

Generación automática de trayectorias para una aplicación de lijado robotizado

Eduardo Fuentes Fierro¹, Erardo Leal Muñoz², Eduardo Diez Cifuentes³

¹Departamento de Ingeniería Mecánica, Universidad de La Frontera, Chile.

Email: e.fuentes07@ufromail.cl

Departamento de Ingeniería Mecánica, Universidad de La Frontera, Chile.

Email: erardo.leal@ufrontera.cl

Departamento de Ingeniería Mecánica, Universidad de La Frontera, Chile.

Email: eduardo.diez@ufrontera.cl

Resumen

En este trabajo se presenta un método de generación automática de trayectorias, basado en el procesamiento digital de la geometría de la pieza. El método comienza con la extracción de las propiedades dimensionales de la pieza. A continuación, mediante un sistema modular de funciones, es posible clasificar las superficies en planas y curvas, para luego generar puntos sobre esas superficies. Luego, considerando las características de la operación de lijado, los puntos generados se ordenan secuencialmente para definir la trayectoria del robot. Finalmente, estas trayectorias son exportadas de manera automática en un script de programación que puede ser leído por el controlador del robot articulado tarea. El algoritmo tiene el potencial de acortar los tiempos de programación de robots industriales, y podría tener aplicación en otras operaciones especiales de manufactura que utilicen robots.

Palabras clave: Programación fuera de línea, generación de trayectorias, robotización, Industria 4.0

Abstract

In this work, a method for automatic path generation is presented, using as input a digital file of the geometry of the production piece. To do this, the dimensional properties of the piece are extracted, and by a modular system of function it's posible to classify flat and curve surfaces, and to create points in them. Then, using these points and considering the characteristics of the sanding process, the points are orderer sequentially. Finally, these paths are exported automatically in a script file that can be readed by the articulated robot for its execution. The use of the algorithm pretends to reduce programming time of industrial robots, and its proposed the development of the same algorithm for other manufacturing processes.

Keywords: fuera de línea programming, path generation, robotization, industry 4.0

1. Introducción

Para las empresas dedicadas a la manufactura, la robótica industrial se presenta como un recurso valioso, debido a que es una tecnología en constante desarrollo, y permite la automatización de tareas del entorno productivo, ayudando a generar procesos más eficientes e incluso mejorar la calidad de los productos debido a la consistencia con la que los robots pueden ejecutar tareas [1]. Sin embargo, la programación de los robots es una tarea compleja y que requiere personal capacitado y con experiencia en el uso de esta tecnología [2]. Además, los robots articulados son poco flexibles cuando se requiere asignar tareas distintas con facilidad, sobre todo a la hora de agregar nuevas piezas de producción, o incluso cambios dentro de los mismos procesos, que pueden consistir en variaciones dependiendo de la geometría de la pieza o la orientación general del espacio de trabajo. Esto representa una barrera para la pequeña y mediana empresa que posee una cantidad limitada de robots articulados y desea sacar el máximo provecho del activo, pues para cambiar de tarea se requiere un programador especializado en el uso del robot y con experiencia en las características especiales de la tarea que se desea automatizar.

Existen varios métodos para la programación de trayectorias de robot articulado y se pueden clasificar como programación por aprendizaje (o manual) y programación fuera de línea. La programación online consiste en que el programador debe mover el robot a través de los puntos deseados para crear trayectorias, ya sea usando el software incluido con el robot, o bien moviendo las articulaciones y registrando los puntos. Por otro lado, la programación fuera de línea consiste en generar las trayectorias con ayuda de software de simulación, en donde ya no se requiere detener la producción, e incluso no es necesario tener acceso al robot para poder programar sus trayectorias.

La programación manual posee varias desventajas respecto a la fuera de línea, pues se requiere personal experimentado moviendo al robot articulado por el espacio de trabajo y esto podría generar accidentes y además tomar mucho más tiempo si la tarea o la pieza de producción son complejas. Como ejemplo, programar manualmente trayectorias de soldadura para la fabricación de estructuras para vehículos grandes puede tomar hasta 8 meses, siendo que la el trabajo de soldadura tarda sólo 16 h aproximadamente [3]. A pesar de esto, la programación por aprendizaje sigue siendo el más utilizado, pues el costo de un software para la programación fuera de línea y la accesibilidad a personal que pueda utilizarlo supone una barrera de entrada mayor

A partir de lo anterior, el desarrollo de métodos para la programación fuera de línea de robots, que sean más accesibles para la industria y en general requieran pocos recursos para ser utilizados supone un aporte importante en el desarrollo de la industria que robotiza sus procesos, y en consecuencia contribuye al desarrollo general de la relación entre los trabajadores y las nuevas tecnologías.

Existe una variedad de estudios que han desarrollado métodos para abordar la programación fuera de línea. En la Figura 1 se muestra una clasificación de los métodos en base a los recursos que se utilizan para la generación de trayectorias. Debido a que la visión artificial podría requerir más recursos, como una cámara, y el desarrollo de modelos que requieren una gran cantidad de datos para funcionar, la programación basada en modelos 3D resulta más sencilla.



Figura 1. Tipos de programación fuera de línea. Fuente: elaboración propia.

La nube de puntos consiste en un conjunto de puntos tridimensionales (x, y, z) que describen la superficie de la geometría de la pieza. Usualmente una nube de puntos se obtiene a partir de un escáner 3D, que resulta ser una herramienta práctica para digitalizar geometrías reales para integrar en métodos de programación fuera de línea.

En Yang et al. [4] se utiliza la nube de puntos para encontrar las juntas que se desean soldar en una pieza, en Bastida et al. [5] se limita el espacio utilizado por la nube de puntos para obtener trayectorias para pintura, y en Wang et al. [6] y Meng et al. [7] son los puntos de la nube que se ordenan de manera óptima para generar trayectorias para el pulido de piezas metálicas.

Las nubes de puntos son ampliamente usadas para la generación de modelos 3D no sólo de piezas de producción, sino también para topología y digitalización de espacios. Dentro de estas funcionalidades, es posible transformar una nube de puntos en un modelo CAD como una malla triangular. Las mallas triangulares consisten en un conjunto de triángulos que conforman la superficie de la pieza. A

pesar de que ambos tipos de digitalización parecen llevar al mismo resultado, se utilizan en aplicaciones distintas. Al igual que para el uso de nube de puntos, existe una variedad de métodos basados en modelos CAD.

En el trabajo de Sheng et al. [8] el problema se aborda descomponiendo superficies complejas en lotes más sencillos para poder generar trayectorias. En Neto et al. [9] se demuestra que es posible generar trayectorias basadas en software CAD y en Lopez et al. [10] se utiliza específicamente la extensión *.stl* como base. En Afroz et al. [11] se identifican superficies y bordes en base a la información dimensional que puede ser extraída de las coordenadas de los triángulos, mientras que en Nikolaj y Henrik [12] muestran un método que funciona en archivos *.stl* especialmente complejos y/o con muchos triángulos.

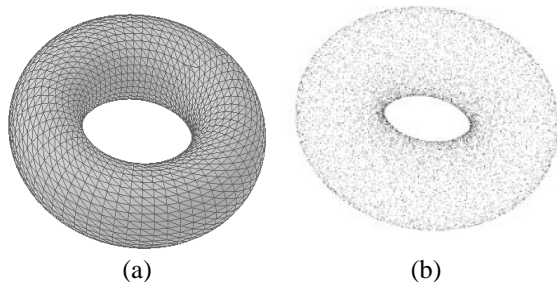


Figura 2. Toroide en formato *.stl* (a) y nube de puntos (b).

A pesar de que las metodologías presentadas en esta revisión no tienen como objetivo el lijado o el pulido, las características de las soluciones para la generación de trayectorias son lo suficientemente generales para poder adaptar las metodologías a los propósitos de esta investigación.

A partir de la revisión, se estima que el uso de modelos CAD en formato *.stl* otorga una generalidad a los procesos de programación, pues es posible transformar una nube de puntos en un archivo de este formato. De esta manera se asegura compatibilidad con los recursos disponibles. De esto, se plantea que la geometría de una pieza en proceso de producción posee información suficiente para generar, mediante algoritmos, las trayectorias de un robot articulado de forma automática, y de ser así, el uso de un algoritmo para la generación de trayectorias de un robot articulado reduciría los tiempos de programación de la tarea del robot.

El objetivo de este trabajo es desarrollar un algoritmo que sea capaz de generar una trayectoria de lijado robotizado sobre una superficie arbitraria de manera automática, que además tenga la capacidad de generar un *script* de programación de robot que sea compatible con los formatos de lectura del

controlador del robot. Además, con el fin de poder manipular los parámetros del algoritmo sin modificar el código, se debe desarrollar una interfaz de usuario.

2. Metodología

El robot utilizado para este trabajo es el UR10e de Universal Robots, y se utilizó una herramienta terminal de lijado especial para el desarrollo de este y otros trabajos. En la Figura 3 (a) se puede ver la estación de trabajo, compuesta por el robot articulado (1), la herramienta terminal de lijado (2), la mesa de trabajo (3) y el utillaje de vacío (4). En la Figura 3 (b) se ve el detalle de la herramienta terminal, con la brida del robot (5), el servo motor (6) y la lija (7).

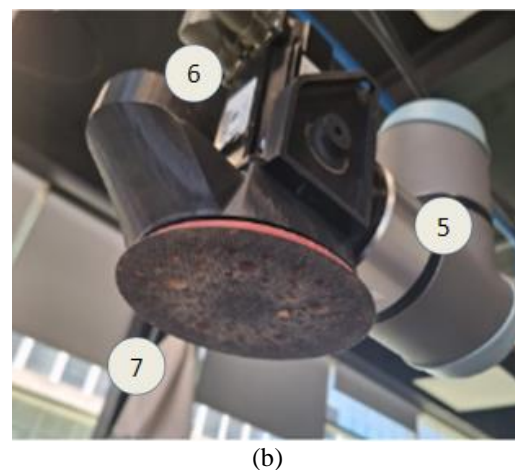
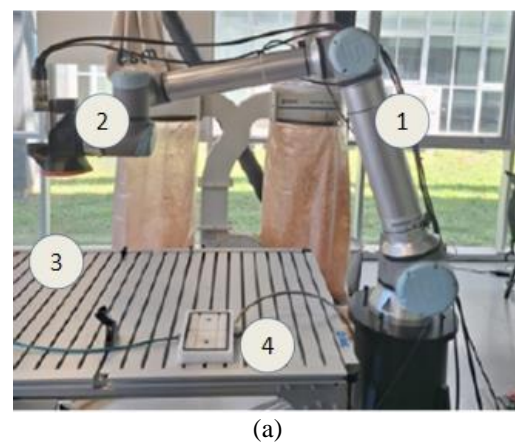


Figura 3. Robot UR10e (a) y herramienta terminal de lijado (b).

En el algoritmo, se requieren dos parámetros que deben ser definidos en el panel de control del robot, que corresponden al TCP (tool center point) y a la base del espacio de trabajo. El TCP consiste en la coordenada tridimensional que posee un punto de la herramienta (en este caso el centro de la lija) respecto a la brida del robot. Similarmente, el punto de base cumple la función de indicar el origen del espacio de trabajo. Ambos elementos consideran la ubicación y

la orientación de los mismos, y cumplen la función de facilitar la cinemática del robot, al ya no ser necesario hablar en términos de las coordenadas del robot, si no en términos de puntos de interés del usuario, en este caso el TCP y la nueva base generada.

En la Figura 4 se puede ver el proceso definido para el uso del algoritmo. Inicialmente se debe poseer un modelo CAD de la geometría en formato *.stl*. Luego el modelo puede ser cargado en el algoritmo, en donde se aplican las distintas funciones para la clasificación de superficies y generación de trayectorias. Como salida, el algoritmo entrega un *script* de programación que se envía al controlador del robot y se integra en el programa base del robot. Finalmente, el robot puede ejecutar los movimientos generados.

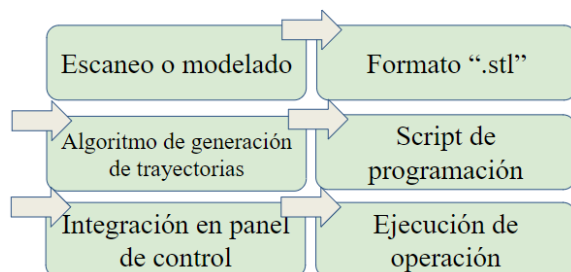


Figura 4. Diagrama de la metodología. Fuente: elaboración propia.

En el desarrollo de cada etapa del algoritmo, se considera el aspecto de modularidad, con el objetivo de facilitar la modificación de las funciones disponibles, o la adición de características en base a nuevos requerimientos.

Para simplificar el desarrollo del algoritmo, se prefieren algunas geometrías con características de interés, específicamente que posean superficies planas y curvas.

Antes de ingresar parámetros como el TCP y el punto de base del área de trabajo en el algoritmo, estos mismos son definidos en base a la herramienta y el espacio de la estación robotizada en una etapa de calibración del robot.

El algoritmo fue desarrollado utilizando el lenguaje de programación Python. La primera etapa del algoritmo consiste en la importación de la geometría al algoritmo que debe haber sido previamente digitalizado en el formato *.stl*. Utilizando la librería *numpy-stl* es posible extraer la información dimensional de la pieza mediante las coordenadas cartesianas de cada uno de los vértices de los triángulos que la conforman. En la Figura 5 se puede ver el conjunto de triángulos que definen la geometría de la pieza de trabajo utilizando la librería *matplotlib*.

La posición y orientación de la geometría respecto al eje de referencia visible en la Figura 5 dependen de la procedencia del modelo CAD. Esto representa un problema, debido a que no siempre se puede establecer un procedimiento para la generación de modelos que garantice una orientación predefinida. Por este motivo, el siguiente paso de la etapa de preparación es seleccionar la orientación de la pieza, y trasladarla hacia un punto determinado, con el fin de imitar la rotación y posición deseada de la pieza respecto al origen del espacio de trabajo del robot.

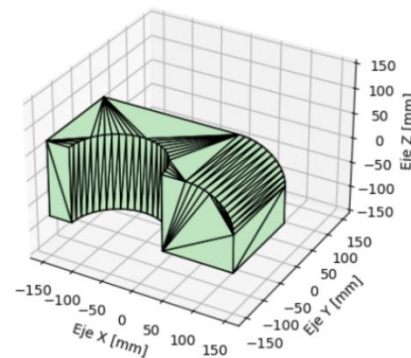


Figura 5. Geometría de ejemplo. Fuente: elaboración propia.

En la Figura 6, se puede ver el resultado de rotar y alinear la geometría con el origen del sistema cartesiano.

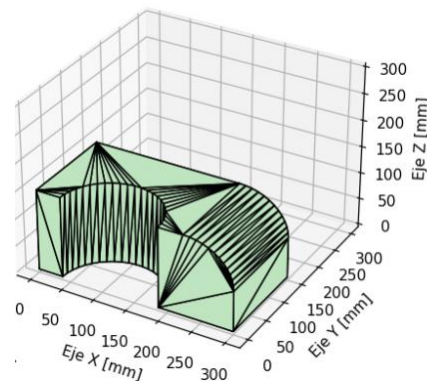


Figura 6. Geometría alineada con la referencia. Fuente: elaboración propia.

Para poder identificar las superficies por donde se desea generar trayectorias, se implementa un sistema de vistas ortogonales que se asemeja a los sistemas de vistas de software CAD. El sistema se basa en asociar una perspectiva con un vector unitario, el cual es comparado con los vectores normales de los triángulos en las superficies y determinar si es posible observar dicha superficie desde ese punto de vista. De esta manera es posible aislar la superficie de interés.

En la Figura 7, se muestra qué superficies serían consideradas con una determinada perspectiva de

observación. La condición para considerar una cara de la geometría como visible se ve en la Ecuación (1)

$$N \cdot V > t \quad (1)$$

En donde N corresponde al vector normal de un triángulo, V al vector unitario que representa la perspectiva, y t una tolerancia para evitar considerar superficies que no corresponden por imprecisiones numéricas.

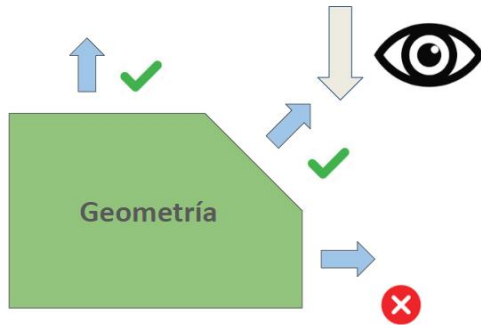


Figura 7. Explicación gráfica del sistema de vistas ortogonales. Fuente: elaboración propia.

En la Figura 8 se pueden ver la superficie que sería aislada desde una perspectiva de vista superior de la geometría de la Figura 5.

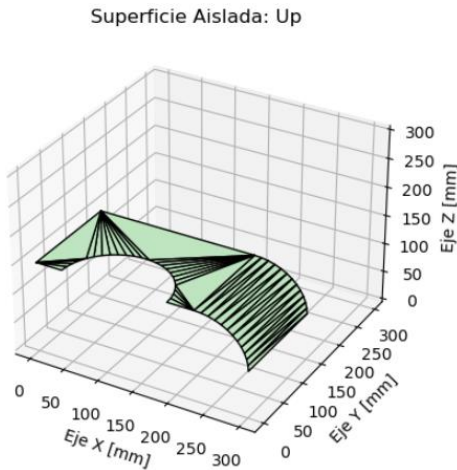


Figura 8. Superficie aislada según vista ortogonal superior. Fuente: elaboración propia.

2.1.1. Etapa de clasificación de superficies

Para la generación de puntos, y posteriormente la generación de trayectorias utilizando dichos puntos, es necesario poder identificar las características en común que poseen los triángulos de la geometría, o bien clasificar las distintas superficies para usar a favor sus características. Para este trabajo, se identifican superficies planas y curvas. Si una superficie cumple con la Ecuación 1 se clasifica como

plana, de lo contrario se clasifica como curva. En la Figura 9 se puede ver un ejemplo, en donde en una misma superficie se identifican los dos tipos de superficie antes mencionados.

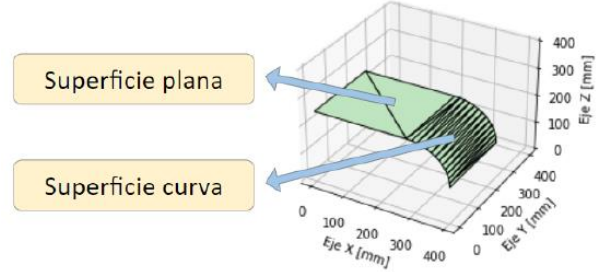


Figura 9. Ejemplo de clasificación de superficies. Fuente: elaboración propia.

Los triángulos pertenecientes a cada tipo de superficie identificada se guardan en listas diferentes, para poder ser tratadas en siguientes dos etapas.

2.1.2. Etapa de generación de puntos

La clasificación de superficies permite tratarlas considerando sus características, por lo que, por cada tipo de superficie, se propone un método distinto para generar puntos. Además, debido a que se tiene la información de la superficie y cómo esta debería ser lijada, se pueden tomar decisiones respecto al ordenamiento de estos puntos para la generación de trayectorias.

Se define como punto:

$$P = [x, y, z, rx, ry, rz]$$

En donde $[x, y, z]$ es la posición en la que se ubica el TCP, y $[rx, ry, rz]$ la orientación que debe poseer la herramienta para lijarse en la posición descrita.

Para las superficies curvas se propone el *método del centroide*, que consiste en generar puntos en el centroide geométrico de cada triángulo de la superficie. A partir de aquí, la posición del punto está descrita por la Ecuación (2), y su rotación está dada por la Ecuación (3).

$$X_i = \frac{x_{i1} + x_{i2} + x_{i3}}{3} \quad (2)$$

En donde x_{ij} representa los vértices del triángulo X_i .

$$\alpha = \cos^{-1}\left(\frac{R \cdot N}{|R| \cdot |N|}\right) \quad (3)$$

En donde N representa el vector normal del triángulo en donde se ubica el punto, R un vector de referencia definido como $R = [0, 0, 1]$ del cual se obtiene el

ángulo α . El signo de α depende de la orientación definida para el TCP durante la calibración del robot. En este caso la referencia que tendría la herramienta si se ubicara apoyada sobre el espacio de trabajo.

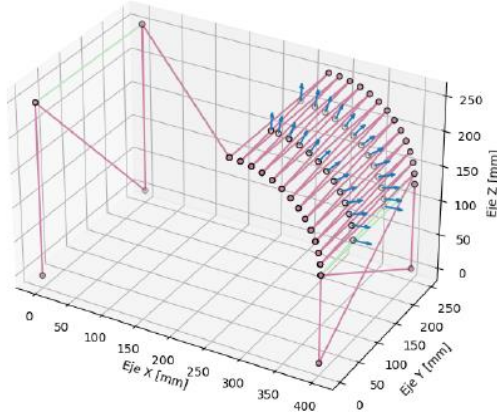


Figura 10. Puntos generados por el “Método del Centroide”. Fuente: elaboración propia.

En la Figura 10 se pueden ver los puntos generados por el *método del centroide* sobre la superficie curva.

Por otro lado, para las superficies planas se propone el *método del bounding box*, que consiste en generar puntos linealmente espaciados en un área rectangular de dimensiones tales que contenga en su totalidad a la geometría de la pieza. En la Figura 11 se puede ver un ejemplo. De esta forma se asegura que existirán puntos que están sobre la superficie.

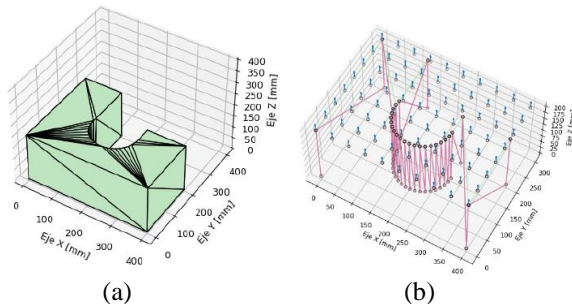


Figura 11. Puntos generados por el “Método del Bounding Box”, en donde (a) es la geometría de la pieza y (b) muestra los puntos generados. Fuente: elaboración propia.

2.1.3. Limpieza de puntos

Para identificar los puntos que no están sobre la superficie de la pieza, se plantea buscar soluciones que respondan al problema *point-in-polygon* que consiste en determinar si un punto está dentro, afuera o en el borde de un polígono [13]. La solución consiste en emitir un rayo en una dirección arbitraria y contar la cantidad de veces que este rayo interseca un borde del polígono. Si la cantidad de intersecciones corresponde a un número par, entonces el punto está afuera del polígono, de lo contrario está

dentro. En la Figura 12 se puede ver un ejemplo, y se visualiza que los rayos emitidos desde los puntos 1 y 3 intersecan un número par de veces los bordes del polígono, y se ve claramente que están fuera del mismo.

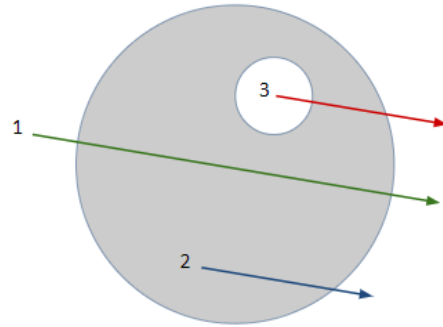


Figura 12. Explicación gráfica de la solución del problema *point-in-polygon*. Fuente: elaboración propia.

Esta solución es válida para polígonos bidimensionales y para generalizarla a tres dimensiones es necesario reemplazar el concepto de borde por triángulo. Para resolver el problema en 3 dimensiones se implementa una parte del método de Moller-Trumbore [14], utilizado en algoritmos para la generación de gráficas en computadoras. El método es capaz de identificar la ubicación en la que un rayo interseca un triángulo. Para este trabajo, la capacidad del método de comprobar si un rayo interseca o no un triángulo basta para resolver el problema del *point-in-polygon*. De esta manera se dibujan rayos en una dirección arbitraria a partir de todos los puntos, y se cuenta la cantidad de veces que estos intersecan con los triángulos de la geometría. Si el rayo de un punto interseca un número par de veces, se remueve de la lista. Utilizando esta metodología, es posible remover los puntos que no están sobre la superficie, como se puede ver en la Figura 13.

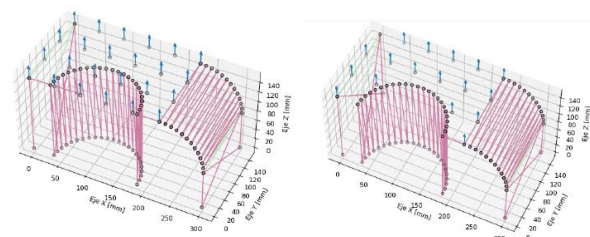


Figura 13. Remoción de puntos externos a la superficie. Fuente: elaboración propia.

Debido a las características de la operación de lijado, los puntos que coinciden con el borde de la superficie no deberían ser considerados como parte de la trayectoria, ya que al estar sólo la mita de la lija en

contacto se podrían generar marcas debido a diferencias de presión de la herramienta sobre la superficie. Por este motivo, dichos puntos eliminados con una función propia que comprueba si las coordenadas del punto en cuestión coinciden con las coordenadas de los bordes del polígono de la superficie. Finalmente, la Figura 14 muestra el resultado de eliminar los puntos que no deben ser considerados para la definición de la trayectoria.

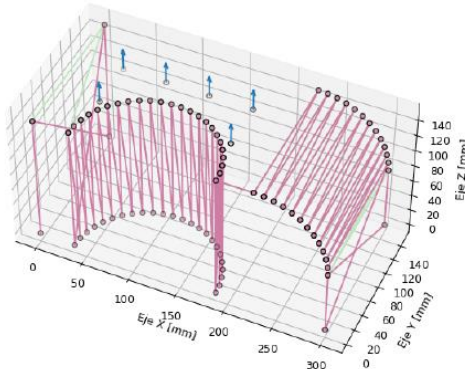


Figura 14. Puntos finales. Fuente: elaboración propia.

2.1.4. Etapa de generación de trayectorias

Debido al procedimiento utilizado para obtener puntos en cada superficie, es necesario reordenar estos puntos para que, de manera secuencial, el robot articulado ejecute una trayectoria óptima. Los criterios utilizados para la generación de trayectorias son priorizar la calidad superficial y el tiempo de ejecución de la trayectoria.

En la Figura 15 se puede ver de que manera están ordenados inicialmente los puntos generados por el *método del centroide*. Siguiendo los criterios mencionados, los puntos son ordenados como se ve en la Figura 16. Esta secuencia representa un tiempo de lijado menor, debido a que existe una menor cantidad de cambios de trayectoria, lo que además evita dejar marcas en la superficie de la pieza.

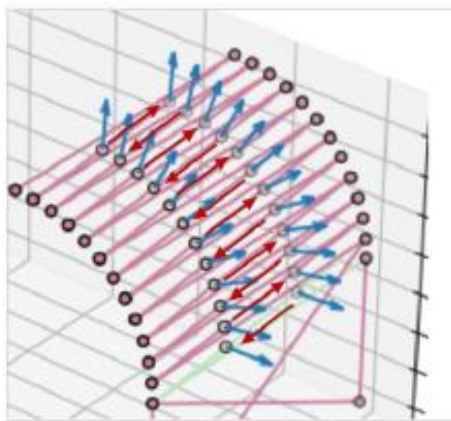


Figura 15. Orden de puntos generado por el “Método del Centroide”. Fuente: elaboración propia

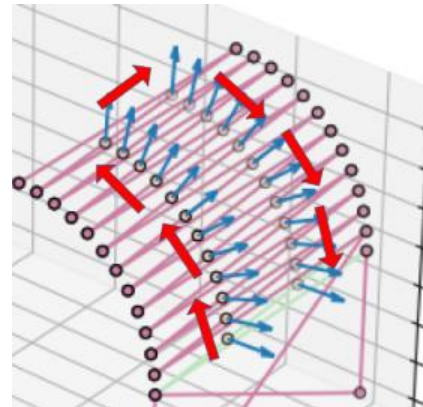


Figura 16. Orden de puntos deseado por el “Método del Centroide”. Fuente: elaboración propia.

Al igual que en el *método del centroide*, el orden de los puntos generados por el *método del bounding box* no corresponden a una secuencia válida para una trayectoria de lijado. En la Figura 17 muestra el orden adecuado para este tipo de generación de puntos, priorizando la menor cantidad de cambios de dirección.

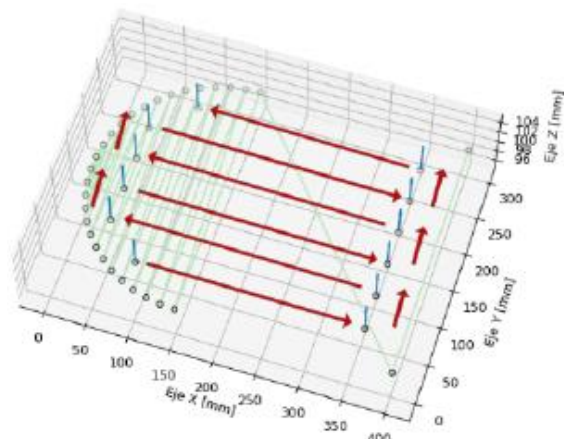


Figura 17. Orden de puntos deseado por el “Método del Bounding Box”. Fuente: elaboración propia.

Para evitar colisiones, se implementan puntos de transición, que son ubicados entre aquellos generados por uno método y el otro. Independientemente de qué tipo de superficie fue lijada primero, los puntos de transición se ordenan y ubican de forma en que se ocupe la menor cantidad de tiempo para ser recorridos. La Figura 18 muestra un ejemplo de puntos de transición para ese tipo de geometría en particular.

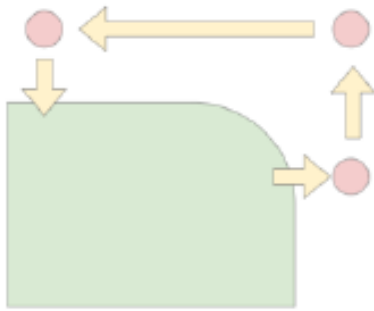


Figura 18. Puntos de transición. Fuente: elaboración propia.

2.1.5. Etapa de generación de script

Los puntos generados y ordenados en la sección anterior, se almacenan de la forma:

$$\text{Global } P = p[x,y,z,rx,ry,rz]$$

Para luego ser utilizados en la misma secuencia por la función:

$$\text{movel}(\text{pose_trans}(\text{base}, P), a=\text{acc}, v=\text{vel})$$

En donde “movel” determina el tipo de movimiento, en este caso lineal entre puntos. La función pose_trans transforma linealmente a P desde el punto base. Finalmente, acc y vel son la aceleración y velocidad de movimiento del robot.

El script generado por esta etapa del algoritmo es integrado en el panel de control del robot, en donde forma parte de un programa base del mismo robot.

Para validar los algoritmos desarrollados, se realizaron ensayos en el robot articulado UR10e. El control del robot posee un programa simple, que lee los puntos generados y ejecuta las trayectorias.

3. Resultados

Para hacer uso del algoritmo y modificar sus parámetros, se programa una interfaz de usuario. De esta manera no es necesario modificar el código del algoritmo para generar scripts de programación. La interfaz permite cargar una geometría, seleccionar su orientación, generar puntos para las superficies seleccionadas, ordenar la secuencia, simular la trayectoria con una animación y finalmente generar un script de programación que se guarda en un fichero especial. La interfaz de usuario puede verse en la Figura 19.

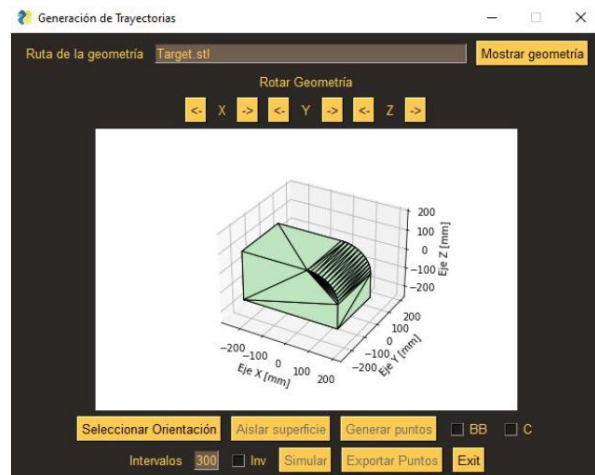


Figura 19. Interfaz de usuario. Fuente: elaboración propia.

El procedimiento para utilizar el algoritmo en conjunto a la interfaz de usuario se describe a continuación:

1. Configurar el robot industrial con las variables que definen el área de trabajo y la herramienta.
2. Conseguir un modelo CAD de la geometría y exportarlo en el formato *.stl*.
3. Ubicar el archivo en la carpeta designada para ser leído por la aplicación.
4. Cargar el archivo en la aplicación y asegurarse que coincide con la pieza que se desea lijar.
5. Utilizando la aplicación, rotar la geometría hasta obtener la orientación deseada y presionar el botón Seleccionar Orientación.
6. Alinear la geometría como indica la aplicación en la mesa de trabajo.
7. Aislar la superficie y generar los puntos utilizando los métodos disponibles.
8. Seleccionar parámetros de optimización y de simulación.
9. Corroborar que la simulación muestra una trayectoria deseada y si no es así, modificar los parámetros de optimización y repetir.
10. Exportar los puntos como un script para extraerlo en algún dispositivo de almacenamiento portátil.
11. Cargar el script en el programa principal del robot usando en panel de control.
12. Utilizando el panel, ejecutar el programa.

Se realizaron ensayos de lijado en la pieza de prueba de la Figura 20. Las trayectorias fueron generadas y se ejecutaron sobre la pieza de manera exitosa.



Figura 20. Pieza de prueba para ensayos (Fabricada en madera).

4. Conclusiones

Tras desarrollar las distintas funcionalidades del algoritmo, es posible validar hasta cierto punto que sólo utilizando la geometría de una pieza es posible generar trayectorias de manera automática para el lijado de sus superficies. Se ha presentado un algoritmo y una metodología que permite que el robot articulado ejecute una trayectoria de lijado de forma automática.

La naturaleza modular con la que fue programado el algoritmo propicia la modificación o integración de nuevas funciones. Este es un aspecto relevante, pues con nuevas funcionalidades el algoritmo toma robustez para casos más complejos y es posible operar nuevas geometrías, mientras sea posible clasificar superficies mediante un criterio, es posible crear nuevos métodos de generación de puntos y de optimización de trayectorias.

El problema de las colisiones no fue resuelto de una manera general, y requiere de la verificación del programador, por lo que parte de un trabajo futuro es considerar la geometría del robot articulado para la generación de puntos, que asegure que la geometría del robot, y la posición y orientación del mismo no interfieran con la pieza de producción ni el espacio de trabajo.

Las características del algoritmo indican que, con la modificación de algunos parámetros en los métodos de generación de puntos y las trayectorias correspondientes, sería posible considerar otras operaciones, como pintado, deposición de adhesivos, o incluso otras operaciones de mecanizado sin realizar modificaciones importantes al funcionamiento base del algoritmo.

5. Agradecimientos

Los autores agradecen a la Agencia Nacional de Investigación y Desarrollo (Chile) por el financiamiento de esta investigación a través del proyecto FONDEF IT2110069 “Desarrollo y validación de una célula inteligente de lijado y pulido robotizado para entornos de manufactura avanzada”.

6. Referencias

- [1] IFR. The impact of robots on productivity, employment and jobs. (2017). Disponible: https://ifr.org/img/office/IFR_The_Impact_of_Robots_on_Employment.pdf
- [2] STEP. Problemas de la robotización de la industria. (2023). Disponible: <https://www.machine-controller.org/info/four-technical-difficulties-in-industrial-robo-79258340.html>.
- [3] Pan, Z., Polden, J., Larkin, N., Duin, S. V., y Norrish, J. (2012, 4). Recent progress on programming methods for industrial robots (Vol. 28). doi: 10.1016/j.rcim.2011.08.004
- [4] Yang, L., Liu, Y., Peng, J., and Liang, Z. (2020). A novel system for off-line 3d seam extraction and path planning based on point cloud segmentation for arc welding robot. *Robotics and Computer-Integrated Manufacturing*, 64.
- [5] Nieto Bastida, S., & Lin, C. Y. (2023). Autonomous Trajectory Planning for Spray Painting on Complex Surfaces Based on a Point Cloud Model. *Sensors*, 23(24). <https://doi.org/10.3390/s23249634>
- [6] Wang, M., Song, Y., and Wang, P. (2024). A novel grinding path generation method for removing the parting line of large casting. *International Journal of Advanced Manufacturing Technology*, 131:201–209.
- [7] Meng, Y., Jiang, Y., Li, Y., Pang, G., and Tong, Q. (2024). Research on point cloud processing and grinding trajectory planning of steel helmet based on 3d scanner. *IEEE Access*, 12:3085–3097.
- [8] Sheng, W., X. N. S. M. . C. Y. (2001). Cad-guided robot motion planning. *Industrial Robot*, 28(2), 143–151.
- [9] Neto, P. and Mendes, N. (2013). Direct off-line robot programming via a common cad package. *Robotics and Autonomous Systems*, 61:896–910.

[10] López-Arrabal, A., Álvaro Guzmán-Bautista, Sol órzano-Requejo, W., Franco-Martínez, F., and Villaverde, M. (2024). Axisymmetric non-planar slicing and path planning strategy for robot-based additive manufacturing. *Materials and Design*, 241.

[11] Afroz, A. S., Inglese, F., Stefanini, C., and Milazzo, M. (2021). Stlprocess: A .stl-based preprocessor for robot path planning in manufacturing and quality control processes. *SoftwareX*, 15.

[12] Nikolaj W., Henrik G. (2020) Robot Path Generation for Curve Manufacturing Processes from Un-structured CAD Data: An Off-line Programming Method. *ISR 2020 ; 52th International Symposium on Robotics*. Odense, Denmark.

[13] Kularathne D., Jayarathne L. (2018) Point in Polygon Determination Algorithm for 2-D Vector Graphics Applications. *National Information Technology Conference, NITC 2018*, art. no. 8550057 doi: 10.1109/NITC.2018.8550057

[14] T. Moller, B. T. (1997). Fast, minimum storage ray-triangle intersection.