

Introducción

En la logística y transporte de mercancías existe una actividad diaria de gran importancia que consiste en el empaquetamiento de cajas dentro de contenedores (vehículos), esta tarea representa un problema clásico y de alta complejidad en la Investigación de Operaciones. Industrias que han alcanzado un alto nivel de automatización luchan constantemente por resolver de forma eficiente problemas como el empaquetamiento óptimo de mercancías, encontrando en el camino lo complejo que resulta este problema bajo restricciones reales y lo difícil que puede ser la instalación de soluciones automáticas.

Grandes superficies de abastecimiento y empresas consolidadas de transporte deben resolver el problema de carga/descarga de mercancía en contenedores diariamente, encontrando que los tiempos requeridos para esta labor realizada de forma manual no suelen ser razonables. Dado que un buen empaquetamiento de mercancías es aquel que logra aprovechar al máximo el volumen disponible, la tarea de carga/descarga consiste entonces en posicionar cajas dentro del contenedor de una forma lógica, calculando los espacios, superficies de apoyo y límites de peso para cumplir las condiciones con que se debe transportar la carga sin exceder los límites de tiempo impuestos para esta labor. Este problema que a simple vista parece un problema de optimización ha sido resuelto a través de diferentes metodologías de solución, una muy común suele ser la automatización (robotización) cuya finalidad es eliminar el elemento humano dado que la tarea llevada a cabo por estos (transporte y posicionamiento de las cajas de su lugar de reposo a su posición dentro del contenedor) suele ser la de mayor consumo de tiempo. Otra metodología común es la optimización matemática cuya finalidad es garantizar un patrón de empaquetamiento de la mercancía que minimice el espacio (volumen) desperdiciado en el proceso de carga. Dado que ambos enfoques intentan resolver el mismo problema desde diferentes perspectivas, estos son aplicados dependiendo de la capacidad solvente (monetaria) de la empresa. Ya que la infraestructura automatizada de carga/descarga no guarda los mismos niveles de inversión que una solución informática de optimización. Es evidente pensar entonces que una solución eficiente para el problema general de empaquetamiento de mercancías es aquella metodología que integra la automatización de tareas con la optimización matemática de estas.

Este estudio contribuye en resolver de forma integrada el problema de empaquetamiento de mercancías en un ambiente automatizado (robotizado) donde los patrones de empaquetamiento deben satisfacer las restricciones prácticas de la industria. En este trabajo se presenta entonces una metodología de solución para el problema de carga de contenedores (3D-SLOPP, Three dimensional Single Large Object Placement Problem) considerando restricciones de la vida real como orientación y límites de apilabilidad de las cajas, peso máximo de carga, carga fraccionada en diferentes destinos y estabilidad estática de la carga. Además de presentarse un modelo cinemático inverso para la programación de un robot estándar que permita traducir patrones de empaquetamiento complejos.

GENERADOR DE CÓDIGO PARA ROBOT YASKAWA HP20D NX-100- APLICACIÓN DE CORTE EN DOS DIMENSIONES

RESUMEN

Este reporte técnico presenta un generador automático de código para programar el robot HP20D-NX100 de Yaskawa; este generador implementa el modelo geométrico inverso para la conversión de variables cartesianas a variables articulares. El reporte utiliza como referencia el caso de aplicación: corte rectangular óptimo en dos dimensiones. Se espera que este documento así como los archivos que lo acompañan, reduzca los esfuerzos de estudiantes, docentes e investigadores al momento de realizar experimentación con el robot Yaskawa HP20D-NX100 y los equipos relacionados.

DESCRIPCIONES PREVIAS

En esta sección se presentan descripciones asociadas con: (1) los equipos de laboratorio utilizados y (2) el caso de aplicación tomado como referencia.

Equipos utilizados.

Para el desarrollo del reporte técnico se utilizaron los siguientes componentes:

Hardware:

- Robot Yaskawa HP20D de Motoman [1]
- Controlador DX100 [2]
- Sistema de corte por plasma Cebora Prof 163 [3]
- Sistema de extracción de vapores JetClean DF [4]
- Estación PC con conexión a internet y aplicaciones instaladas

Software

- MotoSimEG Version 3.63 con licencia [5]
- Matlab y Compilador de C++

Caso de aplicación: corte rectangular óptimo en dos dimensiones

Se tiene una pieza rectangular mayor, en donde se deben distribuir piezas rectangulares más pequeñas, de las que se conoce el tamaño y un costo. El objetivo es maximizar el beneficio asociado a cada una de las piezas pequeñas, sin sobreponer las piezas y sin sobrepasar los límites

de la pieza rectangular mayor. Este problema tiene incidencia directa en el costo final de productos fabricados en empresas de corte, transporte y almacenamiento de materiales [6].

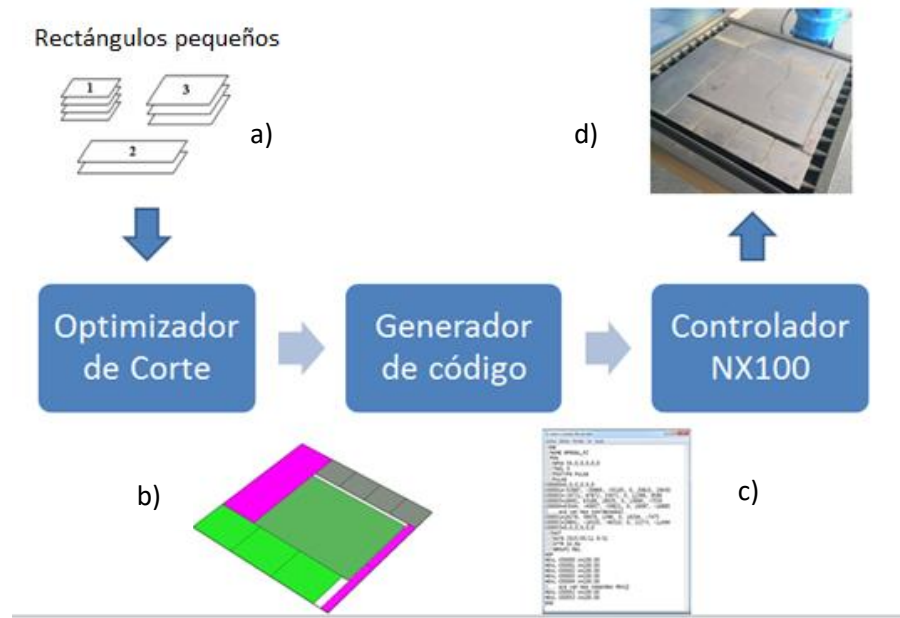


Fig. 1. Etapas de procesamiento para resolver el corte óptimo en dos dimensiones. Fuente: Los autores

Una posible arquitectura para la solución del problema es ilustrada en la Fig. 1. En esta figura la entrada está compuesta por el inventario y las medidas de rectángulos pequeños que se desea obtener, así como las dimensiones del rectángulo mayor. La salida son las piezas cortadas como se presenta en la Fig. 1-d. A continuación se describen los bloques de procesamiento

- **Optimizador de Corte.** Este bloque recibe dos parámetros: (1) dimensiones de las piezas rectangulares pequeñas, (2) cantidad de piezas rectangulares pequeñas con la misma dimensión, (3) dimensiones de la pieza rectangular mayor. El bloque se encarga de localizar los rectángulos pequeños de forma óptima, es decir, maximizando el beneficio asociado a cada una de las piezas pequeñas, sin sobreponer las piezas y sin sobrepasar los límites de la pieza mayor. En la Fig. 1-b se presenta una distribución calculada por el bloque.
- **Generador de Código.** Este bloque recibe un conjunto de coordenadas en el espacio cartesiano. Las coordenadas indican la localización de esquinas de rectángulos pequeños. El bloque se encarga de generar un programa que gobierne al robot para realizar la tarea de corte de los rectángulos pequeños. La Fig 1-c presenta uno de los programas generados. En la siguiente sección se describirán detalles de dicho programa.
- **Controlador NX100.** Este controlador se encarga de ejecutar el programa generado, haciendo que el robot realice los cortes sobre la pieza rectangular.

El presente reporte técnico se enfoca en la descripción del bloque *Generador de Código*. Los bloques Optimizador de Corte y Controlador NX100 pueden ser consultados en las referencias [6] y [2], respectivamente.

GENERADOR DE CÓDIGO

El generador de código será presentado en dos secciones: (1) requerimiento y (2) solución propuesta.

Requerimiento para el generador de código

La entrada al sistema será un conjunto de coordenadas de n rectángulos. Cada rectángulo es representado por dos esquinas de este (la esquina inferior izquierda y la esquina superior derecha) dadas en un sistema de coordenadas cartesianas. La Fig. 2 ilustra las dos coordenadas $c1$, $c2$, de un rectángulo de 29 unidades de largo y 14 de ancho (las unidades son dadas en milímetros).

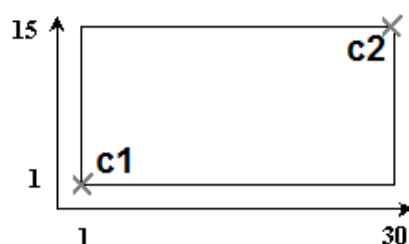


Fig. 2. Coordenadas del rectángulo. $c1$ es (1,1), $c2$ es (30,15)

La entrada formal consiste en un archivo plano con nombre *input.txt*. Dicho archivo contiene $n+1$ líneas. Cada línea representa las coordenadas $c1$, $c2$ en espacio articular, del rectángulo a ser cortado. La primera línea corresponde con el rectángulo mayor o placa de material de la cual debemos recortar las piezas demandadas. Un ejemplo de la entrada se presenta en la Fig. 3-a. En este ejemplo se tiene una placa de 800mm x 800mm de la cual se desean sacar once rectángulos.

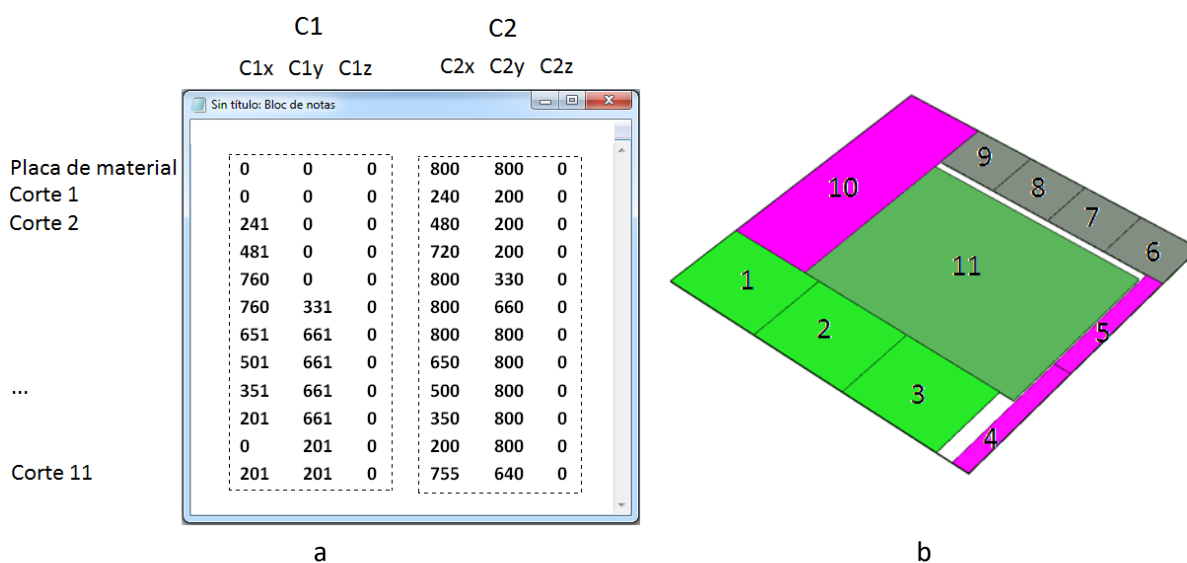


Fig. 3. Ejemplo de entrada al generador de código. (a) estructura del archivo de entrada input.txt. (b) representación gráfica del archivo input.txt

La salida requerida es un programa *off-line* (Programa.JOB) para controlar la tarea de corte con el robot YASKAWA- HP20D. Este programa está escrito en el lenguaje de bajo nivel utilizado por el controlador NX100. En su forma más simple, este lenguaje define una sintaxis que puede ser resumida en tres secciones: (1) encabezado, (2) definición de constantes articulares y (3) comandos de actuación para el robot. La Fig. 4-b presenta un ejemplo de salida.

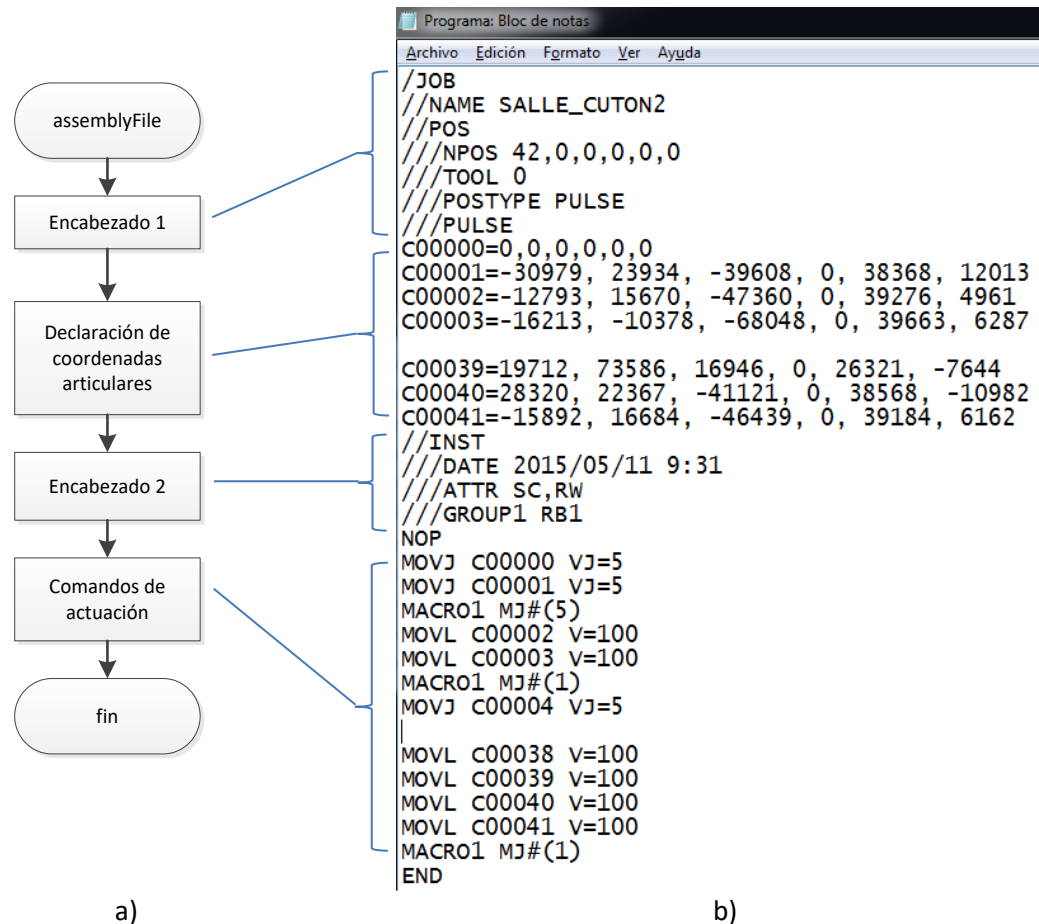


Fig. 4. Obtención del archivo de salida: Programa.JOB. (a) flujograma propuesto para el bloque assemblyFile. (b) ejemplo de salida; en este ejemplo se omiten varias de las coordenadas y de los comandos de actuación. Para una descripción de los comandos consulte [7]

Solución propuesta -

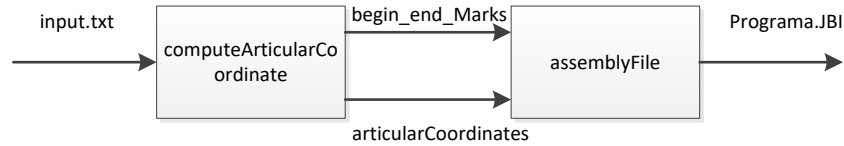


Fig. 5. Diagrama de bloques

La Fig. 5 presenta el diagrama de bloques propuesto para resolver el problema. En este diagrama, el bloque denominado *computeArticularCoordinate*, se encarga de convertir los datos en el archivo *input.txt* a coordenadas del espacio articular. Estas coordenadas son consumidas por el bloque *assemblyFile*, el cual se ocupa de ensamblar el archivo *Programa.JOB* (ver Fig. 4-b). El bloque *assemblyFile* requiere marcadores de la coordenada inicial y final en cada rectángulo; esta información está contenida en el vector *begin_end_Marks*.

Este reporte solo describirá en detalle el bloque *ComputeArticularCoordinate* puesto que es el bloque de mayor complejidad. Un flujograma para la solución del bloque *assemblyFile* es presentado en la Fig. 4-a.

Bloque *ComputeArticularCoordinate*

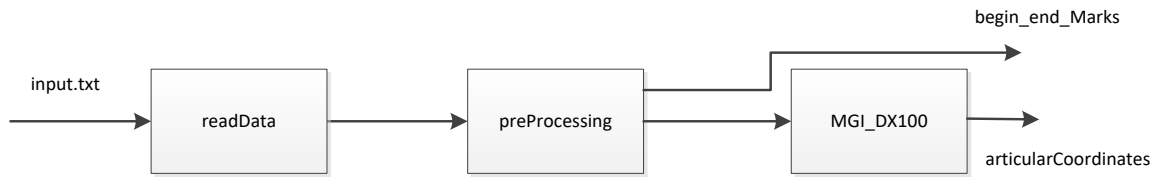


Fig. 6 Sub-bloques o módulos del generador de código

Este bloque se encarga de tres tareas básicas: (1) leer datos de entrada, (2) precondicionar datos y (3) calcular coordenadas articulares. Estas operaciones son ilustradas en la Fig. 6. Cada sub-bloque de la Fig. 6 se formuló como un módulo software, compuesto por funciones de usuario. Las Tablas 2 a 6 describen cada función tomando la plantilla de la Tabla 1 como referencia. El grupo de funciones diseñadas se integra según el flujograma de la Fig. 7

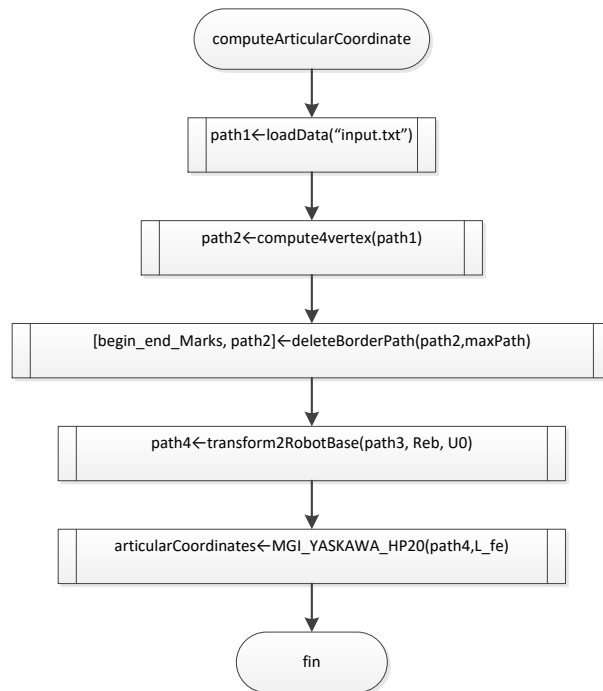


Fig. 7. Flujograma de integración de funciones para el bloque computeArticularCoordiante

Tabla 1. Plantilla para la documentación de interfaces. Ref [xx]

Nombre de componente	Nombre con el que se declara la función
Retornos	Descripción del valor o valores de retorno de la operación
Argumento 1	Descripción del primer argumento
Argumento 2	Descripción del segundo argumento
Argumento n	Descripción del n-ésimo argumento
Descripción	Descripción de las responsabilidades de la función
Nota	Especificación de restricciones de ejecución y posibles modos de fallo
Módulo	Nombre del módulo al que pertenece la función

Tabla 2. Función loadData

Nombre de componente	loadData								
Retornos	<p>Datos de coordenadas de cada rectángulo. Los datos de un rectángulo se entregan en un vector de dos filas y tres columnas, por ejemplo</p> <table><tr><td>C1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>C2</td><td>240</td><td>200</td><td>0</td></tr></table> <p>C1 representa la coordenada 1 en el espacio articular x, y, z. Esquina inferior izquierda C2 representa la coordenada 2 en el espacio articular x, y, z. Esquina superior derecha</p>	C1	0	0	0	C2	240	200	0
C1	0	0	0						
C2	240	200	0						
Argumento 1	Archivo de texto con nombre input.txt y formato indicado en la Fig. 3-a								
Descripción	Esta función se ocupa de leer el archivo input.txt, crear una vector y cargar los								

	datos en dicho vector. La función debe acceder al archivo input.txt
Nota	<p>Posibles modos de fallo.</p> <ul style="list-style-type: none"> • Inexistencia del archivo input.txt • Archivo vacío • Archivo input con tipos de datos errados • Archivo input con estructura errada <p>Esta función puede cargar datos de múltiples rectángulos. Para este caso el retorno tendrá dimensiones $[2N,3]$, en donde, N es la cantidad de rectángulos</p>
Módulo	readData

Tabla 3. Función compute4vertex

Nombre de componente	compute4vertex																				
Retornos	<p>Conjunto de vértices para cada rectángulo. Espacio cartesiano x, y, z. Los vértices son dados en la secuencia C1 C3 C2 C4 C1. Esta secuencia define la ruta de corte del rectángulo. Note que la coordenada C1 debe repetirse con el propósito de cerrar el rectángulo</p> <p>Por ejemplo,</p> <table><tr><td>C1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>C3</td><td>0</td><td>200</td><td>0</td></tr><tr><td>C2</td><td>240</td><td>200</td><td>0</td></tr><tr><td>C4</td><td>240</td><td>0</td><td>0</td></tr><tr><td>C1</td><td>0</td><td>0</td><td>0</td></tr></table>	C1	0	0	0	C3	0	200	0	C2	240	200	0	C4	240	0	0	C1	0	0	0
C1	0	0	0																		
C3	0	200	0																		
C2	240	200	0																		
C4	240	0	0																		
C1	0	0	0																		
Argumento 1	<p>Conjunto de dos vértices para cada rectángulo. Espacio cartesiano x, y, z. Los vértices son dados en la secuencia C1, C2</p> <p>Por ejemplo,</p> <table><tr><td>C1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>C2</td><td>240</td><td>200</td><td>0</td></tr></table>	C1	0	0	0	C2	240	200	0												
C1	0	0	0																		
C2	240	200	0																		
Descripción	Esta operación se encarga de calcular los vértices C3 y C4 del rectángulo a partir de los vértices C1 y C2. Ver Fig. 2																				
Nota	<p>Posibles situaciones de fallo</p> <ul style="list-style-type: none">Errores en relación de magnitudes C1x<=C2x C1y<=C2y C1z< C2zEntrada con dimensiones erradas <p>Este bloque puede recibir coordenadas asociadas a más de un rectángulo. En esta situación el argumento1 tendrá dimensiones [2*N , 3], en donde, N es el número de rectángulos. La salida para esta situación tendrá dimensiones [2*N+1 , 3]</p>																				
Módulo	preProcessing																				

Tabla 4. Función deleteBorderPath

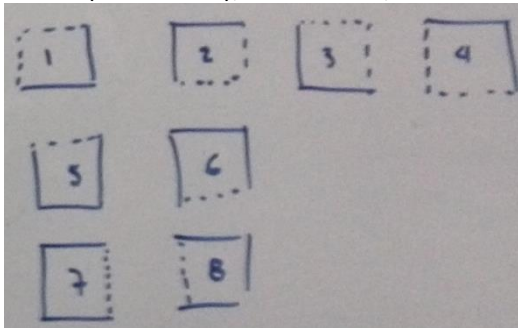
Nombre de componente	deleteBorderPath
Retorno1	Vector con marcadores de inicio y fin de la ruta de corte del rectángulo. El inicio es marcado con el número 1. El fin es marcado con el número 2. Las coordenadas intermedias son marcadas con el número 0. Por ejemplo, [0 -1 1]
Retorno2	<p>Conjunto reducido de vértices para el corte de cada rectángulo. La cantidad de vértices depende de la ubicación del rectángulo en el área de la pieza a cortar. Existen nueve posibles ubicaciones detalladas en la descripción.</p> <p>Ejemplo de salida,</p> <pre> C1 0 200 0 C2 240 200 0 C3 240 0 0 </pre>
Argumento 1	<p>Conjunto de vértices para cada rectángulo. Espacio cartesiano x, y, z. Los vértices son dados en la secuencia C1 C3 C2 C4 C1</p> <p>Por ejemplo,</p> <pre> C1 0 0 0 C3 0 200 0 C2 240 200 0 C4 240 0 0 C1 0 0 0 </pre>
Argumento 2	<p>Coordenadas del material que será cortado. Por ejemplo,</p> <pre> Cmax1 0 0 0 Cmax2 800 800 0 Cmax3 0 800 0 Cmax4 800 0 0 </pre>
Descripción	<p>Este bloque se encarga de eliminar las rutas de corte que se superponen con los bordes de la pieza a cortar. Estas rutas pueden aparecer en 8 ubicaciones documentadas en la Fig. 8. En esta figura, la línea punteada representa la ruta que debe ser eliminada. El caso del ejemplo documentado corresponde con la ubicación 4</p> <p>Existe una situación en la que la ruta de corte es equivalente a los cuatro bordes del material a ser cortado (situación 9); en ese caso, se rechaza el corte</p>  <p>Fig. 8. Posibles ubicaciones del rectángulo en la pieza a cortar. La línea punteada indica superposición con el borde de la pieza a cortar</p>
Nota	Especificación de restricciones de ejecución y posibles modos de fallo
Módulo	Preprocesamiento

Tabla 5. Función transform2RobotBase

Nombre de componente	transform2RobotBase
Retornos	Coordenadas de los vértices del rectángulo. Son dadas en el sistema de referencia con origen en la base del robot. Por ejemplo, 928.4100 -396.0260 174.0300 928.4100 -156.0260 174.0300 728.4100 -156.0260 174.0300
Argumento 1	Coordenadas de los vértices del rectángulo. Son dadas en el sistema de referencia situado en el origen de la pieza a cortar. Por ejemplo, C1 0 200 0 C2 240 200 0 C3 240 0 0
Argumento 2	Matriz de transformación de coordenadas. La transformación indica la orientación del efector final con respecto a la base del robot. Para orientaciones perpendiculares a la base del robot se usa la siguiente expresión: $R_{eb} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}$
Argumento 3	Este es un parámetro de la función. Representa la posición del origen de la pieza a cortar con respecto al sistema de referencia de base del robot. Por ejemplo, U0=[728.410;-396.026;174.0300];
Descripción	Esta operación implementa las ecuación (2) del Anexo 1.
Nota	Especificación de restricciones de ejecución y posibles modos de fallo.
Módulo	Pre-Procesamiento

Tabla 6. Función MGI_YASKAWA_HP20

Nombre de componente	MGI_YASKAWA_HP20D
Retornos	Conjunto de coordenadas en el espacio articular. Por ejemplo, -30979 23934 -39608 0 38368 12013
Argumento 1	Conjunto de coordenadas en el espacio cartesiano. Este espacio debe tomar como punto de origen la base del robot. Por ejemplo, 928.4100 -396.0260 174.0300
Argumento 2	Longitud del efector final a lo largo del eje z, dada en mm. Esta longitud depende del efector que se utiliza con el robot. Por ejemplo, 639.8000
Descripción	Esta función implementa las ecuaciones (4) a (23) indicadas en el anexo 1
Nota	Especificación de restricciones de ejecución y posibles modos de fallo. En caso de que la zona a la que se desee llegar esté por fuera del espacio de trabajo del robot, esta función debe rechazar la operación.

Método

Resultados

Conclusiones

Trabajos citados

- [1] YASKAWA Motoman Robotics, «Solutions in Motion-HP20,» 06 2012. [En línea]. Available: http://www.motoman.com/datasheets/HP20D_HP20D-6.pdf. [Último acceso: 27 08 2015].
- [2] YASKAWA Motoman Robotics, «Solutions in motion-NX100,» 06 2011. [En línea]. Available: <http://www.motoman.com/datasheets/NX100%20Controller.pdf>. [Último acceso: 27 8 2015].
- [3] CEBORA Welding and cutting, «PLASMA PROF 163 ACC,» [En línea]. Available: http://www.cebora.it/depliant/art_957.pdf. [Último acceso: 27 8 2015].
- [4] CORAL Antipollution systems, «CORAL,» 2012. [En línea]. Available: <http://www.coral.eu/catalogo/cleaning-jetcleanDF.pdf>. [Último acceso: 27 8 2015].
- [5] YASKAWA Motoman Robotics, «Solutions in Motion,» 06 2011. [En línea]. Available: <http://www.motoman.com/datasheets/MotoSim%20EG.pdf>. [Último acceso: 27 8 2015].
- [6] S. Imahori, M. Yagiura, S. Umetani, S. Adachi y T. Ibaraki, «Local Search Algorithms for the Two-Dimensional Cutting Stock Problem with a Given Number of Different Patterns,» de *Metaheuristics: Progress as Real Problem Solvers*, Springer Link, 2002, pp. 181-202.
- [7] YASKAWA , «INFORM MANUAL NX100 - MANUAL NO. RE-CKI-A444,» 04 2004. [En línea]. Available: <http://www.ro.feri.uni-mb.si/predmeti/robotizacija/inform.pdf>. [Último acceso: 26 8 2015].
- [8] D. Mariño Lizarazo, INTERFAZ NATURAL PARA LA PROGRAMACIÓN DE UN ROBOT MANIPULADOR A TRAVÉS DE UN KINECT, Bogotá: Unisalle, 2014.

Anexos

Anexo 1 – Ecuaciones implementadas en el bloque computeArticularCoordinate

Calcula vértices

La función compute4vertex debe calcular los cuatro vértices en el sistema de referencia situado en el origen de la pieza a cortar. Este cálculo busca obtener los vértices c3 y c4 a partir de c1 y c2. Una ecuación para esta obtención es,

$$\begin{aligned} c_3 &= (c_{1x}, c_{2y}, 0) \\ c_4 &= (c_{2x}, c_{1y}, 0) \end{aligned} \quad 1$$

Transformación de coordenadas a la base del robot.

Este bloque busca realizar un desplazamiento de los cuatro vértices hacia el sistema de referencia situado en el origen del robot. Para ello se usa la siguiente expresión para transformación de sistemas de coordenadas:

$$P_e^b = P_o + R_{eB}x \quad 2$$

En donde,

P_o es la posición del origen de la pieza a cortar con respecto al sistema de referencia de base del robot. Puede ser tomada como [550, -450, 460]¹.

R_{eB} es la matriz de transformación de coordenadas calculada tomando como referencia la Fig. 20 en [8]. Esta matriz asume orientación constante del efector final.

$$R_{eb} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \quad 3$$

x es la coordenada en el sistema de referencia situado en el origen de la pieza a cortar; es decir, aquí es donde se reemplazan las coordenadas c_1 a c_4 .

P_e^b es la coordenada transformada;

Modelo geométrico inverso del robot

Este bloque se encarga de convertir cada una de las coordenadas cartesianas (o vértices del rectángulo) en un grupo de seis coordenadas articulares. Cada coordenada articular indica la posición de cada uno de los actuadores del robot. Este bloque se implementa mediante uso de las ecuaciones presentadas a continuación y extraídas del documento [8].

¹ Esta coordenada fue tomada a partir de la localización de la antorcha usada durante simulación (ver Fig 1). Para pruebas con el robot instalado en laboratorio, la longitud debe ser actualizada con valores tomados en Laboratorio

La Tabla 7 presenta los parámetros del Robot

Tabla 7. Parámetros del robot YASKAWA – HP20. Modificado de [8]

Parámetro	Valor (mm)	Descripción
L1z	505	Longitud de articulación 1 en eje z
L1x	150	Longitud de articulación 1 en eje x
L2	760	Longitud de articulación 2
L3	140	Longitud de articulación 3
L4	795	Longitud de articulación 4
Lh	105+(L_elementoFinal)	Longitud de articulación 5
L_elementoFinal	340 ¹	Longitud de elemento final

¹ Esta longitud fue tomada a partir de la antorcha usada durante simulación (ver Fig 1). Para pruebas con el robot instalado en laboratorio, la longitud debe ser actualizada con valores tomados en Laboratorio

Calculo de coordenada q1

$$q_1 = \text{atan2}(P_{aux2}, P_{aux1}) \quad 4$$

En donde

P_{aux2} es el elemento 2 del vector Paux

P_{aux1} es el elemento 1 del vector Paux

$$P_{aux} = P_e^b - R_e^b \begin{bmatrix} 0 \\ 0 \\ L_h \end{bmatrix} \quad 5$$

Para un trabajo de corte sobre una superficie horizontal la matriz R_e^b es constante y se calcula como:

$$R_e^b = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \quad 6$$

De otro lado, la matriz P_e^b representa la matriz de coordenadas en el espacio cartesiano, siendo

$$P_e^b = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad 7$$

Note que al considerar corte en plano horizontal, la coordenada z será constante.

Calculo de coordenada q3

$$q_3 = \Theta_2 + y \quad 8$$

En donde,

$$y = \tan^{-1} \left(\frac{L_4}{L_3} \right) \quad 9$$

$$\Theta_2 = \text{Atan2}(\sin(\Theta_2), \cos(\Theta_2)) \quad 10$$

$$\sin(\Theta_2) = -\sqrt{1 - \cos(\Theta_2)^2} \quad 11$$

$$\cos(\Theta_2) = \frac{P_{w1}^2 + P_{w3}^2 - L_2^2 - a_1^2}{2L_2a_1} \quad 12$$

$$a_1 = \sqrt{(L_3^2 + L_4^2)} \quad 13$$

$$P_w = \begin{bmatrix} \sqrt{P_{aux1}^2 + P_{aux2}^2 - L_{1x}^2} \\ 0 \\ P_{aux3} - L_{1z} \end{bmatrix} \quad 14$$

Al calcular esta coordenada es importante tener en cuenta la restricción de zona de trabajo [8]:

“Si $\sqrt{P_{w1}^2 + P_{w3}^2} > (L_2 + a_1)$, entonces, la zona a la que se quiere llegar está por fuera del espacio de trabajo, por lo que no existe solución”

Cálculo de coordenada q2

$$q_2 = \frac{\pi}{2} - \Theta_1 \quad 15$$

En donde,

$$\Theta_1 = \alpha + \beta \quad 16$$

$$\alpha = \text{Atan2}(P_{w3}, P_{w1}) \quad 17$$

$$\beta = \cos^{-1} \left(\frac{L_2^2 + P_{w1}^2 + P_{w3}^2 - a_1^2}{2L_2\sqrt{P_{w1}^2 + P_{w3}^2}} \right) \quad 18$$

Cálculo de coordenadas q4 a q6

Para calcular estas coordenadas, es necesario resolver la siguiente matriz,

$$R_e^3 = R_b^3 R_e^b = \begin{bmatrix} n_x & s_x & a_x \\ n_y & s_y & a_y \\ n_z & s_z & a_z \end{bmatrix} \quad 19$$

Los elementos de esta matriz pueden ser calculados con las siguientes expresiones

$R_b^3 = R_3^{b^T}$; en donde el súper índice T indica matriz transpuesta

Así mismo:

$$R_3^b = \begin{bmatrix} -s_1 & c_1 c_2 c_3 + c_1 s_2 s_3 & c_1 c_2 s_3 - c_1 s_2 c_3 \\ c_1 & s_1 s_2 s_3 + s_1 c_2 c_3 & s_1 c_2 s_3 - s_1 s_2 c_3 \\ 0 & \sin(q_3 - q_2) & -\cos(q_3 - q_2) \end{bmatrix} \quad 20$$

En donde para aligerar la escritura c_i representa $\cos(q_i)$, s_i representa $\sin(q_i)$

Finalmente, las coordenadas q4 a q6 pueden ser calculadas como

$$q_4 = \text{Atan2}(-a_x, a_z) \quad 21$$

$$q_5 = \text{Atan2}\left(a_y, \sqrt{(n_y)^2 + (s_y)^2}\right) \quad 22$$

$$q_6 = \text{Atan2}(-n_y, s_y) \quad 23$$

Bloque de ganancia.

Este bloque se encarga de convertir las variables articulares a pulsos de encoder. Para ello es suficiente multiplicar cada variable por las constantes de encoder presentadas en la Tabla 8.

Tabla 8. Coeficientes de conversión de grados de articulación a pulsos de encoder en articulación. Tomado de [8]

Articulación	Pulsos/grado
q_1	$1341 \cdot 180/\pi$
q_2	$1783 \cdot 180/\pi$
q_3	$1376 \cdot 180/\pi$
q_4	$910 \cdot 180/\pi$
q_5	$909 \cdot 180/\pi$
q_6	$520 \cdot 180/\pi$