PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

Escuela de Posgrado

Desarrollo de un robot móvil con inteligencia artificial para recolectar botellas de plástico

Tesis para obtener el grado académico de Magíster en Ingeniería Mecatrónica que presenta:

Ing. Manuel Antonio Chávez Salinas

Asesor:

Dr. Ing. Celso De La Cruz Casaño

Lima, 2022

RESUMEN

La robótica en la actualidad busca incrementar los lazos entre robots y seres humanos, de forma que realizar las tareas en conjunto sea más amigable y no se requiera de un extenso periodo de adaptación. Ello, en ocasiones se ve perjudicado por la operación manual constante que requiere un robot debido a que un individuo emplea su tiempo para manipular el robot y no se genera la independencia anhelada en este campo. Por lo descrito, surge como alternativa de solución los robots autónomos que emplean visión artificial y algoritmos de Inteligencia Artificial (IA) para el cumplimiento de tareas.

Este trabajo de tesis, se centra en el diseño y desarrollo de un robot móvil utilizando visión artificial para identificar botellas de plástico en las playas del litoral limeño. Se emplea una RCNN para el procesamiento de imágenes y esta red es entrenada por imágenes de autoría propia, las cuales se realizaron en las playas mencionadas, hasta obtener un óptimo reconocimiento de las botellas y las personas. Asimismo, se generaron propuestas de diseño de los dominios mecánico, electrónico y de control, de tal forma que el proyecto resultante abarque de forma integral el funcionamiento esperado.

Se realizaron pruebas de simulación al sistema de control y visión artificial y se obtuvo que, efectivamente consta de una alternativa de solución robusta, pues reconoce las botellas de plástico y se acerca hacia ellas, esquivando obstáculos y consiguiendo su objetivo. Se realizaron simulaciones donde se obtuvieron parámetros que confirman lo designado en el diseño.

Esta tesis, también tiene un impacto significativo en el medio ambiente, pues se trata de un robot eco amigable que trabaja para el beneficio de las playas, de tal forma que se inspiren futuros trabajos de investigación que tengan por consigna impulsar la sostenibilidad.

DEDICATORIA

A Dios, por darme la oportunidad y guiar mis pasos

A mi madre Graciela, por el apoyo y amor incondicional

A mi novia Aracelli, por el amor y la motivación

A mi asesor Celso, por la confianza y la ayuda brindada

Al profesor Willy Carrera, por exigirme al máximo, Q.E.P.D.

Al profesor Renzo Zanabria, gran entrenador y amigo, Q.E.P.D

Les dedico esta tesis y les expreso mi gratitud.

AGRADECIMIENTOS

Agradezco a mi asesor, el Dr. Ing. Celso De La Cruz, por su ayuda desde el respaldo de la concepción del proyecto hasta el desarrollo realizado. Asimismo, expreso mi gratitud por la motivación constante y la confianza brindada desde que lo conozco.

Le agradezco a mi versión del pasado por nunca rendirse y confiar que existe un arcoíris al final de cada tormenta, gracias a eso mi mejor presente es un buen futuro.

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Pág.

INTRODUCCIÓN

La tecnología es el conjunto de conocimientos sistematizados para la creación de medios con el fin de satisfacer necesidades específicas del ser humano. El estudio de las ramas de la robótica, mecánica y electrónica son ciencias que estudian las técnicas y procesos a seguir para la invención de robots y maquinas que contribuyen al trabajo del ser humano.

Un robot móvil se desplaza sobre un territorio firme por medio de patas o llantas siguiendo una trayectoria. En este trabajo de tesis se realizará el desarrollo de algoritmos para un robot móvil tipo vehículo. Se busca que en un futuro el robot móvil diseñado contribuya a la limpieza de las playas, identificando las botellas de plástico para su posterior recolección mediante un software de control con tecnología de visión artificial, donde no tendrá una trayectoria definida, sino que se desplazará conforme detecte las botellas requeridas. La culminación del prototipo cumplirá con el objetivo y servirá como un antecedente o guía para investigaciones futuras en el campo tecnológico en beneficio del medio ambiente. En complemento, se tendrán propuestas de diseño para los dominios mecánico y electrónico correspondientes al robot móvil. De esta manera, la articulación entre la visión artificial, el sistema de control y los dominios mencionados será integral.

Es importante cumplir el objetivo de diseñar el algoritmo de visión artificial para agilizar la limpieza de playas como se ha descrito, pero también es importante sembrar conciencia, pues de lo contrario las personas que asisten a las playas continuarán ensuciándolas. Por ello, mientras el robot identifica las botellas de plástico, mostrará imágenes del medio ambiente y como la contaminación afecta al planeta, así se busca aportar en la educación y responsabilidad ambiental.

CAPÍTULO 1

PROBLEMÁTICA Y ESTADO DEL ARTE

En el presente capítulo, se exponen las carencias de los métodos actuales para recolectar botellas de plástico, el cómo no hay una forma de realizar la identificación y recolección de botellas de forma autónoma. Esto, establece la necesidad de desarrollar un robot móvil autónomo que emplee visión artificial y permita identificar las botellas de plástico para su recolección. De esta forma, se puede contribuir a la limpieza de playas y protección del medio ambiente. Asimismo, en el capítulo también se explora la cultura tecnológica a nivel nacional e internacional mediante estudios y proyectos realizados en diversos países del mundo, relacionados a la tecnología empleada en esta tesis.

1.1. Problemática

En el contexto actual del país, existen comunidades que presentan una alta tasa de contaminación y una muestra de ello es la cantidad de botellas de plástico que se encuentran tanto en desmontes, como en ríos y playas del litoral limeño [MATEUS, 2013]. Los trabajadores y recicladores de botellas plásticas, a menudo exponen su salud debido a que las botellas no son almacenadas en un contenedor exclusivo y suelen estar acompañadas de otros desperdicios dañinos, por lo que estos trabajadores se ven obligados a realizar una separación y selección manual.

En la actualidad, no existe un método que les permita recolectar botellas de plástico sin exponerse a otros residuos y sin detener las actividades de la zona donde se realice la recolección [ADEME, 2018]. Es de esta necesitad explorada, que surge el requerimiento de diseñar un robot recolector que sea eficiente en autonomía y que, mediante una propuesta de diseño de dominios mecánico, electrónico y de control, logre complementar el proceso de identificación de botellas empleando visión artificial, donde se busca reconocerlas dentro un determinado entorno, en este caso las playas. A continuación, se puede ver el proceso de recolección y reciclaje de botellas de plástico descrito por el ministerio del ambiente.

- Se depositan las botellas plásticas reciclables a los contenedores designados para que sean recicladas.
- Los camiones de las plantas recicladoras recogen las botellas y las llevan a cada planta.
- Se prensan las botellas y se forman grandes bloques.
- Se lavan las botellas, se eliminan sus etiquetas, se secan y las clasifican según el tipo de plástico y para que se vaya a usar.
- Se trituran las botellas en trozos pequeños y se vuelven a lavar y secar.
- Se reutilizan las botellas en un nuevo proceso de fabricación de productos como botellas nuevas de plástico, envases, fibra textil, ropa como forros polares, juguetes, etcétera.

El proceso de recolección de botellas de plástico se clasifica en dos grupos:

a. Recolección manual: En este grupo, las botellas de plástico se recogen manualmente siguiendo el proceso descrito. Esto requiere que los trabajadores se encuentren operando todo el tiempo que se desee recolectar las botellas de plástico, así como disponer de una logística para transporte, equipamiento y costos elevados en el tratamiento del plástico. Es decir, se emplea tiempo y esfuerzo físico en exceso al mismo tiempo.

b. Recolección automática: Este grupo, consta de todo proceso que no solo depende de los trabajadores, sino que se ha agregado un agente externo capaz de mejorar la recolección de botellas de plástico, reducir el tiempo o el esfuerzo físico que demanda a los trabajadores recoger las botellas y eliminar las condiciones adversas en las que trabajan.

En estos grupos, se han explorado soluciones anteriormente, sin embargo, hay aspectos que deben optimizarse. Entre ellos se encuentran, la autonomía del robot, el tiempo de funcionamiento ininterrumpido y la política no invasiva, es decir, que no se interrumpan las actividades de las personas que lo rodean.

Por lo descrito, se puede ver que es improbable realizar un proceso manual en una playa,

ya que se interrumpiría de forma invasiva las actividades de visitantes y trabajadores para recolectar las botellas plásticas en camiones. Por otro lado, los recolectores continuarían sometidos al esfuerzo no ergonómico que implica separar las botellas de plástico de otros residuos a lo largo de la playa. Es por ello, que se plantea el diseño de un robot móvil para recolectar botellas de plástico en la orilla de las playas limeñas, desplazándose en la arena usando visión artificial para identificar las botellas de plástico y de esta manera recolectarlas de forma no invasiva.

A diferencia de las soluciones que se han planteado hasta el momento, se quiere incluir la visión artificial, ya que de esta manera el robot se desplazará directamente a las botellas sin necesidad de estar en constante movimiento, lo cual le permite menor consumo de potencia y evita el desgaste innecesario de componentes.

Es fundamental la concepción de un robot de esta naturaleza dado que las playas limeñas poseen un alto índice de contaminación, como se puede ver en el siguiente estudio. Según el ministerio del ambiente, en una entrevista se obtuvo la siguiente declaración: "Según el especialista consultado [HERNANDEZ, 2019], más del 50% de la basura marina que contamina las playas y el mar peruano es plástico de diversas formas, que termina en el agua por culpa de una mala gestión de los residuos sólidos y la falta de conciencia y educación ambiental ciudadana".

1.1.1. Objetivos

- Objetivo General:

Desarrollar algoritmos de Inteligencia Artificial para detectar las botellas de plástico en la playa bajo condiciones reales.

- Objetivos específicos:
  - Elaborar el estado del arte abarcando un diseño en el subsistema de control.
  - Extraer requerimientos de la problemática para el diseño del robot y sus dominios respectivos.

- Desarrollar el algoritmo de visión artificial.
- Evaluar el algoritmo de visión artificial para detectar botellas en imágenes de playa.
- Realizar los cálculos y selección de componentes del dominio de control del diseño conceptual propuesto, incluyendo análisis de la dinámica del movimiento a lo largo de la arena.
- Evaluar el algoritmo de control del robot móvil por simulación.

1.1.2. Alcances

El presente trabajo está delimitado o tiene el siguiente alcance:

Con el fin de mejorar el tiempo de respuesta de trabajos existentes, se realizará la propuesta de un sistema de control y algoritmos que permitan el uso de visión artificial para identificar las botellas de plástico y recolectarlas de una manera no invasiva para no alterar las actividades de las personas asistentes a las playas. Asimismo, se tendrá una propuesta de diseño de los dominios correspondientes para que el robot pueda operar óptimamente tanto en sus funciones como en su interacción con los asistentes.

1.1.3. Resultados esperados

Al término del trabajo de tesis se tendrá el algoritmo correspondiente al control del robot para recolectar botellas de plástico usando visión artificial y la RCNN, así como los componentes de los dominios mecánico y electrónico. En complemento, se tendrán las simulaciones donde se aprecie el funcionamiento del algoritmo con información propia, capaz de adaptarse a diferentes circunstancias.

1.1.4. Impacto del proyecto

El impacto del proyecto trasciende a diversos campos. A continuación, se enuncia el impacto más significativo en cada uno de ellos

- Impacto ambiental

En el medio ambiente, la tecnología resultante en la presente tesis servirá de base para desarrollar productos que tendrán un impacto muy positivo dado que reduce el alto índice de contaminación por botellas plásticas en nuestro país, de esta manera se brinda una

solución eco- amigable.

- Impacto tecnológico en la ingeniería

En el sector de tecnología en la ingeniería habrá un claro impacto dado que al seguir una metodología pensando en el usuario como lo es “design thinking”, se mejorarán los lazos que existen entre la tecnología y sectores de la población no comúnmente familiarizados a la misma, lo cual aumentará la difusión de la ingeniería y lo que se puede realizar. Además, este proyecto busca consolidar la base de proyectos con Inteligencia Artificial para futuras investigaciones, de esta forma se impulsa la tecnología en el ámbito local e internacional que utilice esta tecnología o similares.

- Impacto social

El impacto social que ocasionará tener visión artificial en este robot, se verá reflejado en la toma de conciencia por parte de las personas de no arrojar botellas de plástico en las playas, ya que mientras el robot recolecte las botellas, las personas buscarán comunicar la noticia con sus comunidades, generando difusión y sembrando conciencia.

1.1.5. Metodologías para alcanzar los objetivos

Con la finalidad de alcanzar los objetivos propuestos, se harán uso de diversas metodologías que se describen a continuación. De esta forma, se tienen más enfoques sobre el trabajo de tesis desarrollado.

- Metodología Modular:

Se separará el diseño total en módulos, de tal manera que se pueda trabajar aplicando ingeniería concurrente en cada módulo y al acoplarlos se logre el resultado esperado. Los módulos están conformados por el dominio de control, mecánico y electrónico.

Hacer uso de esta metodología, también permite identificar fallas más fácilmente, dado que se puede localizar en que dominio se encuentran y al repararlas, no se ven involucrados los otros dominios protegiendo así el funcionamiento integral del robot y los componentes que lo conforman.

Como se menciona en el artículo desarrollado por el Dr. Marcos Echevarría-Quintana, de la Universidad Politécnica de Cataluña [ECHEVARRÍA, 2015], cada vez se busca optar con más frecuencia por metodologías que permitan establecer vínculos entre las variables que se aplican al concepto de diseño y desarrollo del proyecto, para facilitar la comunicación y comprensión entre sistemas globales.

Es necesario establecer módulos en el proyecto a desarrollar, pues de esta forma se garantiza la independencia en funcionamiento entre los bloques definidos. En el artículo consultado se fortalece esta idea, pues desarrollar esta metodología permite reducir el tiempo de diseño del producto, así como involucrar a profesionales especializados en un módulo en específico. De esta forma, también se introduce el concepto de ingeniería concurrente, pues se realiza un avance en paralelo en las etapas que lo permitan y se aumentan los filtros de cada etapa para obtener un mejor control de estado.

- Metodología Social: "Design thinking"

Se buscará familiarizarse con los requerimientos del usuario que se expondrán en la problemática para brindar una solución amigable, muchas veces las soluciones desde la perspectiva de la ingeniería pueden ser eficaces, pero al no conectar con la población quedan obsoletas dado que no hay un grado de familiaridad con el producto. Al conocer su día a día, se podrá generar una apariencia amigable del robot y de fácil uso para que se logre el objetivo específico y su funcionamiento en general.

Si bien en ocasiones este ámbito no es una prioridad, la realidad es que puede ser crucial al decidir si un proyecto tiene futuro o no, pues si no impacta de manera positiva en la población, es probable que el producto no funcione al carecer del respaldo popular.

En el estudio realizado por Barrueta et al [BARRUELA, 2018], se analiza la importancia de conocer al usuario final y cómo se debe generar el desarrollo del proyecto en base a los requerimientos, pero no son requerimientos arbitrarios, sino que están en constante dinamismo conforme el usuario requiera realizar algún cambio. Un aspecto característico de esta metodología, es la no emisión de ideas preconcebidas. Esto significa que, al

abordar un problema, no se deben generar ideas al instante, sino que se realizan métodos más creativos como lluvia de ideas, más conocido como brainstorming. De esta forma, los desarrolladores comentan lo que llegue a su mente sin encasillarse en conceptos previos o soluciones previas que hayan podido ver a problemáticas diferentes, pues ello indicaría contradicción con la razón de ser de esta metodología.

- Metodología Técnica:

Con respecto a la metodología técnica, se emplearán los pasos listados a continuación en el orden establecido:

- Revisar códigos fuentes existentes para desarrollar la visión artificial
- Verificar la capacidad de multiprocesos en paralelo necesaria para elaborar el algoritmo.
- Hacer uso de un simulador para probar el algoritmo e interacción con el movimiento y posibles casos que puedan suceder con el robot. De esta manera, el método de prueba y error se realiza virtualmente sin dañar componentes reales y es más efectivo.
- Realizar un banco de imágenes propio, de esta forma se puede uniformizar el formato empleado y garantizar la autoría del trabajo.

1.2. Estado del arte

A continuación, se revisará el contexto de la contaminación nacional en los últimos años y los antecedentes de proyectos orientados a la sostenibilidad y reducir la contaminación. Se dará énfasis en los proyectos que emplearon tecnologías similares a las propuestas en este trabajo de tesis y la importancia de su aporte a la ingeniería y al medio ambiente.

1.2.1. Realidad y problemática nacional

En la actualidad, se tienen las siguientes estadísticas obtenidas del MINAM:

- En el mundo se utilizan 5 billones de bolsas plásticas al año, casi 10 millones de bolsas por cada minuto.
- Cada año se vierten hasta 8 millones de toneladas de plástico en los océanos.
- Según la Fundación Ellen MacArthur [MACARTHUR, 2019], si los actuales patrones

de producción y consumo de plástico permanecen, en 2050 habrá mucho más plástico que peces en el océano, aproximadamente 98% de aves habrán ingerido plástico y la basura marina perjudicará a 700 especies marinas. El 16% de especies afectadas por ingestión y enredamiento con basura marina plástica se encontrarán en peligro de extinción.

Anualmente se generan 310 millones de toneladas de residuos plásticos, con respecto a este hecho se puede enfatizar los siguientes puntos [MINAM, 2019].

- Las bolsas de plástico se han encontrado hasta en la cumbre del monte Everest, casquetes polares y lugares más profundos y escondidos del océano.
- Las bolsas plásticas suelen ser confundidas con medusas u otros alimentos por la fauna marina. En junio del 2018, apareció un lobo marino muerto en las costas de España, se encontró en su interior 32 kilos de bolsas plásticas, redes y un tambor.
- Los residuos plásticos como sorbetes o aros de plásticos de botellas de latas pueden lastimar físicamente a los animales, introduciéndose en su cuerpo. En el 2015 se hizo público un video en el que un grupo de biólogo extrajo un sorbete de 14 centímetros de las fosas nasales de una tortuga.
- Los micros plásticos son ingeridos por los peces confundiéndolos con alimentos, acumulándose en el animal y luego magnificándose cuando es ingerido por otros seres vivos, incluyendo al ser humano.
- A nivel mundial, el 50 % del total de residuos plásticos son plásticos de un solo uso.
- En 2017, Algalita Marine Research and Education [MINAM, 2020], descubrió unas islas de plástico, frente a las costas de Chile y nuestro país. Estimaron que tiene una superficie aproximada de 2.6 millones de kilómetros cuadrados, casi dos veces la superficie de Perú.

Con respecto a nuestro país, se puede encontrar las siguientes cifras obtenidas del diario La República [LA REPÚBLICA, 2018]:

- En promedio, se usan al año aproximadamente 30 kg de plástico por ciudadano.
- Al año se suman cerca de 3 mil millones de bolsas plásticas, casi 6 mil bolsas por cada

minuto.

- En Lima Metropolitana y el Callao se generan 886 toneladas de residuos plásticos al día, representando el 46% de dichos residuos a nivel nacional.

En primer lugar, se ahonda en estadísticas de la problemática, contexto y situaciones de la contaminación del plástico desde el punto de vista nacional. Según el Ministerio del Ambiente del Perú [MINAM, 2019], el plástico representa actualmente el 10% de todos los residuos que se generan en el país y carece de solución. Esta cifra continúa causando preocupación hoy en día, ya que el plástico no es biodegradable y, por lo tanto, permanece en la naturaleza durante muchos años. En el caso de las botellas de plástico, el tiempo que debe transcurrir para degradarse es entre 100 y 600 años. En la Figura 1.1 se puede apreciar como la contaminación por botellas de plástico está presente en diversos escenarios posibles, siendo el Rio Santa en Huaraz una prueba de lo mencionado.

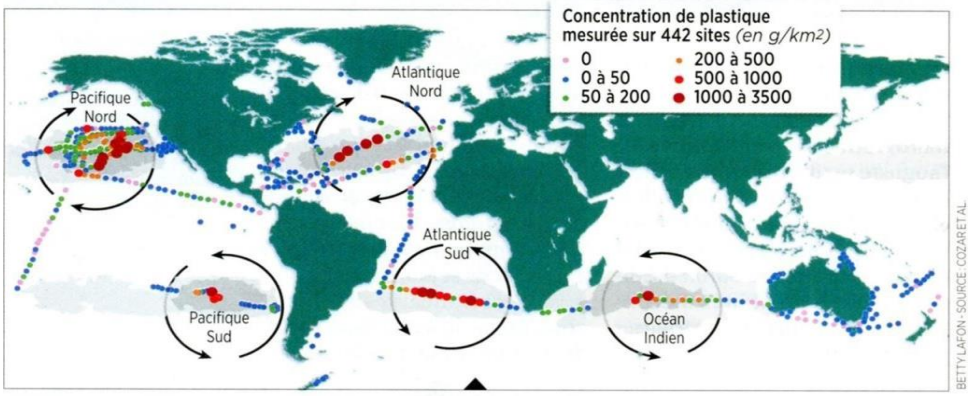
Figura 1.1. Contaminación del Rio Santa en Huaraz [LA REPÚBLICA, 2018]

Según el Ministerio del Medio Ambiente [MINAM, 2019], el 50% de los desechos plásticos del territorio nacional termina en vertederos dedicados a este material, el 46% termina en ríos y diversas playas, y el 4% es devuelto a un uso regular mediante las campañas de reciclaje. Esto, permite establecer que más de la mitad de la basura que se recolecta en las playas de lima es de material plástico.

Se sabe que los objetos plásticos son una de las principales causas de la contaminación marina, con 250 kg de plástico vertidos por segundo en diversos océanos del mundo. No solo dañan el ecosistema y fauna marina, sino que como se mencionó anteriormente, muchos desechos plásticos terminan acumulados en diferentes playas. Una muestra de esto, son las 230 toneladas de desechos plásticos que se encontraron en las costas del Océano Índico en el año 2017.

En el Perú, se viene tomando conciencia poco a poco sobre la importancia de mantener las playas libres de plástico. Por ejemplo, en febrero 2018, más de 150 jóvenes acudieron como voluntarios para limpiar las playas Agua Dulce y Pescadores en Chorrillos, mediante el programa "Yo Voluntario" de la Municipalidad de Lima, lo cual contribuye a un buen prospecto en las generaciones futuras. Tales iniciativas, buscan principalmente sensibilizar a la población acerca de las consecuencias de la contaminación. Sin embargo, pese a que se fomenta movilizar y sensibilizar a la gente, en particular a los jóvenes, no se tiene un impacto significativo en la limpieza de playas y en la reducción de la presencia de plástico en las mismas a corto plazo.

Hoy en día, en el mundo hay 5 movimientos oceánicos que conducen el plástico del mar a zonas de acumulación oceánica [FRANCEINFO, 2014], estas trayectorias pueden verse en la Figura 1.2.



Les plastiques flottants se retrouvent piégés dans les cinq gyres (vastes tourbillons) océaniques. En gris, les zones d'accumulation prévues par un modèle récent de circulation océanique.

Figura 1.2. Mapa de las zonas de acumulación de plástico en los océanos [FRANCEINFO, 2014]

Los lugares de depósito descritos previamente se incrementan mediante el flujo marino señalado en la Figura 1.3.

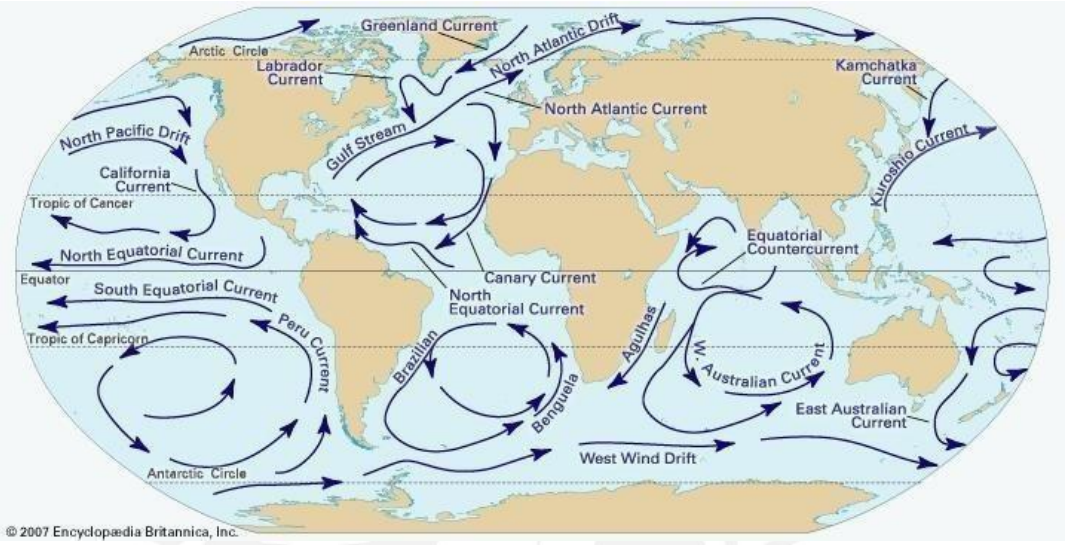


Figura 1.3. Mapa de las corrientes marinas [FRANCEINFO, 2014]

En consecuencia, según lo revisado, los residuos de plástico vertidos en la costa peruana se trasladan en mayor porcentaje al lugar de depósito del sur del Océano Pacifico, cuya ubicación se sitúa frente al litoral peruano, es decir en las playas de la costa nacional. En la Figura 1.4, se puede ver otra estadística sobre los residuos plásticos.

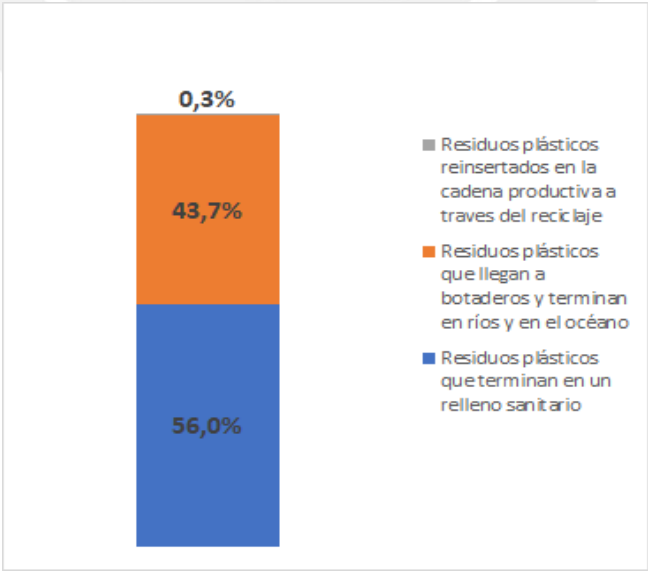


Figura 1.4. Gestión de los residuos plásticos en el Perú [MINAM,2019]

Pese a que el país se encuentra en un periodo tardío en el procedimiento de recolección, se han tomado decisiones que buscan sembrar conciencia y proponer un cambio. En diciembre del año 2018, el congreso de la República aprobó la ley que prohíbe el uso de bolsas de plástico en el territorio peruano: la "Ley de Plásticos" (Ley N°30884). Por lo tanto, los supermercados, los servicios de automóviles, los almacenes y los servicios en general tuvieron 36 meses de efectividad de la ley, para reemplazar la distribución de bolsas de plástico que no se pueden reutilizar por bolsas que se pueden reutilizar y/o descomponer. En el caso de las botellas de plástico, la ley otorgó un período de 36 meses para incluir el índice de reciclaje de 18 % de PET en la formación de la botella en sí.

Sin embargo, esto abarca un periodo extenso de cambio, y si a ello se le agrega que el índice de reciclaje planteado se mantiene en un nivel inferior en comparación a marcas principales, cuya política de reciclaje está por encima de este mínimo, se puede enunciar que existe una deuda con la naturaleza.

Dada la situación en Lima, cabe recalcar que según un informe del MINAM del año 2017, existen 6.8 millones de toneladas de residuos sólidos que se generan en el Perú anualmente, de los cuáles, la capital es responsable de más de 3 millones de toneladas por año [EL COMERCIO, 2018]. De esta clasificación, los residuos orgánicos son los más comunes, de los cuales un 54% se encuentra en la capital, seguidos por los residuos plásticos con 3500 toneladas anuales, que constituyen un 10%. Como es entendible, el nivel de acumulación de residuos es diferente en cada distrito de Lima y esto se debe principalmente a la diversa inversión económica y supervisión por parte de las autoridades locales y condiciones de vida de los ciudadanos tomando como base la educación y recursos económicos. Por ejemplo, según el diario La República [LA REPÚBLICA, 2018], los distritos con menores índices de calidad de ambiente en Lima son:

- San Juan de Lurigancho
- Chilca
- Villa María del Triunfo
- Carabayllo

Frente a lo descrito, existen diversos startups de reciclaje que le hacen frente a la contaminación causada por botellas de plástico, algunas organizaciones se muestran en la Tabla 1.1

Tabla 1.1. Organizaciones de reciclaje [LA REPÚBLICA, 2018]

	EMAUS	RECYCLEAN	RECICLA,PE!	MANCHAY
¿Qué?	Conocer y entender la importancia del reciclaje de los plásticos	Apoya la preservación del medio ambiente generando trabajo sostenible para todos sus colaboradores, clientes y proveedores	ONG peruana creada por las hermanas Daniela y Sandra Tagle que busca concientizar y educar a la población sobre el reciclaje de los plásticos PET y los beneficios que conlleva la protección del medio ambiente	Cuidar el medio ambiente y reciclaje
¿Cómo?	Campañas de reciclaje, publicaciones en páginas webs, redes sociales, aportes a través de diversos medios de comunicación	Comercialización de materiales reciclables	Brindarles las herramientas necesarias a los ciudadanos para poner en marcha una nueva era de reciclaje	Recoge todo tipo de residuos sólidos reciclables de las diferentes empresas que requieran servicio de recojo de sus residuos sólidos

Es importante recalcar que la problemática que se ha analizado es real y estas organizaciones pueden mostrar interés en un robot móvil que se encargue de acompañarlos en la tarea que ellos realizan desinteresadamente para el bienestar de la sociedad, llevando la tecnología y la ingeniería con humanidad y ayuda solidaria.

1.2.2. Tecnologías en la actualidad

En la actualidad, existen diversas tecnologías emergentes que proponen soluciones ante la contaminación ambiental. Estas tecnologías son empleadas por investigadores y desarrolladas en un ámbito académico y en ocasiones comercial. Algunos de estos proyectos se han desarrollado en Perú y otros en el extranjero, por lo que esta sección permitirá ver el aporte nacional e internacional de tecnologías compatibles con Inteligencia Artificial y sus características.

1.2.2.1.Contexto nacional

En el país existen diversos proyectos que buscan solucionar problemáticas relacionadas al medio ambiente que afectan a diversas comunidades. Entre estos proyectos, tenemos los

realizados por equipos de ingeniería de forma multidisciplinaria con otras especialidades y los realizados en su totalidad por el área de ingeniería. En ambas situaciones, los proyectos se realizan en beneficio de las personas y el medio ambiente de manera proactiva buscando constantemente una mejora en la calidad de vida. A continuación, se puede apreciar algunos ejemplos desarrollados en el pasado.

- **IRBin**

En la capital, se desarrolló un robot dedicado al reciclaje cuyo nombre es IRBin. En la Figura 1.5, se puede ver el módulo IRBin en desarrollo.

Figura 1.5. IRBin [ANDINA, 2019]

IRBin es un proyecto que surgió de un proyecto de la Pontificia Universidad Católica del Perú, donde la consigna era realizar un tacho inteligente. Así, los inventores de IRBin desarrollaron un módulo que permitía separar los residuos en tres diferentes clasificaciones: botellas plásticas, de vidrio y otros residuos. Se propuso que el módulo pueda situarse en locaciones de alta concurrencia de personas, como centros comerciales, supermercados, entre otros.

En una entrevista realizada por el portal Andina a Erich Carranza [ANDINA, 2019], el vocero de este startup comentó lo siguiente sobre el desarrollo de este robot: “Inicialmente, la idea era un tacho clasificador, pero luego nos dimos cuenta que eso era una máquina fría donde simplemente dejas los residuos y te vas. Pero, como la cultura de reciclaje en el país es nula, pensamos que teníamos que hacer algo que no solo clasifique, sino que de alguna manera enseñe a la gente. Ese es como el objetivo más difícil que

tenemos con nuestro proyecto. No lo hemos visto antes y nos pareció interesante aplicar el concepto de robótica social, sobre todo en una realidad como la de nuestro país donde simplemente no te educan en esta área."

- **Sullkapata: robot para limpieza y vigilancia de playas**

En los últimos años, la contaminación provocada por la generación de desechos y residuos, se ha convertido en un problema en la sociedad moderna. En este contexto, la playa ha sido la más afectada. Estas emisiones provocadas por el hombre representan el 80% de las mayores fuentes de desechos marinos en la actualidad. Además, tiene un gran impacto en el ecosistema porque lo destruye y afecta negativamente a la vida marina. Asimismo, afecta al público en general porque es perjudicial para la salud pública y son perjudiciales para el medio ambiente y la economía, siendo este último el resultado del declive de la industria turística. Los factores que contribuyen a esta contaminación incluyen una mala gestión y cultura de los desechos y la eliminación de otros residuos, principalmente en los países en desarrollo. Es por ello, que en un proyecto realizado por la PUCP y el área de ingeniería [PUNTOEDU, 2019] se diseñó un sistema mecatrónico que tenía como objetivo mantener las playas limpias, tener una supervisión activa sobre ellas, y así influir en visitantes de forma positiva para generar una mayor conciencia sobre el cuidado y disposición de los residuos. En la Figura 1.6 se puede apreciar el robot con la propuesta teledirigida.

Figura 1.6. Sullkapata [PUNTOEDU, 2019]

Este robot partió con la consigna de ser controlado manualmente mediante un control remoto externo de tal manera que se le pueda teledirigir para así recorrer las diversas playas sin colisionar con las personas ni con obstáculos de forma guiada.

1.2.2.2. Contexto internacional

En el contexto internacional, también existen proyectos de robótica e Inteligencia artificial como alternativas de solución para contribuir con la limpieza de playas y el medio ambiente. Asimismo, en este apartado se amplía el panorama de proyectos de carácter industrial, cuyo impacto académico ha permitido su ingreso al mercado; y proyectos de investigación, presentados como artículos provenientes de diversos países. Los aportes de estos proyectos son referentes tecnológicos para este trabajo de tesis por el impacto significativo que han tenido en su campo de estudio.

- **Robot limpiador de arena de playa Dronyx Solarino [DRONYX, 2018]**

Este robot que se muestra en la Figura 1.7 forma parte del estado del arte del sector comercial el cual está a la venta y su función principal es la de limpiar arena de playas en México.

Figura 1.7. Dronyx Solarino

El Robot Limpiador de Arena de Playa Dronyx Solarino es una combinación entre el robot móvil con seguimiento denominado XBOT y un brazo trasero agregado. Es el primer robot eco amigable orientado a la limpieza de playas en el país en el que fue desarrollado. Asimismo, es importante resaltar que es controlado de forma remota y cuenta con la capacidad de moverse tanto en terrenos arenosos secos como húmedos mientras elimina

la basura y otras materias extrañas. Es impulsado de forma eléctrica, alimentado por baterías GEL completamente aisladas y también por energía solar. Este producto proporciona una solución cómoda y ambiental para la limpieza de playas ya que es robusto y silencioso.

Sus especificaciones son las siguientes:

- Ancho de trabajo: 1300 [mm]
- Capacidad de trabajo: 3000 m2/hora
- Profundidad máxima de trabajo: 15 [cm]
- Distancia al suelo: 350 [mm]
- Peso: 500 [kg]

Su potencia de salida y batería viene dada por:

- Paquete de batería: 2 x 12 [V], 200 [Ah] batería AGM sellada Potencia total: 4800 [W]
- Autonomía estándar de la batería: 2-3 [Horas] Autonomía de batería de alta resistencia: 1,5 [horas]

Asimismo, sus dimensiones se pueden observar en la Figura 1.8

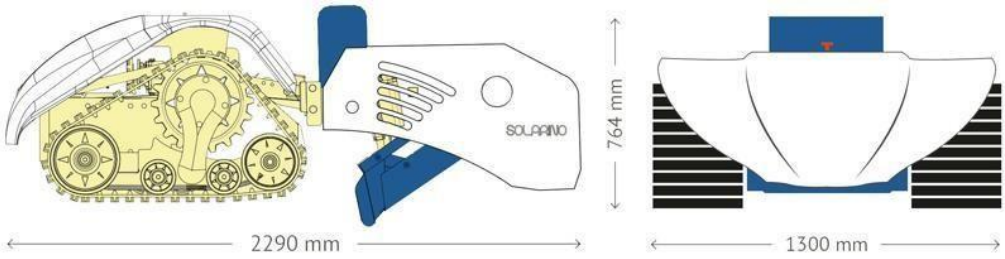


Figura 1.8. Dimensiones del Dronyx Solarino [DRONYX, 2018]

- **Robot inteligente limpiador de superficie recolector de basura [KONG, 2018]**

En este artículo, se ha desarrollado un sistema de robot llamado IWSCR para limpiar la superficie del agua para recoger basura plástica flotante. Es capaz de realizar tres tareas principales de forma autónoma, es decir, crucero y detección, seguimiento y dirección, y agarre y recolección. Los desafíos detrás de estas tareas implican cómo realizar la detección de basura precisa y en tiempo real, cómo resistir las perturbaciones mientras IWSCR realiza dirección basada en la visión y cómo agarrar la basura flotante de manera confiable a pesar de las condiciones turbulentas en la superficie del agua. Con el fin de superar estas dificultades, se proponen tres técnicas clave para IWSCR. Primero, la red YOLOv3, que se aplica ampliamente en el campo de detección de objetos de alta velocidad y precisión, está capacitado en el conjunto de datos de basura flotante propuesto para realizar una detección de basura precisa y en tiempo real. Luego, para mejorar la capacidad de resistencia perturbaciones, una ley de control basada en el controlador de modo deslizante se propone para la dirección basada en visión. Además, inspirado en La estabilidad de las botellas flotantes en el fluido, una estrategia de agarre factible se utiliza para IWSCR. Finalmente, los resultados experimentales demuestran que IWSCR es competente para llevar a cabo la tarea del agua. En la Figura 1.9 se observa la estructura del IWSCR.

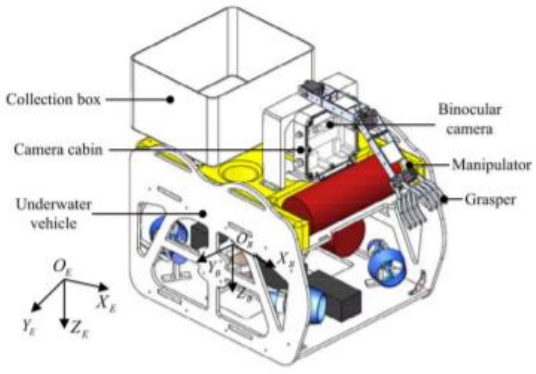


Figura 1.9. Estructura del IWSCR [KONG, 2018]

- **Sistema de agarre robótico basado en la visión que utiliza Deep learning para clasificación de basura [ZHIHONG, 2017]**

Este artículo propone un sistema de agarre robótico para clasificar automáticamente la basura en función de la visión artificial, el cual se puede apreciar en la Figura 1.10. El sistema logra la identificación y el posicionamiento de los objetos determinados como objetivo en un fondo complejo antes de usar el manipulador para agarrar automáticamente los objetos de clasificación.

La identificación de objetos respecto de un fondo complejo es el problema clave de la visión artificial y este algoritmo lo busca resolver. Este documento utiliza el método de Deep learning para lograr la identificación de autenticidad del objeto marcado como objetivo en el fondo complejo. En la Figura 1.10, se observa el esquema de control por orientación del robot.

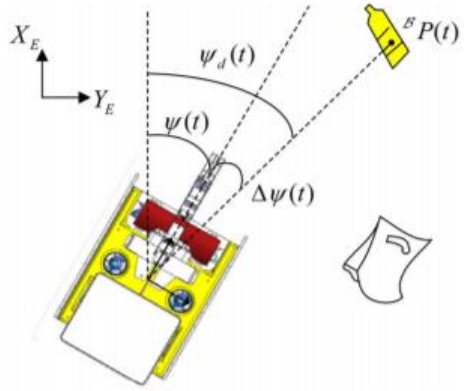


Figura 1.10. Control por orientación [ZHIHONG, 2017]

Para lograrlo, se aplica la generación propuesta de región (RPN) y el modelo VGG-16 para reconocimiento de objetos y estimación de pose. El sistema de visión artificial envía la información de las coordenadas geométricas centrales y el ángulo del lado largo del objeto de destino al manipulador que completa la clasificación y agarre del objeto objetivo. Los resultados del experimento de clasificación de las botellas en la basura muestran que el algoritmo de visión y el método de control del manipulador del sistema propuesto pueden lograr la clasificación de basura de manera eficiente. A continuación,

en la Figura 1.11 se puede apreciar la arquitectura interna sobre los módulos empleados y la perspectiva de objetos vistos desde la cámara donde se contempla el rango visual.

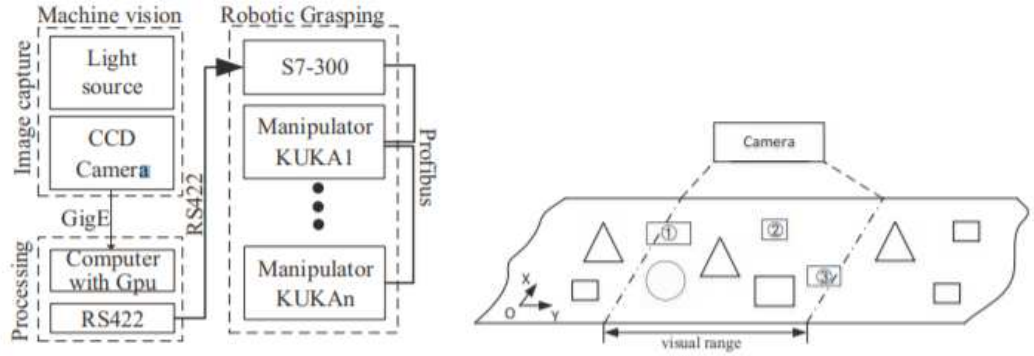


Figura 1.11. Arquitectura del procesador y cámara [ZHIHONG, 2017]

- **Mapa de profundidad de súper resolución para una cámara RGB-D rentable [TAKAOKA, 2015]**

Recientemente, cámaras rentables RGB-D que miden la distancia a los objetos enfocados se han utilizado en muchos campos. Haciendo uso de una cámara de profundidad equipada con la cámara RGB-D, se puede reconocer fácilmente la profundidad y los objetos tridimensionales que tienen dificultad en la distinción solo con imágenes RGB. Las cámaras de profundidad son de baja resolución en comparación con las RGB e incluyen errores de medición. Por lo tanto, en este caso los investigadores presentan un método para reducir los errores de medición de la cámara de profundidad y mejorar la resolución de un mapa de profundidad utilizando una imagen de cámara RGB de alta resolución.

Especialmente alrededor de los contornos del objeto, los mapas de profundidad de súper resolución se obtienen mediante un proceso de interpolación de los píxeles que selecciona la similitud de color y los píxeles de selección que almacenaron un valor de profundidad confiable. Luego, se conforman los mapas de profundidad para que se tenga la misma resolución de la imagen RGB, y son más similares a la forma de los objetos indicados por la imagen en color que solo el mapa de profundidad expandido. Además, se confirmó que el resultado mejora al aplicar este enfoque de la aplicación combinado con alta resolución de la imagen RGB.

En las Figuras 1.12 y 1.13, se pueden apreciar la estructura de pixeles y el proceso de segmentación de las imágenes respectivamente.

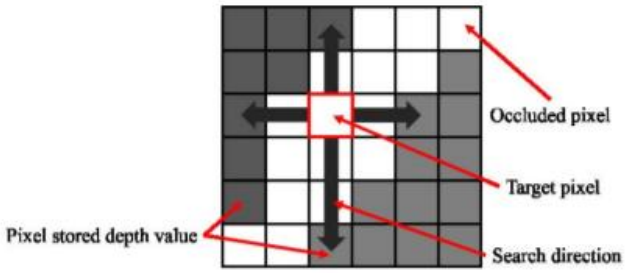


Figura 1.12. Estructura de pixeles [TAKAOKA, 2015]

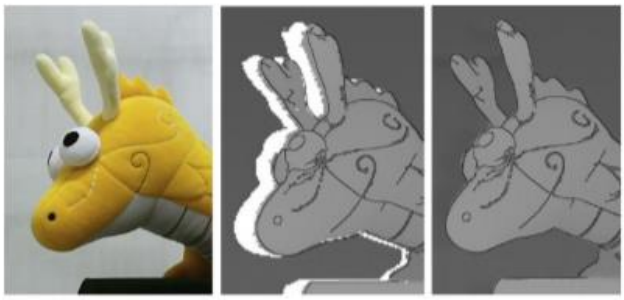


Figura 1.13. Segmentación [TAKAOKA, 2015]

En este artículo, como se comentó, se propuso un método de súper resolución de la cámara de profundidad utilizando una cámara a color de alta resolución. Mediante la selección de píxeles que pueden ser confiables alrededor del contornos del objeto, se logró la adquisición de la profundidad mapa cuyos contornos de objeto son similares a la imagen RGB. Se espera que se pueda mejorar este enfoque genéricamente combinando las cámaras RGB con mucho mayor resolución y mejora de algoritmos en los eventos futuros.

- **Uso efectivo de robots ligeros en estaciones de trabajo Humano-Robot con monitoreo vía RGB-D -Cámara [BOTHE, 2018]**

Se dice que la nueva generación de robots ya no requiere separación física entre humanos y robots. Para optimizar los procesos de producción, se espera que humanos y robots interactúen en conjunto. Este artículo describe la integración e implementación de

tecnologías destinadas a aumentar la flexibilidad y seguridad de la cooperación humano-robot. Además, un posible modelo para gestionar esta transición es descrito en detalle, que implica el uso de una cámara RGB-D.

Con esta cámara, los autores plantearon la posibilidad de detectar el estado y cambios de posición de las personas en una estación de trabajo humano-robot y consecuentemente adaptar los movimientos del robot. En general, el objetivo esencial de este trabajo es sugerir formas de aumentar eficiencia económica dentro de los procesos de ensamblaje. En las Figuras 1.14 y 1.15 se puede apreciar el entorno y características de trabajo en el espacio humano-robot y la arquitectura de conexiones respectivamente.

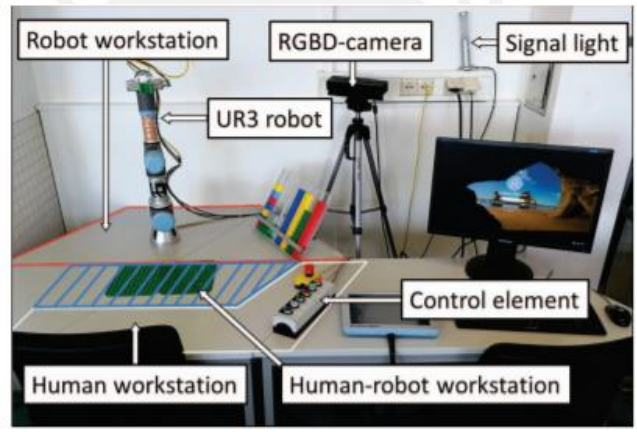


Figura 1.14. Arquitectura del procesador y estructura [BOTHE, 2018]

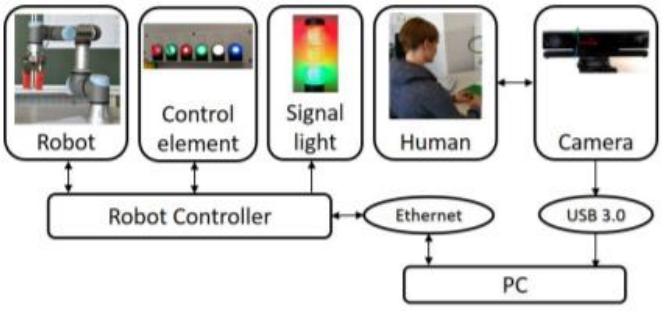


Figura 1.15. Arquitectura de conexiones [BOTHE, 2018]

- **Robot localizador basado en RGB-D en redes de alcantarillado [ALEJO, 2017]**

En la Figura 1.16 se aprecia el vehículo de alcantarillado y como los subdominios han sido implementados en el robot móvil.

Figura 1.16. Vehículo de alcantarillado [ALEJO, 2017]

Este artículo presenta un sistema de localización basado en el sistema de visión para la estimación de control de un robot de inspección de alcantarillado que tiene como objetivo corroborar la información previa dada de la red de alcantarillado por las instituciones locales. El sistema se basa en una localización en Montecarlo, es un sistema que utiliza edometría RGB-D para la etapa de predicción. La etapa de actualización tiene en cuenta la topología de la red de alcantarillado para descartar hipótesis equivocadas. Además, este paso es más refinado, cada vez que se detecta un elemento discreto de la red (es decir, boca de inspección) se repite de tal forma que se encuentre el error. Se utiliza otra cámara RGB-D apuntando hacia arriba para la detección precisa de alcantarillas. Una red neuronal convolucional ha sido entrenada con éxito para clasificar imágenes con y sin registros con 96% de precisión sobre el conjunto de datos probado. El sistema completo ha sido validado con datos reales obtenidos de las alcantarillas de Barcelona obteniendo resultados precisos de localización. En la Figura 1.17 se observan las alcantarillas mencionadas.

Figura 1.17. Alcantarillas estudiadas. [ALEJO, 2017]

- **Implementación de un aspecto basado en la apariencia en tiempo real y mapeo en un robot de búsqueda urbana totalmente autónomo [CHAN, 2018]**

Esta investigación describe los hallazgos del uso en tiempo real de un mapeo 3D, el mapeo basado en la apariencia en tiempo real (RTAB-Map) utilizado para un determinado proyecto. Viniendo de un diseño de plataforma robótica móvil controlado de forma inalámbrica capaz de navegar un entorno desconocido, se crea una reconstrucción 3D de dicho medio ambiente, al detectar fuentes humanas que podría encontrar mientras se realiza en tiempo real de la reconstrucción 3D del entorno dado.

El propósito de este proyecto fue ayudar a los operativos de rescate proporcionando un modelo del entorno generado por computadora y permitir el rescate de equipos para analizar y aprender el diseño general del área sin afrontar los peligros de entornos desconocidos, así como identificar la presencia de humanos, daño hecho y posibles efectos de reacción peligrosa. Se identifica el equipo para buscar y enrutar dando prioridad a buscar los sobrevivientes posibles. El proyecto se realizó utilizando una plataforma móvil. y una cámara RGB-D, a través de navegación autónoma o control manual, para generar una representación gráfica del área transversal a la plataforma móvil y ejecutar un algoritmo de detección de rostros para dar aviso urgente a los operadores ya sea por señal de audio o sonido del video. En las Figuras 1.18 y 1.19 se puede ver el módulo de cómputo y el entorno modelado en 3D.

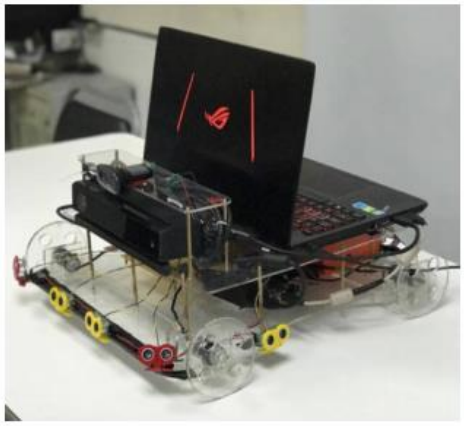


Figura 1.18. Módulo de cómputo [CHAN, 2018]

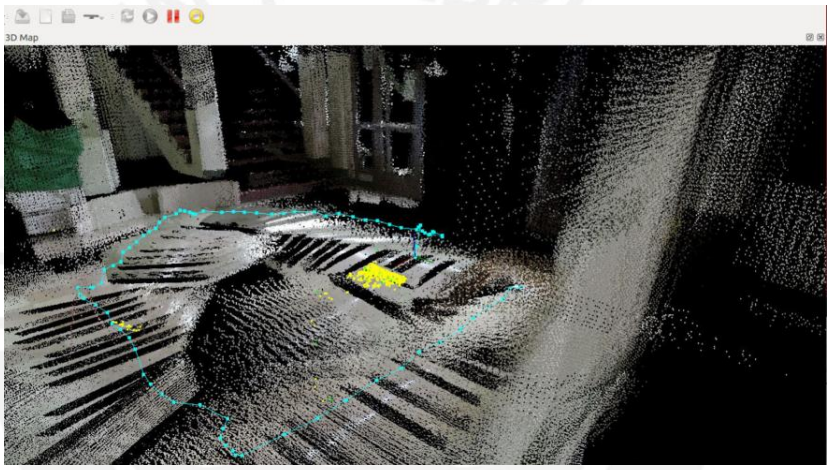


Figura 1.19. Entorno 3D modelado [CHAN, 2018]

- **Calibración volumétrica y registro de sensores RGB-D [BECK, 2015]**

En este artículo, el autor presenta una investigación integrada que busca la calibración y registro de sensores de color y profundidad (RGB-D) para un sistema coordinado sin realizar una identificación explícita de parámetros intrínsecos y extrínsecos de una cámara. En la Figura 1.20 se evidencia ello y sus respectivas etapas.

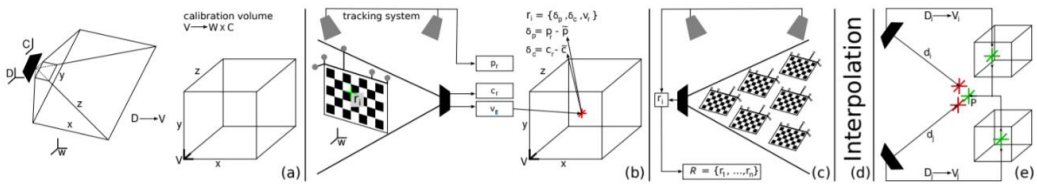


Figura 1.20. Camará RGB-D y etapas [BECK, 2015]

Se midió la precisión y los efectos del muestreo de referencia escaso y denso y se calibró el Kinect V2 en el sistema de coordenadas conjuntas para cubrir un volumen de captura de aproximadamente 1,5 mx 1,8 m. x 1,5 m. La calibración inicial se realizó con OpenCV. A los autores, les tomó aproximadamente 2000 muestras de referencia y las dividieron en tres conjuntos: Reval de tamaño 1000 para evaluación, Rdense de tamaño 1000 y Rsparse de tamaño 500, un subconjunto de Rdense.

Los conjuntos Rdense y Rsparse luego se usaron como entrada para la interpolación. Se buscó estimar el error en términos de la distancia absoluta a la verdad del terreno. En el Error 3D, la verdad fundamental es el sistema de seguimiento; mientras que en el error 2D, la verdad del terreno es el punto de cruce detectado en el espacio de la imagen en la muestra de referencia ri ∈ Reval. Los resultados para NNI, así como IDW para diferentes barridos k (5, 10 y 20) y para diferentes resoluciones del volumen de calibración, se enumeran en la tabla que se ha recopilado de dicho artículo. La evaluación muestra claramente que el método desarrollado es capaz de lograr alta precisión. Se escala con la densidad del conjunto de referencia (Rdense vs Rsparse) y con la resolución del volumen de calibración. En la Tabla 1.2, se ilustra más lo referido.

Tabla 1.2. Camará RGB-D y etapas [BECK, 2015]

Method	3D dense	3D sparse	2D dense	2D sparse
Initial	35.7 (13.0)[75.0]	-	24.2 (1.0)[28.0]	-
IDW5	3.2 (2.1)[14.0] 3.2 (2.1)[13.6]	4.2 (2.9)[18.5] 4.1 (2.9)[17.0]	0.3 (0.2)[1.1] 0.3 (0.2)[1.9]	0.3 (0.2)[1.2] 0.3 (0.2)[1.2]
IDW10	3.1 (2.0)[14.7] 3.1 (2.1)[13.9]	4.3 (2.8)[16.0] 4.2 (2.8)[15.8]	0.3 (0.2)[1.2] 0.3 (0.2)[1.2]	0.3 (0.2)[1.2] 0.3 (0.2)[1.2]
IDW20	3.0 (2.0)[17.7] 3.0 (2.1)[16.5]	4.5 (2.7)[15.8] 4.4 (2.8)[16.5]	0.3 (0.3)[5.0] 0.3 (0.2)[1.0]	0.4 (0.2)[1.1] 0.4 (0.2)[1.1]
NNI	1.7 (1.0)[5.0] 1.7 (1.1)[5.8]	2.0 (1.2)[5.7] 2.0 (1.3)[6.9]	0.2 (0.2)[1.5] 0.2 (0.2)[1.3]	0.3 (0.2)[1.5] 0.3 (0.2)[1.9]

- **Investigación teórica y experimental sobre ruedas de arrastre y rendimiento del robot móvil con ruedas en arena suelta [LIU, 2011]**

Basado en la teoría del equilibrio límite, se llevó a cabo una investigación para analizar la deformación de la arena suelta y la fuerza de contacto entre las orejetas y la arena suelta. Así como la relación existente entre ellas. Luego, para el modelo teórico se buscó hacer coincidir razonablemente la orejeta y se estableció el espaciado y la altura. En la Figura 1.21, se muestra el modelo teórico representado de forma gráfica.

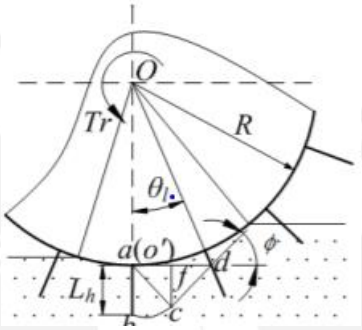


Figura 1.21. Modelo teórico ilustrado [LIU, 2011]

Mientras tanto, el efecto de análisis de los parámetros de las orejetas sobre el rendimiento del movimiento, sobre el rendimiento de tracción y maniobrabilidad de la dirección de tacos rueda, se llevó a cabo en un recipiente de arena suelto. Todas las pruebas se realizaron a hundimiento libre de la rueda y deslizamiento del 0 al 60% en un banco de pruebas de una sola rueda. A través del análisis cualitativo y comparaciones de movimiento. El rendimiento entre la rueda lisa y las ruedas con diferentes tacos, los resultados mostraron que la altura del taco y el deslizamiento influyen significativamente en el rendimiento del movimiento de la orejeta y el espacio. Por el contrario, el grado de efecto del parámetro de orejeta en el rendimiento de tracción y la maniobrabilidad de la dirección son opuestos.

Luego de aplicar los índices de evaluación del rendimiento del movimiento analizando los datos experimentales, los resultados indicaron que el parámetro de tacos preferibles para la rueda suave experimental con un radio de 135 mm son espaciamiento de orejetas de 15°, altura de orejetas de 15 mm, que es consistente con el resultado teórico derivado de

la coincidencia razonablemente el espaciado y la altura de las orejetas, el valor óptimo del deslizamiento de las ruedas es del 13% desde la perspectiva de ahorrar energía al emplear arena suelta similar en el experimento. En la Ecuación 1.1 se observa el comportamiento del ángulo, radio y otros parámetros [LIU, 2011]

$$\frac{tg(\theta_l)}{L_h} = \frac{2}{R} e^{(45+\frac{\varphi}{2})} tg\varphi \cos\,(45 - \frac{\varphi}{2}) \tag{1.1}$$

Cuando la rueda con tacos rueda hacia delante sobre arena suelta, los tacos empujan la arena y la empujan hacia atrás. La resistencia al cizallamiento La resistencia al cizallamiento y la capacidad de carga de la tierra vegetal son tan débiles que la partícula bajo la rueda motriz suele estar en estado fluido. estado fluido. La arena puede considerarse un material elástico-plástico sobre en base a la teoría del equilibrio límite.

De los artículos científicos y proyectos revisados en el estado del arte, se afirma que en la actualidad la Inteligencia Artificial se emplea para diversos propósitos. Por ejemplo, reconocer cuerpos de objetos en el espacio, recrear ambientes, monitorear en tiempo real objetos, entre otras novedosas aplicaciones. Asimismo, en los artículos descritos, se puede apreciar que un elemento en común es el uso de una cámara RGB-D para aplicaciones que emplean visión artificial. En los últimos años, se ha marcado una tendencia en incorporar esta cámara a proyectos que emplean IA, pues permite obtener la distancia a la que se encuentra el objeto de estudio cuya imagen ha sido capturada, y por facilitar la identificación de objetos mediante visión artificial.

Las diferencias que presentan los proyectos son a nivel de contexto y de aplicación específica. Sin embargo, el esquema de capturar elementos y aplicar IA para el procesamiento e identificación de datos, permite que estos proyectos formen parte del mismo universo pese a tener distinta motivación. Esto permite tener un enfoque global de como orientar el trabajo de tesis para fomentar la investigación y aportar al universo mencionado.

1.3. Fundamentos de visión artificial.

Según la Automated Imaging Association (AIA), la visión artificial abarca todas las aplicaciones industriales y no industriales en las que una integración de hardware y software brinda una guía operativa a los dispositivos en la ejecución de sus respectivas funciones acorde a la captación y procesamiento de imágenes para un posterior accionamiento, ya sea autónomo o manual.

Al eliminar el contacto físico entre el sistema de prueba y los componentes por usar, la visión artificial previene el daño de los componentes y elimina el tiempo de mantenimiento y los costos asociados con el desgaste mecánico. La visión artificial proporciona beneficios operativos y de seguridad adicionales al reducir la participación humana en la producción y el manejo. Además, previene la contaminación humana de salas limpias en un ambiente estéril y protege a las personas de ambientes peligrosos.
La visión artificial ayuda a alcanzar objetivos estratégicos como se puede apreciar en la Tabla 1.3.

Tabla 1.3. Objetivos y aplicaciones IA

Objetivo	Aplicaciones
Incrementar la calidad	Inspeccionar, medir, calibrar y verificar la integración.
Incrementar la productividad	Las tareas cotidianas realizadas manualmente, ahora son más sencillas con uso de visión artificial.
Reducir el tiempo de inactividad y configuración de las máquinas	Mantenimiento y plan de cambios.
Tener mayor control de procesos rigurosos y precisos	Obtención de datos por ordenador para generar informes.
Reducir los gastos del presupuesto interno	Emplear visión artificial favorece la supervisión de una máquina, evitando así que se dañe y no realizar un gasto innecesario en una nueva compra.
Reducir la tasa de productos defectuosos	Precisión y medición de productos usando IA.
Realizar inventarios sin factor humano	Uso de la visión artificial para tomar fotos de los productos y procesar los datos para obtener el total.
Reducir espacio	Uso de una máquina moderna en lugar de una antigua, que a su vez es más grande y no brinda movilidad.

Los elementos fundamentales de un sistema de visión son la luz, la lente óptica, los sensores, el procesamiento de imágenes y la transmisión de información. Primero, se debe iluminar la parte que se requiere examinar para captar los detalles de forma clara mediante la cámara. Luego, se captura la imagen empleando la lente y se expone al sensor en forma de luz. De esta forma, el sensor de la cámara convierte esta luz en una imagen digital que es enviada al procesador donde se analiza. El procesamiento de la visión implica desarrollar algoritmos que se encarguen de escanear la imagen, extraer las características correctas, diseñar las pruebas necesarias y tomar decisiones. [COGNEX, 2018]

Durante el desarrollo de estos algoritmos se estudiaron diversas opciones para que el robot móvil pudiera moverse constantemente por las orillas, identificando botellas de plástico. Finalmente, el flujo de información suele circular mediante señales discretas de entrada y salida, o trama de información enviada mediante una conexión serial al dispositivo de registro de información para almacenamiento inmediato o uso en pseudotiempo real.

La mayoría de los componentes de hardware de los sistemas de visión artificial, como unidades de iluminación, sensores y procesadores, son productos disponibles comercialmente (COTS) [COGNEX, 2018]. Los sistemas de visión artificial pueden fabricarse a partir de productos COTS o comprarse como un sistema integrado con todos los componentes en una sola máquina. En la Figura 1.22 se puede apreciar el esquema de visión artificial y sus etapas.

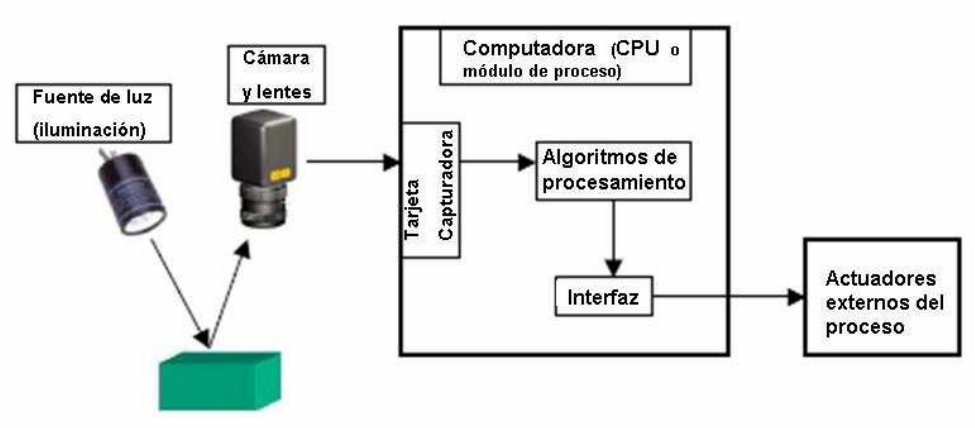


Figura 1.22 . Esquema de visión artificial

Se dice por lo descrito que la visión artificial es la extracción de información a partir de imágenes digitales de un proceso o control de calidad. La mayoría de los fabricantes utilizan la visión artificial automatizada en lugar de probadores humanos porque responden mejor a las tareas de prueba repetitivas [COGNEX, 2018]. Es más rápido, más objetivo y funciona de forma continua. La visión artificial puede inspeccionar cientos o incluso miles de piezas por minuto, proporcionando los resultados de inspección más consistentes y confiables las 24 horas del día, los 7 días de la semana. Cuando los trabajadores pueden reducir el rendimiento debido a la fatiga normal.

La teoría aplicada en los antecedentes vistos, sugieren las siguientes consideraciones:

- Debido a que emplear visión artificial es un concepto que va muy ligado con la continuidad del movimiento y la calidad de las imágenes, se ha visto por conveniente realizar algoritmos que permitan hacer uso de la visión artificial en condiciones reales en las playas para detectar las botellas de plástico y así recolectarlas.
- Se deberá analizar todo el ambiente que rodea las botellas de plástico e identificar los componentes que se han descrito previamente, de esta manera la iluminación promedio que se encuentre en las playas, así como el terreno y la estabilidad que existirá en el robot, son factores que sin duda definirán que tipo de software emplear, así como los sensores, actuadores y la cámara mediante la cual se tomaran las imágenes para poder usar visión artificial en tiempo real.
- La solución de un robot móvil que emplea visión artificial para la limpieza de playas identificando y recolectando botellas de plástico, es eficiente, dado que cuando se tienen problemas de índole ambiental, una medición fundamental de eficacia es si el remedio no es peor que la enfermedad. Es decir, que la solución en este caso el robot, no perjudique más al medio ambiente de lo que lo ayuda. Por ejemplo, si el robot consumiese mucha energía o si tuviera que ser tele-operado constantemente, para ser dirigido quitando el tiempo a la persona encargada, entonces no se tendría una solución sino una ligera mejora al problema. Con lo descrito, se desarrolla una solución sin efectos secundarios.

En los anexos A y B ubicados en el apartado de anexos, se enuncian dos algoritmos fundamentales como parte de los fundamentos contemplados para la visión artificial. El

primero de ellos enuncia un filtro Gaussiano que permite obtener una sección de la imagen a la entrada, este algoritmo es fundamental para el diseño de algoritmos más complejos y personalizados. El segundo algoritmo desarrolla la obtención de coordenadas especificas tras pasar por el filtro de Gauss.

CAPÍTULO 2

MARCO TEÓRICO

En este capítulo, se explora el marco teórico; es decir, los conceptos y fundamentos teóricos necesarios para el desarrollo del diseño del robot móvil, así como las tecnologías existentes relacionadas a Inteligencia Artificial. Esto se realiza, con el fin de elaborar el algoritmo para la red RCNN que permita la identificación y reconocimiento de las botellas de plástico. En complemento, se abordan conceptos de índole mecánico, electrónico y de control para definir las tecnologías a emplear en la propuesta de diseño de estos dominios en el capítulo posterior.

2.1. Introducción a la Visión Artificial

La visión artificial tiene como objetivo extraer información del mundo que nos rodea utilizando computadoras que recopilan, procesan, analizan y comprenden imágenes.
El sistema de visión artificial representa la realidad y proporciona información sobre brillo, color, forma, etc. Estas representaciones, suelen incluir imágenes en movimiento, escenas en 3D o imágenes fijas.

La imagen bidimensional, se puede expresar en función del hecho de que cada par de coordenadas (x, y) tiene un valor correspondiente relacionado con alguna característica del punto que representa (por ejemplo, grados de luz o color). Del mismo modo, se sabe que una imagen cromática es una imagen que no aporta ninguna información sobre formas, es una imagen en la que a cada punto solo se asocia información relacionada con la luminancia, y se puede representar como una superficie donde la altura de cada punto indica su nivel de brillo. Finalmente, una imagen a color se puede representar en formato RGB asociando un conjunto de tres valores a cada punto, que representa las intensidades de las tres luces (una roja, una verde y una azul). Finalmente, se puede representar una imagen espectral completa asociando un espectrograma de emisión de color a cada punto.

2.1.1. Definiciones

En este apartado, se revisan algunos conceptos fundamentales de física para comprender la concepción de ciertas metodologías de diseño al desarrollar e implementar sistemas de visión artificial en informática y robótica.

En general, desde la perspectiva del procesamiento digital de imágenes, es necesario definir la luz como una onda y no como corpúsculo. Según el modelo ondulatorio, las características de un haz de luz consisten en amplitud y longitud de onda. Pese a que las ondas de luz constituyen un pequeño porcentaje de la clasificación de ondas electromagnéticas, con frecuencia son de especial interés debido a que se captan mediante los ojos y son procesadas por el cerebro, debido a que el ojo humano es capaz de diferenciar entre 400 y 700 nanómetros (nm.) [PERTUZ, 2017].

El sistema sensorial visual, interpreta las diferentes amplitudes y longitudes de onda de la luz, produciendo las sensaciones que se conocen como brillo y color respectivamente. Es decir, no se tratan de propiedades intrínsecas abstractas, sino que son realidad, ya que son captadas por los seres humanos como se ha descrito. Como se puede observar en la Figura 2.1, la parte de la radiación electromagnética que constituyen las ondas luminosas abarca desde el fin del espectro ultravioleta (400 nm) hasta el comienzo del espectro infrarrojo (700 nm).

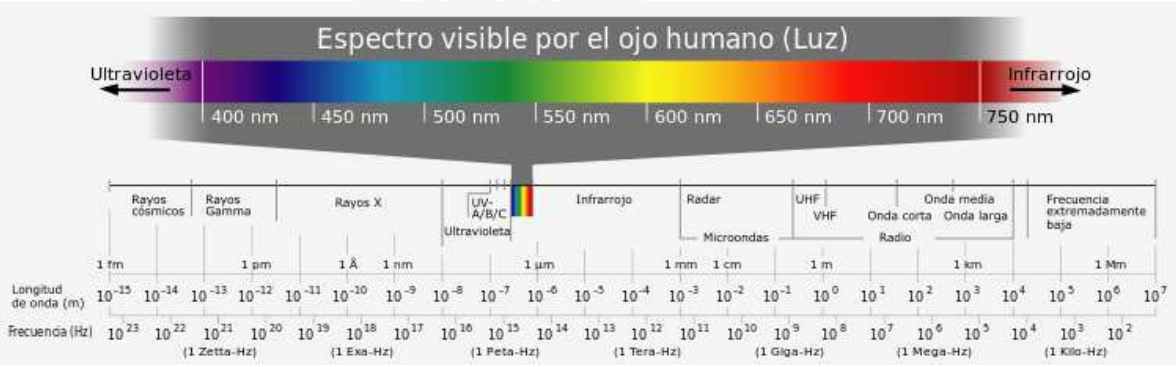


Figura 2.1. Espectro visible por el ojo humano [PERTUZ, 2017]

- **Distribución espectral de energía (DeP)**

Una curva de distribución espectral de energía, representa la cantidad de energía (en watts) asociada a cada longitud de onda en un espectro de radiación electromagnética. Si se ilustra el diagrama espectral de una radiación electromagnética que posee una longitud de onda específica, y se obtiene un gráfico con un pico en la longitud correspondiente y 0 en el resto se dice que es una luz monocromática, se entiende como si estuviera sincronizada a dicha longitud de onda. En general las radiaciones no suelen ser tan puras y resultan de la mezcla de diferentes longitudes de ondas con diferentes haces [LLINARES, 2019]. Además, entre más monocromático sea un haz de luz, tendrá menos energía asociada, por lo que será más difícil percibirlo, es por ello que los diagramas espectrales que se encuentran en la naturaleza son más parecidos a los que se presentan por ejemplo en las curvas de emisión de la Figura 2.2.

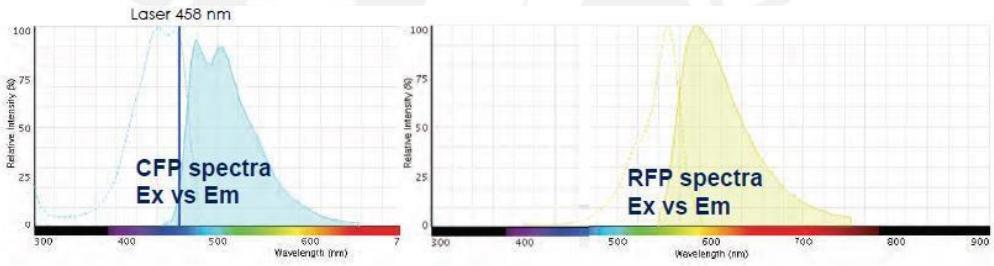


Figura 2.2 Distribución espectral de energía [LLINARES, 2019]

- **Flujo luminoso**

El flujo luminoso es la parte del flujo de luz radiante que detecta el ojo. La unidad de flujo luminoso es el lumen (L) [YUJILEDS, 2013]. El lumen, corresponde al flujo luminoso que procede de una abertura de 1/60 $cm2$ en un cilindro de material refractario que contiene un material estándar que irradia a través de un cono, la radiación del estereorradián. El flujo de luz, se puede medir con un fotómetro y se caracteriza por el símbolo Φ. En la Figura 2.3 se puede apreciar la medición descrita.

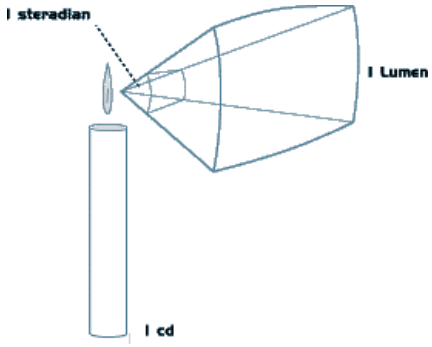


Figura 2.3. Medición de luminosidad [YUJILEDS, 2013].

- **Luminancia o brillo**

La luminancia o brillo de una fuente de luz es la intensidad luminosa por unidad de superficie [YUJILEDS, 2013]. Así, por ejemplo, si se dividen secciones de áreas iguales y se mide el brillo de una estrella y de un foco, se aprecia que el foco tiene un brillo de más intensidad. Sin embargo, la intensidad luminosa de cualquier objeto de nuestro entorno es menor que la de la estrella, pues la intensidad luminosa no depende de la distancia.

- **Saturación**

La saturación mide cuantitativamente la relación entre la longitud de onda predominante y las demás longitudes de onda existentes. Se ve un color específico cuando se percibe una combinación particular de matiz, saturación y brillo, cabe recalcar que deben estar presentes los elementos mencionados.

- **Histograma**

El histograma de una imagen es un gráfico que muestra la intensidad del color de dicha imagen y su clasificación por niveles. Esto se realiza con respecto a la cantidad de píxeles existentes para cada nivel.

2.1.2. Etapas de un sistema de visión artificial

El ser humano captura la luz a través de los ojos, y esta información es dirigida a través del nervio óptico hasta el cerebro donde se procesa, y existen razones para creer que el

primer paso de ese procesamiento es descomponer la imagen en elementos más simples como segmentos y arcos. La visión artificial busca reproducir este comportamiento y normalmente cuenta con cuatro fases principales [MA, 2002]:

- Captura o adquisición: Se adquieren las imágenes mediante un sensor de un tipo establecido según la aplicación.

- Tratamiento digital de las imágenes: Esta fase consta de procesamiento previo y tiene por objeto facilitar las etapas posteriores, consiste en eliminar las partes indeseables o realzar partes interesantes específicas de la imagen mediante filtros y transformaciones.

- Segmentación: Consiste en aislar los elementos que interesan de una escena para comprenderla de mejor manera, eliminando los componentes distractores.

- Reconocimiento o clasificación: En esta etapa se pretende distinguir los objetos segmentados, gracias al análisis de ciertas características que se establecen previamente para diferenciarlos.

Es importante resaltar que estas cuatro etapas no se emplean de forma secuencial, pues es necesario que se retroalimenten entre sí para asegurar que el sistema funcione óptimamente. Por ello, es válido volver a la etapa de segmentación si la etapa de reconocimiento presenta algún fallo o si existen errores en la etapa de preproceso, o incluso a veces es necesario volver a la etapa de captura cuando falla alguna de las etapas que aparentemente vienen después. Esto se ilustra en el diagrama de Figura 2.4.

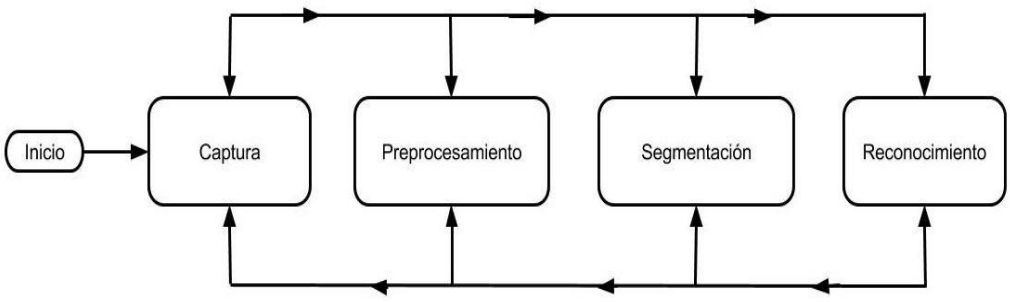


Figura 2.4. Etapas de procesamiento [MA, 2002]

Adicionalmente, existen otras etapas que también son importantes. Estas etapas están conformadas por:

- Adquisición y representación de imágenes digitales: En esta etapa se contemplan conceptos relacionados a la adquisición de la imagen del mundo físico real y su paso al dominio discreto abstracto y virtual informático.
  Cuando se digitaliza una imagen bidimensional, ésta resulta constituida por un conjunto de elementos llamados pixeles. Cada píxel ofrece información acorde a región fundamental de la imagen. En imágenes donde solo hay niveles de gris esta información es conocida como brillo.

- Captura y digitalización: Las imágenes digitales son "señales" discretas cuyo origen suele situarse en una "señal" continua. Por ejemplo, una cámara digital captura imágenes del mundo real que es continuo tanto en tiempo como en espacio.
  En el proceso de obtener imágenes digitales se establece la clasificación en dos etapas, la primera se conoce como captura, emplea un dispositivo, generalmente óptico con el que se obtiene información relativa a una escena.
  En la segunda etapa, también conocida como digitalización, se transforma dicha información en una imagen digital, que es una señal con todas sus componentes discretas, en otras palabras, es el proceso de paso del dominio continuo (o analógico) al dominio discreto (o digital). En la digitalización resaltan dos procesos: el muestreo y la cuantificación.

- Muestreo: El muestreo de una señal continua se basa en su medición por intervalos tomando como referencia una variable determinada (generalmente el tiempo o el espacio), con lo cual se obtiene su parámetro fundamental que es la frecuencia de muestreo. Esta representa el número de veces que se mide un valor analógico por unidad de cambio. El muestreo permite convertir una imagen IC, del dominio continuo, en una matriz discreta ID de NxM pixeles de dominio digital. La cantidad de muestras por unidad de espacio sobre el objeto original define el concepto de resolución espacial de la imagen. Ésta es establecida como la distancia real que representa un píxel. En la Figura 2.5 se puede apreciar el muestreo aplicado de diversas formas en las etapas descritas.

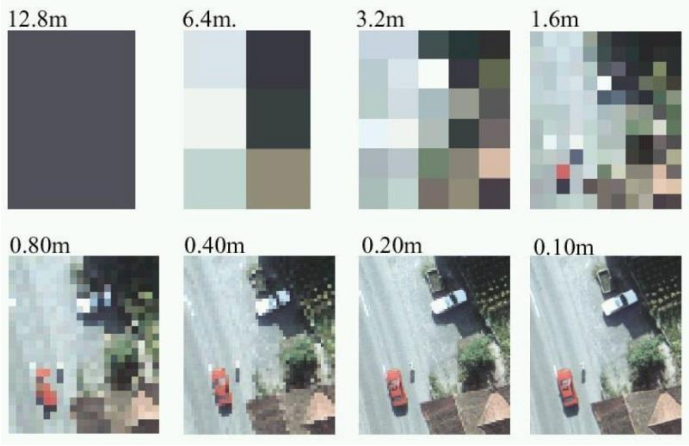


Figura 2.5. Procesamiento de la imagen [MA, 2002]

Así, el proceso de muestreo de una imagen asocia un valor real a cada punto, y cambia la imagen al formato: $IC\ (x, y) \in \Re$ donde $x, y \in \Re$ / $D\ (x, y)$ donde $x, y \in N$ y 0 $x \leq$ - 1, 0 $\leq y \leq M$ - 1

- Cuantificación: Consiste en asignar un valor a cada píxel. El nivel de cuantización es generalmente una potencia de 2 para facilitar el almacenamiento de la imagen en su computadora. De esta forma, $ID\ (x, y) \in \Re$ se convierte en $IDC\ (x, y) \in N$ (una imagen cuantitativa separada).

La precisión radiométrica determina el número de niveles posibles en cada caso. Cuando las imágenes contienen solo información de luminancia, se denominan imágenes en escala de grises y normalmente contienen 256 niveles que se utilizan para representar los tonos intermedios del negro (0) al blanco (255). En los casos en que solo se permitan dos niveles de cuantificación (normalmente blanco y negro), se utilizará el concepto de imágenes binarias o bitonos. En el caso de los colores, comúnmente se utilizan 256 niveles para representar la intensidad de cada uno de los tres colores primarios (RGB). De esta forma se obtienen unos 16 millones de colores y se hace referencia a imágenes en color real.

2.2. Conceptos requeridos para el desarrollo del proyecto

En esta sección, se hará una revisión los conceptos necesarios para el desarrollo del proyecto, la importancia de seleccionar de manera adecuada cada componente se basa en conocer la base teórica. La definición de los productos comerciales descritos es extraída de lo proporcionado por cada fabricante.

2.2.1. Cámara RGB

La cámara RGB es una cámara equipada con un sensor CMOS estándar a través del cual se obtienen imágenes en color de personas y objetos [MA, 2002]. En la Figura 2.6 se puede apreciar las etapas de una imagen capturada por una cámara de este tipo.

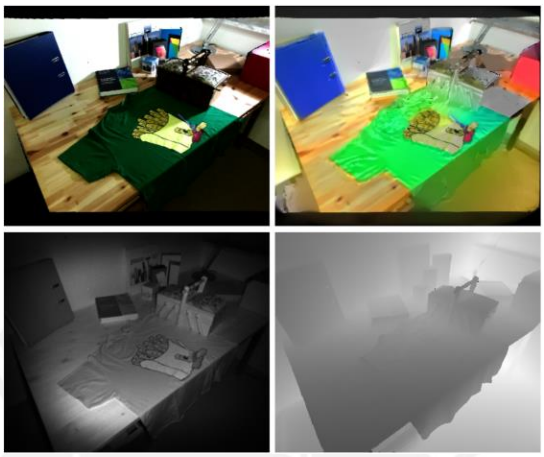


Figura 2.6. Etapas de la cámara RGB [MA, 2002]

La adquisición de fotos estáticas generalmente se expresa en megapíxeles (por ejemplo, 12MP, 16MP) que definen la cantidad de píxeles (es decir, longitud x altura) que componen una foto. Mientras que la adquisición de videos generalmente se expresa con términos explicativos como Full HD (es decir, 1080 x 1920 píxeles con 30 cuadros por segundo) o Ultra HD (es decir, 3840 x 2160 píxeles con 30/60 cuadros por segundo).

2.2.2. Cámara RGB-D

Análogamente al caso anterior, la cámara RGB-D es fundamental para esta tesis, debido a que combina las características de una cámara RGB convencional con una dimensión de profundidad para conocer la posición exacta de formas y objetos, así como su volumen en el espacio [MA, 2002]. En la Figura 2.7 se puede ver la ilustración de esta.

Figura 2.7. Cámara RGB-D [MA, 2002]

Su principal atributo, es la captura de imagen y cálculo de distancia para el procesamiento de imágenes aplicadas hacia el control de la orientación y navegación, aumentando la eficiencia de la aplicación.

2.2.3. Software Matlab [MATLAB, 2015]

La descripción obtenida del sitio comercial, enuncia que Matlab es un entorno de computación numérica de paradigmas múltiples y un lenguaje de programación patentado desarrollado por MathWorks. Aunque MATLAB está destinado principalmente a la computación numérica, una caja de herramientas opcional utiliza el motor simbólico MuPAD que permite el acceso a las capacidades de computación simbólica. En la Figura 2.8 se aprecia el ícono oficial de Matlab.

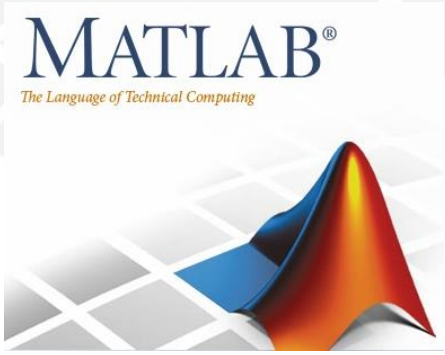


Figura 2.8. Icono de matlab [MATLAB, 2015].

En este proyecto se empleará Matlab y su conexión con Simulink para realizar el modelamiento de una parte del sistema de control.

2.2.4. V-REP [COPPELIASIM, 2012]

Parte de la descripción obtenida del sitio oficial de V-REP, define al entorno como un simulador de robot 3D basado en una arquitectura de control distribuida. Los programas de control (o scripts) pueden conectarse directamente a los objetos de escena y ejecutarse de forma simultánea con o sin hilos. Esto hace que V-REP sea muy versátil e ideal para aplicaciones multi robot, y permite a los usuarios modelar sistemas robóticos de forma similar a como lo hacen en la realidad, donde el control también se distribuye la mayor parte del tiempo. V-REP le permite editar y simular sistemas robóticos completos o subsistemas. Ofrece una multitud de funcionalidades que pueden ser fácilmente integradas y combinadas a través de una exhaustiva funcionalidad de API y script. V-REP puede utilizarse para la monitorización remota, para el control de hardware, para la rápida creación de prototipos y verificación, para el rápido desarrollo de algoritmos/ajuste de parámetros, para la doble comprobación de la seguridad, para la formación relacionada con la robótica, para simulaciones de automatización de fábricas, etc. En la Figura 2.9 se presenta el ícono oficial de V-REP.

Figura 2.9. Icono de V-REP [COPPELIASIM, 2012]

El sistema de control propuesto para la navegación, posición y orientación del robot se validará en el software V-REP que permite simular un entorno de una playa con las dimensiones de los objetos requerida para observar el movimiento y la dinámica del robot una vez se haya reconocido el objeto y se disponga a recolectar.

2.2.5. Sistemas de locomoción robótica

El movimiento del robot se refiere a las diferentes formas en que el robot debe moverse en una superficie determinada, siendo las más comunes a través de ruedas o patas. Los vehículos con ruedas son la solución más simple para moverse en terrenos extremos (es decir, muy duros o blandos), que pueden alcanzar velocidades relativamente altas.

Los robots móviles utilizan diferentes tipos de movimiento de las ruedas, cada uno con diferentes características y características [OLLERO, 2001]. Esto dependerá de la eficiencia energética, la movilidad y la carga útil. Los modos de transporte más comunes se detallan a continuación.

- **Ackerman**

El modelo Ackerman se suele utilizar en vehículos de cuatro ruedas, dos de las cuales se utilizan para tracción, y los componentes se ubican en la parte trasera paralela al chasis principal del vehículo. Las ruedas delanteras se utilizan para la dirección y cada rueda tiene dos ángulos de dirección diferentes, lo que puede causar problemas cuando desea realizar su propio control. En algunos casos, el ángulo de dirección es uniforme, pero esto no debe hacerse, porque los dos caminos no serán paralelos, por lo que la rueda no seguirá el camino ideal y se deslizará en la curva. Su principal desventaja es su movilidad limitada.

- **Tracción omnidireccional**

Este sistema de tracción especial se basa en el uso de tres volantes y ruedas motrices. Esta configuración tiene tres grados de libertad, por lo que puede moverse libremente y colocarse en cualquier posición independientemente de la dirección. Su ventaja es que no tiene restricciones cinemáticas, pero la desventaja es que los neumáticos mencionados anteriormente se desgastan a menudo.

- **Tracción diferencial**

El núcleo de esta dirección es la diferencia de velocidad entre las dos ruedas laterales. Cada rueda lateral está montada en un eje, por lo que cada rueda se acciona y controla individualmente para proporcionar tracción y dirección. Mirando sus capacidades de movimiento, se puede concluir que es un sistema muy útil porque puede cambiar de dirección sin realizar movimiento de traslación, es decir, puede rotar alrededor de su propio eje. Además, existen otras ruedas llamadas ruedas auxiliares que pueden ayudar a distribuir la carga mecánica y el peso. Esta configuración se suele utilizar para pequeños robots.

- **Triciclo clásico**

Como su nombre indica, se basa en las ruedas delanteras que se utilizan para la tracción y la dirección. El eje trasero tiene dos ruedas laterales que se mueven libremente, por lo que funciona de forma pasiva. Su maniobrabilidad es mejor que la del sistema Ackerman porque tiene un solo volante y el recorrido depende de este volante. Si bien esto es una ventaja a este respecto, tiene el inconveniente de que el vehículo es inestable en terrenos difíciles porque no puede soportar movimientos bruscos debido a la configuración de sus ruedas antes mencionada.

- **Skid – steer**

En este modelo, se utilizan más de dos ruedas a cada lado del vehículo y funcionan simultáneamente. El movimiento obtenido es causado por una combinación de velocidades de rueda izquierda y derecha. La cadena de arrastre del vehículo en forma de cadena se utiliza para lograr la dirección y la potencia necesarias. En terrenos irregulares, se recomienda utilizar rieles de deslizamiento, ya que la transmisión está menos restringida por el deslizamiento y tiene menor resistencia al desgaste.

- **Síncrona**

El movimiento sincronizado implica accionar todas las ruedas al mismo tiempo, que giran sincrónicamente para mover el robot móvil. La transmisión se realiza mediante correas concéntricas. Cada rueda se puede conducir y dirigir de forma independiente. La configuración más común, incluye tres volantes montados en los vértices de un triángulo equilátero en una plataforma cilíndrica o circular. Los modelos mencionados pueden observarse en la Figura 2.10, en donde se muestran los tipos de ruedas y la ubicación de estas para cada caso mencionado.

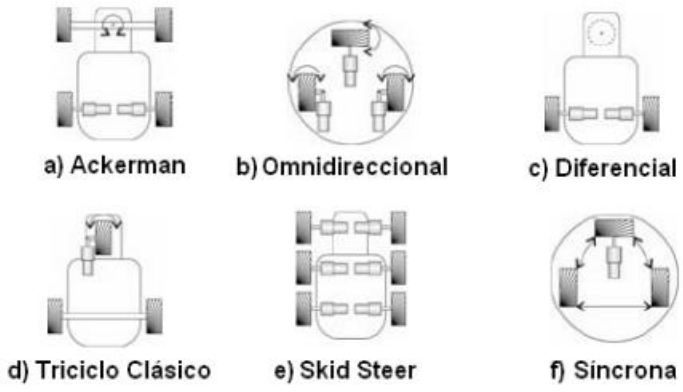


Figura 2.10. Diagramas de sistemas de locomoción para robots móviles [OLLERO, 2001].

2.2.6. Redes Neuronales

Las redes neuronales artificiales (RNA) son modelos de la Inteligencia Artificial que se inspiran en el comportamiento de las neuronas y las conexiones cerebrales para resolver problemas [LIU, 2016]. La Inteligencia Artificial (IA, en inglés AI o Artificial Intelligence) es una rama de la ciencia computacional que tiene como objetivo modelar la conducta de forma inteligente. Los sistemas de IA emplean algoritmos y modelos para analizar, ordenar, procesar y transformar datos. El objetivo es adquirir información útil al momento de tomar decisiones. Una red neuronal es un modelo simplificado simulado de los cerebros y la composición neuronal de los seres vivos, en especial del humano. Las redes neuronales poseen diferencias respecto a otros modelos de IA, ello radica en la capacidad de aprendizaje de forma automática una vez que ha sido entrenada de forma didáctica. Este proceso también se conoce como machine learning o aprendizaje de máquina. En la Figura 2.11 se presenta un esquema de una Red neuronal convencional de 3 capas para mayor ilustración.

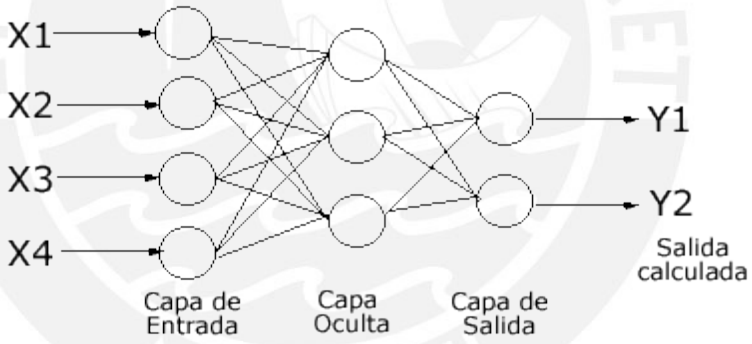


Figura 2.11. Red neuronal convencional de 3 capas [LIU, 2016].

Algunas de las aplicaciones de las redes neuronales artificiales son:

- Sistemas inteligentes para aplicar estrategias en la gestión tecnológica.
- Predicción.
- Reconocimiento de causas y efectos.
- Reconocimiento de patrones y gestión de riesgo.
- Máquinas inteligentes con aprendizaje continuo.
- Domótica.
- Sistemas de visión por computadora y procesamiento de imágenes.
- Vehículos automatizados que emplean energías renovables.

2.2.7. RCNN

El concepto de RCNN se define en forma general como una red neuronal convolucional que permite obtener dos salidas para determinadas aplicaciones [SHI, 2019]. Como se verá más adelante, ese concepto es fundamental en el desarrollo de este trabajo de tesis y es preciso conocerlo dado que su información no suele ser divulgada y la mayoría de sus aplicaciones no son públicas por seguridad.

En la Figura 2.12 se puede apreciar las diferentes etapas de una máscara de RCNN y como se obtiene las dos salidas mencionadas para distintos propósitos.

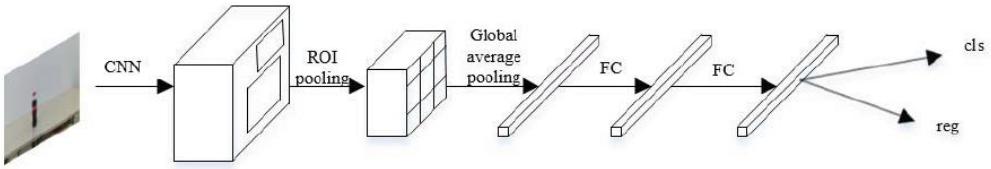


Figura 2.12. Etapas de la máscara RCNN [SHI, 2019].

Este tipo de redes puede ser entrenada de tal forma que se introduzcan filtros en el proceso para poder obtener el clasificador y el registro al final de cada imagen. De esta forma se podrán obtener las botellas de plástico y distinguir su ubicación exacta en el espacio. En este caso se realiza un entrenamiento sucesivo para garantizar la robustez de la red y la integridad de análisis de procesamiento de imágenes.

2.2.8. Técnicas y algoritmos de visión artificial

Como se ha mencionado previamente en este capítulo, existen diversas etapas en un proceso de visión artificial. A continuación, se describirán las técnicas más usadas para cada etapa, así como el aporte tecnológico que se tiene en este ámbito. Existen aspectos que son fundamentales para la etapa de adquisición de imágenes y se enfatizan a continuación [SONG, 2010]:

- **Cámara**
  - Cámaras digitales o analógicas: cada vez son más las cámaras digitales con nuevas características que entran en el mercado, la ventaja de estas cámaras frente a las analógicas es la velocidad y calidad de visualización de las imágenes. Las cámaras digitales transmiten información directamente en forma digital, lo que puede proporcionar una mejor calidad que el ruido que puede aparecer en los componentes de transmisión (conectores, cables, etc.). También pueden

implementar bits de detección / corrección de errores para verificar y corregir la información recibida La desventaja negativa es que requieren cables bastante gruesos (que constan de una gran cantidad de cables). Desafortunadamente, hay muy pocas cámaras y tarjetas en el mercado y sus precios son mucho más altos. Independientemente de si se trata de una tarjeta digital o una tarjeta analógica, se debe tener especial cuidado en la estructura del cable para asegurar un buen blindaje para evitar ruidos en la imagen grabada.

- Tipos de salida de video: actualmente hay tres tipos de salida de video: video compuesto, salida de video digital y salida RGB (para cámaras en color). Las ventajas de las cámaras de salida digital se han explicado en el punto anterior. Por otro lado, las cámaras de salida RGB separan los tres colores básicos, lo que hace posible operar cada color directamente desde el hardware; en cualquier caso, no hay más ventaja.
- Formato PAL o NTSC: hay dos formatos de vídeo principales: PAL europeo (50 Hz) o NTSC estadounidense (60 Hz). Algunas cámaras tienen un formato u otro. Debe asegurarse de que la tarjeta pueda tomar fotografías en el formato de su cámara [SONG, 2010].
- Salida entrelazada o no entrelazada: existen cámaras que tienen salida entrelazada, no entrelazada, o le permiten elegir entre las dos. La salida entrelazada es adecuada para la salida directa de monitores entrelazados y, por lo general, se utiliza para cualquier aplicación que verifique objetos estacionarios o que se mueven lentamente, mientras que la salida no entrelazada es más adecuada para capturar imágenes en movimiento. Una cámara que pueda lograr ambas funciones al mismo tiempo es ideal.
- Sincronización vertical y horizontal: Una buena cámara debe poder introducir la sincronización horizontal y vertical desde el exterior, así como extraer su propia sincronización interna. La salida o entrada sincronizada permite una iteración completa entre la tarjeta y la cámara para capturar el cuadro en el momento deseado. Es posible que algunos eventos sucedan muy rápido, lo cual es necesario para usar la sincronización o disparador que tiene la cámara para grabar correctamente la imagen deseada. En la salida RGB, las cámaras en este formato suelen tener sincronización en la salida G, aunque algunas cámaras pueden optar por mantener la sincronización a través de otra salida independiente.
- Velocidad de obturación: para obtener imágenes dinámicas, se debe controlar el

tiempo de exposición. Una cámara con buena relación calidad-precio debe poder programar su velocidad de obturación (mediante software o mediante puentes). Por lo general, hay cámaras con velocidades de obturación de 1/60, 1/125, 1/250, 1/500, 1/1000, 1/2000, 1/4000, 1/10000 segundos.

- Características del CCD: El CCD es una de las partes más importantes de una cámara, a la hora de elegir, no ignore las características básicas del CCD:
- Resolución: Cuanto mayor sea la resolución de la cámara, mejor será nuestro rendimiento. Se pueden obtener los detalles más pequeños del objeto Distinguir. Por supuesto, una resolución más alta significa precios de cámara más altos y tiempo de procesamiento de imágenes más largo, por lo que debe sopesar la resolución con estos otros factores.
- Relación de calidad de ruido: cuanto mejor sea la relación, mayor será la calidad de la imagen grabada. Existen algunas técnicas de filtrado (discutidas más adelante) para eliminar el ruido eléctrico generado en el CCD. El ruido eléctrico producirá una ligera niebla en la imagen que debe filtrarse.
- Programación RS232: una cámara puede calibrar directamente todos o parte de los parámetros desde el ordenador, esto permite automatizar el proceso de calibración.
- Otros atributos: cámaras con otros atributos, como las cámaras progresivas de dos canales permiten grabar en la mitad del tiempo, porque transmiten imágenes en dos canales al mismo tiempo, con líneas en un lado y líneas en el otro, datos pares o impares, etc.

De los puntos anteriores se puede concluir que a la hora de elegir una cámara se debe tomar en cuenta lo siguiente:

- Determinar qué proceso se debe realizar y en base a esto decidir si se hará uso de color o se necesita una cámara en blanco y negro.
- Elegir de acuerdo con el presupuesto, los requisitos de calidad, la velocidad de adquisición, etc.; si la cámara debe ser analógica o digital.
- Elegir la cámara con las mejores propiedades de velocidad de obturación, propiedades CCD, tipo de sincronización, programación a través de software, etc. de las cámaras disponibles.
- Encontrar la cámara más completa dentro de las posibilidades. Las cámaras más adecuadas para el propósito de este proyecto incluyen aquellas que tienen alta resolución y permiten la detección de la distancia al objeto enfocado, es decir, las

cámaras RGB-D vistas previamente.

- **Óptica**

La óptica a utilizar depende en particular de las condiciones ambientales y de la distancia de medición. Dependiendo del brillo del entorno circundante, es útil usar filtros para eliminarlos y permitirle capturar cómodamente los objetos que se desea explorar. Por otro lado, se pueden utilizar filtros especiales para enfatizar un determinado tipo de color y se pueden segmentar más fácilmente.

En la selección de componentes ópticos, especialmente en la determinación de las características geométricas de los objetos, se deben evitar todas las lentes que distorsionan la imagen: lentes que amplían o reducen el campo de visión, etc. Con el propósito de encontrar la mejor configuración, se debe buscar una lente de zoom que se pueda comprar para determinar qué lente fija usar en cada situación práctica. El grupo de anillo también se puede utilizar para lograr un rango de distancia focal más amplio.

- **Tarjeta de adquisición de la imagen**

Otro aspecto importante por considerar es el capturador de fotogramas. Hay muchas tarjetas de nivel profesional con múltiples funciones en el mercado y es difícil elegir una de ellas. Aquí están los más importantes:

- Tipo de entrada de video: La entrada de video debe admitir la salida de video de la cámara, ya sea video compuesto, RGB o video digital. Existen tarjetas en el mercado que permiten el uso de diferentes tipos de boletos. Por otro lado, tiene sentido utilizar varias cámaras al mismo tiempo. Por ejemplo, en un sistema de procesamiento de imágenes tridimensionales, esto también debe tenerse en cuenta al elegir una tarjeta de procesamiento de imágenes.
- Memoria incorporada y controlador DMA patentado: algunas tarjetas del mercado utilizan la memoria de la computadora para almacenar imágenes, mientras que otras tienen mucha memoria propia. Las tarjetas que utilizan memoria de computadora suelen ser más baratas, pero suelen tener varias desventajas. Las dos principales deficiencias de este tipo de placa base son: Por un lado, debido a la mala gestión de la memoria, el uso de la memoria de la computadora provocará conflictos con el sistema operativo, y los controladores instalados reducen la

memoria del sistema operativo para otros programas; Por otro lado, estas tarjetas deben gestionar la propia memoria del sistema, y deben acceder al chipset de la computadora, que suele ser incompatible con determinadas arquitecturas que utilizan chipsets antiguos o diferentes a los especificados en el manual de la tarjeta de visión. La tarjeta tiene memoria propia, además de un eficiente sistema de control DMA, que se puede transferir a la memoria del ordenador, que es la opción más adecuada3. De hecho, usar o no usar la memoria en la tarjeta puede significar consumir 1% o 42% del tiempo de la CPU, dependiendo del tipo de tarjeta 4 (el tiempo total que se tarda en transferir la imagen a la memoria usando la CPU) perdemos para otros procesos de capacidad).

- Entrada y salida síncrona: La tarjeta de video permite que el control de la señal de sincronización de la cámara sea muy importante, especialmente en el caso de capturar varias cámaras al mismo tiempo. La tarjeta tiene una salida síncrona de video compuesto y generalmente se usa para controlar una o más cámaras. Este proceso se denomina bloqueo de sincronización y permite la sincronización de señales al cambiar de canal o al capturar imágenes de dos o tres cámaras simultáneamente en un solo paquete.
- Se pueden usar disparadores externos, entradas y salidas digitales: Cada tarjeta de visión que se precie debe tener una entrada de disparador para disparar cuando ocurre un evento externo específico. Suponga que un sistema automático de detección de etiquetas de botellas de vino puede detectar con precisión cuándo el sensor de detección de posición indica que se puede detectar la botella. Hay muchas tarjetas con entradas y salidas digitales, que se pueden controlar fácilmente sin tarjetas de captura digital adicionales.
- Uso de DSP: Dado que las aplicaciones en entornos industriales deben ser en tiempo real, es muy importante que la tarjeta tenga un procesador de señal digital que se pueda programar para tareas de preprocesamiento, lo que reduce el tiempo de procesamiento de la CPU host. Por supuesto, uno o más DSP con capacidades de procesamiento paralelo aumentarán significativamente el precio final de la tarjeta de visión. El costo debe compararse con el tiempo máximo de cálculo estimado.
- Uso de procesadores dedicados para el procesamiento pipeline: para los sistemas de visión en tiempo real, actualmente existen tarjetas hechas de procesadores diseñados específicamente para el procesamiento pipeline, lo que puede aumentar

sustancialmente la velocidad de cálculo.

- **Preprocesamiento**

Cualquier imagen captada por medios ópticos, electroópticos o electrónicos se verá afectada en cierta medida por la degradación, que se manifiesta en la pérdida de ruido, nitidez y fidelidad de la imagen [SONG, 2010]. La degradación está provocada por el ruido del sensor de detección, la inexactitud del enfoque de la cámara, el movimiento de la cámara o interferencias aleatorias, entre las que es relevante la influencia de la propagación de la radiación en el medio de transmisión. El mecanismo que intenta contrarrestar estos efectos se incluye en la etapa de preprocesamiento, denominada operación de recuperación. En general, el preprocesamiento tiene como objetivo reparar errores en la imagen que son generados o no eliminados por el hardware: este último se deforma, introduce ruido, el contraste o brillo es pequeño o grande, carece de la corrección adecuada, etc. Los algoritmos de preprocesamiento le permiten ajustar los cambios de la imagen para eliminar el ruido, las transformaciones geométricas, aumentar la intensidad o el contraste, y se busca intentar mejorar el resultado final de la imagen capturada. Debido a que estos algoritmos requieren mucho tiempo de procesamiento, se prefiere utilizar el hardware adecuado para evitarlos. En el proceso de visión artificial, estos algoritmos deben usarse lo menos posible, un uso excesivo afectará a todo el tiempo del proceso e indicará que la calibración, iluminación y selección de elementos en la etapa de registro no son suficientes.

Además de la degradación de la calidad de la imagen, todavía hay algunas propiedades de la imagen que deben mejorarse en muchos casos. Por lo tanto, a menudo debe aumentar el contraste, el brillo, la escala de grises, eliminar el deslumbramiento, aumentar los bordes, mejorar la textura, etc. Todas estas técnicas se denominan operaciones de mejora de la imagen [SONG, 2010]. La parte de preprocesamiento del sistema de visión artificial consta de estas dos operaciones; H. Todas las correcciones y mejoras de la imagen se procesan para facilitar el procesamiento de los siguientes pasos. Este capítulo explica los usos más prácticos de las funciones de preprocesamiento comunes.

En cualquier caso, algunos algoritmos son difíciles de clasificar en una etapa u otra, lo que lleva a algunos autores a clasificar algunos algoritmos en la etapa de

preprocesamiento y clasificar otros algoritmos en la etapa de segmentación. En cuanto a preprocesamiento y segmentación, extracción de parámetros y análisis de información, se han realizado una serie de estudios sobre los mecanismos clásicos del procesamiento de imágenes. Entre todos los algoritmos que utilizan estas tecnologías, se seleccionan los más útiles en aplicaciones de visión artificial. La descripción de estos algoritmos no es exhaustiva, sino que se basa en aplicaciones prácticas y aplicaciones de reconocimiento de formas y visión artificial [SONG, 2010]. Se hará mención de algunas de las técnicas de preprocesamiento de imágenes más usadas, y sus expresiones características, recopiladas por el grupo de investigación EDMANS de la Universidad de La Rioja en España [EDMANS, 2006].

- **Inversión**

Como su nombre lo indica, consta del valor inverso de cada pixel y se denota por la siguiente expresión:

IMB(i,j)=1/IMA(i,j)

- **Operaciones aritméticas**
  - Adición y substracción: Consiste en sumar o restar imágenes unas con otras.
    IMC(i,j)=IMA(i,j)±IMB(i,j)
  - Producto o división por una constante: Multiplicar o dividir el valor de cada pixel por una constante. Se usa para aumentar o disminuir el brillo en una imagen.
    IMB(i,j)=C*IMA(i,j) o IMB(i,j)=IMA(i,j)/C
  - Logaritmo exponencial: Se usa para aumentar el contraste de las zonas oscuras contra las zonas claras o viceversa.
    IMB(i,j)=K*Log10(1+IMA(i,j)) o IMB(i,j)=K*exp(IMA(i,j)-1)

- **Operaciones lógicas: AND, OR y XOR:**

Se emplean para obtener o eliminar una determinada porción de la imagen.

IMC(i,j)=IMA(i,j) & IMB(i,j) (AND)

IMC(i,j)=IMA(i,j) | IMB(i,j) (OR)

IMC(i,j)=IMA(i,j) ^ IMB(i,j) (XOR)

- **Transformaciones no lineales**

Funciones elementales como por ejemplo corrección gamma, afilado de la imagen, etc. Se realizan haciendo uso de una función de transformación h(x).

IMB(i,j)=h(IMA(i,j))

- **Slicing**

Se usa para resaltar una determinada franja de niveles de gris que se aproximan a un valor primario o máximo, mientras que los valores de luminosidad que restan se quedarán en 0 o su valor previo.

- **Clipping**

Convierte todos los niveles de grises superiores a un valor limite a un valor fijo y los valores menores a un valor mínimo a otro valor fijo. De tal forma que se tengan dos cotas y un intervalo, haciendo una analogía con el dominio de la frecuencia se entendería como un filtro pasa banda.

IMB(i,j) = Fmin, si IMA(i,j) < Valmin

- **Umbralización**

Se conoce como valor umbral o de threshold y consiste en suprimir los valores superiores o inferiores respecto a un valor establecido como umbral.

IMB(i,j)=IMA(i,j))*(IMA(i,j)≥umbral)

- **Binarización**

Es una variante lo descrito previamente y consta en dejar en cero todos los pixeles menores a un umbral y pone en uno los pixeles que son iguales o mayores quedando la imagen final polarizada entre 0 y 1.

2.2.9. Control del sistema de movimiento de robots móviles

Existen diversos tipos de control empleados en robótica. En este apartado se revisan los principales 3 controles que se emplean en robot móviles para identificar el más adecuado basado en el propósito y experiencias recopiladas en el estado del arte.

- **Control adaptativo óptimo**

Este tipo de control intenta minimizar los comportamientos indeseables relacionados con las limitaciones físicas del sistema [SHANGTAI, 2007]. Si tiene una estructura de tracción con dos ruedas, su función es dar direcciones, y las dos ruedas van delante. Se comprueba que el sistema de dirección es aproximadamente lineal, pero hay un pequeño sistema de avería que no forma parte del modelo de vehículo, que hay que tener en cuenta, ya que pueden ser la causa de la inestabilidad del sistema. Para compensar los efectos de las perturbaciones del entorno externo mencionadas anteriormente, este tipo de control se implementa en los actuadores y componentes involucrados en el movimiento del sistema.

- **Control Proporcional Derivativo Integral (PID)**

Este tipo de control es el más común porque los cálculos necesarios para determinar los parámetros del controlador son más fáciles de realizar que otros tipos de control. Por tanto, se está convirtiendo en una alternativa más consistente a los robots móviles que no requieren alta precisión ni realizan una intervención continua. Es común encontrar un controlador PID integrado con otras estrategias más sofisticadas [RITHIRUN, 2021], como las técnicas de Inteligencia Artificial, para garantizar que las perturbaciones no interfieran con el sistema principal y una vez se conoce la estabilidad del sistema, se emplean configuraciones para realimentar la red y de esta forma generar que la red aprenda continuamente sobre los fallos.

- **Control basado en Inteligencia Artificial**

La estrategia de control de la Inteligencia Artificial se basa en una red neuronal, un algoritmo difuso, etc. Este tipo de algoritmo se utiliza a menudo para robots que se ven perturbados debido a su estructura liviana o entorno de trabajo, lo que significa que el agente de parámetros externos no es completamente conocido y está aislado de otros tipos de control. Por lo general, se combinan con otros controles para manejar subsistemas integrados en el sistema principal [KHALIL, 2002]. En este proyecto, este tipo de control se destaca porque tiene como objetivo obtener la identificación de botellas de plástico en tiempo real, reduciendo el tiempo de respuesta y optimizando el tiempo total, ya que puede aumentar el índice de recogida de botellas por hora.

Entre los tipos de control de la Inteligencia Artificial, encontramos el control de tipo servovisual. La función especial de este tipo de control es registrar datos a través de un sensor de visión, la cámara. La cámara puede extraer los parámetros requeridos por el sistema para desarrollar un controlador estable que le permita generar la trayectoria del objeto detectado, que contiene información sobre la distancia que debe recorrer y el camino a seguir, sin tener que interactuar con otro objeto. La colisión es en movimiento o estacionaria. Como sugiere el nombre, el actuador se puede usar con un servomotor para usar Inteligencia Artificial. De esta forma, todo el aprendizaje generado por la red neuronal también está conectado a este tipo de control, que se manifiesta como la dependencia entre el dominio de procesamiento de imágenes y el área de control. Cuanto mejor se vuelva a realizar la detección de botellas, más confiable serán los datos proporcionados por la cámara para el mejor control posible.

CAPÍTULO 3

DISEÑO Y DESARROLLO

En este capítulo, se presenta la propuesta de diseño del robot móvil y los dominios que lo conforman. Asimismo, se desarrolla el algoritmo de visión artificial que permite el entrenamiento de la red RCNN, donde se emplean imágenes de autoría propia capturadas en playas del litoral limeño.

3.1. Dominio mecánico

En el dominio mecánico, se propone el diseño como alternativa de solución para el desplazamiento del robot móvil tras la detección e identificación de la botella plástica. Se ha tomado en cuenta los proyectos revisados en el estado del arte para encontrar el mejor mecanismo que permita la recolección de botellas de plástico, así como en la selección de partes mecánicas para su diseño. Además, se ha considerado el modo en el que otras propuestas de robot móviles se han desplazado en terrenos adversos como tuberías, arena, césped, entre otros. Por ello, se hace uso de un factor para simular el terreno arenoso y con ello hacer la selección correspondiente como se verá posteriormente.

3.1.1. Esquema y construcción

En el diseño del robot móvil, se tomaron en cuenta aspectos revisados previamente en el estado del arte del documento, donde se puede inferir que la mejor alternativa para esta aplicación es un sistema de motores de acción lineal y un mecanismo de una bandeja accionada por dos motores que permitan recoger las botellas de plástico una vez hayan sido detectados, a continuación se muestra la propuesta de diseño elaborada, así como una tabla donde se explica cada componente y el propósito del mismo. En la Figura 3.1 y Figura 3.2 se muestra el diseño mecánico del robot móvil desde dos perspectivas.

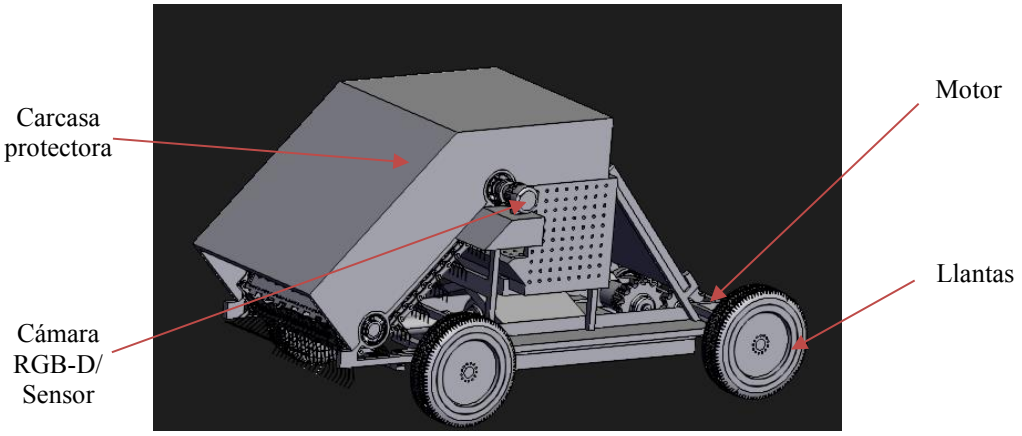


Figura 3.1. Esquema del robot móvil

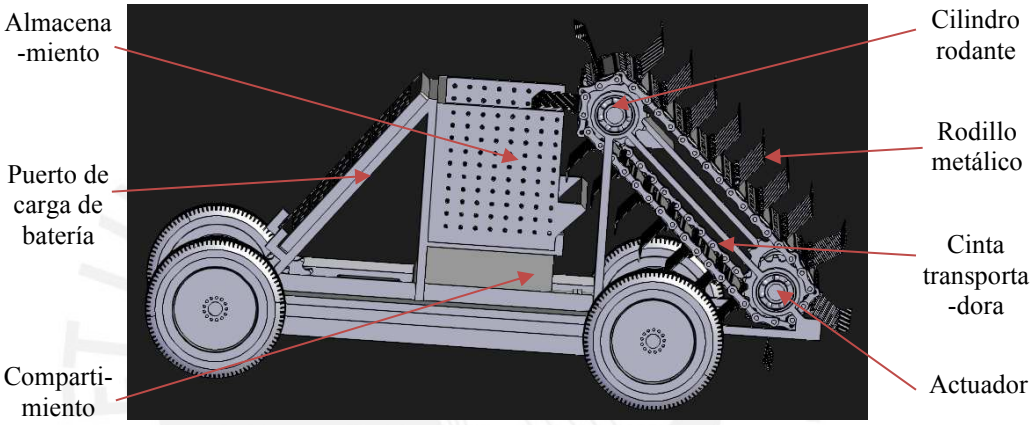


Figura 3.2. Vista lateral del robot móvil

A continuación, se muestra la Tabla 3.1, correspondiente a la clasificación de cada parte del robot móvil y la justificación de su importancia en el diseño.

Tabla 3.1. Lista de elementos mecánicos

Componente	Función
Llantas	Cada llanta será accionada por un motor DC con torque elevado.
Motor	Motorreductores 24VDC 2600rpm. Motor no en contacto con agua.
Actuador lineal 12V	Se usará para empujar las botellas almacenadas una vez que el espacio designado esté cargado al máximo

Cámara RGB-D/Sensor	Se encuentra equipado con una cámara RGB-D y un sensor de presencia. Cuando se detecte una botella, la faja será activada y las botellas subirán por la rampa interna.
Rodillo metálico	Accionado por dos motores permitirá la subida de las botellas hasta la etapa de almacenamiento
Cinta transportadora	Empleada para tomar y transferir las botellas recolectadas hacia el almacenamiento
Carcasa protectora	Funciona como prevención en caso el robot tenga contacto con el agua o cúmulos de arena. También se usa como puesta a tierra.
Puerto de carga de la batería	Permitirá cargar la batería sin tener que reemplazarla cuando el robot se encuentre detenido en la estación de carga.
Almacenamiento	Usado para mantener en ese espacio las botellas recolectadas de hasta 750ml cada una.
Cilindro rodante	Una vez que las botellas son recolectadas y almacenadas en el espacio designado, todos los cilindros se activarán y girarán para asegurar que las botellas ocupen menos espacio y que tengan aproximadamente la misma altura. El cilindro rodante es accionado por un motor 12VDC.
Compartimiento	Compartimiento hecho de aluminio anodizado relleno de sílice para prevenir que se dañen o mojen los componentes.

Las dimensiones consideradas para este vehículo son de 2.10m de largo x 1.52m de ancho. Estas dimensiones, se definieron tras analizar en el estado del arte, que los robots móviles se dimensionan para la aplicación en específico a realizar y se toman en cuenta experiencias previas de los investigadores principales para la toma de decisiones. Si bien está diseñado para un futuro funcionamiento al aire libre, se consideró el traslado hacia la playa en un automóvil y la facilidad para sostenerlo por si ocurriese alguna emergencia, por lo que su manipulación debe ser ergonómica.

En las siguientes Figuras, se pueden apreciar las piezas mecánicas modeladas en 3D bajo condiciones de carga estática, de tal forma que se visualiza parte de lo trabajado en este dominio del proyecto.

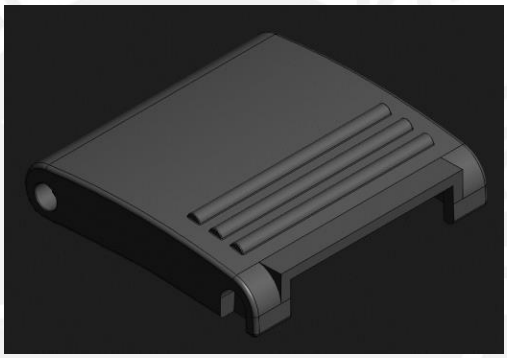


Figura 3.3. Tapa posterior

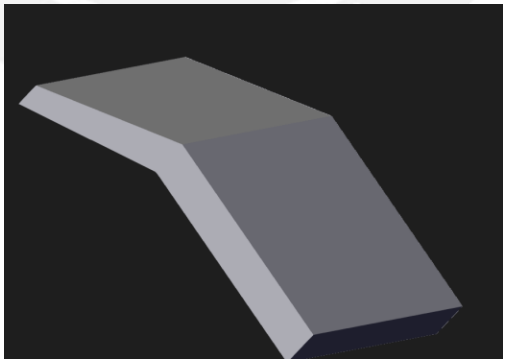


Figura 3.4. Tapa superior

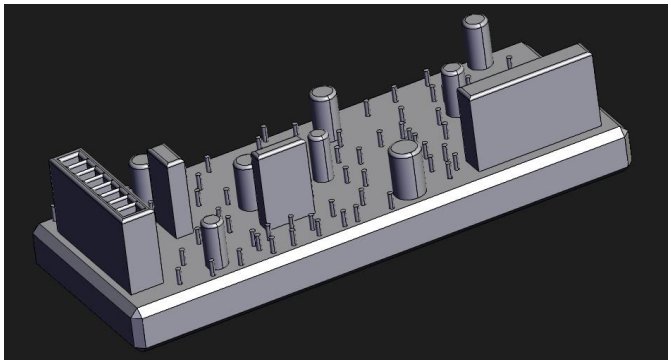


Figura 3.5. Esquema de la placa electrónica

Figura 3.6. Rieles del interior del robot móvil

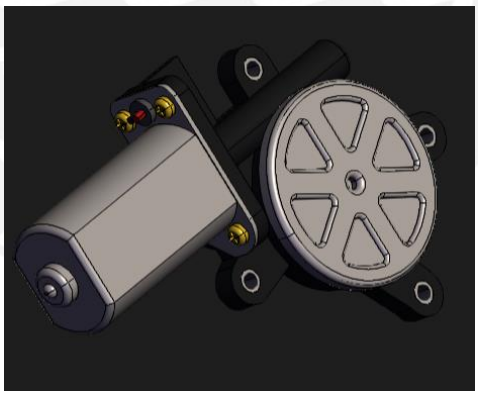


Figura 3.7. Motores DC

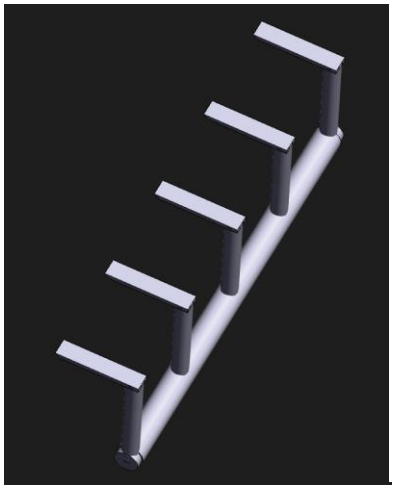


Figura 3.8. Aplanadores de botellas

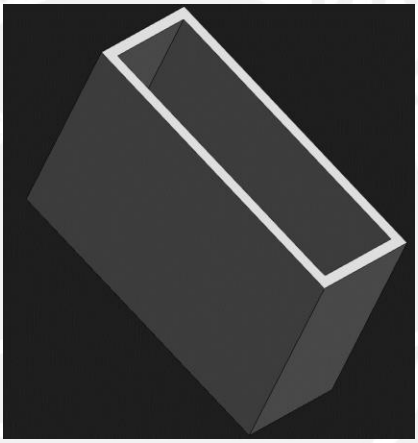


Figura 3.9. Caja de almacenamiento de botellas

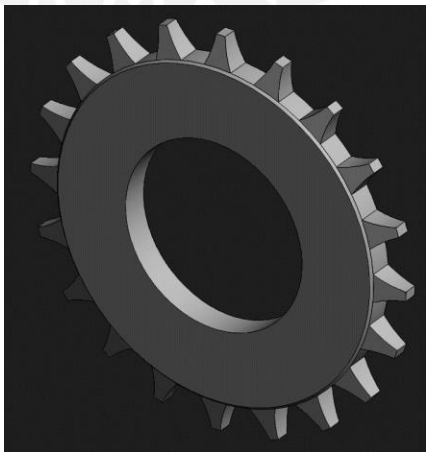


Figura 3.10. Engranajes de uniones

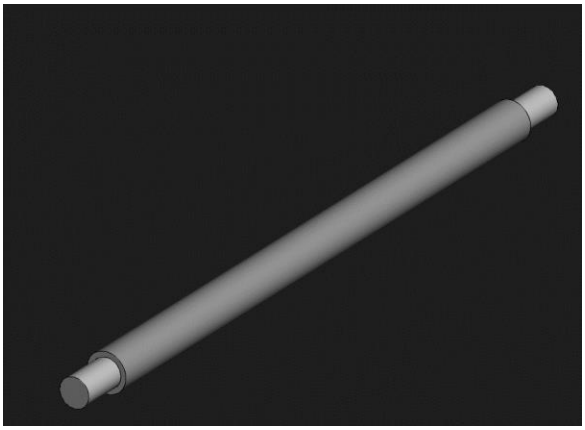


Figura 3.11. Eje de tracción

Figura 3.12. Llantas del vehículo

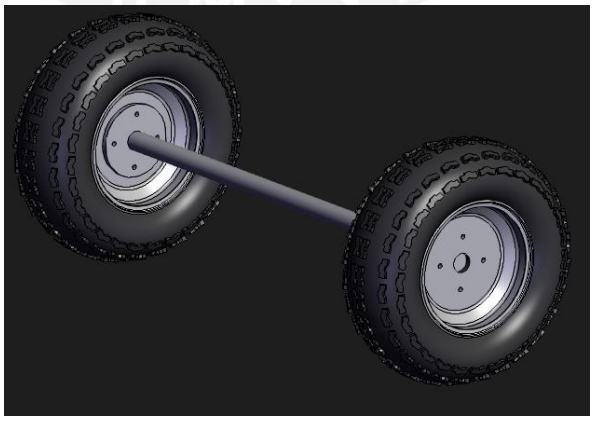


Figura 3.13. Llantas unidas en tracción diferencial

3.1.2. Selección de motores

En esta sección, se realiza la selección de motores para el dominio mecánico propuesto que permita la integración simulada con los demás dominios y permita el objetivo principal.

3.1.2.1. Peso total y requerimientos de tracción

Se definen los parámetros esenciales para la correcta selección de motores listados a continuación, basados en el requerimiento mencionado en la problemática y en la necesidad de diseñar un dominio mecánico, que permita el desplazamiento simulado en un terreno adverso. En la Tabla 3.2, se presentan las características del diseño de este dominio.

Tabla 3.2 Características del diseño mecánico

Características	Símbolos	Valores
Peso del robot	M_R	15 kg
Peso máximo de carga	M_L	3 kg
Velocidad Nominal	V_N	1 m/s
Velocidad de levantamiento de carga	V_L	0.3 m/s
Tiempo de aceleración	T_A	0.8 s
Pendiente máxima	K	25%

El valor de pendiente máxima se establece como 25% debido al terreno no uniforme presente en la arena. Tomando en cuenta el estado del arte revisado previamente, en diseños de robots que se desplazan en terrenos no uniformes, se establece este parámetro como factor de compensación para un correcto dimensionamiento y elección de los motores, con la intención de que pueda movilizarse sin problemas en este terreno no plano.

3.1.2.2. Cálculo de la velocidad de rotación de la llanta de tracción

Se realiza el cálculo de la velocidad de rotación de la llanta mencionado. Para esto, es necesario conocer el diámetro de la llanta y la velocidad nominal del robot. En este caso, tomando un diámetro de la llanta de 20cm se tendrá la siguiente expresión matemática descrita en la Ecuación 3.1.

$$V_R = \frac{60V_N}{\pi D_W} = \frac{60\ x\ 1}{0.2\pi} = 95.49\ rpm \quad (3.1)$$

Donde:

D_W: Diámetro de la llanta

3.1.2.3. Cálculo de la potencia y torque del motor de tracción

Se requiere estimar el torque del motor necesario para surcar las pendientes en la arena de valor definido previamente, por lo que se calcula la fuerza de empuje que permite superar la gravedad en la pendiente como se expresa en la Ecuación 3.2:

$$F_E = gK(M_R + M_L) = 9.81\ x\ 0.25\ x\ 18 = 44.14\ N \quad (3.2)$$

Tomando en cuenta que el robot se moverá a la velocidad nominal, la potencia mecánica requerida para lograrlo se calcula en la Ecuación 3.3:

$$P_v = F_E V_N = 44.14\ x\ 1 = 44.14W \quad (3.3)$$

La potencia mecánica calculada total se distribuye en los motores de tracción, por lo cual se tendrá 22.07W en cada uno. El torque correspondiente para cada llanta de tracción viene dado por la Ecuación 3.4:

$$\frac{1}{2}\ x\ \frac{D_W}{2} x\ F_E = 0.5\ x\ 0.5\ x\ 0.2\ x\ 44.14 = 2.207\ Nm \quad (3.4)$$

Se han calculado con éxito los parámetros más importantes requeridos para la elección apropiada de motores de tracción.

3.1.2.4. Cálculo de la velocidad de rotación del motor de elevación

De forma análoga a lo realizado para los motores de tracción, en este caso para calcular la velocidad de los rodillos que se encargan de elevar las botellas hasta el compartimiento de depósito, se hace uso del diámetro del engranaje y la velocidad de levantamiento de carga.

Tomando en cuenta que el diámetro es de 4cm, se tendrá la siguiente velocidad de rotación nominal expresada en la Ecuación 3.5.

$$N_L = \frac{60V_L}{\pi D_E} = \frac{60\ x\ 0.3}{0.04\pi} = 143.24\ rpm \tag{3.5}$$

Donde:

D_E: Diámetro del engranaje para la carga

3.1.2.5. Cálculo de la potencia y el torque del motor de elevación

Con el objetivo de calcular la potencia y el torque requeridos para elevar la carga; es decir las botellas de plástico, se tomará en cuenta un peso adicional. Esto se fundamenta, en que al elevarse la carga no solo se elevará ésta, sino que componentes que forman parte de la cadena de elevación también lo harán por lo que se agregará un peso extra de 1kg. Por ende, la fuerza necesaria para levantar la carga total se muestra en la Ecuación 3.6:

$$F_L = g(M_C + M_L) = 9.81\ x\ 4 = 39.24N \tag{3.6}$$

Donde:

M_C: Peso de componentes

Así, el torque correspondiente para el engranaje que acciona el mecanismo de elevación de carga se calcula en la Ecuación 3.7:

$$T_L = \frac{D_E}{2} x\ F_L = 0.5\ x\ 0.04\ x\ 39.24 = 0.785\ Nm \tag{3.7}$$

Asimismo, la potencia mecánica necesaria para elevar la carga a la velocidad nominal de elevación se obtiene en la Ecuación 3.8:

$$P_L = F_L V_L = 39.24\ x\ 0.3 = 11.77W \tag{3.8}$$

Con lo calculado, se tienen los siguientes valores hallados hasta el momento, los cuales se agrupan en la Tabla 3.3.

Tabla 3.3. Características halladas

Características	Motor de tracción (2)	Motor de elevación
Velocidad rotacional nominal	95.49 rpm	143.24 rpm
Torque	2.207 Nm	0.785 Nm
Potencia	22.07 W	11.77W

Con estos datos, se realizó la selección del motor para satisfacer los requerimientos tomando como base los parámetros establecidos y hallados previamente. Se optó por el motor de modelo JC/LC-578VA-4720 de la marca Mabuchi cuyas gráficas se muestran a continuación. En la Figura 3.14 tras haber tomado los parámetros hallados e inspeccionado el catálogo de la conocida marca, se analizó las curvas del modelo mencionado verificando que efectivamente es ideal para el robot móvil.

Como se puede verificar en las rectas trazadas, se obtiene que para una potencia de aproximadamente 22W corresponden:

- Torque: 3.1 Nm (Superior al requerido)
- Corriente: 7A (Soportado por el controlador contemplado)
- Eficiencia Total: 26
- Velocidad rotacional empírica (NRE): 65rpm

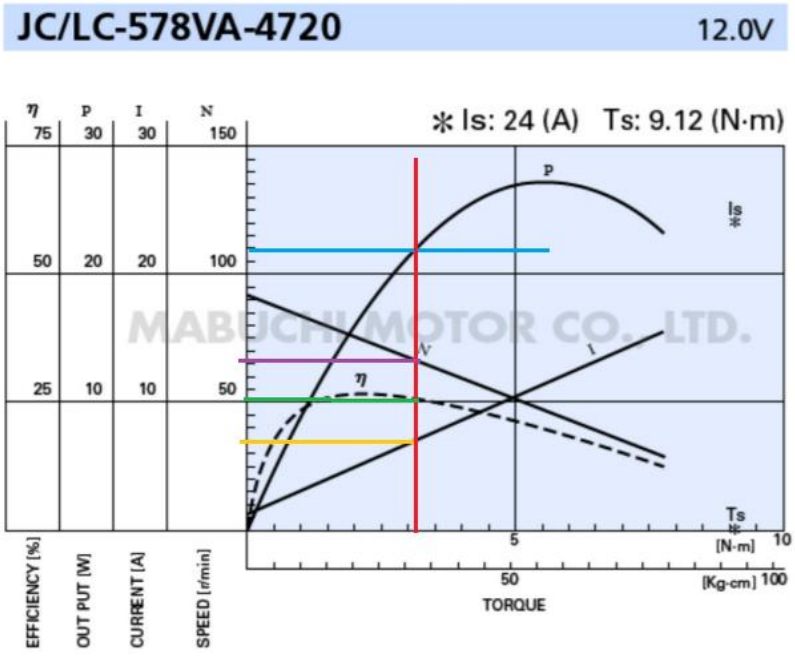


Figura 3.14. Gráfico de curvas del motor seleccionado

Del último valor obtenido en la gráfica, se halla la relación de transmisión final, cuya relación se expresa en la Ecuación 3.9.

$$R_T = \frac{V_R}{N_{RE}} \tag{3.9}$$

3.2. Dominio electrónico

En el diseño del dominio electrónico, se hará uso de un módulo de cómputo NVIDA denominado Jetson nano, dicho módulo permitirá el procesamiento de imágenes, así como el entrenamiento de la red y el accionamiento de los motores con diferentes propósitos específicos. Por recomendación de los expertos y tras haber revisado los procesadores empleados en los diversos proyectos analizados en los antecedentes relacionados a IA, se ha elegido el módulo de cómputo mencionado por la alta capacidad de integración que tiene con proyectos que involucran visión artificial. A continuación, se muestra el módulo de cómputo mencionado en la Figura 3.15.

Figura 3.15. Módulo Jetson nano NVIDIA [LL, 2021]

Este módulo tiene las siguientes especificaciones técnicas [LL,2021]:

- GPU Maxwell de 128 núcleos
- Procesador Quad-core Arm A57 a 1,43 GHz
- Memoria del sistema – 4GB 64-bit LPDDR4 @ 25.6 GB / s
- Almacenamiento: ranura para tarjeta microSD (devkit) o flash de 16 GB eMMC (producción)
- Codificación de video – 4K @ 30 | 4x 1080p @ 30 | 9x 720p @ 30 (H.264 / H.265)
- Decodificación de video – 4K @ 60 | 2x 4K @ 30 | 8x 1080p @ 30 | 18x 720p @ 30 (H.264 / H.265)
- Dimensiones – 70 x 45 mm.

Se empleará el controlador MD20A para controlar los motores. Este controlador ha sido seleccionado puesto que permite manipular y controlar motores de 20A a 6-30V, lo cual lo convierte en la alternativa ideal para accionar los motores de desplazamiento, así como los motorreductores empleados para accionar la bandeja y otros componentes. En la Figura 3.16 se puede apreciar el controlador de corriente descrito.

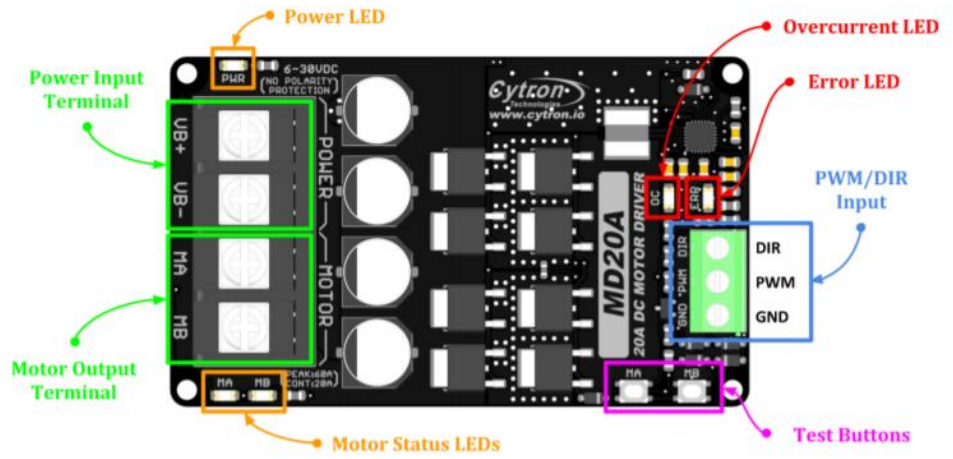


Figura 3.16. Controlador de corriente MD20A

Se ha tomado en cuenta el dimensionamiento de la batería, dado que es importante establecer el tamaño exacto para considerar un factor de seguridad apropiado para el funcionamiento del robot sin correr riesgo alguno. Asimismo, en la Figura 3.17 se muestra el diagrama de conexiones empleando el controlador descrito y el módulo de cómputo Jetson nano, junto con la batería y los motores.

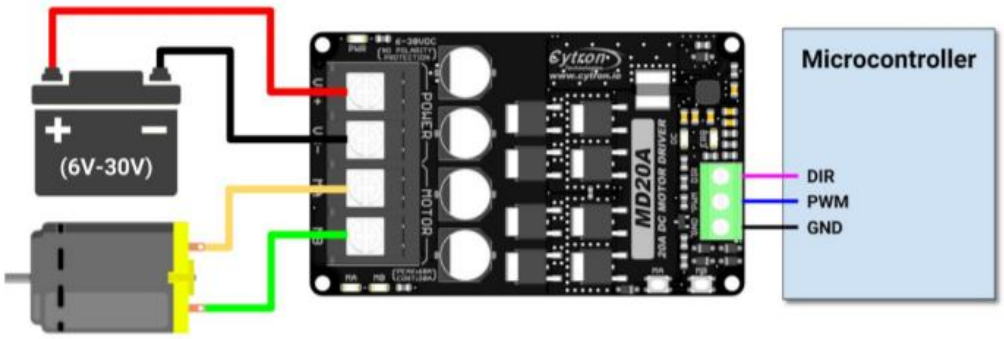


Figura 3.17. Diagrama de conexiones del dominio electrónico

Con el ideal de garantizar la seguridad del robot y de las personas que se encuentren en el entorno donde se desplazará el móvil, se hará uso de sensores de ultrasonido A02YYUW cuya ilustración se encuentra en la Figura 3.18. Estos sensores fueron elegidos porque tienen la particularidad de tener resistencia al agua IP67, tener un mayor recubrimiento y funcionan bien en robots móviles al aire libre.

Figura 3.18. Sensor de ultrasonido A02YYUW

En cuanto a especificaciones técnicas se refiere, se pueden encontrar las siguientes en su hoja de datos.

- Voltaje de operación: 5V
- Corriente de operación < 8mA
- Temperatura máxima: +60°
- Angulo de referencia: 60°
- Distancia máxima por detectar: 450cm
- Ángulo de detección: 100°
- Dimensiones: 73 x 30 x 5 mm
- IP67 resistencia al agua

Los sensores se distribuirán como se muestra en la Figura 3.19, de esa forma se logrará sensar constantemente la periferia del robot y dicha data se almacenará para evitar colisiones en los siguientes instantes. Este sistema basado en el sistema comercial Pioneer se emplea como un grupo constructivo extra y apoyará al sistema principal de visión para tener un valor agregado en la seguridad del robot y de las personas en la playa.

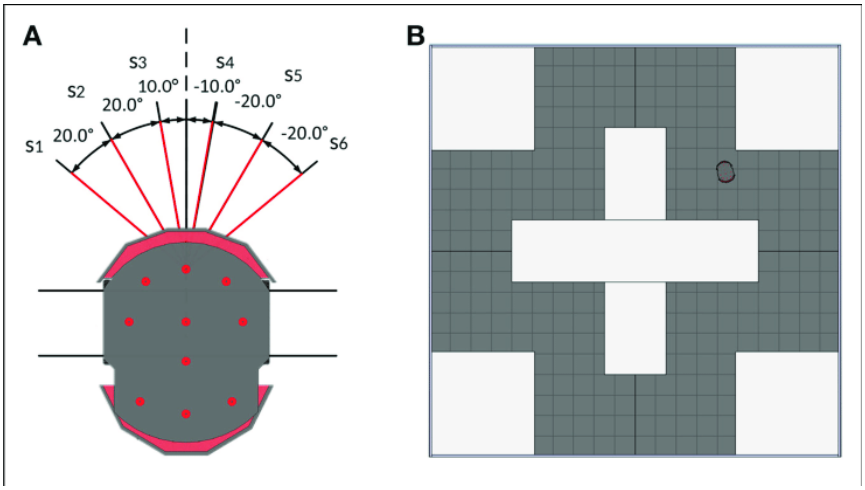


Figura 3.19. Arreglo de sensores de ultrasonido basado en Pioneer

Se colocará un arreglo de 8 sensores en la cara frontal y en la cara posterior del robot móvil para una mayor eficiencia. De esta forma, se evitará cualquier tipo de colisión posible y no se incrementará el trabajo del sistema de control por visión, ya que se usarán puertos GPIO del procesador Jetson NVIDIA nano. Se ilustra lo descrito en la Figura 3.20.

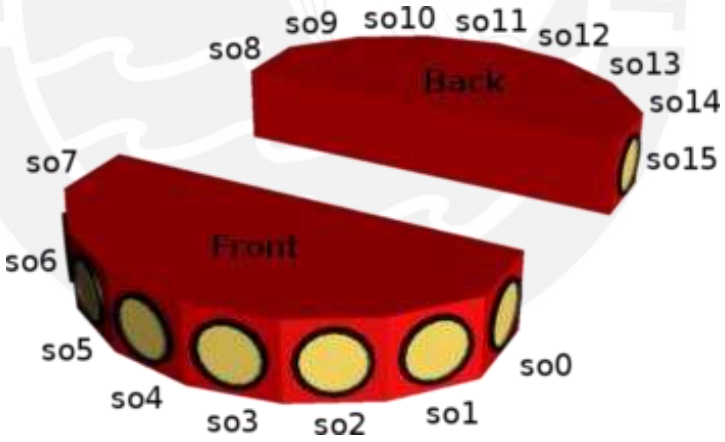


Figura 3.20. Vista isométrica de arreglo de sensores de ultrasonido basado en Pioneer

La cámara seleccionada para adquirir las imágenes es la Intel® realsenset depth camera d435/d435i, cuya ilustración se encuentra en la Figura 3.21.

Figura 3.21. Cámara RGB-D intel

La justificación de esta elección es debido a la necesidad de capturar la imagen y su ubicación en el espacio, así como la profundidad de esta. Es decir, se toma en cuenta la distancia que existe hasta la imagen capturada. Esta cámara, cuenta con la particularidad de tener 4 lentes y capas de fabricación expuestas en la Figura 3.22, donde cada uno de los lentes está designado para una tarea en particular. Esta cámara presenta una compatibilidad de hasta 90% con proyectos relacionados a visión artificial como se ha podido apreciar en el estado del arte.

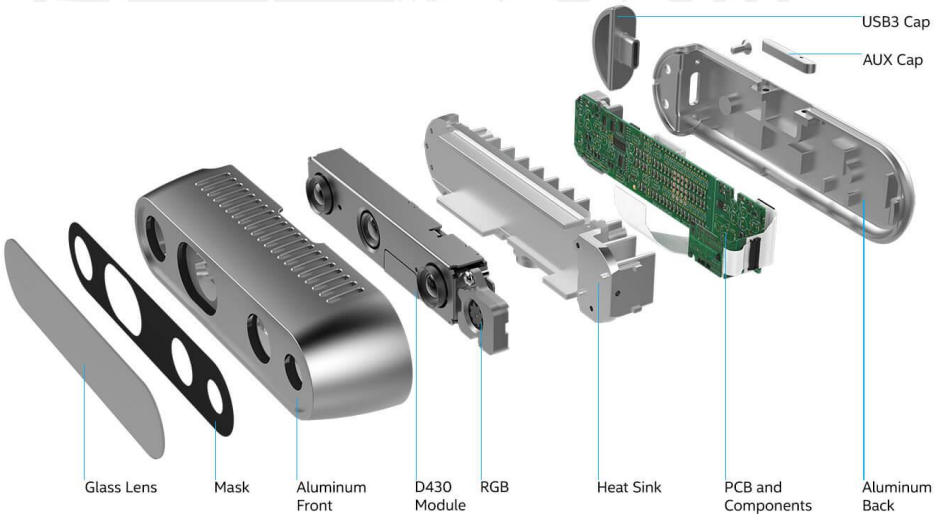


Figura 3.22. Descripción de cámara RGB-D intel

Con respecto a la alimentación del robot móvil, se hará uso de una Batería de 12V - 30Ah de tal forma que pueda alimentar el controlador revisado previamente y permita el correcto funcionamiento del robot móvil. Con la batería señalada en la Figura 3.23 podrá funcionar sin problemas 3 horas que es el tiempo estimado en el que recolecte las botellas de plástico antes de dirigirse a la estación de carga y donde se despeje el compartimiento de almacenamiento.

En la Figura 3.23, se puede apreciar la batería empleada. Cuando el robot móvil se encuentre a 20% de su carga nominal, se desplazará hacia el módulo de carga, donde se hará uso de este circuito se podrá realizar la carga sin problemas.

Figura 3.23. Batería seleccionada

En complemento, el circuito acondicionador para carga de batería se muestra en la Figura 3.24. Se empleará para cargar la batería de 12V una vez que el robot se encuentre detenido en la estación de carga. En este circuito se desarrolla la etapa de rectificación en el puente de diodos, luego la etapa de filtración mediante los condensadores y la etapa reguladora mediante el regulador 78ADJ el cual se encargará de reducir el valor de la tensión. Asimismo, se usan componentes para eliminar el ruido y se designa una etapa de auto corte por seguridad.

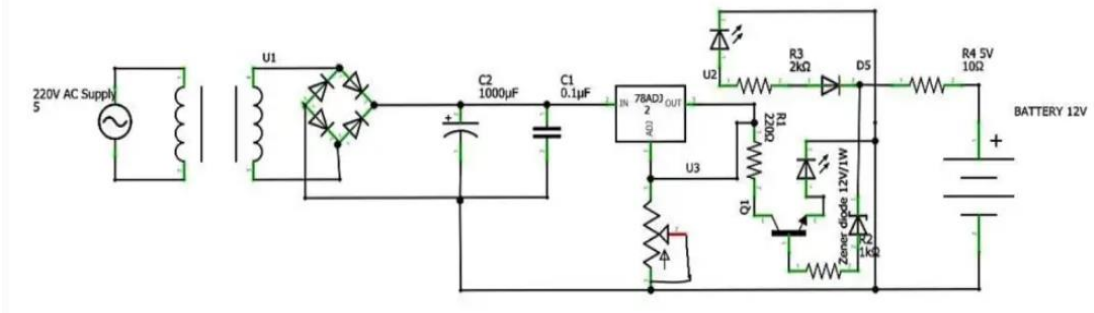


Figura 3.24. Circuito para carga de batería

3.3. Dominio de control

El robot diseñado tiene un módulo de construcción y una amplia gama de posibilidades en cuanto al dominio de control para funcionar de forma óptima. Es así que se ha considerado necesario establecer un control de velocidad, ya que resulta ser de mucha utilidad en este escenario. Asimismo, es necesario señalar que para diseñar el controlador es necesario identificar la dinámica de los motores.

Se procederá a modelar el sistema a través de una función de transferencia, de tal forma que se establezca un modelo matemático que relacione la entrada y la salida de un tiempo lineal sistema invariante. La función de transferencia de un sistema, H (s), es la relación entre la transformada de LaPlace de la salida Y (s) y la transformada de LaPlace de la entrada U (s), como se puede ver en la Ecuación 3.10

$$H(s) = \frac{Y(s)}{U(s)} \tag{3.10}$$

Hay muchos métodos para identificar un modelo. El método elegido fue la respuesta escalonada, que consiste en alimentar el sistema con entradas conocidas (voltaje aplicado al motor) y observar la salida (velocidad medida con el encoder). El motor real no tiene mucha linealidad, por lo que el sistema de control será lo suficientemente robusto para otorgar estabilidad y seguimiento de referencias para todo el rango de operación. A veces, un modelo simple pero adecuado puede ser suficiente para representar la dinámica del sistema permitiendo el proyecto del controlador. Algunas pruebas demostraron que una función de transferencia de primer orden da una aproximación suficiente para modelar los motores, y un controlador PID constituye la mejor alternativa para el control de velocidad.

El diagrama de bloques presentado en la Figura 3.25 representa una estructura simplificada del desarrollado sistema de circuito cerrado para el control de velocidad del motor. La estructura PID elegida facilita la implementación en los microcontroladores, para las ganancias individuales que se disponen en paralelo, entonces las ganancias calculadas en tiempo continuo son lo mismo que en tiempo discreto, como se ve comúnmente en controladores industriales de alta gama.

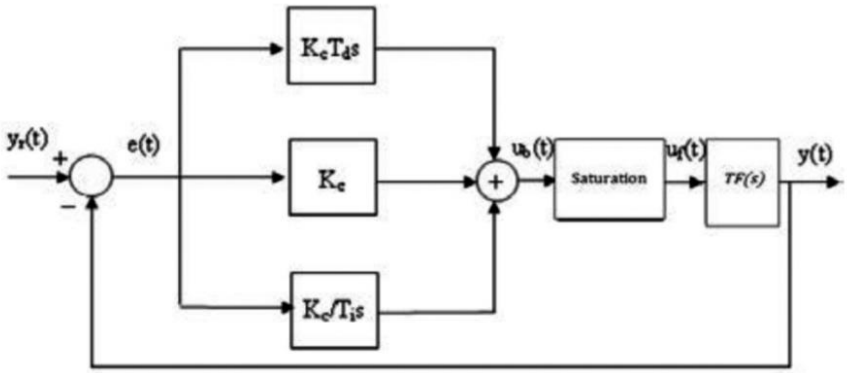


Figura 3.25. Diagrama PID

Se aplicó una entrada de voltaje tipo escalón para identificar la dinámica del sistema. En el ensayo realizado, el microcontrolador aplicó 3.92VRMS al controlador que a su vez amplificó la corriente hacia el motor y se midió la velocidad angular. Los datos recopilados de la respuesta del sistema en lazo abierto es lo que se muestra en la Figura 3.26, representada por la curva con la etiqueta "Output".

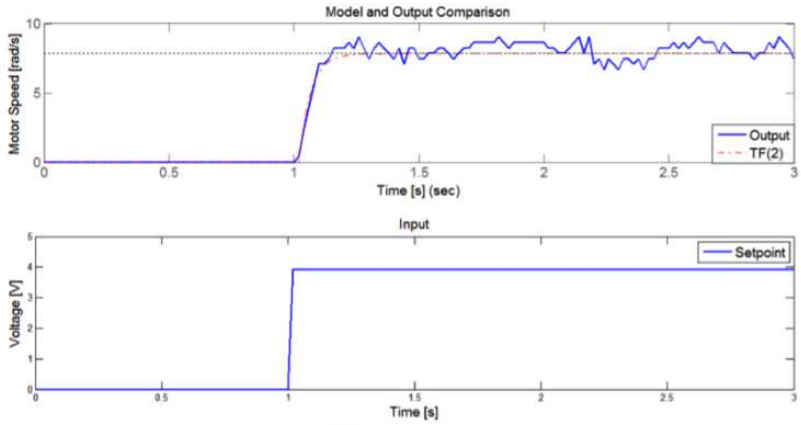


Figura 3.26. Respuesta del sistema modelado a una entrada de 3.92V[MABUCHI,2021]

Después de adquirir la respuesta en salida, es posible determinar una función de transferencia de primer orden que modele adecuadamente el sistema. Esta función se representa en la Ecuación 3.11

$$TF(s) = \frac{K_p e^{-\theta s}}{\tau s + 1} \tag{3.11}$$

Se requiere un análisis de la variación de salida en relación con la variación de la entrada para definir la ganancia estática Kp. Usando los datos adquiridos de la prueba escalonada el valor obtenido se aprecia en la Expresión 3.12

$$Kp = \frac{\Delta Y}{\Delta U} = 2 \tag{3.12}$$

El momento en que el sistema presenta el 95% de su valor final, llamado tiempo de asentamiento y representado por t5%, tiene relación con la constante de tiempo del sistema. Esto se aprecia en la Expresión 3.13

$$\tau = \frac{t_{5\%}}{3} = 0.0467 \tag{3.13}$$

Puede existir algún tiempo de inactividad en el sistema, lo cual se representa en la ecuación vista previamente como $e^{-\theta s}$. El tiempo muerto del sistema puede aproximarse a 0.02 segundos, pudiendo incrementarse si fuera necesario. Entonces, después del experimento y los cálculos, se puede modelar el sistema de forma óptima como se aprecia en la Ecuación 3.14.

$$TF(s) = \frac{K_p e^{-\theta s}}{\tau s + 1} = \frac{2e^{-0.02s}}{0.0467s + 1} \tag{3.14}$$

La respuesta del sistema modelado a la entrada escalón también se muestra en la Figura 3.26, en la curva con la etiqueta "TF (2)".

3.3.1. Sintonización PID

Con el modelado del motor realizado, se diseña en su totalidad el controlador para el sistema, el cual estará constituido como se ha detallado en la Figura 3.25. Dado que existen diversas formas de ajustar sus parámetros, se aplica de forma definitiva la regla de sintonización AMGO para este proceso [RITHIRUN, 2021]. Usando el modelo descrito por la Ecuación Final, es posible calcular los parámetros del controlador, con las expresiones descritas en las Ecuaciones 3.15, 3.16 y 3.17 respectivamente.

$$K_p = (\frac{1}{K_p})(0.2 + \frac{0.45\tau)}{\theta} \tag{3.15}$$

$$T_i = (\frac{0.8\tau + 0.4\theta}{0.1\tau + \theta})\theta \tag{3.16}$$

$$T_d = \frac{0.5\tau\theta}{\tau + 0.3\theta} \tag{3.17}$$

El controlador desarrollado se representa en tiempo continuo, pero el microcontrolador, responsable del control de lazo cerrado, funciona en tiempo discreto, por lo que la ley de control procede a ser discretizada. La estructura PID elegida permite una conversión directa al tiempo discreto, conservando las ganancias calculadas de forma eficaz.

3.3.2. Modelo cinemático de la plataforma móvil diferencial

El controlador PID diseñado se tomará para elaborar un controlador más complejo que permita controlar y regular al robot móvil en su totalidad, tomando como base los conceptos vistos en este capítulo y previamente en el estado del arte.

Este estudio considera una plataforma de robot móvil simple con dos ruedas motrices independientes y una rueda libre del echador. Se instala un bolígrafo a lo largo de una línea perpendicular entre el centro de las dos ruedas para representar una imagen. Dejar (q1, q2) conducir la velocidad angular de la rueda, r es el radio de la rueda motriz, L es la mitad de la distancia entre las dos ruedas y D es la perpendicular distancia desde la posición de la pluma hasta el centro de las dos ruedas, como se muestra en la Figura 1. Definimos (xb, yb) como la posición del centro de las dos ruedas y (x, y) como la posición de la punta del lápiz en el mundo marco {W}. El origen del bastidor base del robot {B} se asigna a la mitad de dos ruedas, y q es el ángulo de guiñada del robot móvil y es el ángulo de orientación desde (x, y) apuntando a (xb, yb) relativo al eje x del marco de referencia mundial {W}, como se muestra en la Figura 3.27.

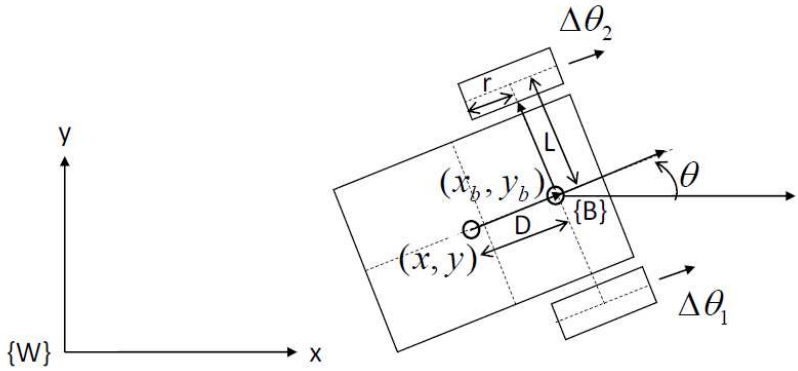


Figura 3.27. Cuadros combinados de la plataforma del robot móvil. [JAIN, 2017]

En la Figura 3.27. La referencia del observador externo viene dada por {W} y la base del robot por {B}.

Se definirá la velocidad linear instantánea como "v" vista desde el origen y "ω" como la velocidad angular de la referencia de la base del robot. La ecuación cinemática de velocidad de la plataforma del robot móvil se expresa en las Ecuaciones 3.18 y 3.19

$$\begin{pmatrix} \dot{x_b} \\ \dot{y_b} \\ \theta \end{pmatrix} = \begin{pmatrix} cos\theta & 0 \\ sen\theta & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} v \\ \omega \end{pmatrix} \tag{3.18}$$

$$\begin{pmatrix} v \\ \omega \end{pmatrix} = \begin{pmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{2L} & \frac{-r}{2L} \end{pmatrix} \begin{pmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{pmatrix} \tag{3.19}$$

Como es habitual en la mayoría de tipos de robots móviles con dos o más grados de libertad, esta cinemática no es omnidireccional. De esa forma, se esquematiza que la posición cartesiana de la cámara en este caso (x,y) es el punto representativo de la plataforma del robot móvil y se representa como se muestra en la Ecuación 3.20.

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_b \\ y_b \end{pmatrix} - D \begin{pmatrix} cos\theta \\ sen\theta \end{pmatrix} \tag{3.20}$$

Con ello, se obtiene la Ecuación 3.21.

$$\begin{pmatrix}\dot{x}\\ \dot{y}\end{pmatrix} = \begin{pmatrix}cos\theta & Dsen\theta\\ sen\theta & -Dcos\theta\end{pmatrix}\begin{pmatrix}v\\ \omega\end{pmatrix} \tag{3.21}$$

La cinemática de la plataforma del robot móvil está ahora definida como se aprecia en la Ecuación 3.22

$$\begin{pmatrix}\dot{x}\\ \dot{y}\end{pmatrix} = J(\theta)\begin{pmatrix}\dot{\theta}_1\\ \dot{\theta}_2\end{pmatrix} \tag{3.22}$$

Donde la matriz del Jacobiano $J(\theta)$ estará dada por la Expresión 3.23.

$$J(\theta) = \frac{r}{2}\begin{pmatrix}cos\theta + \rho sen\theta & cos\theta - \rho sen\theta\\ sen\theta - \rho cos\theta & sen\theta + \rho cos\theta\end{pmatrix} \tag{3.23}$$

Donde $\rho = D/L$. El determinante de la matriz del Jacobiano es una constante cuyo valor es: det $(J) = \rho r^2/2$. Por ello, no se debe colocar la cámara en un punto que equidiste de las llantas (a la mitad) garantizará una configuración de buena práctica, ya que $\rho > 0$ y que la matriz del Jacobiano esté bien generada según las condiciones establecidas.

Asimismo, la matriz inversa del Jacobiano se puede expresar como la Ecuación 3.24.

$$J^{-1}(\theta) = \frac{1}{r}\begin{pmatrix}cos\theta + \rho^{-1}sen\theta & sen\theta - \rho^{-1}cos\theta\\ cos\theta - \rho^{-1}sen\theta & sen\theta + \rho^{-1}cos\theta\end{pmatrix} \tag{3.24}$$

Con lo desarrollado, el control angular en el ángulo yaw ha sido transformado a control angular en el movimiento lineal lateral. Por ello, se puede apreciar que mientras se incremente el valor de ρ, aumenta la movilidad en el movimiento lateral.

Sin considerar la orientación angular, la plataforma del robot móvil se comportará como una plataforma omnidireccional en el plano cartesiano.

La descomposición del valor singular de la matriz del Jacobiano $J(\theta)$ se formula como se muestra en la Ecuación 3.25

$$J = U \sum V^T = (u_1 \quad u_2) \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix} \begin{pmatrix} v_1^T \\ v_2^T \end{pmatrix} \tag{3.25}$$

Donde el valor de cada parámetro mencionado se muestra en la Tabla 3.4

Tabla 3.4. Valores de parámetros

$\sigma_1 = \frac{r}{\sqrt{2}}$	$v_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$	$u_1 = \begin{pmatrix} cos\theta \\ sen\theta \end{pmatrix}$
$\sigma_2 = \rho \frac{r}{\sqrt{2}}$	$v_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} -1 \\ 1 \end{pmatrix}$	$u_2 = \begin{pmatrix} -sen\theta \\ cos\theta \end{pmatrix}$

Así, se determinan los valores singulares máximos y mínimos de la matriz del Jacobiano $J(\theta)$

$$\sigma_{max} = \begin{cases} 1\ ,\ 0 < \rho \leq 1 \\ \rho\ ,\ \rho > 1 \end{cases} \quad \text{y} \quad \sigma_{min} = \begin{cases} \rho\ ,\ 0 < \rho \leq 1 \\ 1\ ,\ 1 < \rho \end{cases}$$

Luego, la relación entre ambos permite obtener el número condición,

$$cond(J) = \frac{\sigma_{max}}{\sigma_{min}} = \begin{cases} \rho^{-1}\ , & 0 < \rho \leq 1 \\ \rho\ , & \rho > 1 \end{cases}$$

Cuando se da el caso que el valor de ρ es 1, el número condición es igual a 1 y las columnas de la matriz del Jacobiano son ortogonales y serán de igual magnitud para todas las posiciones del robot móvil. Por ello la plataforma del robot móvil y el sistema de la cámara describirán isotropía cinemática.

En el caso específico mencionado, cada rueda hace que el movimiento de la cámara sea igual y ortogonal al causado por la otra rueda. En este caso, se aprovecha al máximo los grados de libertad de las dos ruedas en el movimiento cartesiano, manteniéndolo estable.

3.3.3. Control de seguimiento de trayectoria y estabilidad

Como se indicó anteriormente, la posición de la cámara (x(t), y(t)) es el punto representativo de la plataforma móvil del robot, que debe seguir una trayectoria cartesiana referenciada $\left(x_r(t), y_r(t)\right), 0 \leq t \leq T$, en presencia del error inicial. Dado

que la matriz Jacobiana $J(\theta)$ está bien condicionada para todas las posiciones del robot móvil, se aplica la tasa resuelta basada en cinemática inversa más la ley de control proporcional para cumplir con el requisito de control de la trayectoria de la cámara y como capturará al objetivo que es la botella.

Se define $e_1(t) = x_r(t) - x(t)$ y $e_2(t) = y_r(t) - y(t)$, como los errores de seguimiento de trayectoria para cada par ordenado en el plano cartesiano. La ley de control de la tasa resuelta más el control P se expresa en la Ecuación 3.25

$$\begin{pmatrix} \dot{\theta}_1(t) \\ \dot{\theta}_2(t) \end{pmatrix} = J^{-1}(\hat{\theta}) \begin{pmatrix} \dot{x}_r(t) + k_1 e_1(t) \\ \dot{y}_r(t) + k_2 e_2(t) \end{pmatrix} \tag{3.25}$$

Donde $\hat{\theta}(t)$ es el ángulo $\theta(t)$ de orientación estimado, y $k_1\, y\, k_2$ son ganancias positivas $k_1, k_2 > 0$. A continuación, realizo la siguiente sustitución $\hat{\theta} = \theta + \delta$, de tal forma que se pueda efectuar $J(\theta)\, J^{-1}(\hat{\theta}) = \begin{pmatrix} cos\delta & sen\delta \\ -sen\delta & cos\delta \end{pmatrix}$, y el error dinámico se convierte en un SLVT, es decir un Sistema Lineal y Variante en el Tiempo, quedando expresado en la Ecuación 3.26.

$$\begin{pmatrix} \dot{e}_1(t) \\ \dot{e}_2(t) \end{pmatrix} = \begin{pmatrix} -k_1 cos\delta & -k_2 sen\delta \\ k_1 sen\delta & -k_2 cos\delta \end{pmatrix} \begin{pmatrix} e_1(t) \\ e_2(t) \end{pmatrix} + \begin{pmatrix} 1 - cos\delta & -sen\delta \\ sen\delta & 1 - cos\delta \end{pmatrix} \begin{pmatrix} \dot{x}_r(t) \\ \dot{y}_r(t) \end{pmatrix} \tag{3.26}$$

A continuación, se detallará la estabilidad y propiedades del sistema en lazo cerrado descrito previamente.

- Caso 1: Estabilidad exponencial global

En el caso en el que $\dot{x}_r(t) = \dot{y}_r(t) = 0$, el sistema cerrado presenta un equilibrio estable exponencial global cuando $(e_1, e_2) = 0$. Ello se basa en el uso de la función de Lyapunov para determinar estabilidad y se enuncia en la Ecuación 3.27.

$$V = \frac{e_1^2 + e_2^2}{2} > 0\ ,\ (e_1, e_2) \neq 0 \tag{3.27}$$

Se obtiene la derivada con respecto al tiempo, $\dot{V} = e_1\dot{e}_1 + e_2\dot{e}_2 = -cos\delta(k_1e_1^2 + k_2e_2^2) \leq -\alpha V$, donde $\alpha = 2cos\delta_{max}\min\{k_1, k_2\}$, lo cual indica que $cos\delta \geq cos\delta_{max} > 0$ y $k_1, k_2 > 0$.

- Caso 2: Estabilidad de entrada a estado

Debido a la capacidad de velocidad limitada de los motores, la velocidad de seguimiento de la trayectoria es limitada. Asumiendo que las derivadas de la trayectoria de referencia son limitadas, de modo que $|\dot{x}_r(t)| \leq m$ y $|\dot{y}_r(t)| \leq m$, donde m es un límite superior constante. Se establece que un sistema no forzado cuya solución trivial es globalmente exponencialmente estable es estable de entrada a estado (ISS) si se somete a una entrada acotada (o perturbación) [KHALIL, 2002]. Por lo tanto, el sistema cerrado en la Ecuación 9 es de entrada acotada estable en estado acotado (BIBS) estable. Además, los límites de los errores de seguimiento de trayectoria son aproximadamente: $|e_1| \leq k_1^{-1}\delta\, m$ y $|e_2| \leq k_2^{-1}\delta\, m$ según corresponde. Estos errores son directamente proporcionales a la velocidad de trayectoria m, error de medición δ y ganancias k_1^{-1} y k_2^{-1}.

3.3.4. Método de control de seguimiento en tiempo discreto

Se requiere implementar el control de seguimiento de trayectorias una vez se identifique el objeto con la cámara en tiempo discreto. Por ello, es necesario emplear los siguientes parámetros y ecuaciones. Tc será el tiempo de actualización del control de trayectorias, mientras que la trayectoria de la cámara referencial viene dada por $(x_r(k), y_r(k))$ en un tiempo discreto k. Asimismo, se considera el movimiento incremental generado expresado por $\Delta\Omega(k)$ y $\Delta S(k)$.

$$\begin{pmatrix} x_b(k) \\ y_b(k) \end{pmatrix} - D\begin{pmatrix} cos\theta(k) \\ sen\theta(k) \end{pmatrix} = \begin{pmatrix} x_r(k) \\ y_r(k) \end{pmatrix} \tag{3.28}$$

Donde

$$\begin{pmatrix} x_b(k) \\ y_b(k) \end{pmatrix} = \begin{pmatrix} x(k-1) \\ y(k-1) \end{pmatrix} + \begin{pmatrix} cos\theta(k-1) \\ sen\theta(k-1) \end{pmatrix}(D + \Delta S(k)) \tag{3.29}$$

Además, se emplea la Ecuación 3.30, que se muestra a continuación

$$\theta(k) = \theta(k-1) + \Delta\Omega(k) \tag{3.30}$$

Al reescribir la ecuación de cinemática inversa vista previamente se obtiene la Ecuación 3.31.

$$\begin{pmatrix} cos\theta(k-1) \\ sen\theta(k-1) \end{pmatrix}\big(D + \Delta S(k)\big) - D\begin{pmatrix} \cos\big(\theta(k-1) + \Delta\Omega(k)\big) \\ sen\big(\theta(k-1) + \Delta\Omega(k)\big) \end{pmatrix} = \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \tag{3.31}$$

Donde, se analiza la Ecuación 3.32.

$$\begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} x_r(k) \\ y_r(k) \end{pmatrix} - \begin{pmatrix} x(k-1) \\ y(k-1) \end{pmatrix} \tag{3.32}$$

Es necesario definir que existe un límite d_{max}, hasta el cual puede variar la posición de la cámara en una sola actualización de tiempo debido al continuo movimiento. Este límite se define tal que $|\Delta x| \leq d_{max}$ y $|\Delta y| \leq d_{max}$. Si $(x_r(k), y_r(k))$ dista mucho de $x(k-1), y(k-1))$, será necesario mover la posición del objetivo en cercanía a la actual posición de la cámara, teniendo que insertar uno o más puntos intermedios. Los términos de control $\Delta\Omega(k)$ y $\Delta S(k)$ se obtienen como se muestra en la Ecuación 3.33.

$$\Delta\Omega(k) = sen^{-1}\left(\frac{\Delta x sen\theta(k-1) - \Delta y cos\theta(k-1)}{D}\right) \tag{3.33}$$

$$\Delta S(k) = Dcos\Delta\Omega(k) - D + \Delta x cos\theta(k-1) + \Delta y sen\theta(k-1)$$

Los comandos de la posición incremental del motor vienen dados por la Ecuación 3.34 mostrada a continuación.

$$\begin{pmatrix} \Delta\Theta_1(k) \\ \Delta\Theta_2(k) \end{pmatrix} = \frac{1}{r}\begin{pmatrix} \Delta S(k) + L\Delta\Omega(k) \\ \Delta S(k) - L\Delta\Omega(k) \end{pmatrix} \tag{3.34}$$

También es necesario definir que $|\Delta\Theta_1(k)| \leq \Delta\Theta_{max}$ y $|\Delta\Theta_2(k)| \leq \Delta\Theta_{max}$, donde el valor máximo hace referencia al comando de posición incremental máximo posible de cada motor. Los valores de $\Delta\Theta_1(k)$ y $\Delta\Theta_2(k)$ se suman para usar dicho resultado como una referencia absoluta para los comandos de posición de los motores, que son posteriormente enviados a cada PID de forma independiente para controlar los motores dc en lazo cerrado. Cabe resaltar que el PID fue diseñado en la sección previa y se implementa aquí como un controlador ya diseñado en este trabajo de investigación.
A continuación, en la Figura 3.28 se observa el diagrama de bloques del sistema para el control propuesto de tal forma que permita seguir la trayectoria determinada por la cámara tras identificar las botellas.

Se emplean los parámetros $T_C = 20ms$, como el tiempo de actualización del control de trayectoria y $T_s = 1ms$, como el tiempo de muestreo para el control del controlador de los motores.

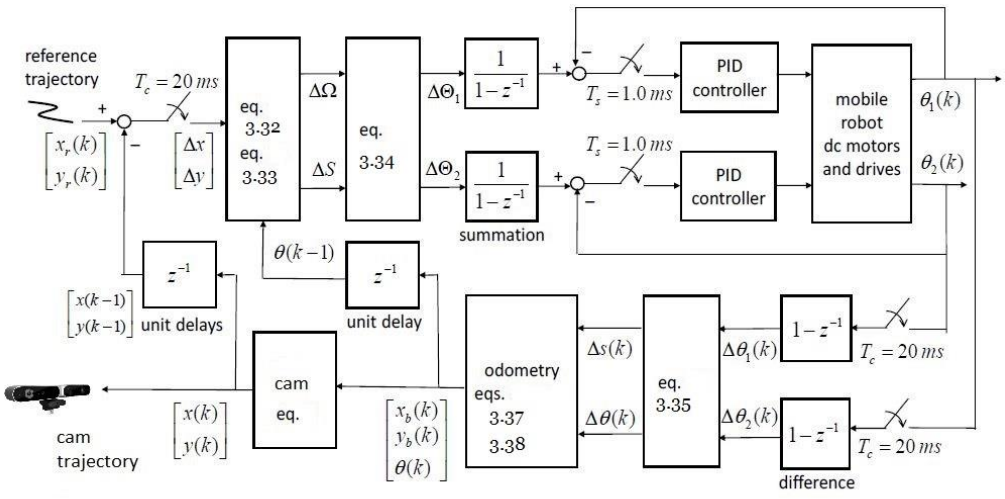


Figura 3.28. Diagrama de bloques del sistema del control propuesto

Se necesita conocer la posición del robot móvil y el ángulo yaw con el fin suministrar información al lazo de control para seguimiento de trayectoria descrito previamente. Asimismo, se emplea predicción odométrica de los encoders de los motores para computarizar la posición del robot y el ángulo yaw. El tiempo de actualización

odométrico también es 20ms. Los cambios de movimiento incremental son calculados desde la lectura respectiva de los encoders con la Ecuación 3.35.

$$\begin{pmatrix}\Delta S(k)\\ \Delta\theta(k)\end{pmatrix} = r\begin{pmatrix}\frac{\Delta\theta_1(k)+\Delta\theta_2(k)}{2}\\ \frac{\Delta\theta_1(k)-\Delta\theta_2(k)}{2L}\end{pmatrix} \tag{3.35}$$

Donde, se hace uso de la Ecuación 3.36

$$\begin{pmatrix}\Delta\theta_1(k)\\ \Delta\theta_2(k)\end{pmatrix} = \begin{pmatrix}\theta_1(k)-\theta_1(k-1)\\ \theta_2(k)-\theta_2(k-1)\end{pmatrix} \tag{3.36}$$

La posición base del robot móvil $(x_b(k), y_b(k))$ y el ángulo de orientación son actualizados según las Ecuaciones 3.37 y 3.38.

$$\begin{pmatrix}x_b(k)\\ y_b(k)\end{pmatrix} = \begin{pmatrix}x_b(k-1)\\ y_b(k-1)\end{pmatrix} + \begin{pmatrix}cos\theta(k-1)\\ sen\theta(k-1)\end{pmatrix}\Delta S(k) \tag{3.37}$$

$$\theta(k) = \theta(k-1) + \Delta\theta(k) \tag{3.38}$$

La localización del robot usando la odometría descrita será lo suficientemente precisa para permitir el movimiento del robot en la superficie requerida.

3.3.5. Generación de trayectoria

En complemento a lo mencionado previamente, es necesario determinar la trayectoria que seguirá el robot móvil cuando detecte la botella de plástico. En la Figura 3.29, se ilustra el esquema de detección de la imagen respecto al sistema de coordenadas relativas del móvil (V, U). También, se puede apreciar el sistema de coordenadas absolutas, que se definen como (Vo, Uo), las cuales se mantienen fijas y no cambian de dirección.

Se establece el ángulo "ɵ" como el ángulo que rotará el robot cuando localice la botella de tal forma que se alinee con el eje U o V según corresponda.

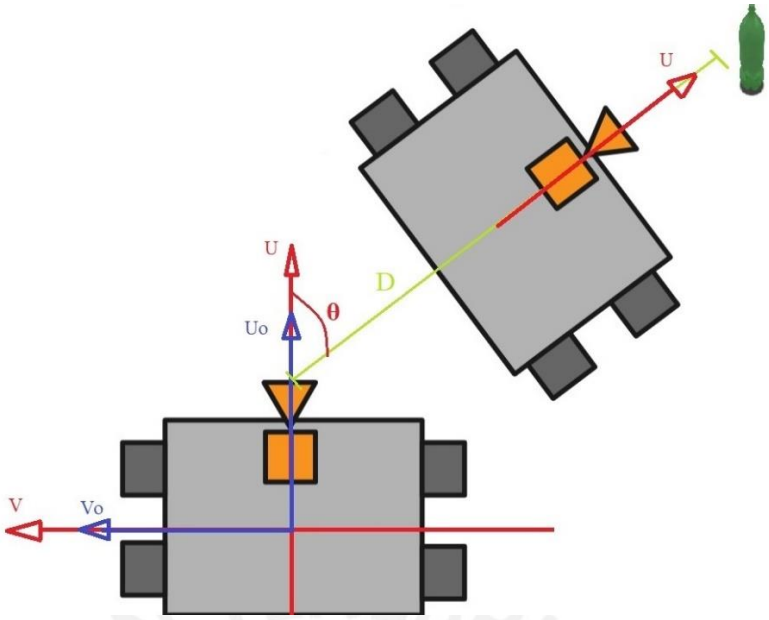


Figura 3.29. Esquema de orientación del robot móvil

Mediante la cámara RGB-D empleada, se puede obtener la distancia en profundidad hasta el objeto detectado. En este esquema está representado por la letra "D". Asimismo, en la Figura 3.30 se puede apreciar un esquema de lo visto por la cámara desde una perspectiva frontal. En este caso, se define la distancia en píxeles del eje focal a la imagen denominada "Dp". Esta variable, cuyo valor varía según la posición del objeto que detecte, permitirá obtener el ángulo necesario para girar el robót móvil y posteriormente desplazarse en línea recta hasta el objetivo.

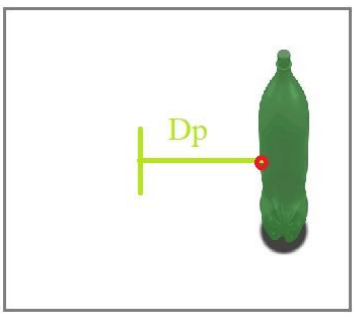


Figura 3.30. Vista frontal de la cámara RGB-D

Finalmente, en la Figura 3.31 se ilustra el esquema de captura de imagen de la cámara RGB-D. Se puede apreciar el foco representado por el punto de color anaranjado, la distancia en píxeles al plano de la imagen cuyo valor es constante y se representa por "Lp" y la distancia "Dp" mencionada previamente.

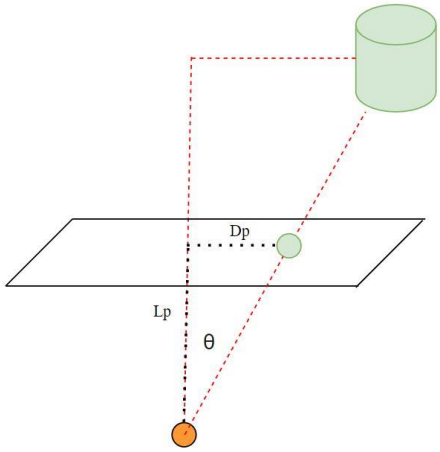


Figura 3.31. Esquema de captura de imagen de la cámara RGB-D

Con el esquema expuesto en la Figura 3.31 y lo revisado en las Figuras 3.29 y 3.30, se expresa el ángulo ɵ en términos de Lp y Dp como se enuncia en la ecuación 3.39

$$\theta = \tan^{-1}(\frac{Dp}{Lp}) \tag{3.39}$$

Con la obtención del ángulo mencionado, el sistema de control propuesto tendrá la información para integrar la posición relativa de la botella que se obtiene del sistema de visión y se determinará la trayectoria a seguir. Luego, el control de seguimiento de trayectoria seguirá esta línea.

3.4. Procesamiento de imágenes.

Se hace uso de Inteligencia Artificial para realizar el procesamiento de imágenes de manera óptima. Como se ha analizado en el marco teórico, es una estrategia que permite establecer un aprendizaje continuo al momento de tomar decisiones.

En este caso, se hace uso de una red neuronal del tipo RCNN. Esta red se caracteriza por aplicarse en modelos familiares a machine learning y tienen un uso particular en detección de objetos por lo que su uso toma más relevancia cada vez en el ambiente.

Se diseño una red neuronal convolucional de 6 capas, que permitan el entrenamiento de la red mediante imágenes capturadas en las playas de Lima, Perú, de tal forma que reconozca las botellas de plástico y personas cercanas.

Se hizo uso del entorno de Google Colab y programación en Python, para realizar el correcto entrenamiento de este tipo de redes orientadas hacia el aprendizaje continuo.

Sobre esta red neuronal convolucional, se trabajó con cuatro etapas y se siguieron los criterios de focalizar la botella plástica, identificarla y asimilar el aprendizaje en la red para una identificación posterior de otras botellas nunca antes vistas por la red. El sistema se configuró de esta forma y se realizó la identificación de elementos cercanos posibles encontrados en playas como bolsas de basura, personas, etc. Esto reforzó la capacidad de discernimiento de la RCNN.

En la Figura 3.32, que se puede apreciar a continuación, se puede ver la interacción con el interfaz mencionado. Es importante comentar, que las secciones de código están divididas por etapas de tal forma que se pueda identificar el error con precisión, si se ejecutará un solo script, no se sabría con exactitud los fallos encontrados.

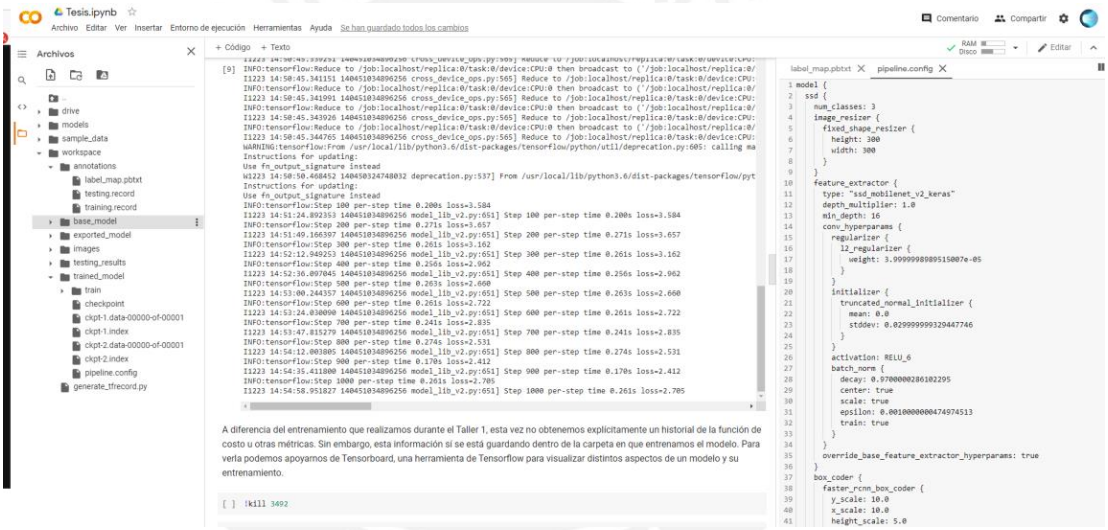


Figura 3.32. Foto en el entorno Google Colab

A continuación, se muestran algunas de las más de 150 imágenes que se tomaron para entrenar la red con un setup amplio en comparación de otras aplicaciones. En las imágenes tomadas se tuvieron en cuenta los conceptos mencionados en el marco teórico, la iluminación, la hora fue seleccionada específicamente de tal forma que permita capturar las imágenes con luz natural, pero sin interferir en la nitidez.

Figura 3.33. Foto propia en playa 1

Figura 3.34. Foto propia en playa 2

Figura 3.35. Foto propia en playa 3

Asimismo, en la Figura 3.36 se pueden apreciar parte de las fotos empleadas para entrenar la red neuronal convolucional RCNN.

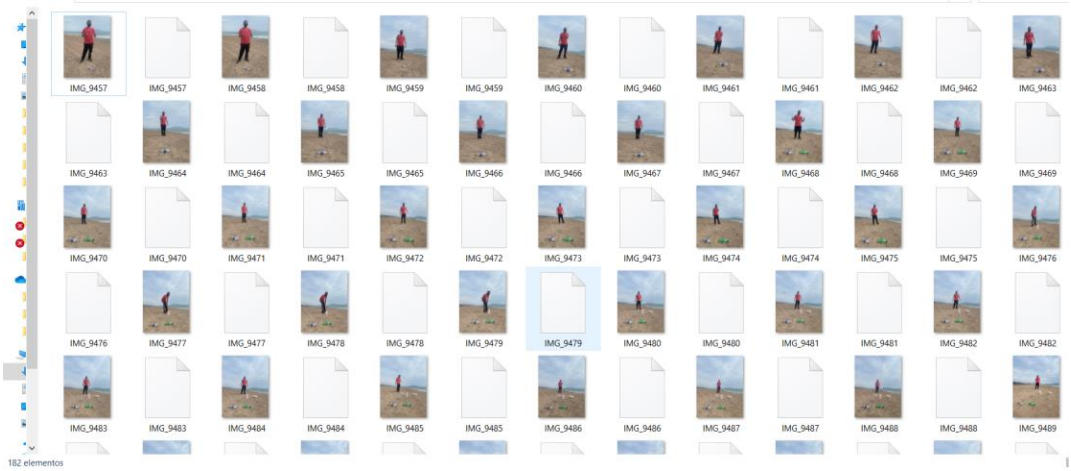


Figura 3.36. Foto de imágenes para el entrenamiento

Una vez que se tomaron todas las fotos, se procedió a realizar el etiquetado de foto por foto como se puede ver en la Figura 3.37, de tal forma que la red reciba el correcto entrenamiento al entregarle los objetos etiquetados sin fallo alguno. Esto es importante pues como se ha revisado anteriormente, la eficiencia de la red dependerá de la calidad del setup que se arme.

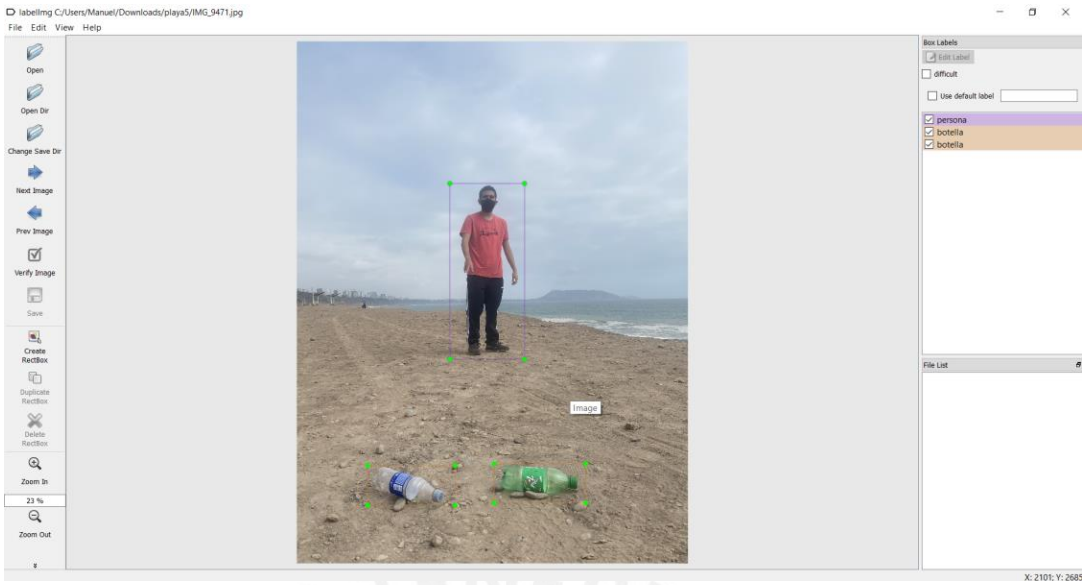


Figura 3.37. Entorno labelImg para generar los archivos .xml

CAPÍTULO 4

RESULTADOS

En el capítulo final, se presentan los resultados obtenidos en este trabajo de tesis con respecto a la identificación y reconocimiento de botellas de plástico empleando Inteligencia Artificial. Se hace uso de métricas de desempeño para evaluar la red RCNN y obtener resultados que validen el cumplimiento de los objetivos. Asimismo, se explora la tasa de aprendizaje de la red neuronal y el índice de pérdida de información, así como la relación de estos parámetros con el reentrenamiento de la red.

Con respecto a las propuestas de diseño para los dominios complementarios, se valida mediante simulación la integración de los dominios diseñados con el algoritmo desarrollado para la red RCNN.

4.1. Imágenes identificadas

Luego de ir a la orilla del mar y tomar cerca de 200 fotos, se entrenó la red neuronal convolucional RCNN. Este proceso se realizó de manera continua por dos días de tal forma que la red estuvo aprendiendo constantemente tanto de sus fallos como de sus aciertos. Desarrollando la capacidad de al existir más de una botella en el mismo punto ubicado en el espacio, poder identificarlas y agruparlas en la etiqueta “botella”.
Se obtuvieron índices de perdida menores a 10% en cada paso de ejecución de la red, lo cual es un índice de éxito alcanzado, pues significa que a nivel interno, la red ha sido entrenada correctamente y es por ello que se alcanzan los resultados esperados al detectar no solamente las botellas, sino también a las personas que estén dentro del campo visual designado para el trabajo de tesis.

Luego, se procedió a obtener los resultados desde Google Colab y exportarlos hacia una carpeta de destino local, a continuación se muestran los resultados obtenidos satisfactoriamente.

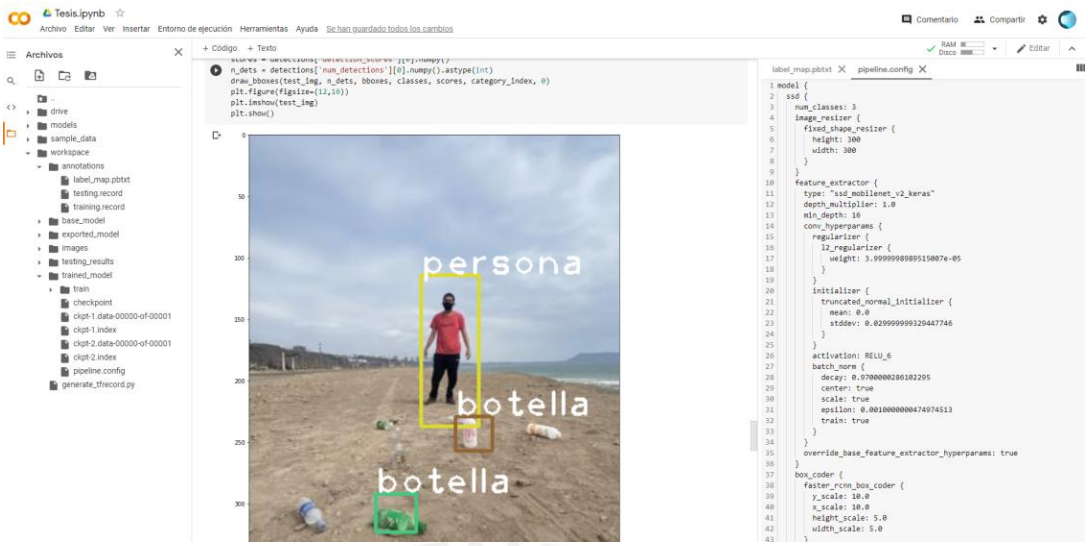


Figura 4.1. Resultados obtenidos en Google Colab

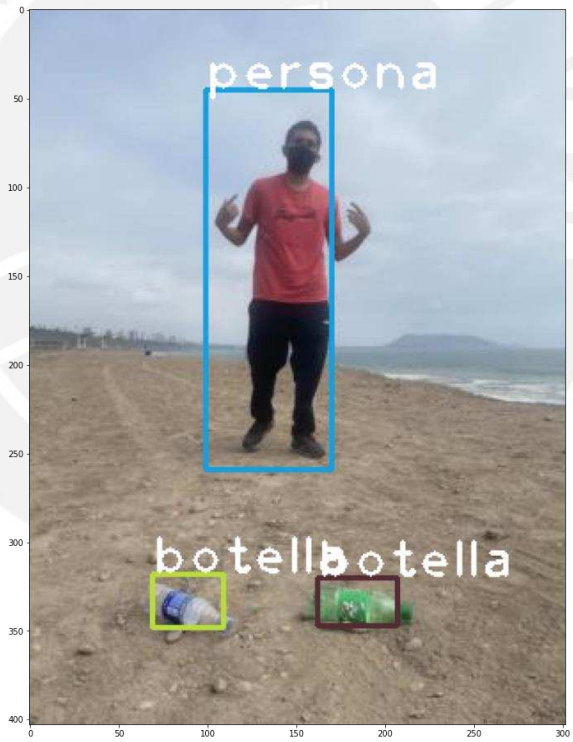


Figura 4.2. Resultados en la playa 1

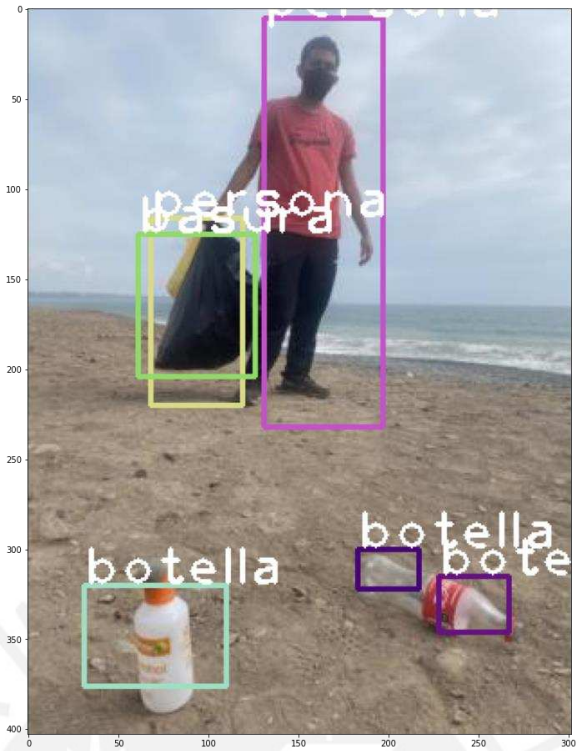


Figura 4.3. Resultados en la playa 2

Figura 4.4. Resultados en la playa 3

Figura 4.5. Resultados en la playa 4

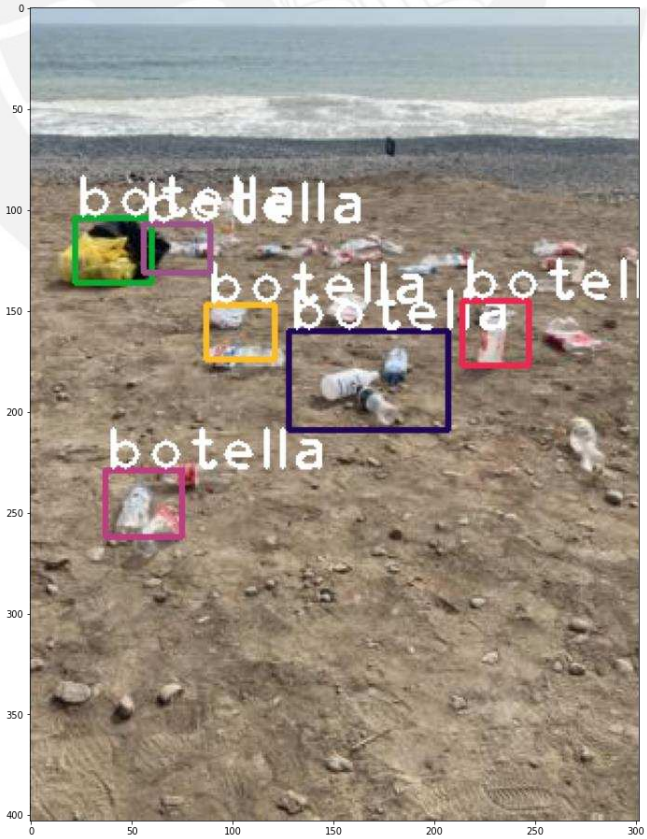


Figura 4.6. Resultados en la playa 5

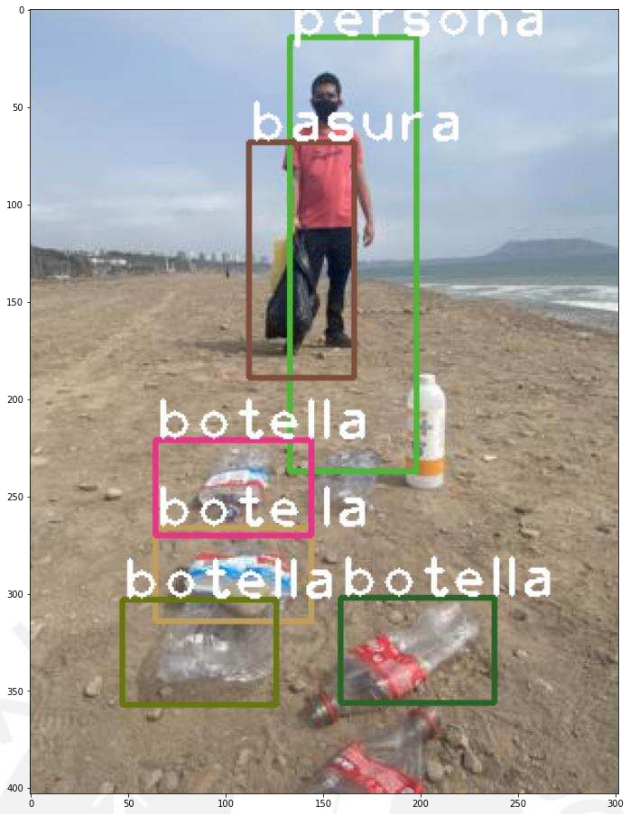


Figura 4.7. Resultados en la playa 6

Las siguientes imágenes son el resultado tras agrandar la imagen a su tamaño inicial.

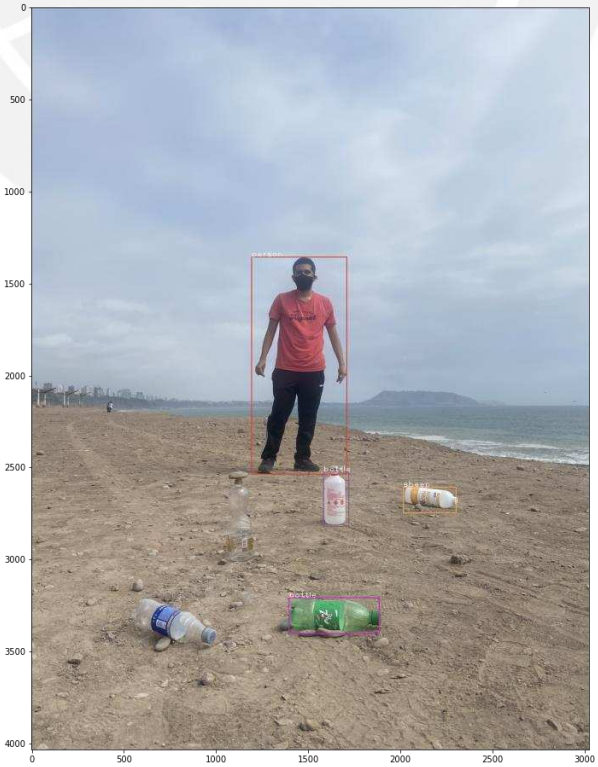


Figura 4.8. Resultados en la playa tamaño real

Figura 4.9. Resultados en la playa tamaño real 2

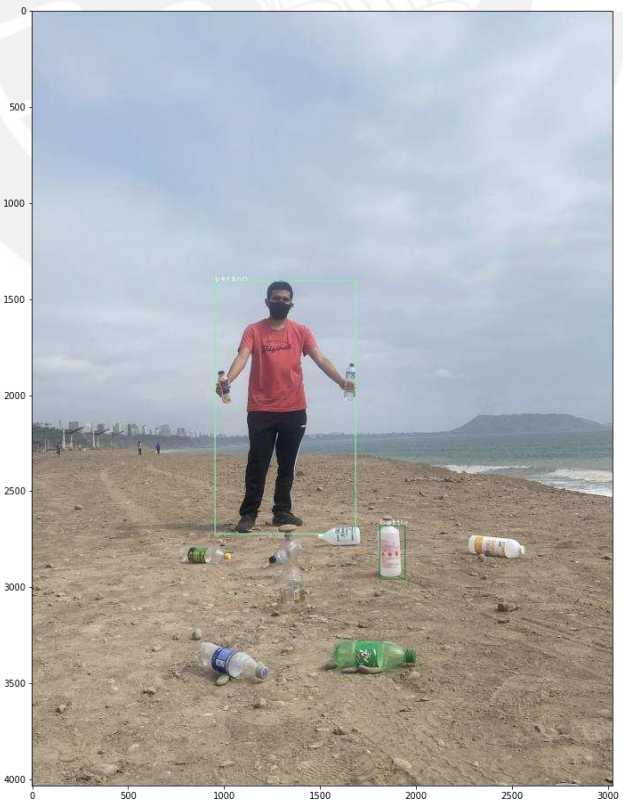


Figura 4.10. Resultados en la playa tamaño real 3

4.2. Métricas de desempeño

A continuación se presentan métricas de desempeño del entrenamiento realizado de la RCNN, las cuales evidencian el correcto aprendizaje y permiten realizar estimaciones y proyecciones en base a lo obtenido. En la Figura 4.11 se puede apreciar la tasa de aprendizaje. Esta tasa refleja como la red va aprendiendo conforme se agregan más imágenes al banco de entrenamiento hasta llegar a un 100% de aprendizaje. Cuando ya se empleó la totalidad de la muestra, la RCNN ya es capaz de identificar los objetos para los cuales se ha entrenado y al no tener conocimiento nuevo que adquirir, la gráfica disminuye gradualmente. Lo cual nos indica que la red es robusta ya que llegó a su pico de aprendizaje durante el entrenamiento.

Adicionalmente a lo mencionado, en la Figura 4.11 se puede observar que la pérdida total de información que la red no pudo procesar a lo largo de toda la muestra va disminuyendo, de tal forma que si se realiza una aproximación lineal o cuadrática, se puede apreciar que es una función decreciente. Esto indica que conforme la red se va entrenando con más muestras, se refuerza la identificación de los objetos seleccionados y la pérdida de información se va reduciendo gradualmente.

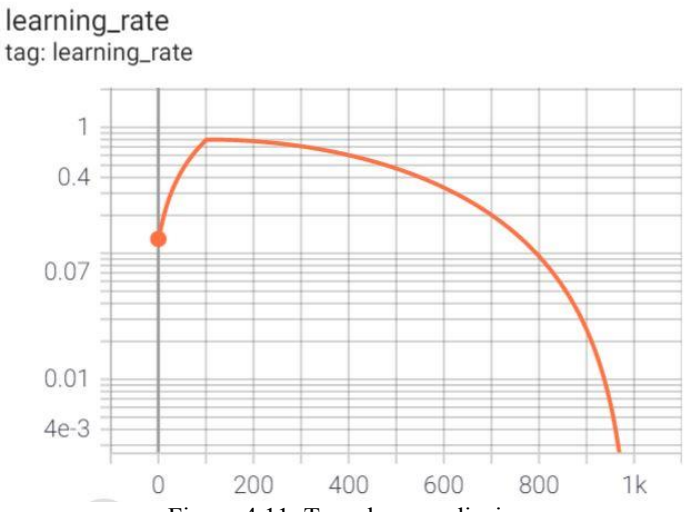


Figura 4.11. Tasa de aprendizaje

Finalmente, en la Figura 4.12 se puede apreciar como la pérdida de información es decreciente y tiende a cero. La RCNN se mantiene estable convergiendo a un valor mínimo de pérdida en todo momento. Esto demuestra, la robustez de la red realizada ya que cuenta con la propiedad de regulación y adaptación para sobreponerse a las

pérdidas de información, de tal forma que el aprendizaje no se vea afectado en ningun momento y así obtener el resultado óptimo.

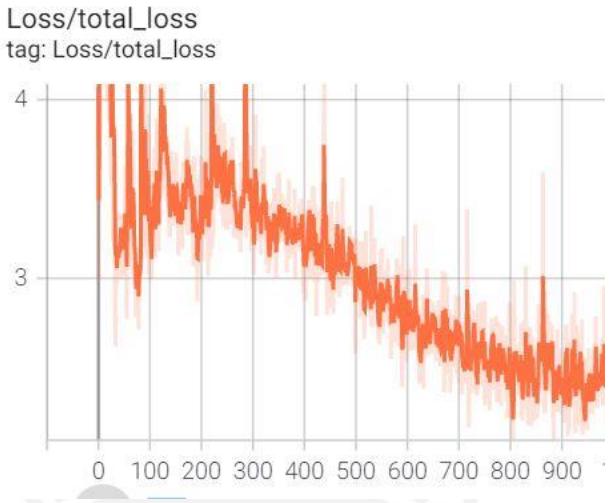


Figura 4.12. Pérdida total decreciente

Asimismo, en la Figura 4.13 se observa la regularización de la pérdida de información de la red. Es decir, el factor de compensación de la RCNN para recuperar la data faltante. Con esto, se refuerza el carácter predictivo de la red y se aprecia que la red ha aprendido correctamente, pues busca regularizar la pérdida de información con aprendizaje previo.

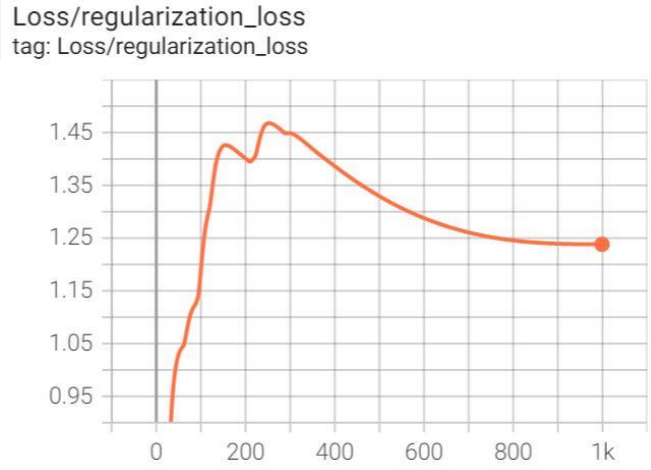


Figura 4.13. Regularización de la pérdida

4.3. Simulación y validación

En relación a la simulación, se validó la correcta identificación de las botellas, así como la capacidad de eludir obstáculos dada por los sensores de ultrasonido los cuales

verifican la presencia de un determinado objeto o persona en una proximidad definida. En la Figura 4.14 se aprecia un entorno simulado con paredes que representarían obstáculos ubicados en posiciones aleatorias en la playa y como el robot móvil logra identificar el objeto de atención en este caso las botellas y trazar una trayectoria hasta el mismo. El correcto reconocimiento de las imágenes integrado con el modelo cinemático diseñado en el capítulo 3 permitió que, en simulación, el robot móvil se desplace hacia el objetivo, es decir, las botellas plásticas.

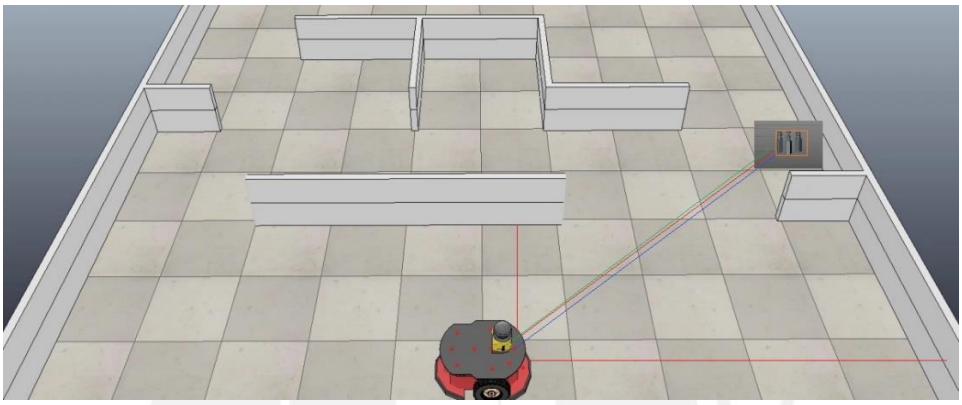


Figura 4.14 Simulación del móvil

Asimismo, se identificaron posibles trayectorias con la finalidad de reconocer las botellas plásticas, de tal forma que el robot móvil pueda desplazarse sin problemas identificando el objetivo y evadiendo posibles colisiones con obstáculos, basado en su visión artificial y el uso de los sensores de ultrasonido mediante los algoritmos detallados en anexos. En la Figura 4.15 se puede apreciar como reconoce la botella.

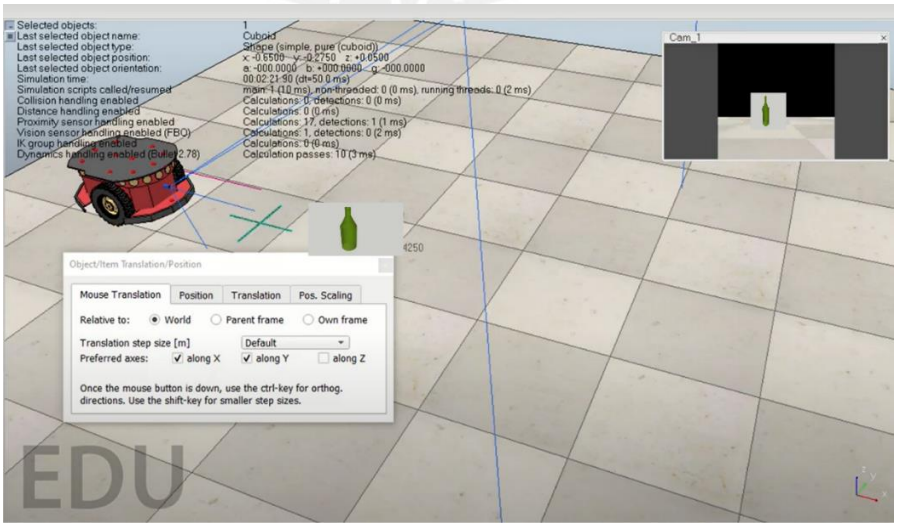


Figura 4.15 Visión de la botella

Adicional a la simulación y los resultados hallados, se hizo una prueba para hacer más aguda la detección de colores y formas, se mostraron a la cámara diferentes prismas de dimesiones similares como se puede ver en la Figura 4.16 y mediante el algoritmo de la RCNN entrenada se reconoció los objetos, se determinó la proximidad desde el robot movil y el acercamiento a cada objeto. Asimismo, se mejoró la detección de los sensores ultrasónicos complementarios.

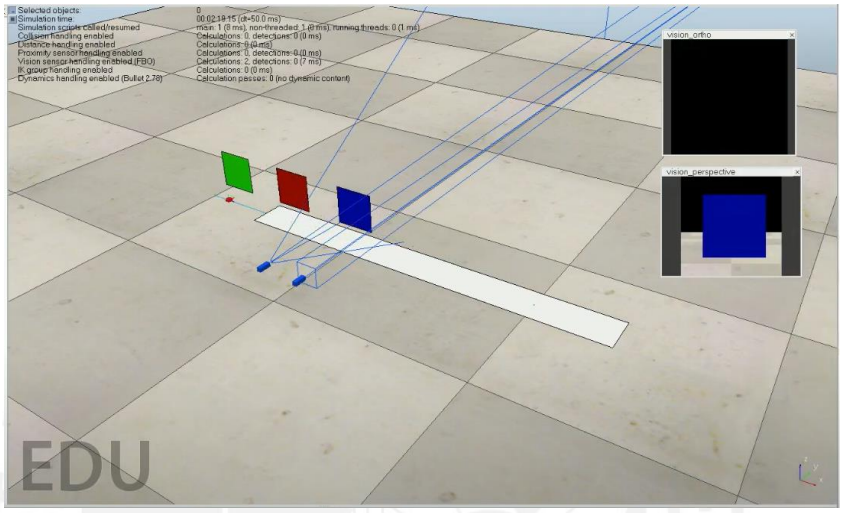


Figura 4.16 Prueba de sensores

En la Figura 4.17 se observa al robot móvil y su decisión de pese a tener otras trayectorias marcadas, decide seguir la trayectoria que acaba de percibir mediante la RCNN y la identificación de botellas. De tal forma, que seguir la visión artificial le permitirá acercarse a la botella reconocida verificando así la robustez de la RCNN diseñada y el correcto diseño del control al evadir obstáculos.

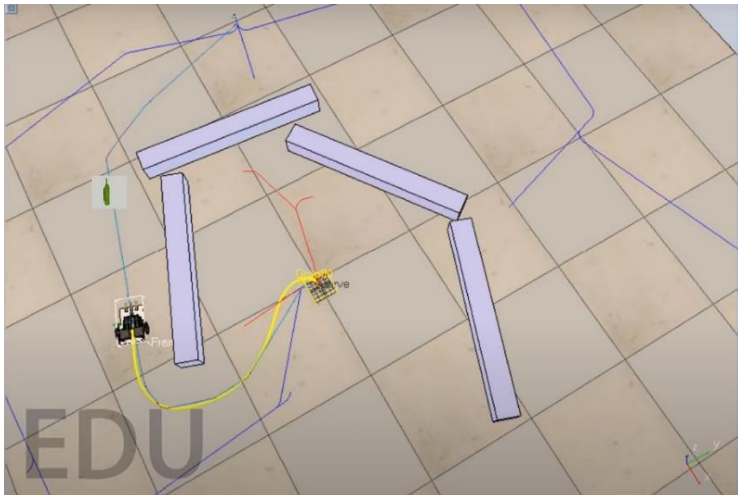


Figura 4.17 Trayectoria diferenciada

Finalmente, se realizó la simulación final, la cual consistió en ubicar 4 botellas en diferentes puntos del plano y permitir que el robot use el algoritmo diseñado de visión artificial y la red entrenada cuyos resultados se han observado previamente para poder definir una trayectoria y acercarse hasta la botella simulando la recolección de la misma.

En la Figura 4.18 se puede apreciar la trayectoria que realizó el robot y su distancia recorrida en el eje de coordenadas XY. Una vez que el robot móvil detectó la botella en cada punto B, C, D, E se dirigió hacia el objetivo para recolectarlo y una vez logrado el objetivo buscó la siguiente botella y así en forma cíclica por los puntos señalados. Cabe recalcar que los tramos de color verde significan que se realizó con éxito y en color rojo los que están pendientes por realizar por parte del robot móvil.

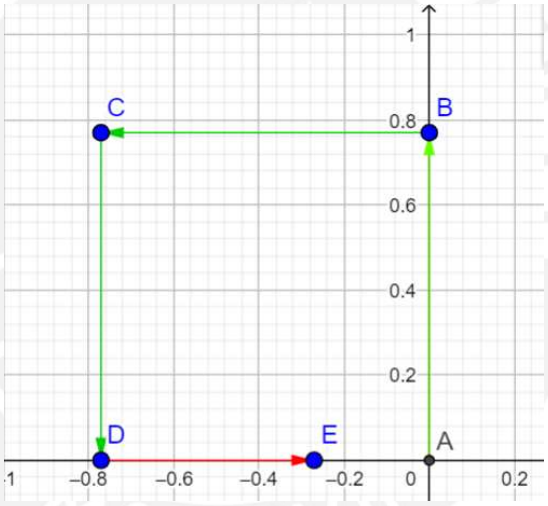


Figura 4.18 Trayectoria obtenida del robot móvil

De la simulación realizada, se obtuvieron las siguientes Figuras 4.19 y 4.20 donde se puede ver la distancia a la que se encuentra la botella vista desde el robot móvil y la rapidez del robot móvil con respecto al tiempo respectivamente.

Con el objetivo de explicar la Figura 4.19, es necesario mencionar que es la distancia medida respecto al robot y que inicialmente el robot detecta la botella hasta que la localiza en el espacio y luego se acerca hasta que la distancia entre ambos es 0. En esta Figura se ve que ocurre de forma cíclica que la distancia entre ambos "crece" hasta que la localiza por completo en el plano y una vez que la identifica empieza a acercarse reduciendo así la distancia hasta recolectarla. Este proceso se repite 3 veces

hasta que recolecta la que se encuentra en el punto D y de forma intencional se apagan los motores del robot al llegar al punto D, de tal forma que localiza la botella E y genera una trayectoria hacia esa botella, pero no puede avanzar, por lo cual se vuelve una trayectoria pendiente y ello explica el color rojo indicado en la Figura previa 4.18. Como se ha mencionado, a continuación, se puede apreciar la distancia a la cual se encuentra la botella a lo largo de la trayectoria señalada. Detecta una botella por vez, y con este resultado se confirma la robustez del proyecto ya que si se continuaran agregando más botellas de plástico seguiría recolectándolas una por una siguiendo el diseño establecido y con eficacia.

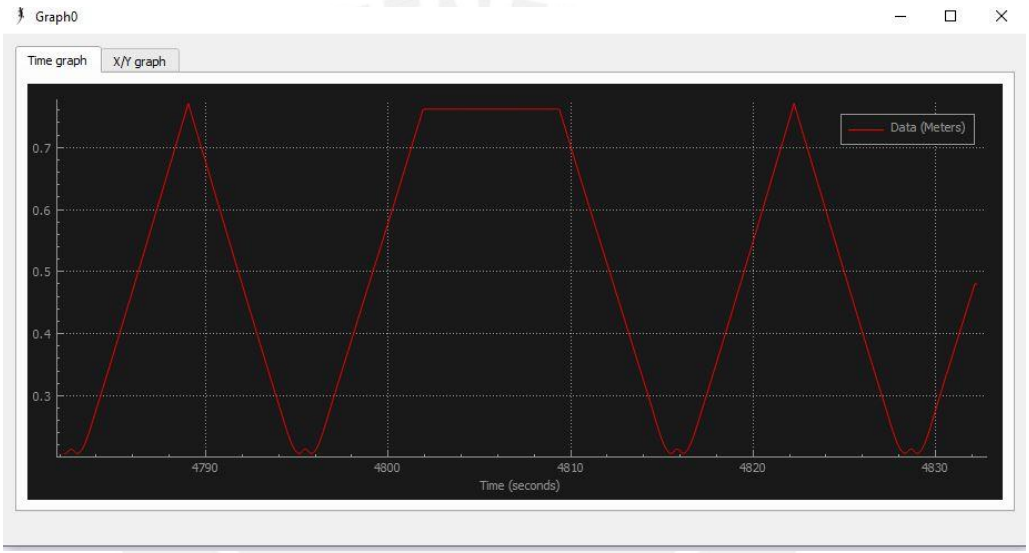


Figura 4.19. Distancia medida desde el robot a la botella

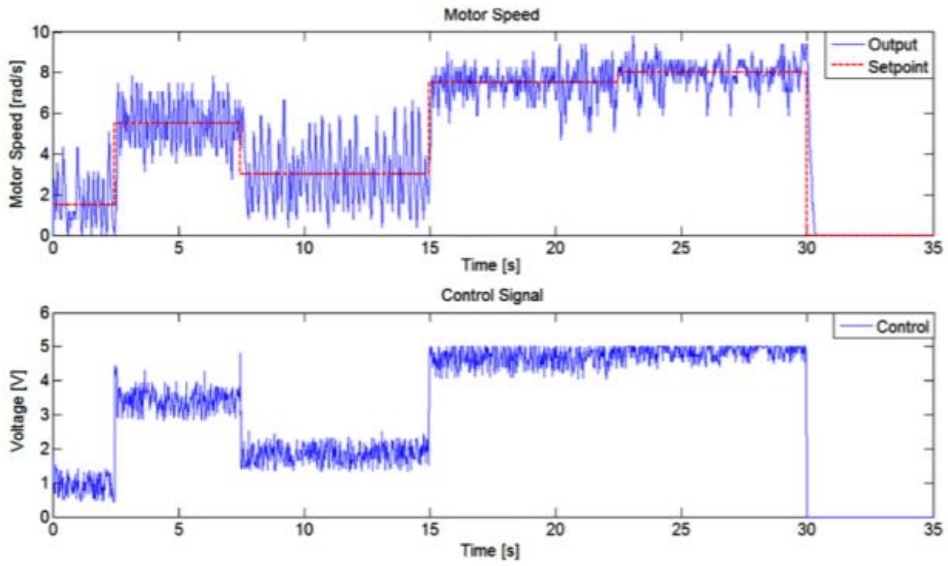


Figura 4.20. Gráficas de voltaje y velocidad del motor.

En la Figura 4.19, se puede ver que el valor de la rapidez es aproximadamente constante debido a que la distancia que recorre el robot móvil hasta la botella es la misma en intervalos de tiempo de valor similar. Con ello, se obtiene que la rapidez en la simulación del movimiento del robot móvil sea aproximadamente constante para evitar vibraciones y perturbaciones externas. En los tramos en los que detecta la botella y la identifica, el robot móvil se mantiene detenido y una vez que determina a dónde dirigirse basado en la identificación y reconocimiento de la botella, inicia el movimiento hasta que logra recolectar la botella. Esta experiencia se realizó constantemente con las 4 botellas que se situaron en su entorno confirmando una propiedad extra de valor agregado denominada adaptabilidad.

Las gráficas de velocidad del motor y del voltaje en la Figura 4.20, son un valor agregado para comprobar que los parámetros empleados son correctos ya que se siguen los sets points de forma correcta. Con esto, se pueden apreciar los resultados de la integración simulada de los diseños de los dominios que conforman el sistema mecatrónico. Se hace uso de cada uno de ellos de forma integral y se valida la integración por la relación intrínseca que existe entre el desempeño de un dominio en colaboración con los demás. Luego de que el robot móvil diseñado detecta las botellas plásticas, se acerca a ellas simuladamente haciendo uso de los demás dominios que constituyen el robot móvil.

OBSERVACIONES

- Se tomaron las primeras fotos para entrenar la red neuronal convolucional RCNN en el año 2020. En esa época, las playas se encontraban cerradas por el azote del virus COVID-19 y las medidas de prevención correspondientes de la cuarentena nacional frente a la pandemia. Se comunicó a las autoridades que se trataba de un trabajo de tesis de posgrado de la PUCP, así permitieron el ingreso para tomar las fotos respectivas y entrenar la red con fotos propias.

- Se comentó la existencia de este trabajo de tesis con especialistas de Chile. Este evento, incentivó la interacción entre diversos participantes de Latinoamérica e investigadores dedicados a Inteligencia Artificial. Luego, en una comunicación posterior, se presentaron los resultados obtenidos en esta tesis, lo cual despertó el interés de los investigadores, por el aporte realizado en el campo de visión artificial.

- En la concepción original del trabajo de tesis, se tenía pensado diseñar un robot móvil que se comporte como un robot anfibio que pueda desplazarse tanto en tierra como en agua. Sin embargo, con ayuda del Ing. Celso De La Cruz, se delimitó y replanteó el trabajo de tesis, encontrando un aporte significativo en el campo de la visión artificial.

- En las métricas empleadas para analizar el aprendizaje de la red, se puede apreciar que, si bien la información total perdida tiende a 0, existe ruido que recubre la señal. Esto se debe principalmente, a la cantidad de imágenes empleadas y la superposición de botellas en una imagen. Se disminuyó a 0 la información perdida con la premisa de que, si las botellas están muy juntas, la red las clasifica con una sola etiqueta. Si se quisiera eliminar el ruido por completo, se podría reentrenar la red ya existente con más imágenes.

- Se evaluó el algoritmo de visión artificial en términos de índice de procesamiento de datos. Se procesaron cerca de 200 imágenes en 6 horas, donde la red neuronal convolucional identificó con éxito las botellas de plástico, objetos y personas.

RECOMENDACIONES

- Formar parte de un grupo de investigación relacionado a Inteligencia Artificial desde una fase temprana. Esto sin duda, facilitará mucho el entendimiento de futuros proyectos de investigación, pues se tendrá familiaridad con los temas y conocimiento empírico importante para integrar al proyecto.

- Emplear grupos constructivos, tal que se mantenga el enfoque del trabajo en el objetivo principal. Es importante que, para conseguir el objetivo, se despeje el camino lo más posible de otras necesidades que si bien son importantes, no son el objetivo principal y por ende pueden ser distractores. El uso de grupos constructivos puede ser considerado para futuros proyectos.

- En caso de que el trabajo de investigación requiera implementación, buscar financiamiento en convocatorias como FONDECYT de Concytec, siempre que no exista una pandemia de por medio. Si un trabajo futuro, se sitúa en torno a una idea innovadora pero no está relacionado a diseño de equipos médicos para atención emergencista de COVID-19, no buscar el financiamiento, pues se dará prioridad a los equipos médicos. Se recomienda esperar un periodo de tiempo para que el financiamiento sea adquirido.

- No rendirse en ningún momento. Inevitablemente, existirán circunstancias adversas al trabajo de investigación, trabajo de tesis, o cual sea el trabajo futuro en el que se trabaje. Sin embargo, permanecer firme es un requisito indispensable, pues no puede existir un trabajo de tesis sin un autor, y para que exista un autor es necesario perseverancia, entereza y fuerza de voluntad.

CONCLUSIONES

1. El desarrollo del algoritmo de visión artificial usando Python, GoogleColab y Tensorflow como principal framework de IA, permitió diseñar y entrenar la RCNN para que sea capaz de detectar la ubicación de las botellas y reducir el índice de pérdida de información a 5% según los resultados de tensorflow.

2. La RCNN fue entrenada correctamente. La tasa de aprendizaje de la red neuronal convolucional desarrollada, llegó hasta 85% con cerca de 200 imágenes que se emplearon en el entrenamiento. Este indicador, cuyo valor actual permite que la red identifique las botellas, puede aumentar aún más si se emplean más imágenes, por lo que se concluye que la relación entre la tasa de aprendizaje y la cantidad de imágenes para entrenar la red es proporcional.

3. En imágenes donde aparecen personas y bolsas de basura, se logró distinguir los elementos diferentes y agrupar los elementos iguales cuando se superponen. El sistema que emplea el algoritmo elaborado logró clasificar cada tipo de objeto con el que se ha entrenado la red, teniendo un resultado óptimo, pues no solo identifica la botella que es el objeto de atención, sino que identifica y clasifica los otros elementos para mapear la zona y registrar los eventos que sucedan. Cabe recalcar que se podrá mejorar la efectividad aumentando el número de fotos para entrenar la red.

4. Existen tecnologías emergentes relacionadas al tema de tesis expuesto en este documento. Se encontró que la Inteligencia Artificial aplicada a proyectos integrales embebidos participa en ámbitos comerciales y académicos, por lo que su aplicación en este aspecto es muy importante. Asimismo, se exploró la necesidad de autonomía para el robot móvil, tomando como base el ambiente al cual estará expuesto y las condiciones adversas que puedan suceder para la concepción de su diseño y desarrollo.

5. Se elaboraron las propuestas de diseño en los diversos dominios: mecánico, electrónico, procesamiento de imágenes y de control, y se validó su integrados en un sistema mecatrónico mediante simulación en V-REP. Con ello, se logró el

objetivo de relacionar la RCNN con las propuestas de diseño de para cumplir la tarea designada.

6. Se simuló con éxito un entorno donde el robot identificó una botella de plástico situada a una distancia arbitraria inicial y se acercó hasta que su distancia fue mínima. En complemento, como valor agregado, se realizó esta experiencia para 4 botellas consecutivamente de tal forma que se pudo monitorear la velocidad del robot y la trayectoria limpia sin colisionar con otros objetos, validando así el correcto uso de los sensores de ultrasonido como esquema de visión extra y también se validó el sistema de visión artificial principal.

BIBLIOGRAFÍA

[MATEUS, 2013] Mateus, S. (2013) Responsabilidad Social Individual, un tema de todos. Recuperado de http://blogs.eltiempo.com/liderazgoarriba/2013/09/06/responsabilidad-social- individual-un-tema-de-todos/

[HERNANDEZ, 2019] Hernandez Muro, A. (2019) Perú: el 90% del reciclaje de plásticos es informal. https://www.sophimania.pe/medio-ambiente/contaminacion-y-salud-ambiental/pera-el-90-del-reciclaje-de-plasticos-es-informal/ (02/11/2019).

[ECHEVARRÍA, 2015] M. Echevarría, "Metodología de diseño conceptual modular para la selección de variables modulares", Doctorado, Universitat Politècnica de Catalunya, 2015.

[BARRUELA, 2018] O. Barrueta, E. Chaparro and T. Tello, "Aplicación de la metodología Design Thinking para el diseño de una propuesta de valor para el networking profesional", Maestría, Universidad de Piura, 2018.

[MACARTHUR, 2019] Ellen MacArthur Foundation (2016) The New Plastics Economy, Rethinking the Future of Plastics. Recuperado 12 de noviembre 2019, https://www.ellenmacarthurfoundation.org/publications/the-new-plastics- economy- rethinking-the-future-of-plastics.

[MINAM, 2020] MINAM, "Cifras del mundo y el Perú", Menos Plástico Más Vida, 2020. [Online]. Available: https://www.minam.gob.pe/menos-plastico-mas-vida/cifras-del-mundo-y-el-peru.

[FRANCEINFO, 2014] franceinfo. "Les gyres, continents de plastiques au sein des océans". (2014) https://www.francetvinfo.fr/. https://www.francetvinfo.fr/monde/les-gyres-continents-de-plastiques-au-sein-des-oceans_3069349.html

[LA REPÚBLICA, 2018] Redacción La República (2018) ¿Cuáles son los distritos más contaminados de Lima? https://larepublica.pe/sociedad/1248037-lima-conoce-seis-distritos-contaminados (28/12/2018).

[MINAM, 2019] MINAM (2019) El plástico representa el 10% de todos los residuos que generamos en el Perú. http://www.minam.gob.pe/notas-de-prensa/minam-el- plastico-representa-el-10- de-todos-los-residuos-que-generamos-en-el-peru.

[POSE, 2013] Pose, "DISEÑO DE UNA METODOLOGÍA PARA LA IMPLANTACIÓN DE TÉCNICAS DEL VALOR EN SISTEMAS INTEGRADOS DE DISEÑO Y FABRICACIÓN", Doctorado, Universidad De Vigo, 2013.

[ANDINA, 2019] ANDINA (2019). "Conozca a IRBin, el robot que enseña cómo reciclar los residuos". https://andina.pe. https://andina.pe/agencia/noticia-conozca-a-irbin-robot-ensena-como-reciclar-los-residuos-756731.aspx (accedido el 21 de junio de 2020).

[PUNTOEDU, 2019] PuntoEdu (2019). "Sullkapata: el hermano de la playa - PuntoEdu PUCP". PuntoEdu PUCP. https://puntoedu.pucp.edu.pe/noticia/sullkapata-el-hermano-de-la-playa/ (accedido el 4 de abril de 2020).

[DRONYX, 2018] Solarino Sand Beach Cleaner Robot. (s. f.). DronyX - A Mobile Robotics Company. https://www.dronyx.com/solarino-beach-cleaner/

[KONG, 2018] S. Kong, M. Tian, C. Qiu, Z. Wu and J. Yu, "IWSCR: An Intelligent Water Surface Cleaner Robot for Collecting Floating Garbage," in IEEE Transactions on Systems, Man, and Cybernetics: Systems, doi: 10.1108/TSMC.2019.2961687.

[ZHIHONG, 2017] C. Zhihong, Z. Hebin, W. Yanbo, L. Binyan and L. Yu, "A vision-based robotic grasping system using deep learning for garbage sorting," 2017 36th Chinese Control Conference (CCC), Dalian, 2017, pp. 11223-11226, doi: 10.23919/ChiCC.2017.8029147.

[TAKAOKA, 2015] R. Takaoka and N. Hashimoto, "Depth Map Super-Resolution for Cost-Effective RGB-D Camera," 2015 International Conference on Cyberworlds (CW), 2015, pp. 133-136, doi: 10.1109/CW.2015.32.

[BOTHE, 2018] K. Bothe, A. Winkler and L. Goldhahn, "Effective Use of Lightweight Robots in Human-Robot Workstations with Monitoring Via RGB-D-Camera," 2018 23rd International Conference on Methods & Models in Automation & Robotics (MMAR), 2018, pp. 698-702, doi: 10.1109/MMAR.2018.8486036.

[ALEJO, 2017] D. Alejo, F. Caballero and L. Merino, "RGB-D-based robot localization in sewer networks," 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 4070-4076, doi: 10.1109/IROS.2017.8206263.

[CHAN, 2018] J. S. P. Siy, R. K. C. Chan and R. G. Baldovino, "Implementation of a Real-time Appearance-based Mapping in a Fully Autonomous Urban Search Robot," 2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology,Communication and Control, Environment and Management (HNICEM), Baguio City, Philippines, 2018, pp. 1-4, doi: 10.1109/HNICEM.2018.8666380.

[BECK, 2015] S. Beck and B. Froehlich, "Volumetric calibration and registration of RGB-D-sensors," 2015 IEEE Virtual Reality (VR), Arles, 2015, pp. 151-152, doi: 10.1109/VR.2015.7223340.

[LIU, 2011] J. Liu, P. Chen, S. Tang and H. Gao, "Theoretical and experimental research on lugged wheel performance for wheel mobile robot on loose sand," 2011 IEEE International Conference on Robotics and Biomimetics, 2011, pp. 614-619.

[PERTUZ, 2017] S. Pertuz and J. Kamarainen, "Region-based depth recovery for highly sparse depth maps," 2017 IEEE International Conference on Image Processing (ICIP), Beijing, 2017, pp. 2074-2078, doi: 10.1109/ICIP.2017.8296647.

[LLINARES, 2019] Llinares Llopis, R, “Señal Banda Base y Densidad Espectral de Potencia Asociada”, 2019

[YUJILEDS, 2013] Understanding Luminous flux (lumen) and Illuminance (lux)_ YUJILEDS. (s. f.). High CRI LED Leader - YUJILEDS. https://www.yujiintl.com/blog/understanding-luminous-flux-(lumen)-and-illuminance-(lux).html

[MA, 2002] L. Ma and K. Khorasani, "Application of adaptive constructive neural networks to image compression," in IEEE Transactions on Neural Networks, vol. 13, no. 5, pp. 1112-1126, Sept. 2002, doi: 10.1109/TNN.2002.1031943.

[MATLAB, 2015] MATLAB Login | MATLAB & Simulink. (s. f.). MATLAB Login | MATLAB & Simulink. https://matlab.mathworks.com/

[COPPELIASIM, 2012] Robot simulator CoppeliaSim: create, compose, simulate, any robot - Coppelia Robotics. (s. f.). Robot simulator CoppeliaSim: create, compose, simulate, any robot - Coppelia Robotics. https://www.coppeliarobotics.com/

[LIU, 2016] Y. Liu, Q. Zhao, H. Zhang, R. Xu, Y. Li and L. Wei, "A New Method to Predict RNA Secondary Structure Based on RNA Folding Simulation," in IEEE/ACM Transactions on Computational Biology and Bioinformatics, vol. 13, no. 5, pp. 990-995, 1 September 2016.

[SHI, 2019] J. Shi, Y. Zhou and W. X. Q. Zhang, "Target Detection Based on Improved Mask Rcnn in Service Robot," 2019 Chinese Control Conference (CCC), 2019, pp. 8519-8524, doi: 10.23919/ChiCC.2019.8866268.

[OLLERO, 2001] Ollero Baturone, Aníbal, 2001, "Robótica: Manipuladores y robots móviles.

[SONG, 2010] Song Gu and Zhou Hulin, "The design of image acquisition and display system," 2010 2nd International Conference on Education Technology and Computer, 2010, pp. V5-23-V5-26.

[SHANGTAI, 2007] J. Shangtai, H. Zhongsheng and W. Weihong, "Model-free Adaptive Control Used in Permanent Magnet Linear Motor," 2007 Chinese Control Conference, 2007, pp. 748-751.

[RITHIRUN, 2021] C. Rithirun, A. Charean and W. Sawaengsinkasikit, "Comparison Between PID Control and Fuzzy PID Control on Invert Pendulum System," 2021 9th International Electrical Engineering Congress (iEECON), 2021, pp. 337-340.

[KHALIL, 2002] Khalil, H.K. Nonlinear Systems, 3rd ed.; Prentice-Hall: Upper Saddle River, NJ, USA, 2002.

[JAIN, 2017] Jain, S.; Gupta, P.; Kumar, V.; Sharma, K. A force-controlled portrait drawing robot. In Proceedings of the IEEE
International Conference on Industrial Technology (ICIT), Seville, Spain, 17–19 March 2017;

[JOURNEY, 2019] Journey Perú (2019) ¿Como se usa la nueva EcoBox?: cinco pasos para reciclar y recibir descuentos. https://www.cocacoladeperu.com.pe/historias/medio-ambiente-c-mo-se-usa-la-nueva-ecobox-cinco-pasos-para-reciclar-y-recibir-descuentos

[LAIGLE, 2013] Laigle, L. (2013) Pour une transition écologique à visée sociétale. Mouvements. 2013/3, n°75, p.135-142.

[LPR, 2019] Le recyclage de la résine PET, un modèle d'économie circulaire. Recuperado 2 febrero 2019, de http://www.lpr-pet.fr/fr_FR/contenu/81/Une-production- ecologique.html

[MCCLENAGHAN, 2019] McClenaghan, M. (2019) Investigation: Coca Cola and the 'fight back' against plans to tackle plastic waste. https://unearthed.greenpeace.org/2017/01/25/investigation-coca- cola-fight- back-plans-tackle-plastic-waste/ (09/11/2019)

[ADEME, 2018] Agence de l'Environnement et de la Maitrise de l'Energie (ADEME) (2018) Bilan des connaissances économiques et environnementales sur la consigne des emballages boissons et le recyclage des emballages plastiques. Recuperado de https://www.ademe.fr/sites/default/files/assets/documents/67688_bilan_cons igne_rdc_ synthese.pdf

[MABUCHI, 2021] "MABUCHI MOTOR CO., LTD". MABUCHI MOTOR CO., LTD. https://www.mabuchi-motor.com/ (accedido el 22 de abril de 2021).

[COGNEX, 2018] "COGNEX | Introducción a la Visión Artificial". Cognex | Guía para automatización de procesos y mejorar la calidad. http://www.cognex.com (accedido el 30 de mayo de 2021).

[LL, 2021] "Nvidia Jetson Nano: una computadora para la implementación de aplicaciones AI". Bienvenidos al "Laboratorio Linux" (LL). https://laboratoriolinux.es/index.php/-noticias-mundo-linux (accedido el 15 de junio de 2021).

[EDMANS, 2006] Grupo de investigación EDMANS, Técnicas y algoritmos básicos de visión artificial. La Rioja: Universidad de La Rioja. Servicio de Publicaciones, 2006.

ANEXOS

A continuación, se presentan los algoritmos empleados para el desarrollo de la red RCNN y el procedimiento ejecutado para obtener un óptimo resultado.

- Anexo A: Filtro Gaussiano

```
import cv2
# Load image
image_path = 'examples/images/bottle1jpg'
image = cv2.imread(image_path)
# Gaussian blur
k = 5
sigma = 0
blur = cv2.GaussianBlur(image, (k, k), sigma)
# Show
cv2.imshow('Original', image)
cv2.imshow('Filtered', blur)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

- Anexo B: Detección de esquinas

```
function [] = Corners(img, m, mu, sigma, N, th)
img = imread(img);

%check if the image is rgb.
%if it is, convert it to grayscale
if size(img, 3) == 3
img = rgb2gray(img);
End

%Perform gaussian filtering on the image
img = GaussianFilter(img, m, mu, sigma);
%compute the gradients
 [Ex, Ey] = Gradient(img);
%compute the list of eigen values with corresponding x and y
%coordinates, then filter out all the overlapping Eigen values (by
%flagging them. Then clean out all the flags, to leave only the corner
%feature points. Then highlight these corner feature points on the
%image and lastly, show the image.
imshow(uint8(highlight(img, clean(filter(compute(Ex, Ey, N, th), N)))));
end

%displays feature points from pos on image
function [img] = highlight(img, pos)
%get size of pos
 [rows, ~] = size(pos);

%iterate over pos, setting each corner pixel to white in the image
for i = 1:rows
img(pos(i,1),pos(i,2)) = 255;
end
```

```
%convert image to uint8
img = uint8(img);
end


%puts all non-flagged x and y coordinates into a new matrix, and returns
%that
function [newImg] = clean(pos)
%get size of pos
 [rows, ~] = size(pos);

%keep count (for adding to newImg)
count = 1;

%container for non-flagged entries
newImg = [];

%iterate over all rows in pos
for i = 1:rows
%if an entry is not -1, then add it to the new container and
%increment count
if pos(i,1) ~= -1
newImg(count, 1) = pos(i,2);
newImg(count, 2) = pos(i,3);
count = count + 1;
end
end
end

%flags the overlapping entries for later removal
function [pos] = filter(pos, N)
%get size of pos
 [rows, ~] = size(pos);

%iterate over rows of pos
for i = 1:rows
%get the row and column values from pos
curr_row = pos(i, 2);
curr_col = pos(i, 3);

%if the entry is not -1, then process. Otherwise, skip.
if pos(i,1) ~= -1

%iterate over all remaining rows
for j = (i+1):rows
%get the row and column values at j
r_row = pos(j, 2);
r_col = pos(j, 3);

%check the boolean conditions at the boundaries
rowCond = ((r_row <= (curr_row + N)) && (r_row >= (curr_row - N)));
colCond = ((r_col <= (curr_col + N)) && (r_col >= (curr_col - N)));

%make sure both conditions are true in order to flag for
%removal
if rowCond && colCond
pos(j,1) = (-1);
pos(j,2) = (-1);
pos(j,3) = (-1);
```

```
end
end
end
end
end


function [pos] = compute(Ex, Ey, N, th)
%compute the size of the neighborhood
SIZE = (2*N) + 1;

%compute the neighborhoods via im2col - for both the X and Y gradient
%images
Nx = im2col(padarray(Ex, [N N]), [SIZE SIZE], 'sliding');
Ny = im2col(padarray(Ey, [N N]), [SIZE SIZE], 'sliding');

%get the sizes
[~, col] = size(Nx);
[Erow, Ecol] = size(Ex);

%pos is a container for the eigen, row, and column values
pos = [];

%eigs is used for storing eigen values with the purpose of
%normalization
eigs = [];

%used for tracking values
count = 1;
x = 0;
y = 1;

%iterate over all columns
for i = 1:col
%reconstruct the neighborhood
tempX = reshape(Nx(:,i),SIZE, SIZE);
tempY = reshape(Ny(:,i), SIZE, SIZE);

%compute the sum of pixel-wise multplication between the X and Y
%gradient neighborhoods
combo = sum(sum(tempX.*tempY));

%compute the sum of pixel-wise squaring between in the X
%gradient neighborhoods
Xsq = sum(sum(tempX.^2));

%compute the sum of pixel-wise squaring between in the Y
%gradient neighborhoods
Ysq = sum(sum(tempY.^2));

%Construct C
C = [Xsq, combo; combo, Ysq];

%Find and keep track of the minimum Eigen values
eigs = [eigs min(eig(C))];
end

%take the range of computed Eigen values
Range = max(eigs) - min(eigs);
```

```
for i = 1:col
%keep track of x and y values corresponding to the Eigen values
x = x + 1;
if mod(i, Erow) == 0
x = 1;
y = y + 1;
end

%same as above
tempX = reshape(Nx(:,i),SIZE, SIZE);
tempY = reshape(Ny(:,i), SIZE, SIZE);
combo = sum(sum(tempX.*tempY));
Xsq = sum(sum(tempX.^2));
Ysq = sum(sum(tempY.^2));
C = [Xsq, combo; combo, Ysq];

%normalize the computed Eigen value before thresholding. If the
%normalized value is greater than the threshold, add to pos.
if ((min(eig(C)) - min(eigs))/(Range)) > th
pos(count, 1) = min(eig(C));
pos(count, 2) = x;
pos(count, 3) = y;
count = count + 1;
end
end

%sort pos in descending order by the Eigen values
pos = sortrows(pos, -1);
end


function [Ex, Ey] = Gradient(img)
%create the vectors to compute the gradient with - via convolution
Gx = [1, 0, -1];
Gy = [-1, 0, 1]';

%compute the gradients
Ex = conv2(double(img), double(Gx), 'same');
Ey = conv2(double(img), double(Gy), 'same');
end


function [img] = GaussianFilter(img, m, mu, sigma)
%use my guassian kernel function to create a kernel
kernel = GaussKernel(m, mu, sigma);


%do convolution with the image and the kernel.
%Then convert the matrix to a uint8 and display it.
img = uint8(conv2(double(kernel), double(reshape(kernel, m, 1)), double(img), 'same'));
end


function [x] = GaussKernel(m, mu, sigma)
%create a an array of length m
points = zeros([1 m]);


%use this to populate the points array with sampling points
position = -1*floor(m/2);
```

```
i = 1;

%Time-step T = 1
while i <= m
points(i) = position;
position = position + 1;
i = i + 1;
end;
```

- Anexo C: Procesamiento de video

```
imaqreset %reset
clear all

vidobj = videoinput('winvideo',1);

set(vidobj, 'FramesPerTrigger',1);
set(vidobj, 'TriggerRepeat',inf);

triggerconfig(vidobj, 'manual');
start(vidobj);
i=1;
n=1;
while 1,
trigger(vidobj);
frame=getdata(vidobj);
bw = rgb2gray(frame);
bw = im2bw(bw,0.27);
bw = wiener2(bw,[12 12]);
bw = wiener2(bw,[12 12]);
bw = wiener2(bw,[12 12]);

BW=bw;
%imshow(bw);
[B,L,N,A] = bwboundaries(BW,'holes');
imshow(frame);
hold on;
colors=['b' 'g' 'r' 'c' 'm' 'y'];
for k=2:length(B)

boundary = B{k};
cidx = mod(k,length(colors))+1;
plot(boundary(:,2), boundary(:,1),colors(cidx),'LineWidth',2);
row = boundary(1,1)+10;
col = boundary(1,2)+25;

h = text(col+1, row-1, [num2str(row) ',' num2str(col)]);
set(h,'Color',colors(cidx),'FontSize',14,'FontWeight','bold');

if row < 220
'go ahead'
elseif row > 260
'go back'
elseif col>380
'go right'
elseif col<320
'go left'
else
```

```
'stop'
end
break;

end
n = n + 1;
if(n>200000
stop(vidobj
break;
end
end
i=i+1;
```

- Anexo D: Procesamiento de la video etapa 2

```
function singleObject(file_name)
%Inicio del video
video_name = Playa_video
vid = VideoReader(Playa_video);
nframes = vid.NumberOfFrames;
Height = vid.Height; % Altura
Width = vid.Width;
thr = 10; % Threshold para la imagen binaria de ruido
%% Definición de filtro
% Se definen las variables para los estados
% state(t) = [X Y dx dy (d^2)x (d^2)y](t)
% X(t+1) = 1/2(a)T^2 + V(t)T + X(t);

% V(t+1) = aT + V(t)
% a(t+1) = a(t) ;
%State(t+1) = A.State(t) + B.u + <State Uncertainty|State Noise>
dt=0.5;
% A = [1 0 dt 0 (dt^2)/2 0;
%      0 1 0  dt 0    (dt^2)/2;
%      0 0 1  0  dt    0;
%      0 0 0  1  0     dt;
%      0 0 0  0  1     0;
%      0 0 0  0  0     1];
A = [1 0 dt 0;
0 1 0 dt;
0 0 1 0 ;
0 0 0 1 ;
];
B = [(dt^2)/2 (dt^2)/2 dt dt]';
%B = [(dt^2)/2 (dt^2)/2 dt dt 1 1]';
% B=0;
% Entrada
u = 4e-3;
% y = [X Y]
%y(t) = H.State(t) + <Measurement Noise>
% H = [1 0 0 0 0 0
%      0 1 0 0 0 0];
H = [1 0 0 0;
0 1 0 0];
State_Uncertainty = 10;
S = State_Uncertainty * eye(size(A,1));
Meas_Unertainty = 1;
R = Meas_Unertainty * eye(size(H,1));
Dyn_Noise_Variance = (0.01)^2;
%Shahin = [(1/2)*(dt^2) (1/2)*(dt^2) dt dt 1 1]'; %Constant Acceleration
% Shahin = [(1/2)*(dt^2) (1/2)*(dt^2) dt dt]'; %Constant Velocity
```

```
Q = [(dt^2)/4 0 (dt^3)/2 0;
0 (dt^2)/4 0 (dt^3)/2;
(dt^3/2) 0 (dt^2) 0;
0 (dt^3)/2 0 (dt^2);
];


%% Kalman Variables
Input = [];
x = [];
Kalman_Output = [];
% x = [Height/2; Width/2; 0; 0;0;-9.8]; % Initial Values
x = [Height/2; Width/2; 0; 0;]; % Initial Values


background_frame = BackgroundExt(video_name);
moving = zeros(Height,Width,nframes);
labeled_frames = zeros(Height,Width,nframes);
bb=0;
for i=1:nframes-1
current_frame = double(read(vid,i));
moving(:,:,i) = (abs(current_frame(:,:,1) - background_frame(:,:,1)) > thr)...
|(abs(current_frame(:,:,2) - background_frame(:,:,2)) > thr)...
|(abs(current_frame(:,:,3) - background_frame(:,:,3)) > thr);
moving(:,:,i) = bwmorph(moving(:,:,i),'erode',2);
labeled_frames(:,:,i) = bwlabel(moving(:,:,i),4);
stats{i} = regionprops(labeled_frames(:,:,i),'basic');
[n_obj,features] = size(stats{i});
area = 0;
if(n_obj ~= 0)
for k=1:n_obj
if(stats{i}(k).Area > area)
id(i) = k;
area = stats{i}(k).Area;
end
end
centroid(:,:,i) = stats{i}(id(i)).Centroid;
else
centroid(:,:,i) = [rand*200 rand*200];
bb = bb+1;
end
end
for r=1:nframes-1
frames = read(vid,r);
if(id(r)==0)
break;
end
frames = insertShape(frames,'circle',[centroid(1,1,r) centroid(1,2,r)
sqrt(stats{r}(id(r)).Area/pi)],'LineWidth',1);
marked_noise(:,:,:,r) = frames;
end
figure;
x_p=[];
y_p=[];
x_x=0;
y_y=0;
myvec=[0,0];
count=3;
for r=1:nframes-1
frames = read(vid,r);
```

```
frames = insertShape(frames,'circle',[centroid(1,1,r) centroid(1,2,r) 4],'LineWidth',2);
if(mod(r,2) == 0)
input = [centroid(1,1,r); centroid(1,2,r)];
else
input=[];
end

x = A*x + B*u;
S = A*S*A' + Q;
K = S*H'*inv(H*S*H'+R);
if(~isempty(input))
x = x + K*(input - H*x);
end
S = (eye(size(S,1)) - K*H)*S;
Kalman_Output = H*x;
x_p(r)=Kalman_Output(1);
y_p(r)=Kalman_Output(2);
myvec(count)= Kalman_Output(1);
myvec(count+1)= Kalman_Output(2);
count=count+2;
frames = insertShape(frames,'Line',myvec,'LineWidth',2,'Color','yellow');
x_x=Kalman_Output(1);
y_y=Kalman_Output(2);
scenario_1(:,:,:,r) = frames;
end
plot(x_p,y_p);
figure;
implay(read(vid));
imshow(uint8(background_frame));
implay(moving);
implay(scenario_1,15);
end
```

Se adjuntan funciones optimizadas para realizar el procesamiento de imágenes y la adquisición de data haciendo uso de la camara RGB-D y VREP. Así como la evasión de obstáculos y el entrenamiento de la RCNN.

- Anexo E: Cámara RGB-D

```
function sysCall_init()
-- Check if ROSInterface is loaded
rosInterfacePresent = sim.isPluginLoaded('ROSInterface')

if rosInterfacePresent then
local modelHandle = sim.getObjectAssociatedWithScript(sim.handle_self)
local objName = sim.getObjectName(modelHandle)
visionSensorHandle=sim.getObjectHandle(objName)
baseHandle=sim.getObjectHandle('snake_arm')
eeHandle=sim.getObjectHandle('snake_body10')
```

```
aprilHandle=sim.getObjectHandle('apriltag_frame')
cameraHandle=sim.getObjectHandle('ee_cam_frame')

local parentHandle = sim.getObjectParent(modelHandle)
rosImageTopic = sim.getUserParameter(parentHandle, 'rosImageTopic')
if (rosImageTopic == nil) then
rosImageTopic = '/snake_cam/image_color'
sim.setUserParameter(parentHandle, 'rosImageTopic', rosImageTopic)
end
print(string.format('rosImageTopic: %s',rosImageTopic))

blaserVideoPub=simROS.advertise('/snake_cam/image_color', 'sensor_msgs/Image')
-- treat uint8 arrays as strings (much faster, tables/arrays are kind of slow in Lua)
simROS.publisherTreatUInt8ArrayAsString(blaserVideoPub)

posePub = simROS.advertise('/snake_arm/pose', 'geometry_msgs/Pose')
end
end

function sysCall_actuation()
-- put your actuation code here
end

function sysCall_sensing()
-- Publish the image of blaser cam
if rosInterfacePresent then
local data,w,h=sim.getVisionSensorCharImage(visionSensorHandle)
sim.transformImage(data,{w,h},4)

d={}
d['header']={seq=0,stamp=simROS.getTime(), frame_id="a"}
d['height']=h
d['width']=w
d['encoding']='rgb8'
d['is_bigendian']=1
d['step']=w*3
d['data']=data
simROS.publish(blaserVideoPub,d)
end
```

```
-- Publish transforms and poses
if rosInterfacePresent then
-- world -> base_link
tf_world_base_link=getTransformStamped(
baseHandle,'base_link',-1,'world')
simROS.sendTransform(tf_world_base_link)

-- base_link -> ee_link
tf_base_link_ee=getTransformStamped(
eeHandle,'ee_link',baseHandle,'base_link')
simROS.sendTransform(tf_base_link_ee)

-- ee_link -> camera
tf_ee_link_cam=getTransformStamped(
cameraHandle,'camera',eeHandle,'ee_link')
simROS.sendTransform(tf_ee_link_cam)

-- base_link -> apriltag
tf_base_april=getTransformStamped(
aprilHandle,'tag_0',baseHandle,'base_link')
simROS.sendTransform(tf_base_april)

pose_base_link_ee=getPose(eeHandle,baseHandle)
simROS.publish(posePub,pose_base_link_ee)
end
end


function sysCall_cleanup()
-- do some clean-up here
if rosInterfacePresent then
simROS.shutdownPublisher(blaserVideoPub)
sim.addStatusbarMessage('Cleaned up: Snake Camera')
end
end

function getTransformStamped(objHandle,name,relTo,relToName)
-- This function retrieves the stamped transform for a specific object
t = simROS.getTime()
p = sim.getObjectPosition(objHandle,relTo)
```

```
o = sim.getObjectQuaternion(objHandle,relTo)
return {
header={
stamp=t,
frame_id=relToName
},
child_frame_id=name,
transform={
translation={x=p[1],y=p[2],z=p[3]},
rotation={x=o[1],y=o[2],z=o[3],w=o[4]}
}
}
end

function getPose(objectHandle, relTo)
-- This function get the object pose at ROS format geometry_msgs/Pose
p = sim.getObjectPosition(objectHandle,relTo)
o = sim.getObjectQuaternion(objectHandle,relTo)
return {
position={x=p[1],y=p[2],z=p[3]},
orientation={x=o[1],y=o[2],z=o[3],w=o[4]}
}
end
```

Anexo F: Detector de obstáculos

```
import math
from distanceMx import *
from geotransform import *
safety_distance = D
waypoint_file = ("./mission.txt")
obstacle_file = ("./obstacle")
waypoint = []
obstacle = []
waypoint_final = []
waypoint_final_ = []
def toM(feet):
        return (0.3048*feet)

def read_waypoint(waypoint_file):
        waypoint_ = []
```

```
        wp = []
        with open(waypoint_file) as wf:
                line = wf.readline()
                waypoint_.append(line.split(" "))
                while line:
                        line = wf.readline()
                        waypoint_.append(line.split(" "))
        wp = waypoint_[3:]
        for hi in range(len(wp)):
                wp[hi][0] = wp[hi][0].split('\t')
        wp = wp[:-1]
        for hello in range(len(wp)):
                longi = wp[hello][0][8]
                lati = wp[hello][0][9]
                waypoint.append([longi,lati])

def read_obstacle(obstacle_file):
        obstacle_ = []
        ob = []
        with open(obstacle_file) as of:
                line = of.readline()
                obstacle_.append(line.split(" "))
                while line:
                        line = of.readline()
                        obstacle_.append(line.split(" "))
        ob = obstacle_[2:]
        for hi in range(len(ob)):
                ob[hi][0] = ob[hi][0].split('\t')
        ob = ob[:-1]
        for hello in range(len(ob)):
                longi = ob[hello][0][8]
                lati = ob[hello][0][9]
                rad = ob[hello][0][10]
                obstacle.append([longi,lati,rad])

read_waypoint(waypoint_file)
print ("initial", waypoint)
read_obstacle(obstacle_file)
#print ("obstacle = ",obstacle)
```

```
#############################################################################
def convert(arr):
        for m in range(len(arr)):
                for n in range(len(arr[0])):
                        arr[m][n] = float(arr[m][n])
convert(waypoint)
convert(obstacle)
#############################################################################
for i in range(len(waypoint)):
        cwp = waypoint[i]
        clong = cwp[0]
        clat = cwp[1]
        cx,cy = translatell2xy(clat,clong)
        waypoint[i] = [cx,cy]
#print ("old : ",waypoint,"\n")

for j in range(len(obstacle)):
        cobs = obstacle[j]
        clong = cobs[0]
        clat = cobs[1]
        cx, cy = translatell2xy(clat, clong)
        crad = toM(cobs[2])
        obstacle[j] = [cx,cy,crad]
#############################################################################
# obstacle avoidance
def distance(p1,p2,co):
        x1 = p1[0]
        y1 = p1[1]
        x2 = p2[0]
        y2 = p2[1]
        x0 = co[0]
        y0 = co[1]
        num = abs((y0 - y1)*(x2 - x1) - (x0 - x1)*(y2 - y1))
        den = math.sqrt((x2 - x1)**2 + (y2 - y1)**2)
        return (num/den)
```

- Anexo G: Esfera en VREP

```
Function sysCall_init()
rosInterfacePresent = sim.isPluginLoaded('ROSInterface')
if rosInterfacePresent then
targetPosePub = simROS.advertise(
'/gst_desired', 'geometry_msgs/Pose')
end

local modelHandle = sim.getObjectAssociatedWithScript(sim.handle_self)
local objName = sim.getObjectName(modelHandle)
sphereHandle = sim.getObjectHandle(objName)

baseLinkHandle = sim.getObjectHandle('snake_arm')
end

function sysCall_actuation()
-- put your actuation code here
end
function sysCall_sensing()
-- put your sensing code here
if rosInterfacePresent then
-- base_link -> gst_desired
tf_base_desired=getTransformStamped(
sphereHandle,'gst_desired',baseLinkHandle,'base_link')
simROS.sendTransform(tf_base_desired)

pose = getPose(sphereHandle, baseLinkHandle)
simROS.publish(targetPosePub, pose)
end
end

function sysCall_cleanup()
-- do some clean-up here
end

function getTransformStamped(objHandle,name,relTo,relToName)
-- This function retrieves the stamped transform for a specific object
t = simROS.getTime()
p = sim.getObjectPosition(objHandle,relTo)
```

```
o = sim.getObjectQuaternion(objHandle,relTo)
return {
header={
stamp=t,
frame_id=relToName
},
child_frame_id=name,
transform={
translation={x=p[1],y=p[2],z=p[3]},
rotation={x=o[1],y=o[2],z=o[3],w=o[4]}
}
}
end

function getPose(objectHandle, relTo)
-- This function get the object pose at ROS format geometry_msgs/Pose
p = sim.getObjectPosition(objectHandle,relTo)
o = sim.getObjectQuaternion(objectHandle,relTo)
return {
position={x=p[1],y=p[2],z=p[3]},
orientation={x=o[1],y=o[2],z=o[3],w=o[4]}
}
end
```

- Anexo H: Generar las cajas que ubican las botellas identificadas

```
import numpy as np
import cv2

def draw_bboxes(img, n_dets, bboxes, scores, thres):
# Establecemos una cota superior al nÃºmero de cajas por dibujar. Opcional.
max_boxes = 20

# Obtenemos el ancho y alto de la imagen.
h, w, _ = img.shape

# Determinamos el nÃºmero de cajas a dibujar como el mÃ-nimo entre el nÃºmero de
# detecciones y el mÃ¡ximo de cajas establecido por nosotros.
n_boxes = np.min([n_dets,max_boxes])

# Iteramos por las primeras n_boxes cajas.
for i in range(n_boxes):
# Si el score de detecciÃ³n supera el umbral...
if scores[i] > thres:
# Obtenemos la caja correspondiente. Note que sus elementos corresponden
# a coordenadas normalizadas (0-1), por lo que debemos convertirlas a
# coordenadas de la imagen multiplicando respectivamente por el ancho y alto.
```

```
bbox = bboxes[i]

# Note que el orden de los elementos de bbox es (y0,x0,y1,x1).
pt1 = (int(w*bbox[1]), int(h*bbox[0]))
pt2 = (int(w*bbox[3]), int(h*bbox[2]))

# Dibujaremos todos los objetos en rojo.
color = (255, 0, 0)

# Esta funciÃ³n de OpenCV nos permite dibujar rectÃ¡ngulos en una imagen.
# El ultimo argumento es el grosor de la linea.
img = cv2.rectangle(img,pt1,pt2,color,3)

# Finalmente, tras iterar por todas las cajas definidas, retornamos la imagen.
return img
```

- Anexo I: Lectura de la cámara

```
import argparse
import cv2

parser = argparse.ArgumentParser(
description='Obtener imÃ¡genes desde un video o cÃ¡mara web usando OpenCV.')
parser.add_argument('--input_video',help='Ruta hacia la carpeta con el modelo.', default=None)

args = parser.parse_args()

input_video = args.input_video

if input_video is None:
cap = cv2.VideoCapture(0)
else:
cap = cv2.VideoCapture(input_video)

while(cap.isOpened()):
ret, frame = cap.read()
if ret == True:
cv2.imshow('Ejemplo', frame)
if cv2.waitKey(1) == ord('q'):
break
else:
break

cap.release()
```

- Anexo J: Ejecución del video

```
import argparse
import cv2
import time
import tensorflow as tf
import numpy as np
from draw_bboxes import draw_bboxes
import matplotlib.pyplot as plt

parser = argparse.ArgumentParser(
description='Ejecutar un modelo entrenado en Tensorflow y exportado en \
formato SavedModel sobre un video o cÃ¡mara web.')
parser.add_argument('--model_dir', help='Ruta hacia la carpeta con el modelo.')
```

```
parser.add_argument('--input_video',help='Ruta hacia la carpeta con el modelo.', default=None)

args = parser.parse_args()

model_dir = args.model_dir
input_video = args.input_video
print('Loading Model...')
loaded = tf.saved_model.load(model_dir)
print('Successfully loaded the model.')
infer = loaded.signatures["serving_default"]
test_input = (np.random.rand(1,320,320,3)*255).astype('uint8')
print('Running on test image')
_ = infer(tf.convert_to_tensor(test_input))
print('Success! Now opening Video.')

if input_video is None:
cap = cv2.VideoCapture(0)
else:
cap = cv2.VideoCapture(input_video)

out = cv2.VideoWriter('output.mp4',cv2.VideoWriter_fourcc(*'mp4v'), 20.0, (1920,1080))

n_frames = 0

total_time = time.time()
total_fps = []
while(cap.isOpened()):
ret, frame = cap.read()
if ret == True:
frame = frame[:,:,::-1]
input_tensor = tf.convert_to_tensor(np.expand_dims(frame, 0), dtype=tf.uint8)
fps = time.time()
detections = infer(input_tensor)
fps = 1/(time.time()-fps)
total_fps.append(fps)
bboxes = detections['detection_boxes'][0].numpy()
scores = detections['detection_scores'][0].numpy()
n_dets = detections['num_detections'][0].numpy().astype(int)
out_frame = frame.copy()
draw_bboxes(out_frame, n_dets, bboxes, scores, 0)
n_frames += 1
print("Terminamos de procesar el frame "+str(n_frames))
out.write(out_frame[:,:,::-1])
else:
break

cap.release()
out.release()

total_time = (time.time() - total_time)/60
av_fps = np.array(total_fps)
av_fps = np.mean(av_fps)
print("Total time: %.3f"%total_time)
print("Average FPS: %.3f"%av_fps)
plt.figure()
plt.plot(total_fps)
plt.savefig('output.png')
```

- Anexo K: Generar Tfrecord

```
optional arguments:
-h, --help            show this help message and exit
-x XML_DIR, --xml_dir XML_DIR
Path to the folder where the input .xml files are stored.
-l LABELS_PATH, --labels_path LABELS_PATH
Path to the labels (.pbtxt) file.
-o OUTPUT_PATH, --output_path OUTPUT_PATH
Path of output TFRecord (.record) file.
-i IMAGE_DIR, --image_dir IMAGE_DIR
Path to the folder where the input image files are stored. Defaults to the same directory as XML_DIR.
-c CSV_PATH, --csv_path CSV_PATH
Path of output .csv file. If none provided, then no file will be written.
"""

import os
import glob
import pandas as pd
import io
import xml.etree.ElementTree as ET
import argparse

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'    # Suppress TensorFlow logging (1)
import tensorflow.compat.v1 as tf
from PIL import Image
from object_detection.utils import dataset_util, label_map_util
from collections import namedtuple

# Initiate argument parser
parser = argparse.ArgumentParser(
description="Sample TensorFlow XML-to-TFRecord converter")
parser.add_argument("-x",
"--xml_dir",
help="Path to the folder where the input .xml files are stored.",
type=str)
parser.add_argument("-l",
"--labels_path",
help="Path to the labels (.pbtxt) file.", type=str)
parser.add_argument("-o",
"--output_path",
help="Path of output TFRecord (.record) file.", type=str)
parser.add_argument("-i",
"--image_dir",
help="Path to the folder where the input image files are stored. "
"Defaults to the same directory as XML_DIR.",
type=str, default=None)
parser.add_argument("-c",
"--csv_path",
help="Path of output .csv file. If none provided, then no file will be "
"written.",
type=str, default=None)

args = parser.parse_args()

if args.image_dir is None:
args.image_dir = args.xml_dir

label_map = label_map_util.load_labelmap(args.labels_path)
```

```
label_map_dict = label_map_util.get_label_map_dict(label_map)


def xml_to_csv(path):
"""Iterates through all .xml files (generated by labelImg) in a given directory and combines
them in a single Pandas dataframe.

Parameters:
----------
path : str
The path containing the .xml files
Returns
-------
Pandas DataFrame
The produced dataframe
"""

xml_list = []
for xml_file in glob.glob(path + '/*.xml'):
tree = ET.parse(xml_file)
root = tree.getroot()
for member in root.findall('object'):
value = (root.find('filename').text,
int(root.find('size')[0].text),
int(root.find('size')[1].text),
member[0].text,
int(member[4][0].text),
int(member[4][1].text),
int(member[4][2].text),
int(member[4][3].text)
)
xml_list.append(value)
column_name = ['filename', 'width', 'height',
'class', 'xmin', 'ymin', 'xmax', 'ymax']
xml_df = pd.DataFrame(xml_list, columns=column_name)
return xml_df


def class_text_to_int(row_label):
return label_map_dict[row_label]


def split(df, group):
data = namedtuple('data', ['filename', 'object'])
gb = df.groupby(group)
return [data(filename, gb.get_group(x)) for filename, x in zip(gb.groups.keys(), gb.groups)]


def create_tf_example(group, path):
with tf.gfile.GFile(os.path.join(path, '{}'.format(group.filename)), 'rb') as fid:
encoded_jpg = fid.read()
encoded_jpg_io = io.BytesIO(encoded_jpg)
image = Image.open(encoded_jpg_io)
width, height = image.size

filename = group.filename.encode('utf8')
image_format = b'jpg'
xmins = []
xmaxs = []
ymins = []
```

```
ymaxs = []
classes_text = []
classes = []

for index, row in group.object.iterrows():
xmins.append(row['xmin'] / width)
xmaxs.append(row['xmax'] / width)
ymins.append(row['ymin'] / height)
ymaxs.append(row['ymax'] / height)
classes_text.append(row['class'].encode('utf8'))
classes.append(class_text_to_int(row['class']))

tf_example = tf.train.Example(features=tf.train.Features(feature={
'image/height': dataset_util.int64_feature(height),
'image/width': dataset_util.int64_feature(width),
'image/filename': dataset_util.bytes_feature(filename),
'image/source_id': dataset_util.bytes_feature(filename),
'image/encoded': dataset_util.bytes_feature(encoded_jpg),
'image/format': dataset_util.bytes_feature(image_format),
'image/object/bbox/xmin': dataset_util.float_list_feature(xmins),
'image/object/bbox/xmax': dataset_util.float_list_feature(xmaxs),
'image/object/bbox/ymin': dataset_util.float_list_feature(ymins),
'image/object/bbox/ymax': dataset_util.float_list_feature(ymaxs),
'image/object/class/text': dataset_util.bytes_list_feature(classes_text),
'image/object/class/label': dataset_util.int64_list_feature(classes),
}))
return tf_example


def main(_):

writer = tf.python_io.TFRecordWriter(args.output_path)
path = os.path.join(args.image_dir)
examples = xml_to_csv(args.xml_dir)
grouped = split(examples, 'filename')
for group in grouped:
tf_example = create_tf_example(group, path)
writer.write(tf_example.SerializeToString())
writer.close()
print('Successfully created the TFRecord file: {}'.format(args.output_path))
if args.csv_path is not None:
examples.to_csv(args.csv_path, index=None)
print('Successfully created the CSV file: {}'.format(args.csv_path))


if __name__ == '__main__':
tf.app.run()
```

- Anexo L: Script de algoritmo para desarrollo de la red neuronal RCNN

```
#Instalación de Object Detection API
%%bash
git clone --depth 1 https://github.com/tensorflow/models
cd models/research
protoc object_detection/protos/*.proto --python_out=.
cp object_detection/packages/tf2/setup.py .
pip install .
```

```
#Montar el drive para tener acceso
from google.colab import drive
drive.mount('/content/drive')

#Pre procesamiento
%%bash
mkdir /content/workspace
mkdir -p /content/workspace/images/training
mkdir -p /content/workspace/images/testing
mkdir /content/workspace/annotations
%%bash
cp /content/drive/'My Drive'/dataset_botellas/label_map.pbtxt /content/workspace/annotations
cp /content/drive/'My Drive'/dataset_botellas/generate_tfrecord.py /content/workspace
import PIL.Image as pil
import os
import glob
import xml.etree.ElementTree as ET

input_path = '/content/drive/My Drive/dataset_botellas/training/'
output_path = '/content/workspace/images/training/'
#input_path = '/content/drive/My Drive/dataset_botellas/testing/'
#output_path = '/content/workspace/images/testing/'

scale = 0.10

max_samples = 100
#max_samples = 10

n = 0
for xml_file in glob.glob(input_path + '/*.xml'):
tree = ET.parse(xml_file)
root = tree.getroot()
fname = xml_file.split('/')[-1].split('.')[0]

# Actualizamos el tamaño de la imagen.
w = int(root.find('size')[0].text)
h = int(root.find('size')[1].text)
root.find('size')[0].text = str(int(w*scale))
root.find('size')[1].text = str(int(h*scale))

# Actualizamos cada bounding box
for obj in root.findall('object'):
xmin = int(obj[4][0].text)
ymin = int(obj[4][1].text)
xmax = int(obj[4][2].text)
ymax = int(obj[4][3].text)
obj[4][0].text = str(int(xmin*scale))
obj[4][1].text = str(int(ymin*scale))
obj[4][2].text = str(int(xmax*scale))
obj[4][3].text = str(int(ymax*scale))
new_size = ( int(w*scale) , int(h*scale) )
with pil.open(input_path+fname+'.jpg') as img:
new_img = img.resize(new_size,pil.BILINEAR)
# Guardamos los cambios en la carpeta de salida.
tree.write(output_path+fname+'.xml')
new_img.save(output_path+fname+'.jpg')
n += 1
if n >= max_samples:
break
```

```
%%bash
python workspace/generate_tfrecord.py \
-x workspace/images/training \
-l workspace/annotations/label_map.pbtxt \
-o workspace/annotations/training.record

python workspace/generate_tfrecord.py \
-x workspace/images/testing \
-l workspace/annotations/label_map.pbtxt \
-o workspace/annotations/testing.record
%%bash
mkdir /content/workspace/base_model
wget http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_320x3
20_coco17_tpu-8.tar.gz
tar -xf ssd_mobilenet_v2_320x320_coco17_tpu-8.tar.gz -C /content/workspace/base_model --strip-
components 1
rm ssd_mobilenet_v2_320x320_coco17_tpu-8.tar.gz
%%bash
mkdir /content/workspace/trained_model
cp /content/drive/'My Drive'/dataset_botellas/pipeline.config /content/workspace/trained_model


#Entrenamiento de la red
%%bash
mkdir /content/workspace/testing_results
cd models/research/
python object_detection/model_main_tf2.py \
--model_dir=/content/workspace/trained_model \
--pipeline_config_path=/content/workspace/trained_model/pipeline.config
```

Se hacen uso de los siguientes archivos para el algoritmo mostrado:

- Label_map.pbtxt

```
item{
id: 1
name: 'botella'
}
item{
id: 2
name: 'persona'
}
item{
id: 3
name: 'basura'
}
```

- Pipeline.config

```
model {
ssd {
num_classes: 3
image_resizer {
fixed_shape_resizer {
height: 300
width: 300
}
}
feature_extractor {
type: "ssd_mobilenet_v2_keras"
depth_multiplier: 1.0
min_depth: 16
```

```
conv_hyperparams {
regularizer {
l2_regularizer {
weight: 3.9999998989515007e-05
}
}
initializer {
truncated_normal_initializer {
mean: 0.0
stddev: 0.029999999329447746
}
}
activation: RELU_6
batch_norm {
decay: 0.9700000286102295
center: true
scale: true
epsilon: 0.0010000000474974513
train: true
}
}
override_base_feature_extractor_hyperparams: true
}
box_coder {
faster_rcnn_box_coder {
y_scale: 10.0
x_scale: 10.0
height_scale: 5.0
width_scale: 5.0
}
}
matcher {
argmax_matcher {
matched_threshold: 0.5
unmatched_threshold: 0.5
ignore_thresholds: false
negatives_lower_than_unmatched: true
force_match_for_each_row: true
use_matmul_gather: true
}
}
similarity_calculator {
iou_similarity {
}
}
box_predictor {
convolutional_box_predictor {
conv_hyperparams {
regularizer {
l2_regularizer {
weight: 3.9999998989515007e-05
}
}
initializer {
random_normal_initializer {
mean: 0.0
stddev: 0.009999999776482582
}
}
activation: RELU_6
```

```
batch_norm {
decay: 0.9700000286102295
center: true
scale: true
epsilon: 0.0010000000474974513
train: true
}
}
min_depth: 0
max_depth: 0
num_layers_before_predictor: 0
use_dropout: false
dropout_keep_probability: 0.800000011920929
kernel_size: 1
box_code_size: 4
apply_sigmoid_to_scores: false
class_prediction_bias_init: -4.599999904632568
}
}
anchor_generator {
ssd_anchor_generator {
num_layers: 6
min_scale: 0.20000000298023224
max_scale: 0.949999988079071
aspect_ratios: 1.0
aspect_ratios: 2.0
aspect_ratios: 0.5
aspect_ratios: 3.0
aspect_ratios: 0.33329999446868896
}
}
post_processing {
batch_non_max_suppression {
score_threshold: 0.3
iou_threshold: 0.5
max_detections_per_class: 25
max_total_detections: 25
use_static_shapes: false
}
score_converter: SIGMOID
}
normalize_loss_by_num_matches: true
loss {
localization_loss {
weighted_smooth_l1 {
delta: 1.0
}
}
classification_loss {
weighted_sigmoid_focal {
gamma: 2.0
alpha: 0.75
}
}
classification_weight: 1.0
localization_weight: 1.0
}
encode_background_as_zeros: true
normalize_loc_loss_by_codesize: true
inplace_batchnorm_update: true
```

```
freeze_batchnorm: false
}
}
train_config {
batch_size: 8
data_augmentation_options {
random_horizontal_flip {
}
}
data_augmentation_options {
ssd_random_crop {
}
}
optimizer {
momentum_optimizer {
learning_rate {
cosine_decay_learning_rate {
learning_rate_base: 0.8
total_steps: 1000
warmup_learning_rate: 0.13
warmup_steps: 100
}
}
momentum_optimizer_value: 0.9
}
use_moving_average: false
}
fine_tune_checkpoint: "/content/workspace/base_model/checkpoint/ckpt-0"
num_steps: 1000
max_number_of_boxes: 25
unpad_groundtruth_tensors: false
fine_tune_checkpoint_type: "detection"
fine_tune_checkpoint_version: V2
}
train_input_reader {
label_map_path: "/content/workspace/annotations/label_map.pbtxt"
tf_record_input_reader {
input_path: "/content/workspace/annotations/training.record"
}
}
eval_config {
metrics_set: "coco_detection_metrics"
use_moving_averages: false
visualization_export_dir: "/content/workspace/testing_results/"
}
eval_input_reader {
label_map_path: "/content/workspace/annotations/label_map.pbtxt"
shuffle: false
num_epochs: 1
tf_record_input_reader {
input_path: "/content/workspace/annotations/testing.record"
}
}
```