Escuela
Politécnica
Superior

Control visual de un robot móvil para aplicación de pegamento en un proceso industrial

Grado en Ingeniería Robótica

Trabajo Fin de Grado

Autor:
Aurelio Ortiz de Salazar Peris
Tutor/es:
Fernando Torres Medina
José Doñate Alfaro

Mayo 2022

Universitat d'Alacant
Universidad de Alicante

Control visual de un robot móvil para aplicación de pegamento en un proceso industrial

Diseño de una solución robótica para un proceso industrial mediante el control por visión de un robot móvil

Autor
Aurelio Ortiz de Salazar Peris

Tutor/es
Fernando Torres Medina
B149 FÍSICA, INGENIERÍA DE SISTEMAS Y TEORÍA DE LA SEÑAL
José Doñate Alfaro
Automática y Control Numérico

Grado en Ingeniería Robótica

Escuela Politécnica Superior

Universitat d'Alacant
Universidad de Alicante

ALICANTE, Mayo 2022

Preámbulo

"Este proyecto fue propuesto por la empresa Automática y Control Numérico en Julio de 2021 tras realizar una estancia de prácticas en la empresa. El proyecto surge como la solución planteada ante un proceso industrial que se quiere automatizar. La finalidad del proyecto es desarrollar un prototipo de la solución y comprobar su viabilidad."

Índice general

1 Introducción **1**
1.1 Marco de proyecto 2
1.2 Objetivos 3

2 Estado del arte **5**
2.1 Introducción a la robótica móvil 5
2.2 Ejemplos de robots móviles en la industria 7
2.3 Control visual de robots 9

3 Metodología **13**
3.1 Descripción de la tarea y el entorno 13
3.2 Sistema de control del robot 15
3.2.1 Localización del robot 16
3.3 Robot móvil 18
3.3.1 Selección del robot 18
3.3.2 Anatomía del robot 20
3.3.3 Comunicación con el robot 21
3.4 Reconstrucción del entorno 22
3.4.1 Calibración 23
3.5 Control del robot 24
3.5.1 Instrucciones por tiempo 24
3.5.2 Movimiento según la lectura de los encoders 26
3.5.3 Controlador todo-nada 27
3.5.4 Controlador proporcional 28

4 Resultados **33**

5 Conclusiones **37**

Bibliografía **39**

Índice de figuras

1.1 Proceso actual de aplicación de pegamento 1

2.1 Robot móvil RB-ROBOUT de Robotnik[1] 7
2.2 Manipulador móvil de Robotnik[2] 8
2.3 Robot móvil SUMMIT-XL de Robotnik[3] 8
2.4 Robot móvil customizable RB-VULCANO PRO de Robotnik[4] 9
2.5 Esquema de control visual 9
2.6 Configuraciones de la cámara 10
2.7 Combinación de las dos configuraciones 11

3.1 Extensión de los materiales sobre la mesa 13
3.2 Proceso actual de aplicación de pegamento 14
3.3 Muestras de mallas de infusión sobre las que navegará el robot 14
3.4 Representación del sistema de control 15
3.5 Microsoft LifeCam Studio Webcam 15
3.6 Ejemplo de uso de los Apriltag para localización e identificación de varios robots[5] 16
3.7 Robot con el Apriltag colocado para su localización 16
3.8 Detección de la pose del Apriltag 17
3.9 Detección de la pose del robot 17
3.10 Comparación del sensor estropeado y el funcional 18
3.11 iRobot Create 2 19
3.12 iRobot Roomba 669 20
3.13 Medidas físicas del robot [6] 20
3.14 Puerto Mini-DIN de 7 pines del robot 21
3.15 Diagrama de conexión entre el robot y la raspberry 21
3.16 Reconstrucción del entorno en casa 22
3.17 Reconstrucción del entorno en AyCN 23
3.18 Rango de visión en AyCN 23
3.19 Calibración de la cámara en ambas reconstrucciones del entorno 24
3.20 Interfaz de control del robot 25
3.21 Interfaz de realización de trayectoria utilizando los encoders 26
3.22 Representación del controlador 27
3.23 Casos de movimiento del robot que se quieren evitar 29
3.24 Ejemplo de funcionamiento para llegar a un punto con el aplicador de pegamento 30
3.25 Ejemplo de cálculo del punto naranja en una trayectoria 31
3.26 Puntos generados para cumplir con la trayectoria 32

4.1 Detección de la pose del robot con el Apriltag 33
4.2 Prueba de funcionamiento del controlador proporcional 36

1 Introducción

El presente trabajo trata de utilizar un robot móvil para cubrir un proceso industrial que consiste en depositar pegamento en determinadas ubicaciones para pegar dos materiales. Para ello, el robot móvil deberá ser capaz de navegar por el entorno para alcanzar la ubicación deseada y depositar el pegamento. El entorno sobre el que va a trabajar el robot es una mesa de 25 metros de largo y 6 metros de ancho sobre la cual hay colocados distintos materiales que el robot no debe dañar al navegar sobre ellos. Actualmente, el proceso lo realizan dos personas, una coloca el pegamento con una pistola de silicona y la segunda lo prensa. En las siguientes imágenes se puede ver el proceso:

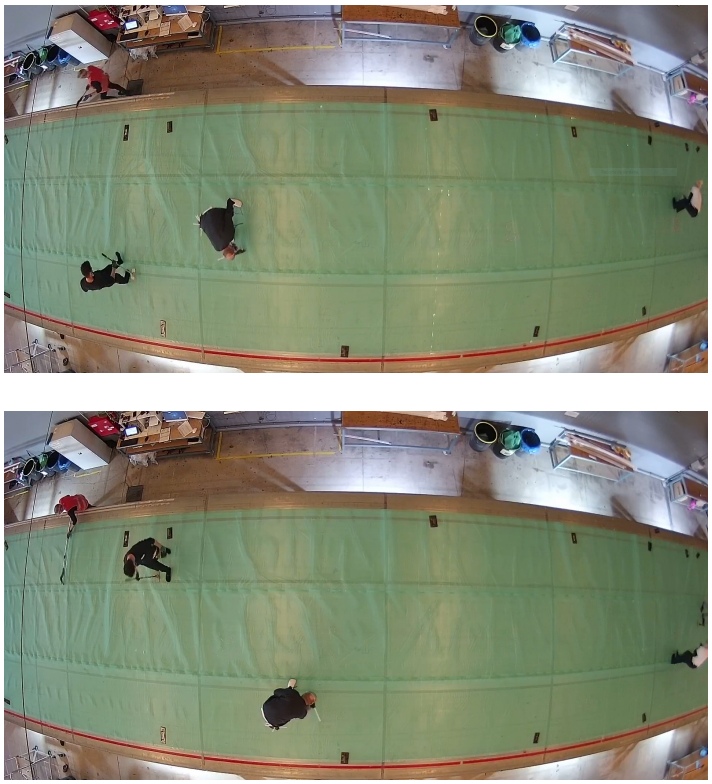


Figura 1.1: Proceso actual de aplicación de pegamento

El sistema robótico planteado está compuesto por una cámara, un ordenador, un robot móvil y un microcontrolador. La cámara estará situada perpendicular al plano de la mesa aportando imágenes sobre el robot y el entorno. El ordenador será el cerebro del sistema, analizará la imagen de la cámara y obtendrá información de utilidad, como la pose del robot,

y también determinará la acción a realizar para llegar a la ubicación y depositar el pegamento. El microcontrolador se encargará de recibir la información del ordenador y enviar los comandos correspondientes al robot móvil. El robot móvil que se va a utilizar es un robot aspiradora de la marca iRobot, concretamente el modelo 669 de la serie 600.

Durante este documento se verá cómo se ha realizado el proyecto, explicando con detalle las fases por las que ha pasado y cómo se han trabajado. A continuación, se va a explicar brevemente de qué consta cada uno de los apartados del proyecto:

- Introducción. Consta de dos apartados:
  - Marco de proyecto, en el que se explicará el contexto en el que se ha realizado el proyecto, ya que surge como una propuesta por la empresa de Automática y Control Numérico.
  - Objetivos, en el que se explicarán las metas que se quieren cumplir con el proyecto.
- Estado del arte (State of the art), en el que se introducirán los robots móviles y sus aplicaciones en procesos industriales así como los métodos de control visual.
- Metodología, en el que se detallará todo el proceso de desarrollo del proyecto. Destacan apartados como: la selección del robot, la reconstrucción del entorno o el controlador del robot.
- Resultados, en el que se mostrarán los resultados conseguidos a medida que el proyecto avanza. En este apartado se podrá apreciar la mejoría de estos resultados desde el primer controlador planteado hasta el controlador que permite el correcto funcionamiento de la tarea.
- Conclusiones, en el que se determina la solución desarrollada y se comprueba que se hayan cumplido los objetivos deseados.

1.1 Marco de proyecto

Este proyecto surge como propuesta de TFG con empresa en virtud del acuerdo existente entre la UA y Automática y Control Numérico, AyCN. El proyecto trata de aplicar técnicas de visión para la navegación de robots móviles para un proceso industrial concreto.

Durante el mes de Julio de 2021, se realizó una estancia de prácticas en esta empresa y al finalizarla, se propuso la realización de este proyecto. El proyecto está dirigido a automatizar un proceso industrial de una segunda empresa, por lo que se quiere desarrollar un prototipo en AyCN extrapolable a esa empresa con el fin de proponérselo y llevarlo a cabo en ella.

El proceso consiste en la aplicación de pegamento en determinadas ubicaciones para pegar dos materiales. Actualmente se lleva a cabo por dos personas, una deposita el pegamento y la otra lo prensa. La idea del proyecto surgió cuando la empresa de AyCN se percató (durante una visita a la segunda empresa) de que había dos personas agachándose y levantándose para depositar el pegamento y prensarlo durante gran parte de su jornada laboral, por lo que decidieron plantear una solución para cubrir ese proceso. De esta manera, los trabajadores pueden realizar otras tareas de mayor valor para la empresa y para ellas mismas.

1.2 Objetivos

Los objetivos de este proyectos son los siguientes:

- Selección de un robot móvil adecuado para la tarea. El robot debe poder navegar con facilidad por los materiales sin dañarlos. Además, debe de tener un coste bajo ya que se trata de un prototipo.
- Desarrollo de un sistema de control de útil y extrapolable. El sistema que se encargará de controlar la tarea que realiza el robot debe poder ser extrapolable a la empresa a la que va destinada sin provocar grandes cambios en el funcionamiento del proceso actual.
- Desarrollo de un controlador de movimiento del robot. El robot debe depositar pegamento en determinadas ubicaciones con una precisión satisfactoria para el proceso, por lo que se necesita la creación de un controlador de movimiento que lo permita. Este controlador debe servir para trayectorias largas, cortas e intermedias así como para puntos aislados.
- Creación de un prototipo de solución. Los objetivos anteriores engloban la creación de un prototipo de solución para la automatización del proceso.
- Mostrar el potencial de incorporar un robot móvil al proceso en cuestión.

2 Estado del arte

En esta sección se va a realizar una introducción de cómo surge la robótica móvil y los dos tipos de robots móviles que existen de acuerdo al tipo de navegación, los guiados y los autónomos. Posteriormente, se comentarán varios robots industriales de la marca Robotnik, una empresa española especializada en el desarrollo de productos robóticos y proyectos de I+D en robótica. Esta marca destaca por el diseño de robots móviles y manipuladores móviles para aplicaciones industriales. Además, cuenta con diseños de robots móviles generalizados, que pueden servir para múltiples aplicaciones, y con diseños particulares (customizados) que se crearon por encargo y sirven para tareas muy específicas. Esto resulta interesante ya que la creación de robots móviles que puedan servir para una amplia gama de aplicaciones es una tarea complicada, por ello, normalmente se crean robots específicos para realizar una tarea en concreto.

Posteriormente, se hablará del control visual de robots. Se comentarán aspectos como la configuración de la cámara o el tipo de control que se puede realizar, control visual basado en imagen o control visual basado en posición.

2.1 Introducción a la robótica móvil

La robótica es una ciencia que ha ido evolucionando junto al ser humano y se basa en la creación de robots. El término robot surge en 1921, acuñado por el escritor checo Capek en su obra *Rossum's Universal Robots*, a partir de la palabra checa *robota*, que significa servidumbre o trabajo forzado (ver García Sánchez y cols., 2007). Por lo tanto, un robot es una máquina que se utiliza para realizar tareas de servidumbre o trabajo forzado para el ser humano. Sin embargo, el rápido avance de este campo ha provocado la expansión de las aplicaciones de los robots a muchos campos que se alejan de la servidumbre y de trabajos forzados, por lo que una definición más cercana a lo que los robots son hoy en día es que un robot es un sistema electromecánico reprogramable que permite realizar diferentes tareas repetitivas que requieren un elevado grado de precisión (definición de García Sánchez y cols., 2007).

En la década de los 70 la robótica estaba presente en muchos procesos de la industria pero no como se conoce actualmente(ver Ollero Baturone, 2001) (y Cruz, 2008). Con el avance de la tecnología y la competencia en la industria, muchas fábricas modificaron su proceso de producción para ser más competentes en el mercado. Esta modificación se basaba en combinar procesos ya existentes para generar nuevos productos o en ser más eficientes en el transporte entre procesos, mejorando así el tiempo de producción. Esto creó la necesidad de extender el campo de la robótica, que se encontraba restringido al alcance de una estructura mecánica anclada en uno de sus extremos, y surgieron los primeros robots móviles para la industria, vehículos móviles guiados por rieles para realizar un transporte de mercancía eficaz entre los distintos procesos de producción. A estos robots móviles se les conocen como AGV, vehículos guiados automáticamente.

Un robot móvil se define como un sistema electromecánico capaz de desplazarse de manera autónoma sin estar sujeto físicamente a un solo punto. Posee sensores que permiten monitorear a cada momento su posición relativa a su punto de origen y a su punto de destino. Normalmente su control es en lazo cerrado. Su desplazamiento es proporcionado mediante dispositivos de locomoción, tales como ruedas, patas, orugas, etc (definicion de García Sánchez y cols., 2007).

Retomando a los robots móviles AGV, la inclusión de este tipo de robots en la industria supuso un gran beneficio en cuanto a tiempos de producción, sin embargo, requieren una modificación del entorno de trabajo para incluir la guía que seguirán los robots. La más habitual era un cable enterrado que crea un campo magnético y debe ser detectado por el vehículo. Aunque actualmente, una de las más habituales es el optoguiado para detectar una línea o bandas de un color determinado, la implementación de estas guías son menos costosas porque eliminan la necesidad de enterrar un cable en el suelo (ver Ollero Baturone, 2001). Gracias a este hito de inclusión de robots móviles en la industria, la investigación y el diseño de robots móviles creció exponencialmente, creciendo también sus aplicaciones hasta el punto de que abarca campos como: exploración minera, exploración planetaria, misiones de búsqueda y rescate de personas, limpieza de desechos peligrosos, automatización de procesos, vigilancia, reconocimiento de terreno, asistencia médica, exploración marítima, entretenimiento, investigación y desarrollo, investigación militar, agricultura, inspección, transporte, etc.

El diseño de robots móviles varía notablemente en función de la tarea que vayan a realizar en cuanto a tipo de locomoción, estructura y características físicas se refiere. Los robots móviles se pueden clasificar de varias maneras, por ejemplo, según su tipo de locomoción, según sus aplicaciones, según su arquitectura o según el tipo de navegación por el entorno. Este último determina dos grupos claramente diferenciados, los robots móviles AGV y los AMR (ver Aguilera Hernández y cols., 2007). Como se ha comentado anteriormente, los AGV son vehículos de guiado automático que siguen una trayectoria delimitada por una guía de la que nunca se saldrán y que no debe obstaculizarse. Esto permite la navegación con una inteligencia sencilla y sensores básicos, y es realmente útil y eficiente en muchos casos. Además, permite la adición de más robots al sistema sin demasiadas complicaciones. Por otro lado, los robots AMR, robots móviles autónomos, no siguen una trayectoria predefinida, si no que la calculan y son capaces de abandonarla y retomarla o calcular una nueva en caso de que exista algún entorpecimiento. Estos robots están dotados de mayor inteligencia y de sensores más sofisticados para permitir la navegación. Normalmente, el entorno es conocido, es decir, el robot tiene la información del mapeado del entorno (que puede haber obtenido él mismo previamente) y por ello puede calcular rutas de un punto a otro evitando obstáculos. Los beneficios que aportan los AMR a la industria son los siguientes: no se necesita una modificación del entorno de trabajo, son flexibles, es decir, las tareas que pueden abordar son muy variadas (ya que puede navegar por el entorno de forma inteligente), las aplicaciones a las que puede estar destinado están limitadas por la física del robot y no por su sistema de navegación, el robot es capaz de adaptarse a cambios en el entorno y los sistemas AMR son escalables (se pueden añadir varios robots sin entorpecer el funcionamiento).

Por lo tanto, los robots móviles han avanzado notablemente desde su creación debido a la gran cantidad de aplicaciones que pueden realizar y a su gran demanda. La creación de estos robots no es generalizada para llevar a cabo una gran cantidad de tareas, si no que se crean en función de la tarea que van a realizar. Esto provoca que las empresas que se dedican a

crear robots móviles creen robots móviles muy demandados por la industria, como pueden ser robots móviles de logística, pero también creen robots particulares (customizados) por encargos para tareas muy específicas.

2.2 Ejemplos de robots móviles en la industria

Los robots móviles en la industria, como se ha comentado anteriormente, se diseñan para cubrir una tarea en específico y no una gran variedad de ellas. Por ello, las empresas intentan llegar al mayor número de clientes creando robots algo generalizados para realizar una pequeña variedad de tareas. A continuación, se comentarán robots móviles creados por Robotnik que simpatizan con lo comentado. Robotnik ofrece 3 tipos de productos a los que categoriza como robots móviles, manipuladores móviles y customizados.

En cuanto a los robots móviles, la mayoría son de logística. Por ejemplo, el robot móvil RB-ROBOUT está concebido para transportar cargas de hasta 1 tonelada de peso en entornos industriales de interior, como fábricas o almacenes. Gracias a los láseres de seguridad, es un robot autónomo y colaborativo capaz de compartir el espacio de trabajo con los operarios de forma inteligente detectando obstáculos, buscando rutas alternativas y optimizando las trayectorias.[1]. Las características físicas de este robot limitan su funcionamiento a entornos interiores y sin desniveles, sin embargo, ofrece otras características como la capacidad de carga de 1 tonelada y sistema de elevación. Este robot es ideal para transportar cargas muy elevadas por entornos de interior.

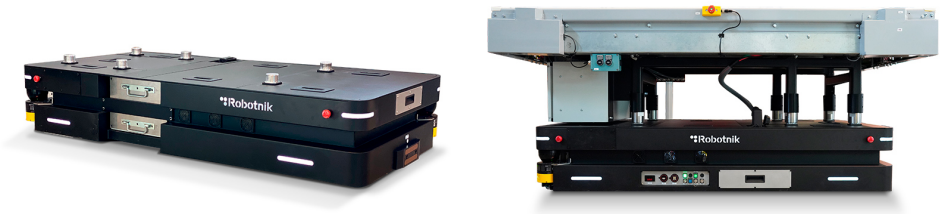


Figura 2.1: Robot móvil RB-ROBOUT de Robotnik[2]

Como se ha comentado, muchos de los robots móviles de Robotnik son de logística. Esto es debido a que puede que un robot no se adapte a las necesidades del cliente, y por ello crean varios similares pero con características propias para distintas aplicaciones dentro de la logística, adaptándose a sí a muchas más tareas. El robot móvil RB-THERON es similar al RB-ROBOUT pero con menor capacidad de carga (200 kg) y menores dimensiones. El robot SUMMIT-XL STEEL ofrece una capacidad de carga de 250 kg y un chasis de acero inoxidable, ideal para trabajar con materiales corrosivos o en entornos húmedos. El robot RB-VOGUI ofrece navegación en interiores y exteriores. Todos los robots mencionados anteriormente ofrecen navegación autónoma y comparten la característica de poder soportar carga y transportarla, sin embargo, cada uno ofrece características distintas para realizar una tarea en concreto.

[1]Definición del robot obtenida de la página oficial de Robotnik
[2]Fotos obtenidas de la página oficial de Robotnik

No obstante, estos robots ofrecen muchas posibilidades para ser customizados. Por ejemplo, al poder soportar carga, se le puede ensamblar un robot manipulador y convertir al robot en un manipulador móvil, ampliando enormemente su campo de utilidad.

Figura 2.2: Manipulador móvil de Robotnik[3]

Por otro lado, Robotnik cuenta con un robot móvil muy versátil para trabajar en interiores y exteriores, SUMMIT-XL. Este robot puede soportar cargas de hasta 65 kg y puede servir para multitud de tareas, entre las que destacan tareas de vigilancia, militares y de exploración.

Figura 2.3: Robot móvil SUMMIT-XL de Robotnik[4]

La ventaja de este robot es que dispone de una configuración de ruedas apropiada para entornos exteriores y que puede navegar de forma autónoma o ser teleoperado. Al tener libre el techo del robot (y poder soportar hasta 65 kg) se pueden incorporar diversas herramientas para realizar distintas tareas, de ahí su gran versatilidad.

Aparte de robots móviles y manipuladores móviles, Robotnik también diseña robots customizados para aplicaciones concretas. Estos robots los diseñan para proyectos de I+D o por encargos. Por ejemplo, el robot RB-ARES es especial para transporte de palés o el robot ELI es un carro de compra inteligente. Sin embargo, dentro de esta sección existe un robot muy interesante, el RB-VULCANO PRO. Este robot es una plataforma móvil vacía diseñada

[3]Foto obtenida de la página oficial de Robotnik
[4]Fotos obtenidas de la página oficial de Robotnik

para integrar diversos componentes para la realización de numerosas tareas en entornos industriales. Entre esos componentes se encuentran herramientas, estructuras, soportes, brazos robóticos, elevadoras, camino de rodillos, cisternas y unidades neumáticas/hidráulicas, entre otros.

Figura 2.4: Robot móvil customizable RB-VULCANO PRO de Robotnik[5]

Este robot es especialmente útil para un entorno industrial cambiante. En el que un robot móvil puede realizar multitud de tareas simplemente añadiendo las herramientas necesarias a la mesa de trabajo. Además, esta mesa de trabajo permite un fácil intercambio de herramientas, cambiando por completo la funcionalidad del robot.

2.3 Control visual de robots

El uso de cámaras para controlar la posición del robot, control visual, es una técnica muy utilizada actualmente para multitud de robots ya sean manipuladores, móviles, drones, robots paralelos, etc. Este control se basa en que la información que aporta una imagen adquirida por una cámara es la realimentación al controlador del robot:

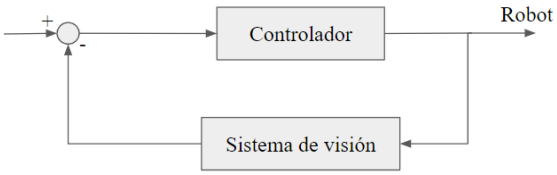


Figura 2.5: Esquema de control visual

Existen básicamente dos aproximaciones para realizar el control visual: control visual basado en imagen, en la que la señal de error se obtiene directamente de la imagen y se traduce en una acción de control; y, control visual basado en posición, en la que a partir de la imagen

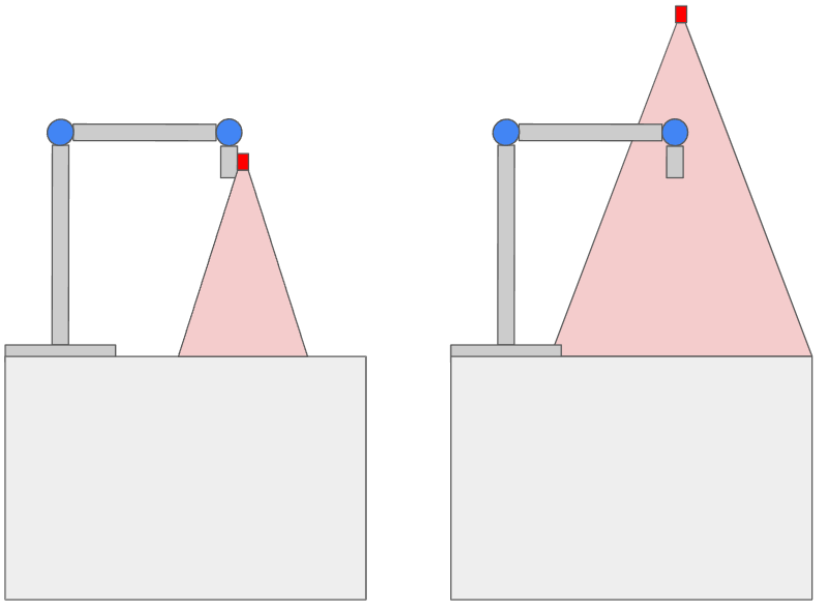
[5]Foto obtenida de la página oficial de Robotnik

adquirida se reconstruye el espacio 3D de trabajo y la acción de control se obtiene respecto a éste (ver Corke y Hutchinson, 2000).

En el control visual basado en posición se calcula la posición 3D del punto a alcanzar a partir de la información visual. Con la ubicación 3D del punto y con la posición cartesiana del robot, se calcula un error en posición y orientación (cartesiano) para controlar al robot hasta el punto destino. Este control es muy sensible a errores de calibración.

Por otro lado, en el control visual basado en imagen no se calcula el punto 3D, si no que se utiliza la información del plano 2D (imagen) para determinar la acción de control, la función de error que realimenta el bucle de control es la diferencia entre los valores de las características deseadas y las obtenidas en cada momento. Gracias a ello, este método es menos sensible a errores de calibración y al mismo tiempo reduce los tiempos de procesamiento (ver Garcia y cols., 2004).

Las dos configuraciones más habituales del sistema de visión se conocen como *eye-in-hand*, la cámara se sitúa en el extremo del robot, y *eye-to-hand*, la cámara se sitúa en el entorno. La primera garantiza una buena precisión y capacidad de moverse por el espacio de trabajo con una vista limitada. La segunda garantiza una visión más amplia del espacio de trabajo pero una menor precisión.



(a) Configuración *eye-in-hand* **(b)** Configuración *eye-to-hand*

Figura 2.6: Configuraciones de la cámara

En la imagen anterior se puede apreciar claramente el rango de visión de cada configuración y cómo afecta a su precisión. Por ejemplo, la configuración *eye-in-hand* es muy útil si el área de trabajo de la tarea que tiene que realizar el robot es un plano (como la mesa en el caso de la imagen). Sin embargo, si el área de trabajo es más dinámica, la información que puede aportar la configuración *eye-to-hand* es de más utilidad. Por ello, el uso de ambas configuraciones

al mismo tiempo facilita la ejecución de tareas complejas y ofrece una mayor flexibilidad ante un escenario dinámico (ver Lippiello y cols., 2005). Ya que conserva la precisión de la configuración *eye-in-hand* y el rango de visión de la configuración *eye-to-hand*.

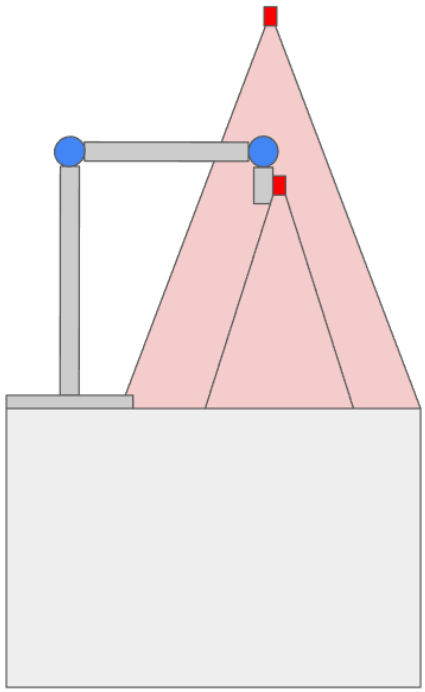


Figura 2.7: Combinación de las dos configuraciones

En este proyecto se utilizará la visión basada en imagen y la configuración *eye-to-hand*. La cámara se encuentra perpendicular al plano de la mesa y se extraerán las características de interés para el control del robot a partir de la imagen. Estas características son la pose actual del robot, que se obtendrá en píxeles, y la posición destino, que será un punto en la imagen. La acción de control del robot viene determinada por la distancia entre la posición del robot y el punto destino y por el error existente en el ángulo. Por lo tanto, se podría concluir que en el control visual basado en imagen que se va a realizar, la función de error que determina la acción de control se calcula a partir de las diferencias entre las características 2D del robot en la imagen y las deseadas (pose actual del robot y posición destino). Cabe destacar que en este tipo de control hay que tener en cuenta que la extracción de las características de interés de la imagen pueden suponer un coste temporal relevante para el controlador, sin embargo, en este caso no ocurre ya que la extracción de la pose del robot es rápida (localización mediante Apriltags, que se verán más tarde) y el punto destino se conoce previamente (no hay que calcularlo o identificarlo); además, el cálculo de las diferencias entre la pose actual y la destino se basa en operaciones matemáticas simples (como la distancia entre dos puntos o el ángulo que forman dos puntos)

3 Metodología

3.1 Descripción de la tarea y el entorno

Como se ha comentado anteriormente, la tarea que tiene que realizar el robot es depositar pegamento en determinadas ubicaciones. Estas ubicaciones son conocidas de antemano y varían según la pieza que se quiera pegar, pueden conformar el contorno de la pieza que se quiere pegar o pueden ser puntos aislados.

El proceso industrial en el que se quiere incluir el robot consta de varios pasos. En primer lugar, se extienden los materiales sobre los que se va a trabajar en la mesa:

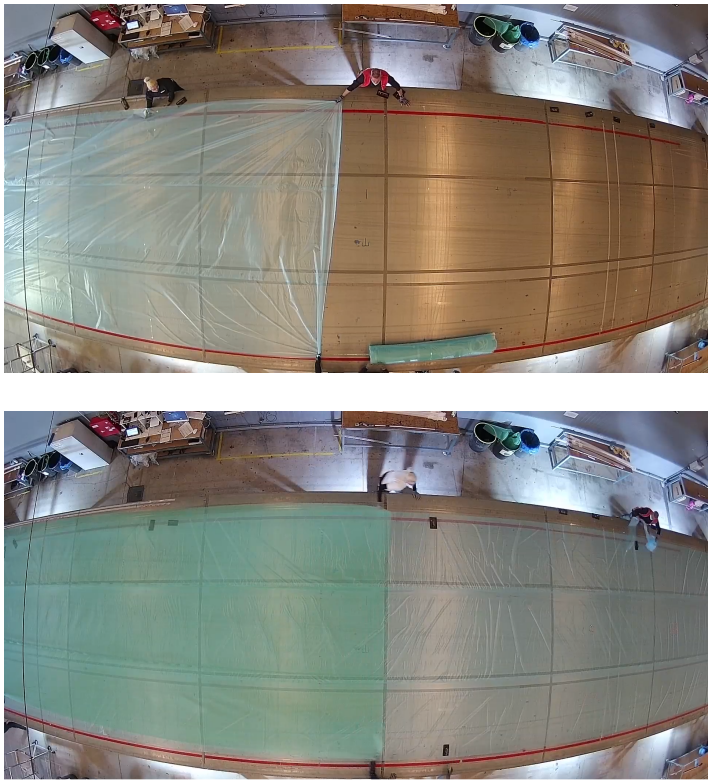


Figura 3.1: Extensión de los materiales sobre la mesa

Tras una serie de pasos que realizan los trabajadores, se disponen a realizar la tarea se quiere hacer con el robot, que es aplicar pegamento y prensarlo. La tarea actualmente se realiza por dos personas, una deposita el pegamento y la otra lo prensa. La ubicación viene

determinada por una proyección láser proyectada sobre las mallas. En las siguientes imágenes se puede ver el proceso real.

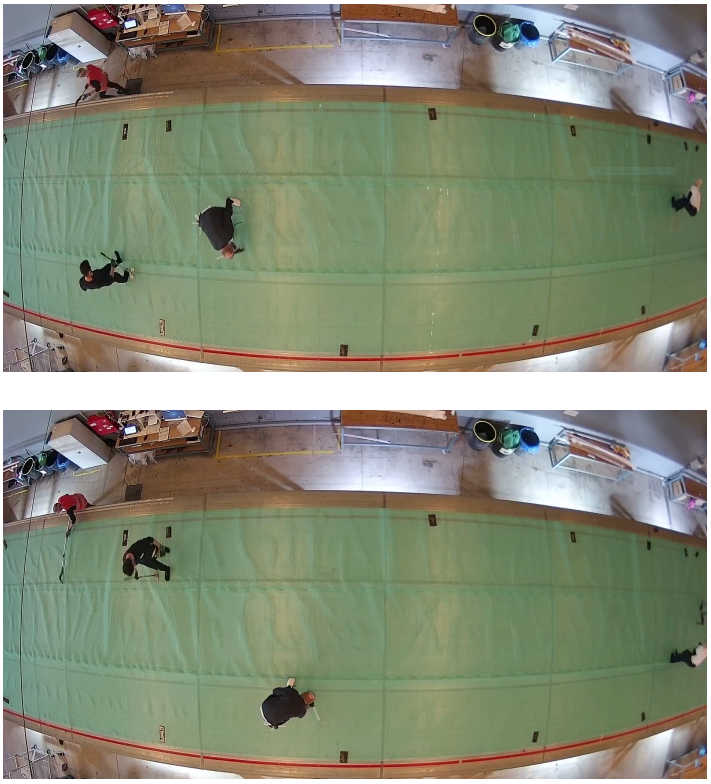


Figura 3.2: Proceso actual de aplicación de pegamento

El entorno sobre el que va a trabajar el robot es una mesa de 25 metros de largo y 6 de ancho sobre la que hay colocados distintos materiales, mallas de infusión. Las mallas de infusión se utilizan en procesos de infusión en los que se quiere ayudar a mejorar el flujo de la resina en el medio laminado. La infusión de resina es un proceso que se utiliza en la industria del composite, que consiste en crear materiales compuestos que se forman por la unión de dos o más materiales con el fin de combinar sus propiedades.

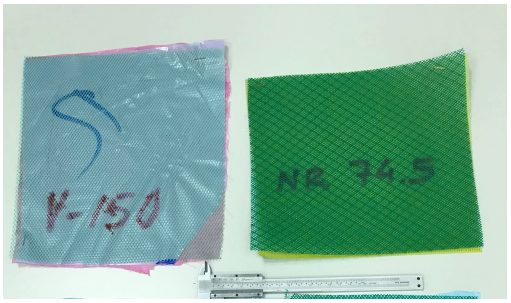


Figura 3.3: Muestras de mallas de infusión sobre las que navegará el robot

3.2 Sistema de control del robot

El diseño del sistema de control planteado para cumplir con la tarea se basa en control visual, en concreto en el control visual basado en imagen. Para ello, se utilizará la información que proporciona una cámara situada perpendicularmente al plano de la mesa. El sistema de control del robot estará formado por la cámara, un microcontrolador y un ordenador. El ordenador será el responsable de adquirir la imagen de la cámara, analizarla y enviar la información necesaria al microcontrolador y el microcontrolador enviará los comandos correspondientes al robot en función de la información recibida.

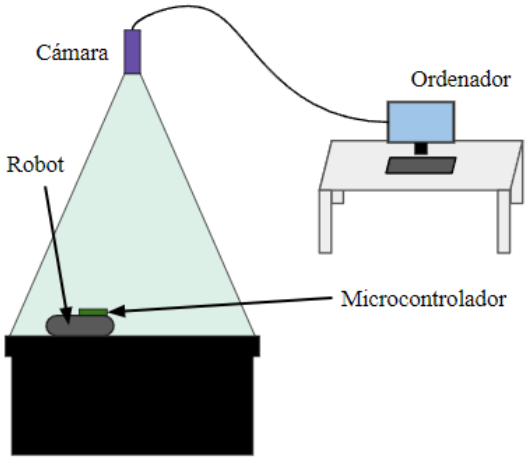


Figura 3.4: Representación del sistema de control

La cámara que se va a utilizar es la Microsoft LifeCam Studio Webcam HD, cuyo precio ronda los 70€. Esta cámara se suele utilizar en la empresa AyCN para primeras pruebas de proyectos en los que es necesario un aporte visual, por lo que ya se tenía disponibilidad de ella. Es una cámara con una resolución de imagen de 1920x1080, se conecta mediante USB y cuenta con autoenfoque.

Figura 3.5: Microsoft LifeCam Studio Webcam

El microcontrolador que se va a utilizar es la Raspberry Pi Model 3B+. Este microcontrolador posee Wifi para recibir la información enviada por el ordenador y comunicación por puerto serie para comunicarse con el robot.

3.2.1 Localización del robot

Para localizar al robot con la cámara, se han utilizado los Apriltags. AprilTag es un sistema visual de referencia, útil para una gran variedad de tareas, como la realidad aumentada, la robótica y la calibración de cámaras. Los objetivos pueden crearse a partir de una impresora normal, y el software de detección de AprilTag calcula la posición 3D precisa, la orientación y la identidad de las etiquetas en relación con la cámara. La biblioteca AprilTag está implementada en C sin dependencias externas. Está diseñada para ser incluida fácilmente en otras aplicaciones, así como para ser portátil en dispositivos integrados. El rendimiento en tiempo real puede lograrse incluso en procesadores de nivel de teléfono móvil.[1].

Figura 3.6: Ejemplo de uso de los Apriltag para localización e identificación de varios robots[2]

A pesar de que son parecidos a los códigos QR, los Apriltags identifican más rápido y únicamente aportan información geométrica, como sus esquinas y su centro. Existe un repositorio oficial[3] en el que se pueden ver todos los Apriltags (que son limitados) que se pueden emplear. Existen 3 familias *tag16h5*, *tag25h9* y *tag36h11*. Para localizar el robot, el Apriltag se coloca en el centro del robot para que en la detección del Apriltag, su centro coincida con el centro del robot.

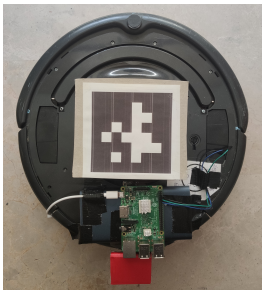


Figura 3.7: Robot con el Apriltag colocado para su localización

Además, como también proporciona la ubicación de las esquinas se puede conocer el ángulo con respecto a la cámara (ya que las esquinas están identificadas). Con estos datos se puede conocer la pose del robot. A continuación, se van a mostrar varias imágenes en las que se

[1]Definición de Apriltag obtenida de la página oficial de Apriltag
[2]Foto obtenida de la página oficial de Apriltag
[3]Repositorio oficial de Apriltag

identifica la pose [x, y] en píxeles y el ángulo del Apriltag. En estas imágenes se puede ver que se detecta aun teniendo un ángulo de visión complicado:

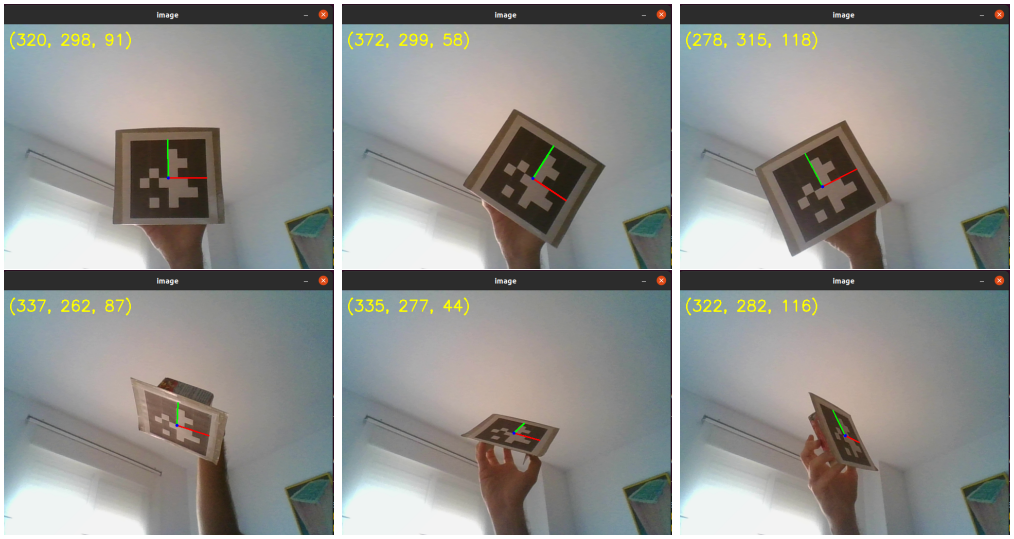


Figura 3.8: Detección de la pose del Apriltag

Si se coloca el Apriltag en el robot, los resultados son adecuados para poder identificar la pose del robot, como se puede ver en las siguientes imágenes:

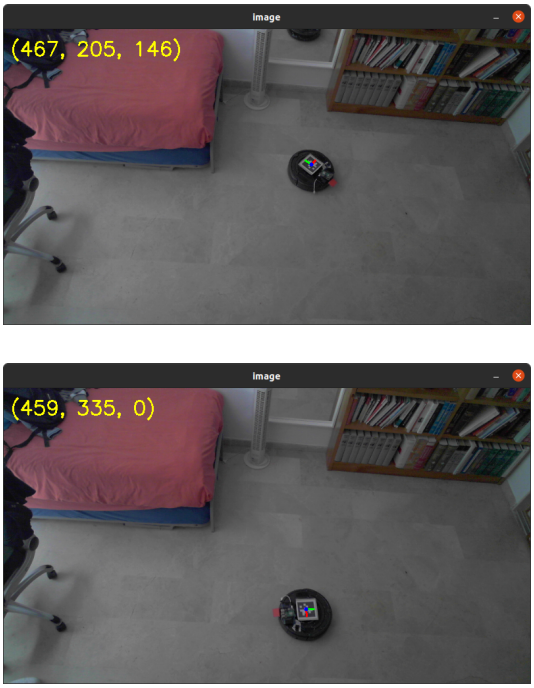


Figura 3.9: Detección de la pose del robot

3.3 Robot móvil

3.3.1 Selección del robot

La selección del robot móvil es uno de los aspectos más importantes del proyecto. Debe ser de coste bajo, apropiado y específico para la tarea a realizar. Actualmente, los robots móviles en el mercado en el ámbito industrial son principalmente de logística, robots AGV (Autonomous Intelligent Vehicle). Suelen ser de 4 ruedas, superficie plana y con gran capacidad de carga. Navegan por rutas previamente definidas aunque están dotados de inteligencia para generar nuevas rutas en caso de tener que hacer frente a obstáculos o situaciones imprevistas que impidan continuar con la ruta anterior. Además, suelen llevar sensores para la navegación autónoma como sensores LiDAR o cámaras. Sin embargo, en este caso no es necesario un robot de estas características, se necesita un robot móvil que pueda navegar a ubicaciones determinadas mediante control visual. La utilización de una cámara situada en el techo para visualizar el entorno sobre el que navega el robot hace innecesario el uso de los sensores de navegación mencionados ya que se puede conocer la posición del robot en todo momento, así como factores del entorno.

Durante varias semanas se planteó construir el robot móvil comprando los elementos hardware por separado, lo cual supuso una gran tarea de investigación de los componentes que debería llevar, como motores, ruedas, controlador o encoders. Se tomó como referencia la base de los Turtlebot. La idea era imprimir con una impresora 3D el modelo del Turtlebot3 Waffle y comprar dos motores de corriente continua que se controlarían con una placa de Arduino o una Raspberry Pi. Sin embargo, surgió la posibilidad de utilizar un robot aspiradora como base móvil. Este robot aspiradora pertenecía a un conocido de la familia y estaba estropeado debido a un mal uso de él cuando se intentó que limpiase la terraza tras un día de lluvia. El suelo estaba mojado y los sensores que identifican si el robot estaba frente a un precipicio se ensuciaron. Los robots aspiradora tienen protocolos que identifican estos casos y se identificó correctamente pidiendo una limpieza de los sensores. Los propietarios del robot intentaron limpiarla pero terminaron estropeando el sensor sucio. El robot detectaba que en todo momento estaba frente a un precipicio y se detenía, por lo que no era capaz de limpiar. Ante este problema decidieron sustituir este modelo por uno más actual, dejándolo sin utilizar.

(a) Sensor estropeado **(b)** Sensor funcional

Figura 3.10: Comparación del sensor estropeado y el funcional

Al conocer la situación, se solicitó a los propietarios el robot para utilizar sus componentes hardware para construir el robot, sin embargo, al desmontarlo se identificó una entrada de

comunicación por puerto serie con el robot. Tras investigar la marca y el modelo, iRobot Roomba 669 de la serie 600, se descubrió que la marca lanzó a la venta un robot aspiradora programable orientado a la educación, iRobot Create 2 (una plataforma de robot móvil pre-ensamblada para programar comportamientos, sonidos, movimientos, así como añadir componentes electrónicos adicionales mediante un ordenador o un microcontrolador[4]). Este robot está basado en los modelos de la serie 600 y por ello, el robot del que se dispone para el proyecto será compatible con todo el desarrollo software del iRobot Create 2.

Figura 3.11: iRobot Create 2

Atendiendo al manual del iRobot Create 2 [5], en la sección de modos, se explica que existen tres modos, modo pasivo (*passive mode*), modo seguro (*safe mode*) y modo completo (*full mode*). El modo seguro ofrece un control total sobre el robot pero tiene en cuenta 3 medidas de seguridad que provocarán que el robot se detenga y entre en modo pasivo: detección un precipicio, levantamiento una rueda y conexión con el cargador. Sin embargo, el modo completo ignora las condiciones de seguridad anteriores, por lo que el problema del sensor estropeado que proporciona medidas falsas detectando precipicios desaparece y el robot puede funcionar perfectamente.

Tras una primera prueba conectando el Arduino al puerto serie, se consiguió que el robot ejecutara instrucciones como *muévete hacia delante* o *gira en sentido contrario a as agujas del reloj*, por ello, al ver que realmente era compatible con los comandos del iRobot Create 2, se investigó más a fondo el manual para descubrir todas las posibilidades que ofrece. Este manual explica todos los comandos que pueden controlar distintas partes del robot como motores, ruedas, luces o sensores. A continuación, se mencionan algunos relevantes para la navegación:

- *Drive*: comando que controla las ruedas del robot especificando la velocidad y el radio de giro
- *Drive direct*: comando que permite controlar la velocidad de cada rueda por separado de forma independiente
- *Distance*: comando que devuelve la distancia que ha recorrido el robot desde la última vez que se solicitó
- *Angle*: comando que devuelve el ángulo que ha girado el robot desde la última vez que se solicitó

[4]Definición del iRobot Create 2 obtenida de la página oficial
[5]Manual de iRobot Create 2

- *Right encoder counts*: comando que devuelve el contador del *encoder* de la rueda derecha
- *Left encoder counts*: comando que devuelve el contador del *encoder* de la rueda izquierda

Una vez se comprobó que el robot podía controlarse con un microcontrolador, se probó a navegar por encima de los materiales que se colocan en la mesa. El resultado fue correcto, el robot no dañaba el material y era capaz de navegar sobre él. Sin embargo, este robot ha sido utilizado como robot de limpieza durante mucho tiempo, lo que influye en el desgaste de algunos de sus componentes. El más relevante es la rueda izquierda. A la hora de moverse en línea recta, el movimiento resultante no es completamente recto, si no que traza un pequeño arco debido al desgaste de la rueda. A pesar de ello, ese efecto no tiene gran importancia, como se podrá ver más adelante.

Por tanto, el iRobot model 669 es el robot móvil que se ha escogido para llevar a cabo el proyecto.

Figura 3.12: iRobot Roomba 669

3.3.2 Anatomía del robot

El peso del robot es de de 3.5 kg. En la siguiente imagen se detallada su anatomía:

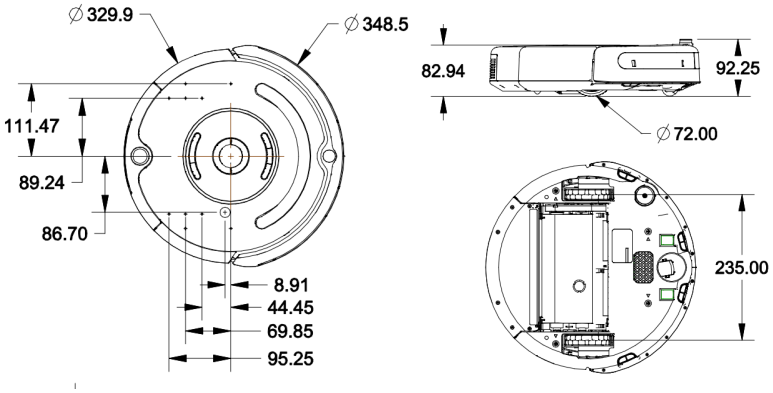


Figura 3.13: Medidas físicas del robot [6]

De la figura anterior se pueden destacar algunas medidas que serán importantes para el posterior desarrollo del proyecto: la distancia entre las ruedas, 235 mm, y el radio de las

[6]Imagen obtenida del manual del iRobot Create 2

ruedas, 36 mm. Estas dos medidas se utilizan para calcular la cinemática directa del robot que se interviene en el desarrollo del controlador.

3.3.3 Comunicación con el robot

El iRobot Roomba 669 dispone de un puerto Mini-DIN de 7 pines con el cual se puede comunicar el robot con un microcontrolador mediante una comunicación serie TTL (0-5V). Además, también incorpora conexión con la batería.

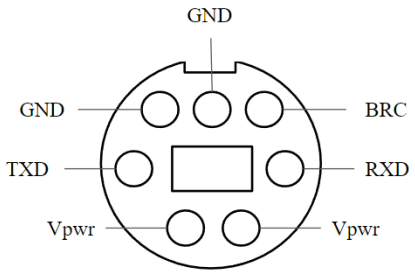


Figura 3.14: Puerto Mini-DIN de 7 pines del robot

Los pines *Vpwr* están conectados a la batería del robot sin regular, por lo que hay que utilizar un regulador de voltaje para adaptar los 14V de la batería de la roomba a los 5V de alimentación de la Raspberry. Los pines *TXD* y *RXD* son los pines de comunicación serie que irán conectados a la Raspberry y el pin *BRC* sirve para cambiar el *baudrate* de la comunicación.

A pesar de que se dispone de un pin de acceso a la batería de la roomba, esta se desgastaba rápidamente si se conecta a la raspberry, por lo que se decidió conectar la raspberry a una batería externa de 5V. El diagrama de conexión entre el robot y la raspberry sería el siguiente:

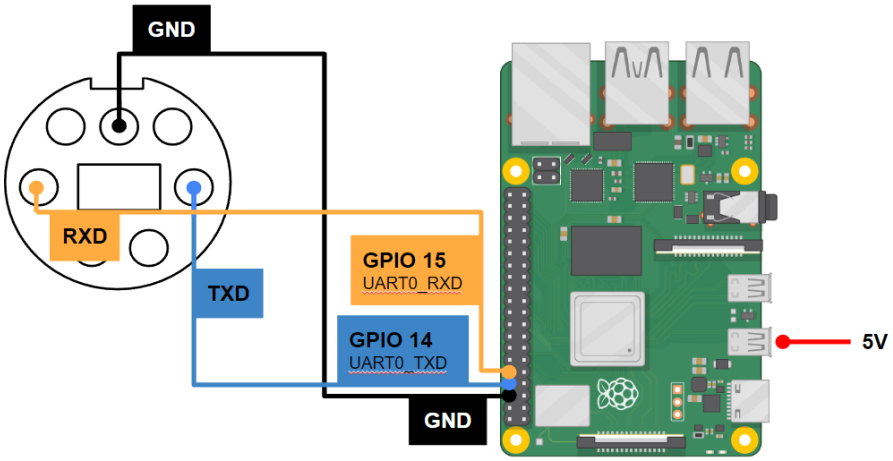


Figura 3.15: Diagrama de conexión entre el robot y la raspberry

Por defecto, el puerto serie en la raspberry está reservado para el Bluethooth, sin embargo, como no es necesario utilizarlo, cambiando los ajustes internos de la raspberry mediante la herramienta *raspi-config* se puede activar la comunicación serie en los pines GPIO_14 (RX) y GPIO_15 (TX).

A continuación, se va a explicar cómo se envían comandos por puerto serie al robot. En primer lugar, el robot se comunica por defecto con un baudrate de 115200 (se puede cambiar si se desea). En segundo lugar, el robot posee 3 modos, *passive*, *safe* y *full*. En el modo *passive* únicamente se puede acceder a las medidas de los sensores, no se pueden enviar comandos de control de comportamiento. En el modo *safe* se tiene control completo del robot pero contiene medidas de seguridad que si ocurren harán que el robot pase a modo *passive*. Estas medidas son que se detecte un acantilado, que se detecte el levantamiento de una rueda o que el cargador esté conectado. En el modo *full* se tiene control completo del robot y se ignoran las medidas de seguridad mencionadas anteriormente. Este último modo es el que se va a utilizar. Los comandos que se envían al robot deben comenzar con un byte propio de la instrucción y posteriormente se enviarán los bytes restantes que configuran esa instrucción. Por ejemplo, el comando *Drive Direct* constaría de los siguientes bytes: [145] [Velocidad rueda derecha byte 1] [Velocidad rueda derecha byte 2] [Velocidad rueda izquierda byte 1] [Velocidad rueda izquierda byte 2]. Esto quiere decir que [145] es el byte propio del comando *Drive Direct* y el resto de bytes determinan la velocidad de cada rueda. Cabe destacar que la instrucción se debe enviar completa, es decir, cuando se le envía [145] al robot, se queda esperando a recibir el resto de bytes de esa instrucción. Si se envían de forma incompleta seguirá esperando hasta recibirlos o quedarse sin batería.

3.4 Reconstrucción del entorno

Al ser un entorno tan grande (25x6 metros), se hizo una reconstrucción en casa y en la empresa de AyCN.

Para la reconstrucción del entorno en casa, como no se disponía de los recursos suficientes, se colocó la cámara en un trípode encima de un piano de pared para estar lo más cercana al techo posible y tener así más rango de visión, como se puede ver en la siguiente imagen:

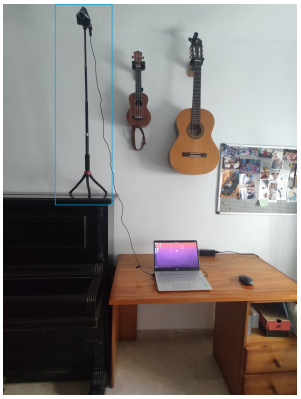


Figura 3.16: Reconstrucción del entorno en casa

El rango de visión conseguido es el que se puede apreciar en la 3.19. El problema que surge con esta reconstrucción es que al no estar la cámara perpendicular al plano del suelo, la calibración es más complicada.

Sin embargo, en la reconstrucción realizada en AyCN sí que se pudo colocar la cámara perpendicular al plano del suelo. Además, también se pusieron en el suelo las mallas por las que navegará el robot. En las siguientes imágenes se puede apreciar la reconstrucción:

Figura 3.17: Reconstrucción del entorno en AyCN

El rango de visión que se tiene es mayor que el de la reconstrucción hecha en casa y es el siguiente:

Figura 3.18: Rango de visión en AyCN

3.4.1 Calibración

Para controlar el robot mediante la imagen de la cámara, se intentó calibrar el entorno. De esta manera se puede obtener la posición del robot en el mundo real a partir de su posición (en píxeles) en la imagen.

Para obtener los parámetros intrínsecos y extrínsecos de la cámara, se utilizó el método

del *chessboard* que se ha visto en la carrera [7] y posteriormente se colocaron varios Apriltags en el suelo (ya que era fácil identificar su centro en la imagen) y se anotaba su posición (en píxeles) en la imagen, su posición en el mundo real y la distancia hasta la cámara. Con estos datos se pueden obtener los parámetros intrínsecos y extrínsecos de la cámara y realizar la calibración. En las siguientes imágenes se puede ver el proceso:

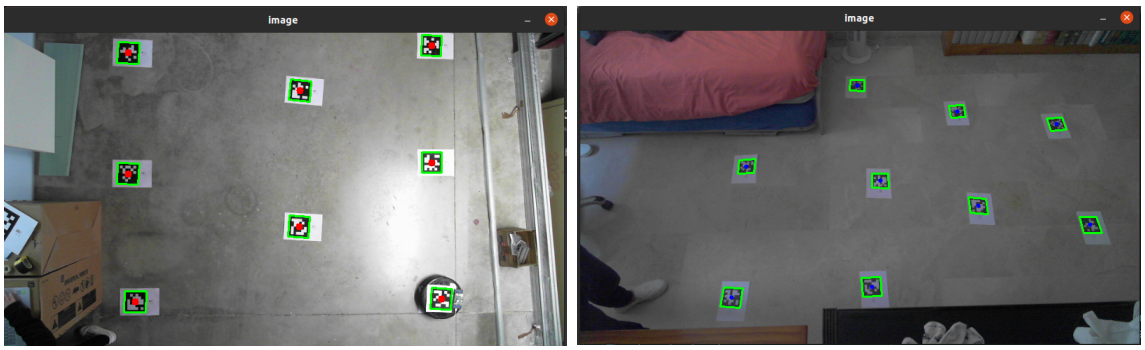


Figura 3.19: Calibración de la cámara en ambas reconstrucciones del entorno

El resultado que se consiguió con la calibración no era lo suficientemente bueno. El error en la calibración realizada en casa era bastante notorio. Sin embargo, el error en la calibración realizada en AyCN (3-5 cm) permitía poder trabajar con la posición del robot en coordenadas del mundo real, ya que la precisión para la tarea de depositar pegamento no requería mayor exactitud. A pesar de ello, se trabajó con las coordenadas de la imagen en píxeles ya que servía para las dos reconstrucciones del entorno y de esa manera se podía trabajar en las dos.

3.5 Control del robot

En este apartado se van a explicar los controladores realizados para controlar el movimiento del robot, desde los más básicos como mandar instrucciones por tiempo o actuar en función de la lectura de los encoders hasta el controlador proporcional que consigue controlar el robot a la perfección.

3.5.1 Instrucciones por tiempo

En la interfaz de comandos del iRobot Create 2 se encuentra un comando que permite controlar la velocidad y la dirección de giro del robot. El comando es *Drive*, cuyo *Opcode* (byte característico) es el 137. Este comando recibe 4 bytes de 16 bits con signo codificados en complemento a dos. Los dos primeros se utilizan para codificar la velocidad y los dos siguentes para el radio de giro. Las propiedades del comando son las siguientes:

- Secuencia: [137] [Velocidad byte alto] [Velocidad byte bajo] [Radio byte alto] [Radio byte bajo]
- Rango de la velocidad: (-500, 500 mm/s)

[7] Calibración de cámara mediante *chessboard* documentación oficial de OpenCV

- Rango del radio (-2000, 2000 mm)

A continuación, se muestra el valor de los bytes del radio para los casos básicos:

- Moverse en línea recta: 32768 o 32767 → 0x8000 o 0x7FFF
- Girar en el sentido de las agujas del reloj: -1 → 0xFFF
- Girar en el sentido contrario a las agujas del reloj: 1 → 0x0001

El envío de este comando supone que el robot se quede realizando el movimiento enviado de forma indefinida. Por lo tanto, si el comando se codifica de forma que se mueva en línea recta a una velocidad de 200 mm/s, el robot cumplirá esa orden hasta que se ordene la parada. Si se ordena la parada tras 2 segundos de haberse ejecutado el comando, el robot debería haber avanzado 400 mm, ya que va a una velocidad de 200 mm/s durante de 2 segundos.

Se hicieron varias pruebas con este método pero evidentemente los resultados no fueron nada buenos. Además teniendo en cuenta el efecto del desgaste de la rueda izquierda que provoca que haga un pequeño arco al moverse en línea recta, por lo que este método es completamente inservible. Sin embargo, sí que fue útil para hacer pruebas con las mallas sobre las que debe navegar el robot y para familiarizarse con la interfaz en aspectos como:

- Cómo codificar bytes para las instrucciones.
- Cómo comenzar y terminar la comunicación. Una vez se entra en el modo *full* el robot ignora los métodos de ahorro de batería, por lo que había que finalizar la comunicación volviendo al modo pasivo. De esta manera, si pasaban 5 minutos sin que el robot recibiera un comando se apagaba para no consumir la batería.
- En general, como funcionaba el envío de comandos al robot.

También fue útil para utilizar el ordenador como *master* en lugar de la raspberry. Primero se elaboró la interfaz en la raspberry y se comprobó su funcionamiento. Posteriormente, se elaboró en el ordenador y se enviaba a la raspberry la información necesaria para que ésta enviase al robot un comando u otro.

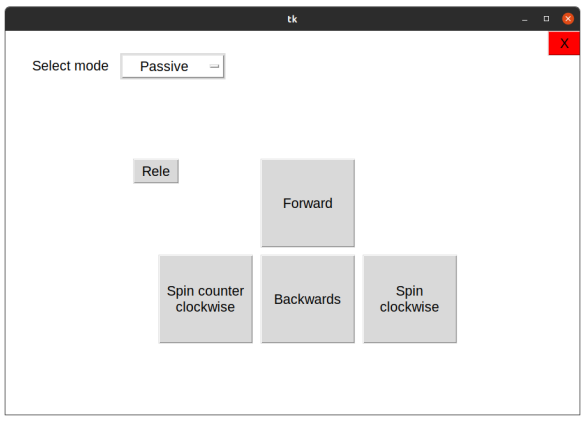


Figura 3.20: Interfaz de control del robot

3.5.2 Movimiento según la lectura de los encoders

En la interfaz de comandos se encuentra un comando que devuelve la lectura de los encoders de las ruedas del robot, por lo que se puede saber cuánto se ha movido cada rueda. Para convertir la medida de los encoders, N, en distancia recorrida hay que aplicar la siguiente fórmula:

$$distancia = N \cdot \frac{72\pi}{508.8}$$

Esta fórmula la proporciona la interfaz de comandos y 508.8 [vueltas/revolución] es el número de vueltas por revolución del encoder y 72 [mm] es el diámetro de las ruedas.

Para comprobar la funcionalidad de la medida devuelta por los encoders se realizó una interfaz gráfica en la que se dibuja una trayectoria a partir de la selección de varios puntos. Una vez dibujada, el robot realiza la trayectoria utilizando la medida devuelta por los encoders. De esta manera, se puede comprobar como la acumulación de error hacía inviable el uso de este método para controlar el robot.

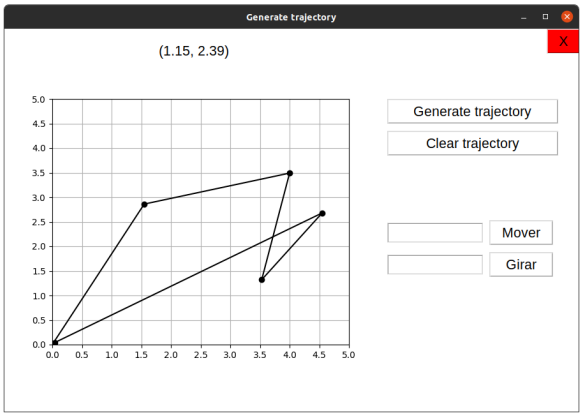


Figura 3.21: Interfaz de realización de trayectoria utilizando los encoders

En la imagen anterior se ve que se ha dibujado una trayectoria que comienza en el punto (0, 0), pasa por 4 puntos más y vuelve al (0, 0). En las pruebas que se realizaron siempre se partía y acababa en el mismo punto, de esa forma podía verse más fácilmente la acumulación del error (ya que debía terminar donde había empezado).

El programa funciona de manera que la posición (0, 0, 0) [x, y, theta] es la posición del robot al empezar el programa. Una vez dibujada la trayectoria, calcula el ángulo que tiene que girar el robot para orientarse hacia el siguiente punto y la distancia a la que se encuentra. Posteriormente, el robot gira hasta que las lecturas de los encoders indican que ha girado los grados deseados y después se mueve en línea recta hasta que las lecturas de los encoders indican que se ha recorrido la distancia deseada. Sin embargo, la acumulación de error hacía imposible este control. El punto de comienzo del robot en el mundo real y el último punto de la trayectoria (que son el mismo punto) eran muy diferentes.

3.5.3 Controlador todo-nada

Este controlador es el primero que incorpora la utilización de la cámara para determinar la pose del robot mediante la identificación Apriltag colocado encima del robot. Dado un punto destino, se determina el ángulo que debe tener el robot para quedarse orientado hacia el punto y gira hasta que con la cámara se detecte que se ha alcanzado esa orientación. Posteriormente, el robot se mueve en línea recta hasta que con la cámara se determina que se ha alcanzado el punto destino. Es la misma filosofía que el control mediante encoders pero utilizando la cámara en su lugar.

Sin embargo, el robot puede salirse de la trayectoria debido a una mala orientación inicial hacia el punto o debido al desgaste de la rueda izquierda que provoca que trace un pequeño arco al moverse en línea recta, por lo que hay que comprobar ese factor. Si ocurre, se vuelve a orientar y comienza a moverse en línea recta de nuevo. Para ello, se da un rango en el que se determina si el robot se ha salido de la trayectoria y debe recalcularla. En la siguiente imagen se muestra una representación visual de este controlador:

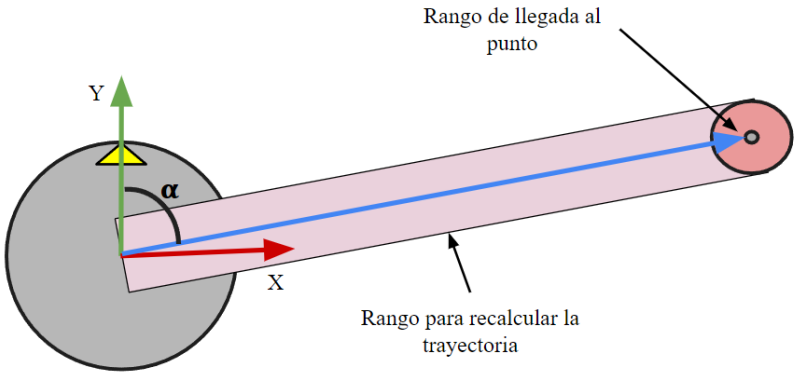


Figura 3.22: Representación del controlador

En la imagen anterior, el robot está orientado con un ángulo de 90º y necesita girar un ángulo α para quedar orientado hacia el punto destino. Cuando la cámara detecte que la orientación es correcta comenzará a moverse en línea recta. Mientras no se salga del área rosa el robot seguirá moviéndose en línea recta, eso significa que está en la trayectoria. Cuando el centro del robot esté fuera de ese área, se habrá salido de la trayectoria, por lo que se recalculará el ángulo y volverá a ejecutar el controlador hasta alcanzar el punto deseado.

El controlador consigue que el robot llegue al punto deseado, sin embargo, para llegar con una precisión y un tiempo razonable no es válido. Al disminuir el área de la trayectoria (y por tanto mejorar la precisión) el robot se salía continuamente de la trayectoria y la recalculaba de nuevo, y así sucesivamente. Podía tardar alrededor de 1 minuto para alcanzar un punto a 1 metro de distancia, y recalculaba la trayectoria demasiadas veces. Por lo tanto este controlador, a pesar de que alcance el punto deseado, no cumple con los resultados deseados.

3.5.4 Controlador proporcional

Este controlador es el más sofisticado que se ha realizado y el que consigue completar la tarea cumpliendo con los requisitos de precisión y tiempo deseados. Es un controlador únicamente proporcional porque a pesar de que podría haber sido PID, no ha hecho falta la inclusión de la acción integral ni la de la derivativa para conseguir su cometido. La acción del controlador provoca que el error existente entre el ángulo del robot y el ángulo necesario para llegar al punto deseado vaya disminuyendo y se mantenga lo más cercano a 0 posible. Este controlador se basa en el modelo del vehículo diferencial para controlar la velocidad de cada rueda del robot por separado (el comando *Drive Direct* explicado en el apartado 3.3.3 permite este control de las ruedas por separado). La velocidad de cada rueda viene determinada por las siguientes fórmulas:

$$v_L = \frac{2v - wL}{2r}$$

$$v_R = \frac{2v + wL}{2r}$$

En las fórmulas anteriores, v es la velocidad lineal, w la velocidad angular, L la distancia entre ruedas (235 mm) y r el radio de las ruedas (36 mm). La velocidad lineal se calcula de forma que sea proporcional a la distancia al objetivo. De esta manera, cuanto más cerca se encuentre de él, más lento irá. Con esta acción se consigue que se mueva a una velocidad adecuada durante todo el transcurso de la trayectoria. La velocidad angular se calcula a partir de la acción proporcional en la que se multiplica el error en el ángulo por una constante.

$$v = K_v \sqrt{(x_d - x)^2 + (y_d - y)^2}$$

$$e = \arctan \frac{y_d - y}{x_d - x} - \theta$$

$$w = K_w \cdot e$$

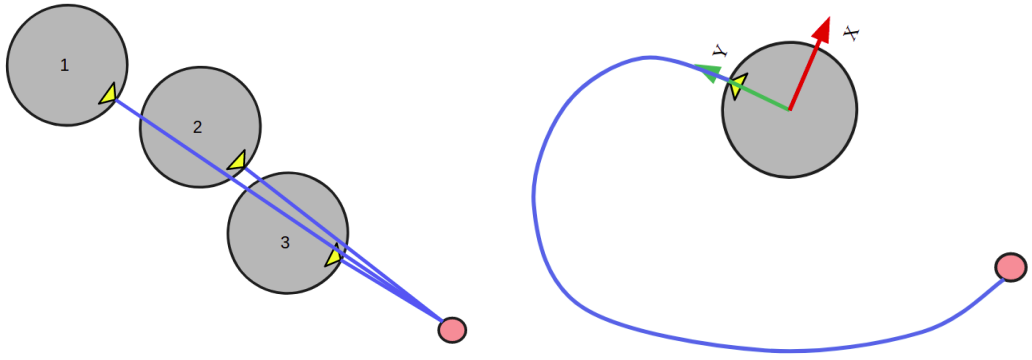
En las fórmulas anteriores, K_v es la constante de velocidad lineal y K_w la constante de la velocidad angular. La pose actual del robot es $[x, y, \theta]$, la posición deseada es $[x_d, y_d]$ y e es el error entre el ángulo deseado y el ángulo del robot.

Ajustando de formas distintas las constantes K_v y K_w se consiguen distintos comportamientos:

- $K_v >> K_w$, el robot se moverá hacia delante mientras se orienta trazando un arco (relativamante grande).
- $K_v << K_w$, el robot girará en el sitio y una vez que esté orientado se moverá hacia delante.
- $K_v = K_w$, es el caso intermedio entre los dos casos anteriores, el robot se moverá en línea recta mientras gira pero el arco que trace será menor que en el primer caso.

El ajuste de estas variables depende del tipo de movimiento que se quiera conseguir. En los primeros ajustes realizados la K_w era mayor, provocando que el robot girase en el sitio y luego se moviera en línea recta. Sin embargo, una vez empieza el movimiento en línea recta, el robot se puede desviar y que el error aumente. La acción del controlador para corregirlo

provoca que el robot se detenga y se oriente de nuevo (situación 1). Esto es debido al ajuste con la K_w alta y provoca un aumento en el tiempo de llegada al punto. Por otro lado, si la K_v es mayor el empieza a moverse mientras corrige el error. En puntos cercanos de distancia pero en los que se ha de girar mucho, el robot hace un arco demasiado grande recorriendo una gran distancia, por lo que no es eficiente (situación 2). En las siguientes imágenes se muestran estas dos situaciones.



(a) Situación 1 **(b)** Situación 2

Figura 3.23: Casos de movimiento del robot que se quieren evitar

Para evitar estas situaciones se han ajustado de forma dinámica en función del error existente en el ángulo. Cuando el error en el ángulo es grande, K_w es mayor que K_v para que gire en el sitio, esto evita la situación de trazar un arco grande, y cuando alcanza un umbral de error específico, la K_v aumenta y la K_w disminuye, esto evita que se pare a rectificar mientras está navegando en línea recta. De esta manera, se produce un movimiento rápido para orientarse y suave mientras sigue la trayectoria. Además, el tiempo empleado llegar al punto es correcto así como la trayectoria que realiza.

Cabe destacar que el controlador se ha diseñado de forma que cumpla con los 3 principales aspectos de navegación para la tarea: ir a un punto aislado, ir a un punto con una orientación y recorrer una trayectoria. Como la tarea del robot consiste en depositar pegamento en un punto, el robot debe contar con un aplicador de pegamento. Para ello, se simuló la presencia de un aplicador en la parte de atrás del robot con un trozo de cartón rojo, como se pude ver en la figura 3.7. Es con ese cartón rojo con lo que el robot debe alcanzar en punto, y no con su centro. Para llevar a cabo esta tarea, se puede hacer de varias maneras, la opción por la que se ha optado es añadir un punto auxiliar a la trayectoria que provoque que el robot llegue orientado hacia el punto. En las siguientes figuras se puede comprobar el funcionamiento:

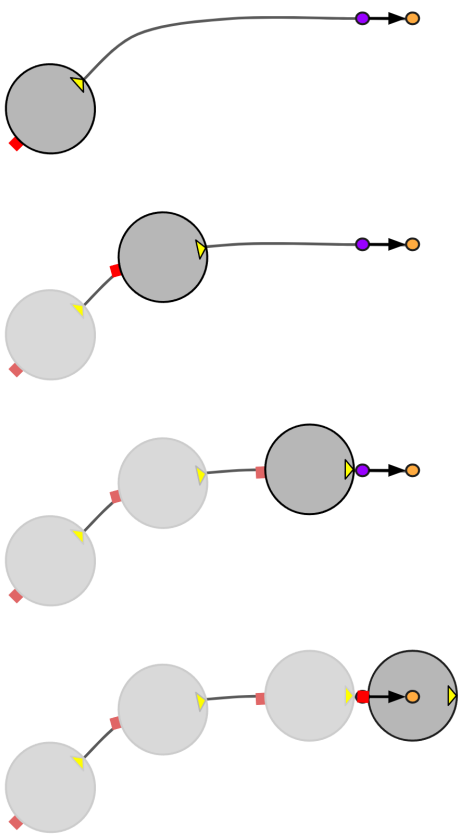


Figura 3.24: Ejemplo de funcionamiento para llegar a un punto con el aplicador de pegamento

En la representación anterior el punto morado es la localización donde se quiere poner el pegamento y el punto naranja se encuentra a una distancia del punto morado igual a la distancia entre el aplicador y el centro del robot. De esta manera, pasando por el punto morado y después por el punto naranja, el aplicador coincide con el punto morado cuando el centro del robot alcanza el punto naranja. Con este método el robot puede poner pegamento en cualquier punto.

El cálculo del punto auxiliar (punto naranja) depende de si la tarea consiste en ir a un punto aislado o en completar una trayectoria. En el caso de ir a punto aislado no importa dónde colocarlo, aunque lo más eficiente es colocarlo siguiendo la recta que traza el punto origen del robot y el punto destino. En el caso de calcular el punto auxiliar en una trayectoria en la que hay más puntos, se ha de colocar en la recta que forma el punto actual de la trayectoria y el punto siguiente:

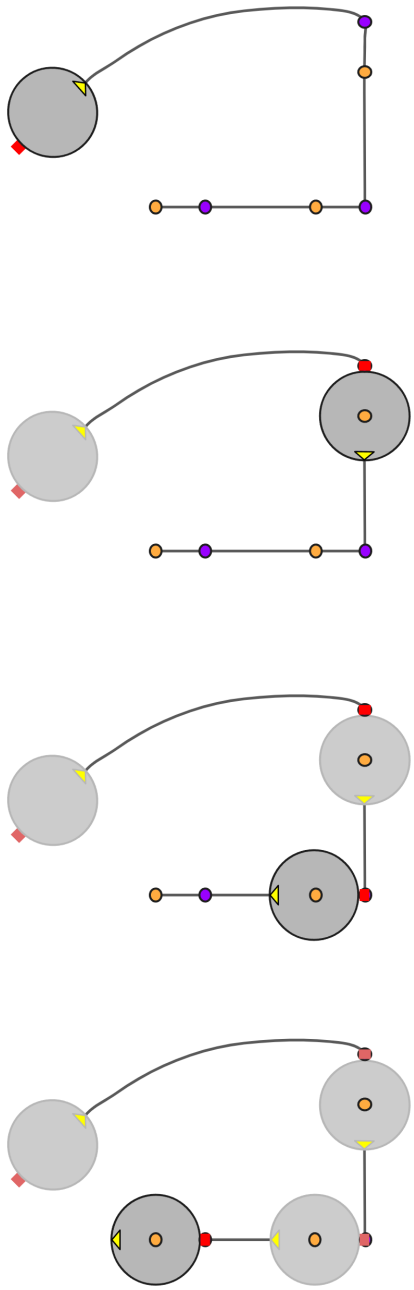


Figura 3.25: Ejemplo de cálculo del punto naranja en una trayectoria

Como se puede ver en la imagen anterior, hay tres puntos morados. El primer punto naranja se coloca entre el primero y el segundo, es decir, con un ángulo de -90º. La distancia a la que se coloca se ha comentado anteriormente, se coloca a la misma distancia que la distancia entre el centro del robot y el aplicador. El siguiente punto naranja se coloca entre el segundo

y el tercero morado y el ángulo es de 180º. Para el último punto naranja, como no hay un cuarto punto morado, se coloca con el mismo ángulo que el punto anterior (180º), de esta forma sigue el movimiento anterior.

Este método y el ajuste dinámico de las constantes K_v y K_w permiten que el robot sea capaz de depositar pegamento en puntos aislados y en trayectorias. Cabe destacar que funciona tanto para trayectorias en las que los puntos están poco distanciados entre sí como para trayectorias en las que están más alejados.

En las siguientes imágenes se muestra cómo se ven los puntos en una ejecución en el entorno reconstruido:

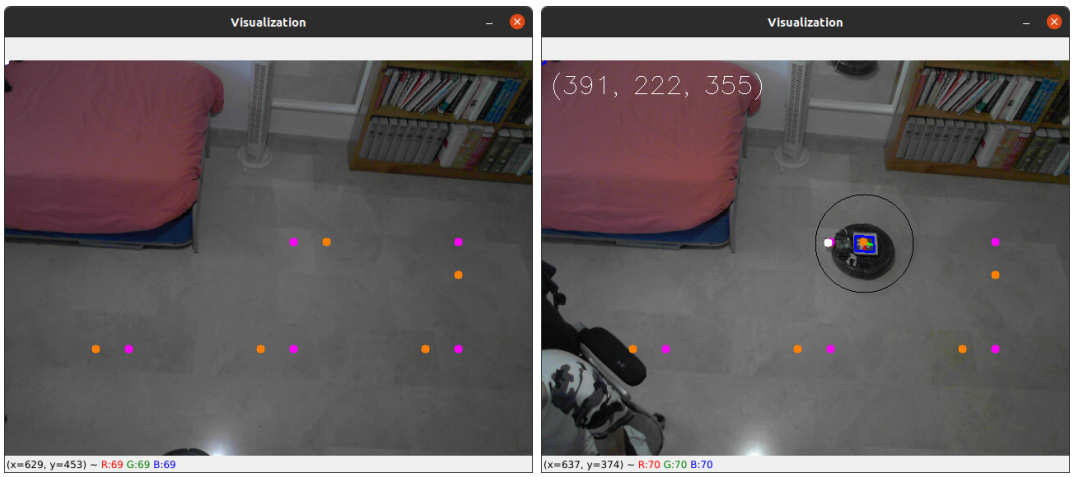


Figura 3.26: Puntos generados para cumplir con la trayectoria

4 Resultados

En este apartado se detallarán los resultados que se han conseguido con el control del robot, teniendo en cuenta aspectos tanto de localización como control de movimento.

En primer lugar, la posición del robot se obtiene mediante los Apriltags. Como se ha podido ver en las figura 3.13 la detección de la pose en píxeles se consigue a pesar de que el ángulo es complicado. Si se hacen las mismas pruebas con el Apriltag colocado en el robot funcionando en el entorno reconstruido, los resultados son similares. Como se puede ver en las siguientes imágenes, la pose del robot se detectada en posiciones complicadas de la cámara y por lo tanto, en todo el área de trabajo:

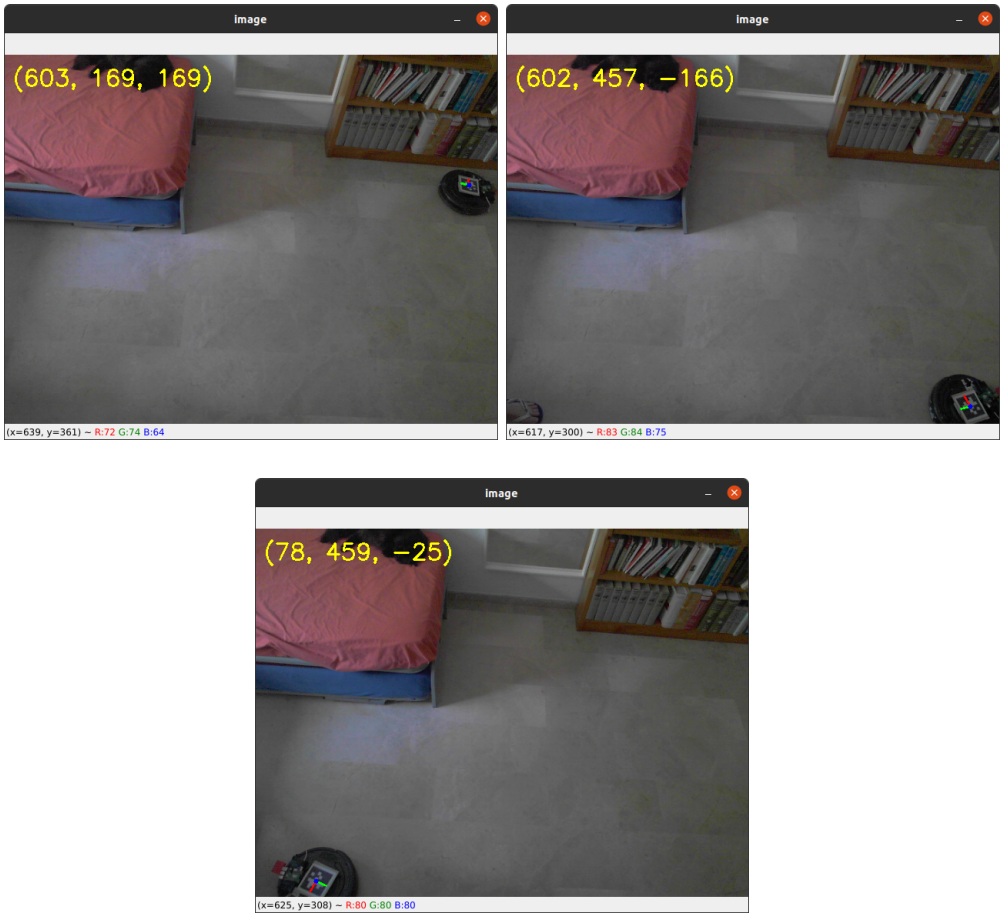


Figura 4.1: Detección de la pose del robot con el Apriltag

A pesar de que la detección de la posición en píxeles es precisa, la posición en el mundo real obtenida a partir de la calibración realizada no es lo suficientemente exacta. En la calibración realizada en casa puede deberse a que la cámara no está situada perpendicular al plano del suelo. Por otro lado, la calibración en AyCN en la que la cámara sí que está perpendicular al plano proporcionaba medidas con un error de 3 a 5 centímetros. Este error podría permitir trabajar en píxeles y coordenadas del mundo real, calculando previamente el punto (en píxeles) donde hay que colocar el pegamento. Es decir, teniendo la localización del punto objetivo en coordenadas del mundo real, obtener su posición en el plano de la imagen a partir de la calibración, lo que supondría un error de 3 a 5 cm con el que se podría trabajar. A pesar de ello, en la reconstrucción en casa el error es notablemente mayor y varía en función de la situación del robot (debido al ángulo de la cámara) por lo que hace imposible trabajar con coordenadas del mundo real. Por ello, se decidió trabajar en píxeles en ambas reconstrucciones del entorno para asegurar el funcionamiento de control del robot.

En cuanto al control del robot, durante este proyecto se han realizado varios controles: instrucciones por tiempo, controlador según las medidas de los encoders, controlador todo-nada y controlador proporcional. Únicamente el controlador proporcional ha conseguido el control del robot. A continuación, se muestra una breve explicación sobre los demás controladores y por qué no funcionaron:

- Instrucciones por tiempo. Este controlador consiste en mandar comandos durante un tiempo determinado y comprobar si el resultado es el esperado. Por ejemplo, girar en el sentido de las agujas del reloj a una velocidad de 45º/s durante 2 segundos debería resultar en un giro de 90º con respecto al ángulo inicial. Sin embargo, los resultados nunca eran precisos y variaba el error en cada ejecución. Esto se debe a múltiples factores en los que interviene el desgaste de las rueda izquierda, el rozamiento con el suelo, la velocidad con la que se ejecute el giro, etc. A pesar de que es inservible, se recuerda de que sirvió para aspectos de interés para el proyecto como familiarizarse con la interfaz de comandos, probar el movimiento sobre las mallas de infusión o establecer el PC como *master* y la raspberry como *slave*.

- Controlador según las medidas de los encoders. Este controlador consiste en realizar una instrucción genérica y detenerla cuando la lectura de los encoders indique que se ha cumplido. Por ejemplo, para girar 90º en el sentido contrario a las agujas del reloj, se envía al robot la instrucción del giro y se accede a la lectura de los encoders. Una vez muestren que el giro realizado es de 90º se envía una instrucción de parada. Este controlador es mucho más preciso en los giros que el anterior, y el error que aportaban los encoders no era muy elevado. Sin embargo, la acumulación de este error provoca que el funcionamiento para varias instrucciones seguidas no sea correcto. Es decir, para un giro aislado el error es pequeño y el resultado es bueno, pero para varios giros durante una trayectoria, la acumulación de error es notable. Por tanto, este controlador tampoco sirve para el control del robot.

- Controlador todo-nada. Este controlador incorpora el control por visión. Es la misma filosofía que el anterior pero se sustituye la lectura de los encoders por la lectura de la posición del robot a partir de la cámara. Es decir, para el ejemplo anterior en el que hay que girar 90º, se ejecuta la instrucción de girar hasta que la orientación del

robot obtenida en la imagen de la cámara sea la orientación a conseguir. Con un punto destino funciona de la misma manera, el robot se orienta hacia el punto y se dirige en línea recta a él. En todo momento, el controlador comprueba que el robot no se salga de la trayectoria y en caso de que lo haga ejecuta de nuevo la orden de orientarse y posteriormente la de navegar en línea recta. Este controlador funcionaba con un error en la trayectoria alto, si se quería disminuir el controlador no era eficiente ya que se salía de la trayectoria demasiadas veces durante el movimiento, lo que lo hacía inservible.

El controlador proporcional sí que consiguió controlar el movimiento del robot con trayectorias suaves y eficientes y con una precisión muy buena. Este controlador se basa en el control hacia un punto de un robot diferencial en el que se calcula la velocidad de cada rueda a partir de la velocidad lineal y angular, como se explicó en el apartado 3.5.4. En este control intervienen dos variables, K_v y K_w, que determinan el movimiento del robot durante la trayectoria. Por un lado, incrementar K_v dota de más importancia a la velocidad lineal, por otro lado, incrementar K_w dota de más importancia a la velocidad angular. Un mal ajuste de estas variables provoca que el controlador no sea eficiente para alcanzar el punto destino o que nunca lo llegue a alcanzar. Por ejemplo, un mal ajuste de la K_v provoca que el robot haga un arco demasiado grande durante la trayectoria para orientarse provocando recorrer mucha distancia para llegar al punto destino. Un mal ajuste de la K_w provoca que el robot gire en el sitio cuando pierde la orientación. Estas situaciones se quieren evitar y son las que se muestran en la figura 3.23. El ajuste de estas variables se realizó de forma que el robot pueda navegar a puntos aislados o realizar trayectorias con puntos muy distanciados o poco distanciados. A continuación se muestra una prueba de funcionamiento en la que el robot completará una trayectoria. Esta trayectoria consta de puntos amarillos y rojos, en los puntos amarillos hay que depositar el pegamento, por lo que el aplicador (representado como un punto blanco en el robot) debe coincidir con ellos. Los puntos rojos son puntos que se calculan para que el aplicador coincida con los puntos amarillos. Esto está explicado con mayor detalle en el apartado 3.5.4.

Como se ve en la figura 4.2, el controlador proporcional permite que el robot realice trayectorias de manera suave y eficiente gracias al ajuste de las constantes K_v y K_w. En este caso, la trayectoria contiene 26 puntos (13 amarillos y 13 rojos) que están poco distanciados entre sí. Si se mantienen únicamente los puntos amarillos de las esquinas (y sus respectivos puntos rojos) la trayectoria pasa a tener 8 puntos (4 amarillos y 4 rojos) y el funcionamiento es igual de correcto. Cabe destacar con este diseño del controlador el robot no para en cada punto amarillo para depositar el pegamento, sino que es capaz de navegar sin detenerse pasando por ellos. Por lo tanto, hace mucho más eficiente la deposición de pegamento.

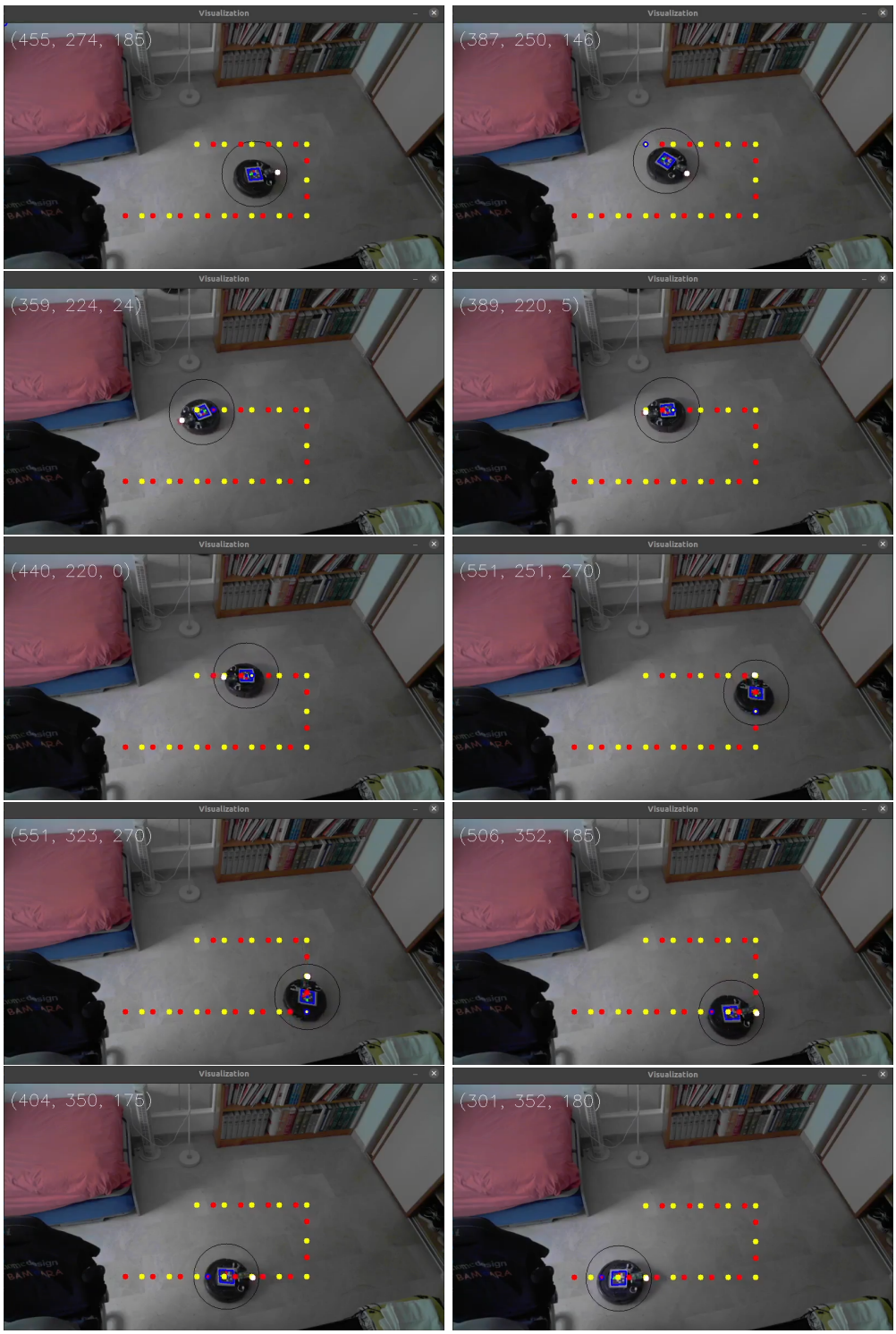


Figura 4.2: Prueba de funcionamiento del controlador proporcional

5 Conclusiones

Para concluir con el proyecto se va a comprobar si se han conseguido los objetivos del proyecto y en qué medida, recordando los objetivos:

1. Selección de un robot móvil adecuado para la tarea.
2. Desarrollo de un sistema de control de útil y extrapolable.
3. Desarrollo de un controlador de movimiento del robot.
4. Creación de un prototipo de solución.
5. Mostrar el potencial de incorporar un robot móvil al proceso en cuestión.

En cuanto al primer objetivo, se considera que el robot que se ha escogido es idóneo para el proyecto aunque no para la aplicación real de la tarea debido a sus desperfectos. Tiene problemas de desgaste en las ruedas, sensores estropeados (que en el proceso real pueden llegar a utilizarse). No puede soportar mucha carga y tiene una geometría complicada para incluir sistemas en él (como el aplicador de pegamento). Además, en un futuro puede que aparte de realizar la deposición de pegamento, también lo prense, lo que provocará colocar una herramienta en él para prensar (volviendo a la desventaja del poco soporte de carga y la geometría complicada). Sin embargo, teniendo en cuenta que el robot es un robot aspiradora es lógico que no cuente con esas características. Por otro lado, ha tenido un coste nulo y se tuvo disponibilidad de él nada más solicitarlo (lo que no suele pasar con otros robots que hay que comprar). Además, las pruebas que se han podido realizar con él son suficientes y muestran el potencial de incorporar al proceso industrial una solución con un robot móvil, que es otro de los objetivos del proyecto.

En cuanto al sistema de control planteado, se considera que es el adecuado para la empresa a la que va destinado. La empresa ya cuenta con una cámara situada perpendicularmente al plano de la mesa que se podría utilizar para esta solución, además, en las pruebas se ha podido comprobar que los elementos utilizados en el sistema (ordenador, microcontrolador, cámara y robot) son suficientes para llevar a cabo la tarea. Por lo tanto, no son necesarios grandes cambios en el proceso actual para incorporar la solución planteada.

En cuanto al controlador de movimiento del robot, se ha diseñado un controlador proporcional que es capaz de controlar al robot a partir de la imagen de la cámara, control visual. Con él se pueden realizar trayectorias suaves y eficientes para depositar pegamento. Además, no es necesario que el robot esté parado para depositar el pegamento, puede hacerlo mientras recorre la trayectoria. El funcionamiento del controlador se ha podido comprobar en el apartado 4.

En cuanto a los dos últimos objetivos, se considera que el prototipo de solución creada a partir del iRobot Roomba 669, una cámara web y una raspberry muestra el potencial de añadir al proceso industrial un robot móvil para llevar a cabo la deposición de pegamento.

Bibliografía

Aguilera Hernández, M. I., Bautista Miguel, A., y Iruegas, J. (2007). Diseño y control de robots móviles. *Instituto Tecnológico de Nuevo Laredo*.

Corke, P., y Hutchinson, S. (2000). Real-time vision, tracking and control. En *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)* (Vol. 1, p. 622-629 vol.1). doi: 10.1109/ROBOT.2000.844122

Cruz, H. (2008). *Una introducción a los robots móviles.*

Garcia, G. J., Pomares, J., y Torres, F. (2004). *Control visual de robots manipuladores. una herramienta para su diseño y aprendizaje.*

García Sánchez, J. R., Silva Ortigoza, R., y Barrientos Sotelo, V. R. (2007). Robots móviles: Evolución y estado del arte. *Polibits*.

Lippiello, V., Siciliano, B., y Villani, L. (2005). Eye-in-hand/eye-to-hand multi-camera visual servoing. En *Proceedings of the 44th ieee conference on decision and control* (p. 5354-5359). doi: 10.1109/CDC.2005.1583013

Ollero Baturone, A. (2001). *Robótica. manipuladores y robots móviles*. MARCOMBO, S.A.