

Universidad Internacional de La Rioja (UNIR)

ESIT

Máster Universitario en Inteligencia Artificial

Segmentación semántica RGB-D para navegación en interiores

Trabajo Fin de Máster

Presentado por: Mikel Aldalur Corta

Director/a: Dr. Salvador Cobos Guzman

Ciudad: Bilbao
Fecha: 15/09/2021

Resumen

Los robots y coches autónomos son cada vez más populares en nuestro entorno y la seguridad en cuanto a la navegación autónoma tiene que ser eficiente y efectiva. Este trabajo presenta una herramienta de segmentación semántica que se puede utilizar en cualquier tipo de robot para predecir estados o escenarios futuros cuando un robot está navegando por entornos de interior. Para la predicción, se utilizan las redes convolucionales LSTM que predicen cual es el posible escenario futuro. Para que el sistema sea más eficiente que utilizando solo una cámara de color, se utilizan también las profundidades obtenidas por las cámaras infrarrojas. Para una posterior planificación, se introduce el modelo en ROS para su utilización.

Palabras Clave: Aprendizaje profundo, segmentación semántica, visión RGBD, robots autónomos, ROS.

Abstract

Autonomous robots and cars are becoming more and more popular in our environment and the safety of autonomous navigation must be efficient and effective. This paper presents a semantic segmentation tool that can be used on any type of robot to predict future states or scenarios when a robot is navigating in indoor environments. For prediction, LSTM convolutional networks are used to predict the possible scenario. To make the system more efficient than using only a colour camera, the depths obtained by infrared cameras are also used. For further planning, the model is introduced so that it can be used in ROS.

Keywords: deep learning, semantic segmentation, RGBD vision, autonomous robots, ROS.

Índice de contenidos

1. Introducción	1
1.1 Motivación	1
1.2 Planteamiento del trabajo.....	2
1.3 Estructura de la memoria	2
2. Contexto y estado del arte	4
2.1. SLAM mediante visión por computador.....	4
2.1.1. Cámara monocular	4
2.1.2. Cámara estéreo doble	6
2.1.3. Cámaras estereoscópicas	8
2.1.4. LiDAR	9
2.1.4. IMU	10
2.3. Detección y segmentación	11
2.3.1. Detección de objetos	11
2.3.2. Segmentación	13
2.3.2. Posición y trayectoria del ser humano	15
2.4 Resumen	17
3. Objetivos y metodología de trabajo.....	18
3.1. Objetivo general	18
3.2. Objetivos específicos.....	18
3.3. Metodología del trabajo.....	19
4. Identificación de requisitos.....	22
5. Descripción de la herramienta software desarrollada	25
5.1. Arquitectura del hardware	26
5.2. Utilización de la cámara Intel-RealSense.....	26
5.3. Elección del algoritmo de segmentación semántica RGB-D	29
5.4 Entrenamiento del modelo de segmentación semántica.....	33

5.4.1 Datasets.....	33
5.4.2 Modelo 1	35
5.4.3 Modelo 2	37
5.5 Adaptación del modelo de segmentación para predecir escenarios futuros	39
5.6 Entrenamiento de la red neuronal predictor de trayectoria	39
5.6.1 Datasets.....	39
5.4.2 Modelo LSTM 1	42
5.4.2 Modelo LSTM 2	42
5.7 Preparando el modelo para Jetson Nano.....	44
6. Evaluación	45
6.1 Métricas obtenidas de los modelos	45
6.2 Evaluación de diferentes escenarios.....	47
7. Conclusiones y trabajo futuro.....	51
7.1. Conclusiones	51
7.2. Líneas de trabajo futuro.....	52
8. Bibliografía	53
Anexos	59
Anexo. Artículo de investigación	59

Índice de tablas

Tabla 1. Medidas de acelerómetro.....	10
Tabla 2. Medidas del giroscopio	10
Tabla 3: Planificación de esprints para el proyecto.....	21
Tabla 4. Tabla de requisitos.....	22
Tabla 5. Técnicas de la herramienta.....	23
Tabla 6. Especificaciones de la tarjeta Jetson Nano	25
Tabla 7. Especificaciones de la cámara Intel-RealSense D435i.....	25
Tabla 8: Métricas del modelo de segmentación semántica sin predicción.	45
Tabla 9: Métricas del modelo 1 de segmentación semántica con predicción.	46
Tabla 10: Métricas del modelo 3 de segmentación semántica con predicción.	46

Índice de figuras

Figura 1: Imagen utilizando el algoritmo ORB para detectar objetos mediante características.	5
Figura 2: Simulación ORB SLAM de desde diferentes perspectivas (<i>ORB SLAM Proposal for NTU GPU Programming Course 2016</i> , n.d.)	5
Figura 3: Sistema 3D creado por ORB-SLAM (<i>ORB SLAM Proposal for NTU GPU Programming Course 2016</i> , n.d.)	6
Figura 4: Principios de ajuste para cámaras estéreo (<i>Principle Drawing of a Stereo Camera Setup. Objects (1,2) in Various... Download Scientific Diagram</i> , n.d.)	6
Figura 5: Generando un mapa de cámaras estereos (<i>Stereo Visual SLAM System Overview: First, We Undistort and Rectify the... Download Scientific Diagram</i> , n.d.)	7
Figura 6: Tipo cámara estéreo ZED (<i>ZED 2 - AI Stereo Camera Stereolabs</i> , n.d.)	7
Figura 7: cámara Intel-RealSense-D435i (Intel, 2019)	8
Figura 8: Imagen de profundidad de la cámara estereoscópica	8
Figura 9: Creación de SLAM mediante cámaras estereoscópicas (Zhang et al., 2021)	9
Figura 10. Mapa creado en <i>RViz</i> por un sensor <i>LiDAR</i> (<i>Incorrect Visualization of LiDAR · Issue #32 · Carla-Simulator/Ros-Bridge</i> , n.d.)	9
Figura 11: Ejemplo de detector de cara con <i>Haar Cascade</i> (Jiwon, 2019)	12
Figura 12: Comparando los algoritmos de detección de objetos en base al tiempo de respuesta(Bochkovskiy et al., 2020)	12
Figura 13: Evolución de Imagenet (<i>Review: SENet — Squeeze-and-Excitation Network, Winner of ILSVRC 2017 (Image Classification) by Sik-Ho Tsang Towards Data Science</i> , n.d.)	13
Figura 14: Imagen exterior segmentada (<i>Illustration of Challenges in Semantic Segmentation. (a), Input Image.... Download Scientific Diagram</i> , n.d.)	13
Figura 15: Mapa semántico para navegación (Deng et al., 2020)	14
Figura 16: Detección de objetos y segmentación de una imagen (<i>Supercharge Your Computer Vision Models with Synthetic Datasets Built by Unity Unity Blog</i> , n.d.)	14
Figura 17: Estimación de posicionamiento de humanos en entorno segmentado (Seichter et al., 2020)	15

Figura 18: Segmentación de trayectorias después de procesado (Tamaki et al., 2019).....	15
Figura 19. Detección de trayectoria para cálculo de navegación (Bruckschen et al., 2020).	16
Figura 20. Se muestran las diferentes trayectorias de las personas.	17
Figura 21: Ejemplo de la metodología Scrum (<i>SCRUM - ECLEE European Center for Leadership and Entrepreneurship Education</i> , n.d.).....	19
Figura 22. Imagen de Jetbot con la cámara integrada.....	26
Figura 23. Imagen RGB de resolución 1920x1080	27
Figura 24. Imagen RGB de resolución 1280x960	28
Figura 25. Imagen de profundidad de resolución 1280x960	28
Figura 26. Imagen RGB de resolución 640x480	28
Figura 27. Imagen de profundidad de resolución 640x480.....	29
Figura 28: Se muestra cómo se realiza el bloque residual del aprendizaje (<i>What Is Resnet or Residual Network How Resnet Helps?</i> , n.d.).	30
Figura 29:Se muestra cómo se realiza el bloque Non-Bottleneck-1D (He et al., 2016).....	31
Figura 30: Se muestra el modelo U-Net (Weng & Zhu, 2021).	31
Figura 31: Se muestra la fusión del modelo ESANet (Seichter et al., 2020).	32
Figura 32: Ejemplo de la imagen de NYUv2.....	34
Figura 33: Ejemplo de la imagen de SUNRGBD (531x681).	34
Figura 34: Ejemplo de la imagen de SUNRGBD (96x128).	35
Figura 35: Ejemplo de la imagen de SUNRGBD con 4 categorías (96x128).....	35
Figura 36: exactitud y pérdida durante el entrenamiento del modelo 1 con SGD.....	36
Figura 37: Exactitud y pérdida durante el entrenamiento del modelo 1 con <i>transfer learning</i>	37
Figura 38: Exactitud y pérdida durante el entrenamiento del modelo 2 con Adam.....	38
Figura 39: Exactitud y pérdida durante el entrenamiento del modelo 2 con <i>transfer learning</i>	38
Figura 40: Imágenes de entrada para entrenamiento de datos para predicción 1	40
Figura 41: Imágenes de entrada para entrenamiento de datos para predicción 2.	41
Figura 42: Imágenes de entrada para entrenamiento de datos para predicción 3.	41

Figura 43: Exactitud y pérdida durante el entrenamiento del modelo de predicción 1.	42
Figura 44: Imagen del Modelo simplificado para entender el funcionamiento.	43
Figura 45: Exactitud y pérdida durante el entrenamiento del modelo de predicción 3.	43
Figura 46: Resultado del modelo 1 con LSTM.....	47
Figura 47: Imágenes y resultado de la escena A.....	48
Figura 48: Imágenes y resultado de la escena B.....	48
Figura 49: Imágenes y resultado de la escena C.....	49
Figura 50: Imágenes y resultado de la escena D.....	49
Figura 51: Imágenes de la escena E.	50
Figura 52: Resultado de la escena E.....	¡Error! Marcador no definido.

1. Introducción

La navegación autónoma es cada vez más popular entre los robots y los vehículos móviles. De esta manera, estos robots o vehículos pueden moverse en función de los sensores de captación que integran, siendo necesario la eficacia y la correcta limitación de navegación. Las disciplinas de visión por computador y aprendizaje automático son las que realizan la investigación en el área de navegación autónoma y cada vez realizan avances más elaborados y profundos.

Estos avances han ayudado a que los robots sean cada vez más fiables en cuanto a evitar la colisión con los obstáculos del entorno, crear mapa de navegación, percepción de personas o la segmentación. Los procesos de analizar el entorno han ido cambiando y la seguridad de estos sistemas es esencial en los entornos donde se encuentran personas, animales, plantas o cualquier tipo de obstáculo. La navegación puede ser dentro de un entorno aprendido o un entorno nuevo según las necesidades, pero en todos los casos es necesario identificar o captar bien el entorno para luego poder elegir el camino adecuado. En este caso se realizará una segmentación del entorno captado para que la navegación sea más fácil y fiable. También se incorporará la predicción de movimientos para que los movimientos del robot no molesten y sean más precisos, es decir, se ajusten a cada entorno y predigan los movimientos según lo aprendido. Como ejemplos podemos tener diferentes tipos de entornos como oficinas, escuelas, hospitalares o cualquier tipo de edificio donde se quiera introducir un robot.

1.1 Motivación

Los robots autónomos son de una gran ayuda para realizar cualquier tarea y la navegación y la percepción son unos de los puntos en los que se están realizando las mejoras. Hoy en día, podemos ver cómo los robots limpian la casa, acompañan a personas, realizan búsquedas de elementos en diferentes entornos o realizan las tareas de inventario de almacenes. Estos robots tienen que ser seguros y evitar todo tipo de obstáculos. Estos obstáculos pueden ser seres vivos u otros elementos sensibles y pueden estar en peligro si el robot no realiza bien su trabajo.

La segmentación semántica es una de las áreas en progreso para este tipo de problemas y los últimos avances de la navegación autónoma se centran en la visión por computador para realizarla. La segmentación mediante la captación de imágenes del entorno es esencial para que un robot navegue autónomamente con la suficiente fiabilidad y seguridad para que no ocurra ningún imprevisto.

Es necesario entrenar a los modelos de visión por computador mediante aprendizaje automático para que realicen una segmentación correcta de los elementos que se encuentran en el entorno y los clasifique para que el robot tenga claro cuál es el sitio por donde puede circular. En este caso en concreto el robot tiene que navegar en el interior y tiene que ser capaz de tomar decisiones correctas a la hora de realizar la navegación, aunque el entorno pueda variar.

1.2 Planteamiento del trabajo

En este trabajo se intenta abordar el problema en los entornos de interior donde el robot tiene que elegir el camino por donde circular. Para ello, en vez de utilizar solo un sensor RGB para la segmentación, se utilizará un sensor RGB-D para tener una mejor segmentación donde se le añade la profundidad del entorno. A la cámara Intel RealSense D435i se le añadirá un sistema de aprendizaje profundo para detectar las trayectorias de los humanos para que realice una correcta segmentación.

Para la movilidad se necesita un robot que sea capaz de realizar movimientos en el entorno. Este robot móvil será el que se llama Jetbot, el robot móvil de código libre que utiliza como módulo de control el equipo de desarrollo Jetson Nano.

Todo el programa irá dentro del *framework* ROS “Robot Operating System”, el código libre que facilita el desarrollo de los proyectos robóticos.

1.3 Estructura de la memoria

En el capítulo 2 se realiza el análisis y estado del arte de todo lo referente a los robots móviles y la percepción por computador. Se analizan diferentes publicaciones como sistemas de percepción de entorno para la localización y mapeo simultáneo (SLAM), Detección de objetos o segmentación semántica. Se parte de conceptos básicos para finalmente centrarse en los conceptos que se abordaran en el presente trabajo.

En el capítulo 3 se detallan los objetivos y la metodología a utilizar, mientras que en el capítulo 4 se identifican los requisitos para la elaboración del presente trabajo de la herramienta software.

Todo el desarrollo a detalle se lleva a cabo en el capítulo 5 y seguidamente se muestran los resultados obtenidos en el capítulo 6, las pruebas realizadas y sus valoraciones para ser más exactos. Finalmente, en el capítulo 7 se muestran las conclusiones obtenidas.

2. Contexto y estado del arte

La navegación autónoma está avanzando a pasos agigantados, para ello son evidentes los coches autónomos, robots humanoides o cuadrúpedos que ayudan en muchas de las tareas que tienen que realizar los seres humanos. La interacción de los robots con la humanidad también está creciendo ya que cada vez son más seguros y fiables por el avance de la tecnología y la investigación en este ámbito. La utilización de los diferentes sistemas, con extensas características, es indispensable para establecer la seguridad de los humanos y del entorno. En este apartado se analizarán estos elementos y algoritmos de procesamiento de esta información.

2.1. SLAM mediante visión por computador

La visión por ordenador se encarga de capturar y procesar las imágenes. Existen diferentes algoritmos para procesar las imágenes y obtener una perspectiva de tres dimensiones. En este apartado se analizan diferentes formas de crear un mapa utilizando diferentes configuraciones de cámaras para que los robots autónomos tengan conocimiento previo del entorno.

2.1.1. Cámara monocular

Las cámaras monoculars, las que se han utilizado toda la vida para sacar las fotos, son las más utilizadas en todos los sistemas por la simplicidad y el pequeño coste que suponen. Con estas cámaras y los algoritmos desarrollados hasta ahora, se pueden realizar muchas operaciones.

Para obtener la posición de un elemento o comparar los elementos de las imágenes se utilizan los algoritmos de extracción de características. Con la extracción de las características se sacan los descriptores que se comparan con la siguiente imagen. Uno de los algoritmos utilizados es *rotated brief* (ORB) (R. Wang et al., 2019), que viene después de varios algoritmos como SIFT o SURF para mejorar la eficiencia y el coste computacional. Estos dos algoritmos se utilizan para la extracción de las características de imágenes. En concreto, SIFT se utiliza para detectar y describir las características locales y pasárselas a vectores. Las transformaciones son invariantes a otras transformaciones y guarda la información de la orientación. Con esto se guarda la descripción de un trozo de imagen en vectores para

después realizar la comparación en otras imágenes como se puede ver en la Figura 1. El algoritmo SURF es un algoritmo similar con un procesado más rápido.



Figura 1: Imagen utilizando el algoritmo ORB para detectar objetos mediante características.

Este mismo algoritmo que se utiliza para la detección de objetos, realizar las imágenes panorámicas o para la mejora de imágenes en movimiento, también se ha utilizado en conjunto con una unidad de mediciones de inercia (IMU) para crear mapas de tres dimensiones (Mur-Artal et al., 2015). Con el sensor de inercias para detección de velocidad, orientación y aceleraciones se capta el movimiento realizado entre cada fotograma y realiza la estimación del entorno en 3D.

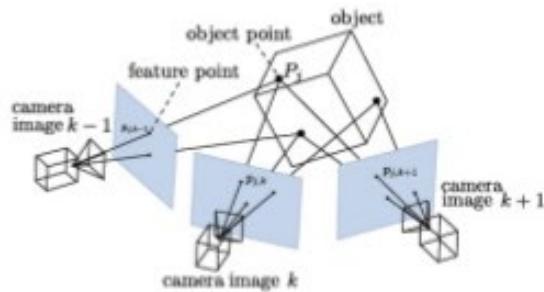


Figura 2: Simulación ORB SLAM de desde diferentes perspectivas (*ORB SLAM Proposal for NTU GPU Programming Course 2016*, n.d.).

La diferencia de movimiento calculado por el seguimiento de posición realizado por el sensor IMU y los puntos de la imagen generados por la cámara en cada fotograma por el algoritmo elegido, que puede ser ORB, se convierten en un sistema de varias fotos. Estas fotos van a

tener puntos comunes sacados por el algoritmo, como en la Figura 2, y se pueden juntar para crear una nueva perspectiva sabiendo la posición de captura, realizando los cálculos con el sensor IMU y de esta manera crear un sistema de tres dimensiones.

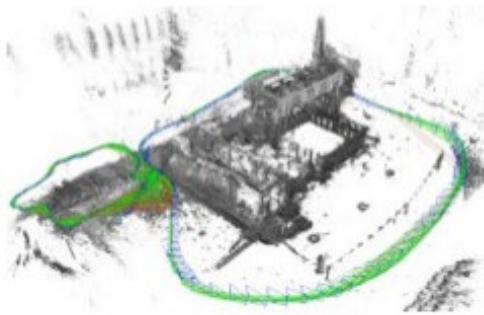


Figura 3: Sistema 3D creado por ORB-SLAM (*ORB SLAM Proposal for NTU GPU Programming Course 2016*, n.d.).

2.1.2. Cámara estéreo doble

El sistema monocular puede funcionar bien para ciertas funcionalidades, pero para asegurarse de que las mediciones se hacen de una manera más exacta, siempre es mejor contar con dos cámaras.

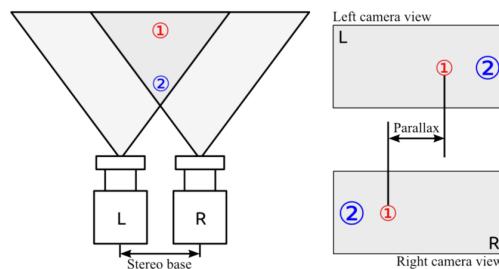


Figura 4: Principios de ajuste para cámaras estéreo (*Principle Drawing of a Stereo Camera Setup. Objects (1,2) in Various... | Download Scientific Diagram*, n.d.).

El sistema de funcionalidad es similar a la de cámara monocular, la diferencia es que estas cámaras, cada una de ellas, tienen diferentes ángulos de visión tal y como se puede ver en la Figura 4. Se ve claramente que la perspectiva de la profundidad plasmada en las dos imágenes que ayudan a realizar las medidas necesarias para obtener la visión estéreo

(Zaarane et al., 2020). En este caso se puede ver que con las cámaras estéreos es posible medir las distancias para detectar obstáculos.

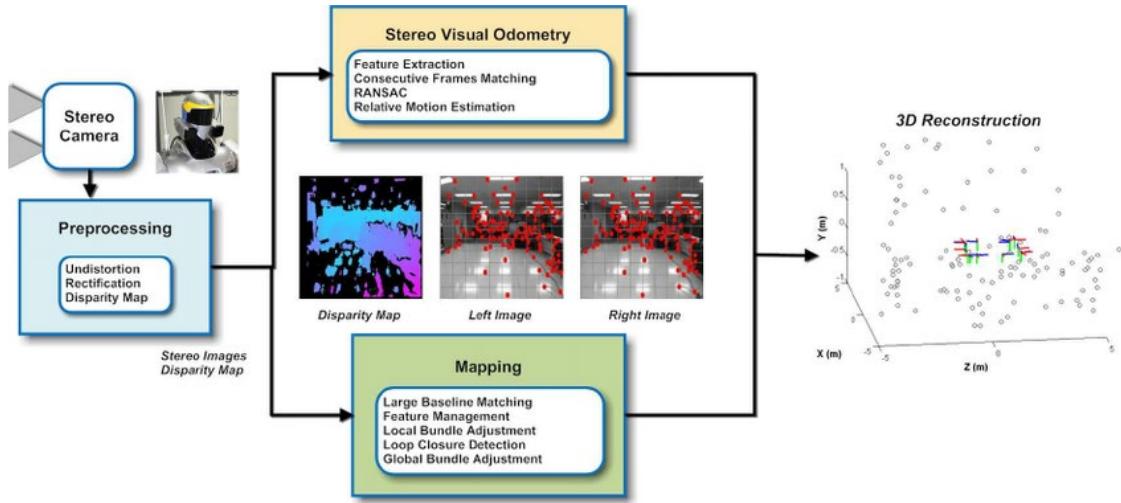


Figura 5: Generando un mapa de cámaras estéreos (*Stereo Visual SLAM System Overview: First, We Undistort and Rectify the...* | Download Scientific Diagram, n.d.).

Estos sistemas también se utilizan para generar mapas de entornos ya que tienen referencias más precisas de las que se obtienen con cámara monocular (Krombach et al., 2018). Se extraen las características del entorno y las distancias para que en cada fotograma que se detecte el movimiento se autogeneren los puntos y distancias generando mapas de tres dimensiones.



Figura 6: Tipo cámara estéreo ZED (ZED 2 - AI Stereo Camera | Stereolabs, n.d.).

En la Figura 6 se puede visualizar el tipo de cámara que se puede utilizar como sistema de captura estéreo. Puede ser que haya dos cámaras instaladas independientemente o que las cámaras estén dentro de un encapsulado como la que vemos en la Figura 6.

2.1.3. Cámaras estereoscópicas

Las cámaras estéreos no son las únicas que pueden ser más exactas. Las cámaras estereoscópicas son otro tipo de cámaras que pueden medir las distancias. Como ejemplo la cámara Intel RealSense D435i que tiene un sensor infrarrojo para la detección de profundidad con dos receptores diferentes, con similar función a la cámara estéreo, y la cámara RGB.



Figura 7: cámara Intel-RealSense-D435i (Intel, 2019).

Estas cámaras son denominadas RGB-D, ya que miden la profundidad mediante los sensores infrarrojos. Se pueden utilizar cada uno de los sensores por separado o también traen incorporada la opción de utilizar todos los sensores en conjunto. Utilizando todos los sensores en conjunto se puede obtener la imagen de la cámara con su respectiva profundidad. Esto no es práctico para sacar una imagen de un solo fotograma, la imagen que genera suele ser una imagen con manchas negras por la detección de la profundidad, ver Figura 8.

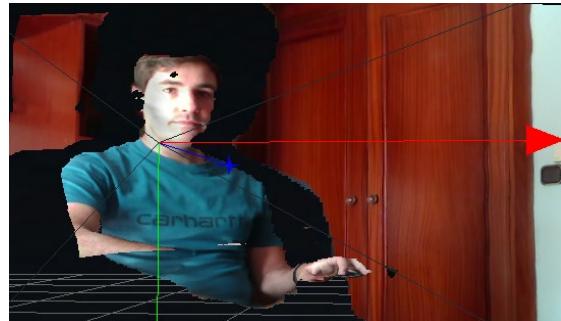


Figura 8: Imagen de profundidad de la cámara estereoscópica.

También lleva incorporado el sensor IMU lo que facilita el proceso de creación de mapas o entornos en tres dimensiones (Zhang et al., 2021). Existen diferentes algoritmos para la creación de mapas, Figura 9, que ayudan en la creación y entrenamiento de los robots para que se localicen e identifique su posición.

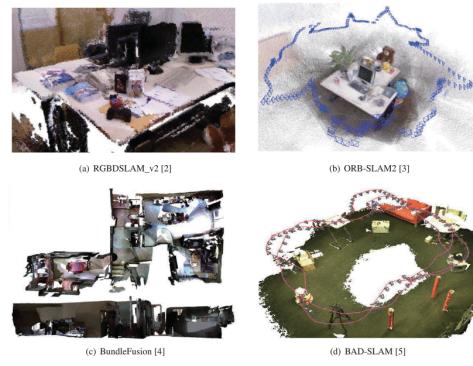


Figura 9: Creación de SLAM mediante cámaras estereoscópicas (Zhang et al., 2021).

2.1.4. LiDAR

A parte de las cámaras, hay sistemas para la detección de entornos. Uno de ellos se denomina detección de luz y rango (*LiDAR*, siglas en inglés), que se puede ver hoy en día en diferentes robots, como los robots aspiradores. Estos sistemas de captación crean un mapa tipo rejilla en dos dimensiones siendo capaces de identificar la localización del robot (Chang et al., 2020), ver Figura 10.

Todos los elementos mencionados previamente se utilizan para un objetivo similar, aunque en realidad sean diferentes. Por ello se han realizado varios estudios identificando puntos de mejora donde se puede ver que si se juntan estos sistemas de captación, los sistemas funcionan de una manera más eficiente (Shin et al., 2020) (Mu et al., 2020).

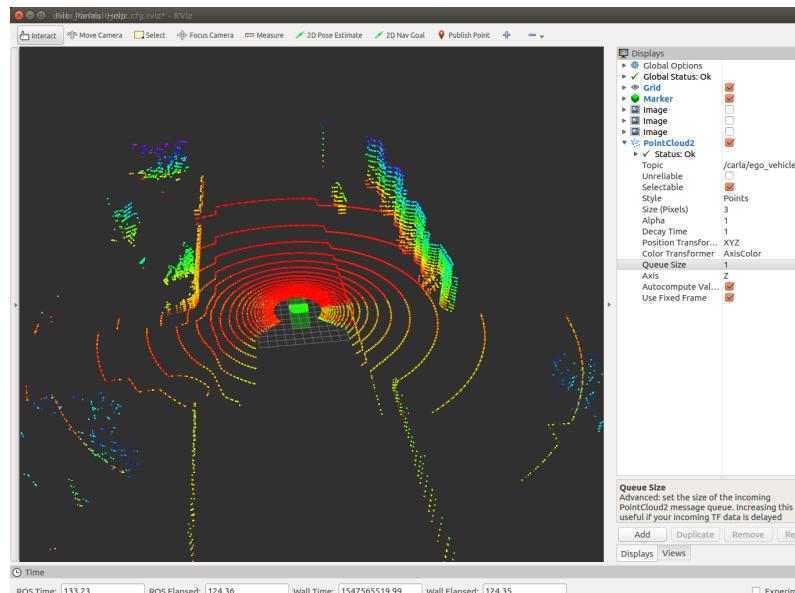


Figura 10. Mapa creado en *RViz* por un sensor *LiDAR* (*Incorrect Visualization of LiDAR · Issue #32 · Carla-Simulator/Ros-Bridge*, n.d.)

2.1.4. IMU

El sensor de inercia es un elemento que facilita el seguimiento de la posición en los robots móviles para su navegación. Estos elementos o sensores dan la habilidad de saber cómo se mueve el robot. Normalmente, constan de dos elementos separados en su interior, que medirán el giro y la aceleración.

Este elemento puede ser de gran utilidad si quieras utilizar tu robot con el sistema ROS. En el entorno o *framework Robot Operating System* este sistema se utiliza para estimar la posición del robot y además de eso es necesario si se quiere realizar un control exacto de las velocidades, porque los comandos de estos suelen ser la velocidad lineal y angular.

En caso de querer realizar un control de estas velocidades es necesario realizar un programa que controle el sistema mediante la adquisición de los datos de este sensor de inercia. Para un mejor entendimiento se pueden ver las tablas de la cámara Intel RealSense 435i que trae un sensor IMU integrado como se ha comentado anteriormente, ver Tabla 1 y Tabla 2 para ver las variables que se pueden obtener.

Tabla 1. Medidas de acelerómetro

Tipo	Acelerómetro		
Formato	MOTION_XYZ32F		
Numero de captura	5293		
Tiempo (ms)	1622619854455.03		
Eje (m/segundo^2)	0.0000000	-9.2084446	0.9316317

Tabla 2. Medidas del giroscopio

Tipo	Giro		
Formato	MOTION_XYZ32F		
Numero de captura	10688		
Tiempo (ms)	1622619824299.42		
Eje (º/ segundo)	-0.0052360	0.0000000	0.0017453

En las tablas se puede ver la importancia que tiene el tiempo de adquisición, ya que la variable suele cambiar mucho.

2.3. Detección y segmentación

Los sistemas descritos en la sección anterior, las de captación, pueden ayudar a crear mapas y con ello identificar por donde circula el robot. Pero en realidad, lo interesante es que el robot identifique el lugar donde se encuentra y además detecte los objetos que tiene alrededor para que no se choque y tenga una perspectiva más clara para realizar la circulación por el camino que le corresponda.

La detección y la segmentación de objetos son esas áreas que tratan que los robots autónomos identifiquen (Cao et al., 2019) y etiqueten (Mu et al., 2020) respectivamente los objetos de su entorno. Para ello es indispensable saber cómo se puede realizar un detector de objetos. Hoy en día, parece muy simple este concepto de detectar los objetos ya que lo podemos ver en distintas aplicaciones de nuestros móviles o en internet, pero es interesante saber cómo funciona para poder crear o modificar modelos de redes neuronales convolucionales (CNN) que realizan este proceso. La segmentación semántica de objetos se hace de una manera similar, pero en vez de detectar esos elementos e identificar su posición en la respectiva imagen, colorea la imagen de tal manera que se pueden diferenciar los objetos de forma semántica y de esta manera se crea un mapa de la misma imagen con números o nombres de los objetos identificados.

2.3.1. Detección de objetos

Como se menciona anteriormente, la detección de objetos es uno de los sistemas para que el robot identifique los elementos que le rodean. Es necesario realizar la ejecución estos algoritmos en los sistemas locales, ya que no se prescinde de mucho tiempo para ejecución de estos algoritmos (Xu & Wu, 2020).

Antes de nada, para crear estos modelos de detección de objetos es necesario saber cómo funcionan las CNN. Pero antes de estas redes tan complejas se empezó a hacer la detección de diferentes elementos mediante *Haar Cascade Classifiers*, que consta de filtros morfológicos con diferentes *kernel* para la detección de los elementos como puedes ser las caras humanas.

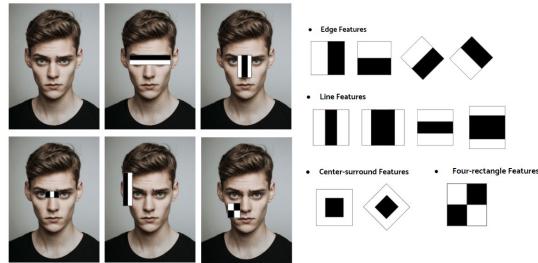


Figura 11: Ejemplo de detector de cara con *Haar Cascade* (Jiwon, 2019).

Estos sistemas pueden funcionar bien para ciertos elementos, pero hoy en día, hay diferentes modos de utilización de CNN más fiables. El proceso de detección de objetos se diferencia en tres fases que son la clasificación de imágenes (Chan et al., 2014), clasificación con localización del objeto y detección de diferentes objetos en una sola imagen (Ren et al., 2017).

Los clasificadores han evolucionado para mejorar la eficiencia, muestra de ello es la Figura 12, que ya en el año 2017 había mejorado mucho el tiempo de ejecución de detección de objetos. Hoy en día, ya se realiza la ejecución casi a tiempo real, muestra de ello es Yolov4 (Bochkovskiy et al., 2020) que realiza un buen desempeño teniendo en cuenta la velocidad o el tiempo de respuesta que tiene el algoritmo.

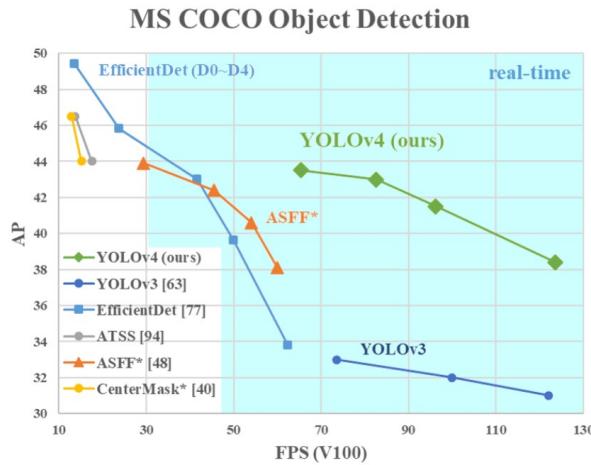


Figura 12: Comparando los algoritmos de detección de objetos en base al tiempo de respuesta(Bochkovskiy et al., 2020).

Uno de los principales motivos de las mejoras de estos sistemas son las competiciones de Imagenet donde las empresas más punteras en el sector de la inteligencia artificial (IA) toman parte. Podemos ver en la Figura 13 como han evolucionado durante años los resultados de estas redes neuronales.

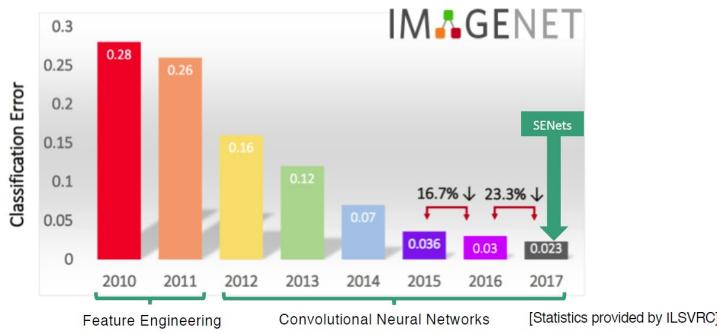


Figura 13: Evolución de Imagenet (*Review: SENet — Squeeze-and-Excitation Network, Winner of ILSVRC 2017 (Image Classification) | by Sik-Ho Tsang | Towards Data Science, n.d.*).

La detección de objetos en una imagen se puede realizar tanto en dos dimensiones como en tres, utilizando para ello los elementos descritos en el apartado de SLAM mediante visión por computador. La detección volumétrica de los elementos es realmente interesante para los robots autónomos, de esta manera, pueden realizar las mediciones de distancias.

2.3.2. Segmentación

En adición a todo lo anterior, la segmentación es uno de los métodos más utilizados últimamente en los sistemas de navegación. Se puede decir que es una extensión a la detección de objetos, para que los sistemas computacionales entiendan mejor todo lo que están visualizando (Wolf et al., 2014).

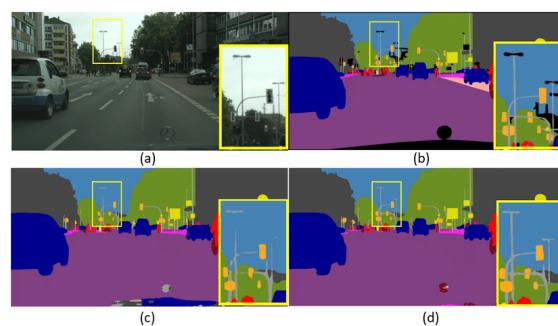


Figura 14: Imagen exterior segmentada (*Illustration of Challenges in Semantic Segmentation. (a), Input Image.... | Download Scientific Diagram, n.d.*).

La segmentación puede ser realizada por diferentes tipos de algoritmos con los distintos sistemas vistos anteriormente. Se pueden ver sistemas monoculares (Y. Wang et al., 2020), sistemas que utilizan profundidad (Seichter et al., 2020), mapas creadas en función de la segmentación (Deng et al., 2020), para la navegación interior (Teso-Fz-Betoño et al., 2020) o para navegación exterior añadiendo detección de objetos (Feng et al., 2021).

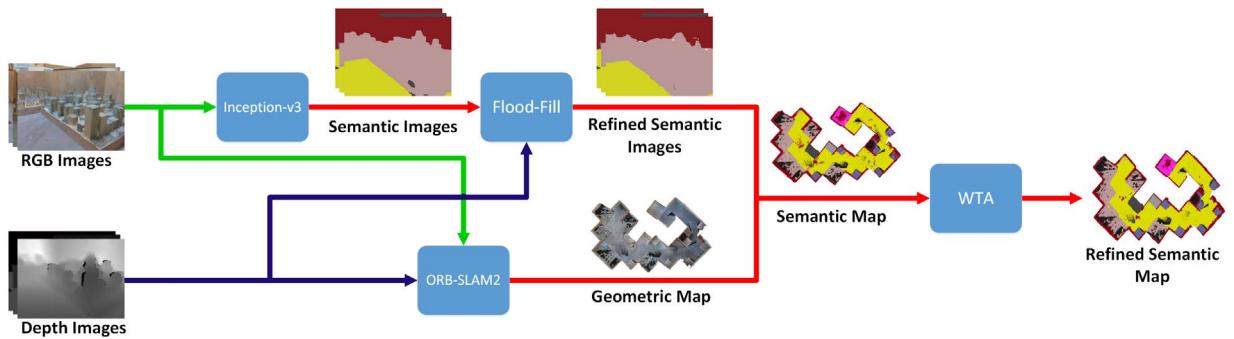


Figura 15: Mapa semántico para navegación (Deng et al., 2020).

Uno de los grandes problemas para los robots es el tiempo de ejecución de estos modelos (Li et al., 2020). Otra cosa tener en cuenta es que las personas no son elementos fijos en estos entornos, por lo que es necesario borrarlos para realizar mapas segmentadas (Ai et al., 2020). También es posible realizar el mapa segmentado mediante un sistema monocular que capta el entorno y lo segmenta (Miyamoto et al., 2020).

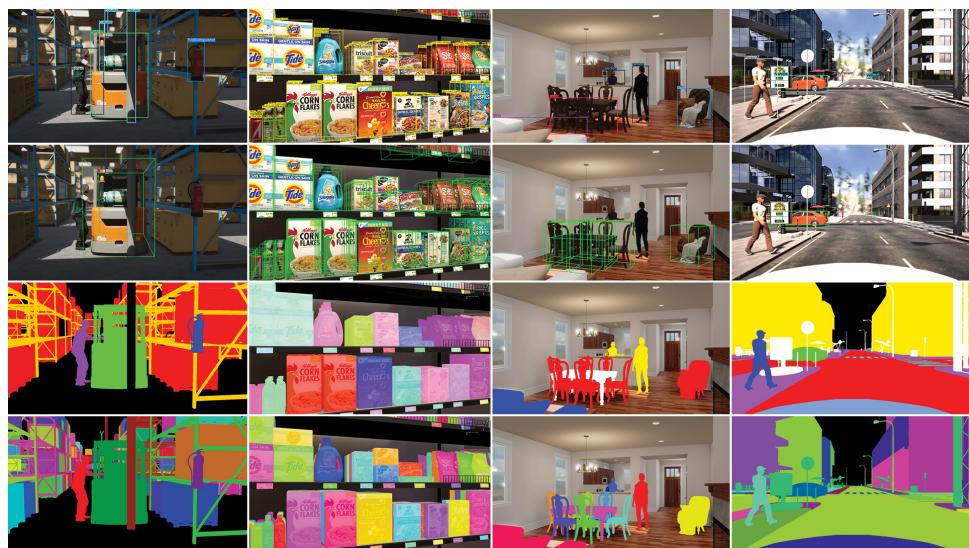


Figura 16: Detección de objetos y segmentación de una imagen (*Supercharge Your Computer Vision Models with Synthetic Datasets Built by Unity | Unity Blog*, n.d.).

Estos algoritmos de segmentación pueden ser entrenados de diferentes maneras, supervisado y no supervisado. La diferencia entre ellos es que el no supervisado (Jin et al., 2020), como bien dice el nombre, no se entrena o no realiza el entrenamiento con los resultados, en cambio, el supervisado se entrena con el resultado para que puede obtener el mejor resultado posible.

2.3.2. Posición y trayectoria del ser humano

En todos los casos vistos anteriormente, es indispensable saber el estado del ser humano, donde se encuentra y cuál es su pose (Cui et al., 2020). Esto nos ayuda en evitar las colisiones o mantener distancias de seguridad por si el ser humano se mueve.



Figura 17: Estimación de posicionamiento de humanos en entorno segmentado (Seichter et al., 2020).

Para la eficiencia de un robot autónomo es de gran ayuda saber por dónde andarán los humanos (Tamaki et al., 2019) o también predecir por donde pueden ir para evitar esa trayectoria y elegir la más rápida posible (Liu et al., 2015), ayudando con la gesticulación y mirada del mismo (Moors et al., 2015).

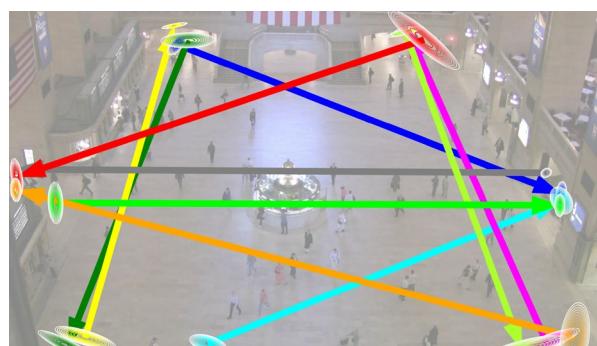


Figura 18: Segmentación de trayectorias después de procesado (Tamaki et al., 2019).

Para poder realizar todo lo mencionado con un buen funcionamiento se necesitan entender bien los entornos y entrenar los algoritmos de una manera eficaz a como se hace para cada lugar donde se encontrará el robot autónomo. Para ello es necesario tener bases de datos de distintos lugares como supermercados (Lewandowski et al., 2020), hospitales (H. M. Gross et al., 2017) o domésticos (H.-M. Gross et al., 2019).

Si se quiere predecir lo que va a hacer un ser humano, es importante tener en cuenta el tiempo, por lo que es necesario analizar lo que realiza un ser humano durante un tiempo determinado para poder predecir las próximas veces lo que pueda hacer.

Para poder definir bien el método de captación y realizar estas previsiones se necesitan las redes neuronales recurrentes, que serán los que analizarán el proceso de los movimientos para una previsión posterior de lo que pueda suceder.

Hay casos en los que se realiza la búsqueda de una trayectoria de las personas para que el robot pueda ayudar y con la previsión realizada minimizar el estorbo y el tiempo de espera del humano (Bruckschen et al., 2020). En este artículo se demuestra que los robots pueden predecir los movimientos de los humanos para no estorbar y además de eso, puede anteponerse a la situación que va a suceder.

Tal y como se puede ver en la Figura 19, la persona que se está moviendo lleva un táper o algo similar en las manos, también puede ser algún alimento o algún utensilio de cocina. El robot detecta al humano y su movimiento, para después identificar el elemento que lleva en las manos y predecir para donde se va a dirigir la persona. Se puede ver la trayectoria del humano, la trayectoria de color morado, que es la predicha como la más probable y por esa causa, el robot evita realizar su navegación por ese lugar, la trayectoria de color blanco. Esa decisión la realiza para no estorbar a las personas y además de eso realiza otra búsqueda de navegación para poder llegar antes al lugar deseado.

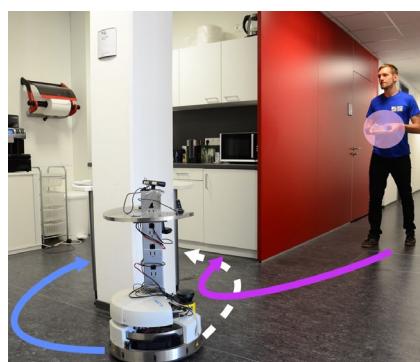


Figura 19. Detección de trayectoria para cálculo de navegación (Bruckschen et al., 2020).

Tal y como se ha realizado la búsqueda, también se puede hacer un mapa de movimientos realizados como se muestra en el artículo (Munaro & Menegatti, 2014). En este artículo se muestra que se pueden realizar los mapas según los movimientos de los humanos, por lo que se crea un mapa de dos dimensiones con las trayectorias de las personas que han pasado por donde se encuentra el robot, se pueden ver los mapas creados en la Figura 20. Esto puede ayudar a la hora de crear un diseño personalizado de un sistema para prever los siguientes estados que se puedan generar.

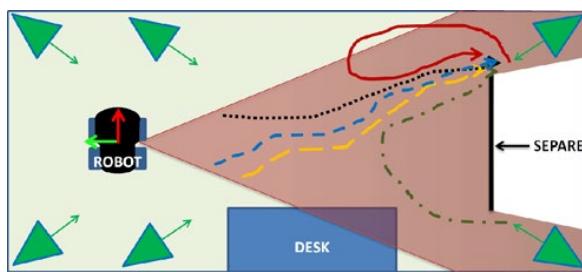


Figura 20. Se muestran las diferentes trayectorias de las personas.

2.4 Resumen

Al analizar el estado del arte de los segmentadores semánticos, se ha visto que existen mejoras aplicables en cuanto al análisis de movimientos y la predicción futura. En este concreto caso vemos la posibilidad de mejorar los modelos existentes con la previsión de esos movimientos aplicado a la segmentación. Para realizar un sistema completo, se ha visto que es necesario obtener la información del entorno de una forma correcta y aplicar los algoritmos o redes neuronales que se utilizan en aprendizaje automático. También es importante que pueda realizar la segmentación de interiores de una forma correcta con un índice de *MIoU* (*mean intersection over union*) alto además de las métricas generales, con números bastante admisibles y que al mismo tiempo sea capaz de predecir lo que pueda suceder en un futuro, es decir, los movimientos del ser humano o cualquier otro obstáculo que pueda interferir. Para ello, se necesita una red neuronal con una arquitectura bastante compleja. Ya que el entrenamiento de esta red neuronal puede variar mucho según la base de datos utilizada o los hiperparámetros que se elijan, puede ser interesante referirse a los modelos previamente existentes a la hora de empezar a realizar el modelo y entrenamiento.

3. Objetivos y metodología de trabajo

El análisis previamente realizado nos lleva a la conclusión de que los sistemas de captación y procesamiento están mejorando de forma continua. Esto conlleva al aumento del uso de los robots en entornos sociales, siendo estos robots autónomos cada vez más seguros. Por la causa de la seguridad de los robots autónomos y las mejoras, se decide realizar este proyecto donde se pretende mejorar el sistema de navegación realizando un avance en los procesos actuales, valorando los resultados obtenidos.

3.1. Objetivo general

El objetivo general es realizar la segmentación con una predicción de los movimientos en entornos interiores. Para realizar las pruebas, se utiliza un robot móvil pequeño de código abierto que permite ajustarlo a las necesidades que conlleva el proyecto, suficiente para realizar las pruebas y obtener unos resultados en función del trabajo realizado. Este robot se llama Jetbot, es un robot creado por Nvidia que puede ser montado por cualquiera que tenga una impresora 3D.

En el proceso de ejecución del proyecto, se intentará mejorar el sistema de navegación, creando unas trayectorias inteligentes que permitan evitar estorbar o colisionar con cualquier tipo de objeto que se ponga en su trayectoria. De esta manera, se lograría proteger la integridad y aumentar la eficiencia a la hora de navegar utilizando para ello las técnicas de aprendizaje profundo.

3.2. Objetivos específicos

En este trabajo se van a realizar las siguientes tareas que ayudaran a describir mejor las especificaciones técnicas.

- Realizar análisis del estado del arte para sistemas de segmentación para navegación autónoma de robots y creación de SLAM
- Detección de personas para asegurarnos de que no haya colisión entre robot y humano.
- Detectar movimiento en el sistema y predecir su trayectoria para poder esquivar y prevenir el paro del robot.

- Realizar o elegir el algoritmo de segmentación semántica que se ajuste a las necesidades.
- Elegir el algoritmo de navegación que se ajuste a las necesidades del sistema y probar si es la correcta.
- Juntar los objetivos anteriores y valorar si el sistema de predicción de trayectorias es eficiente.
- Implementar el resultado del proceso en la tarjeta de Nvidia Jetson Nano para poder probarlo con Jetbot, el robot creado para ello.
- Realizar la implementación en ROS.

3.3. Metodología del trabajo

Para poder llevar a cabo los objetivos descritos se establecerá una cierta estrategia. Esta estrategia o metodología será la denominada Scrum, que consiste en objetivo a corto plazo o sprints que por si se cambian los pequeños objetivos u ocurre algún cambio en el desarrollo. Se puede ver el ejemplo en la Figura 19. Esta metodología se utiliza mucho en los desarrollos software, que normalmente suelen ser complejos y pueden producirse cambios en cualquier momento durante el transcurso del proyecto.



Figura 21: Ejemplo de la metodología Scrum (SCRUM - ECLEE | European Center for Leadership and Entrepreneurship Education, n.d.).

Para poder empezar a plantear esta metodología es necesario tener un proyecto y realizar una planificación u objetivos previos. Esta planificación no va a ser la definitiva, pero ayudara en tener un objetivo fijo y tener idea de que trata el proyecto, es decir, tener una visión clara del proyecto en general. Para poder realizar este planteamiento es necesario realizar el

estudio del proyecto, analizar la información que existe y mirar si hay recursos necesarios para poder llevar a cabo el proyecto.

Teniendo la información analizada y los recursos a disposición, se realizará el planteamiento general, que a su vez contendrá los pequeños esprints u objetivos que se marcarán para el cumplimiento del proyecto. Hay que tener claro de que cada sprint no está estrictamente definido, por lo que se realizará este proceso de Scrum cada vez que trabajemos en el proyecto, ya sea para repasar lo hecho hasta ese punto o realizar algún cambio que se ha detectado.

Cada sprint consistirá en distintas tareas que habrá que llevarlas a cabo en un tiempo determinado, siendo este tiempo flexible. Una vez realizada la tarea, es conveniente que sea subida a una plataforma de desarrollo colaborativo para que los integrantes puedan analizar lo que se ha hecho. En este caso al ser un solo integrante no es necesario subir, pero es interesante para llevar un control de versiones.

Puede ser que cada sprint tenga sus apartados o subtareas a realizar. Por ello en cada sprint o subtarea, según donde se encuentre se analizará lo que se ha realizado, como se ha realizado y si existen otras maneras de abordar el problema para poder buscar mejoras para del proyecto.

En este caso los objetivos serán los sprints generales que se han definido y estos tendrán una fecha específica para cumplir el plazo de entrega. Como se comenta anteriormente se evaluará y valorará el trabajo realizado y respecto a esa evaluación se realizarán las modificaciones necesarias para que el proyecto avance.

Para una mejor gestión de este sistema se realizará el concepto de Kanban para que en todo momento se sepa el estado del proyecto. En Kanban se encontrarán los sprints que contendrán las subtareas y en todo momento se podrá saber el estado de cada sprint.

Podemos ver la Tabla 3 donde se definen los sprints del proyecto. En este proyecto se definen los sprints pensando que todo lo que se realiza funcionará a la primera, con lo que podemos decir que no es un Kanban realista, pero incita a que se cumplan las fechas señaladas.

Tabla 3: Planificación de esprints para el proyecto

ESPRINT	OBJETIVO	FECHA ESTIMADA
1	Análisis o estado del arte	6/05/2021
2	Realizar o elegir algoritmo de segmentación	20/05/2021
3	Realizar la predicción de futuros escenarios	30/05/2021
4	Elegir el algoritmo de navegación	02/06/2021
5	Juntar los algoritmos para implementarlos en Jetbot	10/06/2021
6	Implementación de los algoritmos en el Sistema ROS	20/06/2021
7	Pruebas para comprobar y calificar el funcionamiento	24/06/2021

4. Identificación de requisitos

Para el desarrollo del problema indicado, la segmentación semántica y predicción de movimiento para una mejor navegación de los robots autónomos, se considera útil definir bien el problema para su mejor entendimiento, donde se identifican las necesidades para realizar el desarrollo software y hardware de la herramienta.

Tabla 4. Tabla de requisitos

Función	Resultado	Ejemplo
Detección de camino libre.	Valor booleano: <ul style="list-style-type: none"> - Si - No 	Si se detecta suelo en la segmentación: Si Si se no detecta suelo en la segmentación: No
Detección de personas.	Valor booleano: <ul style="list-style-type: none"> - Si - No 	Si se detecta persona: Si Si se no detecta persona: No
Predicción de un estado futuro en segmentación semántica.	Dibujar la imagen con la segmentación semántica.	
Navegación.	Se toman las decisiones necesarias para que el robot pueda llegar al lugar indicado. <ul style="list-style-type: none"> - Elegir la ruta - Cambio de ruta 	

Los requisitos del sistema se muestran en la Tabla 4, donde se muestran las funciones iniciales requeridos. Es necesario tener un algoritmo de segmentación semántica para diferenciar los distintos elementos que se encuentran en el entorno y definir bien el camino que se encuentra a disposición del robot. Para realizar la segmentación, se requiere de un modelo de aprendizaje profundo que debe ser capaz de diferenciar los elementos en colores para su posterior tratamiento de navegación. Debe tener un sistema o algoritmo de detección de personas que, al mismo tiempo, en caso de detectar una persona realice el seguimiento para predecir su trayectoria. Para la predicción de trayectoria también se requiere de un modelo de aprendizaje profundo que prediga los pasos que puede dar esa persona y dibujar

en el mapa el camino hacia donde pueda moverse. Al mismo tiempo que se realiza la segmentación. Para la navegación es necesario utilizar algoritmos de búsqueda y árboles de comportamiento.

Los diferentes tipos de algoritmos van a ser analizados y en caso de que existan o se puedan utilizar opciones externas que realicen el trabajo, se analizan y se elige el que mejor se adapta al sistema utilizado. Las especificaciones de los algoritmos se encuentran en la Tabla 5.

Tabla 5. Técnicas de la herramienta

Etapa	Algoritmo/Técnica	Fuente
Segmentación semántica RGBD.	Entrenamiento de una red neuronal profundo con <i>encoder</i> y <i>decoder</i> . Que realizara la segmentación en base a 2 entradas, RGB y profundidad.	Autoría propia.
Reconocimiento y detección de trayectoria.	Entrenamiento de una red neuronal profundo con <i>encoder</i> con LSTM y <i>decoder</i> . Que realizara la segmentación en base a 2 entradas, RGB y profundidad.	Autoría propia.
Navegación y toma de decisiones	Algoritmo de búsqueda para que un robot se mueva de A hasta B.	Nav2 (ROS <i>Navigaiton Stack</i>)

Es imprescindible que la información recibida por la cámara se procese de forma adecuada para que el sistema de navegación pueda realizar un desempeño correcto a la hora de la búsqueda. Por eso es por lo que se han analizado los algoritmos de segmentación realizados por expertos y en base a sus resultados se ha realizado uno que tenga las características de algunos de ellos. Además de eso, se decide analizar los algoritmos o modelos que utilizan o trabajan con capturas de imágenes y su profundidad para un mejor funcionamiento.

En cuanto a la detección de movimiento, es necesario definir cuál es el resultado que se quiere obtener, utilizando para ello la profundidad como medidor de distancia. De esta manera, con

el procesado de la ejecución de secuencias, se puede predecir la dirección de movimiento o las personas identificadas.

5. Descripción de la herramienta software desarrollada

El presente proyecto consiste en avanzar la segmentación semántica y contribuir en el apartado de previsión de los movimientos de las personas para un mejor funcionamiento de la navegación autónoma del robot utilizando el software libre ROS para facilitar el uso del software creado. Para la implementación del sistema se ha escogido la tarjeta NVIDIA Jetson Nano con el que muchos desarrolladores realizan sus propios robots o prototipos de robótica, sus especificaciones se indican en la Tabla 6.

Para el apartado de visión, se utiliza la cámara Intel-RealSense D435i que permite realizar capturas en color y profundidad para detectar elementos del entorno y las distancias a las que se encuentra, sus especificaciones indicadas en la Tabla 7. De esta manera se mide la distancia a la que se encuentran los objetos para su posterior análisis secuencial para poder predecir el movimiento futuro.

La herramienta desarrollada en Python utiliza algunas librerías externas para el procesamiento de imágenes en el entrenamiento y también para la creación de las redes neuronales para su posterior tratamiento.

Lo ideal es tener un solo sistema que realice la segmentación y la predicción del movimiento, pero al no tener una base de datos con la segmentación y los movimientos segmentados todo en uno, se ha decidido realizar dos sistemas diferentes que funcionen uno en base al otro, es decir, que las imágenes predichas por el segmentador semántico se utilicen para entrenar el modelo de predicción.

Tabla 6. Especificaciones de la tarjeta Jetson Nano

GPU	128 núcleos de NVIDIA CUDA para ejecución paralela
CPU	ARM Cortex®-A57 de 4 núcleos
Memoria	LPDDR4 de 4 GB y 64 bits

Tabla 7. Especificaciones de la cámara Intel-RealSense D435i

Rango	0.3m a 3 m
Profundidad	A partir de 28 cm y una exactitud de 2% a los 2 m
	Resolución: hasta 1280 × 720
RGB	Resolución: hasta 1920 × 1080
	RGB frame rate: 30 fps

5.1. Arquitectura del hardware

El robot con el que se realizan todas las pruebas es Jetbot, el robot educacional creado por los desarrolladores de NVIDIA. Este robot consta de la tarjeta Jetson Nano mencionada anteriormente, con una cámara monocular que se ha decidido cambiar por la cámara Intel-RealSense D435i para un mejor funcionamiento del sistema de segmentación. También contiene dos motores con su controlador de puente H para poder hacer el control de velocidad de estos motores. Se puede ver el Jetbot configurado a nuestra manera en la Figura 22.



Figura 22. Imagen de Jetbot con la cámara integrada.

El procesador Jetson Nano tiene instalado el sistema operativo Ubuntu 18.04 LST para poder utilizarlo como un ordenador. Esta tarjeta se utiliza modo embebido para un funcionamiento más rápido. En caso de utilizar sistemas en la nube hay que tener en cuenta que el robot debe tener en todo momento buena conexión a internet, por lo que se descarta esa opción.

Al tener instalado el sistema operativo Ubuntu, se pueden instalar paquetes adicionales para su mejor funcionamiento. Por eso, se instalan los paquetes de ROS necesarios. Algunos de los paquetes instalados son los que en la actualidad no se soportan en Ubuntu 18, pero los desarrolladores han puesto contenedores Docker que se pueden utilizar para la instalación de cualquier versión de ROS que tenga soporte.

5.2. Utilización de la cámara Intel-RealSense

Las cámaras creadas para la visión RGB-D suelen tener su propio funcionamiento y adquieren los datos de una manera particular. Las cámaras monoculares adquieren una imagen en tres colores, rojo, verde y azul, y con la mezcla de estas tres se construye la imagen. La “D” hace referencia a la profundidad de una imagen, que se representa con la distancia desde la cámara

hasta donde se encuentra el objeto o el elemento capturado. Por ello se va a utilizar la librería del fabricante pyrealsense2 y las librerías creadas en ROS para obtener las imágenes a tiempo real, las cuales son especializadas en capturas de la cámara por separado, por un lado, la captura RGB y por otro la profundidad o directamente la captura de la imagen en RGB-D.

Esta cámara adquiere también información sobre la inercia para detectar la posición en la que se encuentra el dispositivo en cada momento. Estas son dos variables, la aceleración y el giroscopio, que se adquieren también por la librería del fabricante.

Para realizar las pruebas de la velocidad de captura de imágenes se ha realizado un script en Python para medir el FPS (*Frames Per Second*) que mide cuantas fotos es capaz de sacar en un segundo.

Se han probado distintas dimensiones de captura de imágenes para saber a qué velocidad es posible realizar esta captura.

- Si se pone la captura de las dimensiones en 1920x1080, se puede ver que la cámara RGB realiza la captura de 8 imágenes por segundo, pero la cámara infrarroja no puede detectar tantos pixeles ya que su máximo es 1280x960. Los resultados se muestran en la Figura 23.

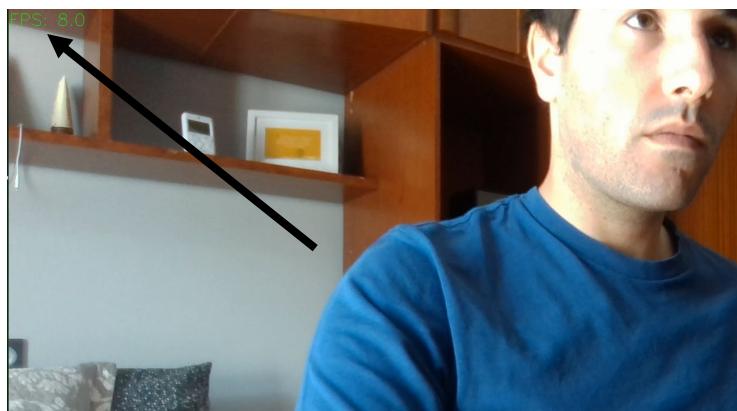


Figura 23. Imagen RGB de resolución 1920x1080

- Si se pone la captura de las dimensiones en 1280x960, se puede ver que la cámara RGB realiza la captura de 15 imágenes por segundo, pero la cámara infrarroja realiza la captura de 6 imágenes por segundo. Los resultados se muestran en la Figura 24 y 25.



Figura 24. Imagen RGB de resolución 1280x960



Figura 25. Imagen de profundidad de resolución 1280x960

- Si se pone la captura de las dimensiones en 640x480, se puede ver que la cámara RGB realiza la captura de 16 imágenes por segundo, pero la cámara infrarroja realiza la captura de 16 imágenes por segundo. Los resultados se muestran en la Figura 26 y 27.

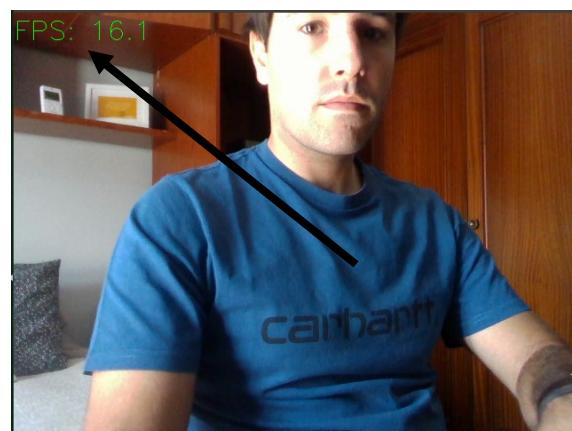


Figura 26. Imagen RGB de resolución 640x480



Figura 27. Imagen de profundidad de resolución 640x480

En las imágenes capturadas se muestra la velocidad de captura de imagen que se obtiene para las diferentes cámaras utilizando para ello la librería OpenCV. Pero existe la opción de realizar lo mismo y más cosas utilizando la librería pyrealsense2 que proporciona la facilidad de realizar la adquisición de las distancias, cámara RGB, la imagen RGB-D y también los datos proporcionados por los sensores de inercia.

Se han realizado diferentes captaciones en ROS con las librerías o los servicios que ofrece Intel para su comunicación en este sistema y se han logrado obtener 15 fotogramas por segundo.

5.3. Elección del algoritmo de segmentación semántica RGB-D

Para elegir el algoritmo de segmentación se ha de realizar la búsqueda de algunos algoritmos de código abierto. Esto se hace por causa de que estas redes neuronales están bastante logradas y la creación y entrenamiento de una red neuronal es compleja y podría tardarse mucho en generar y entrenar. Antes de realizar ninguna selección se ha realizado la búsqueda de diferentes algoritmos y se ha visto que algunos de los que obtienen los mejores resultados son los siguientes, los resultados de los algoritmos se puede ver en el artículo de ESANet (Seichter et al., 2020):

- ESANet-R34-NBt1D.
- SA-Gate.
- RDFNet.

- SGNet.
- ACNet.
- ERFNet

Estos resultados son los que se han obtenido en la tarjeta Jetson AGX Xavier, pero se va a realizar un sistema parecido en el Jetson Nano, para ello asumiendo que los resultados obtenidos serán proporcionales en cuanto a exigencias de computación. Para la creación del modelo, se ha creado un modelo propio similar a los que se utilizan en las investigaciones de estos modelos diferentes. Aun y todo, se les dará importancia a los resultados obtenidos, pero también a la captura de imágenes por segundo que es capaz de procesar el sistema, ya que con ese procesamiento el robot tiene que funcionar a tiempo real.

Para el entrenamiento del modelo, se utilizará la base de datos SunRGBD (Song et al., 2015), ya que se ha visto que en algunos casos ha dado mejores resultados que con NYUV2 (Silberman et al., 2012). Pero en caso de querer realizar un modelo general, sería conveniente entrenarlo y testearlo con ambas bases de datos y tener una visión más específica de como adaptarlo al entorno donde se utilizará el robot. También se pueden utilizar otras bases de datos como Cityscapes (Cordts et al., 2016) para exteriores, habitualmente utilizados para vehículos autónomos o robots para exteriores.

Estos algoritmos de segmentación suelen tener un codificador, “encoder” en inglés, y un descodificador, denominado “decoder” en el dialecto británico, para que puedan funcionar correctamente. El codificador es una red neuronal convolucional normal, que realiza las convoluciones para la identificación de los elementos y sacar la imagen de características. En este caso en concreto se ha utilizado el sistema de Resnet (He et al., 2016) Figura 28 como principal método de *encoder*, que en realidad quiere decir “*Residual Network*”. Estos bloques de aprendizaje residual nos ayudaran a que el sistema aprenda mejor la función que tiene que realizar.

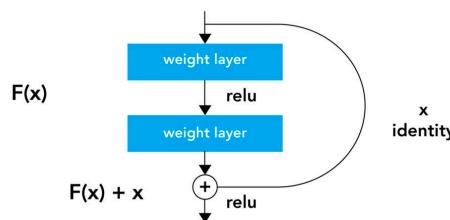


Figura 28: Se muestra cómo se realiza el bloque residual del aprendizaje (*What Is Resnet or Residual Network | How Resnet Helps?*, n.d.).

Una vez extraídas las características, hay que generar la imagen resultante. Para ello se utiliza el decodificador, que realiza la máscara de la imagen partiendo de la imagen del codificador. Como bloque principal de decodificación, se quería utilizar el sistema de Non-Bottleneck-1D-blocks Figura 29, para un funcionamiento óptimo que es utilizado en ERFNet (He et al., 2016) que reduce la dimensión de los pesos generados por la red, pero al requerir un entrenamiento más complejo se ha optado a no utilizarlo.

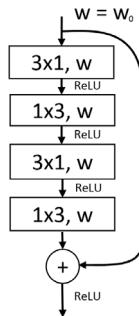


Figura 29: Se muestra cómo se realiza el bloque Non-Bottleneck-1D (He et al., 2016).

Para crear el modelo entero se ha escogido el modelo U-Net (Weng & Zhu, 2021), que en su día evolucionó los modelos para mejorar la velocidad de respuesta. Para ello, es necesario realizar diferentes “encoder” y “decoder”, donde en cada uno de los codificadores se guarda el resultado para poder utilizarlo en el decodificador, ver Figura 30. En este caso se han utilizado 4 codificadores Resnet y otro 4 decodificadores Non-Bottleneck-1D-blocks

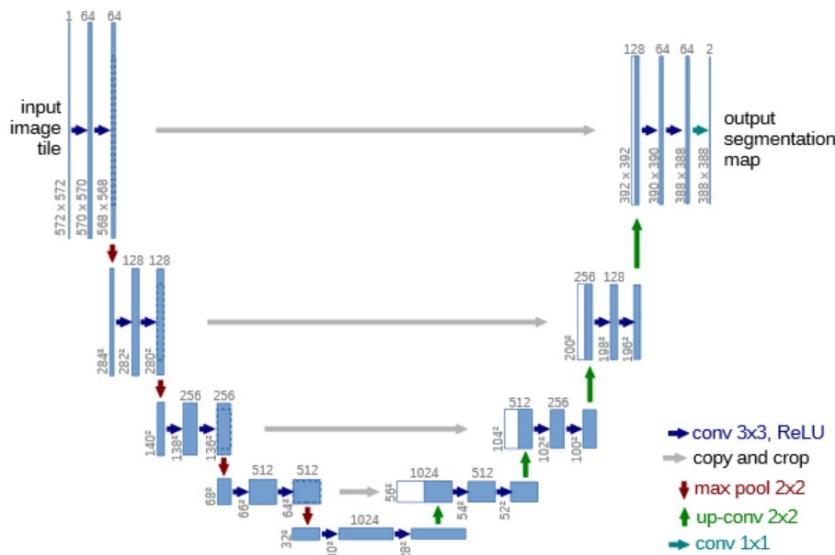


Figura 30: Se muestra el modelo U-Net (Weng & Zhu, 2021).

Para realizar la fusión de la imagen RGB con la imagen de profundidad se ha utilizado el modelo de ESANet, que realiza una especie de suma de las dos imágenes con su *AveragePooling* con su respectiva convolución multiplicada. Se puede ver el proceso de la fusión de imágenes en la Figura 31.

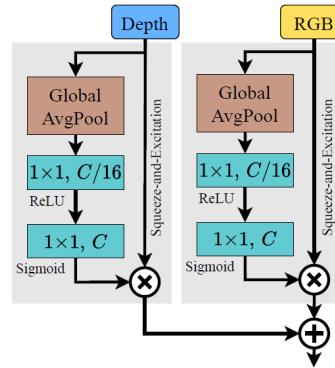


Figura 31: Se muestra la fusión del modelo ESANet (Seichter et al., 2020).

Tras haber realizado modelo, se definen las pérdidas y métricas que se utilizaran para valorar el modelo.

Las pérdidas se utilizan para realizar el cálculo del coste durante el entrenamiento del modelo. En este caso se definen diferentes cálculos que facilitaran el entrenamiento del modelo.

- *Dice_coef*: es un coeficiente estadístico para comparar la similitud de dos imágenes.
- *Dice_loss*: la perdida se calcula realizando la resta de $1 - dice_coef$, indicado anteriormente.
- *Bce_dice_loss*: Es la suma de *binary crossentropy* y *dice_loss*, donde *binary crossentropy* es la probabilidad de que el píxel del resultado sea 1 o 0 calculado desde la distancia del valor esperado.
- *Bce_logdice_loss*: Es la suma de *binary crossentropy* y el logaritmo de *dice_loss*.
- *Weighted_bce_loss*: Es *bce_loss* (*binary crossentropy*) utilizando el peso para modificar el resultado.
- *Weighted_dice_loss*: Es *dice_loss* utilizando el peso para modificar el resultado.
- *Weighted_bce_dice_loss*: Es *bce_dice_loss* utilizando el peso para modificar el resultado.
- *True_positive_rate*: El porcentaje de aciertos en pixeles que se dan en cada imagen.

Una vez definidos las métricas para calcular las pérdidas, se definen los *callbacks* que se utilizaran durante el entrenamiento. Es decir, las funciones que se ejecutarán cada vez que en el entrenamiento se pasa el *dataset* entero, lo que se denomina *epoch*. Los *callback* utilizados son los siguientes.

- *Checkpoint*: Guarda los valores de los pesos asignados cada vez que el modelo mejora. De esta manera, aunque se pare la ejecución del entrenamiento, los pesos del modelo entrenado quedan guardados.
- *ReduceLROnPlateau*: Reduce el *learning rate* del optimizador si el modelo no se está entrenando correctamente y no mejora la pérdida del modelo, reduzca el *learning rate*.
- *LrScheduler*: Reduce el *learning rate* según lo establecido. Se puede utilizar para que vaya reduciendo según pasa el tiempo o por los *epoch* que han pasado entrenando el modelo. En este caso se utiliza el segundo comentado, para que vaya reduciendo el lr según avanza el entrenamiento.
- *EarlyStopping*: El entrenamiento se para si no está mejorando.

5.4 Entrenamiento del modelo de segmentación semántica

El entrenamiento de una red neuronal es fundamental para que funcione correctamente. Por ello, lo primero es entender la base de datos a utilizar y establecer el método para tratarla. En este caso, se han escogido las bases de datos de NYUv2 y SUNRGBD donde se encuentran las imágenes de distintos escenarios. Para el manejo de estas bases de datos se han utilizado las herramientas proporcionadas por sus creadores.

5.4.1 Datasets

La base de datos NYUv2, realizada con la cámara de Kinect v2, contiene 1449 imágenes, que cada una de ellas tiene su imagen RGB, imagen de profundidad y la imagen segmentada. Existen diferentes categorías de etiquetado, en concreto, hay 894 clases etiquetadas que se analizarán para entrenar el modelo. Todas las imágenes de NYUv2 tienen una dimensión de 480x640 pixeles. En cambio, la base de datos de SUNRGBD tiene opción de elegir la cámara (Kinect, Xtion o RealSense) con la que se han realizado las fotos para su posterior entrenamiento. Este *dataset* tiene 1159 imágenes con 47 categorías o clases, que como el *dataset* anterior, cada una contiene su imagen RGB, imagen de profundidad y la imagen segmentada. Todas las imágenes de SUNRGBD tienen una dimensión de 531x681 pixeles.

Estas bases de datos contienen las etiquetas de las imágenes con las que se entrena el modelo previamente creado. Para poder entrenar estos modelos es indispensable saber las etiquetas a las que corresponde cada valor y preparar los datos que se quieren obtener como resultado. Es necesario modificar y crear una base de datos nueva para que el entrenador pueda entender como entrenar el modelo, por ello se realiza una nueva imagen para cada número o clase. De esta manera si tenemos diez clases, con el resultado de la imagen segmentada, obtendremos 10 imágenes que contienen 1 si la clase de ese píxel es la adecuada y 0 si no es la adecuada. Estas imágenes contienen las mismas dimensiones que las anteriores, pero suelen estar en blanco y negro ya que es la máscara que indica si el píxel es de una clase o no.

Tras haber preparado las bases de datos de NYUv2 y SUNRGBD para su entrenamiento, se pueden ver en las Figuras 32 y 33 dos ejemplos de imágenes ya preparadas. En ellas se pueden ver la imagen RGB (llamada *Color*), la imagen de profundidad (llamada *Depth*) y el resultado de la segmentación o las etiquetas que se quieren obtener (llamada *Label*). Este modelo se entrenará utilizando imágenes de este tipo para intentar obtener el resultado segmentado.

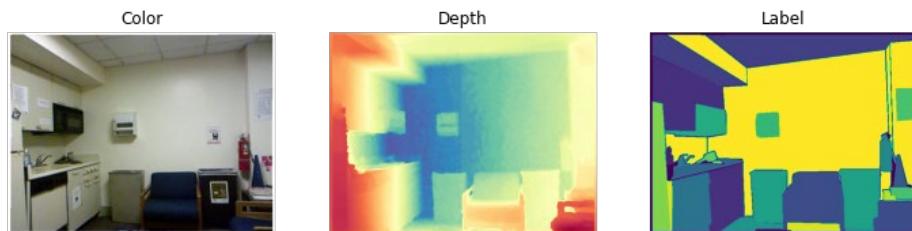


Figura 32: Ejemplo de la imagen de NYUv2.



Figura 33: Ejemplo de la imagen de SUNRGBD (531x681).

Para un buen entrenamiento de estos modelos se requiere de un computador bueno con una tarjeta gráfica potente. Al no disponer de ello es necesario realizar la reducción de dimensiones de la imagen. En este caso para poder entrenar el modelo se ha tenido que reducir a 96 x 128. Cuando se reducen las dimensiones de una imagen, también se reduce su calidad. Esto puede que empeore el funcionamiento del modelo, porque se han modificado

las etiquetas del *dataset*, pero a su vez al reducirle la calidad quiere decir que el modelo tiene que saber generalizar los objetos de las imágenes. En la Figura 34 se pueden ver las imágenes modificadas.

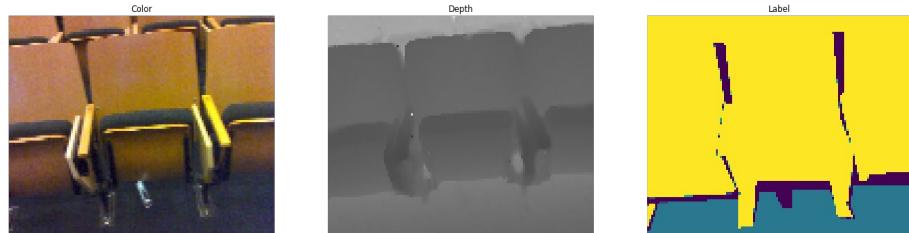


Figura 34: Ejemplo de la imagen de SUNRGBD (96x128).



Figura 35: Ejemplo de la imagen de SUNRGBD con 4 categorías (96x128).

Una vez aplicada la reducción de dimensiones, no es posible que el modelo entrene con esa calidad por lo que se reducen las clases a 4 categorías, suelo, pared, persona y resto, ver Figura 35. Muestra de ello son las gráficas del modelo 1, que no mejora de una exactitud del 60% y por se procede a entrenar el modelo con menos categorías. Aun y todo, es necesario aplicar *data augmentation* para reducir el *overfitting* durante el entrenamiento y tener más imágenes para ello. Los parámetros utilizados para el *data augmentation* son los siguientes:

- Para que la imagen se gire, *rotation_range*=15.
- Para podar o quitarle las esquinas, *shear_range*=0.2.
- Para acercar la imagen, *zoom_range*=0.2.
- Para desplazar la imagen en horizontal, *width_shift_range*=0.2.
- Para desplazar la imagen en vertical, *height_shift_range*=0.2.
- *fill_mode*="nearest"

5.4.2 Modelo 1

La primera opción para entrenar el modelo ha sido realizar el modelo completo desde cero, modificando los elementos que puedan funcionar de una manera más eficiente. Esto se realiza para poder compararlo con los modelos de los estudios mencionados anteriormente, por si se produce alguna pequeña mejora.

Al ser un modelo que requiere de mucho entrenamiento y delicadeza para entrenarlo, no se ha podido hacer un correcto entrenamiento y los resultados obtenidos dejan mucho que desear.

Para entrenar este modelo, los parámetros elegidos son los que se indican abajo. Una vez entrenado el modelo podemos ver si el entrenamiento es el correcto o deseado visualizando las gráficas obtenidas.

1- Prueba con los datos:

- Optimizador: Adam con 0.0001 de *learning rate* que cada 5 *epoch* se va reduciendo.
- Pérdida: Se mide con la perdida *dice_loss*, y se puede ver que una vez llegado a un punto no mejora el entrenamiento.
- Métrica de evaluación: En las Figuras 36 y 37 se muestran las gráficas obtenidas durante el entrenamiento del modelo. Como se puede ver en la Figura 36 que la exactitud no supera ni el 50% de exactitud.

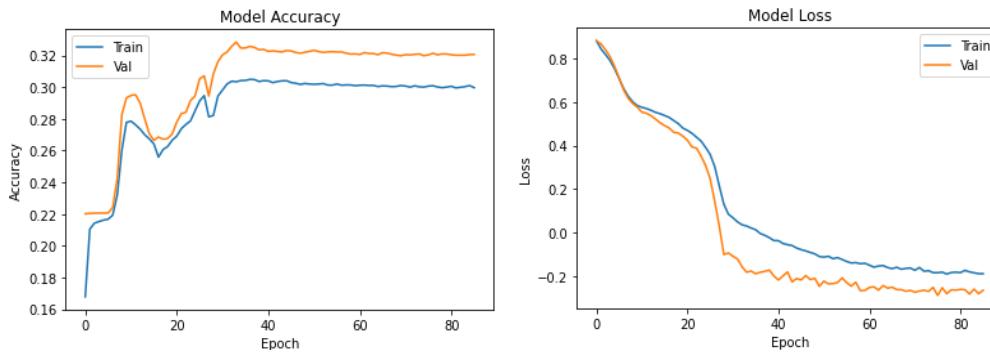


Figura 36: exactitud y pérdida durante el entrenamiento del modelo 1 con SGD.

Tras analizar que los resultados obtenidos no son los deseados, se procede a realizar el entrenamiento con *transfer learning* para que el modelo se entrene en base a los pesos de un modelo previamente entrenado con la arquitectura de resnet-34.

2- Prueba con los datos:

- Optimizador: Adam con 0.00001 de *learning rate* que cada 10 *epoch* se va reduciendo.
- Pérdida: Se mide con la perdida *dice_loss*, y se puede ver que una vez llegado a un punto no mejora el entrenamiento.

- c. Métrica de evaluación: En las Figuras 37 y 38 se muestran las gráficas obtenidas durante el entrenamiento del modelo. Como se puede ver en la imagen el modelo tiene un sobreajuste.

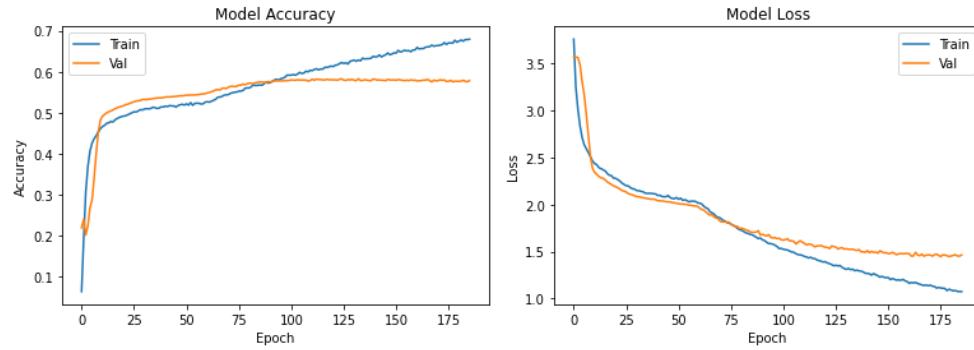


Figura 37: Exactitud y pérdida durante el entrenamiento del modelo 1 con *transfer learning*.

5.4.3 Modelo 2

Se ha visto que los resultados obtenidos del primer modelo no eran los esperados, por eso se procede a buscar una opción más fácil o simple de entrenar, cambiando las etiquetas y simplificándolas a cuatro categorías. De esta manera el entrenamiento del modelo va a ser de una mejor calidad.

Esta segunda opción también se ha probado a utilizar un modelo con los pesos de Resnet-34 utilizados para la base de datos de ImageNet, que facilitan mucho el entrenamiento ya que están entrenados y probados, además, los pesos están bastante ajustados para que funcione de una manera correcta.

Para ello se utiliza un modelo U-Net ya disponible con los pesos indicados anteriormente y se adapta para poder utilizarlo con las imágenes de color y de profundidad. Con esto, se reducen los parámetros a entrenar y es posible entrenar el modelo en un tiempo menor.

Para entrenar este modelo, los parámetros elegidos son los que se indican abajo. Una vez entrenado el modelo podemos ver si el entrenamiento es el correcto o deseado visualizando las gráficas obtenidas.

1- Prueba con los datos:

- a. Optimizador: Adam con 0.00001 de *learning rate*, que se reduce cada 5 *epoch* a la mitad de su valor.

- b. Pérdida: Se mide con la perdida *Categorical_crossentropy*, y se puede ver que una vez llegado a un punto no mejora el entrenamiento
- c. Métrica de evaluación: se utilizan diferentes métricas de evaluación (*dice_coef*, exactitud, *true_positive_rate* e *iou_score*), pero se utilizan graficas de perdida y exactitud para analizarlas visualmente.

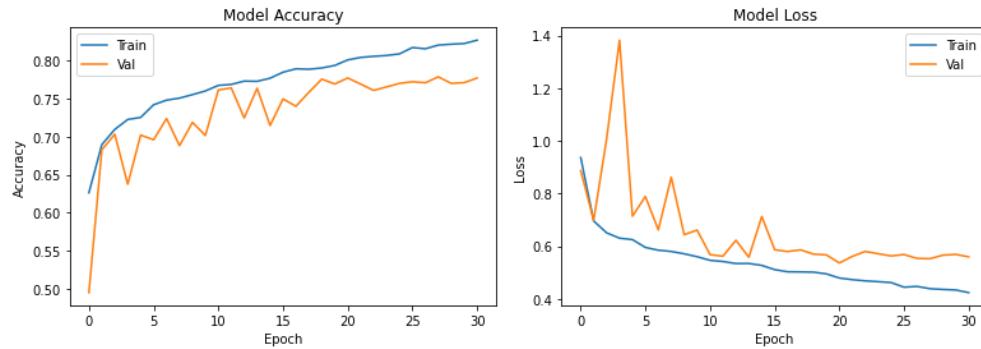


Figura 38: Exactitud y pérdida durante el entrenamiento del modelo 2 con Adam.

2- Prueba *transfer learning* con los datos:

- a. Optimizador: Adam con 0.00001 de *learning rate*.
- b. Pérdida: Se mide con la perdida *bce_jaccard_loss*.
- c. Métrica de evaluación: se utilizan diferentes métricas de evaluación (*dice_coef*, exactitud, *true_positive_rate* e *iou_score*), pero se utilizan graficas de perdida y exactitud para analizarlas visualmente.

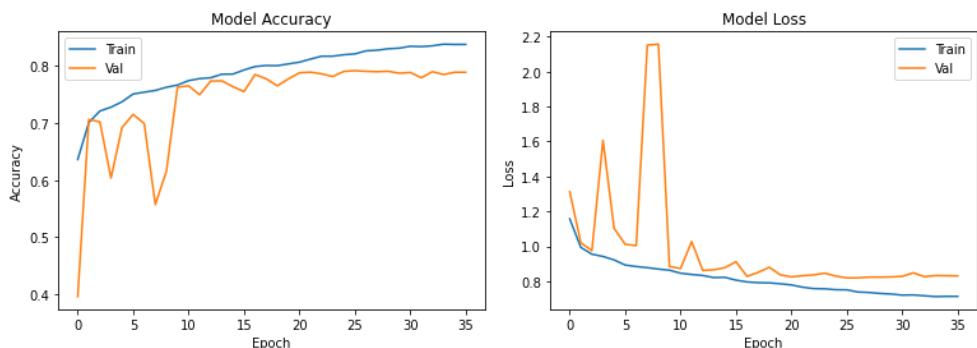


Figura 39: Exactitud y pérdida durante el entrenamiento del modelo 2 con *transfer learning*.

Se puede ver que el modelo se ha entrenado mucho mejor que la primera y se pueden ver mejoras en el entrenamiento.

5.5 Adaptación del modelo de segmentación para predecir escenarios futuros

El objetivo de este proyecto no es precisamente realizar solo un modelo de segmentación, sino que también es predecir las segmentaciones posteriores a la actual para la predicción de movimiento, de esta manera el robot puede realizar un movimiento más inteligente y preciso. Por ello, en este apartado se adapta el modelo creado para la segmentación y se realizan los cambios necesarios para realizar la predicción de los futuros escenarios.

En estos modelos se realizarán diferentes formas de predicción y se elegirá la que mejores resultados obtenga. La primera opción es utilizar la convolución LSTM (*Long Short Term Memory*) en las últimas capas para que el segmentados semántico pueda realizar una predicción parecida a la que se realizaba en el modelo de segmentación sin predicción. En cambio, a la segunda opción se le añaden otras capas LSTM después de las fusiones del modelo, en el decodificador. Estas convoluciones utilizan el sistema LSTM para poder predecir una imagen futura.

5.6 Entrenamiento de la red neuronal predictor de trayectoria

Para el entrenamiento de esta red neuronal es fundamental escoger la base de datos para y prepararla para que funcione correctamente. Para realizar el proceso de este análisis y entrenamiento en el último modelo, es necesario tener la base de datos con los movimientos preparado.

5.6.1 Datasets

En este caso en concreto, se escogió la base de datos KTP-dataset (Munaro & Menegatti, 2014), que contiene fotogramas etiquetadas en *rosbag*, los paquetes de ROS que sirven para guardar o grabar los datos adquiridos para un procesado futuro. Esta base de datos en concreto guarda las imágenes RGB-D para su procesamiento de predicción de trayectoria de personas, pero al no estar segmentadas y utilizar un sistema de reconocimiento de objetos, se ha decidido dejar de lado esta base de datos. En cambio, como la base de datos NYUv2 tiene las imágenes en formato video, es la que se utilizará para el entrenamiento.

Antes de nada, hay que preparar todas las imágenes y segmentarlas correctamente según lo entrenado para que nuestro modelo tenga las mismas características que el modelo entrenado anteriormente.

En esta base de datos se han guardado las imágenes por lugares, que en cada lugar (cocina, baño, salón, comedor...) existen escenarios diferentes donde se han cogido datos moviendo el robot. En cada escenario se ha realizado la segmentación tal y como se comenta anteriormente para así tener las imágenes segmentadas para que las etiquetas no cambien.

Una vez de haber realizado las segmentaciones de las imágenes, se ha procedido a preparar los datos para poder entrenar el LSTM. En este caso, al igual que en el anterior se tienen que reducir los tamaños de las imágenes, pero no se introduce solo una imagen como dato, sino que se realiza un grupo para que el LSTM se pueda entrenar y con esto podamos predecir los escenarios futuros.

En este caso en concreto, se han cogido 16 imágenes para cada grupo y se han formado los grupos para añadir como array en la base de datos. De esta manera cuando le pasemos los datos al modelo para entrenar sabe que son secuencias que debe de entrenar.

En las Figuras 41, 42 y 43 se puede ver una muestra de las imágenes agrupadas para la predicción. Estas imágenes se utilizan como secuencias para entrenar el modelo de predicción LSTM.

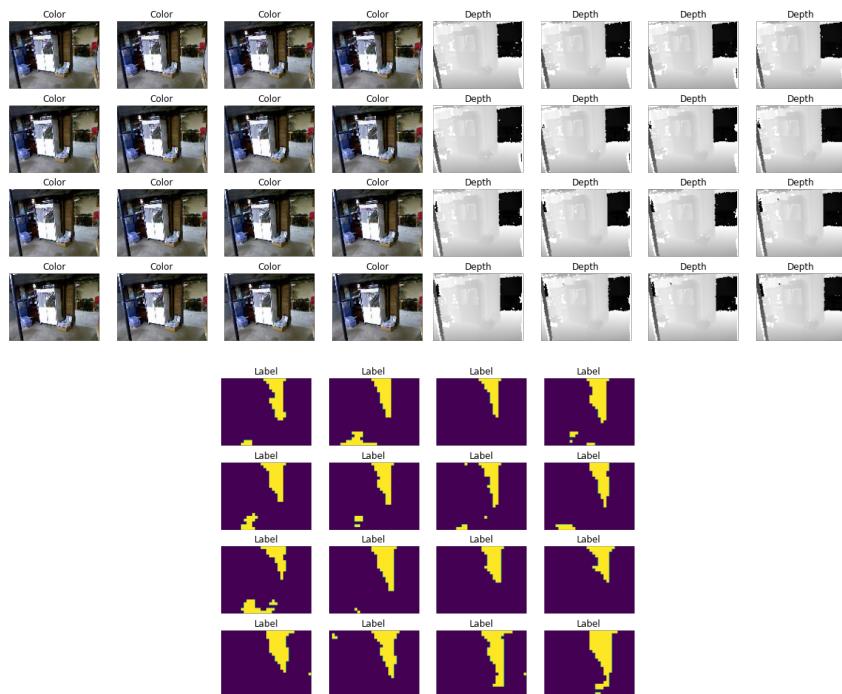


Figura 40: Imágenes de entrada para entrenamiento de datos para predicción 1.

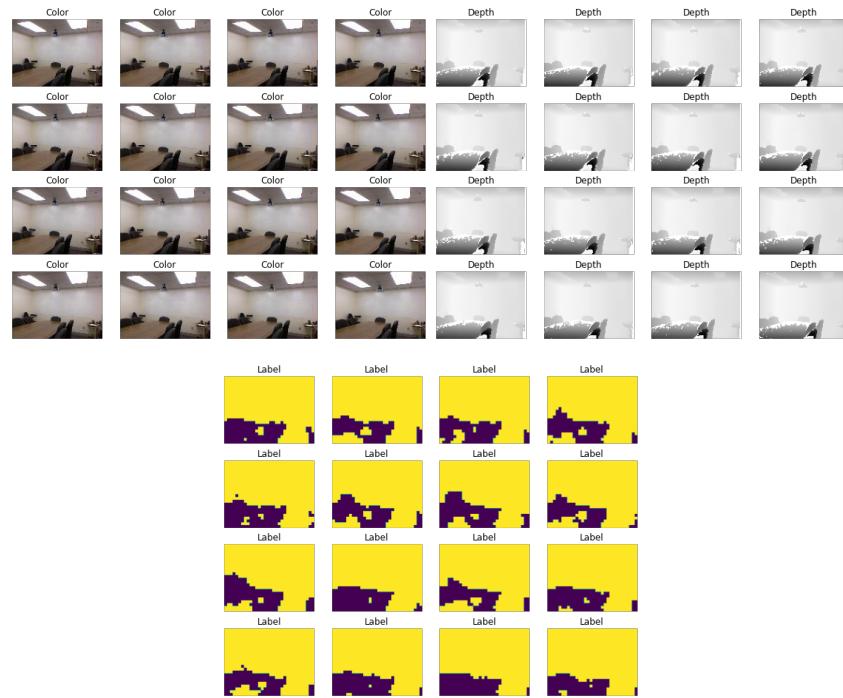


Figura 41: Imágenes de entrada para entrenamiento de datos para predicción 2.

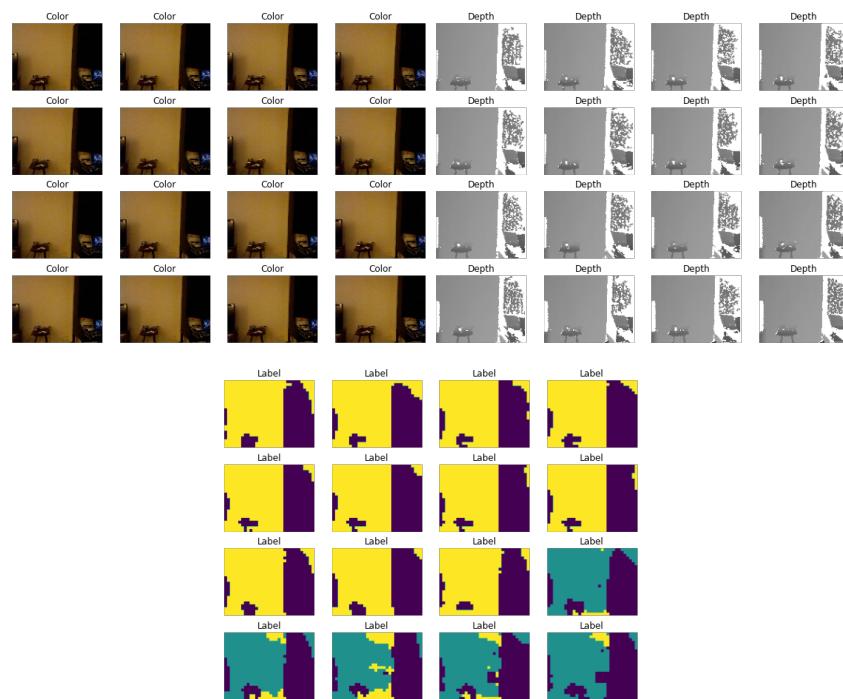


Figura 42: Imágenes de entrada para entrenamiento de datos para predicción 3.

Con estos datos se entrenan dos modelos y se diferencian las predicciones realizadas por las mismas.

5.4.2 Modelo LSTM 1

El modelo escogido ha sido el modelo que ha dado los mejores resultados. Ha este modelo se le han cambiado las últimas tres capas de convolución por otra de convolución LSTM.

Para entrenar este modelo, los parámetros elegidos son los que se indican abajo. Una vez entrenado el modelo podemos ver si el entrenamiento es el correcto o deseado visualizando las gráficas obtenidas.

Prueba con los datos:

- Optimizador: Adam con 0.0001 de *learning rate*, reduciendo a la mitad cada 3 *epoch*.
- Pérdida: Se mide con la perdida *binary_crossentropy*, ya que *categorical_crossentropy* no funciona en este tipo de entrenamiento.
- Métrica de evaluación: se utilizan graficas de perdida y exactitud para analizarlas visualmente

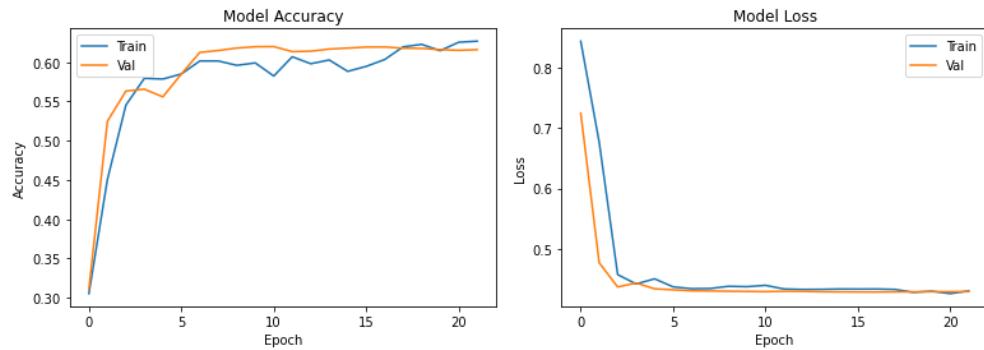


Figura 43: Exactitud y pérdida durante el entrenamiento del modelo de predicción 1.

5.4.2 Modelo LSTM 2

El modelo escogido ha sido el modelo que ha dado los mejores resultados. Ha este modelo se le han cambiado las últimas capas de convolución de cada sección del modelo UNet por una LSTM ver Figura 46 para entender cómo se estructura el modelo. Aparte de cambiar las ultimas capas convolucionales y las capas de salida de fusión, se añaden capas LSTM en cada ampliación del decodificador.

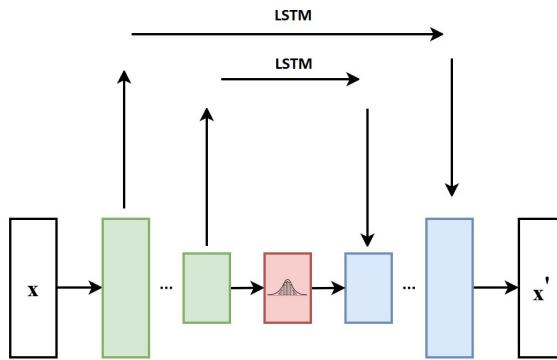


Figura 44: Imagen del Modelo simplificado para entender el funcionamiento.

Para entrenar este modelo, los parámetros elegidos son los que se indican abajo. Una vez entrenado el modelo podemos ver si el entrenamiento es el correcto o deseado visualizando las gráficas obtenidas.

Prueba con los datos:

- a. Optimizador: Adam con 0.0001 de *learning rate*, reduciendo a la mitad cada 3 *epoch*
- a. Pérdida: Se mide con la perdida *binary_crossentropy*, ya que *categorical_crossentropy* no funciona en este tipo de entrenamiento.
- b. Métrica de evaluación: se utilizan graficas de perdida y exactitud para analizarlas visualmente.

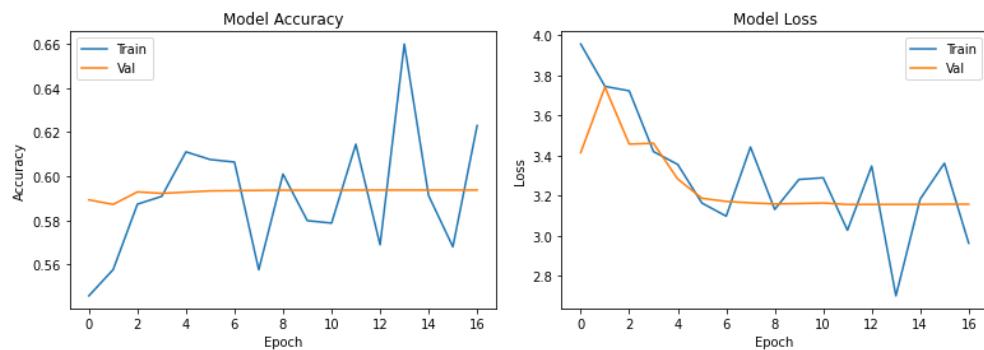


Figura 45: Exactitud y pérdida durante el entrenamiento del modelo de predicción 3.

5.7 Preparando el modelo para Jetson Nano

Este modelo se ha pensado utilizar en una Jetson Nano para poder realizar una predicción del estado para una planificación posterior. De esta manera el robot puede predecir que le puede pasar si continua con los movimientos anteriores.

La cuestión es que la tarjeta de Nvidia tiene tarjeta gráfica, pero para una utilización correcta se tiene que modificar el modelo para convertirlo en modelo TensorRt.

Una vez probado se han realizado las pruebas que se pueden ver en el siguiente apartado.

Para que el modelo sea válido en cualquier tipo de robot, se ha creado un nodo en ROS, de esta manera la imagen puede ser manipulada por la librería OpenCV que a la vez puede implementar el modelo en tiempo real.

El nodo que se ha utilizado para la transformación de imagen se llama cv_bridge que como se ha comentado en el apartado anterior es el nodo utilizado para que ROS convierta las imágenes obtenidas por la cámara Intel 435i en una imagen capaz de manipularlo para que las imágenes que se pasen a la entrada de nuestro modelo sean de las características que se han especificado al crear el mismo.

Una vez creado el sistema se realizan las pruebas para verificar su funcionamiento.

6. Evaluación

Al crear diferentes modelos, unos de segmentación y otros de segmentación con predicción, se procede a evaluar esos modelos creados. Para poder evaluarlos, se analizan los resultados obtenidos como las métricas calculadas. Se analizarán estos modelos, el que obtiene los mejores resultados de segmentación semántica y los dos modelos de predicción de escenarios futuros. Al final de la evaluación, se realiza el análisis de cada modelo visualizando los resultados para verificar su funcionamiento de la manera más simple posible.

Por una parte, se analizan los resultados de las evaluaciones realizadas del modelo de segmentación semántica elegida y por otra, los modelos LSTM. Con las métricas obtenidas se puede saber si la función que realiza el modelo es el correcto.

6.1 Métricas obtenidas de los modelos

Para el modelo de segmentación semántica, sin la predicción, se han analizado diferentes métricas obtenidas para medir cuan efectivo es el modelo entrenado. Esas métricas son la exactitud, *IoU* y *True positive rate*. La métrica *IoU*, la que mide el grado de similitud de dos conjuntos, es muy significativo para este tipo de modelos. El valor de *True positive rate* indica cuantos aciertos tiene el modelo. Por último, la exactitud es la que mide la cercanía que tienen las predicciones respecto a la referencia.

Tabla 8: Métricas del modelo de segmentación semántica sin predicción.

Métrica	Resultado
Exactitud	0,77271235
<i>IoU</i>	0,34801486
<i>True positive</i>	0,76918536

Los resultados de la Tabla 8 indican que el modelo entrenado es lo suficientemente bueno para realizar la segmentación semántica, por un lado, porque tienen una exactitud suficientemente alta como para utilizarla y por otro, la métrica *IoU* llega casi al 35%.

Este modelo es utilizado para crear la base de datos para el modelo de predicción, es decir, el modelo de predicción de escenarios futuros se ha entrenado con las etiquetas creadas por este modelo.

Se puede decir que la base de datos para entrenar el algoritmo de predicción de escenarios futuros no es robusta y ni está perfectamente etiquetada, por lo que los resultados obtenidos se analizan acorde al *dataset* de entrenamiento. Con ello se puede analizar si los modelos creados pueden funcionar en caso de disponerse un *dataset* correctamente creado.

Para los dos modelos de segmentación con predicción, se han obtenido las siguientes métricas:

Tabla 9: Métricas del modelo 1 de segmentación semántica con predicción.

Métrica	Resultado
Exactitud	0,6341
Precisión	0,8713
<i>Recall</i>	0,2734

Tabla 10: Métricas del modelo 3 de segmentación semántica con predicción.

Métrica	Resultado
Exactitud	0,4324
Precisión	0,82
<i>Recall</i>	0,061

Como se puede ver en las tablas de evaluación, estas predicciones funcionan hasta un cierto punto de precision, pero puede que teniendo un dataset preparado expresamente para este proceso los resultados mejoren. Se muestra uno de los resultados obtenidos con la predicción del primer modelo. El segundo modelo, ha sido sobreentrenado y los resultados que obtenemos no representan la imagen de entrada, son aleatorias. Esto se debe a que el modelo solo sabe predecir los estados del entrenamiento.

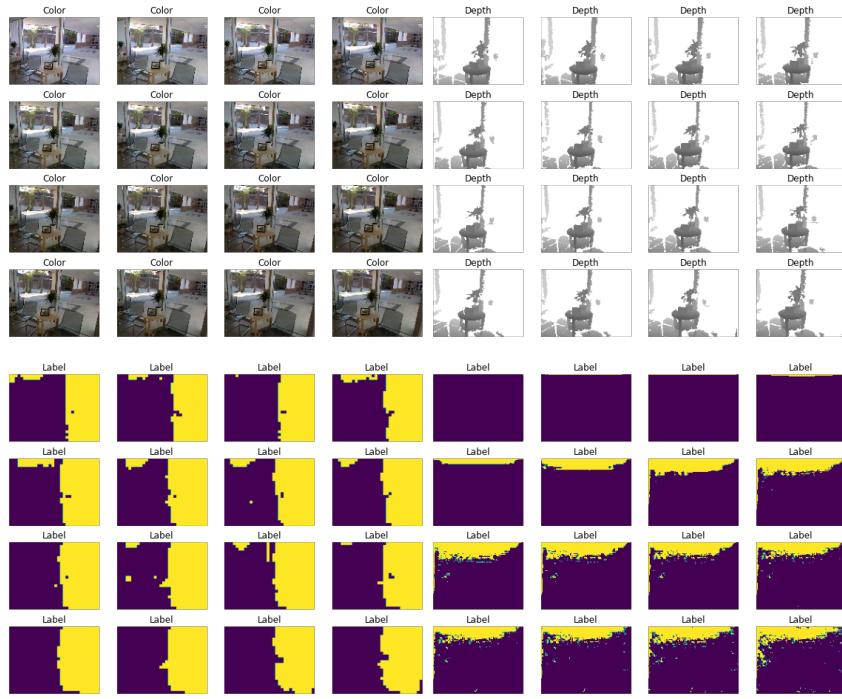


Figura 46: Resultado del modelo 1 con LSTM

6.2 Evaluación de diferentes escenarios

A parte del análisis de las imágenes de prueba, se analizan los resultados de entornos que no se incluyen en la base de datos. Se analizan las imágenes segmentadas por el modelo para verificar si cumple con las especificaciones definidas al empezar con el trabajo. El caso de la navegación del robot no entra en el apartado del algoritmo, por lo tanto, no va a ser evaluado. En cambio, se analizan los objetivos especificados en diferentes escenarios para ver si los modelos responden de una manera correcta. Los objetivos que se analizan son la detección de personas, detección del camino libre para navegar y predicción de estados futuros.

La detección del camino libre y la detección de personas se evaluarán con el modelo de segmentación semántica sin predicción. En cambio, la predicción de un estado futuro se analizará con el modelo preparado con convoluciones LSTM.

En las primeras tres escenas, A, B y C, se muestran imágenes de entornos diferentes donde se analiza el comportamiento del segmentador semántico. Por otra parte, en las escenas D y E se muestran diferentes capturas de video donde el segmentador semántico tiene que predecir cual es el escenario futuro para cada situación.

En la escena A se puede ver una persona y a su alrededor no se puede ver ningún espacio libre. El modelo de segmentación semántica es capaz de detectar la persona, pero en cambio

a su alrededor detecta espacio libre para la navegación. En la Figura 47 se puede ver la detección de la persona el amarillo y el camino libre en rojo. En estos casos abría que mejorar la precisión, ya que el algoritmo intuye que alrededor de la persona va a haber sitio para navegar.

Escena A:



Figura 47: Imágenes y resultado de la escena A.

En la escena B se puede ver una imagen de la entrada de una casa, realizando la captura por el interior, donde hay un perchero y paraguero cerca de la pared. En general el modelo realiza una segmentación semántica adecuada, ya que todo el suelo es detectado como espacio de navegación y los objetos detectados como suelo se encuentran aparte, es decir, en la mitad de las dos detecciones se encuentra lo segmentado como pared, por lo que se puede decir que en la Figura 48 la segmentación semántica realizada por el modelo es adecuada y cumple los requisitos previamente especificados.

Escena B:

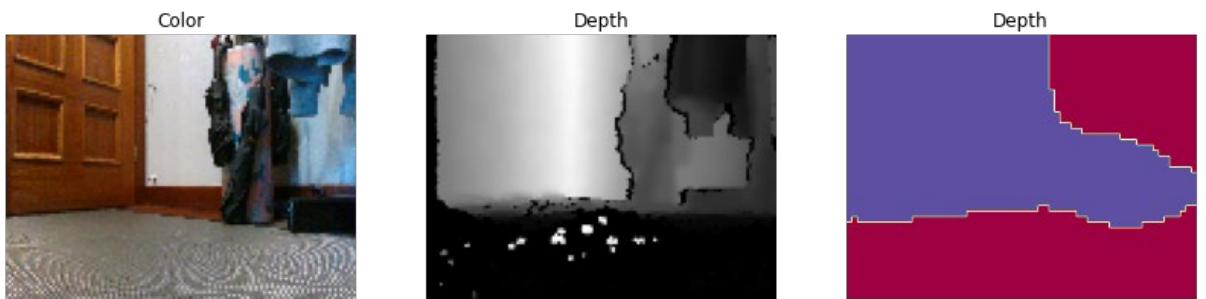


Figura 48: Imágenes y resultado de la escena B.

En la escena C se muestra una imagen similar a la escena B, realizando la captura por el interior, donde el ventilador se encuentra junto a la puerta. En general el modelo realiza una segmentación semántica adecuada, a falta de una clasificación adecuada para el ventilador y la parte de la pared de la derecha. La segmentación del modelo no es del todo adecuada,

pero la parte principal de navegación esta segmentada correctamente. En la Figura 49 se muestra la escena C.

Escena C:

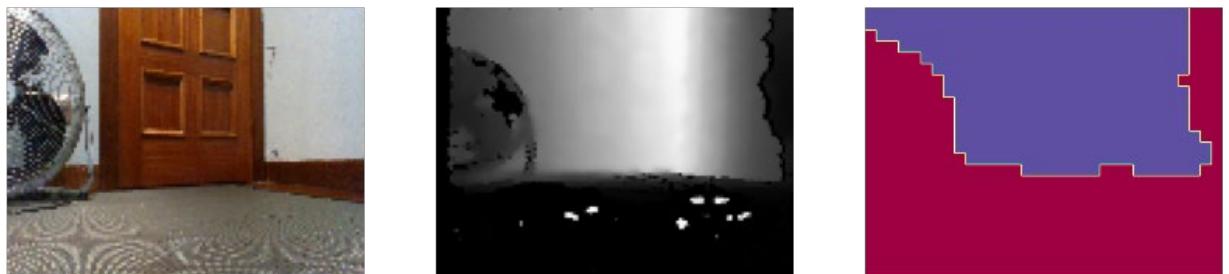


Figura 49: Imágenes y resultado de la escena C.

En la escena D, se muestran las capturas de video cuando el robot está en movimiento. En estas capturas se puede ver el robot moviéndose por una habitación. La predicción de este apartado el robot predice el suelo como amarillo y la pared de morado, por lo que no tiene nada de movilidad para la navegación. En cambio, al final de los movimientos del robot, se puede ver que existe el suelo para que pueda navegar. Se pueden ver las imágenes en la Figura 50.

Escena D:

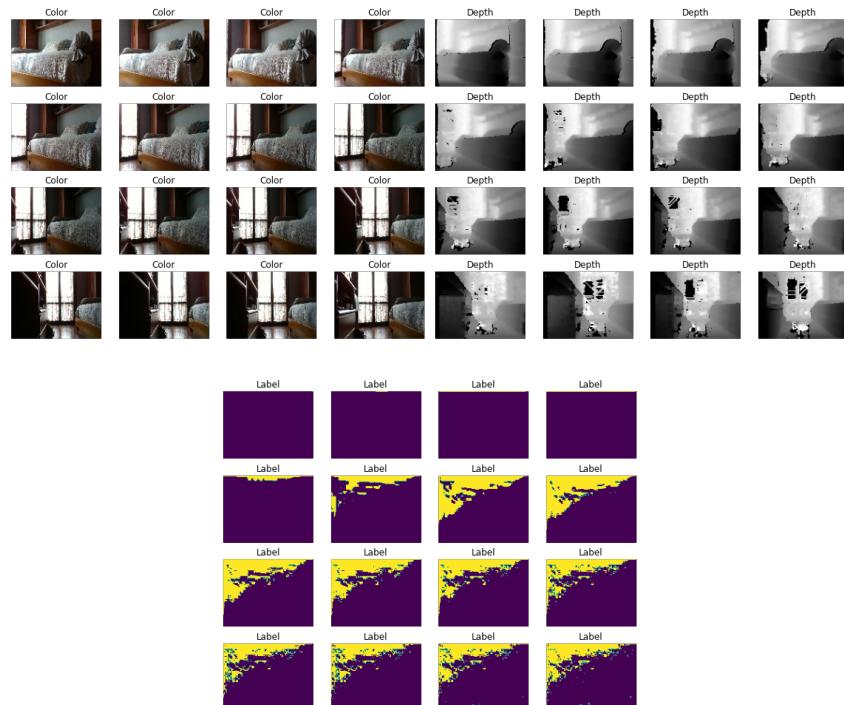


Figura 50: Imágenes y resultado de la escena D.

En la escena E, también se muestran las capturas de video cuando el robot está en movimiento. En estas capturas se puede ver el robot moviéndose por un pasillo. En la predicción de este apartado el robot predice el suelo como amarillo y la pared de morado, por lo que el robot tiene cada vez más posibilidad de navegar por el entorno. En este caso, la predicción no es exacta, pero la predicción mejora en cuanto al lugar de navegación. La Figura 51 muestra la Escena E.

Escena E:

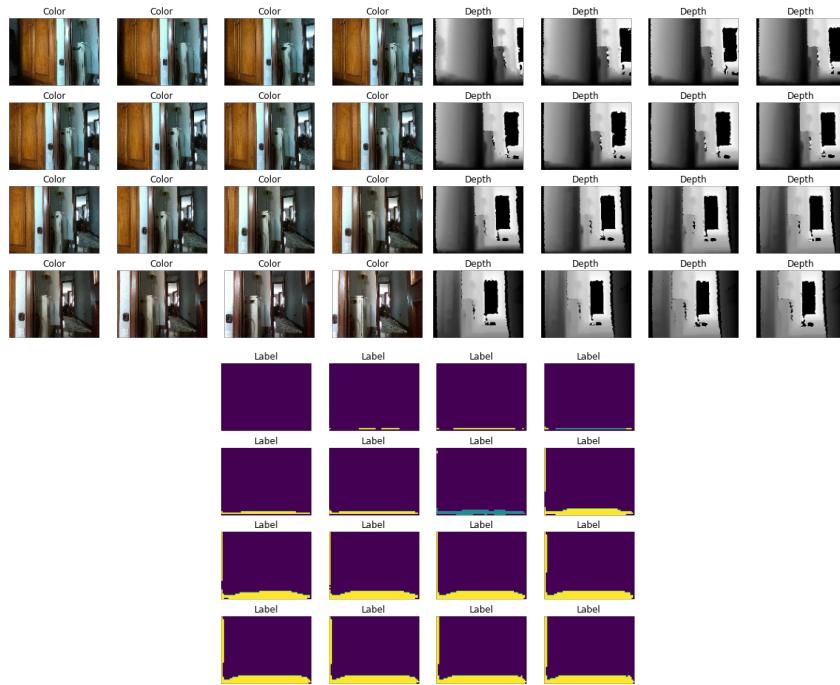


Figura 51: Imágenes y resultados de la escena E.

Los modelos creados predicen de una manera correcta, a falta de mejorar precisión. La segmentación semántica realiza la predicción con algunos defectos, lo que puede ser por la calidad de las imágenes, que a su vez influye en los modelos de predicción de escenarios futuros. El modelo de predicción de escenarios futuros obtiene los resultados justos para poder utilizarlo, pero no previene al robot de las visiones a futuro para una navegación más eficiente. Para el entrenamiento y la verificación del modelo de predicción de estados futuros se necesita un *dataset* preparado expresamente para ello.

7. Conclusiones y trabajo futuro

7.1. Conclusiones

La primera impresión del modelo es buena, pero debido a los tiempos del TFM se necesita un análisis y ejecución en profundidad para abordar el problema y obtener mejores resultados. El desarrollo de esta herramienta creada a partir de los datos preparados con el modelo de segmentación semántica es capaz de predecir los escenarios futuros, identificando los elementos a groso modo y segmentándolos según a la clase a la que corresponde en la mayoría de los casos. En general, las situaciones de evaluación tienen que mejorar, pero hay zonas en las que el modelo realiza una segmentación correcta.

El modelo creado se ha entrenado mediante aprendizaje profundo, que realiza la clasificación y segmentación de los elementos de las imágenes. Esta segmentación contiene 4 clases que puede diferenciar el modelo para que algún algoritmo de planificación realice la mejor ruta partiendo de los escenarios predichos. El rendimiento de este algoritmo es mejorable ya que ha obtenido los resultados de una exactitud poco superior al 60% y las predicciones podrían ser más precisas.

El modelo se ha implementado para utilizarlo en un robot que funcione con Robot Operating System con una cámara de Intel RealSense. Esto puede ser factible ya que este software es de código abierto y se puede implementar en cualquier robot móvil, ya sea de algún fabricante en concreto o de propia creación.

Este algoritmo se ha comprobado en el robot Jetbot con la cámara de Intel RealSense para verificar si las métricas durante el entrenamiento son suficientemente buenas para implementarlas. En las pruebas realizadas se puede ver que el modelo necesita una puesta a punto, por lo que se puede decir que el algoritmo tiene que mejorar para poder usarse como predicción de escenarios futuros en un robot real.

La velocidad del algoritmo para procesar las imágenes es de 6 FPS. Se puede decir que este procesamiento no es suficiente para que un robot autónomo sea capaz de realizar los movimientos óptimos de una manera eficaz, pero puede llegar a ser una alternativa para futuros estudios.

Si se pudiera entrenar con una tarjeta gráfica más potente o alguna plataforma online con TPUs, se podría intentar realizar el modelo con unas imágenes de mejor resolución, con lo que se supone que el modelo tendría una mejora importante. En cambio, la velocidad de

procesamiento no mejoraría, pero se podría modificar la forma de ejecución, ya que todas las placas de Jetson se pueden utilizar con un kit de desarrollo de software (SDK, siglas en inglés) llamado DeepStream, que mejora la velocidad de procesamiento de la tarjeta que hemos utilizado debido a que toda la computación se utiliza para ello.

7.2. Líneas de trabajo futuro

La identificación de los puntos a mejorar de este trabajo, ayudan a la hora de establecer las líneas de trabajo futuras. Una vez que el estudio se ejecuta, se puede ver y analizar lo realizado para una mejora o sacar conclusiones.

Lo primero de todo, se ha comentado anteriormente que la GPU utilizada para el entrenamiento no es suficiente para procesar los tensores al entrenar el modelo, por lo que se tendría que entrenar el modelo con GPU o TPU mejores o alquilados.

Para la predicción de movimiento o escenarios futuros, sería conveniente realizar un *dataset* nuevo. No existe ningún *dataset* que tenga los datos tipo video segmentados para realizar el entrenamiento.

En el mismo sentido, se ha visto interesante realizar la predicción de movimientos de las personas. Para esto, como en la predicción de los escenarios, se necesitaría crear un *dataset* donde se detecten los movimiento y gestos de las personas para un procesamiento de su acción.

Tal y como se ha visto en el estado del arte, se pueden ver ejemplos de segmentación en entornos exteriores, pero no existen modelos de predicción de escenarios futuros. Sería interesante utilizar la idea para los entornos exteriores y realizar la predicción de los elementos que se puedan mover y según los datos intentar predecir lo que pueda suceder, aunque la posibilidad de escenario no tiene por qué ser única.

Por último, sería interesante que el procesador del robot fuera más potente para realizar las predicciones de una manera más rápida y así, mejorar el rendimiento del modelo para una ejecución instantánea.

8. Bibliografía

- Ai, Y., Rui, T., Lu, M., Fu, L., Liu, S., & Wang, S. (2020). DDL-SLAM: A robust RGB-d SLAM in dynamic environments combined with deep learning. *IEEE Access*, 8, 162335–162342. <https://doi.org/10.1109/ACCESS.2020.2991441>
- Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. *ArXiv*. <http://arxiv.org/abs/2004.10934>
- Bruckschen, L., Bungert, K., Dengler, N., & Bennewitz, M. (2020). Human-aware robot navigation by long-term movement prediction. In *IEEE International Conference on Intelligent Robots and Systems*. <https://doi.org/10.1109/IROS45743.2020.9340776>
- Cao, C., Wang, B., Zhang, W., Zeng, X., Yan, X., Feng, Z., Liu, Y., & Wu, Z. (2019). An Improved Faster R-CNN for Small Object Detection. *IEEE Access*, 7, 106838–106846. <https://doi.org/10.1109/ACCESS.2019.2932731>
- Chan, T.-H., Jia, K., Gao, S., Lu, J., Zeng, Z., & Ma, Y. (2014). PCANet: A Simple Deep Learning Baseline for Image Classification? *IEEE Transactions on Image Processing*, 24(12), 5017–5032. <https://doi.org/10.1109/TIP.2015.2475625>
- Chang, L., Niu, X., & Liu, T. (2020). Gnss/imu/odo/lidar-slam integrated navigation system using imu/odo pre-integration. *Sensors (Switzerland)*, 20(17), 1–18. <https://doi.org/10.3390/s20174702>
- Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., & Schiele, B. (2016). The Cityscapes Dataset for Semantic Urban Scene Understanding. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (Vols. 2016-Decem). <https://doi.org/10.1109/CVPR.2016.350>
- Cui, X., Lu, C., & Wang, J. (2020). 3D Semantic Map Construction Using Improved ORB-SLAM2 for Mobile Robot in Edge Computing Environment. *IEEE Access*, 8, 67179–67191. <https://doi.org/10.1109/ACCESS.2020.2983488>
- Deng, W., Huang, K., Chen, X., Zhou, Z., Shi, C., Guo, R., & Zhang, H. (2020). Semantic RGB-D SLAM for Rescue Robot Navigation. *IEEE Access*, 8, 221320–221329. <https://doi.org/10.1109/ACCESS.2020.3031867>
- Feng, D., Haase-Schutz, C., Rosenbaum, L., Hertlein, H., Glaser, C., Timm, F., Wiesbeck, W.,

- & Dietmayer, K. (2021). Deep Multi-Modal Object Detection and Semantic Segmentation for Autonomous Driving: Datasets, Methods, and Challenges. *IEEE Transactions on Intelligent Transportation Systems*, 22(3), 1341–1360. <https://doi.org/10.1109/TITS.2020.2972974>
- Gross, H.-M., Scheidig, A., Muller, S., Schutz, B., Fricke, C., & Meyer, S. (2019). Living with a Mobile Companion Robot in your Own Apartment - Final Implementation and Results of a 20-Weeks Field Study with 20 Seniors*. *2019 International Conference on Robotics and Automation (ICRA), 2019-May*, 2253–2259. <https://doi.org/10.1109/ICRA.2019.8793693>
- Gross, H. M., Meyer, S., Scheidig, A., Eisenbach, M., Mueller, S., Trinh, T. Q., Wengfeld, T., Bley, A., Martin, C., & Fricke, C. (2017). Mobile robot companion for walking training of stroke patients in clinical post-stroke rehabilitation. *Proceedings - IEEE International Conference on Robotics and Automation*, 1028–1035. <https://doi.org/10.1109/ICRA.2017.7989124>
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016-December*, 770–778. <https://doi.org/10.1109/CVPR.2016.90>
- Illustration of challenges in semantic segmentation. (a), Input Image.... | Download Scientific Diagram.* (n.d.). Retrieved June 3, 2021, from https://www.researchgate.net/figure/Illustration-of-challenges-in-semantic-segmentation-a-Input-Image-b-Ground-Truth_fig1_332186435
- Incorrect visualization of LiDAR · Issue #32 · carla-simulator/ros-bridge.* (n.d.). Retrieved June 3, 2021, from <https://github.com/carla-simulator/ros-bridge/issues/32>
- Intel. (2019). *Depth Camera D435 – Intel® RealSense™ Depth and Tracking Cameras.* <Https://Www.Intelrealsense.Com/Depth-Camera-D435/>.
- <https://www.intelrealsense.com/depth-camera-d435/>
<https://www.intelrealsense.com/depth-camera-d435/>
<https://www.intelrealsense.com/depth-camera-d435/>
<https://www.intelrealsense.com/depth-camera-d435/>
- Jin, S., Chen, L., Sun, R., & McLoone, S. (2020). A novel vSLAM framework with unsupervised semantic segmentation based on adversarial transfer learning. *Applied Soft Computing Journal*, 90, 106153. <https://doi.org/10.1016/j.asoc.2020.106153>
- Jiwon, J. (2019). *Computer Vision for Beginners: Part 4 - Towards Data Science.*

<https://towardsdatascience.com/computer-vision-for-beginners-part-4-64a8d9856208>

Krombach, N., Droschel, D., Houben, S., & Behnke, S. (2018). Feature-based visual odometry prior for real-time semi-dense stereo SLAM. *Robotics and Autonomous Systems*, 109, 38–58. <https://doi.org/10.1016/j.robot.2018.08.002>

Lewandowski, B., Wengfeld, T., Muller, S., Jenny, M., Glende, S., Schroter, C., Bley, A., & Gross, H.-M. (2020). Socially Compliant Human-Robot Interaction for Autonomous Scanning Tasks in Supermarket Environments. *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 363–370. <https://doi.org/10.1109/RO-MAN47096.2020.9223568>

Li, F., Li, W., Chen, W., Xu, W., Huang, L., Li, D., Cai, S., Yang, M., Xiong, X., & Liu, Y. (2020). A Mobile Robot Visual SLAM System with Enhanced Semantics Segmentation. *IEEE Access*, 8, 25442–25458. <https://doi.org/10.1109/ACCESS.2020.2970238>

Liu, W., Chan, A. B., Lau, R. W. H., & Manochaeee, D. (2015). Leveraging long-term predictions and online learning in agent-based multiple person tracking. In *IEEE Transactions on Circuits and Systems for Video Technology* (Vol. 25, Issue 3). <https://doi.org/10.1109/TCSVT.2014.2344511>

Miyamoto, R., Adachi, M., Ishida, H., Watanabe, T., Matsutani, K., Komatsuzaki, H., Sakata, S., Yokota, R., & Kobayashi, S. (2020). Visual navigation based on semantic segmentation using only a monocular camera as an external sensor. In *Journal of Robotics and Mechatronics* (Vol. 32, Issue 6). <https://doi.org/10.20965/jrm.2020.p1137>

Moors, P., Germeyns, F., Pomianowska, I., & Verfaillie, K. (2015). Perceiving where another person is looking: The integration of head and body information in estimating another person's gaze. *Frontiers in Psychology*, 6(JUN). <https://doi.org/10.3389/fpsyg.2015.00909>

Mu, L., Yao, P., Zheng, Y., Chen, K., Wang, F., & Qi, N. (2020). Research on SLAM algorithm of mobile robot based on the fusion of 2D LiDAR and depth camera. *IEEE Access*, 8, 157628–157642. <https://doi.org/10.1109/ACCESS.2020.3019659>

Munaro, M., & Menegatti, E. (2014). Fast RGB-D people tracking for service robots. *Autonomous Robots*, 37(3), 227–242. <https://doi.org/10.1007/s10514-014-9385-0>

Mur-Artal, R., Montiel, J. M. M., & Tardos, J. D. (2015). ORB-SLAM: a Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics*, 31(5), 1147–1163.

<https://doi.org/10.1109/TRO.2015.2463671>

ORB SLAM Proposal for NTU GPU Programming Course 2016. (n.d.). Retrieved June 3, 2021, from <https://www.slideshare.net/MindosCheng/orb-slam-proposal-for-ntu-gpu-programming-course-2016>

Principle drawing of a stereo camera setup. Objects (1,2) in various... | Download Scientific Diagram. (n.d.). Retrieved June 3, 2021, from https://www.researchgate.net/figure/Principle-drawing-of-a-stereo-camera-setup-Objects-1-2-in-various-depth-ranges-are_fig5_303307354

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *IEEE Transactions on Pattern Analysis and Machine Intelligence* (Vol. 39, Issue 6). <https://doi.org/10.1109/TPAMI.2016.2577031>

Review: SENet — Squeeze-and-Excitation Network, Winner of ILSVRC 2017 (Image Classification) | by Sik-Ho Tsang | Towards Data Science. (n.d.). Retrieved June 3, 2021, from <https://towardsdatascience.com/review-senet-squeeze-and-excitation-network-winner-of-ilsvrc-2017-image-classification-a887b98b2883>

SCRUM - ECLEE | European Center for Leadership and Entrepreneurship Education. (n.d.). Retrieved June 3, 2021, from <https://www.eclee.com/training/scrum/>

Seichter, D., Köhler, M., Lewandowski, B., Wengefeld, T., & Gross, H.-M. (2020). Efficient RGB-D Semantic Segmentation for Indoor Scene Analysis. *ArXiv.* <http://arxiv.org/abs/2011.06961>

Shin, Y.-S., Yeong, , Park, S., & Kim, A. (2020). DVL-SLAM: sparse depth enhanced direct visual-LiDAR SLAM. *Autonomous Robots*, 44, 115–130. <https://doi.org/10.1007/s10514-019-09881-0>

Silberman, N., Hoiem, D., Kohli, P., & Fergus, R. (2012). Indoor segmentation and support inference from RGBD images. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Vol. 7576 LNCS* (Issue PART 5). https://doi.org/10.1007/978-3-642-33715-4_54

Song, S., Lichtenberg, S. P., & Xiao, J. (2015). SUN RGB-D: A RGB-D scene understanding benchmark suite. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (Vols. 07-12-June). <https://doi.org/10.1109/CVPR.2015.7298655>

Stereo visual SLAM system overview: First, we undistort and rectify the... | Download Scientific Diagram. (n.d.). Retrieved June 3, 2021, from https://www.researchgate.net/figure/Stereo-visual-SLAM-system-overview-First-we-undistort-and-rectify-the-stereo-images_fig1_257523126

Supercharge your computer vision models with synthetic datasets built by Unity | Unity Blog. (n.d.). Retrieved June 3, 2021, from <https://blog.unity.com/technology/supercharge-your-computer-vision-models-with-synthetic-datasets-built-by-unity>

Tamaki, T., Ogawa, D., Raytchev, B., & Kaneda, K. (2019). Semantic segmentation of trajectories with improved agent models for pedestrian behavior analysis. *Advanced Robotics*, 33(3–4), 153–168. <https://doi.org/10.1080/01691864.2018.1554508>

Teso-Fz-Betóñ, D., Zulueta, E., Sánchez-Chica, A., Fernandez-Gamiz, U., & Saenz-Aguirre, A. (2020). Semantic segmentation to develop an indoor navigation system for an autonomous mobile robot. *Mathematics*, 8(5). <https://doi.org/10.3390/MATH8050855>

Wang, R., Zhang, W., Shi, Y., Wang, X., & Cao, W. (2019). GA-ORB: A New Efficient Feature Extraction Algorithm for Multispectral Images Based on Geometric Algebra. *IEEE Access*, 7, 71235–71244. <https://doi.org/10.1109/ACCESS.2019.2918813>

Wang, Y., Chen, Q., Chen, S., & Wu, J. (2020). Multi-Scale Convolutional Features Network for Semantic Segmentation in Indoor Scenes. *IEEE Access*, 8, 89575–89583. <https://doi.org/10.1109/ACCESS.2020.2993570>

Weng, W., & Zhu, X. (2021). INet: Convolutional Networks for Biomedical Image Segmentation. *IEEE Access*, 9, 16591–16603. <https://doi.org/10.1109/ACCESS.2021.3053408>

What is Resnet or Residual Network | How Resnet Helps? (n.d.). Retrieved June 24, 2021, from <https://www.mygreatlearning.com/blog/resnet/>

Wolf, D., Bajones, M., Prankl, J., & Vincze, M. (2014). *Find my mug: Efficient object search with a mobile robot using semantic segmentation.* <http://arxiv.org/abs/1404.5765>

Xu, D., & Wu, Y. (2020). Improved YOLO-V3 with densenet for multi-scale remote sensing target detection. *Sensors (Switzerland)*, 20(15), 1–24. <https://doi.org/10.3390/s20154276>

Zaarane, A., Slimani, I., Al Okaishi, W., Atouf, I., & Hamdoun, A. (2020). Distance measurement system for autonomous vehicles using stereo camera. *Array*, 5, 100016.

<https://doi.org/10.1016/j.array.2020.100016>

ZED 2 - AI Stereo Camera | Stereolabs. (n.d.). Retrieved June 3, 2021, from <https://www.stereolabs.com/zed-2/>

Zhang, S., Zheng, L., & Tao, W. (2021). Survey and Evaluation of RGB-D SLAM. *IEEE Access*, 9, 21367–21387. <https://doi.org/10.1109/ACCESS.2021.3053188>

Anexos

Anexo. Artículo de investigación

En este anexo se añade el artículo del trabajo realizado. En este artículo se muestra en resumido todas las explicaciones sobre ello.

Segmentación semántica RGB-D para navegación en interiores

Mikel Aldalur Corta

Universidad Internacional de la Rioja, Logroño (España)

Fecha 15/09/2021



RESUMEN

Los robots y coches autónomos son cada vez más populares en nuestro entorno y la seguridad en cuanto a la navegación autónoma tiene que ser eficiente y efectiva. Este trabajo presenta una herramienta de segmentación semántica que se puede utilizar en cualquier tipo de robot para predecir estados o escenarios futuros cuando un robot está navegando por entornos de interior. Para la predicción, se utilizan las redes convolucionales LSTM que predicen cual es el posible escenario. Para que el sistema sea más eficiente que utilizando solo una cámara de color, se utilizan también las profundidades obtenidas por las cámaras infrarrojas. Para una posterior planificación, se introduce el modelo para que se pueda utilizar en ROS.

PALABRAS CLAVE

Aprendizaje profundo, segmentación semántica, visión RGBD, robots autónomos, ROS.

I. INTRODUCCIÓN

Los robots autónomos con la navegación autónoma son cada vez más comunes en el día a día, esto se debe a la mejora continua de la herramientas y aplicaciones. Al aumentar la popularidad de los robots autónomos, tanto los domésticos como los de exteriores (coches autónomos), la inteligencia artificial tiene que mejorar los sistemas de percepción para aumentar la seguridad tanto de los seres vivos como los elementos del entorno.

El problema de estos algoritmos es que para aumentar la seguridad de estos elementos ha realizado la segmentación semántica en las imágenes para mejorar la detección de objetos y aumentar el rendimiento y seguridad en cuanto al funcionamiento. Esto ayuda a tomar las decisiones adecuadas y planificar adecuadamente las trayectorias necesarias con el razonamiento lógico, contemplando las cualidades del robot y realizando unos cálculos en base a los datos recibidos.

Este trabajo se ha dividido en tres partes principales: segmentación semántica, previsión de estados futuros mediante la segmentación semántica e implementación del modelo en el robot. Se proporciona una propuesta de segmentación semántica que después se utiliza para preparar la base de datos que se utiliza para predecir los futuros escenarios. Este método de prevenir los estados futuros puede ayudar a realizar las acciones adecuadas antes que utilizando solo la imagen segmentada.

Todo esto se realiza utilizando modelos que tienen como entrada una imagen en color y otra con profundidades. Para ello se hace uso de la cámara de profundidad Intel RealSense que se incorpora en una Nvidia Jetson Nano. Esto se pone en marcha utilizando ROS, que incluye las librerías para la comunicación de los elementos. Para probar el funcionamiento del modelo, se han realizado pruebas en un hogar.

En el artículo podemos ver los siguientes apartados:

En el apartado II se realiza el estado del arte, donde se analizan las investigaciones referentes en cuanto a la visión por computador como navegación autónoma. En concreto, en cuanto a la visión, se realiza el análisis de localización, SLAM, detección de objetos y segmentación semántica, en cambio, en cuanto a navegación, se analizan la planificación de trayectoria y predicción de trayectorias.

En el apartado III se detallan los objetivos específicos para dirigir este trabajo y también la metodología a utilizar, mientras que en el apartado IV se explica lo que se aporta en este trabajo y las características que se tienen en cuenta para realizar esta contribución.

Una vez explicada en qué se basa el trabajo, se analizan los resultados obtenidos y se evalúan en el apartado V junto con una discusión sobre los mismos en el apartado VI. Por último, se analiza el trabajo para mostrar las conclusiones de mismo.

II. ESTADO DEL ARTE

En los últimos años el mundo de la robótica ha dado pasos agigantados, prueba de ello son los coches autónomos, robots humanoides o cuadrúpedos. Estos robots son los que ayudan a los seres humanos realizar o ayudar en las tareas que pueden ser agotadoras o inhumanas. Por eso, para que los robots puedan interactuar con los seres humanos, se está trabajando en lograr un sistema de visión y navegación seguros y viables.

Esto implica que los robots tengan una mínima conciencia de por dónde están circulando y qué es lo que se encuentra a su alrededor. Con esto, se podría garantizar la integridad de las personas, elementos de alrededor como del mismo robot.

Existen variedad de sistemas con los que proporcionar a los robots la inteligencia para realizar esa navegación lo más parecido a como lo realizaría un ser humano.

SLAM mediante visión

SLAM (localización y mapeo simultaneo) es un sistema que percibe el entorno para crear un mapa en base a lo procesado de la percepción recibida. Para realizar este tipo de mapeo existen diferentes métodos a los que se puede referir.

Uno de ellos puede ser la cámara monocular, con el que se puede extraer información del entorno para un posterior procesamiento. Para la extracción de características existen diferentes algoritmos como ORB [1] que es la derivación y mejora de los algoritmos SURF y SIFT. Al realizar la extracción de características, este algoritmo puede identificar los elementos de una imagen y detectar los mismos elementos en la imagen posterior al movimiento. Añadiendo un sensor de inercia se puede realizar un mapa en tres dimensiones, este algoritmo realiza el mapa del entorno más preciso y el algoritmo se llama ORB-SALM [2].

También existen algoritmos que trabajan con cámara estereó. Estas suelen ser dos cámaras, normalmente en paralelo, que al percibir el entorno de dos puntos diferentes pueden llegar a medir las distancias [3]. Funciona con el algoritmo de extracción de características que detectara los elementos en ambas cámaras, según donde se encuentre el elemento en cada imagen, se calcula la distancia a la que se encuentra ese elemento.

Las cámaras estereoscópicas son similares a las cámaras dobles en cuanto al funcionamiento, que perciben el entorno en tres dimensiones, pero estas cámaras tienen cámaras infrarrojos de proximidad para realizar la medición de las distancias. Su ejecución es similar a la cámara doble en cuanto a la creación de mapas [4] con la diferencia de que en estas por un lado está la imagen y por otro la profundidad de esta.

Otro de los elementos comunes en los robots es el *LiDAR*, que realiza la detección de luz y rango para medir las distancias de los elementos de su alrededor. Este elemento se utiliza para distintos usos, uno de ellos es localizar el robot [5] y el otro es generar mapas de dos o tres dimensiones [6][7] partiendo de los datos recibidos.

Detección y segmentación

La detección y clasificación de imágenes es uno de los pilares en cuanto a la navegación de los robots, ya que tienen que entender el entorno. Y para ello se usan la clasificación y detección de objetos.

En estos casos el tiempo de ejecución tiene que ser el mínimo posible y por eso no se suelen usar modelos alojados en la nube, sino que se suelen usar modelos en los sistemas locales [8].

Durante años se ha ido mejorando la detección en las imágenes, pero lo primero que se logró, fue clasificar las imágenes [9]. Después de pocos años, se realizó la detección de varios objetos en una imagen [10], que ayudó mucho para detectar los elementos del entorno y lo último que se ha logrado realizar la detección de objetos prácticamente a tiempo real mediante el modelo Yolo [11].

La detección y clasificación de objetos puede servir para multitud de operaciones, pero al estar hablando de los robots, tienen que percibir el entorno tal y como lo perciben los humanos. Para ello, se empezó a realizar la segmentación semántica, que consiste en

clasificar las partes de imágenes según la categoría a la que pertenece [12].

La segmentación ha traído diferentes usos como la segmentación de sistemas monoculares [13] que realizan la segmentación de una imagen, la segmentación con sistemas de profundidad [14] para tener una segmentación en tres dimensiones, los mapas creados después de segmentar el entorno [15], que consiste en obtener los mapas segmentados, ver Ilustración 1. Esto puede ayudar a los robots a saber en qué lugar se encuentran y por donde deben de realizar las trayectorias.

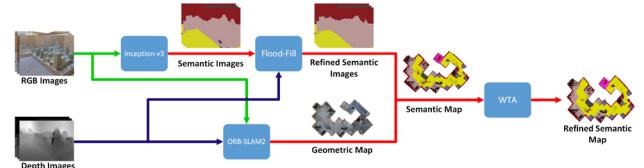


Ilustración 1: mapeo con segmentación semántica [].

Las segmentaciones suelen ser para interior [16] o exterior, que a su vez pueden añadir la detección de objetos [17].

Tal y como se menciona anteriormente, uno de los problemas suele ser la ejecución de los modelos y es necesario mejorarlo para realizar un correcto mapeo [18]. Para realizar mapas es necesario que los elementos móviles no estén en el mapa generado, si no pude que la trayectoria elegida no sea la óptima [19]. Los sistemas monoculares también pueden crear SLAM [20] captando el entorno con sus características y las inercias de movimiento para crear un mapa.

También existen algoritmos de segmentación semántica no supervisados [21], tal y como dice el nombre este modelo no se entrena en base a los resultados.

Trayectoria de los humanos

Saber la posición en la que se encuentra un ser humano está bien, pero puede ayudar mucho más saber la trayectoria que puede llegar a hacer esa persona para poder esquivar y realizar una trayectoria eficiente.

Saber la pose y la dirección de un humano [22] puede ayudar en la función de esquivar, ya que se puede prevenir que va a moverse según su dirección. También puede ser interesante saber las trayectorias más concurrencia de los humanos [23] para intentar evitarlos [24], ayudando con la gesticulación y la mirada [25].

También existen casos donde el robot previene la trayectoria del humano según los objetos detectados en su cuerpo [26]. Esto quiere decir que el robot puede predecir hacia donde irá la gente según los objetos detectados. También se pueden prevenir los movimientos de los humanos según los previos movimientos realizados [27] realizando un mapa de estos.

Para ejecutar este trabajo es necesario tener las bases de datos como [28] [29] [30] donde podemos entrenar los modelos para diferentes entornos. Es interesante entrenar los modelos en diferentes entornos para que puedan realizar un buen trabajo. No es lo mismo realizar la predicción de un entorno donde previamente se ha entrenado el modelo que un nuevo entorno. Para eso es interesante tener distintas bases de datos en las que entrenar los modelos.

También existen *dataset* con las imágenes segmentadas para poder entrenar el modelo [31] [32]. Estas bases de datos contienen imágenes en color y en profundidad, que ayudan en cuanto a realizar la segmentación. Además, contienen las imágenes de

color y profundidad tipo video, en *frames*, para poder realizar una mejor predicción.

III. OBJETIVOS Y METODOLOGÍA

Objetivo general

El objetivo general de este trabajo es realizar la función de segmentación de una imagen y preparar una predicción futura de la segmentación para la navegación de robots.

Objetivos específicos

- Realizar el análisis de los sistemas utilizados para detección del entorno como la segmentación semántica para realizar la navegación.
- Realizar la segmentación semántica del entorno, para que el robot sea capaz de segmentar el entorno en categorías.
- Detectar el movimiento y predecir los estados futuros que puede tener un robot cuando está navegando.
- Implementar el algoritmo de percepción para la predicción de estados con segmentación semántica en el *framework* ROS (Robot Operating System).

Metodología

Con el fin de alcanzar los objetivos indicados en el desarrollo del trabajo, se ha utilizado la metodología Scrum. Esta metodología implica tener objetivos a corto plazo por si ocurren cambios y se tiene que modificar el proceso de desarrollo. En este tipo de metodologías la planificación no suele ser la definitiva, se suele ir modificando para ajustarse lo máximo posible al ejecutar y desarrollar los proyectos.

Para lograr el objetivo final se han establecido los siguientes sprint o subobjetivos:

1. Analizar
2. Realizar el algoritmo de segmentación semántica
3. Realizar el algoritmo de segmentación semántica con predicción.
4. Implementar los algoritmos en una Jetson Nano
5. Implementar el algoritmo en ROS.
6. Pruebas y evaluación.

IV. CONTRIBUCIÓN

La contribución de este trabajo consiste en realizar una segmentación semántica de un entorno y generar la predicción de un escenario futuro orientado a la navegación autónoma de los robots. En este caso, se implementa el modelo en entorno de interior utilizando para ello la cámara Intel RealSense 435i, con el que se recogen los datos del entorno en imagen RGB y también en imágenes de profundidad para un posterior procesado. La implementación se realiza en una Jetson Nano, una de las tarjetas de desarrolladores simple que integra una GPU de 4GB.

Lo que se pretende con este trabajo es realizar las planificaciones de navegación más eficientes y aumentar la

seguridad e integridad de los seres humanos y objetos del entorno donde se encuentra el robot.

Para procesar con la segmentación semántica, se ha utilizado el *dataset* de SUNRGBD [31] y se ha realizado un modelo para predecir la segmentación semántica. Aunque las imágenes capturadas por la cámara Intel son de 480 x 640, en este modelo se han utilizado las imágenes en 96 x 128 ya que el modelo no puede ser entrenado con más resolución.

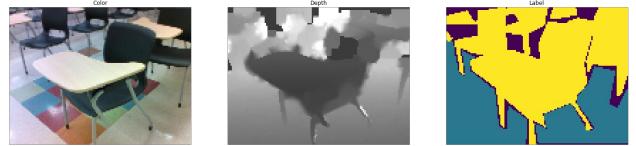


Ilustración 2: muestra de la imagen de la base de datos SUNRGBD.

Para realizar la segmentación, se obtienen dos imágenes de entrada, una la imagen de color y la otra la imagen de profundidad. Las dos imágenes entran en el modelo y como salida se obtiene una imagen segmentada. La obtención de una sola imagen se debe a la implementación de fusionar las dos imágenes que utiliza el modelo ESANet [14] ver Ilustración 2, obteniendo de esta manera de las dos imágenes un único tensor.

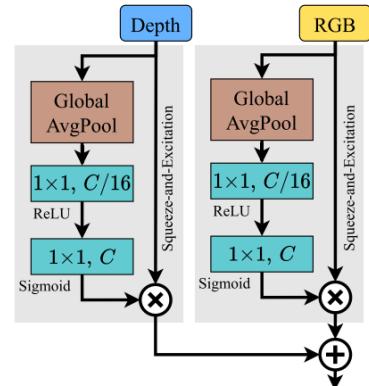


Ilustración 3: Parte del modelo para fusionar las imágenes de color y profundidad.

El modelo escogido para la segmentación semántica ha sido el modelo de UNet [33], donde se utilizan los procesos de convolución y se guardan los vectores para volver a generar la imagen segmentada.

Con el entrenamiento de este modelo, se han logrado obtener buenas métricas. El resultado obtenido, como la misma entrada, es de 96 x 128 y se divide en 38 categorías diferentes.

El modelo previamente entrenado ha servido para crear la base de datos segmentada de NYUV2 [32], donde se encuentras los videos para realizar la predicción de un futuro escenario. Para ello, ha sido necesario utilizar el modelo anterior con los que se han realizado las etiquetas de las imágenes para que cuando se entrene el modelo tenga el objetivo definido.

Estos datos se han dividido en secuencias de 16 imágenes, las cuales están alojadas en entornos diferentes de interiores. Para poder realizar la predicción de los escenarios futuros, se han realizado varias pruebas donde se obtienen resultados diferentes.

Uno de los modelos entrenados contiene tres LSTM en la salida del modelo de UNet para guardar la secuencia predicha en la sección y volver a utilizarla en la decodificación de dicha sección.

Por otro lado, se ha realizado la prueba de tener las convoluciones LSTM en cada fusión, a la entrada de la

decodificación, para que el modelo pueda aprender a procesar esas imágenes y obtener el estado final. En la Ilustración 4 se puede ver cómo es la estructura.

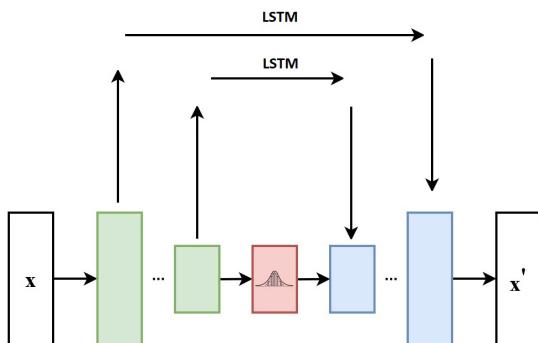


Ilustración 4: Imagen del modelo UNet con LSTM.

Por último, se elige el modelo que ha dado los mejores resultados para hacer la inferencia a la tarjeta Jetson Nano. Para eso ha sido necesario convertir el modelo de Keras a un modelo de TensorRt para que la tarjeta gráfica de la placa realice los cálculos de una manera más eficiente. Una vez obtenido el modelo, se ha realizado un nodo en ROS donde se hace uso de OpenCV utilizando `cv_bridge` y también el nodo de Intel RealSense.

Una vez realizado el modelo e implementado en un robot, se procede a realizar las pruebas para comprobar el funcionamiento.

V. EVALUACIÓN Y RESULTADOS

La evaluación de estos modelos se realiza en dos partes, por un lado, se analiza el modelo de segmentación semántica, la cual se utiliza para generar la base de datos para entrenar el modelo de predicción, el que se evaluará por otro lado.

Evaluación 1

El primer modelo, el de la segmentación semántica, categoriza las imágenes de una forma bastante correcta, aunque hay ocasiones en que no detecta el suelo eficientemente. En la Tabla I se muestran los resultados obtenidos.

TABLA I

MÉTRICAS DEL MODELO

MÉTRICA	RESULTADO
Exactitud	0,77271235
<i>IoU</i>	0,34801486
True positive	0,76918536

Las imágenes que ha predicho el modelo se pueden ver en la Ilustración 5 y 6. En estas ilustraciones se muestran las dos imágenes de entrada y la salida, la predicha por el modelo.



Ilustración 5: Escenario A y la predicción.

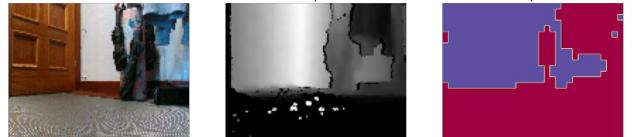


Ilustración 6: Escenario B y la predicción.

Se puede ver que el modelo no detecta las personas, pero detecta el suelo con una predicción alta, pero a veces identifica otros objetos como suelo. Se puede ver en la Ilustración 6 que en la predicción del suelo también entran los objetos.

Evaluación 2

Por otra parte, se analiza el modelo de predicción para detectar los escenarios futuros. Este modelo, al tener el *dataset* creado con el modelo anterior, no detecta realiza una predicción buena.

El primero de los modelos realiza una predicción mejor que la segunda, por lo que solo se muestra esta. Se pueden ver los resultados en la ilustración 9 y sus métricas de evaluación en la tabla 2.



Ilustración 7: Imágenes de color escenario C.

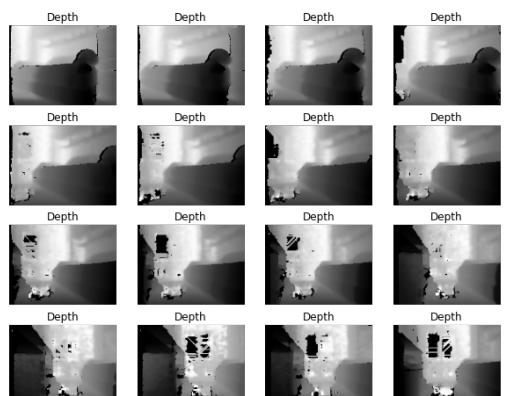


Ilustración 8: Imágenes de profundidad escenario C.



Ilustración 9: Resultados obtenidos escenario C.

Como se puede ver en la imagen, la predicción de los escenarios futuros no es la adecuada. Realizan una predicción similar en muchos de los escenarios probados. Esto se debe a que la base de datos utilizada no es la debida.

Los resultados de este modelo se pueden ver en la Tabla II. Se puede ver claramente que los objetos detectados tienen una alta exactitud, pero el *recall* es muy bajo. Esto significa que el modelo no es bueno para este tipo de predicciones.

TABLA II
MÉTRICAS DEL MODELO

MÉTRICA	RESULTADO
Exactitud	0,6133
Precisión	1,0
<i>Recall</i>	0,1716

VI. DISCUSIÓN

Tras la presentación objetiva de los resultados, se puede decir que los modelos necesitan una calidad mayor en las imágenes ya que con la calidad utilizada no puede detectar los objetos adecuadamente. Para su procesamiento sería necesaria una tarjeta gráfica más potente.

También sería interesante crear una base de datos de código abierto que sea la adecuada para este tipo de problemática, donde las imágenes estén etiquetadas para un posterior entrenamiento.

En cambio, la idea de introducir la herramienta en el sistema ROS es interesante. Esto hace posible que se pueda implementar en cualquier robot, ya sea de algún fabricante en concreto o de autoría propia.

VII. CONCLUSIONES

Se ha desarrollado un sistema que es capaz de segmentar una imagen por categorías, utilizando para ello un modelo creado en base a los artículos e investigaciones anteriores. Los resultados de este modelo son buenos, pero se tiene que para crear una base de datos más completa y preciso. Si se dispusiese de más tiempo se invertiría en realizar un *dataset* adecuado para realizar la predicción futura.

Una vez creado el modelo segmentación semántica, se ha creado un *dataset* y se han analizado las posibilidades de crear

diferentes modelos de predicción. Tras varias pruebas, los modelos más efectivos, con los que se podría trabajar en futuras investigaciones, han dado diferentes resultados. Por un lado, existe un modelo sobreentrenado y por otro, existe el modelo que funciona con una precisión bastante buena. Aun y todo, se puede mejorar el rendimiento aumentando la calidad del *dataset*, que, en este caso, se ha creado con el modelo de segmentación semántica.

Por último, se ha implementado el algoritmo que predice los escenarios futuros en una tarjeta Jetson Nano y se han creado los nodos de ros para ponerlo en funcionamiento. La velocidad de procesado es mejorable, ya que se ha obtenido el procesado de 6 imágenes por segundo.

Al realizar el trabajo se pueden ver mejoras en el proceso y se sugieren las siguientes líneas de trabajo.

- Crear una base de datos robusta para predicción de futuros escenarios.
- Crear otra base de datos para análisis de los movimientos de personas.
- Realizar un nuevo modelo para predicción y robarlo en una tarjeta más potente para la puesta en marcha en un sistema real.
- Realizar la predicción con varias cámaras para tener un rango más amplio de visibilidad.

REFERENCIAS

- [1] Wang, R., Zhang, W., Shi, Y., Wang, X., & Cao, W. (2019). GA-ORB: A New Efficient Feature Extraction Algorithm for Multispectral Images Based on Geometric Algebra. *IEEE Access*, 7, 71235–71244. <https://doi.org/10.1109/ACCESS.2019.2918813>
- [2] Mur-Artal, R., Montiel, J. M. M., & Tardos, J. D. (2015). ORB-SLAM: a Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics*, 31(5), 1147–1163. <https://doi.org/10.1109/TRO.2015.2463671>
- [3] Zaarane, A., Slimani, I., Al Okaishi, W., Atouf, I., & Hamdoun, A. (2020). Distance measurement system for autonomous vehicles using stereo camera. *Array*, 5, 100016. <https://doi.org/10.1016/j.array.2020.100016>
- [4] Zhang, S., Zheng, L., & Tao, W. (2021). Survey and Evaluation of RGB-D SLAM. *IEEE Access*, 9, 21367–21387. <https://doi.org/10.1109/ACCESS.2021.3053188>
- [5] Chang, L., Niu, X., & Liu, T. (2020). Gnss/imu/odo/lidar-slam integrated navigation system using imu/odo pre-integration. *Sensors (Switzerland)*, 20(17), 1–18. <https://doi.org/10.3390/s20174702>
- [6] Shin, Y.-S., Yeong, , Park, S., & Kim, A. (2020). DVL-SLAM: sparse depth enhanced direct visual-LiDAR SLAM. *Autonomous Robots*, 44, 115–130. <https://doi.org/10.1007/s10514-019-09881-0>
- [7] Mu, L., Yao, P., Zheng, Y., Chen, K., Wang, F., & Qi, N. (2020). Research on SLAM algorithm of mobile robot based on the fusion of 2D LiDAR and depth camera. *IEEE Access*, 8, 157628–157642. <https://doi.org/10.1109/ACCESS.2020.3019659>
- [8] Xu, D., & Wu, Y. (2020). Improved YOLO-V3 with densenet for multi-scale remote sensing target detection. *Sensors (Switzerland)*, 20(15), 1–24. <https://doi.org/10.3390/s20154276>
- [9] Chan, T.-H., Jia, K., Gao, S., Lu, J., Zeng, Z., & Ma, Y. (2014). PCANet: A Simple Deep Learning Baseline for Image Classification? *IEEE Transactions on Image Processing*, 24(12), 5017–5032. <https://doi.org/10.1109/TIP.2015.2475625>
- [10] Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (Vol. 39). <https://doi.org/10.1109/TPAMI.2016.2577031>
- [11] Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection.

- ArXiv*. Retrieved from <http://arxiv.org/abs/2004.10934>
- [12] Wolf, D., Bajones, M., Prankl, J., & Vincze, M. (2014). Find my mug: Efficient object search with a mobile robot using semantic segmentation. Retrieved from <http://arxiv.org/abs/1404.5765>
- [13] Wang, Y., Chen, Q., Chen, S., & Wu, J. (2020). Multi-Scale Convolutional Features Network for Semantic Segmentation in Indoor Scenes. *IEEE Access*, 8, 89575–89583. <https://doi.org/10.1109/ACCESS.2020.2993570>
- [14] Seichter, D., Köhler, M., Lewandowski, B., Wengefeld, T., & Gross, H.-M. (2020). Efficient RGB-D Semantic Segmentation for Indoor Scene Analysis. *ArXiv*. Retrieved from <http://arxiv.org/abs/2011.06961>
- [15] Deng, W., Huang, K., Chen, X., Zhou, Z., Shi, C., Guo, R., & Zhang, H. (2020). Semantic RGB-D SLAM for Rescue Robot Navigation. *IEEE Access*, 8, 221320–221329. <https://doi.org/10.1109/ACCESS.2020.3031867>
- [16] Teso-Fz-Betöño, D., Zulueta, E., Sánchez-Chica, A., Fernandez-Gamiz, U., & Saenz-Aguirre, A. (2020). Semantic segmentation to develop an indoor navigation system for an autonomous mobile robot. *Mathematics*, 8(5). <https://doi.org/10.3390/MATH8050855>
- [17] Feng, D., Haase-Schutz, C., Rosenbaum, L., Hertlein, H., Glaser, C., Timm, F., ... Dietmayer, K. (2021). Deep Multi-Modal Object Detection and Semantic Segmentation for Autonomous Driving: Datasets, Methods, and Challenges. *IEEE Transactions on Intelligent Transportation Systems*, 22(3), 1341–1360. <https://doi.org/10.1109/TITS.2020.2972974>
- [18] Li, F., Li, W., Chen, W., Xu, W., Huang, L., Li, D., ... Liu, Y. (2020). A Mobile Robot Visual SLAM System with Enhanced Semantics Segmentation. *IEEE Access*, 8, 25442–25458. <https://doi.org/10.1109/ACCESS.2020.2970238>
- [19] Ai, Y., Rui, T., Lu, M., Fu, L., Liu, S., & Wang, S. (2020). DDL-SLAM: A robust RGB-d SLAM in dynamic environments combined with deep learning. *IEEE Access*, 8, 162335–162342. <https://doi.org/10.1109/ACCESS.2020.2991441>
- [20] Miyamoto, R., Adachi, M., Ishida, H., Watanabe, T., Matsutani, K., Komatsuzaki, H., Sakata, S., Yokota, R., & Kobayashi, S. (2020). Visual navigation based on semantic segmentation using only a monocular camera as an external sensor. In *Journal of Robotics and Mechatronics* (Vol. 32, Issue 6). <https://doi.org/10.20965/jrm.2020.p1137>
- [21] Jin, S., Chen, L., Sun, R., & McLoone, S. (2020). A novel vSLAM framework with unsupervised semantic segmentation based on adversarial transfer learning. *Applied Soft Computing Journal*, 90, 106153. <https://doi.org/10.1016/j.asoc.2020.106153>
- [22] Cui, X., Lu, C., & Wang, J. (2020). 3D Semantic Map Construction Using Improved ORB-SLAM2 for Mobile Robot in Edge Computing Environment. *IEEE Access*, 8, 67179–67191. <https://doi.org/10.1109/ACCESS.2020.2983488>
- [23] Tamaki, T., Ogawa, D., Raytchev, B., & Kaneda, K. (2019). Semantic segmentation of trajectories with improved agent models for pedestrian behavior analysis. *Advanced Robotics*, 33(3–4), 153–168. <https://doi.org/10.1080/01691864.2018.1554508>
- [24] Liu, W., Chan, A. B., Lau, R. W. H., & Manochaeee, D. (2015). Leveraging long-term predictions and online learning in agent-based multiple person tracking. In *IEEE Transactions on Circuits and Systems for Video Technology* (Vol. 25, Issue 3). <https://doi.org/10.1109/TCSVT.2014.2344511>
- [25] Moors, P., Germeyns, F., Pomianowska, I., & Verfaillie, K. (2015). Perceiving where another person is looking: The integration of head and body information in estimating another person's gaze. *Frontiers in Psychology*, 6(JUN). <https://doi.org/10.3389/fpsyg.2015.00909>
- [26] Bruckschen, L., Bungert, K., Dengler, N., & Bennewitz, M. (2020). Human-aware robot navigation by long-term movement prediction. In *IEEE International Conference on Intelligent Robots and Systems*. <https://doi.org/10.1109/IROS45743.2020.9340776>
- [27] Munaro, M., & Menegatti, E. (2014). Fast RGB-D people tracking for service robots. *Autonomous Robots*, 37(3), 227–242. <https://doi.org/10.1007/s10514-014-9385-0>
- [28] Lewandowski, B., Wengefeld, T., Muller, S., Jenny, M., Glende, S., Schroter, C., Bley, A., & Gross, H.-M. (2020). Socially Compliant Human-Robot Interaction for Autonomous Scanning Tasks in Supermarket Environments. *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 363–370. <https://doi.org/10.1109/RO-MAN47096.2020.9223568>
- [29] Gross, H. M., Boehme, H., Schroeter, C., Mueller, S., Koenig, A., Einhorn, E., Martin, C., Merten, M., & Bley, A. (2009). TOOMAS: Interactive shopping guide robots in everyday use - Final implementation and experiences from long-term field trials. *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2009)*, 2005–2012. <https://doi.org/10.1109/IROS.2009.5354497>
- [30] Gross, H. M., Meyer, S., Scheidig, A., Eisenbach, M., Mueller, S., Trinh, T. Q., Wengefeld, T., Bley, A., Martin, C., & Fricke, C. (2017). Mobile robot companion for walking training of stroke patients in clinical post-stroke rehabilitation. *Proceedings - IEEE International Conference on Robotics and Automation*, 1028–1035. <https://doi.org/10.1109/ICRA.2017.7989124>
- [31] Song, S., Lichtenberg, S. P., & Xiao, J. (2015). SUN RGB-D: A RGB-D scene understanding benchmark suite. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (Vols. 07-12-June). <https://doi.org/10.1109/CVPR.2015.7298655>
- [32] Silberman, N., Hoiem, D., Kohli, P., & Fergus, R. (2012). Indoor segmentation and support inference from RGBD images. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*: Vol. 7576 LNCS (Issue PART 5). https://doi.org/10.1007/978-3-642-33715-4_54
- [33] Ronneberger, O., Fischer, P., & Brox, T. (n.d.). *U-Net: Convolutional Networks for Biomedical Image Segmentation*. Retrieved July 14, 2021, from <http://lmb.informatik.uni-freiburg.de/>