

Círculos completos

Para trazar un círculo en sentido de las manecillas del reloj a partir de la posición actual de la herramienta con un avance programado de velocidad, se utiliza la función G02.

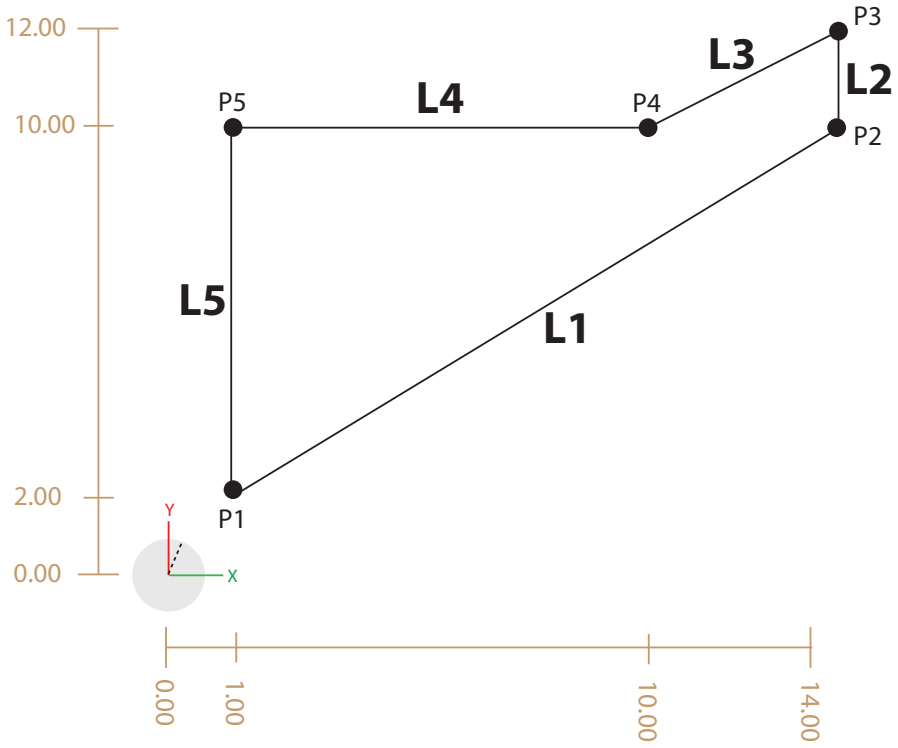


Figura C.5: Ejecución de movimientos lineales.

Es necesario indicar, antes de usar esta función, el tipo de distancia con que se trabajará, ya sea absoluta o relativa según lo establecido por G90.1 o G91.1 respectivamente (ver sección C.2.2).

Antes de usar esta función colocamos la herramienta en un punto del círculo. El formato de está función incluye las coordenadas del centro y es el siguiente:

G02 I$< Num >$ J$< Num >$ K$< Num >$ F$< Num >$

Las coordenadas del centro son los parámetros I, J y K para los ejes X, Y y Z respectivamente, mientras que la velocidad del movimiento está determinado por F.

El Código C.7 es un ejemplo utilizando esta función, donde se traza un círculo con centro en $x = 3$, $y = 4$ y un radio de una unidad. La Figura C.6 ilustra la ejecución del programa.

Código C.7: Círculo usando la función G02.

```
G90   (Programación en distancias absolutas)
G90.1 (distancias absolutas para círculos y arcos)
G20   (Pulgadas como sistema de unidades)
G00 Z1 (Levanta herramienta rápidamente)
G00 X2 Y4 (Colocamos herramienta en una parte del círculo)
G01 Z-0.7 F72.0 (Baja herramienta lentamente)
G02 I3 J4 F250.0 (Traza círculo completo a una mayor velocidad)
G00 Z1 (Levanta herramienta rápidamente)
G00 X0 Y0 (Regresa al origen)
M30 (Fin del programa)
```

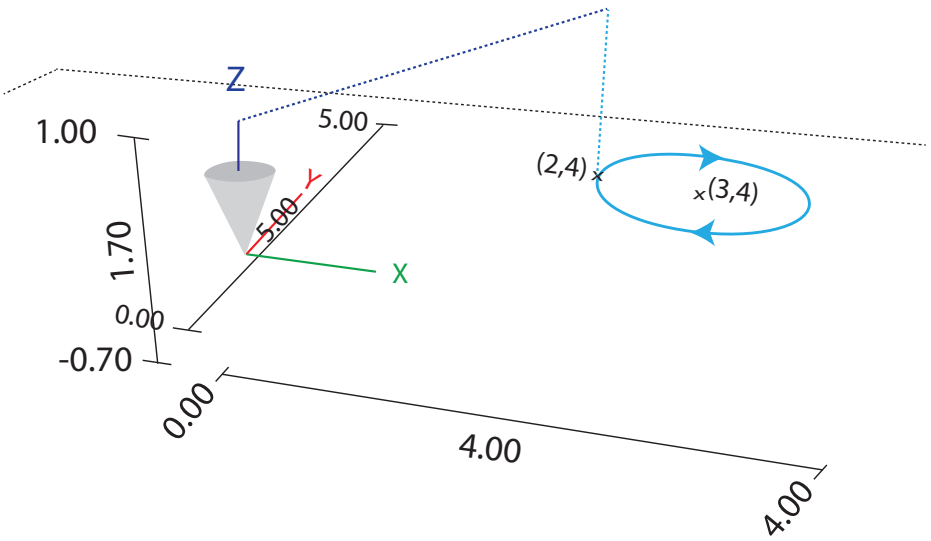


Figura C.6: Círculo usando la función G02.

Segmentos de círculos

Por otro lado, para programar segmentos de círculos (arcos) existen dos formatos permitidos: formato con coordenadas del centro y formato con el radio.

Para el formato de segmentos de círculos con coordenadas del centro, tenemos:

G02 X$< Num >$ Y$< Num >$ Z$< Num >$ I$< Num >$ J$< Num >$ K$< Num >$ F$< Num >$

El arco comienza en la posición actual de la herramienta y termina en las coordenadas X, Y y Z; las coordenadas del centro del círculo están dadas con I, J y K, mientras que la velocidad del movimiento está determinada por F.

El Código C.8 es un ejemplo donde se usa este formato. La Figura C.7 ilustra la ejecución del programa.

Código C.8: Arco con la función G02 usando las coordenadas del centro.

```
G90   (Programación en distancias absolutas)
G90.1 (Distancias absolutas para círculos y arcos)
G20   (Pulgadas como sistema de unidades)
G01 Z-0.7 F72.0 (Baja herramienta lentamente)
G02 X4 Y0 I2 J0 F250.0 (Traza arco con mayor velocidad)
G00 Z1 (Levanta herramienta rápidamente)
M30 (Fin del programa)
```

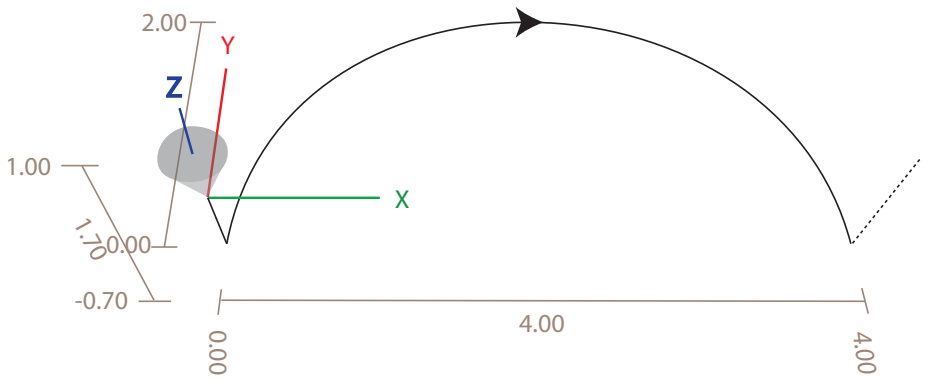


Figura C.7: Arco con la función G02 usando las coordenadas del centro.

La opción de arcos mediante el radio, tiene el siguiente formato:

G02 X$< Num >$ Y$< Num >$ Z$< Num >$ R$< Num >$ F$< Num >$

El arco comienza en la posición actual de la herramienta y termina en las coor-

denadas X, Y y Z; el parámetro R define el radio y la velocidad del movimiento está determinada por F.

El radio es expresado en el sistema de unidades con que esté funcionando la máquina.

El Código C.9 es un ejemplo donde se usa este formato. La Figura C.8 ilustra la ejecución del programa.

Código C.9: Arco usando el formato del radio.

```
G90   (Programación en distancias absolutas)
G90.1 (distancias absolutas para círculos y arcos)
G20   (Pulgadas como sistema de unidades)
G01 Z-0.7 F72.0 (Baja herramienta lentamente)
G02 X4 Y0 R2 F250.0 (Traza arco con mayor velocidad)
G00 Z1 (Levanta herramienta rápidamente)
M30 (Fin del programa)
```

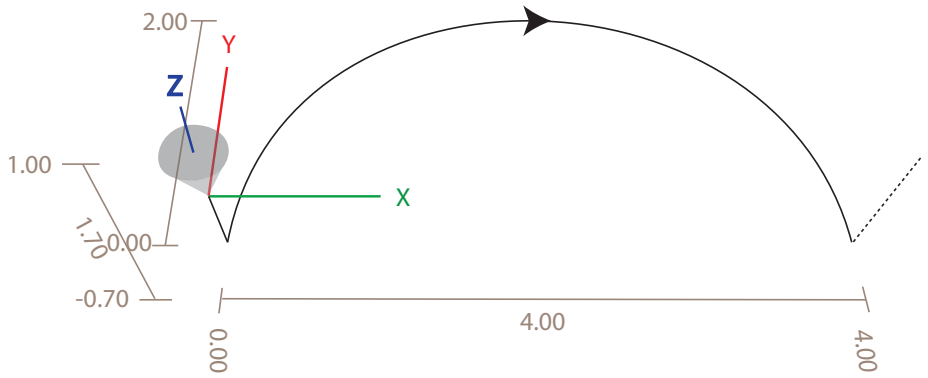


Figura C.8: Arco usando el formato del radio.

C.2.7. Movimiento circular en sentido antihorario (G03)

Traza un círculo o un arco en sentido opuesto a las manecillas del reloj y usa los mismos formatos que G02.

Un ejemplo para un círculo en sentido antihorario está en el Código C.10, cuya ejecución se ilustra en la Figura C.9.

El Código C.11 es un ejemplo para un arco en sentido antihorario usando las coordenas del centro, cuya ejecución se ilustra en la Figura C.7.

Y el Código C.12 es un ejemplo para un arco en sentido antihorario usando el radio, cuya ejecución se ilustra en la Figura C.8.

Código C.10: Círculo en sentido antihorario.

```
G90     (Programación en distancias absolutas)
G90.1   (Distancias absolutas para círculos y arcos)
G20     (Pulgadas como sistema de unidades)
G00 Z1  (Levanta herramienta rápidamente)
G00 X2 Y4  (Colocamos herramienta en una parte del círculo)
G01 Z-0.7 F72.0  (Baja herramienta lentamente)
G03 I3 J4 F250.0  (Traza círculo con mayor velocidad)
G00 Z1  (Levanta herramienta rápidamente)
G00 X0 Y0  (Regresa al origen)
M30  (Fin del programa)
```

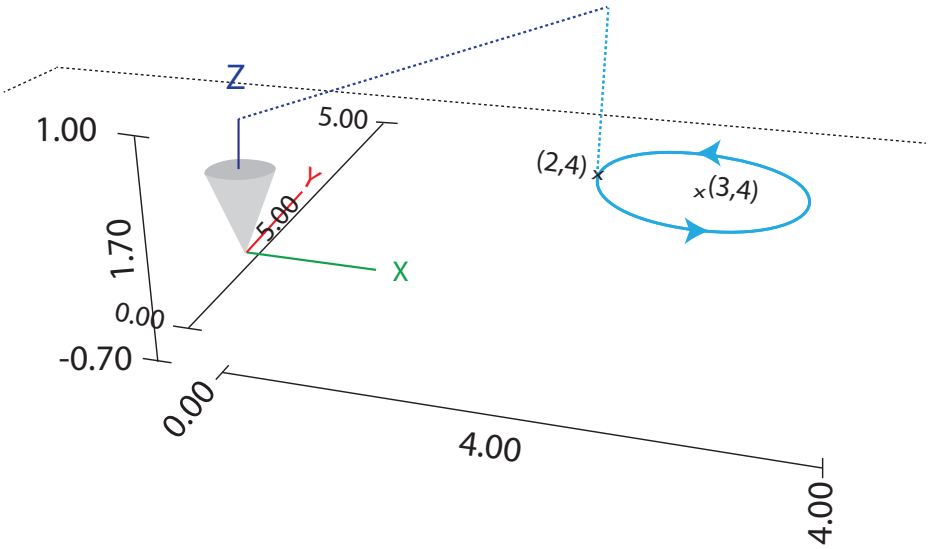


Figura C.9: Círculo en sentido antihorario.

Código C.11: Arco en sentido antihorario usando las coordenadas del centro.

```
G90   (Programación en distancias absolutas)
G90.1 (Distancias absolutas para círculos y arcos)
G20   (Pulgadas como sistema de unidades)
G01 Z-0.7 F72.0 (Baja herramienta lentamente)
G03 X4 Y0 I2 J0 F250.0 (Traza arco con mayor velocidad)
G00 Z1 (Levanta herramienta rápidamente)
M30 (Fin del programa)
```

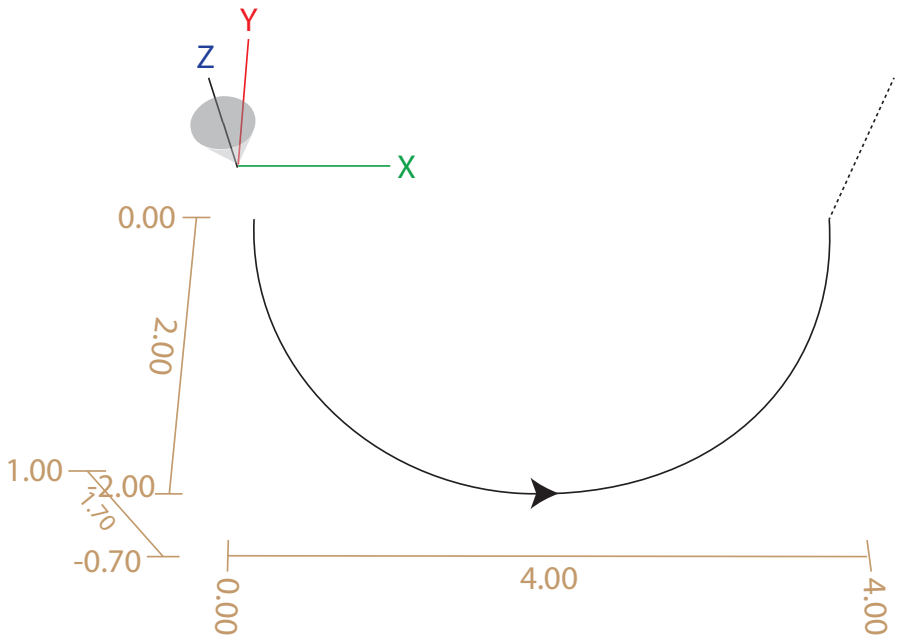


Figura C.10: Arco en sentido antihorario usando las coordenadas del centro.

Código C.12: Arco en sentido antihorario usando el radio.

```
G90   (Programación en distancias absolutas)
G90.1 (Distancias absolutas para círculos y arcos)
G20   (Pulgadas como sistema de unidades)
G01 Z-0.7 F72.0 (Baja herramienta lentamente)
G03 X4 Y0 R2 F250.0 (Traza arco con mayor velocidad)
G00 Z1 (Levanta herramienta rápidamente)
M30 (Fin del programa)
```

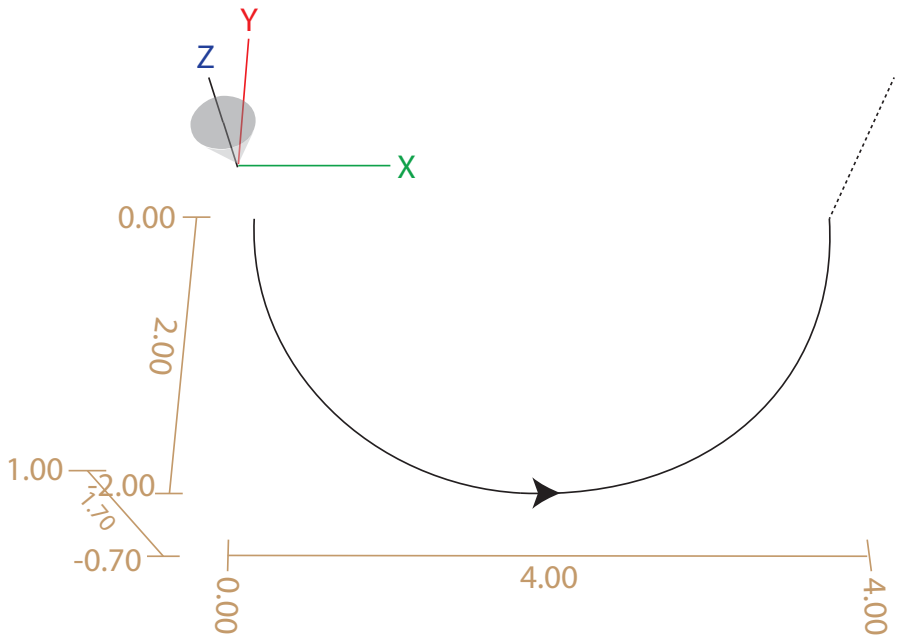


Figura C.11: Arco en sentido antihorario usando el radio.

C.3. Fin del programa (M30)

El fin del programa se define mediante la función M30. Todas las operaciones se detienen y el programa termina su ejecución.

Esta es la única función M que usamos en nuestras máquinas CNC.

Conviene mencionar que la definición del sistema de unidades (milímetros o pulgadas) y el tipo de distancias (absolutas o incrementales) mantienen su vigencia si se carga y se ejecuta un nuevo código que no los modifique, mientras no se termine la aplicación LinuxCNC.

C.4. Un Ejemplo práctico

A continuación presentamos un ejemplo completo de uso de la máquina CNC para fabricar una placa de soporte de un pequeño robot móvil. En la Figura C.12 se ilustra las medidas y características requeridas.

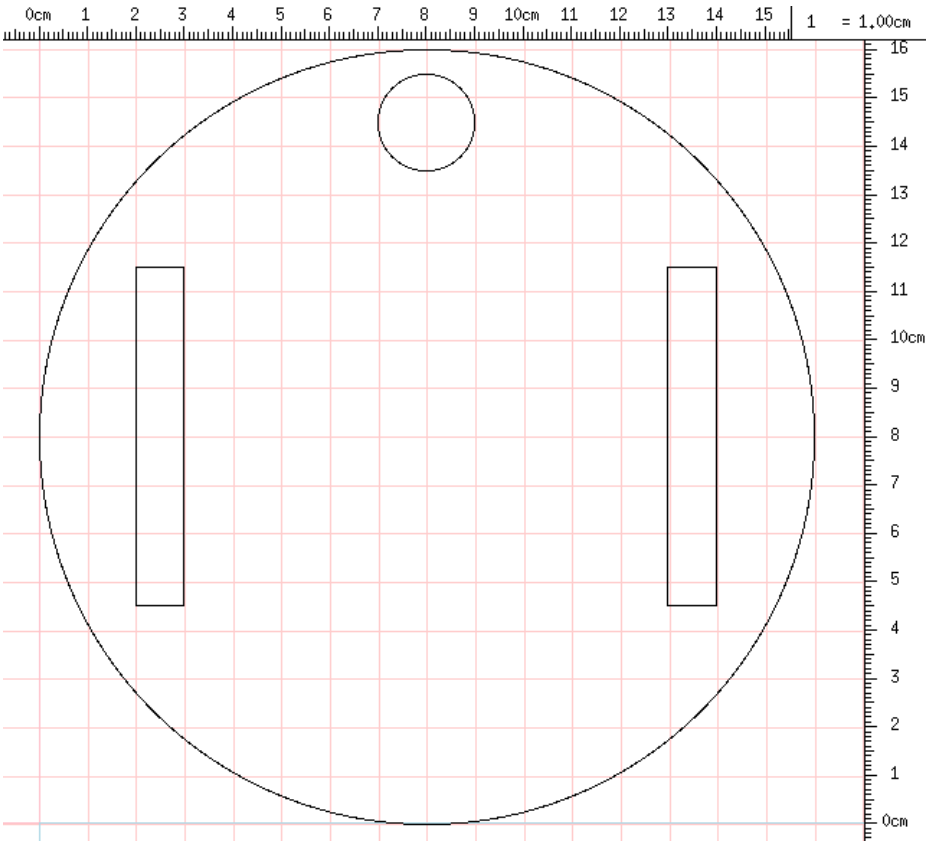


Figura C.12: Dimensiones de la base del robot que crearemos con código G.

Con la especificaciones requeridas y las funciones presentadas en este capítulo, programamos el corte de la placa en la máquina CNC con el Código C.13. En la Figura C.13 podemos ver los movimientos generados.

Código C.13: Generación de una base para robot.

```
G90 (Distancias absolutas)
G21 (Sistema de unidades en milímetros)
G00 Z20 (Levantamos herramienta)
(Iniciamos el rectángulo de la izquierda)
G00 X20 Y115 (Nos colocamos en la esquina izquierda superior)
G01 F256.0 Z-8 (Bajamos la herramienta)
G01 F256.0 Y45 (Nos movemos a la esquina izquierda inferior)
G01 X30  (Y de ahí a la esquina derecha inferior)
G01 Y115 (Y a la esquina derecha superior)
```

```
G01 X20  (Regresa a la esquina izquierda superior)
(Termina primer rectángulo)
G01 F256.0 Z10 (Sube herramienta)
(Inicia segundo rectángulo)
G00 X130 (Nos colocamos en la esquina izquierda superior)
G01 F256.0 Z-8 (Bajamos lentamente la herramienta)
G01 F256.0 X140 (Nos movemos a la esquina derecha superior)
G01 Y45 (A la esquina derecha inferior)
G01 X130 (A la esquina izquierda inferior)
G01 Y115 (Regresa a la esquina izquierda superior)
(Termina cuadro)
G01 F256.0 Z10 (Sube herramienta)
(Inicia círculo para rueda loca)
G00 X80 Y155 (Nos colocamos en un punto del círculo)
G01 F256.0 Z-8 (Bajamos lentamente la herramienta)
G90.1 (Usamos distancias absolutas para el círculo)
G02 I80 J145 Z-8 F256.0 (Traza círculo iniciando donde está, con centro en
    [90,90])
(Termina círculo rueda loca)
G00 Z10 (Sube herramienta)
(Hacemos círculo)
G00 X80 Y160 (Nos colocamos en una parte del círculo)
G01 Z-8 F256.0 (Bajamos herramienta)
G90.1 (Usamos distancias absolutas para el círculo)
G02 I80 J80 Z-8 F256.0 (Traza círculo iniciando donde está, con centro en
    [90,90])
(Termina círculo)
G00 Z10 (Sube herramienta)
G00 X0 Y0 (Regresa al origen)
M30 (Fin del programa)
```

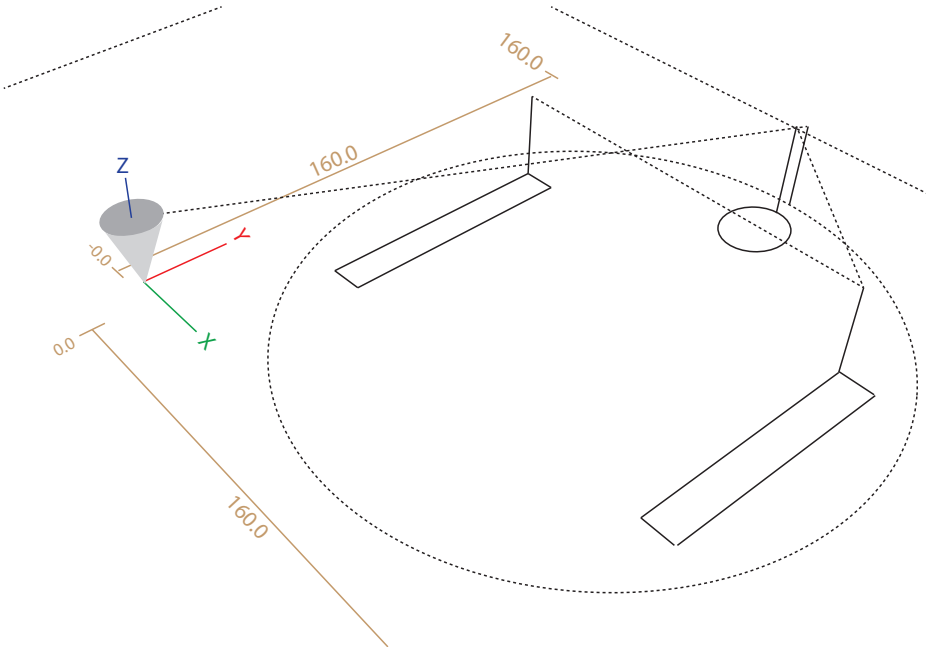


Figura C.13: Generación de una base para robot.

Apéndice D

Un tutorial sobre el método mínimos cuadrados total para ajustar una línea recta

D.1. Introducción

La detección de características geométricas (líneas, círculos, superficies, etc.) de un conjunto de puntos es una tarea fundamental en varios campos de la ciencia y la ingeniería; por ejemplo, la metrología, la visión artificial, la robótica móvil, etc.

Sea $Z = \{z_1, \ldots, z_n\}$ un conjunto de n mediciones o puntos donde cada punto $z_i = \langle x_i, y_i \rangle$ está representada por sus coordenadas rectangulares.

Una relación lineal entre x e y se escribe como:

$$y = ax + b \tag{D.1}$$

donde a es la pendiente de la línea recta y b es la intersección del eje y. En el método clásico de mínimos cuadrados los datos de las abscisas (x_i, $i = 1, ..., n$) se asumen que se conoce con exactitud, mientras que las incertidumbres de los datos de las ordenadas (y_i) se utilizan como pesos para el ajuste de la línea de $\langle a, b \rangle$, dado por (D.1), al conjunto de mediciones Z.

La solución para ajustar una línea usando la regresión por mínimos cuadrados, se explica en varios libros de texto como son: cálculo, álgebra lineal, análisis numérico, probabilidad, estadística, y otros.

Sin embargo, los datos medidos nunca están libres de incertidumbre. Esto significa, para determinar el mejor ajuste de una línea, se requiere un método el cual tome en cuenta las incertidumbres de los x_i e y_i [Krystek y Anton, 2007].

La regresión por mínimos cuadrados totales (por sus siglas en inglés TLS) fue presentado por Golub y Van Loan [Golub, 1973] para hacer frente a ambas incertidumbres.

A pesar de su utilidad y su sencillez, el TLS aún no ha aparecido en el análisis numérico, estadístico o textos de álgebra lineal.

La introducción a TLS es el propósito de este tutorial, y puede complementar los cursos de análisis numérico, estadística o álgebra lineal, o servir como una transición de este tipo de cursos a un curso más avanzado y especializados. Referencias adicionales a TLS son el documento de introducción por Yves Nievergelt [Nievergelt, 1994]; una visión general de los métodos de TLS, por Ivan Markovsky y Sabine Van Huffel [Markovsky y Van Huffel, 2007]; o el libro de Sabine Van Huffel y Joos Vandewalle, sobre el problema TLS [Van Huffel y Vandewalle, 1991].

D.2. Preliminares

D.2.1. Distancia ortogonal de un punto a una línea

La distancia más corta desde un punto dado $z_i = \langle x_i, y_i \rangle$ a una línea $\ell = \langle r, \phi \rangle$, denotado por $d_\perp(z_i, \ell)$, se calcula fácilmente de la siguiente manera.

Una línea ℓ_i a través del punto z_i, paralela a la línea ℓ, está dada por:

$$r_i = x_i \cos\phi + y_i \sin\phi$$

La separación entre esta nueva línea de ℓ_i con parámetros $\langle r_i, \phi \rangle$ y la línea de ℓ con parámetros $\langle r, \phi \rangle$ es la diferencia de $d_i = r_i - r$, debido a que ambas líneas tienen el mismo ϕ (véase la figura D.1). Así la distancia deseada, llamada distancia ortogonal,

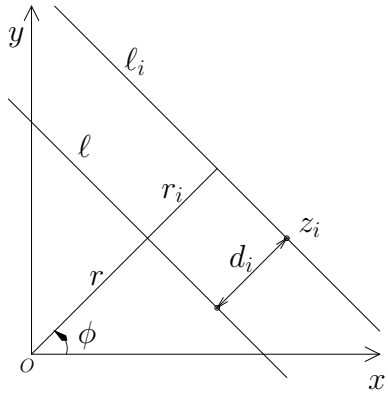


Figura D.1: Distancia ortogonal d_i del punto z_i a la línea ℓ.

está dada por

$$d_{\perp}(z_i, \ell) = x_i \cos\phi + y_i \sin\phi - r \tag{D.2}$$

D.3. Regresión de mínimos cuadrados totales

D.3.1. Definición del problema

En la literatura, el problema de ajustar una línea recta con errores en los datos en ambas coordenadas se formulo primero por Pearson a principios de 1901 [Krystek y Anton, 2007]. Deming en 1943 [Deming, 1943] sugirió minimizar la suma:

$$\chi^2(\ell; z_1, \ldots, z_n) = \sum_{i=1}^{n} \left[\frac{(x_k - X_k)^2}{u_{x,k}^2} + \frac{(y_k - Y_k)^2}{u_{y,k}^2} \right] \tag{D.3}$$

donde (x_k, y_k) son las coordenadas de los puntos con sus correspondientes incertidumbres $(u_{x,k}, u_{y,k})$ y (X_k, Y_k) denota los correspondientes puntos de la línea ℓ. La mejor línea minimiza χ^2. En el caso de $u_{x,k} = u_{y,k} = \sigma, k = 1, \ldots, n$, el problema se reduce a mínimos cuadrados totales y minimizar(D.3) es equivalente a minimizar la distancia ortogonal de

los puntos (x_k, y_k) para ajustar la línea. Por lo tanto, esto es refiere frecuentemente como regresión ortogonal [Krystek y Anton, 2007]. En este caso, la mejor línea minimiza:

$$\chi^2(\ell; Z) = \sum_{i=1}^{n} \frac{d_\perp^2(z_i, \ell)}{\sigma^2} \tag{D.4}$$

D.3.2. Ajuste de la mejor línea

Remplazando la ec. (D.2) en (D.4), la mejor línea ℓ minimiza

$$\chi^2(\ell; Z) = \frac{1}{\sigma^2} \sum_{i=1}^{n} (x_i \cos\phi + y_i sin\phi - r)^2 \tag{D.5}$$

Una condición para que sea minimo es que las derivadas parciales χ^2 con respecto a los parámetros (r and ϕ) de la línea sean: $\frac{\partial \chi^2}{\partial r} = \frac{\partial \chi^2}{\partial \phi} = 0$.

Para $\frac{\partial \chi^2}{\partial r} = 0$ tenemos:

$$\begin{aligned}
\frac{\partial(\frac{1}{\sigma^2}\sum_{i=1}^{n}(x_i\cos\phi + y_i sin\phi - r)^2)}{\partial r} &= 0 \\
\frac{-2}{\sigma^2}\sum_{i=1}^{n}(x_i\cos\phi + y_i\sin\phi - r) &= 0 \\
\cos\phi\sum_{i=1}^{n}(x_i) + \sin\phi\sum_{i=1}^{n}(y_i) - \sum_{i=1}^{n}(r) &= 0 \\
\cos\phi\sum_{i=1}^{n}(x_i) + \sin\phi\sum_{i=1}^{n}(y_i) - nr &= 0 \\
\cos\phi\left[\frac{1}{n}\sum_{i=1}^{n}x_i\right] + \sin\phi\left[\frac{1}{n}\sum_{i=1}^{n}y_i\right] - r &= 0
\end{aligned} \tag{D.6}$$

Las expresiones en los paréntesis cuadrados se conocen como la media de x e y, se definen como sigue:

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n}x_i\ , \quad \overline{y} = \frac{1}{n}\sum_{i=1}^{n}y_i \tag{D.7}$$

Por lo tanto la ec. (D.6) se reduce a:

$$\begin{aligned}
\cos\phi\ \overline{x} + \sin\phi\ \overline{y} - r &= 0 \\
r &= \overline{x}\cos\phi + \overline{y}\sin\phi
\end{aligned} \tag{D.8}$$

Al comparar la ecuación normal de una línea $r = x\cos\phi + y\sin\phi$ y (E.6), se obtiene un importante resultado:

> El centroide de los puntos dados por $\langle \overline{x}, \overline{y} \rangle$ es un punto de la línea ℓ con parámetros $\langle r, \phi \rangle$ el cual minimiza la ec. (D.5).

Remplazando la ec. (E.6) en (D.5), $\chi^2(\ell; Z)$ se puede expresar como:

$$\chi^2(\ell; Z) = \frac{1}{\sigma^2} \sum_{i=1}^{n} [(x_i - \overline{x})\cos\phi + (y_i - \overline{y})\sin\phi]^2 \tag{D.9}$$

En la ec. (D.9)el único parámetro desconocido es ϕ. Entonces, al hacer $\frac{\partial \chi^2}{\partial \phi} = 0$, para encontrar el correcto ϕ

$$\begin{aligned}
\frac{\partial(\frac{1}{\sigma^2}\sum_{i=1}^{n}[(x_i - \overline{x})\cos\phi + (y_i - \overline{y})\sin\phi]^2)}{\partial\phi} &= 0 \\
\frac{1}{\sigma^2}\sum_{i=1}^{n} 2[(x_i - \overline{x})\cos\phi + (y_i - \overline{y})\sin\phi] \times & \\
[-(x_i - \overline{x})\sin\phi + (y_i - \overline{y})\cos\phi] &= 0 \\
\sum_{i=1}^{n} -(x_i - \overline{x})^2 2\cos\phi\sin\phi + 2(x_i - \overline{x})(y_i - \overline{y})\cos^2\phi & \\
-2(x_i - \overline{x})(y_i - \overline{y})\sin^2\phi + (y_i - \overline{y})^2 2\cos\phi\sin\phi &= 0 \\
\sum_{i=1}^{n} 2\cos\phi\sin\phi[(y_i - \overline{y})^2 - (x_i - \overline{x})^2] + & \\
2(x_i - \overline{x})(y_i - \overline{y})(\cos^2\phi - \sin^2\phi) &= 0
\end{aligned} \tag{D.10}$$

Utilizando las siguientes identidades trigonometricas:

$$\sin 2\phi = 2\cos\phi\sin\phi\ , \quad \cos 2\phi = \cos^2\phi - \sin^2\phi \tag{D.11}$$

La ec. (D.10) se simplifica como:

$$\begin{aligned}
\sum_{i=1}^{n} \sin 2\phi[(y_i - \overline{y})^2 - (x_i - \overline{x})^2] + \\
2(x_i - \overline{x})(y_i - \overline{y}) \cos 2\phi = 0 \\
\sin 2\phi \sum_{i=1}^{n} [(y_i - \overline{y})^2 - (x_i - \overline{x})^2] + \\
2 \cos 2\phi \sum_{i=1}^{n} (x_i - \overline{x})(y_i - \overline{y}) = 0 \\
\tan 2\phi = \frac{\sin 2\phi}{\cos 2\phi} = \frac{-2 \sum_{i=1}^{n} [(x_i - \overline{x})(y_i - \overline{y})]}{\sum_{i=1}^{n} [(y_i - \overline{y})^2 - (x_i - \overline{x})^2]} \\
\phi = \frac{1}{2} \arctan \left(\frac{-2 \sum_{i=1}^{n} [(x_i - \overline{x})(y_i - \overline{y})]}{\sum_{i=1}^{n} [(y_i - \overline{y})^2 - (x_i - \overline{x})^2]} \right)
\end{aligned} \tag{D.12}$$

Las ecuaciones (D.12) y (E.6) obtienen los parámetros $\langle r, \phi \rangle$ de la línea deseada.

En la práctica, la ec. (D.12) usa los cuatro cuadrantes del arco tangente (atan2). $atan2(y, x)$ calcula $\arctan(y/x)$ pero utiliza el signo de ambos x e y para determinar el cuadrante en el cual se encuentra el ángulo resultante. Por ejemplo $atan2(-2, -2) = -135°$, mientras $atan2(2, 2) = 45°$, se perdería la diferencia con el uso de un argumento en la función arco tangente. Otro caso práctico es cuando la ec. (E.6) obtiene $r < 0$. En cuyo caso la linea $\langle r', \phi' \rangle$, donde $r' = -r$ y $\phi' = \phi + \pi$, representan la misma línea $\langle r, \phi \rangle$, pero $r' > 0$.

D.3.3. Forma matricial para obtener el ángulo ϕ

La ec. D.9 se puede reescribir en su forma matricial

$$\chi^2(\ell; Z) = \frac{1}{\sigma^2} ||\mathbf{Mp}||^2 \tag{D.13}$$

donde $\mathbf{M}$ es una matriz de dimensión $n \times 2$, $\mathbf{p}$ es un vector,

$$\mathbf{M} = \begin{bmatrix} x_1 - \overline{x} & y_1 - \overline{y} \\ x_2 - \overline{x} & y_2 - \overline{y} \\ \vdots & \vdots \\ x_n - \overline{x} & y_n - \overline{y} \end{bmatrix} \quad , \quad \mathbf{p} = \begin{bmatrix} \cos \phi \\ \sin \phi \end{bmatrix} \tag{D.14}$$

y $\|\mathbf{v}\|$ significa la norma Euclidiana de un vector $\mathbf{v}$ con coordenadas $[v_1\ v_2\ \ldots\ v_k]^t$, definidas como

$$\|\mathbf{v}\| = \sqrt{v_1^2 + v_2^2 + \ldots + v_k^2} \tag{D.15}$$

Ahora el objetivo es encontrar un vector $\mathbf{p}$ que minimice la ec. (D.13). En otras palabras, un vector $\mathbf{t}$ que minimice la norma de la relación lineal: $\mathbf{Mp}$. Nótese que $\mathbf{t}$ es un vector unitario, porque $\|\mathbf{p}\| = \sqrt{\cos^2\phi + \sin^2\phi} = 1$.

Para lograr este objetivo la norma Euclidiana también se expresa utilizando el producto interno del vector consigo mismo,

$$\|\mathbf{v}\| = \sqrt{\mathbf{v}^t\mathbf{v}} \tag{D.16}$$

donde $\mathbf{v}^t$ denota el vector transpuesto de $\mathbf{v}$. Utilizando el producto interno para calcular la norma se puede calcular la ec. (D.13) como:

$$\chi^2(\ell; Z) = \frac{1}{\sigma^2}(\mathbf{Mp})^t(\mathbf{Mp}) \tag{D.17}$$

De esta expresión, utilizando las propiedades de la transpuesta y la ley asociativa para el producto matricial, obtenemos la forma cuadrática:

$$\begin{aligned}\chi^2(\ell; Z) &= \frac{1}{\sigma^2}(\mathbf{p}^t\mathbf{M}^t)(\mathbf{Mp}) \\ \chi^2(\ell; Z) &= \frac{1}{\sigma^2}\mathbf{p}^t(\mathbf{M}^t\mathbf{M})\mathbf{p} \\ \chi^2(\ell; Z) &= \frac{1}{\sigma^2}\mathbf{p}^t\mathbf{A}\mathbf{p}\end{aligned} \tag{D.18}$$

Sea la forma de la matriz $\mathbf{A} = \mathbf{M}^t\mathbf{M}$, de dimensión 2×2,

$$\mathbf{A} = \begin{bmatrix} \sum_{i=1}^n (x_i - \overline{x})^2 & \sum_{i=1}^n (x_i - \overline{x})(y_i - \overline{y}) \\ \sum_{i=1}^n (x_i - \overline{x})(y_i - \overline{y}) & \sum_{i=1}^n (y_i - \overline{y})^2 \end{bmatrix} \tag{D.19}$$

Dado que la matriz $\mathbf{A}$ tiene elementos reales, es simétrica ($\mathbf{A}^t = \mathbf{A}$) y es semidefinida positiva ($\mathbf{v}^t\mathbf{A}\mathbf{v} \geq 0$ para $\mathbf{v} \neq \mathbf{0}$), la matriz $\mathbf{A}$ tiene dos eigenvalores reales : $\lambda_1 \geq 0$, $\lambda_2 \geq 0$; y dos eigenvectores ortonormales(vectores unitarios con producto interno igual a cero). Sea $\mathbf{u_1} = [u_{1,1}\ u_{2,1}]^t$ y $\mathbf{u_2} = [u_{1,2}\ u_{2,2}]^t$ las coordenadas de los eigenvectores $\mathbf{u_1}$ y $\mathbf{u_2}$. Eigenvalores and eigenvectores están relacionados con las ecuaciones:

$$\mathbf{Au_1} = \lambda_1 \mathbf{u_1} = [\lambda_1 u_{1,1}\ \lambda_1 u_{2,1}]^t \tag{D.20}$$

$$\mathbf{Au_2} = \lambda_2 \mathbf{u_2} = [\lambda_2 u_{1,2}\ \lambda_1 u_{2,2}]^t \tag{D.21}$$

Las ecuaciones (D.20) y (D.21) pueden ser reunidas en una relación simple:

$$\mathbf{A}\begin{bmatrix} u_{1,1} & u_{1,2} \\ u_{2,1} & u_{2,2} \end{bmatrix} = \begin{bmatrix} \lambda_1 u_{1,1} & \lambda_2 u_{1,2} \\ \lambda_1 u_{2,1} & \lambda_2 u_{2,2} \end{bmatrix}$$

$$\mathbf{A}\begin{bmatrix} u_{1,1} & u_{1,2} \\ u_{2,1} & u_{2,2} \end{bmatrix} = \begin{bmatrix} u_{1,1} & u_{1,2} \\ u_{2,1} & u_{2,2} \end{bmatrix}\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

Sea $\mathbf{U}$ una matriz cuya primer columna es $\mathbf{u_1}$ y segunda columna es $\mathbf{u_2}$; y sea $\mathbf{D}$ la matriz diagonal con los elementos λ_1 y λ_2. Utilizando estas matrices podemos escribir la ecuación de forma simple

$$\mathbf{AU} = \mathbf{UD} \tag{D.22}$$

La matriz ortonormal $\mathbf{U}$ tiene una propiedad interesante: su inversa es su transpuesta ($\mathbf{UU}^{-1} = \mathbf{UU}^t = \mathbf{I}$, donde $\mathbf{I}$ es la matriz identidad). Utilizando la misma propiedad y la ec. (D.22), la matriz $\mathbf{A}$ se puede expresar en términos de $\mathbf{U}$ y $\mathbf{D}$,

$$\mathbf{A} = \mathbf{UDU}^t \tag{D.23}$$

Reemplazando la matriz $\mathbf{A}$, dada la ec.(D.23), en la ec. (D.18),

$$\begin{aligned}
\chi^2(\ell; Z) = &\quad \tfrac{1}{\sigma^2}\mathbf{p}^t(\mathbf{UDU}^t)\mathbf{p} \\
\chi^2(\ell; Z) = &\quad \tfrac{1}{\sigma^2}(\mathbf{U}^t\mathbf{p})^t\mathbf{D}(\mathbf{U}^t\mathbf{p}) \\
\chi^2(\ell; Z) = &\quad \tfrac{1}{\sigma^2}[\mathbf{u_1}^t\mathbf{p}\ \mathbf{u_2}^t\mathbf{p}]\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}\begin{bmatrix} \mathbf{u_1}^t\mathbf{p} \\ \mathbf{u_2}^t\mathbf{p} \end{bmatrix} \\
\chi^2(\ell; Z) = &\quad \tfrac{1}{\sigma^2}\left\{\lambda_1(\mathbf{u_1}^t\mathbf{p})^2 + \lambda_2(\mathbf{u_2}^t\mathbf{p})^2\right\}
\end{aligned} \tag{D.24}$$

Para ver los valores máximo y mínimo de χ^2, supóngase que $\lambda_1 < \lambda_2$. Tomándose en cuenta que el producto interno de dos vectores con coordenadas $\mathbf{v_1}$ y $\mathbf{v_2}$ se encuentra definido como:

$$\mathbf{v_1^t v_2} = \|\mathbf{v_1}\|\|\mathbf{v_2}\|\cos\gamma \tag{D.25}$$

donde γ es el ángulo en ambos vectores. Dado que $\mathbf{u_1}$, $\mathbf{u_2}$ y $\mathbf{p}$ son la coordenadas de los vectores unitarios y $\mathbf{u_1}$ y $\mathbf{u_2}$ son vectores ortogonales, se tiene:

$$\mathbf{u_1}^t\mathbf{p} = \cos\alpha \tag{D.26}$$

$$\mathbf{u_2}^t\mathbf{p} = \cos(\alpha \pm \pi/2) = \pm\sin\alpha \tag{D.27}$$

donde α es el ángulo entre los vectores $\mathbf{u_1}$ y $\mathbf{t}$. Al reemplazar este resultado en la ecuación (D.24),

$$\begin{aligned}\chi^2(\ell; Z) = &\quad \tfrac{1}{\sigma^2}\left\{\lambda_1\cos^2\alpha + \lambda_2\sin^2\alpha\right\}\\ \chi^2(\ell; Z) = &\quad \tfrac{1}{\sigma^2}\left\{\lambda_1\cos^2\alpha + \lambda_2(1-\cos^2\alpha)\right\}\end{aligned} \tag{D.28}$$

Sea $s = \cos^2\alpha$, donde s es el valor en el rango $[0, 1]$, y $s = 1$ cuando el vector $\mathbf{p}$ es idéntico al vector $\mathbf{u_1}$ ($\alpha = 0$). Utilizando esta nueva variable s en la ec.(D.28), finalmente se obtiene

$$\begin{aligned}\chi^2(\ell; Z) = &\quad \tfrac{1}{\sigma^2}\left\{\lambda_1 s + \lambda_2(1-s)\right\}\\ \chi^2(\ell; Z) = &\quad \tfrac{1}{\sigma^2}\left\{(\lambda_1-\lambda_2)s + \lambda_2\right\}\end{aligned} \tag{D.29}$$

La expresión $(\lambda_1 - \lambda_2)s + \lambda_2$ corresponde a la línea $\langle a, b\rangle$ con pendiente negativa $a = \lambda_1 - \lambda_2$ (porque $\lambda_1 < \lambda_2$) y la intersección en el eje y, $b = \lambda_2$. El valor máximo de $\chi^2(\ell, Z)$ es $\frac{1}{\sigma^2}\lambda_2$ cuando $s = 0$, y el valor mínimo es $\frac{1}{\sigma^2}\lambda_1$ cuando $s = 1$. Por lo tanto,

> Vector $\mathbf{t} = \mathbf{u_1}$, el eigenvector asociado al menor eigenvalor λ_1 de la matriz $\mathbf{M}^t\mathbf{M}$, minimiza $\chi^2(\ell, Z)$.

De $\mathbf{u_1} = [u_{1,1}, u_{2,1}]^t$ y $\mathbf{p} = [\cos\phi \;\; \sin\phi]^t$ se obtiene el ángulo deseado ϕ,

$$\phi = \arctan(u_{2,1}/u_{1,1}) \tag{D.30}$$

Utilizando el cuarto cuadrante del arco tangente (atan2) para calcular ϕ (ec. (D.30)), y para el cálculo de r de la linea se utiliza la ec. (E.6).

Apéndice E

Ajustando puntos a una circunferencia

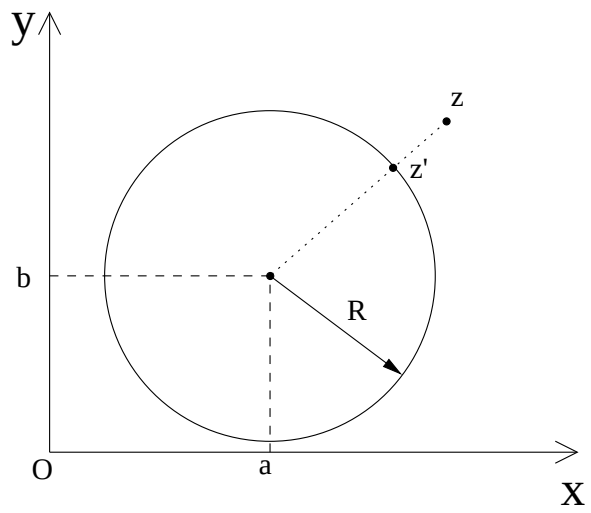


Figura E.1: Parámetros de una circunferencia

Una círcunferencia es una curva plana y cerrada, donde todos sus puntos están a igual distancia de un punto fijo llamado centro. Una circunferencia puede definirse por tres parámetros: (a, b, R) dónde (a, b) son las coordenadas (x, y) del centro y R denota la distancia euclidiana de los puntos de la circunferencia al centro (ver figura E.1). Así, los

puntos que forman la circunferencia cumplen la siguiente ecuación:

$$(x-a)^2 + (y-b)^2 = R^2 \tag{E.1}$$

E.1. Definición del problema

Sea $Z = \{z_1, \ldots, z_n\}$, un conjunto de n puntos, definidos por sus coordenadas cartesianas $z_i = (x_i, y_i)$. Si $n \geq 3$, y los puntos no siguen una línea recta, deseamos obtener la circunferencia (a, b, r) que minimize la suma (S) de cuadrados de distancias de los puntos z_i a sus correspondientes puntos más cercanos $z_i^{'}$ de la circunferencia (representados en la figura E.1 como z y $z^{'}$). Esto es,

$$S(a,b,R) = \sum_{i=1}^{n} \left[R - \sqrt{(x_i-a)^2 + (y_i-b)^2} \right]^2 \tag{E.2}$$

Enseguida se describe el método de mínimos cuadrados completo, el método de mínimos cuadrados reducido y el método de mínimos cuadrados modificado, contenidos en el artículo [Umbach y Jones, 2003]. El propósito de esta sección es mostrar claramente la derivación de las resultados que no se muestran con claridad en el artículo. En el artículo sólo se presentan las expresiones a minimizar, pero no la forma como se obtuvieron, en particular del método de mínimos cuadrados modificado que se utilizó en este trabajo.

E.2. Método de mínimos cuadrados completo

Para encontrar la circunferencia que minimize S en la ecuación E.2 podemos derivar S con respecto a a, b y r e igualar las derivadas a cero. Las derivadas parciales son las siguientes:

$$\frac{\partial S}{\partial r} = 2nR - 2\sum_{i=1}^{n}\sqrt{(x_i-a)^2+(y_i-b)^2} \tag{E.3}$$

$$\frac{\partial S}{\partial a} = 2na - 2\sum_{i=1}^{n}x_i + 2R\sum_{i=1}^{n}\frac{x_i-a}{\sqrt{(x_i-a)^2+(y_i-b)^2}} \tag{E.4}$$

$$\frac{\partial S}{\partial b} = 2nb - 2\sum_{i=1}^{n}y_i + 2R\sum_{i=1}^{n}\frac{y_i-b}{\sqrt{(x_i-a)^2+(y_i-b)^2}} \tag{E.5}$$

Sin embargo, al igualar a cero estas tres derivadas no es posible obtener una solución cerrada para los parámetros de la circunferencia, por lo que se utilizan métodos de aproximación numérica iterativos. A este método se le conoce como el *método de mínimos cuadrados completo*.

E.3. Método de mínimos cuadrados reducido

En la ecuación E.3, podemos igualar la derivada a 0 y despejar R,

$$R = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(x_i-a)^2+(y_i-b)^2} \tag{E.6}$$

Así, R es el promedio de las distancias de los puntos z_i al centro (a,b). Queda pendiente calcular el centro (a,b) a partir del conjunto de puntos Z.

Una idea de cómo obtener el centro (a,b) de la circunferencia se ilustra en la figura E.2(a). Si asumimos que los puntos z_1, z_2 y z_3 forman parte de la cirunferencia mostrada con centro (a,b), las líneas perpendiculares a los segmentos de recta entre z_1 y z_2, y entre z_2 y z_3 pasarán sobre el centro (a,b).

Asumamos que los puntos no están exáctamente sobre la circunferencia (cómo en la figura E.2(b)). Sea d_{ij} la distancia más corta entre el centro de la circunferencia (a,b) y la recta perpendicular a la recta que une los puntos z_i y z_j. Podemos formular el problema de elegir el centro (a,b) que minimize los cuadrados de las distancias d_{ij}, considerando todas las posibles parejas de puntos del conjunto Z. Esto es, se busca el centro (a,b) que minimize la suma D_r que sigue:

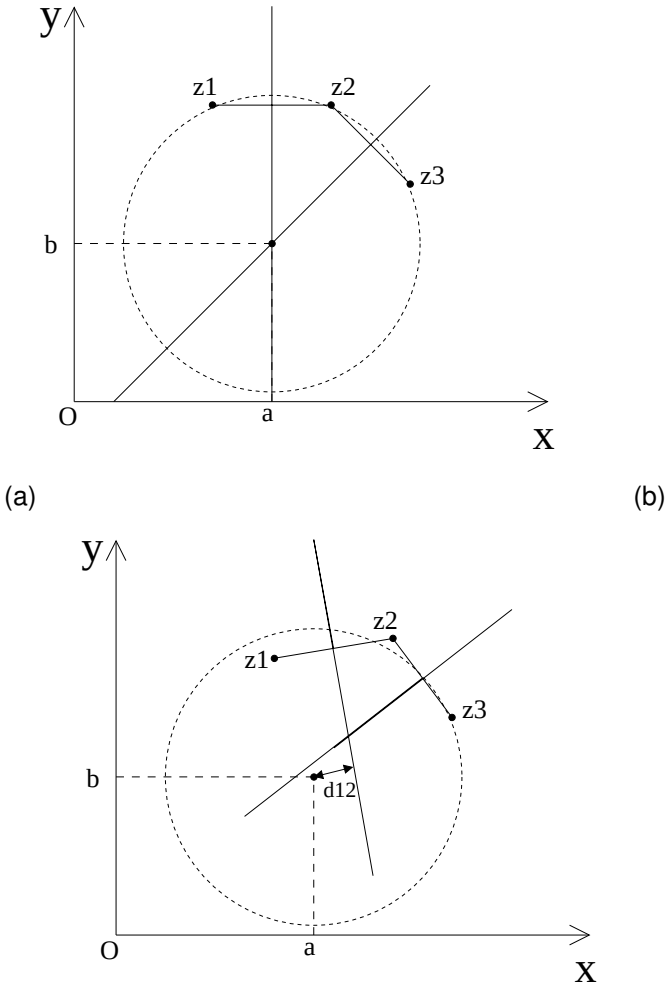


Figura E.2: Método modificado de mínimos cuadrados

$$D_r = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij}^2 \tag{E.7}$$

Para derivar una expresión para d_{ij} consideremos la figura E.3. Considerando la ecuación normal de una recta

$$r = xcos\phi + y \sin \phi \tag{E.8}$$

La cual también se puede expresar en forma matricial como

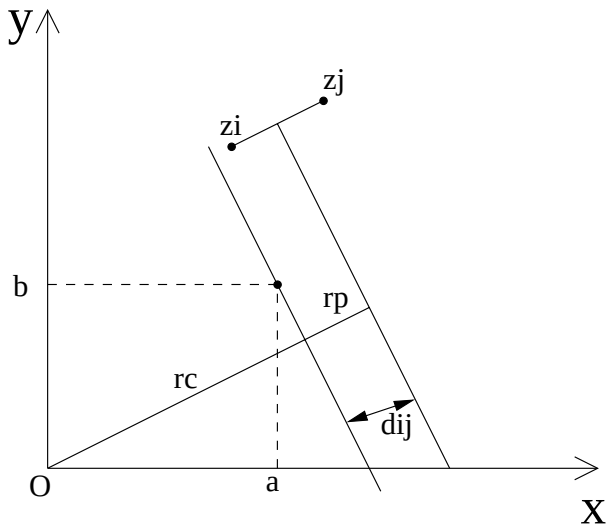


Figura E.3: Ilustración de la distancia d_{ij}

$$r = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} \cos\phi \\ \sin\phi \end{bmatrix} \tag{E.9}$$

Las coordenadas $[x\ y]$ son las coordenadas de un punto sobre la recta y $[\cos\phi\ \sin\phi]^t$ corresponde a un vector $\mathbf{u}$ ortogonal a la dirección de la recta que además es unitario (debido a que $\mathbf{u}^t\mathbf{u} = \cos^2\phi + \sin^2\phi = 1$).

La recta que une los puntos $z_i = (x_i, y_i)$ y $z_j = (x_j, y_j)$ cumple las dos condiciones siguientes:

$$r = x_j cos\phi + y_j \sin\phi \tag{E.10}$$

$$r = x_i cos\phi + y_i \sin\phi \tag{E.11}$$

Restando las dos anteriores ecuaciones, tenemos

$$0 = (x_j - x_i)\cos\phi + (y_j - y_i)\sin\phi \tag{E.12}$$

La cual también puede expresarse como el producto de dos vectores con las coordenadas siguientes:

$$0 = \begin{bmatrix} x_j - x_i & y_j - y_i \end{bmatrix} \begin{bmatrix} \cos\phi \\ \sin\phi \end{bmatrix} \tag{E.13}$$

El primer vector indica la dirección de la recta dados por los dos puntos z_i y z_j; y el segundo vector es el vector unitario ortogonal a la recta. Dado que su producto es 0, ambos vectores son ortogonales. Así, el vector con coordenadas $[x_j - x_i \ y_j - y_i]^t$ corresponde al vector ortogonal de la recta perpendicular a la que va de los puntos z_i a z_j, la recta que estamos buscando. Para convertir este vector a un vector unitario, basta multiplicarlo por $1/\sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$. Reuniendo lo anterior, tenemos que la recta que pasa por un punto $[x \ y]^t$ con este nueva dirección ortogonal se define por la ecuación:

$$r = \frac{x(x_j - x_i) + y(y_j - y_i)}{\sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}} \tag{E.14}$$

Volviendo a considerar la figura E.3, podemos calcular la distancia buscada

$$d_{ij} = r_p - r_c \tag{E.15}$$

donde r_p es la distancia mínima entre la recta perpendicular que pasa por el punto medio entre z_i y z_j y el origen; y r_c es la distancia mínima del origen a una recta paralela a la perpendicular mencionada, pero que ahora pasa por el punto $[a \ b]^t$.

Utilizando la ecuación E.14 y tomando en cuenta que pasa por el punto medio entre z_i y z_j, dado por $[\frac{x_j+x_i}{2} \ \frac{y_j+y_i}{2}]$, tenemos

$$r_p = \frac{\frac{x_j+x_i}{2}(x_j - x_i) + \frac{y_j+y_i}{2}(y_j - y_i)}{\sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}} \tag{E.16}$$

$$r_p = \frac{(x_j^2 - x_i^2) + (y_j^2 - y_i^2)}{2\sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}} \tag{E.17}$$

Considerando ahora la recta que pasa por el centro de la circunferencia $[a \ b]^t$, tenemos:

$$r_c = \frac{a(x_j - x_i) + b(y_j - y_i)}{\sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}} \tag{E.18}$$

Sustituyendo estos resultados en la ecuación E.15, tenemos

$$d_{ij} = \frac{\frac{1}{2}[(x_j^2 - x_i^2) + (y_j^2 - y_i^2)] - [a(x_j - x_i) + b(y_j - y_i)]}{\sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}} \tag{E.19}$$

Finalmente, sustituyendo en la ecuación E.7, tenemos:

$$D_r = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{\left(\frac{1}{2}[(x_j^2 - x_i^2) + (y_j^2 - y_i^2)] - [a(x_j - x_i) + b(y_j - y_i)]\right)^2}{(x_j - x_i)^2 + (y_j - y_i)^2} \tag{E.20}$$

Al igual que en el método de mínimos cuadrados completo, descrito en la sección anterior, al igualar las derivadas parciales de D_r con respecto a los parámetros a y b, no se produce una solución cerrada para a y b; y por lo tanto se pueden utilizar métodos numéricos iterativos. En el artículo [Umbach y Jones, 2003] se menciona que este método no es muy estable, en particular cuando el denominador involucra puntos muy cercanos entre si. Por esta razón se introduce el método de mínimos cuadrados modificado en la siguiente sección.

E.4. Método de mínimos cuadrados modificado

Para minimizar el impacto de los puntos que están muy cercanos entre si, este método considera que el denominador de la ecuación E.20 es unitario. Esto es, la función a minimizar es la siguiente:

$$D_m = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(\frac{1}{2}[(x_j^2 - x_i^2) + (y_j^2 - y_i^2)] - [a(x_j - x_i) + b(y_j - y_i)]\right)^2 \tag{E.21}$$

Como se muestra en el artículo [Umbach y Jones, 2003], al derivar con respecto de a y de b, se obtienen un sistema de dos ecuaciones lineales en a y b, que al resolverse se obtienen fórmulas cerradas para calcular a y b.

Este método resulta ser robusto, con desempeño comparable al método de mínimos cuadrados completo (MCC), pero mucho más rápido; al utilizar una fórmula cerrada,

en lugar de los métodos iterativos. En [Umbach y Jones, 2003] se reporta que para un caso de 8 puntos, el método de MCC requirió 67000 operaciones de punto flotante, mientras que el método de mínimos cuadrados modificado requiere 230 operaciones. Una diferencia muy signficativa.

Apéndice F

Códigos

La aplicación trace2Gcode y su complemento imageCNC fue desarrollada en c++, los códigos son presentados en la página Web
`https://lc.fie.umich.mx/~msantana/index.php?mod=master`
ahí se encuentran los siguientes archivos:
Programa complemento imageCNC:

1. **imageCNC.cpp.** Aquí están las funciones encargadas de corregir las posibles distorsiones proyectivas.

Programa trace2Gcode:

2. **buscaTrazos.cpp.** Contiene las rutinas para llevar los píxeles de la imagen a trazos.
3. **algoritmos.cpp.** Se ubican las funciones con los algoritmos de Seguimiento de línea y Back Tracking.
4. **hacerCirculo.cpp.** Están los procesos para convertir líneas a círculos o segmentos de círculos.
5. **gcode.cpp.** Presenta el procedimiento para convertir las líneas y círculos a código G, así como la optimización de los movimientos.
6. **trace2Gcode.cpp.** Es el archivo principal que hace el llamado a las funciones de los 4 archivos anteriores.

Además se encuentran disponibles los archivos para los casos de prueba presentados en el capítulo 4 y los códigos G presentados en el capítulo C.

Referencias

[Arras y Siegwart, 1997] Arras, K. O. y Siegwart, R. Y. Feature extraction and scene interpretation for map-based navigation and map building. *En Proc. of SPIE, Mobile Robotics XII*. 1997.

[Asato et al., 2002] Asato, O. L., Kato, E. R. R., Inamasu, R. Y., y Porto, A. J. V. Analysis of open cnc architecture for machine tools. *J. Braz. Soc. Mech. Sci.*, 24(3):208–212, 2002.

[AUTOMATION, 2010] AUTOMATION, F. *CNC 8070 Programming*, 2010. `http://www.fagorautomation.com/`.

[Borges y Aldon, 2004] Borges, G. y Aldon, M.-J. Line extraction in 2d range images for mobile robotics. *Journal of Intelligent and Robotic Systems*, 40:267–297, 2004.

[Deming, 1943] Deming, G. C. *Data Reduction and Error Analysis for the Physical Sciences*. New Yoor: Wiley, 1943.

[EMCO, 2003] EMCO. *Manual SINUMERIK 810/820 M*, 2003. `http://www.emco-world.com/uploads/tx_commerce/Sinumerik810820_Mill_en.pdf`.

[Fernandez et al., 2010] Fernandez, C., Moreno, V., Curto, B., y Vicente, J. A. Clustering and line detection in laser range measurements. *Robotics and Autonomous Systems*, 40:720–726, 2010.

[Félix Ledo Pernas, 2006] Félix Ledo Pernas, A. C. A. *Teoría y problemas resuelos en programación control númerico*. Marcombo, ISBN: 9788426713827, primera ed$^{\underline{\text{ón}}}$., 2006.

[Golub, 1973] Golub, G. H. Some modified matrix eigenvalue problems. *Siam Review*, 15(2):318–334, 1973.

[Harris y Stephens, 1988] Harris, C. y Stephens, M. A combined corner and edge detector. *En Proceedings of the 4th Alvey Vision Conference*, págs. 147–151. 1988. http://www.bibsonomy.org/bibtex/22d2048f92453ed8d1426782cfe774b62/zap.

[Horath, 1993] Horath, L. *Computer Numerical Control: Programing of Machines*. Prentice Hall, 1993.

[Instruments, 2012] Instruments, N. *Detailed Specifications Nema 23*, 2012. http://sine.ni.com/ds/app/doc/p/id/ds-311/lang/es.

[International Federation of Robotics, 2012] International Federation of Robotics, S. D. World robotics industrial robots 2012: Statistc, market analysis, forescast and case studies. taipei, taiwan, 20 august 2012. 2012. http://www.worldrobotics.org/uploads/tx_zeifr/Charts_IFR__30_August_2012.pdf.

[International Federation of Robotics, 2013] International Federation of Robotics, S. D. Industrial robot statistics. 2013. http://www.ifr.org/industrial-robots/statistics/.

[Krystek y Anton, 2007] Krystek, M. y Anton, M. A weighted total least-squares algorithm for fitting a straight line. *Measurement Science and Technology*, 18(11):3438, 2007.

[Laganiere, 2011] Laganiere, R. *OpenCV 2 Computer Vision Application Programming Cookbook*. Packt Publishing, 2011.

[Leonardo Romero, 2013] Leonardo Romero, C. A. L., Moises Garcia. *An Extended Line Tracking Algorithm*. Power, Electronics and Computing (ROPEC), 2013 IEEE International Autumn Meeting on, 2013. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6702752.

[LinuxCNC, 2013] LinuxCNC. *LinuxCNC: Software for realtime control*, 2013. http://www.linuxcnc.org/.

[Markovsky y Van Huffel, 2007] Markovsky, I. y Van Huffel, S. Overview of total least-squares methods. *Signal processing*, 87(10):2283–2302, 2007.

[Motor, 2013] Motor, L. *Products*, 2013. `http://www.longs-motor.com/`.

[Nguyen et al., 2007] Nguyen, V., Gächter, S., Martinelli, A., Tomatis, N., y Siegwart, R. A Comparison of Line Extraction Algorithms using 2D Range Data for Indoor Mobile Robotics. *Autonomous Robots*, 23(2):97–111, August 2007.

[Nievergelt, 1994] Nievergelt, Y. Total least squares: State-of-the-art regression in numerical analysis. *SIAM Rev.*, 36(2):258–264, jun. 1994. ISSN 0036-1445. doi: 10.1137/1036055.
URL `http://dx.doi.org/10.1137/1036055`

[Probotix, 2013] Probotix. *Sale items*, 2013. `http://www.probotix.com/`.

[Ramírez, 2008] Ramírez, C. A. O. *Tesis de maestría: Sistema CNC de Corte por Láser*. Instituto Politécnico Nacional, 2008.

[Smid, 2008] Smid, P. *CNC programming handbook: a comprehensive guide to practical CNC programming*. Industrial Press, ISBN: 9780831133474, 2008.

[Steve F. Krar, 2001] Steve F. Krar, P. S., Arthur Gill. *Computer Numerical Control Simplified*. Industrial Press, ISBN: 9780831131333, 2001.

[Umbach y Jones, 2003] Umbach, D. y Jones, K. N. A few methods for fitting circles to data. *IEEE Transactions on Instrumentation and Measurement*, 52(6), 2003.

[Van Huffel y Vandewalle, 1991] Van Huffel, S. y Vandewalle, J. *The total least squares problem: computational aspects and analysis*, tomo 9. Siam, 1991.

[Vergara Reyes, 2009] Vergara Reyes, D. M. La innovación tecnológica en la manufactura mexicana en el período 2001-2006: un análisis empírico. *En Memorias del Congreso Internacional de Sistemas de Innovación para la Competitividad 2009*, pág. 25. 2009. `http://www.humanindex.unam.mx/humanindex/fichas_pdf/Ponencia29777%20VergaraReyes%20La%20innovacion%20tecnologi.pdf`.