UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA
INGENIERÍA ELÉCTRICA – SISTEMAS ELECTRÓNICOS

AUTOMATIZACIÓN DE UNA PLATAFORMA DE MAQUINADO CON VISIÓN ARTIFICIAL PARA UN ROBOT KUKA INDUSTRIAL

TESIS
QUE PARA OPTAR POR EL GRADO DE:
MAESTRO EN INGENIERÍA

PRESENTA:
ING. OSCAR MOLOTLA GARCÉS

TUTOR PRINCIPAL:
DR. JUAN MARIO PEÑA CABRERA
IIMAS

Ciudad Universitaria, CD. MX. Enero 2020

Resumen

En el trabajo desarrollado a continuación, se automatizó una plataforma de maquinado para un robot KUKA industrial, que adquiere y proporciona el punto de inicio de maquinado de forma automática con el manejo de software y hardware. Y además a través de un sistema embebido se desarrollo e implemento un sistema de visión artificial, que permite llevar a cabo dicho proceso. Para la etapa de diseño y manufactura se emplearon herramientas de software CAD/CAM.

Índice general

Capítulo 1

Introducción

1.1. Planteamiento del problema.

Una de las aplicaciones de los robots KUKA es el maquinado de materiales blandos (madera, porexpan, aluminio, etc). Para maquinar una pieza se debe contar con, el diseño de la misma, el código con las trayectorias de corte y el punto de inicio de maquinado. Esté último proceso se realizan normalmente de forma manual por algún operador, lo que lo hace de riesgo, lento y con poca o nula flexibilidad; la operación manual involucra repetir los mismos pasos, cada vez que se desea remplazar la pieza por otra con dimensiones diferentes. Dado esté problema, es deseable automatizar el proceso para adquirir el punto de inicio con visión artificial, a través del manejo de software y hardware (hardware and software driving) que logre una producción mas ágil y flexible respondiendo a la nueva producción de la industria 4.0, basada en herramientas tecnológicas y digitales.

1.2. Justificación

Las aplicaciones de los robots KUKA en la industria son múltiples,por ejemplo en el manejo de materiales, la carga y descarga de máquinas, la soldadura por puntos, la soldadura de arco y el maquinado de piezas, siendo está última aplicación en donde se a realizado un desarrollo reciente del mecanizado con robots. Se creía que el mecanizado robótico no podía reemplazar al maquinado CNC, pero actualmente se considera una herramienta alternativa viable, ya que el Robot tiene la capacidad de cambiar fácilmente la orientación de la herramienta y colocarla casi cualquier lugar que se requiera.

Se elige automatizar el proceso de manufactura, ya que el maquinado con robots se ha incrementado de forma importante en los últimos años, y esto a

llevado a realizar diversas investigaciones y desarrollos entorno a está área. Estos sistema se encuentran en muchas áreas de desarrollo e investigación, lo cual se debe a que son flexibles y pueden implementarse en diferentes procesos.

Un paso importante en la manufactura es obtener el punto de inicio de maquinado, por medio del manejo de software y hardware es que se desea automatizar el proceso manufactura, el cual adquirirá el punto de inicio de maquinado con visión artificial, siendo implementado esté en un sistema embebido, el cual nos permiten llevar a cabo diferentes modificaciones, sin la necesidad de reconfigurarlo para hacer algún cambio, asiendo un proceso de manufactura flexible.

1.3. Objetivos.

1.3.1. Objetivo general.

Se plantea desarrollar un sistema que adquiera y proporcione el punto de inicio de maquinado de forma automática con visión artificial, para posteriormente realizar el proceso de manufactura, esto utilizando herramientas tanto de software como hardware.

1.3.2. Objetivos específicos.

- Diseño y manufactura asistido por computadora para el stock.
- Diseño de algoritmo para obtener el punto de inicio de maquinado.
- Conexión remota con el sistema de visión y el Robot.
- Diseño del sistema de visión artificial embebido.
- Implementación del sistema de visión para automatizar el proceso de manufactura.
- Experimentos y resultados del sistema embebido en el robot.

1.4. Metodología.

- Investigación acerca de sistemas de visión artificial.
- Investigación sobre técnicas de programación del robot KUKA.

- Implementación de rutinas para el robot KUKA.
- Investigación y empleo de software CAD/CAM.
- Investigación sobre técnicas de PDI y sistemas embebidos.
- Propuesta de algoritmo de PDI para obtener el punto de inicio de maquinado.
- Investigación sobre la conexión remota con el software KSS.
- Desarrollo del sistema de visión para automatización del proceso de manufactura.
- Implementación del sistema de visión para automatización del proceso de manufactura.
- Ajustes y mejoras del sistema.
- Experimentos y resultados.

1.5. Metas

- Aprender sobre el uso y manejo del Robot KUKA.
- Aprender algoritmos y técnicas de programación para el Robot KUKA.
- Aprender sobre el manejo de software CAD/CAM.
- Aprender sobre el maquinado de piezas.
- Aprender algoritmos y técnicas para programación de PDI.
- Aprender algoritmos de programación para sistemas embebidos.
- Aprender sobre la conexión remota con el software KSS.

1.6. Infraestructura

- Lugar de trabajo en el laboratorio de electrónica y Automatización para la industria 4.0 que pertenece al Departamento de Ingeniería de Sistemas Computacionales y Automatización (DISCA) del Instituto de Investigación en Matemáticas Aplicadas y Sistemas (IIMAS).
- Robot KUKA KR 5-2 y controlador KRC4.

- Celda de manufactura.
- Equipo de computo.
- Mastercam y Robotmaster.

1.7. Estado del arte

Los primeros estudios sobre el maquinado con robots se informó en la década de 1990. A pesar de que hay investigaciones mundiales continuas sobre esta área desde entonces, el potencial de las aplicaciones de robots en esté campo aún no se ha realizado por completo. Dicho desarrollo puede clasificar básicamente en investigaciones sobre la planificación de rutas, el seguimiento de rutas, la compensación, el desarrollo de sistemas tanto de maquinado de robots como de sistemas inteligentes. Estás investigaciones mejorarán la precisión y eficiencia de los robots sobre el maquinado y proporcionarán referencias útiles en el desarrollo de sistemas en esté campo, en tareas que alguna vez se pensó que solo podrían ser realizadas por máquinas CNC. Las industrias modernas dependen en gran medida de los robots que tienen una amplia gama de aplicaciones, como la transferencia de materiales, el ensamblaje de precisión, la soldadura y el mecanizado. En las aplicaciones que implican el pre-mecanizado de piezas, los robots realizan diversos procesos con tolerancias menores. Se creía que el mecanizado robótico no podía reemplazar al maquinado de control numérico por computadora (CNC), para aplicaciones de tres a cuatro ejes, pero actualmente se considera una herramienta alternativa viable e inmediata para materiales no metálicos y metales según el grado de dureza[1] [2] [3] [4] [5] [6] [7] [8].

Al darse cuenta del potencial del mecanizado robótico, los principales fabricantes de robots industriales del mundo (ABB, Yaskawa, Fanuc, Kuka) están comenzando a proporcionar robots para maquinado junto con paquetes de software. Incluso los desarrolladores de software tradicionales de diseño asistido por computadora (CAD) y de manufactura asistida por computadora (CAM), han incorporado capacidades de mecanizado para robots en sus paquetes de software CAD / CAM tradicionales. Uno de ellos es MasterCAM que ha dado una solución en software (Robotmaster) para fresado y corte en robots. Robotmaster puede crear trayectorias precisas de un robot de seis ejes a partir de los datos de ruta asignados a una herramienta. Los errores de singularidad, colisión, fuera de alcance y extensión de articulación se verifican cuando se genera la trayectoria del robot. La funcionalidad de Robotmaster ha reflejado más o menos los logros de las primeras investigaciones sobre la

planificación de rutas de mecanizado de robots automáticos [4] [9] [10] [11] [12] [13].

Antes de ser aplicado al mecanizado directo, los brazos robóticos se utilizaban para trabajos relacionados con el maquinado. Algunos estudios han demostrado que los robots pueden funcionar bien en pulido, rectificado y desbaste. El propósito principal del pulido es generar una superficie brillante o lisa, y no modificar las dimensiones de la pieza. Las herramientas de pulido a menudo son suaves o flexibles, por lo que el requisito de precisión y posición no es muy alto. Esto ha creado una excelente oportunidad para que los robots sobresalgan en las operaciones de pulido, ya que el robot articulado puede colocar fácilmente la herramienta de pulido en cualquier posición que se necesite. Se probó que el robot genera una mejor superficie de rectificación y pulido frente al fresado de tres ejes, lo que se debe principalmente a la capacidad del robot de cambiar fácilmente la orientación de la herramienta [2] [3] [5] [6] [7] [8].

Uno de los sentidos más importantes de los seres humanos es la visión. De hecho, se sabe que gran parte de las tareas que realiza el cerebro son empleadas en el análisis de la información visual. Casi todas las disciplinas científicas emplean utillajes gráficos para transmitir conocimiento. Por ejemplo, en Ingeniería Electrónica se emplean esquemas de circuitos, a modo gráfico, para describirlos. Se podría hacerlo mediante texto, pero para los humanos nos resulta mucho más eficiente procesar imágenes que procesar texto. Se podría decir que el primer acercamientos al procesamiento de imágenes se dio en la década de los años 20 del siglo XX. Las fotografías periodísticas entre Europa y América tardaban una semana en llegar a través de los barcos. Al emplear las primeras técnicas de procesamiento de las imágenes se pasó sólo a tres horas. Las imágenes se codificaban a cinco niveles de grises y se transmitían por teléfono. Este podría ser el principio de las técnicas de procesamiento de las imágenes. Se podrían citar muchos más ejemplos que durante décadas han transformado la percepción de la ciencia con el procesamiento de las imágenes, alguna veces por separado y otras de forma multidisiciplinar. Sin embargo, el momento histórico que hace que estas técnicas confluyan y den un cuerpo de conocimiento propio, surge en la década de los 80 del siglo XX. La revolución de la Electrónica, con las cámaras de vídeo CCD y los microprocesadores, junto con la evolución de las Ciencias de la Computación hace que sea factible la Visión Artificial [6] [14] [15] [16] [17] [18] [19].

El control de calidad en muchas ocasiones se vale de la ayuda de instrumentos, pero aún hoy en día muchos procesos de inspección se ejecutan basados totalmente en la experiencia de un inspector y su agudeza visual para reconocer defectos en los productos. La migración hacia sistemas de visión automatizados permite ejecutar mediciones precisas, de mayor complejidad,

a mayor velocidad y en una mayor cantidad de la que un humano puede realizar. Cada aplicación de control de calidad por visión puede ser diferente y requerir instrumentos y algoritmos muy particulares, inclusive pensado en la inspección de un mismo producto. Por ejemplo, en el caso de la extracción dimensional y geométrica de piezas mecanizadas es necesario considerar que estas piezas presentan brillos intensos y con alta reflexión de la luz. Como la medición está relacionada con los contornos de las mismas, es necesario evitar degradarlos durante el pre-procesamiento de las imágenes. Otro de los aspectos importantes es la calibración del sistema, paso crucial que tiene como objetivo principal determinar la resolución del sistema y evitar errores significativos en la medición. A continuación se presentan los aspectos más relevantes e información importante sobre los algoritmos usados en cada una de las etapas del procesamiento de la imagen [6] [20] [21] [14] [15] [16].

El proceso de captura está centrado en tres aspectos importantes, necesarios para obtener una imagen de calidad que no altere las condiciones reales agregando errores considerables al sistema. El primero de ellos es la iluminación, la cual debe ser homogénea sin llegar a saturar o crear brillos que puedan ocultar defectos en los contornos de las piezas. Otro aspecto importante son los sistemas ópticos, ya que sufren de un número inevitable de distorsiones geométricas que alteran las proporciones reales de las dimensiones de los objetos en la imagen. Aunque existen múltiples algoritmos para obtener los parámetros de distorsión del lente, es importante para aplicaciones de medición contar con una lente de baja distorsión con el fin de reducir el tiempo de procesamiento y aumentar la exactitud del sistema. Finalmente, la cámara en sí es el elemento más importante en el proceso de captura y debe ser seleccionada de acuerdo con la aplicación, la capacidad para manipular los parámetros internos de captura y las características de resolución, velocidad de captura, envío de datos al computador y, finalmente, el acondicionamiento de la misma para trabajar en ambientes industriales en el caso de sistemas que estén diseñados para realizar inspecciones en la planta de producción [6] [14] [15] [16] [17] [20] [21].

El pre-procesamiento consiste en la aplicación de técnicas que permitan el realce o mejoramiento de algunas características importantes en las imágenes originales para facilitar el proceso de segmentación. En un sistema de medición y de inspección es necesario aplicar técnicas que conserven los contornos pero que ejecuten labores de suavizado porque en el caso específico de los materiales mecanizados se presentan surcos debido al paso de la herramienta. El problema de filtros que preserven bordes ha sido enfocado por varios trabajos y proponen una combinación de filtrado de intervalo y de dominio, denominada filtrado bilateral, el cual reemplaza el valor del píxel en x por un promedio de los valores de píxeles vecinos, pero, a diferencia de

otros filtros, los valores para el cálculo del promedio deben ser similares. En regiones suaves, los valores de los píxeles en un vecindario pequeño son todos muy similares, por lo que el filtro bilateral actúa como un filtro de dominio estándar, promediando y eliminando los valores débilmente correlacionados causados por el ruido. Por el contrario, si el filtro se ubica en un borde o frontera, solamente se tienen en cuenta aquellos valores más parecidos al valor del píxel evaluado según su posición en la frontera [15] [16] [21].

La técnica de segmentación más conocida dentro del procesamiento de imágenes es quizás la umbralización. Este método permite separar dos o más regiones de una imagen a partir de un análisis del histograma. Aunque es un método sencillo, su aplicabilidad en procesos industriales ejecutados por visión artificial ha sido y sigue siendo de gran importancia. Por otro lado, un análisis orientado a bordes permite extraer características más puntuales en cuanto a dimensiones, forma y orientación de los objetos inspeccionados. Existen diversas técnicas en el dominio espacial para encontrar los bordes entre dos regiones. Después de obtener los bordes de las regiones de interés, es necesario seguir segmentando la imagen con el fin de obtener unas primitivas de forma para la evaluación de algoritmos de medición e inspección. Uno de los aspectos más interesantes de un contorno son las esquinas, puntos críticos o puntos de alta curvatura. En la literatura se encuentra una gran cantidad de trabajos orientados a la detección de estos puntos y hacen un recuento de las técnicas de detección de esquinas y puntos de alta curvatura [6] [14] [15] [16].

Capítulo 2

Operación e implementación de Software CAD/CAM

Hoy en día, más y más empresas están buscando nuevas formas de automatizar sus procesos. Para llevar acabo algunas de estas tareas utilizan la tecnología CAD/CAM, con la cual desarrollan diseños que logren solucionar sus necesidades, al mismo tiempo ahorran recursos que pueden invertir en otras áreas. Las siglas CAD y CAM provienen de su denominación en inglés. Para diseñar se usa el Computer Aided Design, mientras que para la fabricación o manufactura se emplea el Computer Aided Manufacturing. Las aplicaciones CAD/CAM se utilizan para diseñar un producto y para programar los procesos de manufactura, especialmente el mecanizado por CNC [13] [27].

Históricamente los CAD comenzaron como una ingeniería tecnológica de computo, mientras los CAM eran una tecnología semiautomática para el control numérico de máquinas. Pero estas dos disciplinas se han ido mezclando gradualmente hasta conseguir una tecnología suma de las dos, de tal forma que los sistemas CAD/CAM son considerados, hoy día, como una disciplina única identificable.

2.1. MasterCAM

Mastercam es el programa CAD/CAM más popular para manufactura en máquinas de control numérico y centros de maquinado CNC. El software abarca la programación de fresadoras, centros de maquinado, tornos, electro erosionadoras de corte por alambre, cortadoras por láser, oxicorte, routers, y más.

Mastercam ofrece una gama de módulos para aplicaciones especiales, tam-

bien incluye modulos de modelado 3D con producción de dibujos 2D para la preparacion de la geometria 3D antes del CAM.

2.1.1. Principales características

La suite de diseño cuenta con una serie de herramientas de diseño de superficies 3D y sólidos, con un motor geométrico CAD que hace el trabajo de diseño más fácil. El diseño de Mastercam permite tanto modelar como editar la geometría, también se puede crear geometría avanzada, incluyendo curvas, dimensionamiento asociativo, extensiones de superficies, blends, filets variables, sólidos, modelado híbrido y más.

2.1.1.1. Interfaz de diseño de Mastercam

La pieza a manufacturar necesita al menos un tipo de máquina antes comenzar a crear trayectorias de corte con la herramienta, para lo cual debemos elegir una, para esté caso seleccionamos una fresadora, esto a través del menú de selección de maquina. Mastercam crea automáticamente un grupo de máquinas para cada una de las seleccionadas, y dichos grupos de máquinas almacenan trabajos completos de cada una de ellas. Por ejemplo, si algunas rutas de herramientas se cortarán en un torno y otras rutas de herramientas en una fresadora, simplemente se crear un segundo grupo de máquinas. Cada grupo de máquinas puede almacenar su propia información y herramientas de configuración de trabajo, y usar un conjunto diferente de valores predeterminados de trayectoria.

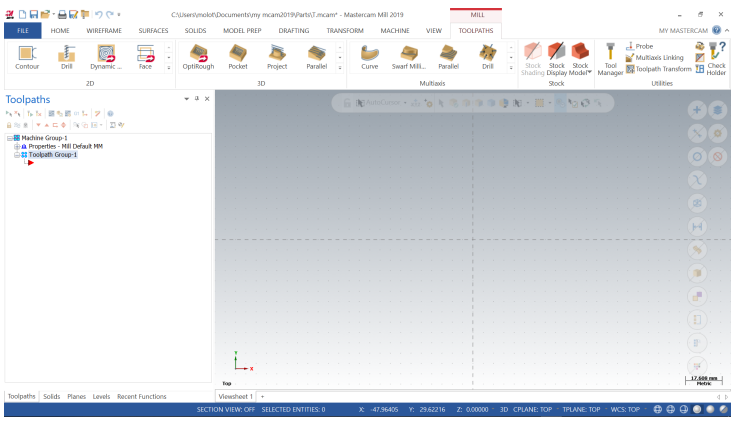


Figura 2.1: Selección del tipo de maquina

Ya cargado el tipo de maquina, podemos pasar al diseño de la pieza. A través de la barra de herramientas, en la pestaña *WIREFRAME*, elegimos

Shapes y seleccionamos *Rectangle*. Se mostrará una ventana para configurar el rectángulo y en la pestaña *Dimensions* ingresamos las medidas del rectángulo. En *Setings* activamos *Anchor to center* que anclará el centro del rectángulo al origen de Mastercam, después damos clic en dicho punto, el rectángulo se mostrará. En la figura 2.2 se muestra la ventana de configuración del rectángulo.

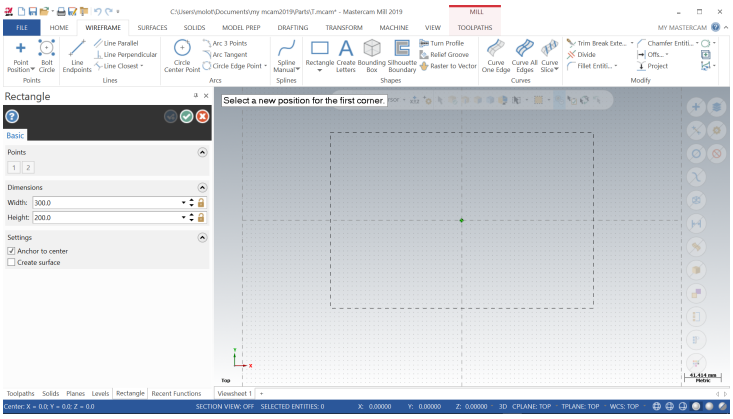


Figura 2.2: Configuración del rectángulo

Para elegir el material o stock con el cual se va a trabajar, desde la ventana de Toolpaths, en *Machine Group 1*, elegimos *Stock setup*. Se muestra la ventana de *Machine Group Properties* y en la pestaña *Stock Setup* elegimos las dimensiones tanto de Y como X y en Z se ingresa el grosor del material a trabajar. En la figura 2.3 se muestra la imagen. En la pestaña *Stock Origin* seleccionamos donde queremos que sea el origen.

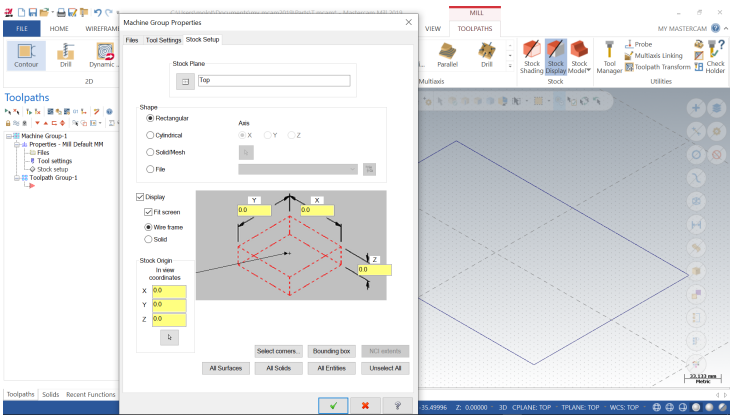


Figura 2.3: Configuración del Stock

Ya agregado el material, elegimos el tipo de mecanizado a utilizar, en la pestaña *TOOLPATHS* seleccionamos el modo *Contour* y se genera la ventana de *Chaining* que nos permite elegir que partes del stock queremos mecanizar, elegimos la opción *Window* y posteriormente seleccionamos todo el rectángulo, a lo que se generara un aviso indicando que seleccionemos un punto de aproximación para iniciar la manufactura, el cual a su vez podría ser el punto de inicio de mecanizado, para este caso elegimos la esquina inferior izquierda, en la figura 2.4 se muestra una imagen del proceso.

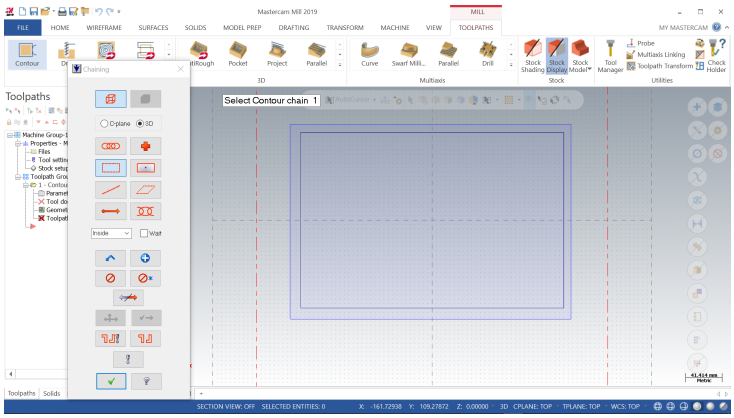


Figura 2.4: Selección del contorno y punto de aproximación del rectángulo

Se genera la ventana de *2D Toolpaths-Contur*, que nos permite configurar los parámetros de maquinado. Como primer paso, en la sección *Toolpaths Type*, elegimos *Contur*, lo que realizará un mecanizado de contorno, como se muestra en la figura 2.5.

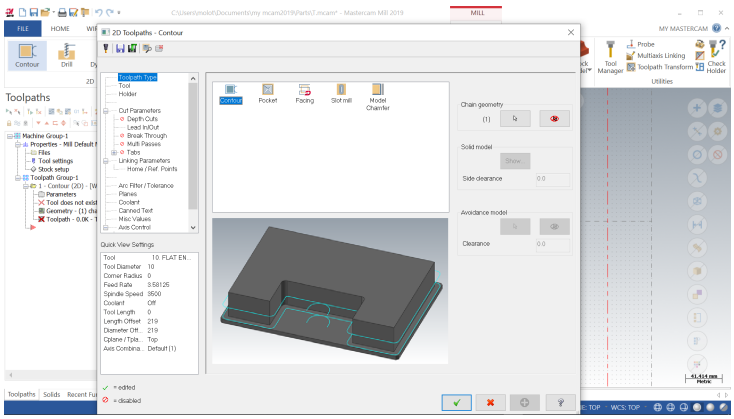


Figura 2.5: Selección del tipo de trayectoria

En el apartado *Tool* elegimos la herramienta que más se ajuste a la que tenemos, seleccionamos *Selec library tool* y posteriormente, en la ventana que se genera, elegimos filtrar la herramienta. Ya en *Tool list filter*, marcamos una fresa de acabado y damos ok para terminar el filtrado, generándose una lista de herramientas con las características elegidas. De dicha lista seleccionamos cualquiera de las herramientas generadas.

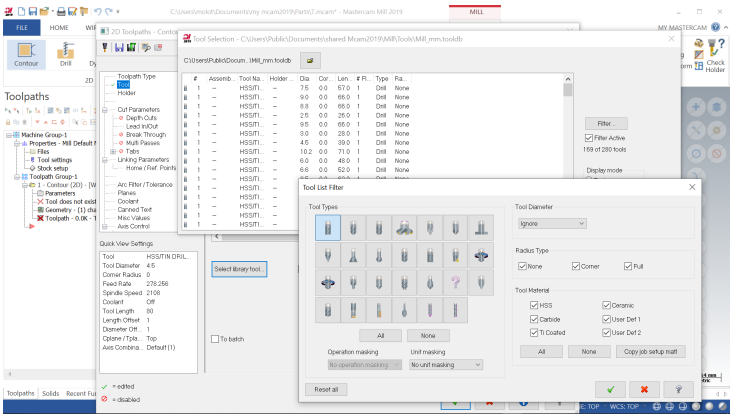


Figura 2.6: Sección de librería de herramienta

El siguiente parámetro a configurar es el tipo Holder a utilizar, aquí se muestran diferentes tipos de holder y con diversas medidas, elegimos el que más se asemeje o podemos configurarlo nosotros para obtener una simulación mas precisa. En la figura 2.7 se muestra el cabezal del spindle utilizado.

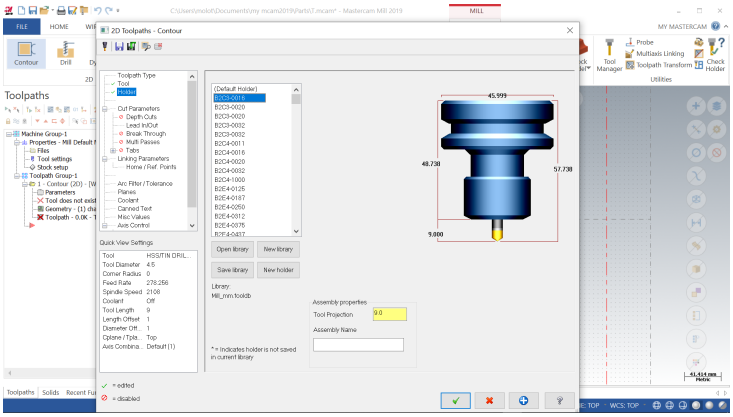


Figura 2.7: Selección del tipo de Holder

Ya en la sección *Cut parameters*, elegimos el tipo de compensación que queremos utilizar, a partir de las cinco opciones posibles; para esté caso

seleccionamos por computadora y en la dirección de compensación elegimos derecha, ya que así realizará el corte del rectángulo por la parte exterior de esté.

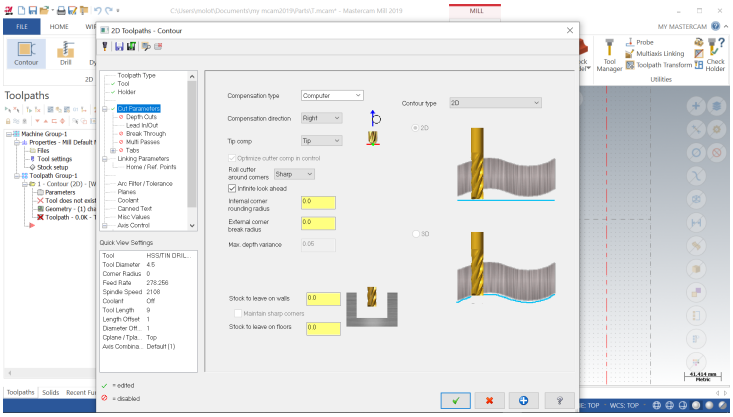


Figura 2.8: Sección de parámetros de corte

De la lista de *Cut parameters* seleccionamos *Depth cuts*, en la cual configuraremos los cortes por pasada que realizará la fresadora. En este caso elegimos que sea de $2mm$ por pasada y con un corte final de $1mm$ para liberar la pieza del stock.

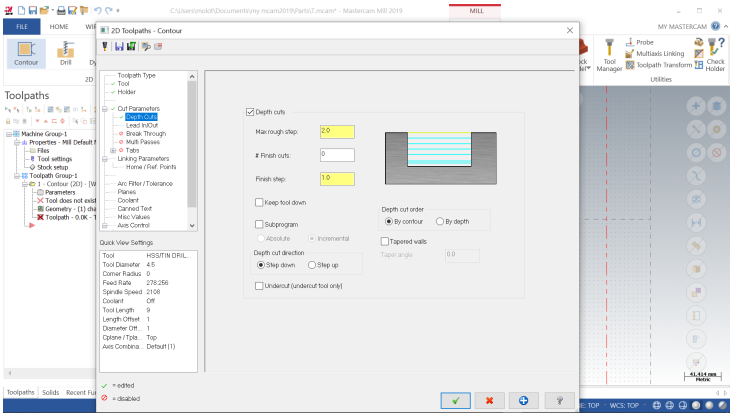


Figura 2.9: Configuración de Cortes por pasada

Finalmente en el apartado *Linkig parameters*, seleccionamos el elemento *Retract*, cuyo valor indica cuantos mm se levantará la fresa entre una inmersión y otra, realizándolo a una velocidad de avance. El *Feed plane* da la distancia a la cuál el spindle cambiará de velocidad de avance a velocidad de inmersión. En la sección *Top of stock* se oprime el botón *Top of stock* para

indicar la parte superior del material, se ingresa la profundidad del stock a cotar en el elemento *Depth* y finalmente, damos ok en la parte inferior derecha de la ventana. En la siguiente figura se muestra la ventana de *Linkig parameters*.

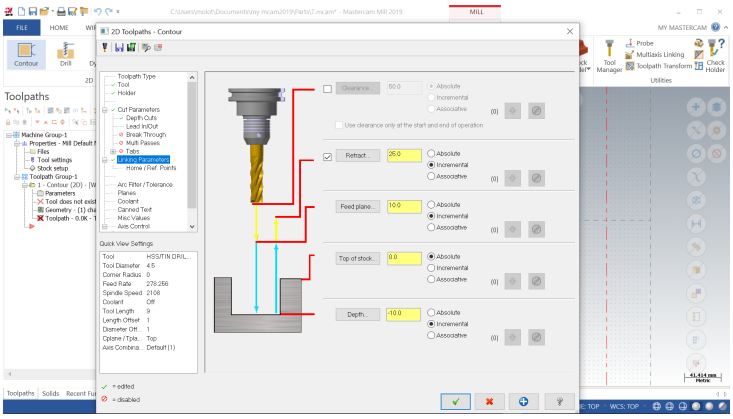


Figura 2.10: Configuración de vinculación de parámetros

Seguido de este proceso de configuración de parámetros de corte, se muestra ya en el espacio de trabajo el maquinado del rectángulo, donde las líneas azules representan el contorno que seguirá la fresadora para cortar la pieza; en esté ejemplo son 6, las 5 primeras líneas corresponden a cada una de las pasadas que realizara la fresadora y la ultima linea sera la que libere el rectángulo del stock. La línea amarilla indica el primer punto donde comenzará a ejecutar las trayectorias de corte del material.

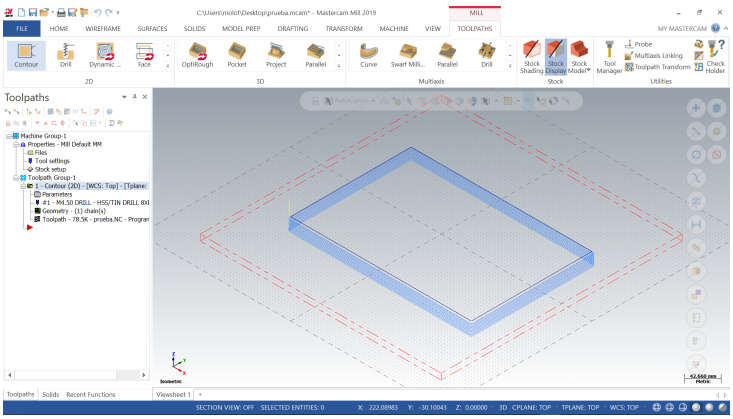


Figura 2.11: Trayectorias de corte

2.1.2. Simulación

Mastercam Machine Simulation es una forma segura y rentable de probar trayectorias de herramientas de fresado de 3, 4 o 5 ejes. La simulación ayuda a detectar colisiones entre el stock, la herramienta y los componentes de la máquina antes de enviar el código a está. Se puede usar la simulación de máquina para probar posibles escenarios de fijación y encontrar la ubicación ideal para un trabajo en particular. Es una herramienta adicional para ayudar a crear programas de ruta de herramienta limpios, eficientes y precisos.

2.1.2.1. Simulación simple

Existen dos formas de simular el proceso de maquinado del stock. La primera se elige en la ventana *Toolpaths*, donde seleccionamos *Backplot*, en la cual se generará un interfaz para iniciar una simulación en el área de diseño. En la figura 2.12 se ilustra dicha simulación. La barra superior que se encuentra a un costado de *Toolpaths*, muestra el botón *play* que inicia la simulación, y aun costado de esté para detener la simulación se encuentra el de *stop*, está misma barra nos permite ajustar la velocidad del maquinado, lo que ayuda a observar con mayor detenimiento el proceso de simulación.

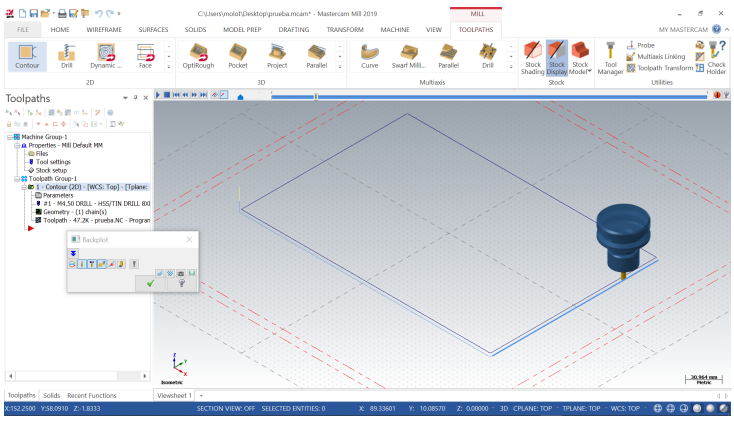


Figura 2.12: Simulación desde el área de diseño

2.1.2.2. Simulación con maquina

Para realizar una simulación en 3D, seleccionamos *Verify* que se encuentra a un costado de *Backplot* desde la ventana *Toolpaths*. Esta simulación genera una interfaz donde se muestra tanto el stock como el mandril a utilizar en la simulación. A través de la cual se elige tanto la precisión como la velocidad de la simulación. A un costado se muestra la ventana con la lista

de los movimientos que se generan en la simulación y en la parte inferior la información de los recorridos o trayectorias de corte. En esta misma interfaz es posible obtener diferentes vistas, y tener un reporte de colisiones y diferentes tipos de asistentes para mejorar la simulación. En la figura 2.13 se muestra dicha interfaz.

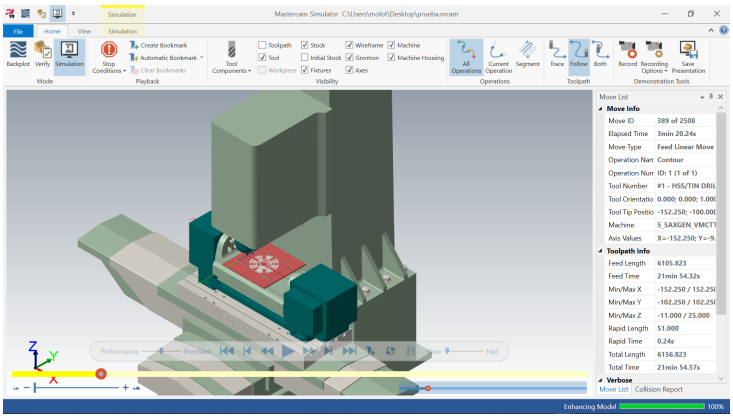


Figura 2.13: Simulación desde la interfaz

2.2. Robotmaster

Robotmaster esta integrado dentro de la plataforma de Mastercam, permitiendo la programación, simulación y generación de código para el robot, esto al disponer de una sólida base CAM que logra abordar de una forma revolucionaria la programación de robots.

Robotmaster es compatible con la mayoría de marcas de robots industriales, lo cual hace posible programar casi cualquier tipo de robot fuera de línea de producción con herramientas CAD/CAM. Esto hace que la programación sea más eficiente a diferencia de los métodos tradicionales que enseñan a los robots de producción, con procesos de ensayo y error, lo cual consume mucho tiempo.

Robotmaster proporciona una programación sencilla y potente, además de brindar una programación fuera de línea, reduciendo drásticamente el tiempo de programación y obteniendo un control preciso de los movimientos de corte, para tareas tales como el recorte, corte, soldadura, devastado, pulido, rectificación, rociado, pintura y maquinado 3D, con una rápida generación del código para el robot y una intervención mínima del programador.

2.2.1. Principales características

Una de las principales características de Robotmaster es la optimización, lo que ofrece una visualización rápida de problemas y opciones para obtener programas de robot optimizados de forma fácil. Su interactividad completa lo convierte en una herramienta integral para encontrar la mejor solución, sin intervención punto por punto, incluso a través de una estrecha banda de oportunidad, en lugar del habitual enfoque de prueba y error.

2.2.2. Interfaz de configuración de Robotmaster

Como primer paso para iniciar cualquier simulación en Robotmaster, se debe acceder a través de la barra de herramientas de Mastercam y ya dentro establecer las configuraciones que iniciarán el proceso de simulación.

2.2.2.1. Interfaz de configuración global de Robotmaster

Configuración de parámetros del **Robot**: ya dentro de Robotmaster a través la barra de herramientas en la pestaña *Home*, seleccionamos *Global* y se abrirá la ventana de configuración de parámetros del robot.

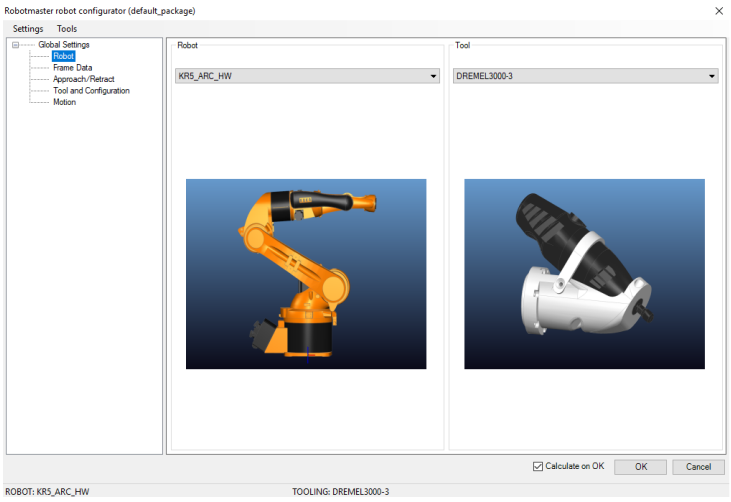


Figura 2.14: Selección de parámetros del Robot

Configuración de parámetros de **Frame Data**: es aquí donde se determina la ubicación de la pieza respecto al Robot, ya que algunos robots tienen el origen en la base y otros en la intersección de los ejes 1 y 2. Los robots industriales típicos tiene el eje de coordenadas de tal manera que el eje X apunta directamente hacia la parte delantera del robot, el eje Y apunta hacia

la izquierda del robot y el eje Z apunta hacia la parte superior del robot, en el caso del KUKA KR5 se encuentra en el origen de esté. Dado lo anterior en la sección *Base Data*, es donde indicaremos a que distancia se encuentra el stock respecto al Robot, con la ayuda de las coordenadas X,Y,Z y W,P,R. En el apartado *Tool Data*, se debe ingresar la posición y orientación de la herramienta con respecto al extremo (brida) del manipulador.

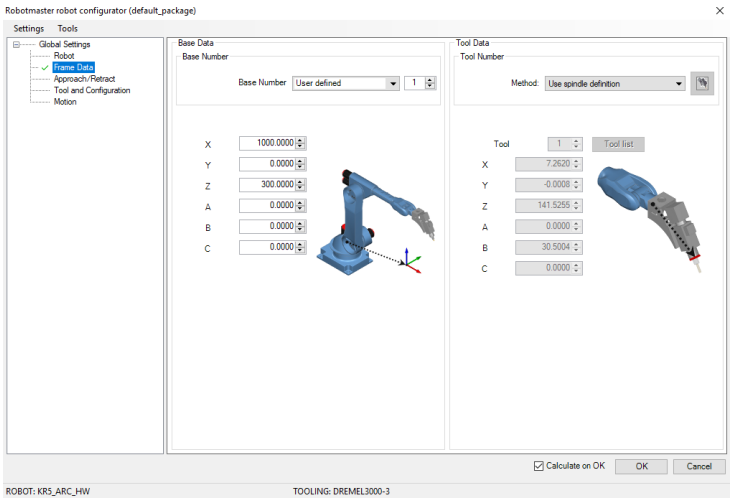


Figura 2.15: Sección de Frame Data

Configuración de parámetros de **Aproach/Retract**: en esta página agregaremos la posición en la cual iniciará y terminará la manufactura el robot.

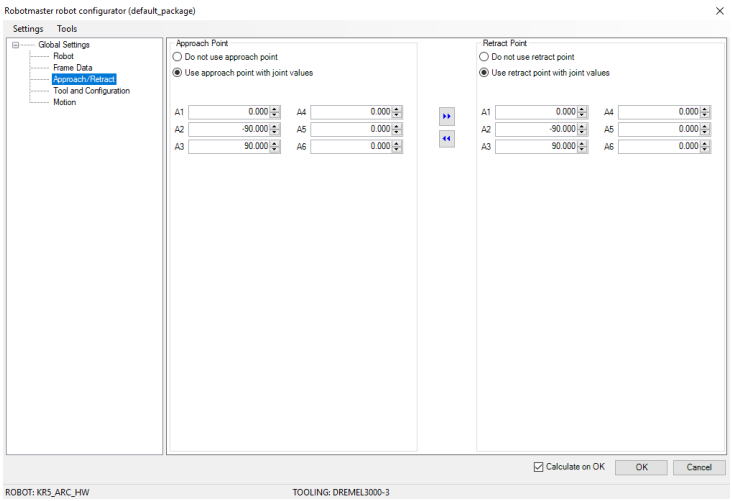


Figura 2.16: Sección de Aproach/Retract

Estos puntos de aproximación y retracción son los 6 valores correspondientes a cada uno de los ejes. En *Approach Point* se indica el punto con que se aproximará el Robot para iniciar la manufactura y en *Retrac Point* se elige el punto a donde se retirará el manipulador.

Configuración de parámetros de **Tool and Configuration**: como primer paso se configura la herramienta, indicandose la llamada a herramienta y configurando la activación de la misma. En el campo *Robot Configuration* se debe agregar el posicionamiento del Robot en el maquinado, indicando la posición de la base, el codo y la muñeca.

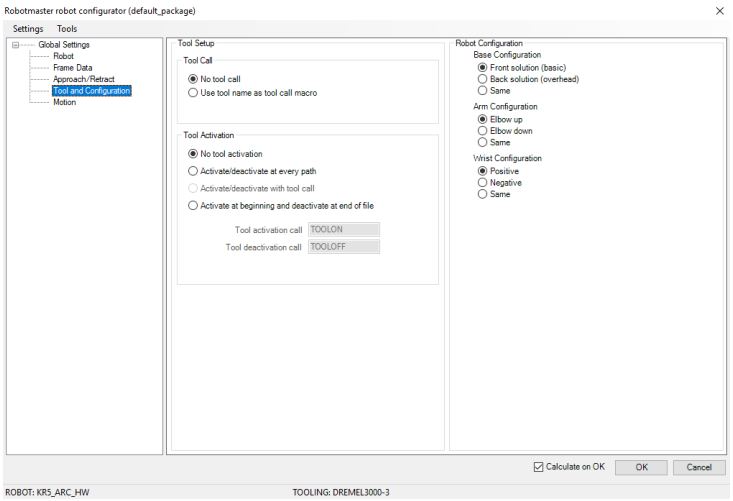


Figura 2.17: Sección de Tool and Configuration

2.2.2.2. Configuración local de Robotmaster

La configuración local se usa para establecer parámetros específicos de la operación. Y está compuesta por la configuración del eje, parámetros de optimización, retracción segura, reposo y eje externo.

Figura 2.18: Operación 1 y configuración Local.

Se selecciona la operación a simular de la lista y está se resaltara. En la pestaña *Home* damos clic en la opción *Local* y se abrirá la ventana de ajustes locales, donde se establece la orientación de la herramienta del robot con respecto a la pieza a cortar y administra la rotación alrededor del vector de la herramienta.

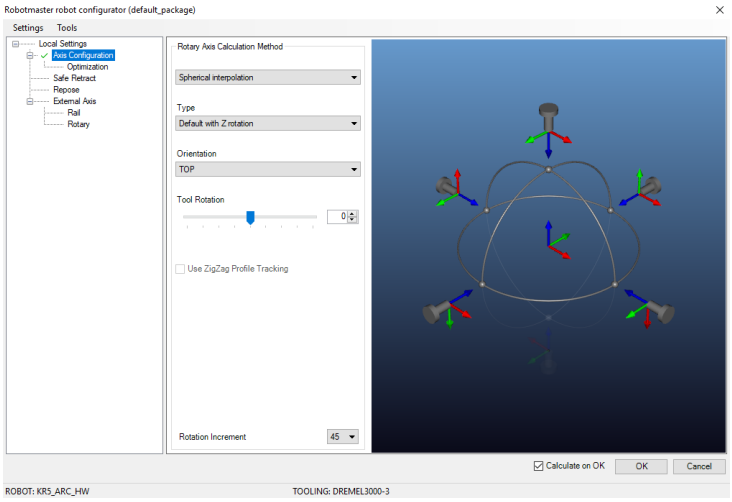


Figura 2.19: Interfaz de ajustes locales.

En el método del cálculo del eje de rotación, seleccionamos la interpolación esférica; para este método existen dos posibles elecciones: que la herramienta rote con respecto a Z y que no rote con respecto a Z. El apartado de orientación tiene tres diferentes planos para orientar al robot Top, Front y Right. En la figura 2.20 se muestra un ejemplo de cada uno de los planos y finalmente se puede elegir rotar la herramienta, si fuese necesario establecer algún giro al spindle.

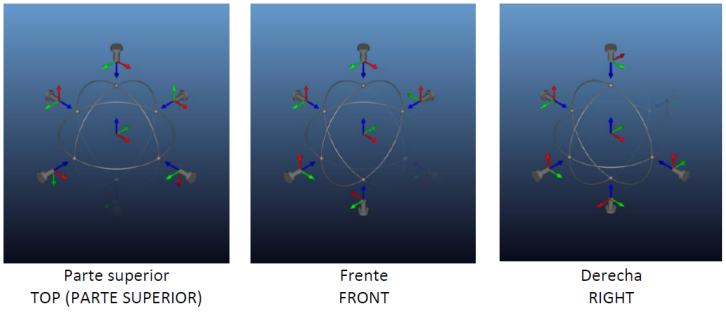


Figura 2.20: Tipos de orientación de la herramienta.

Ya realizado los procesos anteriores, queda como ultimo paso procesar y validar la operación. Desde la barra de herramientas, en la sección simulación, damos clic en calcular. Y si no existen problemas cinemáticos una marca de verificación verde se coloca al lado de la Operación 1, lo que nos indica que el proceso de simulación esta completado y se puede iniciar. En la figura 2.21 se muestra parte de la barra de herramientas, donde se puede ver encerrado en un circulo rojo el botón de calcular y en un rectángulo verde la validación de la operación 1.

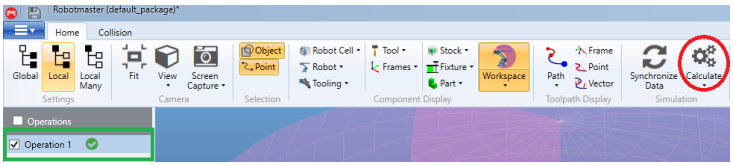


Figura 2.21: Validación de la operación 1

La ventana de simulación que se muestra en la figura 2.22, cuenta con una barra de progreso, una lista de puntos que corresponden a las trayectorias de corte, una barra de control para la simulación, un control de velocidad y un control de eje que muestra el valor del angulo de cada unión.

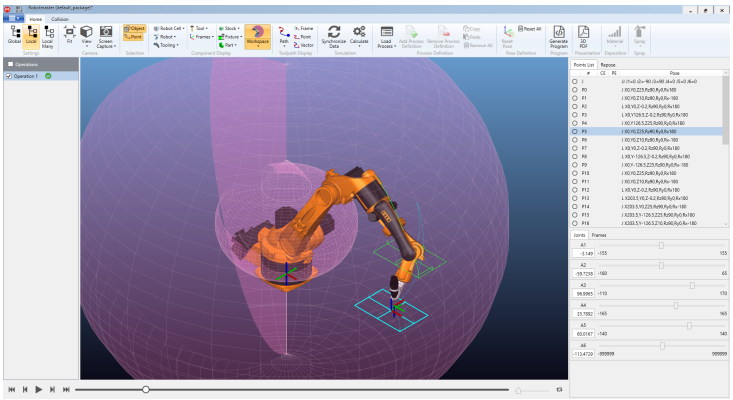


Figura 2.22: Ventana de simulación de Robotmaster

2.2.2.3. Post procesador de Robotmaster

El botón Generar programa ejecuta el post procesador de Robotmaster para crear un archivo de programa listo para el robot, el cual contiene cada uno de los parámetros configurados tanto en Mastercam como en Robotmaster, que genera todas las trayectorias de corte para cada una de las operaciones implementadas en la manufactura de la pieza. Este post procesador

también permite editar el código resultante con lo que se obtiene una mayor flexibilidad al implementar el programa en el manipulador. El tipo de código dependerá de la marca del robot seleccionado. En la imagen 2.23 se muestra la interfaz con el código generado.

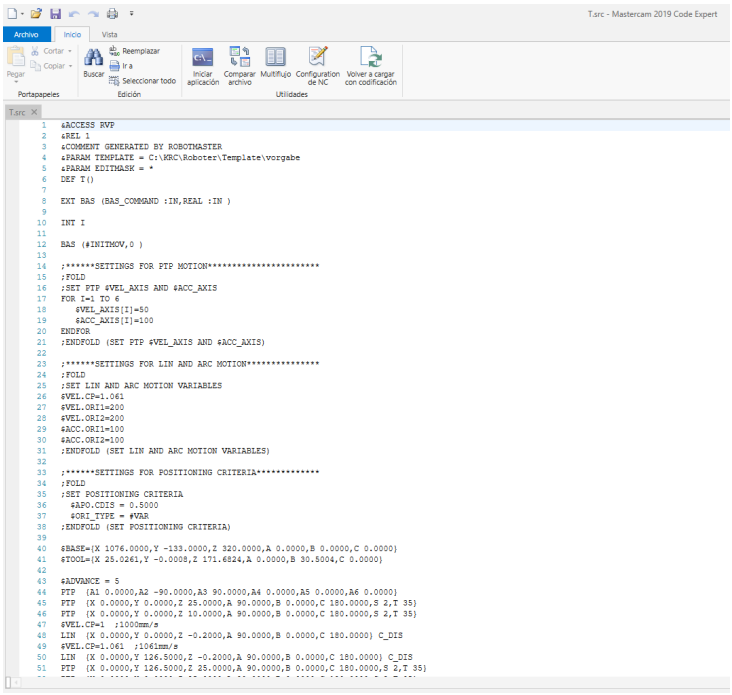


Figura 2.23: Interfaz de pos procesador de Robotmaster

Capítulo 3

Operación y programación del robot

El robot KUKA KR5 es un brazo industrial soldador, con una excelente precision que garantiza una maxima calidad en el proceso; aunado a esto el diseño robusto del brazo aumenta su rigidez y por tanto disminuye las vibraciones. Estas características en combinación con reductores de alta precision logra que el robot alcance una repetibilidad del $\pm 0{,}4mm$. Algunas características y datos se muestran en las figuras 3.1 y 3.2 respectivamente[22] [23].

Type	KR 5 arc HW
Maximum reach	1,423 mm
Rated payload	5 kg
Suppl. load, arm/link arm/rotating col.	12/–/20 kg
Suppl. load, arm + link arm, max.	–
Maximum total load	37 kg
Number of axes	6
Mounting position	Floor, ceiling
Variant	–
Positioning repeatability*	±0.04 mm
Path repeatability*	
Controller	KR C2 edition2005
Weight (excluding controller), approx.	126 kg
Temperature during operation	+10 °C to +55 °C
Protection classification	IP 54
Robot footprint	324 mm x 324 mm
Connection	7.3 kVA
Noise level	< 75 dB

Figura 3.1: Características del robot

Axis data	Range (software)	Speed with rated payload 5 kg
Axis 1 (A1)	±155°	156°/s
Axis 2 (A2)	+65°/–180°	156°/s
Axis 3 (A3)	+170°/–110°	227°/s
Axis 4 (A4)	±165°	390°/s
Axis 5 (A5)	±140°	390°/s
Axis 6 (A6)	Infinitely rotating	858°/s

Figura 3.2: Datos de los ejes

El robot industrial esta formado principalmente por cuatro elementos; el manipulador (robot de 6 grados de libertad), el controlador (driver para el manipulador, y la interfaz entre esté y la interfaz HMI de KUKA), el SmartPAD (tiene todas las funciones de control y visualización requeridas para operar y programar el robot industrial) y, finalmente, los cables de conexión. En la figura 3.3 se muestra una imagen de estos elementos [22] [24].

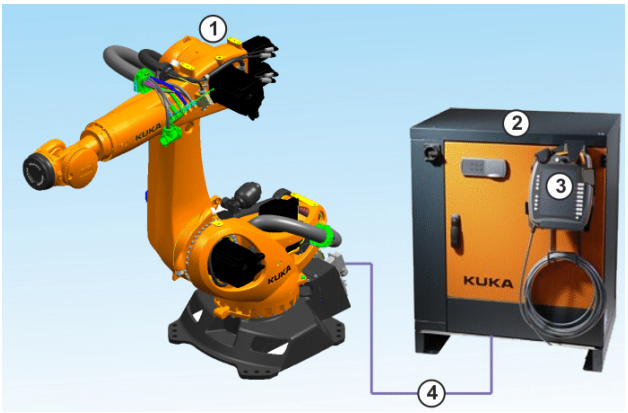


Figura 3.3: 1 Manipulador, 2 Controlador del robot, 3 SmartPAD y 4 Cables de conexión

3.1. KUKA System Software (KSS)

El software del sistema KUKA (KSS) es responsable de todas las funciones básicas de control que el operador puede realizar sobre el robot industrial; manipularlo a través de las teclas para cada eje, administrar los programas, administrar entradas y salidas, accesar a diferentes parámetros del manipulador y visualizar de los mismos.

El software del controlador KRC4 se ejecuta en el sistema operativo Microsoft Embedded Windows 7. La consola de programación muestra una

"HMI"(Human Machine Interface), está interfaz maquina humano desarrollada por KUKA permite al usuario manipula el robot.

3.1.1. Interfaz de usuario KUKA smartHMI

La interfaz de usuario de KUKA muestra una barra de estado, un contador de mensajes, un representador del estado del mouse, un indicador del estado de las teclas de trabajo, las teclas de trabajo para cada eje, la barra de botones y otros elementos necesarios para la operación del robot. En la figura 3.4 se muestra una imagen de KUKA smartHMI, donde se visualiza en la columna izquierda las carpetas, las cuales pueden ser creadas o modificadas y en donde se puede generar dos tipos de archivos; DAT (contiene datos permanentes y coordenadas de puntos) y SCR (contiene el código del programa).

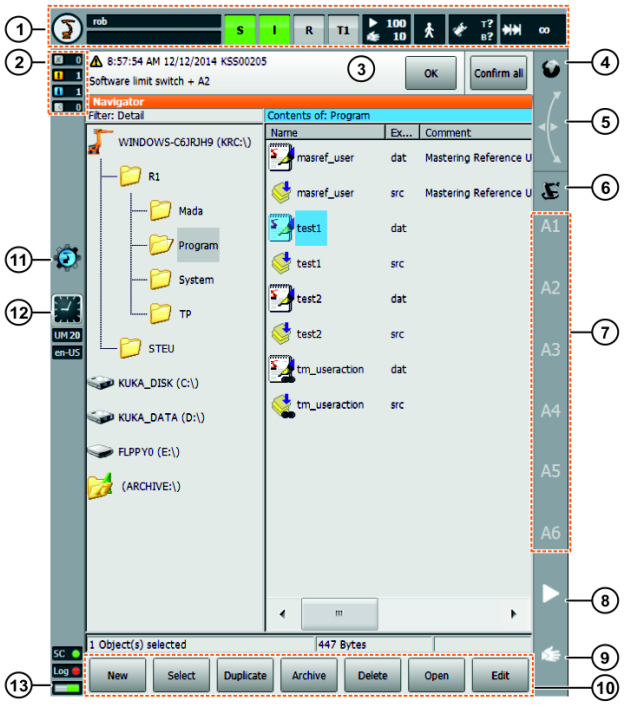


Figura 3.4: Interfaz de usuario KUKA smartHMI

A través de la barra de estado se accede al menú principal, donde se muestra el indicador de estado, el estado de los drivers, el interprete del indicador de el estado del robot y modo de operación. En la figura 3.5 se muestra una imagen de la barra.

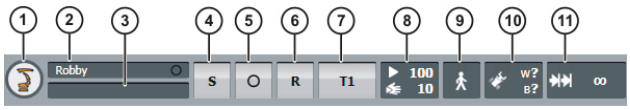


Figura 3.5: Barra de estado KUKA smartHMI

3.1.2. Calibración de herramienta y mesa

3.1.2.0.1. La calibración de herramienta consiste en una serie de pasos que permite calibrar la herramienta montada en la brida y darle al robot la referencia respecto a ésta, en la figura 3.6 se esté proceso. En el proceso de calibración, el usuario asigna el sistema de coordenadas de la herramienta y el punto de origen, el cual es llamado TCP (Tool center point). El TCP esta situado generalmente en el punto de trabajo de la herramienta. Algunas ventajas de la calibración son:

- La herramienta se puede mover en una linea recta en la dirección de la herramienta.
- La herramienta puede rotar cerca del TCP, sin cambiar dicha referencia.
- En modo programa: la velocidad programada se mantiene en el TCP a lo largo de la ruta.
- X, Y, Z: el origen del sistema de coordenadas de la herramienta es relativo al sistema de coordenadas de la brida.
- A, B, C: la orientación del sistema de coordenadas de la herramienta es relativo al sistema de coordenadas de la brida.

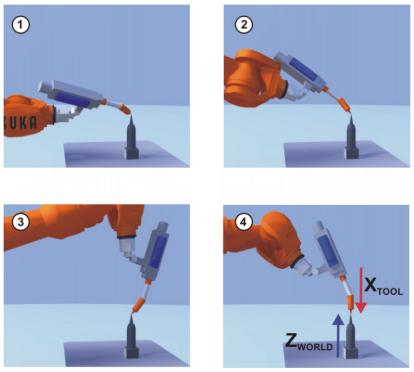


Figura 3.6: Calibración de la herramienta

La calibración de la herramienta consiste en dos pasos y para cada uno existen dos métodos.

- **Definición del origen del sistema de coordenadas de la herramienta.**
  - Método por 4 puntos.
  - Método por referencia.
- **Definición de la orientación del sistema de coordenadas de la herramienta.**
  - Método por 2 puntos.
  - Método por mundo.

3.1.2.0.2. La calibración de la base consiste en una serie de pasos, donde el usuario asigna un sistema de coordenadas cartesianas (sistema de coordenadas BASE) a una superficie de trabajo o la pieza de trabajo. El sistema de coordenadas BASE tiene su origen en un punto definido por el usuario. Algunas ventajas de calibrar la base son:

- El TCP se puede mover a lo largo de la superficie de trabajo o la pieza de trabajo.
- Si se desplaza la base, debido a que el área de trabajo fue desplazada, lo puntos asignados anteriormente se mueven con ella.

Se pueden almacenar hasta un máximo de 32 sistemas de coordenadas BASE. Existen dos formas de calibrar la base, en la figura 3.7 se muestra el método por 3 puntos.

- método por 3 puntos.
- método por referencia.

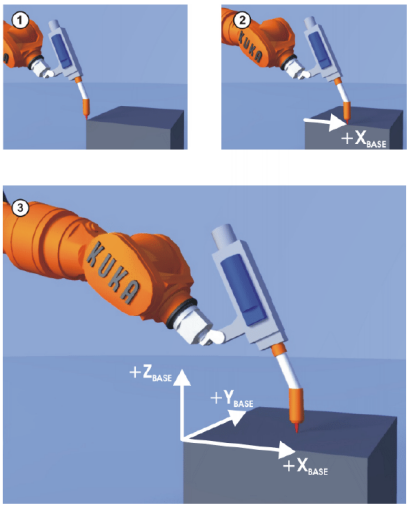


Figura 3.7: Calibración de la base

3.2. Programación de rutinas para el robot

El software KSS tiene dos principales modos de operación principales, usuario y experto; en el primero, algunos archivos del sistema son invisibles, mientras que para el modo experto permite verlos y también editarlos en la ventana de programas. Además, junto a los nombres de archivo y comentarios, en el nivel del experto se puede mostrar también las extensiones, atributos y tamaños de los archivos. Otra de las grandes diferencias que existe es en la forma de programación,ya que el modo usuario solo permite programar entre otras cosas algunos movimientos básicos a través de comandos ya establecidos, y en modo experto se puede acceder a todo el set de instrucciones del software KSS.

La ultima diferencia que existe entre estos dos modos es la ejecución de programas: el modo usuario solo nos permite operar en T1 Y T2, mientras que en modo experto se puede trabajar en T1, T2, AUT y AUT EXT, en la se resaltan las principales características de estos modos ejecución . En la figura 3.8 se muestra un ejemplo de un programa en modo experto[22] [24].

- **Modo de operación (T1)**: para pruebas de funcionamiento, programación y enseñanza, velocidad maxima de operación $250mm/s$.

- **Modo de operación (T2)**: para pruebas de funcionamiento, pero sin restricción de velocidad.

- **Modo de operación (AUT)**: para robots industriales sin controladores de alto nivel, sin restricción de velocidad.

- **Modo de operación (AUT EXT)**: para robot industriales con controladores de alto nivel, sin restricción de velocidad.

```
DEF PROG1()

;----- Sección de declaraciones -----
INT J

;------ Sección de instrucciones -------
$VEL_AXIS[1]=100   ;Especificación de las velocidades de ejes
$VEL_AXIS[2]=100
$VEL_AXIS[3]=100
$VEL_AXIS[4]=100
$VEL_AXIS[5]=100
$VEL_AXIS[6]=100

$ACC_AXIS[1]=100   ;Especificación de las aceleraciones de ejes
$ACC_AXIS[2]=100
$ACC_AXIS[3]=100
$ACC_AXIS[4]=100
$ACC_AXIS[5]=100
$ACC_AXIS[6]=100

PTP {A1 0,A2 -90,A3 90,A4 0,A5 0,A6 0}

FOR J=1 TO 5
   PTP {A1 4}
   PTP {A2 -7,A3 5}
   PTP {A1 0,A2 -9,A3 9}
ENDFOR

PTP {A1 0,A2 -90,A3 90,A4 0,A5 0,A6 0}
END
```

Figura 3.8: Ejemplo de programa de KSS

3.2.1. Programación con movimientos PTP, LIN y CIRC

El software KSS cuenta con varios tipos de movimientos,los principales que pueden ser programados son:

- Movimiento punto a punto (PTP).

- Movimiento lineal (LIN).

- Movimiento circular (CIRC).

3.2.1.1. Movimiento PTP

El robot se mueve del punto inicial al final a lo largo de la ruta mas rápida, lo cual, generalmente, no es la más corta, en la figura 3.9 se muestra un ejemplo. El robot guía el TCP a lo largo de la ruta más rápida al punto final. El camino más rápido generalmente no es el camino más corto y, por lo tanto, no es una línea recta. Como los movimientos de los ejes del robot son rotativos, las trayectorias curvas se pueden ejecutar más rápido que las trayectorias rectas[22] [24].

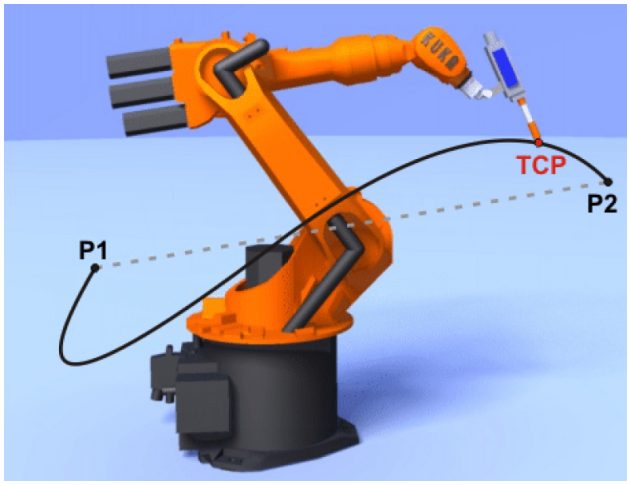


Figura 3.9: Movimiento PTP

El punto final puede especificarse en coordenadas cartesianas o específicas del eje. Las coordenadas cartesianas se refieren al sistema de coordenadas BASE. Si no se especifican todos los componentes del punto final, el controlador toma los valores de la posición anterior para los componentes faltantes. En la figura 3.10 se muestra tres ejemplos. En el primero de ellos se especifica el punto en coordenadas cartesianas, en el siguiente, el punto se muestra en coordenadas de ejes específicos y, finalmente, en el ultimo se especifican solo dos componentes, para el resto el controlador asigna los valores de la posición previa[24] [22].

```
PTP {X 12.3,Y 100.0,Z 50,A 9.2,B 50,C 0,S 'B010',T 'B1010'}
PTP {A1 10,A2 -80.6,A3 -50,A4 0,A5 14.2, A6 0} C_DIS
PTP {Z 500,X 123.6}
```

Figura 3.10: Ejemplos de sintaxis para movimientos PTP

3.2.1.2. Movimiento LIN

El robot guía el TCP a una velocidad definida a lo largo de una ruta recta al punto final, la figura 3.11 muestra un movimiento LIN.

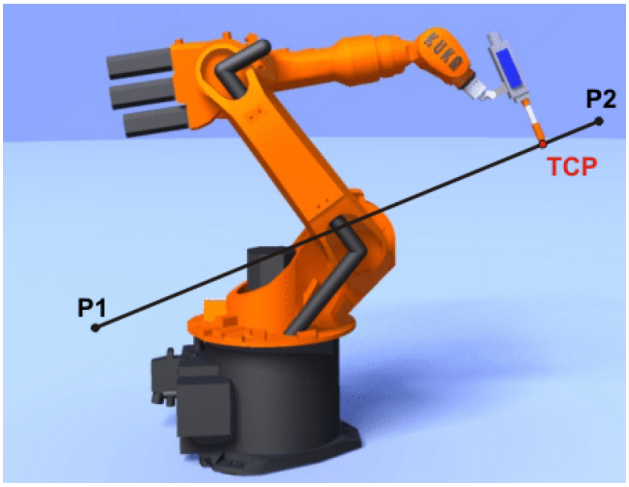


Figura 3.11: Movimiento LIN

Si no se especifican todos los componentes del punto final, el controlador toma los valores de la posición anterior para los componentes faltantes. Las coordenadas se refieren al sistema de coordenadas BASE. En la figura 3.12 se muestra un ejemplo de la sintaxis para un movimiento LIN.

```
LIN {Z 500,X 123.6}
```

Figura 3.12: Ejemplos de sintaxis para movimiento LIN

3.2.1.3. Movimiento CIRC

El robot guía el TCP a una velocidad definida a lo largo de una ruta circular hacia el punto final. La ruta circular está definida por un punto de inicio, punto auxiliar y punto final, en la figura 3.13 se muestra el recorrido a través de estos tres puntos.

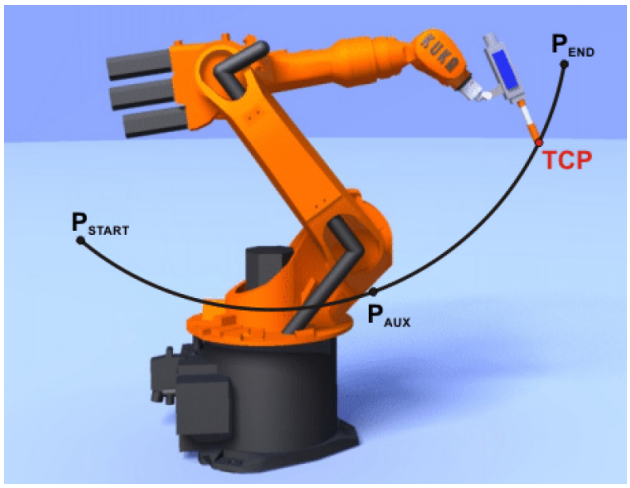


Figura 3.13: Movimiento CIRC

Si no se especifican todos los componentes del punto final, el controlador toma los valores de la posición anterior para los componentes faltantes. El movimiento circular consta de un punto intermedio y un punto final en coordenadas BASE; este ultimo punto es definido por un angulo de 260°. En la figura 3.14 se muestra un ejemplo de la sintaxis de esté movimiento[22].

```
CIRC {X 5,Y 0, Z 9.2},{X 12.3,Y 0,Z -5.3,A 9.2,B -5,C 20}, CA 260
C_ORI
```

Figura 3.14: Ejemplos de sintaxis para movimiento CIRC

3.2.2. Programación con sentencias de control

3.2.2.1. Sentencia de control for

El bloque de instrucciones se repite hasta que el contador supera o cae por debajo de un valor definido. Después de la última ejecución del bloque de instrucciones, el programa se reanuda con la primera instrucción después de ENDFOR. En la figura 3.15 se da un ejemplo de la sintaxis para las sentencias de control, donde B se incrementa en uno en cada ciclo llegando al valor máximo de 5. El conteo inicia con un valor de A=1 hasta 10 con incrementos de 2 [22][24].

```
INT A
...
FOR A=1 TO 10 STEP 2
   B=B+1
ENDFOR
```

Figura 3.15: Ejemplo de sintaxis para sentencia de control For

3.2.2.2. Sentencia de control if

Si la condición es valida, se ejecuta el primer bloque de instrucciones (THEN) y el segundo si es invalida (ELSE). El bloque ELSE puede ser omitido y por tanto si la condición no se cumple, el programa continúa con la instrucción posterior al ENDIF. En la figura 3.16 se muestra un ejemplo de la sentencia if, donde si la entrada 1 es verdadera activamos la salida 17, sino la desactivamos[22] [24].

```
IF $IN[1]==TRUE THEN
   $OUT[17]=TRUE
ELSE
   $OUT[17]=FALSE
ENDIF
```

Figura 3.16: Ejemplo de sintaxis para sentencia de control if

3.2.2.3. Sentencia de control loop

Es un bucle que repite sin cesar un bloque de instrucciones. La ejecución del mismo se puede terminar con EXIT. En el caso de bucles anidados, el externo se ejecuta primero y dentro de esté se desarrolla el interno. En la figura 3.17 se muestra la sintaxis para un loop, donde se ejecutan dos movimientos LIN y una condicional if que pregunta si la entrada 30 esta activa, si está activa salimos del bucle y sino se continua ejecutando los dos movimientos[22] [24].

```
LOOP
   LIN P_1
   LIN P_2
   IF $IN[30]==TRUE THEN
      EXIT
   ENDIF
ENDLOOP
```

Figura 3.17: Ejemplo de sintaxis para sentencia de control loop.

3.2.2.4. Sentencia de control while

La sentencia se repite siempre que la condición sea valida, sin embargo antes de cada bucle se verifica que la condición sea verdadera, sino se cumple el bloque de instrucciones no se ejecuta. Si la condición es falsa, la ejecución del programa se reanuda después de la línea ENDWHILE. En la figura 3.18 se muestra la sintaxis para sentencia while, donde se incrementa w mientras que está sea menor que 100 [22] [24].

```
W=1
WHILE W<100
   W=W+1
ENDWHILE
```

Figura 3.18: Ejemplo de sintaxis para sentencia de control while.

3.2.2.5. Instrucciones con entradas y salidas

El software KSS dispone de 1026 entradas y 1024 salidas, si se activan individualmente, se puede hablar de entradas y salidas binarias. Las salidas binarias pueden tomar 2 estados: Low o High (nivel de voltaje bajo o alto). Por consiguiente, las salidas pueden ser activadas con las variables del sistema.

- $ OUT[Nr] = TRUE
- $ OUT[Nr] = FALSE

El estado de una entrada $ IN[Nr] puede leerse dentro de una variable booleana o bien utilizarse como expresión booleana en las instrucciones de programa, interrupción o de disparo. En la figura 3.19 se muestra un ejemplo donde si la entrada 1 es verdadera, se activa la salida 17, de lo contrario se desactiva [22] [24] [25] [26].

```
IF $IN[1]==TRUE THEN
   $OUT[17]=TRUE
ELSE
   $OUT[17]=FALSE
ENDIF
```

Figura 3.19: Sintaxis para entradas y salidas digitales

3.2.2.6. Programación con funciones y subprogramas

Un subprograma o una función son partes de programas separados, con encabezamiento de programa, sección de declaraciones y de instrucciones, que

pueden llamarse desde cualquier parte del programa según convenga. Una vez procesado el subprograma o la función se produce un salto atrás a la instrucción subsiguiente de la llamada del subprograma. Desde un subprograma o bien una función, pueden llamarse otros subprogramas y/o funciones. En la figura 3.20 se muestra un ejemplo de la sintaxis para funciones y subprogramas, donde se tiene dos funciones $CALC\_1$ y $CALC\_2$, que transfieren un parámetro cuando son llamadas [22] [24].

```
 1  DEF MY_PROG( )
 2  DECL REAL r,s
 3  ...
 4  CALC_1(r)
 5  ...
 6  CALC_2(s)
 7  ...
 8  END
____________________________________
 9  DEF CALC_1(num1:IN)
10  DECL REAL num1
11  ...
12  END
____________________________________
13  DEF CALC_2(num2:OUT)
14  DECL REAL num2
15  ...
16  END
```

Figura 3.20: Ejemplo de sintaxis de funciones y subprogramas

Capítulo 4

Diseño e integración de sub-sistemas para automatizar el proceso de manufactura

En esté capitulo se dará un acercamiento al manejo de software y hardware que se integro para dar solución a la automatización del proceso de manufactura. Se inicia con el diseño de la pieza y configuración de maquinado en Mastercam, para posteriormente realizar los últimos ajustes en Robotmaster. Ya implementados estos procesos, se procede a realizar el desarrollo del programa principal en el software KSS KUKA. Completadas estas dos etapas, se aborda el diseño e implementación del sistema de visión artificial, iniciando con una introducción tanto al sistema embebido como al software a utilizar y finalmente un breve acercamiento al procesamiento digital de imágenes. Por ultimo se tiene el diseño y funcionamiento del programa de visión artificial, mostrando las tres etapas que lo conforman.

4.1. Requerimientos del sistema.

En la figura 4.1 se presenta el diagrama del sistema para automatizar el proceso de manufactura con visión artificial, donde se muestra cada uno de los elementos que integran al mismo. Y a forma de lista se da una breve descripción de cada uno, indicando su función para automatización el proceso de manufactura.

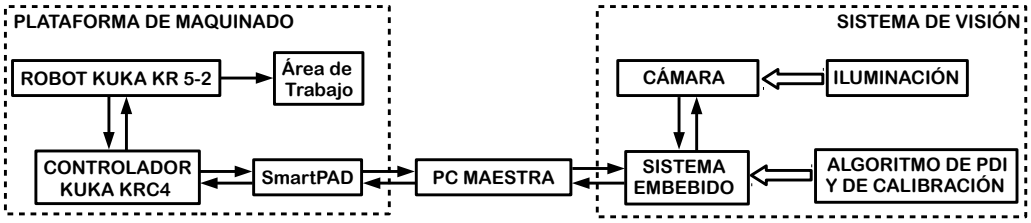


Figura 4.1: Diagrama general del sistema

- **PC maestra:** es el elemento central del sistema, realizará la toma de decisiones y ejecutará cada uno de los subprocesos, estableciendo una conexión remota con el sistema de visión y la plataforma de maquinado.
- **KUKA KR 5-2:** Robot industrial de 6 grados de libertad de tipo poliarticulado, encargado de realizar el maquinado y donde será implementado el sistema de visión.
- **KUKA KRC4:** controlador para el manipulador, y la interfaz entre esté y el SmartPAD.
- **SmartPAD:** tiene todas las funciones de control y visualización requeridas para operar y programar el Robot industrial.
- **Sistema de visión:** deberá estar conformado por los siguientes subsistemas.
  - **Iluminación:** conjunto de elementos que irradian sobre el área de trabajo.
  - **Cámara:** cámara web, a través de la cual se captura las imágenes para la etapa de procesamiento.
  - **Sistema embebido:** flexible tanto en software como en hardware. Estará integrado, por los siguientes elementos.
    - **Software:** es aquí donde se desarrollara en un único programa, el funcionamiento del sistema de visión.
    - **Procesamiento:** partiendo de las imágenes digitales obtenidas, se implementará el algoritmo (PDI) para obtener el punto de inicio de maquinado.
    - **Periféricos:** Gestiona las entradas y salidas del sistema para sincronizarlas sobre dispositivos externos.

- **Comunicaciones:** conexión remota entre el Robot y el sistema embebido.

La automatización del proceso de manufactura a través del sistema de visión embebido, deberá cumplir con los siguientes requerimientos de diseño.

- Diseño y configuración en software CAD.
- Configuración de manufactura en software CAM.
- Sistema visión flexible tanto hardware como en software.
- Conexión remota entre el sistema de visión artificial y el robot.
- Obtener el centroide del stock en cualquier región de la mesa.
- El error de aproximación al centride debe ser cercano a 0.04 mm.
- El sistema de visión deberá ser portable.

4.2. Diseño con software CAD/CAM

Como primer paso iniciaremos con diseño de la pieza y configuración de parámetros de maquinado en Mastercam, para posteriormente realizar la configuración de manufactura del Robot y la herramienta (spindle) en Robotmaster.

4.2.1. Diseño y configuración en Mastercam

Es en esta etapa donde el proceso de manufactura inicia, comenzando con el diseño de la pieza, que sera un rectángulo con dimensiones de 300 mm por 400 mm, y al centro del mismo un circulo con radio de 50 mm, esto para tener una mejor percepción del punto de inicio de maquinado. Se elige un rectángulo con estas medidas, con la finalidad de dividir el área de trabajo en 6 regiones y al elegir estas dimensiones es posible colocar el rectángulo en cualquier región, y así cumplir con uno de los requerimientos del sistema. En la siguiente figura se muestra la interfaz de Mastercam, con el diseño de la pieza y algunos parámetros de configuración para el maquinado de la misma.

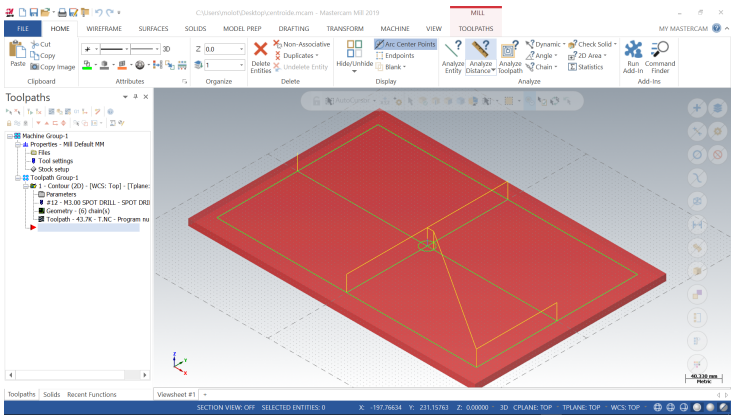


Figura 4.2: Diseño y configuración en Mastercam

4.2.2. Configuración de manufactura en Robotmaster

Es aquí donde configuramos los últimos parámetros de manufactura, comenzando por elegir tanto el tipo de Robot como el spindle a utilizar, para lo cual se selecciona un KUKA KR5 ARC HW y un DREMEL 3000 respectivamente, otra configuración que se debe realizar es el home de inicio y final, eligiendo la siguiente posición $[A1 = 0, A2 = -90, A3 = 0, A4 = 0, A5 = 0, A6 = 0]$, donde el numero seguido de la letra representa el numero de eje del Robot.

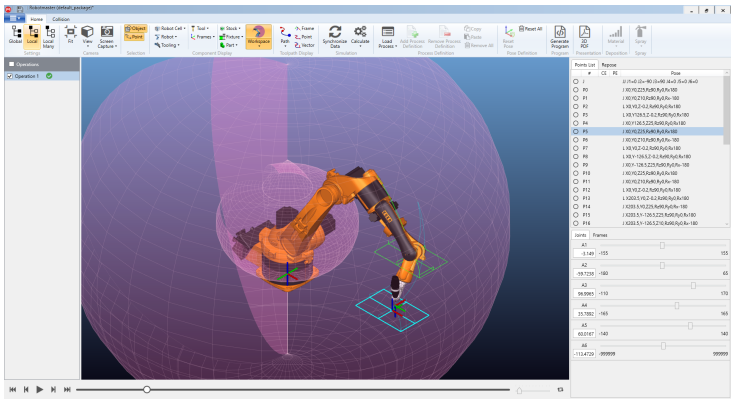


Figura 4.3: Ventana de simulación de Robotmaster

Para terminar este proceso debemos hacer los últimos ajustes respecto a la herramienta, que logren evitar una colisión o elegir la posición mas optima de corte para el Robot, respecto al stock. Se inicia por seleccionar interpolación esférica, que la rotación de la herramienta sea con respecto a Z y que la

orientación sea por la parte superior. Estos parámetros se deben configurar de tal forma que el manipulador logre realizar la manufactura de la pieza en cualquier punto de la mesa y con esto cumplir los requerimientos del sistema. En la figura 4.3 se muestra la simulación del maquinado del stock.

Mediante la interfaz de simulación, es posible verificar cada uno de los pasos del procesos de manufactura de la pieza, si se tiene una respuesta favorable, abrimos el pos-procesador de Robotmaster, para generar el código a cargar en el SmartPAD del Robot y con esto iniciar el siguiente proceso de manufactura.

4.3. Diseño del programa en KUKA System Software

Ya hecho los dos procesos anteriores pasamos al desarrollo del programa principal y el sub-programa en el software KSS de KUKA, que realizaran tanto la toma de decisiones como el procesos de manufactura respectivamente.

4.3.1. Programa principal y sub-programa

El programa principal esta conformado por tres secciones, la primera consta de activar al Robot para realizar la captura del área de trabajo, esto se logro estableciendo una posición predeterminada que realice el procesos siempre desde el mismo punto, y mediante un movimiento punto a punto (PTP) se estableció esta rutina. Para generar los valores del movimiento PTP se comenzó por colocar la cámara en el Robot y realizar diferentes pruebas, hasta conseguir tener una posición donde se tomara el área de trabajo completa y con la menor distorsión de la imagen, dado las prestaciones que se tiene de la cámara.

La sección intermedia consta de tres acciones, la principal es el llamado al sub-programa con el código generado en Robotmaster y las dos restantes son el encendido y apagado del spindle a través de una salida del controlador KUKA KRC4, ya que el mantenerlo activo indefinidamente o hacerlo de forma manual volvería a este proceso ineficiente y no se cumpliría con los requerimientos del sistema. Para llevar a acabo esta acción se comenzó por identificar los elementos del controlador y su funcionamiento, en la figura 4.4 se muestra de forma numera cada uno de ellos.

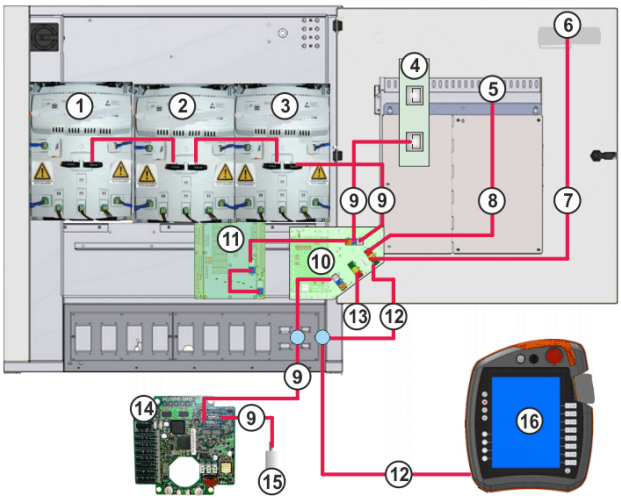
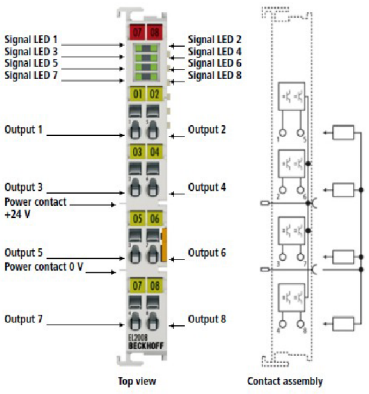


Figura 4.4: Elementos del controlador KRC4

El contralor esta conformado por 16 elementos, incluyendo el smarPAD. El modulo con el numero 13 hace referencia a KUKA Extension Bus (KEB), que permite establecer diferentes configuraciones de operación para el mismo y dando la opción elegir la conexión EtherCAT (protocolo de comunicación industrial) para dispositivos, que establece un protocolo de comunicación entre el controlador y sus terminales, a través del conector EK1100. Dicho elemento consta de un acoplador que se muestra en la figura 4.6 y cualquier numero de terminales, el KRC4 como ya se menciono en el capitulo anterior tiene la posibilidad de manipular un extenso numero de entradas y salidas digitales, en la figura 4.5 se presenta el modulo con el que cuenta el controlador, este contiene 8 salidas digitales.



Technical data	EL2008 | ES2008
Connection technology	1-wire
Number of outputs	8
Nominal voltage	24 V DC (-15 %/+20 %)
Rated load voltage	24 V DC (-15 %/+20 %)
Load type	ohmic, inductive, lamp load
Distributed clocks	–
Max. output current	0.5 A (short-circuit proof) per channel
Short circuit current	typ. < 2 A
Reverse voltage protection	yes
Breaking energy	< 150 mJ/channel
Switching times	typ. Ton: 60 µs, typ. Toff: 300 µs
Current consumption E-bus	typ. 110 mA
Electrical isolation	500 V (E-bus/field potential)
Current consumption power contacts	typ. 15 mA + load
Bit width in the process image	8 outputs
Configuration	no address or configuration setting
Weight	approx. 55 g
Operating/storage temperature	-25…+60 °C/-40…+85 °C
Relative humidity	95 %, no condensation
Vibration/shock resistance	conforms to EN 60068-2-6/EN 60068-2-27
EMC immunity/emission	conforms to EN 61000-6-2/EN 61000-6-4
Protect. class/installation pos.	IP 20/see documentation
Pluggable wiring	for all ESxxxx terminals
Approvals	CE, UL, Ex, IECEx

Figura 4.5: Terminal de salidas digitales y especificaciones

Figura 4.6: Conector EK1100

Para activar el Dremel 3000 se elige la salida numero 1 en el modulo, y por software se le asigna automáticamente el mismo numero. Como ya se menciono anteriormente la principal acción de este bloque es el llamado al subprograma que contiene el código para realizar la manufactura, en la figura 4.7 se muestra una captura de pantalla del smartPAD con las primeras lineas de código generado. La tercera y ultima acción del programa principal es permitir salir del proceso de manufactura, dando la opción de cargar un nuevo código de manufactura o simplemente salir del mismo si así se desea. Completado el diseño con software CAD/CAM y la programación en Software KSS de KUKA, se puede dar paso al desarrollo del sistema de visión artificial.

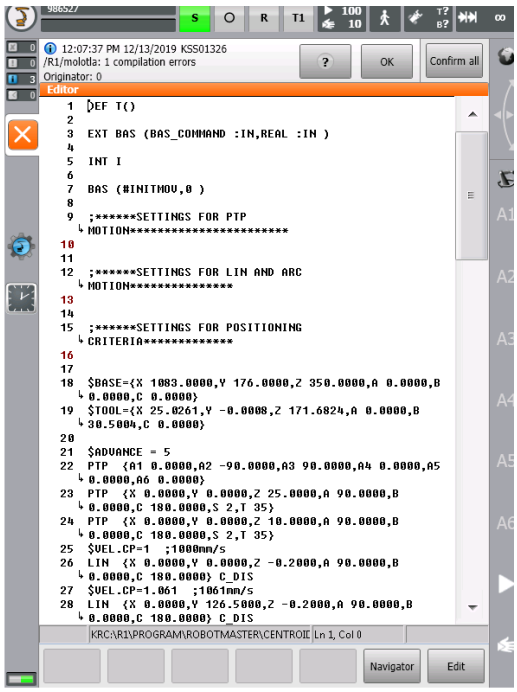


Figura 4.7: Screnshot de subprograma de manufactura desde el smarPAD

4.4. Diseño e implementación del sistema de visión artificial

Ya implementadas las anteriores etapas, comenzamos con una introducción a la Rasberry Pi (donde se implementará el sistema de visión artificial), resaltando las principales características de la misma. Y antes de abordar la sección de desarrollo del programa para el sistema de visión, se dará un preámbulo al lenguaje de programación Python y sus diferentes librerías, como kukavarproxy que es la encargada de establecer la conexión remota con el Robot.

4.4.1. Sistema embebido (Raspberry PI)

La Raspberry Pi es una microcomputadora de propósito general, de bajo costo y desarrollada en Inglaterra, que cuenta con flexibilidad tanto en hardware como en software, siendo esta principal característica por lo cual se elige. Ya que es un elemento primordial para obtener un sistema compacto, portable y adaptable a las variante generadas en el desarrollo del mismo. En la actualidad existen varios modelos y en la presente tesis se utilizo el modelo B +, en la figura 4.8 se muestra una imagen con algunas de sus características.

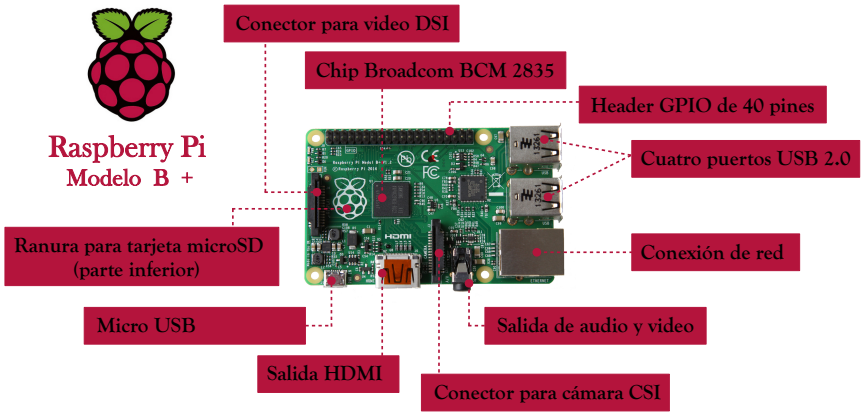


Figura 4.8: Raspberry Pi

El sistema está basado en un chip Broadcom BCM2835, el cual contiene un procesador central (CPU) ARM 1176JZF-S a 700 MHz (de la familia ARM11), un procesador gráfico (GPU) VideoCore IV y una memoria

SDRAM de 512 MB (compartidos con el GPU). Como dispositivo de almacenamiento se utiliza una memoria microSD. El software puede ser descargado desde la página de Raspberry y se tiene la posibilidad elegir la distribución a instalar, Raspbian es la recomendada por la fundación, debido a que es una distribución derivada de Debian que está optimizada para la Raspberry Pi.

4.4.1.1. Lenguaje de programación Python

Python es un lenguaje de programación con sintaxis clara, multiplataforma, orientado a objetos, gratuito y con el cual ya cuenta la distribución instalada en la Raspberry. Por las características mencionadas y por ser tanto un software flexible como contar con una librería que permite establecer una conexión remota con el Robot KUKA, se elige esté lenguaje para desarrollar, en un único programa (gracias a sus prestaciones) cumpliendo con los requerimientos planteados, el funcionamiento del sistema de visión artificial.

4.4.1.2. Librerías

Una de las característica mas importante de Python es su flexibilidad, ya que nos permite agregar módulos o extensiones para complementar su funcionamiento, lo cual resulta interesante cuando se desea ampliar las funciones de un programa; algunas de estas son bases de datos, arreglos, funciones matemáticas, gráficas, acceso a internet, procesamiento digital de imágenes, fechas, tiempos, etc. Para desarrollar el programa de visión se eligieron algunas de estas, las cuales a se explicarán brevemente a continuación.

kukavarproxy: está librería logra establecer una conexión Ethernet (TCP / IP) con el controlador del robot kuka, capaz de extraer y sobre escribir variables del sistema.

OpenCV: es una librería para visión artificial de código abierto y multiplataforma, que contiene más de 500 funciones que abarcan una gran gama de áreas, como reconocimiento de objetos, calibración de cámaras, visión estéres y visión robótica.

matplotlib: está es la librería que permite generar gráficas, gráficos de barras, histogramas, etc. Su forma de graficar y de programar se asemeja a lo hecho con matlab, tanto visualmente como en su exactitud. Matplotlib tiene una gran variedad de complementos, entre los que destaca la opción de graficar en 3D.

numpy : este paquete es una extensión de Python, el cual contiene una librería con funciones matemáticas de alto nivel, que tiene un arreglo de

n-dimensiones, funciones sofisticadas, la capacidad de entregar números aleatorios y hasta transformada de fourier entre otras cosas.

time: este módulo permite acceder a varias funciones relacionadas con el tiempo, como retardos, tiempos de espera, fecha, hora, etc.

4.4.1.3. KUKAVARPROXY

Es un servidor de múltiples clientes para robots KUKA, que implementa una interfaz de comunicaciones cruzadas, la cual logra la interacción con el proceso de control en tiempo real del robot y permite realizar varias operaciones, como la selección o cancelación de un programa específico, la detección de errores, el cambio de nombre de los archivos de programas, el guardado de programas, la lectura y escritura de variables.

Para establecer una conexión remota con KUKAVARPROXY entre el controlador y un pc remota, como primer paso se debe ejecutar en el sistema operativo del KRC4, ya realizado esto se debe configurar la conexión de red en el HMI de KUKA y finalmente en la pc remota (Raspberry Pi), a través de la librería KUKAVARPROXY en python que se ejecuta como un cliente conectado remotamente con el controlador a través de una comunicación TCP/IP, que permite leer y escribir variables, lo cual abre una variedad de posibles aplicaciones.

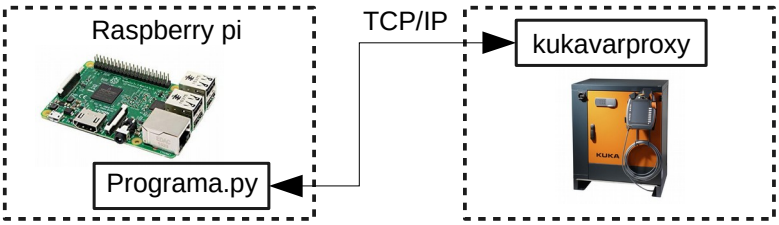


Figura 4.9: Conexión remota KUKAVARPROXY

4.4.2. Procesamiento digital de imágenes

El control de calidad en muchas ocasiones se vale de la ayuda de instrumentos, pero aún hoy en día muchos procesos de inspección se ejecutan basados totalmente en la experiencia de un inspector y su agudeza visual para reconocer defectos en los productos. La migración hacia sistemas de visión automatizados permite ejecutar mediciones precisas, de mayor complejidad, velocidad y cantidad de la que un humano puede realizar.

Cada aplicación de control de calidad por visión puede ser diferente y requerir instrumentos y algoritmos muy particulares, inclusive pensando en la inspección de un mismo producto. Por ejemplo, en el caso de la verificación dimensional y geométrica de piezas mecanizadas es necesario considerar que estas piezas presentan brillos intensos y alta reflexión de la luz. Como la medición está relacionada con los contornos de las mismas, es necesario evitar degradarlos durante el pre-procesamiento de las imágenes, de hecho se deben mejorar [15].

Otro de los aspectos importantes es la calibración del sistema, paso crucial que tiene como objetivo principal determinar la resolución del sistema y evitar errores significativos en la medición. A continuación se presentan los aspectos más relevantes e información importante sobre los algoritmos usados en cada una de las etapas del procesamiento de la imagen[14].

4.4.2.1. Captura del área de trabajo

El proceso de captura está centrado en tres aspectos importantes, necesarios para obtener una imagen de calidad que no altere las condiciones reales, agregando errores considerables al sistema. El primero de ellos es la iluminación, la cual debe ser homogénea sin llegar a saturar o crear brillos que puedan ocultar defectos en los contornos de las piezas[20].

Otro aspecto importante son los sistemas ópticos, ya que sufren de un número inevitable de distorsiones geométricas que alteran las proporciones reales de las dimensiones de los objetos en la imagen. Aunque existen múltiples algoritmos para obtener los parámetros de distorsión del lente, es importante para aplicaciones de medición contar con una lente de baja distorsión con el fin de reducir el tiempo de procesamiento y aumentar la exactitud del sistema.

Finalmente, la cámara en sí es el elemento más importante en el proceso de captura y debe ser seleccionada de acuerdo con la aplicación, la capacidad para manipular los parámetros internos de captura y las características de resolución, velocidad de captura, envío de datos al procesador y, finalmente, el acondicionamiento de la misma para trabajar en ambientes industriales en el caso de sistemas que estén diseñados para realizar inspecciones en la planta de producción[14].

4.4.2.2. Pre-procesamiento de la imagen

El pre-procesamiento consiste en la aplicación de técnicas que permitan el realce o mejoramiento de algunas características importantes en las imágenes originales para facilitar el proceso de segmentación. En un sistema

de medición y de inspección es necesario aplicar técnicas que conserven los contornos pero que ejecuten labores de suavizado porque en el caso específico de los materiales mecanizados se presentan surcos debido al paso de la herramienta.

El problema de filtros que preserven bordes ha sido enfocado por varios trabajos que proponen una combinación de filtrado de intervalo y de dominio, denominado filtrado bilateral, el cual reemplaza el valor del píxel en x por un promedio de los valores de píxeles vecinos, pero, a diferencia de otros filtros, los valores para el cálculo del promedio deben ser similares. En regiones suaves, los valores de los píxeles en un vecindario pequeño son todos muy similares, por lo que el filtro bilateral actúa como un filtro de dominio estándar, promediando y eliminando los valores débilmente correlacionados causados por el ruido. Por el contrario, si el filtro se ubica en un borde o frontera, solamente se tienen en cuenta aquellos valores más parecidos al valor del píxel evaluado según su posición en la frontera[16][15].

4.4.2.3. Segmentación de la imagen

La técnica de segmentación más conocida dentro del procesamiento de imágenes es quizás la umbralización. Este método permite separar dos o más regiones de una imagen a partir de un análisis del histograma. Aunque es un método sencillo, su aplicabilidad en procesos industriales ejecutados por visión artificial ha sido y sigue siendo de gran importancia. Por otro lado, un análisis orientado a bordes permite extraer características más puntuales en cuanto a dimensiones, forma y orientación de los objetos inspeccionados.

Después de obtener los bordes de las regiones de interés, es necesario seguir segmentando la imagen con el fin de obtener unas primitivas de forma para la evaluación de algoritmos de medición e inspección. Uno de los aspectos más interesantes de un contorno son las esquinas, puntos críticos o puntos de alta curvatura. En la literatura se encuentra una gran cantidad de trabajos orientados a la detección de estos puntos.

4.4.3. Diseño y funcionamiento del programa de visión artificial

El programa esta conformado por tres funciones principales, la primera de ellas consiste en capturar el área de trabajo, la segunda acción a realizar es el procesamiento digital de la foto y finalmente la ultima es iniciar el proceso de manufactura. Una opción aunada a estas funciones es terminar dicho proceso. En la figura 4.10 se muestra el diagrama de flujo simplificado del programa desarrollado.

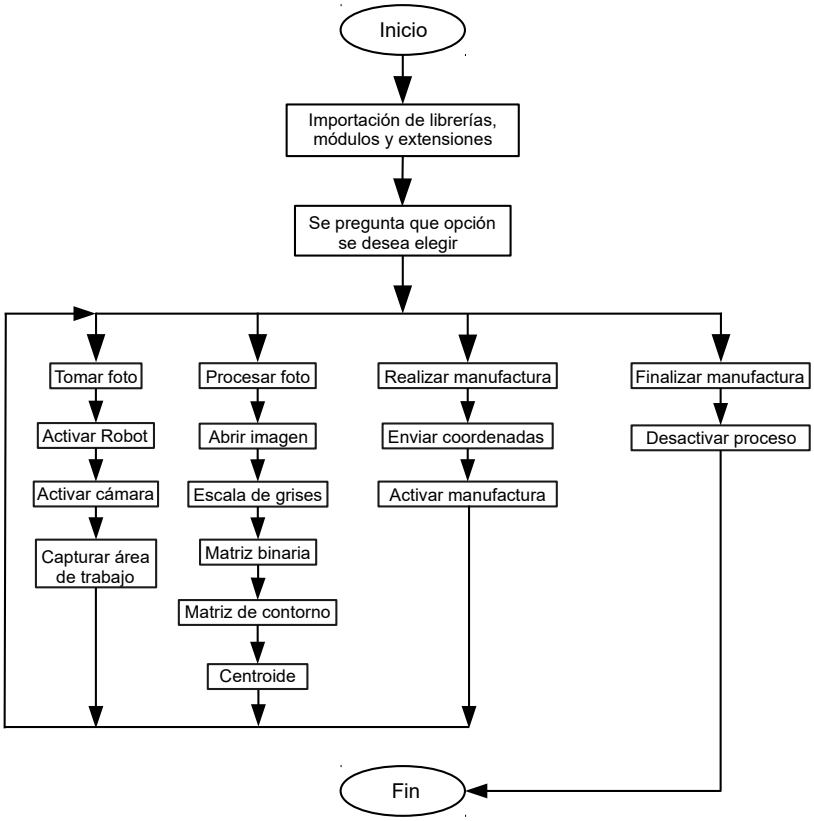


Figura 4.10: Diagrama simplificado del programa para el sistema de visión

Como ya se menciono se tiene tres acciones, la primera de ellas consiste en posicionar al robot para capturar el área de trabajo, posterior a esto acceder a la cámara y finalmente tomar una foto de la mesa donde esta situado el stock, la segunda acción a realizar es el procesamiento digital de la foto, para adquirir el centroide del material en el espacio de trabajo y finalmente enviar las coordenadas de la pieza e iniciar el proceso de manufactura. Se cuenta con cuatro opciones, tomar foto, procesar foto, realizar manufactura y terminar proceso, cada una estas se puede elegir independiente de las otras.

4.4.3.1. Etapa de captura

La función principal de la etapa de captura, es adquirir la imagen del área de trabajo. Para lograr esto, a través de la librería de KUKAVarproxy se establece la conexión remota entre el programa de visión y el Robot, posterior a esto las variables pertinentes son modificadas y enviadas al programa principal en el smartPAD, lo cual activara el subproceso que envía al manipulador

a la posición de tomar foto. Ya que el Robot termina esta rutina, se procede a realizar la captura de la imagen con la ayuda de la librería de openCV. Antes de activar o acceder a la cámara, se configuran algunos parámetros como el formato de la foto, las dimensiones de la misma, el contraste y la ruta donde se desea guardar, para posteriormente acceder a ella. Estableciendo estos parámetros se procede a activar la cámara y realizar la captura del área de trabajo, en la figura 4.11 se muestra el proceso.

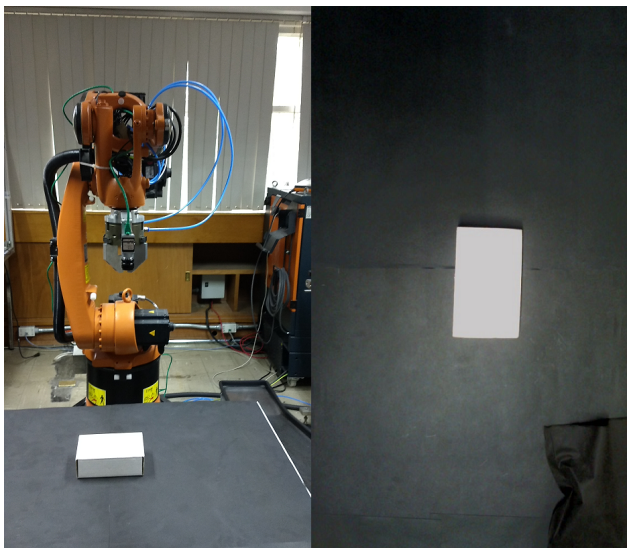


Figura 4.11: Etapa de captura

Se elige una resolución de 1024 x 768 pixeles, dado que estas dimensiones son cercanas a los valores del área de trabajo, lo que no solo permite tener una visión completa de la mesa, sino que la relación entre los pixeles y centímetros de la mesa mejorara al obtener el centroide del stock.

4.4.3.2. Etapa de procesamiento

En esta etapa, como primera acción es acceder a la imagen y realizar una copia de la misma, el segundo paso a realizar es aplicarle una transformación a escala de grises, con lo que se convierte la imagen resultante a una matriz con números de 0 a 255, los cuales indican el valor de gris del punto o pixel, entre mas obscuro sea, mas cerca se esta del valor 0 (negro) y si el numero se aproxima a 255, se tiene un gris próximo a blanco.

Como tercer paso de la etapa de procesamiento, se realiza un análisis de histograma, con la finalidad de conocer en escalas de grises la imagen

y con esto transformar la matriz original a una binaria, y así obtener la segmentación entre el stock y el fondo.

Ya realizadas las anteriores etapas, se procede a encontrar la matriz de contorno, para lo cual se crea una matriz de pesos de la ya generada de unos y ceros, ya sobre esta trabaja para obtener el contorno de la pieza, así pasar al proceso final, que es obtener el centroide del material a trabajar y con esto tener los parámetros de inicio de maquinado, que en el siguiente proceso de manufactura serán ocupados.

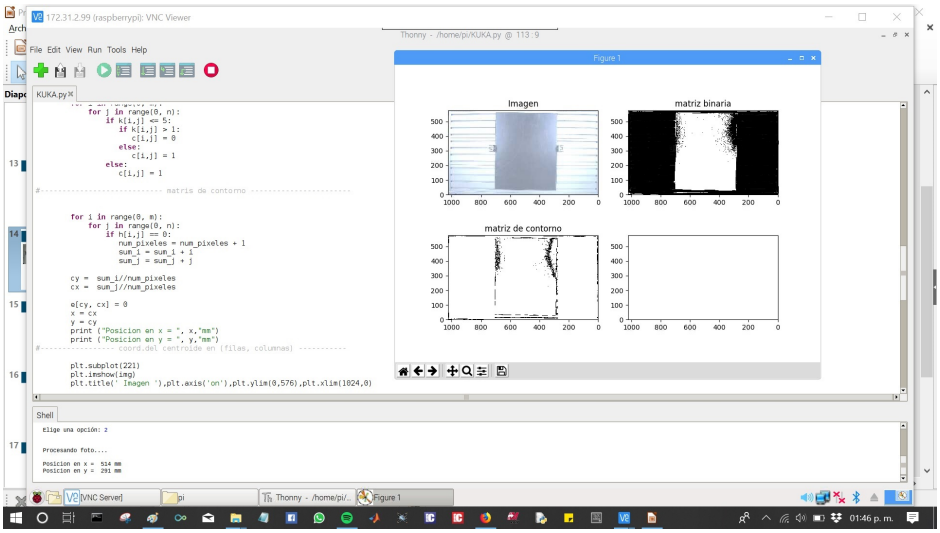


Figura 4.12: Etapa de procesamiento

En la figura 4.12 se muestra una captura de pantalla, con el subplot de cada uno de los procesos que se realizo a modo de ejemplo.

4.4.3.3. Etapa de manufactura

El proceso de manufactura, se inicia enviando las coordenadas del stock al programa principal creado en el smartPAD, ya actualizados estos datos, se activa tanto el spindler y como el sub-programa de manufactura, el cual contiene el código generado en Robotmaster con las trayectorias de corte adecuadas para el robot y el área de trabajo, todo esto en el software KSS de KUKA. Antes de finalizar el código del sub-programa, como ultima acción es regresar al Robot a home, y con esto terminar el proceso de manufactura, para posteriormente regresar al programa principal y desactiva la herramienta, a través de la salida numero 1. En la figura 4.13 da un ejemplo del proceso de manufactura.

Figura 4.13: Etapa de manufactura

Si se desea terminar con el proceso, para cargar un nuevo código generado en Robotmaster o realizar alguna modificación a cualquiera de los sub-procesos, vasta con elegir la opción finalizar manufactura, lo cual activara en el programa principal del smartPAD la rutina que termina con todo el proceso.

Capítulo 5

Experimentos y resultados del sistema

En esté capitulo se describirán los procedimientos realizados en los experimentos para calibrar el sistema. El área de trabajo se dividió en secciones, para identificar en que puntos se tiene una mejor respuesta y en donde decae la aproximación al valor deseado. En cada área se repitieron las pruebas con la finalidad de adquirir un mejor resultado en relación con el error de aproximación, el cual se obtuvo con el promedio del valor esperado y el adquirido en la etapa de procesamiento. Como ya se ha especificado anteriormente, el punto de inicio de manufactura de la pieza se eligió en el centro de la pieza para todas la pruebas. En las siguientes figuras se muestran las fotos de la conexión remota con el sistema, la Raspberry instalada en el robot y la manufactura del logo del IIMAS.

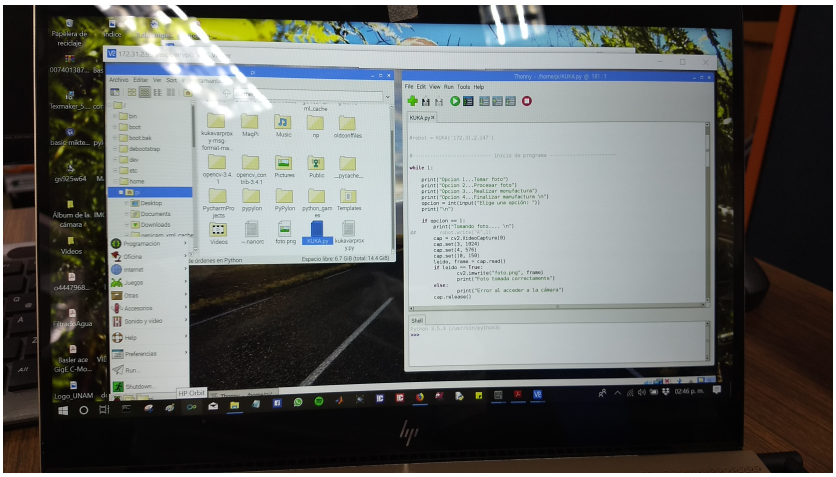


Figura 5.1: Conexión remota con la celda de manufactura

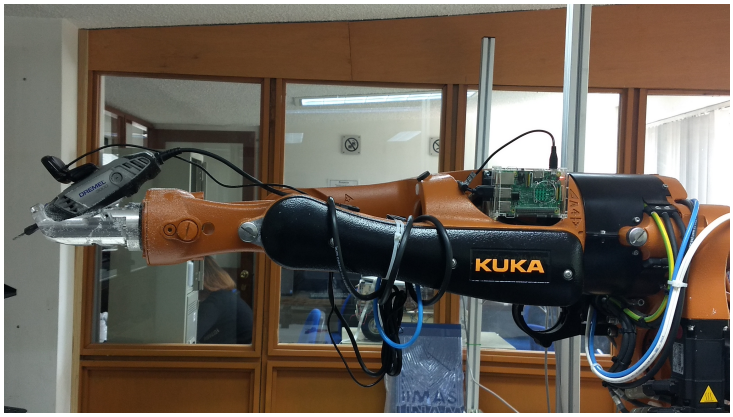


Figura 5.2: y Dremel

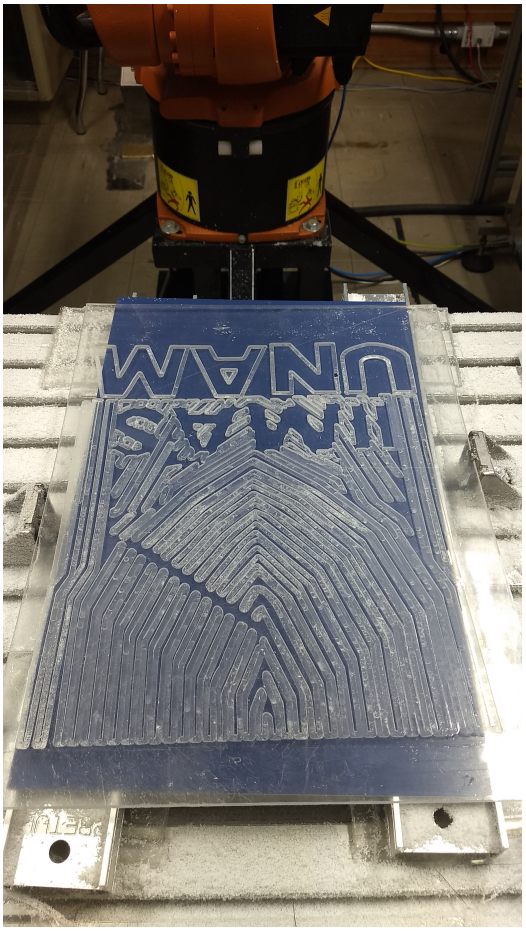


Figura 5.3: Manufactura de logo del IIMAS

5.1. Experimento 1 y resultados

En la figura 5.4 se muestra una foto del área de trabajo, adquirida en la etapa de procesamiento de esté experimento. Para esta primera etapa el área de trabajo se dividió en seis secciones y el stock cuenta con unas dimensiones de 250x400 mm. En los tres experimentos se eligió solo posicionar al robot en el centro de la pieza, sin realizar la manufactura de la misma, ya que se desea comprobar que el sistema responde de manera optima al obtener el centroide del stock y posicionarse en el mismo, lo que nos permitirá abordar la manufactura de forma eficiente.

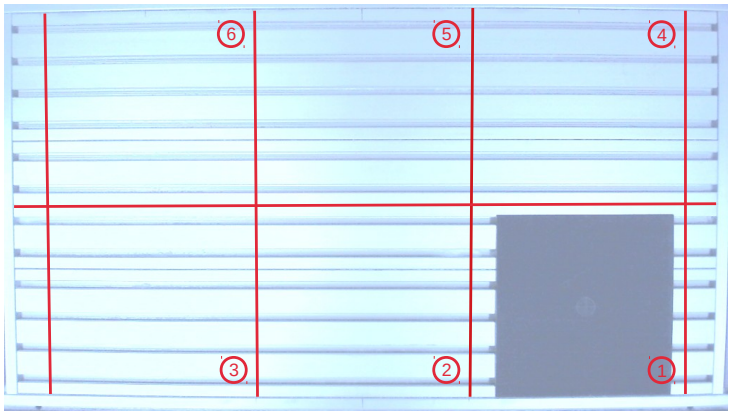


Figura 5.4: Primer experimento

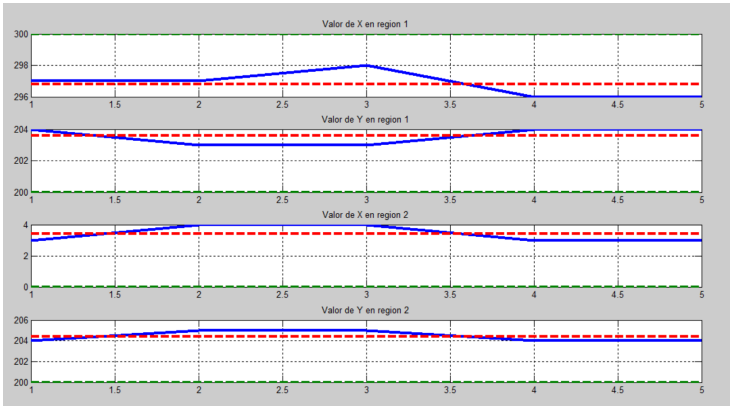


Figura 5.5: Primer experimento región 1 y 2

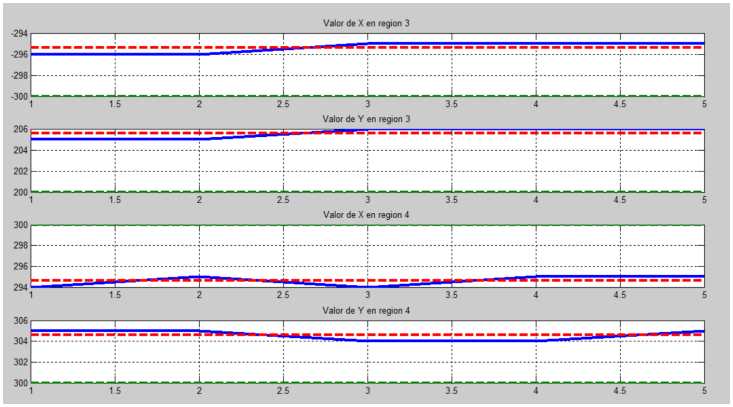


Figura 5.6: Primer experimento región 3 y 4

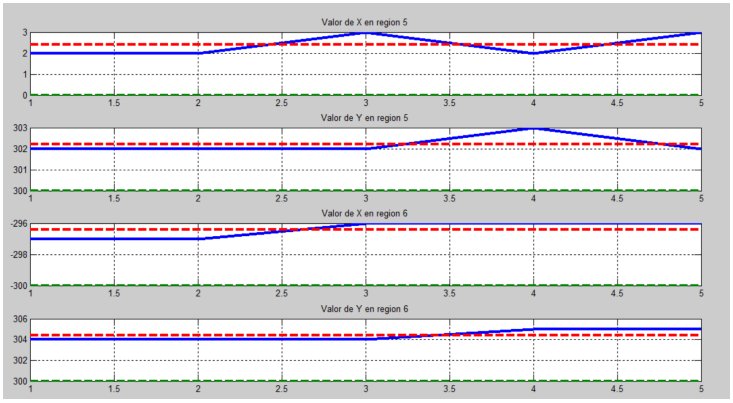


Figura 5.7: Primer experimento región 5 y 6

En la figuras 5.5, 5.6 y 5.7 se muestra a través de gráficas, los resultados que arrojó cada prueba realizada, donde la linea verde representa el valor esperado, la roja el valor promedio y finalmente en color azul se observa las muestras obtenidas. Se observo que al no contar con luz uniforme, se presentan diferentes brillos y en algunas áreas de la mesa estos son más evidentes, lo que repercute en la etapa de procesamiento. Aunado a esto en la etapa de segmentación el resultado no fue el deseado, provocando que el error aumente, ya que al obtener los valores del centroide se generan desviaciones considerables, y en especial en las regiones 3, 6 y 4, esta ultima es donde se muestra un mayor error. En la tabla 5.1 se muestran los resultados para cada región, donde V_e es el valor esperado y e es el error obtenido.

	V_e	e
Región 1	$x = 300, y = 200$	x=3.20,y=3.80
Región 2	$x = 000, y = 200$	x=-2.60,y=-3.00
Región 3	$x = -300, y = 200$	x=-4.20,y=-3.60
Región 4	$x = 300, y = 300$	x=3.80,y=-3.60
Región 5	$x = 000, y = 300$	x=-2.40,y=-2.60
Región 6	$x = -300, y = 300$	x=-3.40,y=-4.40

Cuadro 5.1: Tabla de resultados del experimento 1

5.2. Experimento 2 y resultados

En este segundo experimento, se mejoró la iluminación del área de trabajo, para eliminar la presencia de brillos y obtener una imagen mas uniforme. En la etapa de procesamiento se realizaron modificaciones en el proceso de segmentación ya que en la pruebas anteriores se observó que tras realizar el análisis de umbralización, la separación entre las dos regiones no se realizaba de la forma mas optima. El área se dividió en 9 secciones con la finalidad de identificar con mayor precisión en que puntos el sistema cuenta con una mejor precisión o un mayor error.

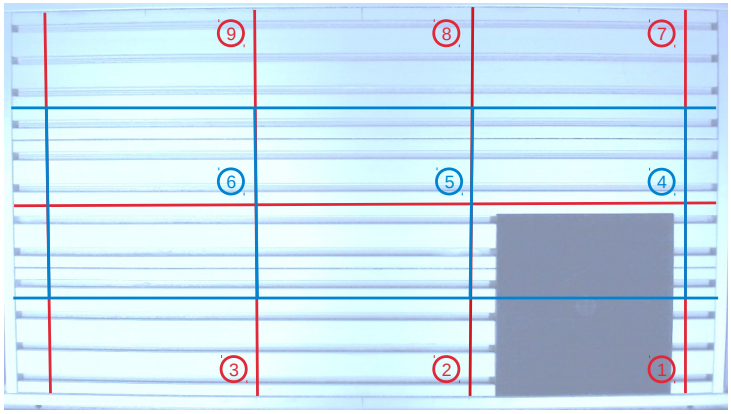


Figura 5.8: Segundo experimento

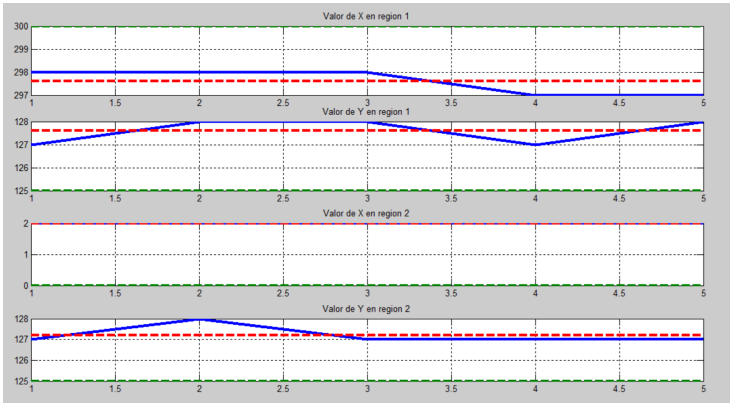


Figura 5.9: Segundo experimento región 1 y 2

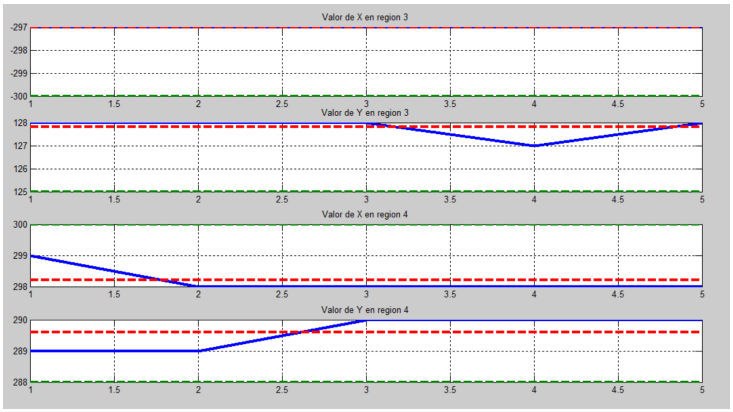


Figura 5.10: Segundo experimento región 3 y 4

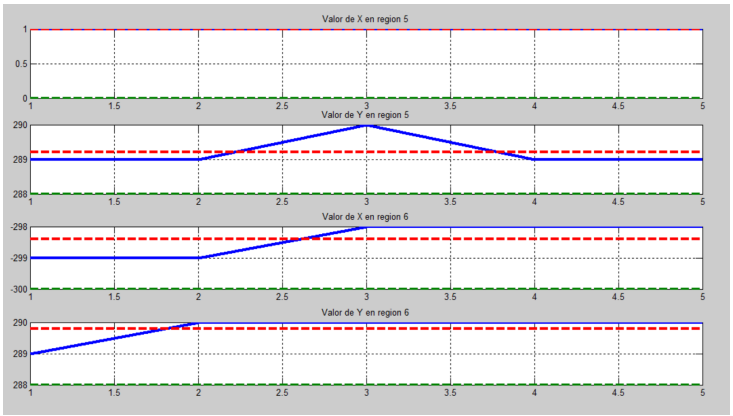


Figura 5.11: Segundo experimento región 5 y 6

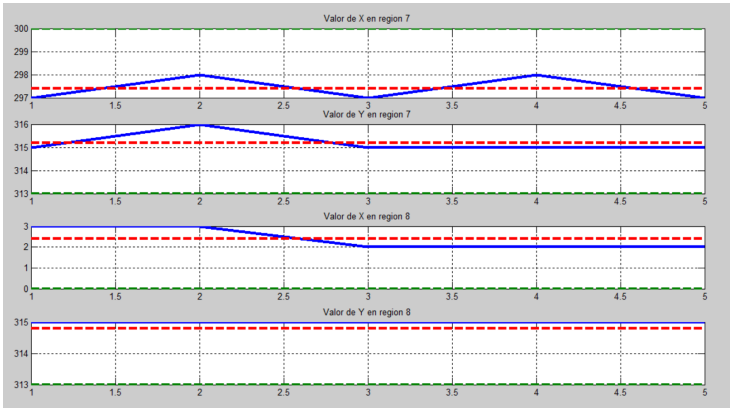


Figura 5.12: Segundo experimento región 7 y 8

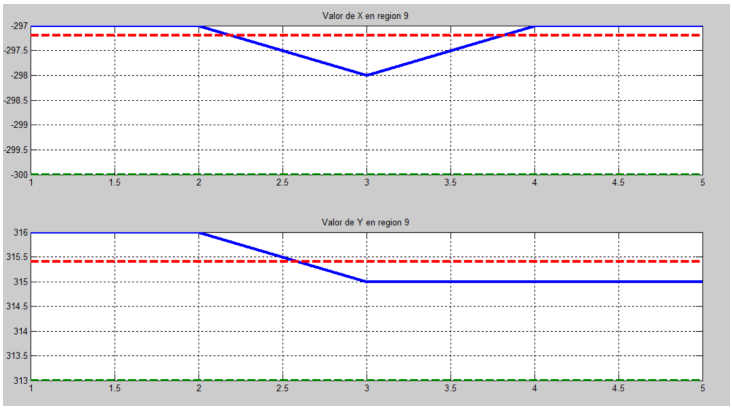


Figura 5.13: Segundo experimento región 9

Para el segundo experimento, los brillos se disminuyeron en comparativa con los reflejados en el primero y se mostró una iluminación mas uniforme en toda el área de trabajo. La modificación en la umbralización mejoró la etapa de segmentación y con esto disminuyó el error en cada una de las regiones, principalmente en las regiones centrales, lo que se refleja en las gráficas mostradas en las figuras 5.9, 5.10, 5.11 y 5.12. El dividir el área trabajo en nueve secciones nos permite saber con mayor exactitud en que regiones tenemos un mayor error. A través de la tabla 5.2 se puede observar la disminución en cada una, pero en algunas de ellas el error sigue siendo considerable. Para el siguiente experimento se deberá seguir modificando la etapa segmentación y la de pre-procesamiento con la finalidad mejorar los bordes.

	$V_{esperado}$	Error
Región 1	$x = 300, y = 125$	x=2.40,y=-2.60
Región 2	$x = 000, y = 125$	x=-2.00,y=-2.20
Región 3	$x = -300, y = 125$	x=-3.00,y=-2.80
Región 4	$x = 300, y = 288$	x=1.80,y=-1.60
Región 5	$x = 000, y = 288$	x=-1.00,y=-1.20
Región 6	$x = -300, y = 288$	x=-1.60,y=-1.80
Región 7	$x = -300, y = 313$	x=2.60,y=-2.20
Región 8	$x = -300, y = 313$	x=-2.40,y=-2.00
Región 9	$x = -300, y = 313$	x=-2.80,y=-2.40

Cuadro 5.2: Tabla de resultados del experimento 2

5.3. Experimento 3 y resultados

En esté ultimo experimento se conservo la división del área de trabajo en las 9 secciones, propuestas en el experimento anterior, ya que como se menciono se logro identificar con mayor precisión en que puntos el sistema cuenta con una mejor aproximación al obtener el centroide del stock. Se observo que en la pruebas anteriores la separación de las dos regiones continua con algunas deficiencias, para mejorar esto se realizaron cambios en las etapas de pre-procesamiento y de segmentación, para la primera etapa se agrego un filtro para bordes, que ayudara a mejorara los contornos, permitiendo que en el segmentado se tengan mejores resultados y finalmente en esté proceso se realizo otro proceso de segmentación al ya realizado sobre las imágenes.

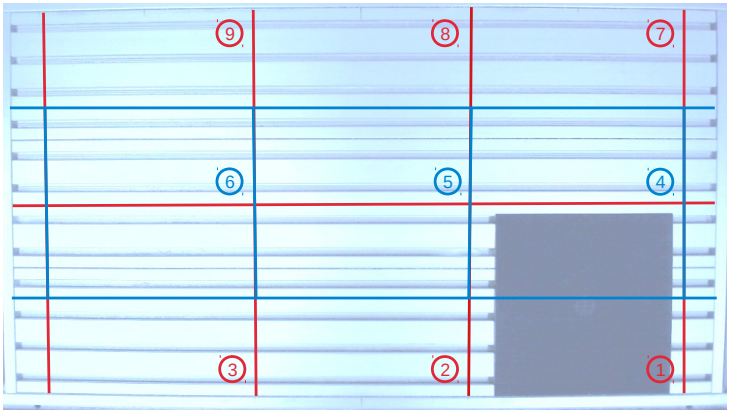


Figura 5.14: Tercer experimento

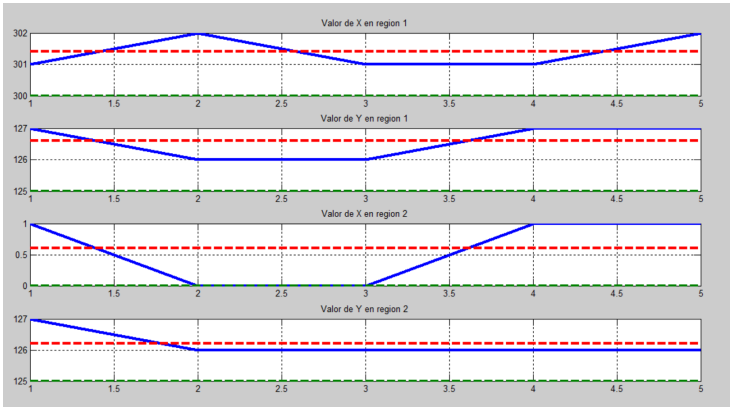


Figura 5.15: Tercer experimento región 1 y 2

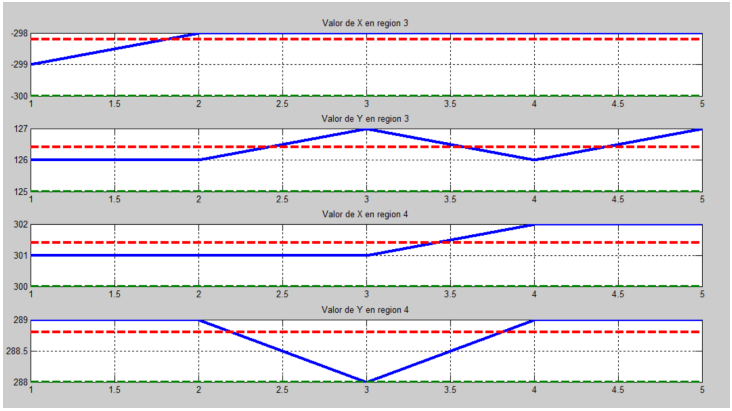


Figura 5.16: Tercer experimento región 3 y 4

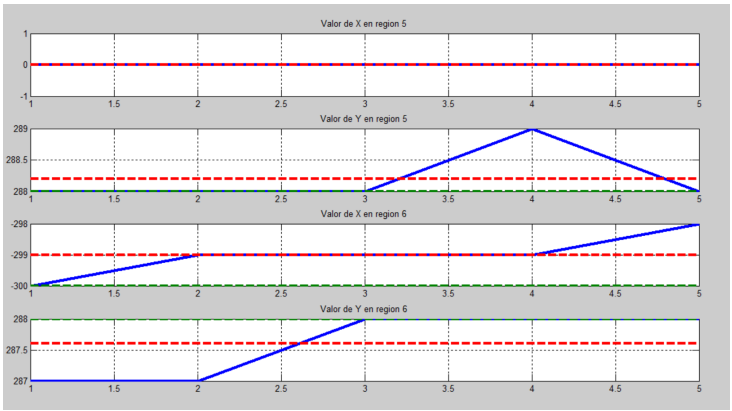


Figura 5.17: Tercer experimento región 5 y 6

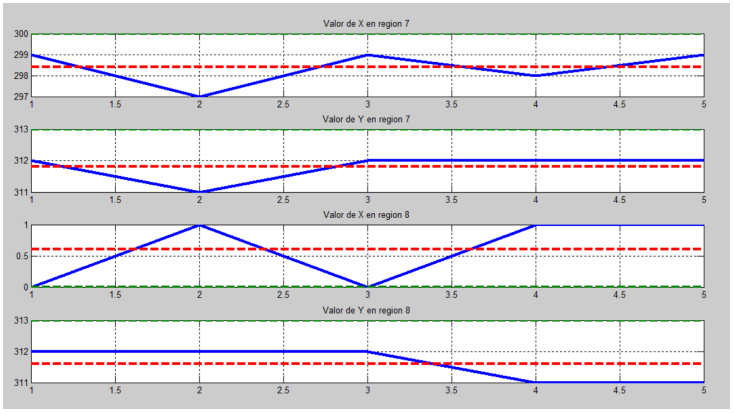


Figura 5.18: Tercer experimento región 7 y 8

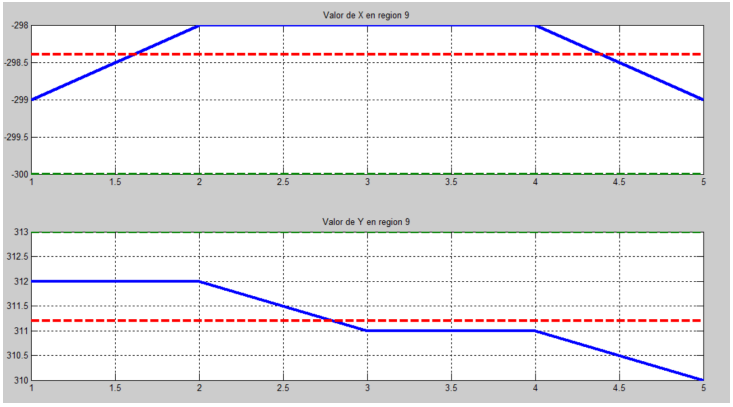


Figura 5.19: Tercer experimento región 9

Como se observa en las figuras 5.15, 5.16, 5.17, 5.18 y 5.19 los cambios realizados ayudaron a tener una mejor respuesta del sistema, logrando que los valores en cada una de las pruebas mejora por mucho su aproximación al valor esperado. En la tabla 5.3 se puede observar que el error disminuyo para cada región, y que para las regiones 2, 4, 5, 6 y 8 se tiene valores por de bajo de un milímetro e incluso en la región 5 se logro obtener en x un valor de 0 milímetros.

	$V_{esperado}$	Error
Región 1	$x = 300, y = 125$	x=1.40,y=-1.60
Región 2	$x = 000, y = 125$	x=-0.60,y=-1.20
Región 3	$x = -300, y = 125$	x=-1.80,y=-1.40
Región 4	$x = 300, y = 288$	x=1.20,y=-0.80
Región 5	$x = 000, y = 288$	x=0.00,y=-0.20
Región 6	$x = -300, y = 288$	x=-1.00,y=-0.40
Región 7	$x = -300, y = 313$	x=1.60,y=-1.20
Región 8	$x = -300, y = 313$	x=-0.60,y=-1.40
Región 9	$x = -300, y = 313$	x=-1.60,y=-1.80

Cuadro 5.3: Tabla de resultados del experimento 3

Capítulo 6

Conclusiones

6.1. Conclusiones

De acuerdo a las observaciones y experimentos realizados se puede concluir que el sistema de automatización para la plataforma de maquinado tiene un desempeño satisfactorio (considerando que es la primera aproximación de un prototipo), logrando realizar la manufactura de materiales blandos de forma automática y remota, y por medio de visión artificial adquirir el punto de inicio de maquinado. Debido a las características de integración y de flexibilidad, tanto de hardware como software, con las que fue diseñado, es posible implementarlo en otros modelos de robots KUKA e incluso seria posible adaptar esté prototipo a otras marcas como ABB, Fanuc, Hass y Yaskawa. Cada uno de los subsistemas cumple adecuadamente con su objetivo, sin embargo en algunos se presentan pequeñas deficiencias en su desempeño, que pueden ser mejoradas como trabajo futuro.

Los software CAD/CAM permitieron la implementacion del proceso de la manufactura; con MasterCAM se logro realizar el diseño del producto, dándonos la posibilidad de preparar la pieza para el proceso de maquinado en una fresa o torno CNC, y configurar algunos parámetros para este, añadiendo la posibilidad de realizar una simulación previa a dicho proceso, los que nos permite observar posibles errores, antes de ingresar a la manufactura del stock. Robotmaster por su lado, realizar una simulación del proceso de manufactura con el Robot KUKA, agregando los aspectos configurados para el maquinado cargados en Mastercam y añadido mas para adecuarlo al manipulador, como son el home de inicio y final, la verificación paso a paso de cada una de las operaciones, el posicionamineto con el cual va a realizar el maquinado el robot, la herramienta a utilizar, posibles colisiones y la generación del código de manufactura adecuado para cargarlo directamente en

el robot.

El robot KUKA mostró realizar el proceso de maquinado de forma satisfactoria, aunque este no sea el propósito para el cual fue creado, ya que como se mencionó en la sección de programación y operación, este es un robot soldador, y la repetibilidad que tiene no es comparable con centros de maquinado CNC especializados, que tiene un error en el orden de micras. El software KSS con el cual se programa y manipula (a través del SmarPAD) el Robot permitió la operación del mismo de una forma clara y eficiente, logrando el desarrollo e implementacion de las diferentes rutinas programadas.

La tarjeta Raspberry Pi mostró ser eficiente para implementar el sistema de visión artificial con la flexibilidad de desarrollarlo en diferentes plataformas de software libre, lo cual resulta interesante ya que da una amplia posibilidad de adecuarse a las necesidades de una celda de manufactura cambiante. La ventaja de tener una computadora embebida con estas prestaciones, aunado a su portabilidad, se reflejan en el desarrollo del sistema de visión en un único software (sin la necesidad de desarrollar o utilizar algún otro programa), el cual adquiere la imagen, la procesa e intercambia datos con los subsistemas que se encuentra conectado vía remota a este.

El procesamiento digital de imágenes implementado es básico pero cumple con su objetivo, que es obtener el punto de inicio de maquinado y sería posible realizar mejoras a esté, que disminuya el error al obtener el centroide del stock, al menos con un valor próximo a la repetibilidad de robot que es de $0{,}04mm$, y con esto logra tener la mayor eficiencia del manipulador al realizar la manufactura.

Seria importante mencionar como un aspecto importante y relevante, la interconexion tanto con la microcomputadora como la que estable esta con el SmartPAD de KUKA gracias a la libreria KUKAVARPROXY, ya que esté tipo de flexibilidad en el software nos permite el manejo del robot de una forma remota y sin la necesidad de estar operandolo directamente.

Una de las desventajas de este sistema embebido es el tiempo en el procesamiento de imágenes, ya que al ser un sistema compacto no cuenta con los recursos necesarios, para realizar este proceso en un tiempo menor(en comparativa con sistemas o computadoras especializadas en PDI), esto se podría mejorar adquiriendo un modelo de Rapsberry PI mas reciente, con mejores prestaciones y como segundo punto optimizar los procesos que adquieren las características de la imagen, lo cual en conjunto reduciría estos tiempo en gran medida.

6.1.1. Trabajo a futuro

Seria conveniente sustituir la cámara por otra con mejores prestaciones, que tenga la capacidad de manipular los parámetros internos de captura y las características de resolución, con lo que se lograría obtener una mejor captura del área de trabajo.

Es preciso mejorar el procesamiento de imágenes, para lograr empatar el error de aproximación al centroide, con la repetibilidad del Robot y así obtener la mayor eficiencia del mismo. Esto mejorando cada una de las etapas del procesamiento digital de imágenes.

La Rasberry Pi tendría que ser sustituida por un modelo mas reciente, que incremente principalmente la velocidad de procesamiento, y que sus memorias de almacenamiento temporal tengan una mayor capacidad.

Otro trabajo a futuro seria establecer un protocolo de comunicación, para detección de fallas en cada uno de los subsistemas o etapas de la manufactura, logrando tener un sistema con mayor seguridad.

Bibliografía

[1] International Federation of Robotics. www.ifr.org. Accedido en marzo 2018.

[2] J. Kruger, T.K. Lien, A. Verl. Cooperation of human and machines in assembly lines. Institute for Machine Tools and Factory Management, Technical University Berlin. Berlin. Germany. 2009.

[3] Y. Chen, F. Dong. Robot machining: recent development and future research issues. Department of Mechanical Engineering. The University of Hong Kong. Pokfulam. Hong Kong. China. 2013

[4] The Robotic Industries Association. www.robotics.org. Accedido en marzo 2018.

[5] T Brogardh. Present and future robot control development An industrial perspective. Annual Reviews in Control. 2010

[6] Bisu C, Cherif M, Gerard A, Nevez J. Dynamic behavior analysis for a six axis industrial machining robot, Proc ICASAAM. Bucharest, Romania. 2011.

[7] Duma C, Caro S, Garnier S, Furet B. Joint stiffness identification of six-revolute industrial serial robots. Robot Comput Integr Manuf. 2011

[8] Olabi A, Bearee R, Nyiri E, Gibaru O. Enhanced trajectory planning for machining with industrial six-axis robots. IEEE International conference on Industrial Technology. France. 2010.

[9] Kuka Robot Group. www.kuka.com. Accedido en marzo 2018.

[10] Yaskawa Motoman Robotics. www.motoman.com. Accedido en marzo 2018.

[11] ABB Robotics. www.abb.com. Accedido en marzo 2018.

[12] Fanuc Robotics. www.fanucrobotics.com. Accedido en marzo 2018.

[13] Robomaster. www.robotmaster.com. Accedido en marzo 2018.

[14] Hyungwon Sung, Sukhan Lee. A Robot-Camera Hand/Eye Self-Calibration System Using a Planar Target. Department of Electrical and Computer Engineering. Department of Interaction Science Sungkyunkwan University Suwon. Republic of Korea, 2013.

[15] Ivan Lundberg, Marten Bjorkman, Petter Ogren.Intrinsic Camera and Hand-Eye Calibration for a Robot Vision System using a Point Marker. 2014 14th IEEE-RAS International Conference on Humanoid Robots (Humanoids). Madrid, España. 2014.

[16] Hema Chengalvarayan Radhakrishnamurthy, Paulraj Murugesapandian, Nagarajan Ramachandran, Sazali Yaacob. Stereo Vision Systema For a Bin Pilcking Adept Robot. School of Mechatronic Engineering University Malaysia Perlis. Jejawi, Perlis, Malaysia. 2007.PDI robot.

[17] Carlos Giraldo, Juan Arroyave. Carlos Montolla. Dimensionamiento de Piezas en un Sistema de Visión Aplicado a una Celda de Manufactura. Scientia et Technica Año XIV No 38, Junio de 2008. Universidad Tecnológica de Pereira. Pereira. Colombia. 2014.

[18] Eddie Sobrado Malpartida, Julio C. Tafur Sotelo, Sistema de Visión Artificial para el Reconocimiento y Manipulación de Objetos Utilizando un Brazo Robot. Pontificia Universidad Católica del Perú. Lima. Perú. 2008.

[19] Rogelio Rojas Hernández, Ramón Silva Ortigoza, María Aurora Molina Vilchis. La Visión Artificial en la Robótica. Instituto Politécnico Nacional UPIITA. Ciudad de México. CDMX, 2007. aaaaaa

[20] M. Peña, I López-Juárez, J. Corona, K.Ordaz, Visión para Robots en Tareas de Ensamble, CIATEQ, Centro de Tecnología Avanzada, Querétaro, México, 2002.

[21] Mario Peña, Ismael López, Reyes Rìos. Proceso de Aprendizaje con Algoritmo Robusto para la Obtención del POSE de Objetos en Líneas de Ensamble con Robots en Tiempo Real (RT). UNAM-IIMAS, CIATEQ-CTA. Información Tecnológica-Vol. 17 N°2-2006, pág.: 61-69.

[22] KUKA System Software 8.3, Operating and Programming Instructions for System Integrators, KUKA Roboter GmbH, KUKA, Augsburg, Germany, KSS 8.3 SI V4, 2015.

[23] KR 5 arc HW, KUKA Roboter GmbH, KUKA, Gersthofen, Germany, 2018.

[24] KR C2, KR C3, Expert Programming, KUKA Roboter GmbH, KUKA, Augsburg, Germany, BA KR C4 extended NA V1, 26 september, 2003.

[25] KR C4 extended NA, KR C4 extended CK NA, Operating Instructions, KUKA Roboter GmbH, KUKA, Augsburg, Germany, BA KR C4 extended NA V1, 2013.

[26] KR C4, KR C4 CK, Operating Instructions, KUKA Roboter GmbH, KUKA, Augsburg, Germany, BA KR C4 extended NA V1, 2015.

[27] https://www.mastercam.com. Accedido en marzo 2018.

[28] Miguel Cazorla, Robótica y Visión Artificial, Universidad de Alicante. Alicante. España. 2000.

[29] Álvaro Guerrero A., Paula Jimena Ramos G., Sistema embebido de bajo costo para visión artificial, Universidad Tecnológica de Pereira, Pereira, Colombia. 2014.