CUADERNO DE LABORATORIO

Sara Rodríguez Fernández

2023/2024

PLANNING DE TAREAS DE LOS MÓDULOS CLAVE DEL PROYECTO

|  |  |
| --- | --- |
| Nº | Descripción |
| 1 | Entrenamiento de la trayectoria del robot y captura con OptiTrack   * 1. El proyecto utilizará el sistema de cámaras OptiTrack para realizar el aprendizaje de la trayectoria de recogida de piezas.   2. El sistema tomará las posiciones y orientaciones de la trayectoria y elaborará un programa en RoboDK desde Matlab usando la API de RoboDK para Matlab.   3. El sistema ejecutará desde Matlab el programa de RoboDK . |
| 2 | Entrenamiento e integración de la red neuronal de distinción de piezas   * 1. Se empleará una cámara para tomar imágenes de las piezas y elaborar una base de datos.   2. Con las imágenes guardadas se entrenará una red neuronal en Matlab.   3. La red distinguirá piezas rectangulares y cudradas y si son válidas o piezas rotas. |
| 3 | Mapeado, navegación e integración del Turtlebot.   * 1. El robot mapeará el entorno de trabajo y se utilizará ese mapa para la navegación.   2. El robot usará el mapa elaborado para navegar a las posiciones objetivo |

Índice sesiones

[ Organización del trabajo 4](#_Toc157426217)

[ Explorar el uso de RoboDK 5](#_Toc157426218)

[ Explorar la Realsense 5](#_Toc157426219)

[ Explorar la cámara RGB 5](#_Toc157426220)

[ Explorar la cámara RGB 6](#_Toc157426221)

[ Conexión de Matlab con RoboDK y el UR3e 7](#_Toc157426222)

[ Conexión de Motive con Matlab, que se conecta con RoboDK y el UR3e 8](#_Toc157426223)

[ Conexión de Motive con Matlab, que se conecta con RoboDK y el UR3e 8](#_Toc157426224)

[ Conexión de Motive con Matlab, que se conecta con RoboDK y el UR3e 9](#_Toc157426225)

[ Conexión de Motive con Matlab, que se conecta con RoboDK y el UR3e y pruebas Machine Learning 10](#_Toc157426226)

[ Conexión de Motive con Matlab, que se conecta con RoboDK y el UR3e y pruebas Machine Learning 11](#_Toc157426227)

[ Conexión de Motive con Matlab, que se conecta con RoboDK y el UR3e y pruebas ROS 13](#_Toc157426228)

[ Conexión de Motive con Matlab, que se conecta con RoboDK y el UR3e y pruebas ROS 14](#_Toc157426229)

[ Mapeado y navegación con Turtlebot4 15](#_Toc157426230)

[ Configuración para el movimiento de recogida de piezas 17](#_Toc157426231)

[ Configuración de la red para integrar el UR3e y el Turtlebot 19](#_Toc157426232)

[ Optimización del movimiento de recogida de piezas 20](#_Toc157426233)

[ Optimización del movimiento de recogida de piezas 21](#_Toc157426234)

[ Reparación del Turtlebot 21](#_Toc157426235)

[ Integración de todos los módulos en un solo programa 22](#_Toc157426236)

[ Integración de todos los módulos en un solo programa 23](#_Toc157426237)

[ Integración de todos los módulos en un solo programa 23](#_Toc157426238)

[ Optimización de la red neuronal e integración en el programa principal 24](#_Toc157426239)

[ Documentación del proyecto 24](#_Toc157426240)

|  |  |
| --- | --- |
| Organización del trabajo | 30/11/2023 |

Objetivo sesión: Organización del trabajo

El propósito fundamental de la presente sesión se centra en la definición y establecimiento de las diversas etapas que conformarán el desarrollo de nuestro trabajo. Para alcanzar este objetivo, se presenta un detallado diagrama de flujo que delineará de manera clara y estructurada cada una de las fases del proceso, que se puede ver en la Ilustración 1.

Diagrama

Descripción generada automáticamente

Ilustración . Diagrama de flujo del proyecto

Después de organizar el trabajo, el equipo se centra en explorar el uso de RoboDK. Al intentar ejecutar un ejemplo simple de la toolbox de Matlab para RoboDK, se descubre que es crucial abrir la simulación en RoboDK antes de ejecutar el código en Matlab para establecer una conexión exitosa entre ambas plataformas. Este hallazgo subraya la importancia de seguir una secuencia específica de pasos para lograr una integración efectiva entre RoboDK y Matlab en el proyecto.

|  |  |
| --- | --- |
| Explorar el uso de RoboDK | 02/12/2023 |

Objetivo sesión: Exploración de RoboDK

El propósito de esta sesión es adentrarse en el entorno de RoboDK y comprender la naturaleza de los comandos y datos utilizados para la programación en dicha plataforma.

Se lleva a cabo una investigación específica sobre cómo manejar los datos provenientes de Motive con el objetivo de programar el robot en RoboDK para replicar los movimientos capturados. Se destaca el descubrimiento de que el comando moveJ() en RoboDK tiene la capacidad de recibir matrices homogéneas, lo que facilita la programación al permitir la generación de dichas matrices homogéneas durante el proceso de programación.

|  |  |
| --- | --- |
| Explorar la Realsense | 04/12/2023 |

Objetivo sesión: Exploración de la cámara Realsense

En esta sesión, el objetivo principal es familiarizarse y explorar el funcionamiento de la cámara RealSense. Se busca obtener un conocimiento detallado sobre su funcionalidad, configuración y posibles aplicaciones. El propósito es proporcionar una base sólida para comprender y trabajar eficientemente con la cámara RealSense.

Se comienza descargando el software de la cámara, para ello se descarga la versión 2.53.1 de la aplicación RealSense, disponible en <https://github.com/IntelRealSense/librealsense/releases/tag/v2.53.1>.

Después de revisar los scripts del software de la cámara para Matlab, no se ha identificado una forma efectiva de trabajar con la cámara RealSense para capturar únicamente los colores, excluyendo la información de profundidad. La raíz de esta limitación podría residir en la configuración actual de la cámara RealSense. Ante esta situación, se propone la consideración de emplear una cámara RGB como una alternativa más simple y directa. Dado que para este proyecto no se requiere la complejidad de la cámara RealSense, esta opción podría ofrecer una solución más adecuada a los requisitos específicos del proyecto, simplificando así la captura de información visual y facilitando su integración en el flujo de trabajo.

|  |  |
| --- | --- |
| Explorar la cámara RGB | 04/12/2023 |

Objetivo sesión: Exploración de la cámara RGB

El objetivo de esta sesión es explorar la adquisición de datos utilizando una cámara RGB, la cual ofrece la capacidad de capturar información relacionada con colores y formas. La simplicidad de esta cámara permite una integración fluida en el proceso de desarrollo, facilitando la obtención de datos visuales para su posterior análisis y aplicación en el proyecto.

Para comenzar, se procede conectando la cámara RGB al ordenador mediante el cable USB. Seguidamente, se verifica la correcta instalación y configuración de la toolbox correspondiente en Matlab, diseñada específicamente para el trabajo con cámaras de vídeo. Este paso resulta esencial, ya que sienta las bases para establecer una conexión eficiente entre la cámara y el entorno de desarrollo. Esta conexión exitosa es crucial para facilitar la captura y procesamiento efectivo de datos visuales a lo largo de la ejecución del proyecto. Al realizar pruebas, se constata que la cámara RGB logra obtener imágenes a color adecuadas para los objetivos del trabajo. Como resultado, se toma la decisión de emplear esta cámara, que se puede ver en la, en lugar de la RealSense, optimizando así el enfoque visual del proyecto.

Imagen que contiene tabla, cerca, mostrador, vidrio

Descripción generada automáticamente

Ilustración . Cámara RGB

|  |  |
| --- | --- |
| Explorar la cámara RGB | 11/12/2023 |

Objetivo sesión: Exploración de la cámara para distinguir piezas válidas de piezas defectuosas

En esta sesión, el objetivo primordial es evaluar la capacidad de la cámara para generar una red neuronal basada en sus imágenes con el fin de distinguir piezas con y sin grietas.

El proceso se inicia utilizando el archivo de GitHub "script\_camara" para realizar comprobaciones exhaustivas y asegurar la captura adecuada de las imágenes. Durante las pruebas iniciales, se identifica que la sensibilidad de la cámara a la luz impacta directamente en la calidad de las imágenes. Con el propósito de optimizar la percepción de las formas de los objetos, se determina la necesidad de situar un fondo negro detrás de la pieza a capturar. Esta medida resulta crucial para mejorar la visibilidad y precisión en la detección de grietas en las piezas.

Sin embargo, al profundizar en el análisis, se nota que la definición de la cámara no alcanza niveles óptimos para captar con nitidez las grietas en las imágenes destinadas a la posterior clasificación.

La conclusión derivada de estas pruebas iniciales resalta la importancia de considerar no solo la iluminación, sino también la calidad y resolución de la cámara al implementar un sistema de clasificación de piezas basado en imágenes, por lo que se plantea la posibilidad de que fuese necesaria en otra cámara en las próximas fases del proyecto.

|  |  |
| --- | --- |
| Conexión de Matlab con RoboDK y el UR3e | 12/12/2023 |

Objetivo sesión: Crear un programa en RoboDK desde Matlab y enviárselo al robot.

El objetivo que se plantea es lograr la creación de un programa en RoboDK mediante un script de Matlab y su posterior envío al robot.

Inicialmente se prueba con un script obtenido de Internet. Sin embargo, no se consigue que el robot se mueva. Después de investigar en recursos en línea, se identificó que la solución podría residir en añadir la llamada robot.Connect() al código.

Con la inclusión de esta llamada en el script, se logró exitosamente el movimiento del robot directamente desde Matlab. Este paso aseguró la correcta comunicación entre Matlab y el robot, permitiendo la ejecución de comandos de manera efectiva.

Tras conseguir el objetivo de esta sesión se prepara el entorno del robot y las cámaras para realizar una captura inicial con el sistema OptiTrack.

En relación a las cámaras en el laboratorio, están integradas a un programa que facilita la visualización de puntos reflejados bajo la luz infrarroja. La interfaz de este programa permite asignar atributos a dichos puntos y agruparlos para formar sólidos rígidos. En nuestro experimento, optaremos por la utilización de tres marcadores para crear un sólido rígido. Esta elección se fundamenta en que el programa, al emplear tres marcadores, facilita la obtención de traslaciones y orientaciones expresadas en el formato de dato cuaternión.

|  |  |
| --- | --- |
| Conexión de Motive con Matlab, que se conecta con RoboDK y el UR3e | 13/12/2023 |

Objetivo sesión: Capturar movimientos con el sistema de cámaras y crear un programa en RoboDK desde Matlab para que el robot imite los movimientos capturados y enviárselo al robot.

Tras haber realizado una captura de movimientos inicial con el sistema OptiTrack, el objetivo que se plantea es lograr la creación de un programa en RoboDK mediante un script de Matlab para que el robot imite los movimientos capturados.

Se genera un script que utiliza los datos del sólido rígido capturado mediante el sistema OptiTrack. Este script tiene como objetivo la creación de matrices homogéneas para cada objetivo específico. Este enfoque se adopta dado que las instrucciones de movimiento en RoboDK son compatibles con matrices homogéneas. Sin embargo, al hacer una prueba inicial, se observa que los datos de Motive son erróneos, puesto que los movimientos son mucho más pequeños que los realmente realizados.

|  |  |
| --- | --- |
| Conexión de Motive con Matlab, que se conecta con RoboDK y el UR3e | 14/12/2023 |

El propósito de esta sesión es realizar una nueva captura de movimiento utilizando el sistema de cámaras, con el objetivo de retomar la tarea anterior.

Durante la experimentación, surgieron varios problemas con Motive, ya que estaba capturando marcadores que no existían. Después de adaptar el espacio de grabación para evitar reflejos y reiniciar el programa, se logró corregir el problema.

Finalmente, se determina que la manera más efectiva de asegurar que los movimientos registrados por las cámaras sean ejecutables por el UR3e es colocar los marcadores en el extremo del robot y grabar su propio movimiento, por lo que se llevó a cabo una nueva captura colocando los marcadores directamente en el robot, como se observa en la Ilustración 1 ,lo que permitió mover el robot y grabar el movimiento con el sistema de cámaras. Sin embargo, al intentar programarlo no se obtiene una versión funcional.

Una recamara con una cama y un escritorio

Descripción generada automáticamente con confianza baja

Ilustración . Captura de movimientos del robot del 14/12/2023

|  |  |
| --- | --- |
| Conexión de Motive con Matlab, que se conecta con RoboDK y el UR3e | 18/12/2023 |

El objetivo de esta sesión consiste en retomar la meta previa que aún no ha sido alcanzada: lograr la creación de un programa en RoboDK mediante un script de Matlab para que el robot imite los movimientos capturados.

Tras un análisis detallado de los datos en Motive, se identificó un problema relacionado con las distancias, ya que Motive utiliza milímetros mientras que RoboDK trabaja con metros. Se realizó una adaptación en el programa para asegurar la correcta transferencia de datos a RoboDK. No obstante, surgió un nuevo inconveniente, ya que se detectó un error que indicaba que los puntos eran inalcanzables. Para abordar este problema, se decidió realizar una nueva captura focalizándose únicamente en la posición usando un solo marcador en lugar de un sólido rígido.

|  |  |
| --- | --- |
| Conexión de Motive con Matlab, que se conecta con RoboDK y el UR3e y pruebas Machine Learning | 19/12/2023 |

Objetivos sesión:

-Capturar movimientos con el sistema de cámaras y crear un programa en RoboDK desde Matlab para que el robot imite los movimientos capturados y enviárselo al robot.

-Crear red neuronal que diferencie piezas cuadradas de rectangulares.

En esta sesión se retoma el objetivo aún no alcanzado de lograr la creación de un programa en RoboDK mediante un script de Matlab para que el robot imite los movimientos capturados. Además, se comienza con el objetivo de crear una red neuronal.

En cuanto a la parte de Matlab y RoboDK, tras diversas pruebas, no se consigue solucionar el problema de las posiciones inalcanzables.

Por otra parte, se prepara el lugar donde el robot identificará las piezas y se coloca la cámara webcam, de tal forma que vea nítidamente las piezas. Además, se toman varias imágenes y se logra crear un script funcional en Matlab de una red neuronal que diferencia piezas cuadradas y rectangulares, cuya arquitectura se define con el siguiente código:

layers = [

imageInputLayer([480 640 3])

convolution2dLayer(3, 2, 'Padding', 'same')

batchNormalizationLayer

reluLayer

maxPooling2dLayer(2, 'Stride', 2)

fullyConnectedLayer(2)

softmaxLayer

classificationLayer];

Finalmente, se prueba el script como se ve en la Ilustración 2.

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

Ilustración . Prueba de la red neuronal que diferencia entre cuadrado y rectángulo

|  |  |
| --- | --- |
| Conexión de Motive con Matlab, que se conecta con RoboDK y el UR3e y pruebas Machine Learning | 20/12/2023 |

Objetivos sesión:

-Capturar movimientos con el sistema de cámaras y crear un programa en RoboDK desde Matlab para que el robot imite los movimientos capturados y enviárselo al robot.

-Crear red neuronal que diferencie piezas rotas de piezas válidas.

El objetivo de esta sesión es retomar el objetivo de lograr la creación de un programa en RoboDK mediante un script de Matlab para que el robot imite los movimientos capturados. Además, de pasar a crear una red neuronal que diferencie piezas válidas de piezas rotas.

Tras realizar un análisis de los programas fallidos anteriores, se llega a la conclusión de que el error que experimenta RoboDK de posiciones inalcanzables, podría deberse a las orientaciones de Motive, por lo que se procede a realizar una nueva grabación del robot con los marcadores formando un sólido rígido, como se ve en la Ilustración 3.

Un escritorio con una computadora

Descripción generada automáticamente con confianza mediaImagen que contiene interior, tabla, cama, escritorio

Descripción generada automáticamente

Ilustración . Colocación de marcadores en el UR3e en la prueba del 20/12/2023

Al examinar el sólido rígido en el programa Motive, se observa que los ejes no coinciden con los del extremo de la herramienta en RoboDK. Para resolver esto, se propone ajustar las orientaciones en Motive aplicando un giro de 180 grados en Z en el script de Matlab para que se alineen con las de RoboDK, como se observa en el extracto de código inferior.

Texto

Descripción generada automáticamente

Ilustración . Fragmento de código de formación de los targets

Finalmente, con la implementación de las rotaciones en las orientaciones de Motive, se logra el objetivo de crear un programa en RoboDK desde Matlab, como se ve en la Ilustración 4. Este programa permite al robot replicar los movimientos capturados y se envía correctamente al robot. No obstante, la ejecución en el robot no puede realizarse directamente desde Matlab. En su lugar, es necesario acceder al programa creado en RoboDK y optar por la opción "Ejecutar en el robot".

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Ilustración . Programa de RoboDK generado desde Matlab el 20/12/2023

|  |  |
| --- | --- |
| Conexión de Motive con Matlab, que se conecta con RoboDK y el UR3e y pruebas ROS | 21/12/2023 |

Objetivos sesión:

-Capturar movimientos con el sistema de cámaras y crear un programa en RoboDK desde Matlab para que el robot imite los movimientos capturados y enviárselo al robot.

-Mapear una sala con el Turtlebot

El objetivo de esta sesión es retomar el objetivo de lograr la creación de un programa en RoboDK mediante un script de Matlab para que el robot imite los movimientos capturados, de manera que el programa se onvíe directamente desde Matlab. Además de pasar a mapear la sala con el Turtlebot4 para comenzar a trabajar con él.

Después de examinar la plataforma web, se experimentan varios comandos para ejecutar el programa en el robot mediante una orden desde Matlab. Desafortunadamente, ninguno de estos comandos resulta efectivo. Además, al revisar la documentación de RoboDK, se percibe que la API de Python ofrece una funcionalidad más completa en comparación con la de Matlab. Este inconveniente podría atribuirse a la posible falta de disponibilidad de dicha función para Matlab.

Por otro lado, se intenta trabajar con el Turtlebot4, pero este se queda sin batería, lo que impide realizar progresos significativos.

|  |  |
| --- | --- |
| Conexión de Motive con Matlab, que se conecta con RoboDK y el UR3e y pruebas ROS | 08/01/2024 |

Objetivos sesión:

-Capturar movimientos con el sistema de cámaras y crear un programa en RoboDK desde Matlab para que el robot imite los movimientos capturados y enviárselo al robot.

-Mapear una sala con el Turtlebot

El objetivo actual es retomar el proceso de mapeo de la sala utilizando el Turtlebot4. Además, se llevará a cabo una última prueba con RoboDK para ejecutar el programa en el robot desde Matlab.

Para alcanzar este propósito, se seguirá el tutorial "Generating a map" de los manuales de Turtlebot4, disponible en la siguiente dirección: https://turtlebot.github.io/turtlebot4-user-manual/tutorials/generate\_map.html. Siguiendo las instrucciones del tutorial, se inicia SLAM con el comando: ros2 launch turtlebot4\_navigation slam.launch.py, y posteriormente se visualiza el mapa en Rviz2 utilizando: ros2 launch turtlebot4\_viz view\_robot.launch.py como se ve en la Ilustración 6.

Una captura de pantalla de una computadora

Descripción generada automáticamente con confianza media

Ilustración . Visualización inicial de Rviz2 al comenzar el mapeo

Sin embargo, tras mapear un rato, el robot se queda sin batería, por lo que se deja cargando y se pasa a la parte de RoboDK.

En cuanto a RoboDK, se consigue en un portátil ejecutar el programa en el robot desde Matlab, utilizando un script que requiere tener previamente el programa creado en RoboDK, y luego se completa desde Matlab. No obstante, al intentar ejecutar el mismo script en el ordenador del laboratorio, se encuentra que no funciona. Se ha intentado resolver este problema actualizando Matlab y la API de RoboDK, pero lamentablemente, no se ha logrado hacer que funcione en dicho entorno.

|  |  |
| --- | --- |
| Mapeado y navegación con Turtlebot4 | 09/01/2024 |

Objetivos sesión:

-Mapear una sala con el Turtlebot

-Enviar a un punto del mapa el Turtlebot

Los objetivos de la sesión se centran en la implementación de dos tareas fundamentales utilizando el Turtlebot4. En primer lugar, se continua con realizar un exhaustivo mapeo de una sala utilizando las capacidades de este robot. En segundo lugar, se pretende desarrollar la habilidad de navegación autónoma del Turtlebot, permitiéndole desplazarse de manera automática y eficiente hacia un punto específico dentro del mapa previamente creado.

Por una parte, se realiza un mapeado con los pasos seguidos el día anterior del tutorial "Generating a map" obteniendo el mapa de la zona de trabajo en Rviz2 que se ve en la Ilustración 7.

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

Ilustración . Visualización final del mapa en Rviz2

A continuación, se crea un directorio para guardar el mapa en él y se guarda con el siguiente comando: ros2 launch turtlebot3\_navigation2 navigation2.launch.py use\_sim\_time:=true map:=maps/mapa\_robolab.yaml como se ve en la Ilustración 8.

Texto

Descripción generada automáticamente

Ilustración . Comando para guardar el mapa

Con el mapa creado y guardado, se pasa al objetivo de navegación, para lo cual se sigue el tutorial “Navigation” , disponible en la siguiente dirección: <https://turtlebot.github.io/turtlebot4-user-manual/tutorials/navigation.html>

Siguiendo el tutorial, se abre una terminal y se ejecuta el comando de inicio de la localización: ros2 launch turtlebot4\_navigation localization.launch.py map:=mapa\_robolab.yaml. A continuación, se lanza nav2 en otra terminal: ros2 launch turtlebot4\_navigation nav2.launch.py. Finalmente, para interactuar con la navegación se lanza Rviz2 con el siguiente comando: ros2 launch turtlebot4\_viz view\_robot.launch.py. Dentro del mapa de Rviz2, se usa la herramienta 2D Pose Estimate para indicar la pose inicial del robot, con lo que se genera el mapa de coste, como se ve en la Ilustración 9.

Interfaz de usuario gráfica, Mapa

Descripción generada automáticamente

Ilustración . Generación del mapa de coste

Luego de la generación exitosa del mapa de coste, se lleva a cabo una prueba inicial de navegación utilizando la herramienta Nav2 Goal, obteniendo resultados satisfactorios.

En una fase posterior, se inicia una investigación con el propósito de entender cómo enviar el objetivo a través de un topic en lugar de utilizar la herramienta Nav2 Goal. En una fase posterior, se inicia una investigación con el propósito de entender cómo enviar el objetivo a través de un topic en lugar de utilizar la herramienta Nav2 Goal. Este enfoque busca explorar opciones alternativas para la definición de objetivos de navegación, de modo que se pueda integrar en el programa de Matlab. Finalmente, se descubre que esta integración es posible mediante el uso del siguiente comando:

ros2 topic pub /goal\_pose geometry\_msgs/msg/PoseStamped "header:

stamp:

sec: 0

nanosec: 0

frame\_id: 'map'

pose:

position:

x: 0.106

y: -2.52

z: 0.0

orientation:

x: 0.0

y: 0.0

z: 0.0

w: 1.0"

Este comando permite la publicación de información de posición y orientación del objetivo a través del topic /goal\_pose, facilitando así la integración deseada en el programa desarrollado en Matlab.

|  |  |
| --- | --- |
| Configuración para el movimiento de recogida de piezas | 10/01/2024 |

Objetivos sesión:

-Configurar puntos objetivo en la navegación del Turtlebot

-Configurar entorno para la recogida de piezas

El primero de los objetivos de la sesión se centra en determinar en el espacio de trabajo los puntos entre los que se desplazará el Turtlebot. Rn segundo lugar, se trabajará en la configuración del entorno del UR3e y el Turtlebot, garantizando un ambiente óptimo para llevar a cabo la recogida de piezas de manera eficaz.

En una primera fase, se identifican los tres puntos a los cuales se dirigirá el Turtlebot: uno designado para la recolección de piezas, otro al cual se trasladará cuando las piezas estén en perfecto estado, y un tercero al que se dirigirá en caso de piezas defectuosas. Estos puntos se destacan en el suelo mediante cinta y se activan los comandos de navegación pertinentes, previamente abordados en la sesión anterior. La Ilustración 12 ofrece una representación visual del Turtlebot posicionado en el punto destinado a la recogida de piezas.

Imagen que contiene tabla, horno

Descripción generada automáticamente

Ilustración . Turtlebot sobre el punto de recogida de piezas

En una segunda fase, se procede a reconfigurar la disposición de las cámaras, realizando ajustes estratégicos para optimizar la captación de imágenes. Simultáneamente, se prepara meticulosamente la zona de trabajo destinada a la recogida de piezas por el robot UR3e. Este proceso implica la organización precisa de los elementos en el entorno, garantizando un espacio eficiente y accesible para el desempeño de la tarea.

Posteriormente, se lleva a cabo una fase de prueba con el objetivo de verificar y afinar el movimiento del brazo robótico UR3e al pasar la pieza al Turtlebot, para ello se añade una pequeña caja a modo de pinza, como se ve en la Ilustración 13. Esta etapa es crucial para asegurar una transferencia fluida y precisa de la pieza entre los dos robots, optimizando así la eficacia del sistema en su conjunto.

Imagen que contiene interior, tabla, lavabo, espejo

Descripción generada automáticamente

Ilustración . Pruebas de trayectorias para pasar la pieza al Turtlebot

Finalmente, se realiza una nueva captura utilizando el sistema OptiTrack, orientada específicamente al seguimiento y registro del movimiento durante la fase de recogida de piezas, como se ve en la Ilustración 14.

Una sala de estar

Descripción generada automáticamente con confianza media

Ilustración . UR3e con los marcadores colocados realizando el movimiento de pasar la pieza

|  |  |
| --- | --- |
| Configuración de la red para integrar el UR3e y el Turtlebot | 11/01/2024 |

Objetivos sesión:

- Conectar Turtlebot y UR3e a la red robolab\_5g

En esta jornada, el enfoque se centra en la configuración de la red robolab5g para permitir la integración y comunicación efectiva entre el Turtlebot y el UR3e. Es esencial lograr que ambos robots estén conectados a la misma red, facilitando así su control desde un único programa en el ordenador central. En la fase inicial, se realiza la conexión física de cada robot a la red robolab\_5g, asegurando una comunicación estable y confiable. Se llevan a cabo ajustes en la configuración de red de cada dispositivo para garantizar una integración sin problemas.

Durante este día se investigan las conexiones de la red hasta que se descubre cómo cambiar el cableado del router al que se conecta UR3e para conectarlo a la red robolab\_5g. Además, se realizan las configuraciones y pruebas necesarias para verificar la comunicación con el Turtlebot y el UR3e, asegurándose de que ambos puedan intercambiar información conectados a la red robolab-5g de manera eficiente.

Tras realizar la configuración de red, se ha intentado ajustar la última trayectoria grabada con Motive para RoboDK ya que, al ejecutar el programa en el robot real, se observa que durante la trayectoria de empuje de la pieza, el robot Turtlebot se aproxima demasiado a la mesa del UR3e, generando problemas de colisión con ella al colocársele una pinza o en este caso la caja, que simula una pinza.

Durante el resto de la sesión, se ha intentado ajustar la posición del eje B0 con el objetivo de mejorar la representación de los puntos objetivo. Sin embargo, es un ajuste complicado y no se ha logrado una trayectoria adecuada, por lo que se plantea volver a grabar el movimiento en la próxima sesión para lograr un movimiento suave y óptimo.

Al final de esta sesión, se ha logrado una configuración de red robusta que permite la colaboración del Turtlebot y el UR3e. Además, se ha planteado el objetivo de la próxima sesión de grabar una trayectoria nueva que resuelte eficiente.

|  |  |
| --- | --- |
| Optimización del movimiento de recogida de piezas | 16/01/2024 |

Objetivos sesión:

-Ajustar la trayectoria del UR3e

En esta sesión, el objetivo central fue perfeccionar la trayectoria del UR3e durante la recogida de piezas para evitar colisiones con la mesa.

Durante la sesión, se llevó a cabo una nueva grabación del movimiento utilizando el sistema de cámaras OptiTrack, como se ve en la Ilustración 15, teniendo en cuenta los problemas previos de colisión. Para lograr una trayectoria más eficiente, se redujo la velocidad del UR3e durante el movimiento y se ajustaron las posiciones de los marcadores para asegurar su captura constante por parte de las cámaras. Este enfoque resultó en una grabación exitosa de una trayectoria más eficiente y segura para el movimiento de recogida de piezas.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

Ilustración . Grabación con OptiTrack para optimizar la trayectoria

Una vez grabada la nueva trayectoria, se procedió a la creación del programa en RoboDK mediante Matlab. Durante este proceso, se observó que el sólido rígido tenía una orientación diferente a otras grabaciones anteriores. Se realizaron los ajustes necesarios, girando la orientación hasta alcanzar la deseada.

Al concluir la sesión, se logró obtener una nueva trayectoria de movimiento optimizada que no presenta colisiones con la mesa. Como siguiente objetivo, se propone en la siguiente sesión identificar los puntos más significativos de la trayectoria para elaborar un programa aún más óptimo.

|  |  |
| --- | --- |
| Optimización del movimiento de recogida de piezas | 17/01/2024 |

Objetivos sesión:

-Ajustar los puntos de la trayectoria para RoboDK

En esta sesión continuativa, el objetivo se centró en extraer los dos puntos más significativos del inicio y fin de la recogida de piezas para elaborar un programa optimizado en RoboDK. Tras la grabación de la trayectoria en la sesión anterior, se realizó un análisis detallado para identificar los puntos críticos que definen el inicio y el final del movimiento.

Utilizando esta información, se procedió a la programación en RoboDK mediante Matlab, empleando los datos capturados por el sistema OptiTrack de la nueva trayectoria que aseguran un movimiento fluido y preciso.

Finalmente, se logró desarrollar un programa en RoboDK que sigue la trayectoria previamente entrenada con OptiTrack, incorporando los puntos significativos extraídos. Este programa optimizado garantiza una ejecución eficiente y segura del movimiento de recogida de piezas, marcando un paso adelante en la integración exitosa entre la grabación de movimientos y la programación del robot.

|  |  |
| --- | --- |
| Reparación del Turtlebot | 18/01/2024 |

Objetivos sesión:

- Diagnosticar y solucionar problemas en el Turtlebot

En esta jornada, se aborda la necesidad de diagnosticar y solucionar problemas específicos que han surgido en el Turtlebot.

La sesión comienza con un análisis detallado de los síntomas y comportamientos anómalos del Turtlebot. Tras consultar al profesor, se llega a la conclusión de que hay que flashear la Raspberry.

Después de completar el flasheo, se procede a reconfigurar el Turtlebot. Tras realizar los ajustes pertinentes con ayuda del profesor se consigue recuperar la funcionalidad del Turtlebot.

Una vez finalizado el proceso de flasheo y configuración, se realiza una serie de pruebas exhaustivas para verificar la funcionalidad restaurada del Turtlebot.

Al concluir esta sesión, se ha restablecido con éxito el Turtlebot a plena funcionalidad.

|  |  |
| --- | --- |
| Integración de todos los módulos en un solo programa | 22/01/2024 |

Objetivos sesión:

- Integrar todas las tecnologías en un solo programa

En esta sesión, el objetivo primordial fue la consolidación de todas las tecnologías y módulos en un único programa integral. Durante el transcurso de la sesión, se llevó a cabo la planificación de la secuencia que el programa debía seguir, y se desarrolló un nuevo script de Matlab que fusiona los módulos de movimiento del UR3e con RoboDK, la ejecución de la red neuronal preentrenada, y la navegación del Turtlebot utilizando ROS2.

Una vez completada la programación, se realizaron pruebas por secciones para evaluar su funcionamiento. Se observó que era necesario ajustar la navegación del Turtlebot para que coincidiera con la pose objetivo de recoger piezas. En un principio, se consideró la posibilidad de que el problema estuviera relacionado con el bumper, el cual no se cargaba correctamente en la simulación y se posicionaba frente al robot, como se ve en la Ilustración 16.

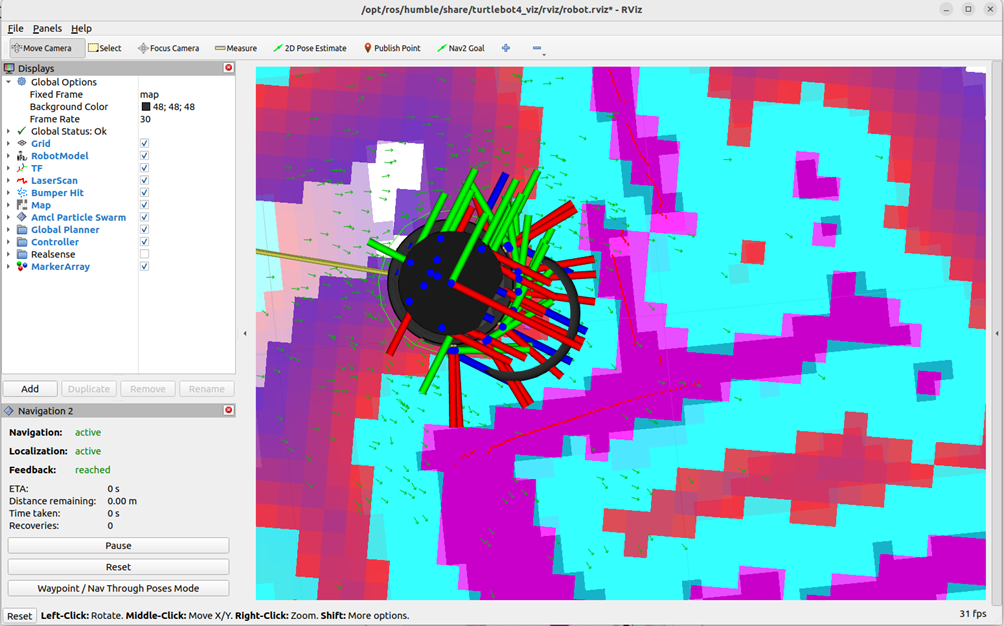


Ilustración . Visualización de la ubicación del bumper

Sin embargo, tras un análisis más detenido, se llegó a la conclusión de que el inconveniente radicaba en el mapa de costes generado. No era posible acercar más el Turtlebot mediante instrucciones de navegación, ya que las poses alrededor del obstáculo eran percibidas como inaccesibles en el mapa de costes. Como solución, se determinó que lo más eficaz sería, una vez alcanzada la posición más cercana a la deseada, publicar en el topic "/cmd\_vel" para dirigir el movimiento del robot hacia la posición de recogida. Para lograr esto de manera precisa, se requiere realizar ajustes tanto en la aproximación como en la orientación del Turtlebot.

Al final de la sesión, se logró la consolidación exitosa de todos los módulos en un solo programa y se propone como objetivo para la siguiente sesión ajustar el acercamiento del Turtlebot a el punto de recogida de pieza.

|  |  |
| --- | --- |
| Integración de todos los módulos en un solo programa | 23/01/2024 |

Objetivos sesión:

- Integrar todas las tecnologías en un solo programa

En esta sesión, el enfoque principal se centró en perfeccionar la aproximación del Turtlebot al punto de recogida de las piezas para optimizar el programa principal.

Durante el transcurso de la sesión, se llevó a cabo una cuidadosa optimización de las instrucciones de navegación proporcionadas al Turtlebot. Durante este proceso, se tuvo en cuenta la necesidad de garantizar que el Turtlebot se acercara de manera óptima al punto de recogida de piezas, evitando posibles colisiones y optimizando la eficiencia del movimiento, para ello se realizaron mejoras en la publicación de comandos de velocidad ("/cmd\_vel") para dirigir con mayor precisión el movimiento del robot hacia la posición deseada.

Al finalizar la sesión, se logró una mejora significativa en el acercamiento del Turtlebot al punto de recogida de piezas.

|  |  |
| --- | --- |
| Integración de todos los módulos en un solo programa | 24/01/2024 |

Objetivos sesión:

- Probar el movimiento programado del Turtlebot

En esta sesión, el objetivo principal fue evaluar el movimiento programado del Turtlebot, específicamente el avance y retroceso, con el propósito de llegar a la posición exacta deseada.

Durante las pruebas se identificó un desafío significativo: las mediciones del LIDAR presentaban variaciones debido al ruido, afectando la consistencia y precisión del movimiento. A pesar de ajustes minuciosos, el Turtlebot no alcanzaba siempre el mismo punto debido a estas fluctuaciones. Para abordar este problema, se decidió reducir la distancia de avance, giro y retroceso, y se propuso la opción de emplear una caja más grande para brindar mayor flexibilidad.

Al finalizar la sesión, se concluye que, aunque se lograron avances, es necesario seguir ajustando el movimiento para optimizar completamente el rendimiento del Turtlebot.

|  |  |
| --- | --- |
| Optimización de la red neuronal e integración en el programa principal | 25/01/2024 |

Objetivos sesión:

-Optimizar la red neuronal e integrarla en el programa principal

Esta sesión, se enfocó a la mejora de la parte del programa de la red neuronal diseñada para diferenciar piezas rotas de piezas válidas y piezas cuadradas de rectangulares.

Inicialmente, se llevó a cabo una prueba utilizando el conjunto de datos utilizado en otras sesiones y se llegó a la conclusión de que podía mejorarse la precisión. Para ello, se tomó la decisión de romper más piezas y capturar nuevas imágenes para que la red tuviese más datos.

Tras tomar nuevas imágenes y ajustar los parámetros de la red se logra una precisión del 94.59% como se ve en la Ilustración 16, lo cual se considera óptimo, con lo que se guarda la red ya entrenada para cargarla en el programa principal y emplearla.

Texto

Descripción generada automáticamente

Ilustración . Cálculo de la precisión de la red optimizada

Al finalizar la sesión, se carga la red en el programa principal y se prueba, dando lugar a unos resultados excelentes, que reflejan el impacto positivo de la ampliación del conjunto de datos y el reentrenamiento. Sin embargo, se observa que, a pesar del ajuste realizado para el movimiento de recogida del Turtlebot, se debe cambiar la caja empleada por el Turtlebot para la recogida de piezas por una más grande para evitar colisiones con las mesas en la zona de recogida y hacer el movimiento más seguro.

|  |  |
| --- | --- |
| Documentación del proyecto | 29/01/2024 |

Objetivos sesión:

- Documentar el proyecto

El objetivo de esta sesión es elaborar un vídeo que incorpore vídeos de las etapas previas del proyecto y de la parte en tiempo real.

Durante esta sesión, se documenta todo el trabajo realizado en el proyecto en un vídeo. Sin embargo, al intentar realizar el vídeo del funcionamiento completo del sistema, se observa que, el Turtlebot no funciona, por lo que hay que explorar posibles soluciones. Finalmente, se vuelve a flashear la Raspberry.

Al finalizar la sesión, se logra que todo funcione con normalidad.