ALGORITMO DE GENERACIÓN DE TRAYECTORIAS EN EL INTERIOR DE CHAPAS PARA LA SUBSANACIÓN DE DEFECTOS

Álvaro Fernández García, Sara Roos Hoefgeest Toribio, Ignacio Álvarez García, Rafael Corsino González de los Reyes

Área de Ingeniería de Sistemas y Automática. Universidad de Oviedo
Campus Universitario de Gijón, s/n.
{fernandezgalvaro, roossara, ialvarez, rcgonzalez}@uniovi.es

Resumen

El presente documento describe un algoritmo de generación de trayectorias para un robot móvil encargado de sanear chapas de acero. Las trayectorias generadas deben asegurar la cobertura total de las superficies poligonales que delimitan los defectos con la herramienta acoplada al robot. Además, el robot deberá resolver la tarea sin abandonar el interior de la chapa. Para resolver el problema, la chapa se divide en distintas zonas de trabajo derivadas de las orientaciones seguras que permiten al robot reparar sin abandonar la superficie de la chapa.

Los defectos se recibirán en forma de polígonos que serán, en primer lugar, divididos de acuerdo con las zonas de trabajo y, a continuación, descompuestos en formas más simples teniendo en cuenta la orientación de trabajo de cada zona. El recorrido completo del defecto podrá realizarse calculando trayectorias paralelas para cada uno de los polígonos simples que lo componen. La trayectoria así calculada corresponde a la seguida por la herramienta, por lo que para calcular la trayectoria del robot bastará con aplicar una traslación.

Palabras clave: Robótica, Planificación de trayectorias, Geometría computacional, Descomposición trapezoidal, ROS

1 INTRODUCCIÓN

En una industria cada vez más automatizada los robots móviles desempeñan cada vez más tareas. Una de estas posibles aplicaciones es la inspección y subsanación de chapas.

Frente a otras alternativas, un robot móvil tiene unas dimensiones y un peso moderados que permite su sencillo traslado a otros lugares de trabajo y facilita su posible colaboración con operarios humanos o con otras máquinas. Otra ventaja de la opción escogida es su versatilidad, siendo capaz de reparar desde hojas de acero con una superficie no mucho mayor que la del propio robot hasta grandes chapas de decenas de metros, como las que pueden encontrarse en la industria siderúrgica. El empleo de, por ejemplo, un robot cartesiano para automatizar la inspección y reparación de estas últimas conllevaría la construcción de una gran estructura.

Se supondrá, por tanto, la existencia de un robot omnidireccional dotado de una herramienta en una posición fija respecto al centro del robot. En este planteamiento el actuador se encarga de subsanar un defecto detectado en la chapa, aunque el algoritmo presentado resulta fácilmente adaptable a otras labores tales como inspección, pintado, limpieza, etc.

En la aplicación propuesta el robot se encontrará trabajando sobre una superficie rectangular plana, y que puede estar elevada una distancia considerable respecto al suelo. Si el robot se sale de la superficie de trabajo, podría resultar peligroso y provocar graves daños en el robot e incluso en operarios humanos.

Para resolver el problema, la chapa se divide en 4 zonas, asociada cada una a una posible orientación del robot paralela a los ejes de la chapa. En cada una de las zonas, el robot podrá, manteniendo la orientación asociada, realizar las pertinentes reparaciones sin correr el riesgo de abandonar el espacio de trabajo.

Los defectos, recibidos en forma de polígono, se dividirán por tanto según las zonas de trabajo. De esta manera, cada división se recorrerá con una única orientación. Con el fin de facilitar la reparación completa de los defectos se llevará a cabo también una descomposición trapezoidal que da lugar a formas simples de reparación más sencilla.

1.1 ESTADO DEL ARTE

Tras las investigaciones realizadas no se ha encontrado bibliografía referida a este conjunto de problemas combinados.

Sí existe, sin embargo, una familia de algoritmos dedicados al objetivo de recorrer la totalidad del área de una superficie. Este tipo de estrategias, denominadas comúnmente como *Coverage Path Planning* se pueden dividir, según el tipo de estrategias a la hora de dividir la superficie, en: algoritmos de división en malla y algoritmos de descomposición. Los primeros dividen la totalidad de la superficie en grandes áreas que se pueden recorrer fácilmente mientras que los segundos descomponen el área en pequeñas celdas idénticas que se van visitando individualmente.

Entre los algoritmos de descomposición en mallas puede destacarse el método Spiral-STC [5] que cubre completamente el área realizando un recorrido espiral. Otro método de interés es propuesto por S. X. Yang y C. Luo [8]. Este establece una malla en la que cada celda se representa por una neurona. Cada neurona atrae al robot si está sin inspeccionar y lo repele si se trata de un obstáculo.

Existe un número considerable de estrategias de descomposición del área. Estrategias como la descomposición trapezoidal [2] y la descomposición bustrófedon [4] se limitan a la división del área con obstáculos poligonales. La descomposición Morse [1] sí contempla objetos no poligonales, pero tiene problemas con obstáculos paralelos al avance del robot. La estrategia de descomposición en *slices* [7] por su parte permite lidiar tanto con obstáculos tanto rectilíneos como curvos.

El artículo de Galceran y Carreras [6] realiza un estudio general sobre el *Coverage Path Planning* y puede ser un buen punto de partida si se desea conocer más sobre estos aspectos. En el capítulo 6 de "Principles of Robot Motion" [3] escrito por H. Choset puede encontrarse un texto dedicado a explicar brevemente diversas estrategias de descomposición en celdas.

1.2 MÉTODO PROPUESTO

El método propuesto recibe como entrada los polígonos que definen la forma de los defectos detectados en la superficie de la chapa.

Para poder acometer la reparación de estos será necesario actuar con la herramienta en toda la superficie del desperfecto. Para ello, los defectos se dividen en dos etapas. En la primera se dividen en base a las distintas zonas de trabajo. Estas zonas se definen en base a las zonas seguras para el robot de manera que no abandone la chapa.

Una vez el defecto se ha dividido en base a las áreas de trabajo las formas de los polígonos pueden continuar siendo complejas para realizar su recorrido completo. Con el fin de facilitar esta tarea se realizará una descomposición trapezoidal que dividirá el polígono en formas simples como trapecios y triángulos. A partir de dichas formas simples se pueden calcular de manera sencilla las trayectorias que lo cubren por completo como trayectorias paralelas a sus segmentos.

La coordinación e implementación de los algoritmos mencionados se realiza en el entorno ROS.

2 METODOLOGÍA

2.1 DIVISIÓN DE LA CHAPA EN ZONAS DE TRABAJO

Se supondrá que la chapa es un rectángulo cuyo centro geométrico está situado en el origen del sistema de referencia global y sus lados están alineados con los ejes, de modo que lado más largo está orientado en la dirección del eje X y el más corto en la dirección del eje Y. La chapa mide $2L_c$ de ancho y $2H_c$ de largo. Tanto L_c como H_c son valores estrictamente positivos.

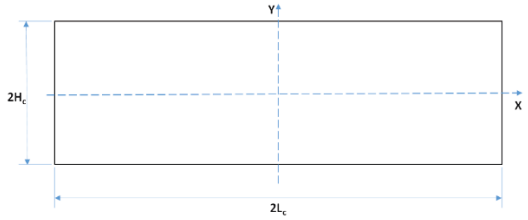

Figura 1. Modelo de la chapa sobre la que debe desplazarse el robot

El robot se establece como un robot omnidireccional (por ejemplo, con ruedas tipo *mecanum*), con una distancia entre los ejes de las ruedas de $2L_r$ y una distancia entre la parte exterior de las ruedas de $2H_r$. Tanto L_r como H_r son valores estrictamente positivos. El sistema de coordenadas asociado al robot se encuentra situado en el centro geométrico del rectángulo formado por las ruedas. El eje X es perpendicular a la dirección de los ejes de las ruedas y el eje Y es paralelo. La posición del origen del sistema de coordenadas del robot respecto al sistema de coordenadas de la chapa es (x_r, y_r). El centro de la herramienta se encuentra en la posición (X_d, Y_d) con respecto al sistema de referencia del robot. X_d e Y_d pueden ser valores reales cualesquiera.

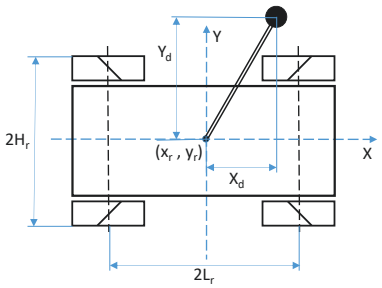

Figura 2. Modelo del robot y de la herramienta.

Se considerará que el robot estará siempre alineado con los ejes de la chapa o con una orientación múltiplo de 90 grados.

El objetivo es lograr que el robot pueda reparar la chapa sin necesidad de salir de la misma. Para ello, se comenzará estudiando cuál es la zona de la chapa sobre la que puede situarse la herramienta sin que el robot se salga de la misma para una orientación del robot de 0 grados.

Se definirá un rectángulo R como la tupla (x_1, y_1, x_2, y_2), de modo que los puntos en el interior del rectángulo son los puntos que cumplen $\{(x,y)|x_1 \leq x \leq x_2 \ \wedge y_1 \leq y \leq y_2\}$. Se basará el razonamiento en operaciones suma y diferencia de Minkowsky, muy utilizadas en morfología.

El primer paso consistirá en determinar qué puntos de la chapa pueden ser ocupados por el robot sin que éste se salga de la misma. Esta operación puede calcularse como la diferencia de Minkowski entre el conjunto de puntos que forman la chapa:

$$\begin{gathered} C = (-L_c, -H_c, L_c, H_c) = \\ = \{(x,y)| -L_c \leq x \leq L_c \ \wedge -H_c \leq y \leq H_c; \\ L_c > 0 \wedge H_c > 0\} \end{gathered} \quad (1)$$

y el conjunto de puntos comprendidos en el rectángulo que definen las ruedas del robot:

$$\begin{gathered} R = (-L_r, -H_r, L_r, H_r) = \\ = \{(x,y)| -L_r \leq x \leq L_r \ \wedge -H_r \leq y \leq H_r; \\ H_r > 0 \ \wedge \ L_r > 0\} \end{gathered} \quad (2)$$

El resultado, es un rectángulo con el mismo centro que la chapa. El ancho y alto se ven reducidos, por cada lado, en L_r y H_r respectivamente, Figura 3.

$$A_1 = (-(L_c - L_r), -(H_c - H_r), L_c - L_r, H_c - H_r) \quad (3)$$

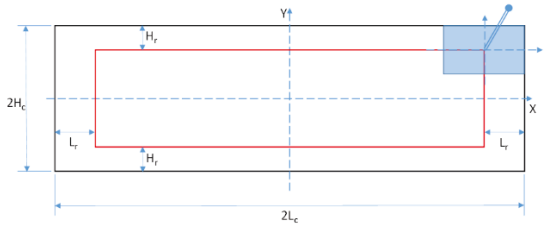

Figura 3. El rectángulo rojo representa la región sobre la que se puede desplazar el robot sin que se salga de la chapa.

La condición necesaria para que exista esta solución es que el robot sea más pequeño que la chapa. Si coinciden en una de las dimensiones y en la otra el robot es más pequeño, el rectángulo A_1 degenera en una recta. En consecuencia, sólo se podrá mover en la dirección indicada por la recta.

El siguiente paso es determinar qué conjunto de puntos debe ocupar el robot para que la herramienta esté dentro de la chapa. Para ello se calcula la diferencia de Minkowsky entre la chapa y el conjunto formado por la posición de la herramienta:

$$D_0 = \{(X_d, Y_d)\} \quad (4)$$

El resultado es un rectángulo del mismo tamaño que la chapa, pero desplazado una distancia $(-X_d, -Y_d)$, ya que los ejes del robot son paralelos a los de la chapa, Figura 4.

$$A_2 = (-(L_c + X_d), -(H_c + Y_d), L_c - X_d, H_c - Y_d) \quad (5)$$

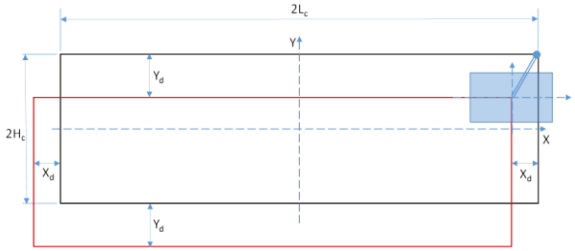

Figura 4. El rectángulo rojo representa la región tal que, al desplazarse el robot dentro de la misma, la herramienta no se saldría la chapa.

En consecuencia, el conjunto de puntos de la chapa que puede ocupar el robot asegurando que tanto la herramienta como el robot están dentro de la chapa es la intersección de los rectángulos A_1 y A_2, Figura 5.

$$\begin{gathered} A_3 = (x_1, y_1, x_2, y_2) \\ x_1 = \max\big(-(L_c - L_r), -(L_c + X_d)\big) = -(L_c - \max(L_r, -X_d)) \\ y_1 = \max\big(-(H_c - H_r), -(H_c + Y_d)\big) = -(H_c - \max(H_r, -Y_d)) \\ x_2 = \min(L_c - L_r, L_c - X_d) = L_c - \max(L_r, X_d) \\ y_2 = \min(H_c - H_r, H_c - Y_d) = H_c - \max(H_r, Y_d) \end{gathered} \quad (6)$$

Donde se ha tenido en cuenta que dados dos números reales a y b, resulta que:

$$max(a,b) = \frac{1}{2}(a+b+|a-b|)$$
$$min(a,b) = \frac{1}{2}(a+b-|a-b|) \quad (7)$$

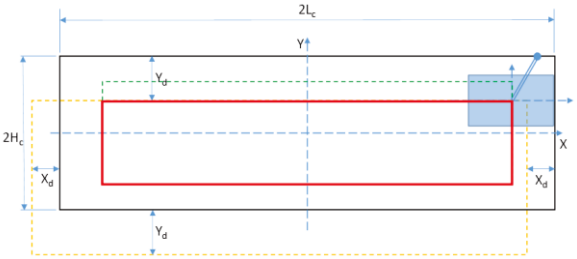

Figura 5. Región por la que puede desplazarse el robot sin que se salga de la chapa ni el robot ni la herramienta.

La zona cubierta por la herramienta cuando el robot se mueve por la región A_3 es igual a la suma de Minkowski de dicha región con la región que define la herramienta (D). El resultado es el rectángulo A_3 trasladado la cantidad (X_d, Y_d), Figura 6.

$$\begin{gathered} Z_0 = (x_1, y_1, x_2, y_2) \\ x_1 = -(L_c - (\max(L_r, -X_d) + X_d)) \\ y_1 = -(H_c - (\max(H_r, -Y_d) + Y_d)) \\ x_2 = L_c - \max(L_r, X_d) + X_d \\ y_2 = H_c - \max(H_r, Y_d) + Y_d \end{gathered} \quad (8)$$

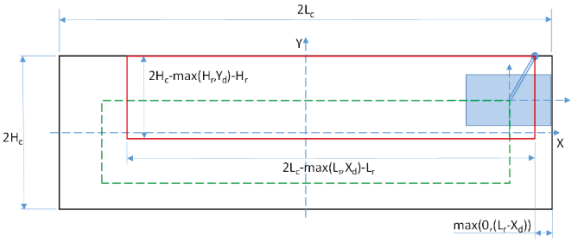

Figura 6. El rectángulo rojo indica la región de la chapa que puede alcanzar extremo de la herramienta cuando el robot se mueva por dentro de la chapa con una orientación coincidente con la del sistema de coordenadas de la chapa.

Al repetir el análisis para el caso en el que el robot ha girado 90 grados, es necesario tener en cuenta que el conjunto de puntos que definen el robot y la herramienta ha cambiado como consecuencia del cambio en los ejes:

$$\begin{gathered} R_{90} = (-H_r, -L_r, H_r, L_r) = \\ = \{(x,y) | -H_r \le x \le H_r \ \wedge -L_r \le y \le L_r \} \\ D_{90} = \{(Y_d, -X_d)\} \end{gathered} \quad (9)$$

En consecuencia, la región por la que se puede mover el robot con esta nueva orientación sin que la herramienta ni el robot se salgan de la chapa es:

$$\begin{gathered} Z_{90} = (x_1, y_1, x_2, y_2) \\ x_1 = -(L_c - (\max(H_r, Y_d) - Y_d)) \\ y_1 = -(H_c - (\max(L_r, -X_d) + X_d)) \\ x_2 = L_c - \max(H_r, -Y_d) - Y_d \\ y_2 = H_c - \max(L_r, X_d) + X_d \end{gathered} \quad (10)$$

Al rotar 180 grados, la solución es equivalente a la obtenida para una orientación del robot de 0 grados, pero con la herramienta en la posición $(-X_d, -Y_d)$, por lo que:

$$\begin{gathered} Z_{180} = (x_1, y_1, x_2, y_2) \\ x_1 = -(L_c - (\max(L_r, X_d) - X_d)) \\ y_1 = -(H_c - (\max(H_r, Y_d) - Y_d)) \\ x_2 = L_c - \max(L_r, -X_d) - X_d \\ y_2 = H_c - \max(H_r, -Y_d) - Y_d \end{gathered} \quad (11)$$

Por último, y siguiendo el mismo tipo de razonamiento, para el caso de una rotación de 270 grados resulta:

$$\begin{gathered} Z_{270} = (x_1, y_1, x_2, y_2) \\ x_1 = -(L_c - (\max(H_r, -Y_d) + Y_d)) \\ y_1 = -(H_c - (\max(L_r, X_d) - X_d)) \\ x_2 = L_c - \max(H_r, Y_d) + Y_d \\ y_2 = H_c - \max(L_r, -X_d) - X_d \end{gathered} \quad (12)$$

2.2 DESCOMPOSICIÓN DE LOS DEFECTOS EN FORMAS SIMPLES

Para poder realizar el recorrido completo de los polígonos se pretende realizar una división de estos en figuras geométricas más simples. Esta segmentación debe completarse con la división de la chapa en zonas de trabajo, de manera que cada uno de los polígonos se encuentre únicamente en una de las zonas anteriormente mencionadas y, por tanto, se repare al completo pudiendo mantener la misma orientación, Figura 7.

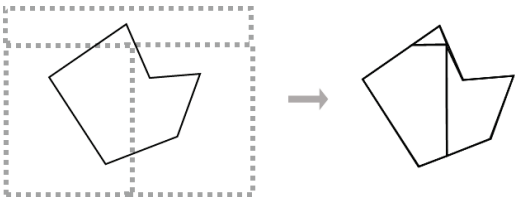

Figura 7. Representación de la división del defecto empleando las zonas de trabajo.

Para ello se busca la intersección entre el defecto y cada una de las 4 zonas de trabajo. Una vez dividido, los polígonos resultantes se ven sometidos a una descomposición en celdas que resulta en las formas simples deseadas, Figura 8.

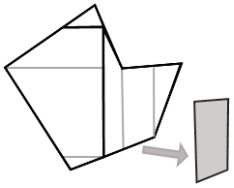

Figura 8. Ejemplo de descomposición trapezoidal de los polígonos divididos por zonas.

La descomposición en celdas consiste en la división del espacio como la unión de un conjunto de estructuras simples que reciben el nombre de celdas. Estos espacios podrán ser recorridos de manera mucho más sencilla, simplificando su reparación.

En este caso, se ha escogido la descomposición trapezoidal por ajustarse bien a nuestros requisitos y por su sencillez a la hora de ser implementado en código frente a otras estrategias.

En este método, introducido por B. Chazelle [2], el espacio libre y los posibles obstáculos dentro de él se representan como polígonos en un sistema de coordenadas XY, Figura 9. Para realizar la descomposición se comprueban las intersecciones que tendría una recta de valor x constante en cada uno de los vértices. Un ejemplo de este proceso puede verse en la Figura 9 sobre el vértice destacado en rojo.

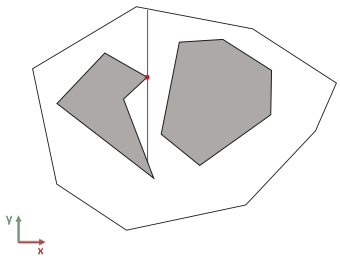

Figura 9. Representación de los polígonos. Figura adaptada de [3].

Los nuevos vértices procedentes de este proceso y los vértices originales se usarán para conformar los trapecios de la descomposición, Figura 10.

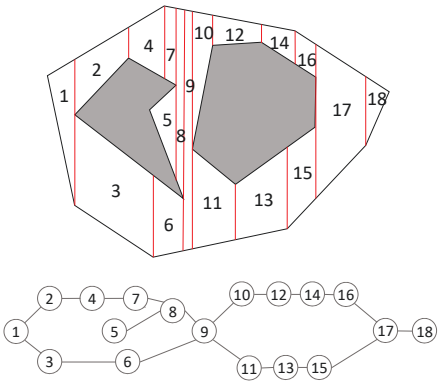

Figura 10. Ejemplo de descomposición trapezoidal. Figura adaptada de [3].

2.3 CÁLCULO DE LAS TRAYECTORIAS EN CADA TRAPECIO

Una vez realizada la descomposición en trapecios el calculo de la trayectoria que recorre toda su área se simplifica. Ante este tipo de figuras, para asegurarse de que se cubre toda la zona basta con realizar trayectorias paralelas separadas por una distancia menor que la mitad del grosor que la herramienta es capaz de reparar, Figura 11. Estas trayectorias deberán tener una separación máxima equivalente al grosor de la herramienta y su inicio y su final se hallará buscando la intersección con los lados no paralelos.

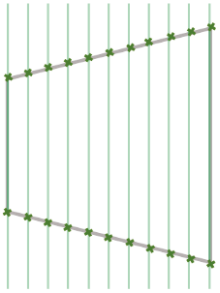

Figura 11. Trayectorias paralelas que garantizan el completo recorrido de una forma trapezoidal.

Es importante tener en cuenta que estas trayectorias generadas serán las que deberá seguir la herramienta actuadora y, por tanto, para obtener las del propio robot se deberá llevar a cabo una transformación sobre las mismas. Al poder desplazarse el robot de manera omnidireccional sin necesidad de rotar sobre sí mismo esta transformación se simplifica en una mera traslación de las posiciones objetivo.

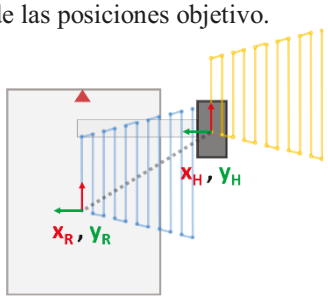

Figura 12: Representación, en ámbar, de la trayectoria para la herramienta calculada sobre el defecto y, en añil, de la trayectoria trasladada que deberá ejecutar el robot.

3 EXPERIMENTOS Y RESULTADOS

3.1 IMPLEMENTACIÓN EN ROS

La implementación de los algoritmos y la simulación de las diversas pruebas se ha realizado en el entorno ROS. En función de las diferentes necesidades se han aprovechado las distintas posibilidades de comunicación que ofrece. En primer lugar, se han desarrollado mensajes[10] sirviendo como estructura de datos básica. Estos mensajes junto a otros serán gestionados por distintas acciones[9] y servicios[12]. Los servicios se reservarán para contener información del problema mientras que las acciones a la ejecución de cálculos o trayectorias, pudiendo estos ser interrumpidos durante su desarrollo.

Para almacenar la información de los defectos y sus descomposiciones se han creado 3 tipos de mensajes que se relacionan de forma jerárquica. De esta manera,

Figura 13, un mensaje del tipo *defecto* contendrá los polígonos que resultan de dividir el polígono con las zonas de trabajo y cada uno de ellos incluirá a todos los trapecios que lo componen.

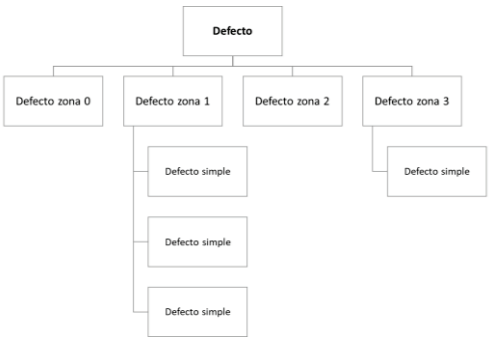

Figura 13. Ejemplo de estructura de mensajes derivados de un defecto.

Además de los vértices del polígono en cuestión el mensaje incluirá otra información de utilidad como la dirección de reparación, la zona a la que pertenecen y si han sido ya reparados o no.

La información de entrada al problema como el polígono que define la superficie del defecto y las dimensiones de la chapa y del robot se guardarán en un servicio (*FCPP_in*). Se desarrolla una acción para tomar esos datos y calcular los resultados de la descomposición que se guardan en un nuevo servicio (*FCPP_out*), Figura 14. De esta manera, se podrá disponer de la información accediendo a los correspondientes servicios sin necesidad de que se publiquen mensajes de manera cíclica.

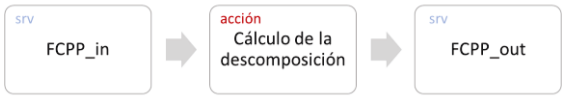

Figura 14. Relación entre los servicios y acciones que se encargan del problema de la descomposición.

Por último, se dispondrá de una acción que permite obtener información de los servicios ya comentados, calcular y ejecutar las trayectorias necesarias para la cobertura. Internamente, esta acción llama a otras dos acciones subordinadas para ejecutar el movimiento y la rotación del robot.

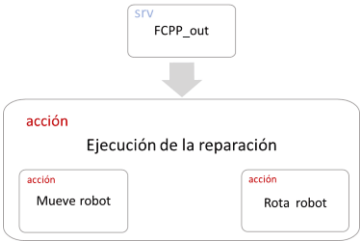

Figura 15. Relación de la acción de reparación con el servicio y las acciones de las que depende.

Las acciones encargadas de mover al robot se basan en un controlador de programación propia. Este envía los correspondientes vectores de velocidad al robot para alcanzar las posiciones objetivo sin ejecutar algoritmos de navegación y, en este caso, tomando directamente del simulador la posición del robot.

De esta manera, se dispondrá de un programa capaz de recibir los polígonos que definen los defectos presentes en una chapa, descomponerlos en formas más simples y calcular y realizar las trayectorias seguras que garantizan la reparación completa del defecto.

La estructura general del entorno creado puede verse en la Figura 16. Como preparación para el sistema será necesario lanzar en primera instancia los nodos servidores que contienen los algoritmos y la información del problema. En segundo lugar, se deberá ejecutar un nodo que tome con la información del problema para poder calcular la descomposición y guardar el resultado en el servicio correspondiente. Por último, el cálculo y la ejecución de las trayectorias para cada uno de los defectos se realizará llamando a la acción con el defecto a reparar como objetivo.

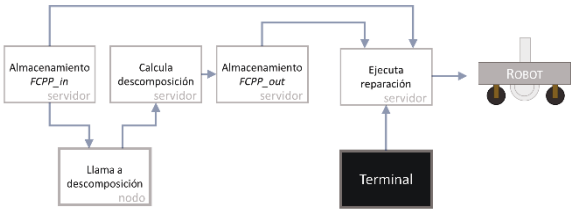

Figura 16. Resumen general de la estructura de nodos en ROS.

El modelo de robot omnidireccional seleccionado para las simulaciones ha sido el SummitXL de Robotnik Automation[11]. Este se encuentra disponible en los paquetes *summit_xl_sim* [14] y *summit_xl_common* [13].

3.2 RESULTADOS

La correcta implementación de los algoritmos se ha comprobado en diversas simulaciones resultando todas ellas satisfactorias.

La descomposición de los polígonos en formas simples ha resultado exitosa siendo capaz de descomponer correctamente polígonos cóncavos y formados por numerosas aristas. Dos ejemplos de descomposición del algoritmo pueden apreciarse en la Figura 17.

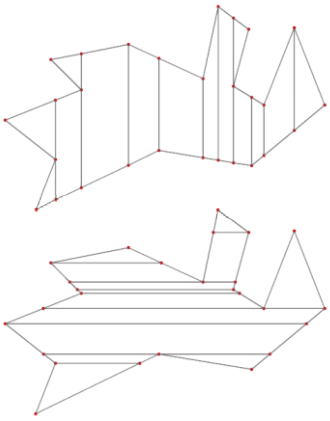

Figura 17. Resultado de la descomposición trapezoidal de un polígono en las dos posibles orientaciones.

Como puede verse en la Figura 18, la correcta descomposición de los polígonos ha podido trasladarse al entorno ROS. En la figura pueden apreciarse también la presencia del sistema de coordenadas de la chapa y los del robot y actuador acoplado.

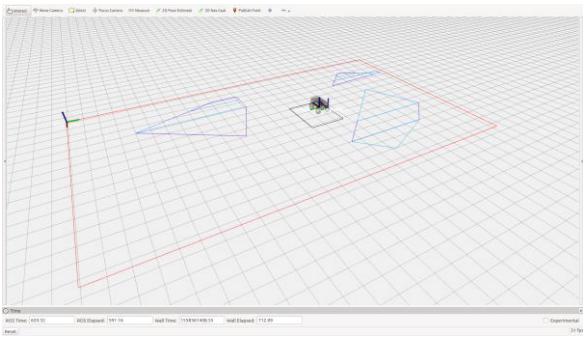

Figura 18. Visualización del escenario simulado incluyendo la chapa, el robot y los defectos descompuestos.

Por último, el cálculo, la coordinación y la ejecución de las trayectorias ha demostrado en distintas simulaciones ser capaz de cubrir la totalidad del área de los defectos para diferentes variables dimensionales del robot y de la chapa, Figura 19.

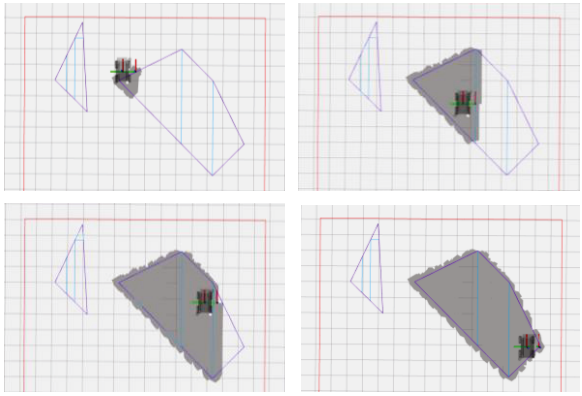

Figura 19. En sentido horario: Evolución del cubrimiento de una chapa representando en gris la trayectoria seguida por la herramienta.

4 TRABAJO FUTURO

Ante chapas de grandes dimensiones puede ser interesante la colaboración con operarios humanos. Para alcanzar este propósito será necesario integrar en el robot capas de seguridad que permitan, en cualquier momento, evitar situaciones de peligro.

En cuanto a mejoras de los métodos expuestos se encuentra contemplar la posibilidad de defectos cuya forma poligonal incluya huecos en los que no sea necesaria la reparación. Para poder enfrentar este tipo de problemas deberían hacerse modificaciones en el algoritmo de descomposición en formas simples.

De cara a optimizar el proceso de reparación se podrá trabajar en métodos que permitan establecer una orientación de reparación óptima no limitándose a las paralelas a los bordes de la chapa. Para ello, deberán efectuarse ampliaciones en el método de cálculo de las zonas de trabajo e implementar un algoritmo para establecer la dirección de reparación preferente de un defecto. También existe margen de mejora a la hora de determinar el orden de reparación de los diferentes trapecios pudiendo implementarse algún algoritmo que calcule el recorrido global óptimo.

Otra posible mejorar podría darse si el robot requiere estar conectado a líneas de suministro (de energía, aire comprimido, etc.) debido a limitaciones en el tamaño o la capacidad de carga de este. En este supuesto sería interesante incluir en las trayectorias generadas las limitaciones que esto puede implicar de cara a la movilidad.

https://doi.org/10.17979/spudc.9788497497169.702

English summary

PATH PLANNING ALGORITHM FOR DAMAGE REPAIR INSIDE A STEEL SHEET

Abstract

This paper describes a path planning algorithm to allow a mobile robot to repair surface defects of a steel sheet. The resulting trajectories ensure that the polygonal surface that encloses the defect is completely covered by the repairing tool carried by the robot. In addition, the robot should work within the surface of the metal sheet.

Defects will be described as polygonal shapes. First, they will be divided according to different working areas. Then, each resulting shape will be divided again into simpler triangular and trapezoidal shapes. To guarantee that the defect is completely covered, the tool trajectory is defined as a set of rectilinear paths parallel to the axis of the sheet is generated. This trajectory is then shifted to obtain the robot trajectory.

Keywords: Robotics, Path planning, Computational geometry, Trapezoidal decomposition, ROS

Referencias

[1] Acar, E.U. et al.: Morse Decompositions for Coverage Tasks. Int. J. Robot. Res. 21, 4, 331–344 (2002). https://doi.org/10.1177/027836402320556359.
[2] Chazelle, B.: Approximation and decomposition of shapes. In: Advances in Robotics 1: Algorithmic and Geometric Aspects of Robotics. pp. 145–185 (1987).
[3] Choset, H. et al.: Principles of Robot Motion. The MIT Press, London.
[4] Choset, H., Pignon, P.: Coverage Path Planning: The Boustrophedon Cellular Decomposition. In: Zelinsky, A. (ed.) Field and Service Robotics. pp. 203–209 Springer London, London (1998). https://doi.org/10.1007/978-1-4471-1273-0_32.
[5] Gabriely, Y., Rimon, E.: Spiral-STC: an on-line coverage algorithm of grid environments by a mobile robot. In: Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292). pp. 954–960 vol.1 (2002). https://doi.org/10.1109/ROBOT.2002.1013479.
[6] Galceran, E., Carreras, M.: A survey on coverage path planning for robotics. Robot. Auton. Syst. 61, 12, 1258–1276 (2013). https://doi.org/10.1016/j.robot.2013.09.004.
[7] Wong, S.C., MacDonald, B.A.: Complete Coverage by Mobile Robots Using Slice Decomposition Based on Natural Landmarks. In: Zhang, C. et al. (eds.) PRICAI 2004: Trends in Artificial Intelligence. pp. 683–692 Springer Berlin Heidelberg, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28633-2_72.
[8] Yang, S.X., Luo, C.: A Neural Network Approach to Complete Coverage Path Planning. IEEE Trans. Syst. Man Cybern. Part B Cybern. 34, 1, 718–724 (2004). https://doi.org/10.1109/TSMCB.2003.811769.
[9] actionlib - ROS Wiki, http://wiki.ros.org/actionlib.
[10] msg - ROS Wiki, http://wiki.ros.org/msg.
[11] Robots/SummitXL - ROS Wiki, http://wiki.ros.org/Robots/SummitXL.
[12] srv - ROS Wiki, http://wiki.ros.org/srv.
[13] summit_xl_common - ROS Wiki, http://wiki.ros.org/summit_xl_common.
[14] summit_xl_sim - ROS Wiki, http://wiki.ros.org/summit_xl_sim.