UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE TECNOLOGÍAS DE LA INFORMACIÓN, TELECOMUNICACIONES E INDUSTRIAL

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y COMUNICACIONES

TEMA:

"NAVEGACIÓN AUTÓNOMA BASADA EN MANIOBRAS BAJO ESTIMACIÓN DE POSTURAS HUMANAS PARA UN ROBOT OMNIDIRECCIONAL KUKA YOUBOT"

Trabajo de Graduación. Modalidad: Proyecto de Investigación, presentado previo la obtención del título de Ingeniero en Electrónica y Comunicaciones.

LÍNEA DE INVESTIGACIÓN: Sistemas de Control y Automatización

AUTOR: Elmer Santiago Barahona Guamani

TUTOR: PhD, Carlos Gordón Gallegos

Ambato – Ecuador

Agosto 2019

APROBACIÓN DEL TUTOR

En mi calidad de tutor del Trabajo de Investigación sobre el tema: "NAVEGACIÓN AUTÓNOMA BASADA EN MANIOBRAS BAJO ESTIMACIÓN DE POSTURAS HUMANAS PARA UN ROBOT OMNIDIRECCIONAL KUKA YOUBOT", del señor Elmer Santiago Barahona Guamani, estudiante de la Carrera de Ingeniería en Electrónica y Comunicaciones, de la Facultad de Tecnologías de la Información, Telecomunicaciones e Industrial, de la Universidad Técnica de Ambato, considero que el informe investigativo reúne los requisitos suficientes para que continúe con los tramites y consiguiente aprobación de conformidad con el numeral 7.2 de los Lineamientos Generales para la aplicación de Instructivos de las Modalidades de Titulación de las Facultades de la Universidad Técnica de Ambato.

Ambato, 2 de agosto del 2019

EL TUTOR

PhD, Carlos Gordón Gallegos

AUTORÍA

El presente Proyecto de Investigación titulado: "NAVEGACIÓN AUTÓNOMA BASADA EN MANIOBRAS BAJO ESTIMACIÓN DE POSTURAS HUMANAS PARA UN ROBOT OMNIDIRECCIONAL KUKA YOUBOT", es absolutamente original, auténtico y personal, en tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, 2 de agosto del 2019

Elmer Santiago Barahona Guamani

CC: 0503431272

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato, para que haga uso de este trabajo de titulación como un documento disponible para lectura, consulta y procesos de investigación.

Cedo los derechos de mi Trabajo de Titulación, con fines de difusión pública, además autorizo su reproducción dentro de las regulaciones de la Universidad.

Ambato, 2 de agosto del 2019

Elmer Santiago Barahona Guamani

CC: 0503431272

APROBACIÓN DE LA COMISIÓN CALIFICADORA

La Comisión Calificadora del presente trabajo conformada por los señores docentes Ing. Mg. Santiago Altamirano e Ing. Mg. Mario García, revisó y aprobó el Informe Final del Proyecto de Investigación titulado "NAVEGACIÓN AUTÓNOMA BASADA EN MANIOBRAS BAJO ESTIMACIÓN DE POSTURAS HUMANAS PARA UN ROBOT OMNIDIRECCIONAL KUKA YOUBOT", presentado por el señor Elmer Santiago Barahona Guamani, de acuerdo al numeral 9.1 de los Lineamientos Generales para la aplicación de Instructivos de las Modalidades de Titulación de las Facultades de la Universidad Técnica de Ambato.

Ing. Mg. Elsa Pilar Urrutia Urrutia
PRESIDENTA DEL TRIBUNAL

Ing. Santiago Altamirano
DOCENTE CALIFICADOR

Ing. Mario García
DOCENTE CALIFICADOR

DEDICATORIA:

A mis Padres, quien con todo su amor y sus enseñanzas están logrado cultivar en sus tres hijos un espíritu de superación, fomentando valores como la humildad, el respeto y la responsabilidad, con su claro ejemplo que han sido capaces de lograr grandes cosas a lo largo de su vida, motivándome a seguir sus pasos y cumplir todos mis metas. Gracias por su todo apoyo incondicional.

A mis hermanas, por estar pendientes siempre de mí y apoyarme en todos los momentos, demostrándome sus actos de cariño y amor sincero.

Santiago Barahona

AGRADECIMIENTO:

*A **Dios**, por darme la salud y fuerza que necesito para seguir cumpliendo metas en mi vida personal, brindándome la oportunidad de vivir experiencias inigualables como hijo, hermano, amigo y ahora como profesional.*

Mil gracias a mis padres Miriam y Juan, por su apoyo incondicional, por haberme brindado desde pequeño todo su amor, gracias por creer y confiar en mí y en que podré culminar con éxito mi carrera, por siempre darme su apoyo en los momentos más difíciles, gracias por todos los conejos y por ser los que siempre guiaron mi camino.

*Gracias a mi tutor PhD**. Carlos Gordon,** por compartir su conocimiento y presentarme este reto, pudiendo desarrollar campos que nunca me habría imaginado aprender, gracias por el tiempo brindado.*

A todos mis amigos que incondicionalmente estuvieron siempre para brindarme apoyo, compartiendo momentos agradables en todo este proceso.

Santiago Barahona

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN EJECUTIVO

El presente proyecto es parte del tema de investigación “Plataforma Móvil Omnidireccional KUKA dotada de Inteligencia Artificial utilizando estrategias de Machine Learnig para Navegación Segura en Espacios no Controlados” coordinado por el PhD, Carlos Gordon; en donde se mejora el control de navegación autónomo del robot Kuka YouBot, aprovechando que su plataforma es de código abierto permitiendo investigación y desarrollo científico. Una vez elegido el algoritmo que mejor se ajuste a nuestras necesidades para la estructura de visión artificial aplicando a la detección objetos y mediante interacción con el robot Kuka YouBot se puede obtener una navegación satisfactoria del robot móvil.

Los resultados que obtenemos del proceso de visión artificial es estable y en tiempo real, gracias al uso de la tarjeta de desarrollo Nvidia Jetson Nano, que es desarrollada especialmente para procesos de Inteligencia artificial.

ABSTRACT

The present project is part of the research topic "KUKA Omnidirectional Mobile Platform equipped with Artificial Intelligence using Machine Learnig strategies for Safe Navigation in Uncontrolled Spaces" coordinated by the PhD, Carlos Gordon; where the autonomous navigation control of the Kuka YouBot robot is improved, taking advantage of the fact that its platform is open source allowing research and scientific development. Once the algorithm that best fits our needs for the artificial vision structure is chosen, applying objects detection and interacting with the Kuka YouBot robot can obtain a satisfactory navigation of the mobile robot.

The results we obtain from the artificial vision process is stable and in real time, thanks to the use of the Nvidia Jetson Nano development card, which is developed especially for artificial intelligence processes.

INTRODUCIÓN

La robótica es el campo de la ciencia que se encarga de estudiar: la construcción, la programación y el diseño de los robots que al interactuarse con campos de ingeniería como son: la electrónica, la mecánica, los sistemas de control, la inteligencia artificial, las redes neuronales, pueden generar sistemas de propósitos específicos, para trabajar en áreas definidas.

Con esta premisa, para implementar una navegación con el robot autónomo Kuka es prudente conocer las características físicas, el software que maneja y las diferentes interfaces de comunicación que tienen el robot, estos se obtienen mediante el análisis de manuales obtenidos desde la plataforma misma del Kuka y antecedentes de trabajos referentes al tema en repositorios de universidades nacionales e internacionales.

La mayor parte de las aplicaciones realizadas con Kuka YouBot, son de navegaciones programadas previamente en su computador en donde desde la API de YouBot se cargan instrucciones para planificación de rutas, enviando datos como posición o velocidad angular hacia la plataforma y el brazo de Kuka, el robot realizaba su aplicación previamente programada y terminaba el proceso, esto obviamente limitaba tener una navegación que se ajuste a las necesidades del operario en caso de necesitar una ruta extra, por lo que para esta investigación se mejora la aplicación realizando una navegación autónoma controlada por un sistema de visión artificial que se encarga de enviar ordenes de rutas y poder realizar las maniobras que se necesiten para cumplir una tarea.

Para cumplir con esta investigación se utiliza aplicaciones de tipo Open Source, utilizando un protocolo de comunicación y una arquitectura, para que el sistema de visión artificial pueda detectar las señales generadas por posturas humanas y envié estos datos hacia el robot Kuka YouBot mediante el protocolo de comunicación.

El Capítulo I del presente Proyecto describe los problemas y las necesidades que tienen los robots móviles autónomos y la estructura de solución para esta investigación.

El Capítulo II habla de los antecedentes de trabajos referentes al tema, en este capítulo también de describe la fundamentación teórica que conlleva la investigación haciendo énfasis en la interacción de plataformas usadas. Y finaliza con la propuesta de solución a la problemática planteada en el capítulo I.

El Capítulo III detalla la metodología que se siguió en la investigación y los objetivos cumplidos.

El Capítulo IV muestra un proceso sistematizado de la implementación de la navegación autónoma del robot Kuka controlado mediante posturas humanas.

Finalmente, en el Capítulo V habla de las conclusiones y recomendaciones obtenidas del proyecto de investigación.

CAPÍTULO I

EL PROBLEMA

1.1 Tema de Investigación

NAVEGACIÓN AUTÓNOMA BASADA EN MANIOBRAS BAJO ESTIMACIÓN DE POSTURAS HUMANAS PARA UN ROBOT OMNIDIRECCIONAL KUKA YOUBOT.

1.2 Planteamiento del Problema

Durante la última década la robótica ha sufrido una profunda transformación, muchos investigadores en robótica que comenzaron sus trabajos en manipuladores industriales han emigrado hacia la investigación de más avanzados sistemas robóticos, particularmente hacia los robots móviles, los que generalmente son destinados a trabajar fuera de las áreas de manufactura (procesos de maquinado, ensamble, la soldadura y el manejo de materiales). [1]

Recientemente, los robots móviles autónomos se han convertido en otra generación de máquinas, principalmente utilizadas como dispositivos de servicio, dedicados a desempeñar tareas sencillas que actualmente son realizadas por personas poco calificadas. Entre estas tareas, se incluyen trabajos de transporte de material, limpieza de pisos y trabajos de servicio en hospitales, por mencionar solo algunos.

La elaboración de dichos sistemas autónomos inteligentes requiere de un trabajo multidisciplinario en donde se agrupen diversas áreas tales como la mecánica, la electrónica y las ciencias computacionales, con la cual pueden atacarse problemas de adquisición de datos (a partir de sensores), manejo de bases de conocimientos (almacenadas en la memoria del robot), visión computacional, interpretación de

información (proveniente de sensores), control de actuadores, planeación de trayectorias y la navegación.[1]

A pesar de los grandes esfuerzos y la gran cantidad de trabajos realizados para tratar de hacer que los robots móviles autónomos se comporten de manera parecida a los sistemas naturales (insectos. aves), en la actualidad aún existen muchos problemas que requieren de mucho más trabajo para lograr.

La navegación, la cooperación, el reconocimiento y el aprendizaje, son algunas de las áreas dentro de las ciencias computacionales en donde aún se presentan problemas a pesar de la gran cantidad de trabajo y esfuerzo en torno a ellos que al resolverlos se lograra hacer que los robots móviles alcancen los niveles de comportamiento esperado. Principalmente esto es debido a que estos problemas son sumamente complejos y por ende difíciles de controlar, de predecir y de modelar matemáticamente. [2]

La navegación es uno los problemas que podemos mencionar. Siempre que se trata de una navegación autónoma nace la pregunta, ¿Que es necesario o que se necesita para que un robot móvil pueda tener la habilidad y/o capacidad de navegar a través de su medio ambiente dentro del mundo real? Es posible observar cómo es que la naturaleza a resuelto el problema de la navegación eficientemente y de diversas maneras. [3]

Los salmones por ejemplo, son capaces de encontrar sus lugares de desove desde cientos de kilómetros de distancia, las palomas por otra parte, son capaces de encontrar sus lugares de destino en días nublados, soleados y aún lluviosos, las abejas son capaces de establecer rutas con las cuales pueden encontrar la fuente de comida, los murciélagos aun estando casi ciegos, son capaces de evitar obstáculos. Por el contrario, muy pocos robots móviles son capaces de navegar a través de un simple pasillo (corredor) sin la constante supervisión y corrección de su trayectoria por parte de un ser humano. [4]

Otro de los problemas aun totalmente sin resolver y que de cierta manera va de la mano con la navegación, es el de reconocimiento, problema que consiste en identificar de manera adecuada el ambiente de trabajo para poder desarrollar eficientemente las tareas dentro del mismo. Dentro de los robots móviles, el reconocimiento de marcas en el terreno dentro de 1os ambientes no estructurados es un problema muy difícil de resolver. [4]

Ya sea utilizando cámaras, sensores piezoeléctricos, sensores de fuerza e incluso micrófonos, el reconocimiento de patrones en el medio ambiente es un problema no trivial. El problema del reconocimiento es un problema que requiere de intensos esfuerzos computacionales el cual está sujeto a problemas y dificultades presentes en el ambiente tales como la luz, oclusión e inclusión de datos o información ruidosa. [5]

Por otro lado, la fusión de la robótica y la tecnología de la IA tiene varias consecuencias, y los primeros en adoptar estos nuevos sistemas robóticos están cosechando los beneficios. KUKA, es un fabricante líder de robots industriales, que está implementando la IA y la tecnología de aprendizaje automático en sus robots colaborativos. [6]

Una de las principales falencias para el estudio de los robots KUKA es el desconocimiento en su uso y aplicación que limita su incorporación en la industria local. La industria ecuatoriana teme involucrarse en esta área, cuya razón se sustenta en no encontrar personal capacitado para su operación en forma adecuada y obtener el máximo rendimiento de los robots en los procesos industriales. [6]

En el ámbito universitario sin la documentación suficiente sobre el manejo de robot KUKA se ha restringido la implementación de prácticas de laboratorio generándose un conocimiento puramente teórico. Esto no permite aportar al avance de la ciudad y el país con ingenieros con conocimientos necesarios para impulsar el desarrollo de nuevas tecnologías. [7]

El resultado es un robot que no sólo es capaz de trabajar con seguridad junto a los humanos, sino que también puede ser reprogramado fácilmente para nuevas tareas, a diferencia de los robots industriales tradicionales que dependen de una extensa programación para cada tarea. [7]

1.3 Delimitación del Problema

1.3.1 Delimitación de Contenidos

Área Académica: Ingeniería

Línea de Investigación: Sistema de control

Sub línea de Investigación: Automatización

1.3.2 Delimitación Espacial

La presente investigación estará orientada a realizarse en la Universidad Técnica de Ambato.

1.3.3 Delimitación Temporal

La presente investigación se desarrollará en el período Marzo – Agosto 2019 de acuerdo con lo establecido en el Reglamento de graduación para obtener el título terminal de tercer nivel de la Universidad Técnica de Ambato.

1.4 Justificación

Los robots autónomos deben tener la capacidad de desplazarse, para situarse en su lugar de trabajo, a veces alejado del puesto de control o en un lugar de muy difícil acceso o situado en un entorno incómodo o peligroso para los humanos. La gran variedad de entornos y situaciones en los que se puede desenvolver un robot exige capacidades de locomoción desarrolladas para estos trabajos. [8]

El presente trabajo de investigación resulta importante por el gran aporte que genera, al permitir que el robot omnidireccional autónomo KUKA YouBot, obtenga la característica de recibir órdenes a través de posturas humanas y de ejecutar las acciones requeridas. En esta primera aproximación, al robot se le enseñan órdenes de movimiento corporal para guiarlo en un entorno típico de interiores formado por pasillos y habitaciones.

La implementación de la característica de que el robot reciba órdenes mediante posturas humanas será el punto de partida para muchas aplicaciones en donde el ser humano necesite la ayuda de un robot autónomo.

Los beneficiarios sería la población en general por las múltiples aplicaciones que esta investigación tecnológica conlleva como por ejemplo se pueden generar alternativas para automatizar tareas, entre las cuales se puede mencionar algunas como la sustitución de la visión humana por la de un robot el cual puede tomar decisiones dependiendo de una variable de entrada como podría ser una imagen, se puede mencionar también la sustitución de la fuerza de alguna persona por la de un brazo robótico el cual sea diseñado para sujetar objetos de gran tamaño y peso, evitando con esto que pueda sufrir algún tipo de lesión debido a la frecuencia y tiempo con la que se realiza dicha tarea; cabe señalar que un robot puede realizar casi las mismas tareas que las de un humano común, la gran diferencia radica en la gran exactitud y

repetitividad que dicho dispositivo se puede mantener realizando la misma tarea por muchas más horas que las que un humano, este podría soportar sin cansancio mental y físico que lo podrían llevar a aumentar el riesgo de ser víctima de su propio cansancio, llevándolo con esto a realizar su trabajo de una mala manera, además de estar cada vez más cerca de sufrir alguna lesión que lo pudiera marginar de sus obligaciones diarias.

El proyecto resulta ser factible debido a que se cuenta con todos los materiales como: elementos electrónicos, informáticos, robot (kuka) necesarios para la ejecución del proyecto en mención, además de contar con el respaldo de una investigación desarrollada ya en la Universidad Técnica de Ambato se presenta el estudio "Autonomous Robot KUKA YouBot Navigation Based on Path Planning and Traffic Signals Recognition".

1.5 Objetivos

1.5.1 Objetivo General

Implementar un sistema de navegación autónoma con comandos para maniobras bajo estimación de posturas humanas para un robot omnidireccional KUKA YouBot

1.5.2 Objetivos Específicos

- Estudiar el Estado del Arte de la detección y seguimiento de objetos basado en visión artificial.

- Analizar la cinemática y odometría del robot omnidireccional autónomo KUKA YouBot.

- Caracterizar los algoritmos para detección de posturas humanas basadas en visión artificial.

- Diseñar un sistema autónomo basado en los principios y fundamentos de estimación de posturas humanas cuya funcionalidad determine el comportamiento físico del robot omnidireccional KUKA YouBot.

CAPÍTULO II

MARCO TEÓRICO

2.1 Antecedentes Investigativos

De la investigación y búsqueda realizada en repositorios de Universidades Internacionales, así como también en artículos científicos relacionados con el tema propuesto, se ha encontrado diferentes trabajos de investigación, los cuales se describe a continuación:

En el año 2018, en la Universidad Técnica de Ambato se presenta el estudio "Autonomous Robot KUKA YouBot Navigation Based on Path Planning and Traffic Signals Recognition" de Carlos Gordón, Patricio Encalada, Henry Lema, Diego León y Cristian Peñaherrera. Incursionan en el desarrollo e implementación de la integración de capacidades como la planificación de rutas y el reconocimiento de señales de tráfico que se logró mediante la integración de los entornos de trabajo ROS, Matlab y OpenCV. ROS permitió la simulación de la navegación del robot autónomo utilizando Gazebo y proporcionó la implementación del algoritmo en la plataforma real del robot KUKA YouBot. Matlab mejoró las tareas de comunicación aprovechando las herramientas de procesamiento de datos en el proceso de planificación de la ruta. Finalmente, OpenCV permite el reconocimiento de las señales de tráfico utilizando el algoritmo SIFT y SURF. El resultado fue la integración de capacidades, la planificación de rutas y el reconocimiento de señales de tráfico se logró mediante la integración de los entornos de trabajo ROS, Matlab y OpenCV. [9]

En el año 2018, en Universidad Nacional del Noroeste de la Provincia de Buenos Aires (UNNOBA) presenta el estudio "Visión artificial aplicada en Agricultura de

Precisión" de Claudia Russo, Hugo Ramón, Sandra Serafino, BenjaminCicerchia, Mónica Sarobe, Agustín Balmer, Álvarez Eduardo, Pablo Luengo, Gustavo Useglio, Martín Faroppa. La creación de una plataforma robótica móvil terrestre de navegación de cultivos, se espera dotar de un sistema de visión artificial que le permita a esta poder navegar el cultivo intra/extra surcos y/o a través de caminos. La plataforma busca recolectar datos biofísicos del cultivo a lo largo de la evolución de sus estados fenológicos, sumado al monitoreo y evaluación de los mismos. Para determinar dicho instante el sistema debe ser capaz de detectar las diferentes plantas del cultivo, teniendo en cuenta además cuáles de ellos ya fueron censados. Con la particularidad que el tracking será invertido, ya que el objeto a seguir (planta) estará quieto y lo que se mueve es la cámara. Sobre la base este proyecto se plantean además avances y mejoras orientadas al desarrollo de plataformas multipropósitos y auto navegables mediante visión artificial, que puedan coordinarse con vehículos aéreos utilizando estas mismas técnicas sumadas a geo posicionamiento mediante DGPS. Como resultado se espera dotar de sistemas de visión artificial a una plataforma robótica móvil con un sistema de navegación autónoma en escenas outdoor, y en condiciones de suelo irregular, que pueda aplicarse al recorrido de ensayos de diferentes tipos de cultivo en el campo. Sumado a un sistema de visión para un vehículo aéreo no tripulado. Donde ambos sistemas puedan estar relacionados entre sí y que sirvan de aporte el uno del otro. [10]

En el año 2014, en el Centro de Investigaciones en Óptica de León, Guanajuato, México presenta la tesis "Sistema automatizado de reconocimiento y manipulación de objetos usando visión por computadora y un brazo industrial" de Ing. Alberto Martínez Rodríguez. Tiene como propósito el exponer como pueden ser integrados a un robot industrial una serie de dispositivos electrónicos y algoritmos matemáticos para la clasificación, selección y almacenaje de objetos, piezas o productos de manufactura. El desarrollo del presente proyecto implica el uso de código de barras bidimensional, identificación por radiofrecuencia, transmisión inalámbrica de datos, procesamiento por computadora y manipulación de objetos. Por lo que se hará uso de una webcam, un dispositivo de lectura/escritura para identificadores RFID, transmisor y receptor Bluetooth, una PC y un robot industrial. Se trata de un proyecto que demuestra la factibilidad de integrar elementos que, por separado parecen ser meras curiosidades.

Puede tener seguimiento para poder leer un código de barras lineal con funcionalidades parecidas a las que se tratan en esta tesis. También puede aplicarse en ferias de ingeniería para atraer a los jóvenes a inscribirse a alguna disciplina afín, y contribuir al desarrollo del país al mediano o largo plazo. Además, la tendencia a la movilidad y la omnipresencia, hacen que cada vez sean más utilizados los sistemas inalámbricos, y el objetivo es ir evitando los cables en todo tipo de comunicación, no solo en el campo de televisión y telefonía, sino en seguridad y domótica. Para este proyecto, se aplicará a sistemas robotizados. [11]

En el año 2018, Universidad de Pamplona de Cúcuta, Colombia se presenta la investigación "Diseño e implementación de un sistema de visión artificial usando una técnica de mapeo y localización simultánea (SLAM) sobre una plataforma robótica móvil" de Luz Karime Garzón Obregóna, Luis Albero Forero Rincónb, Oscar Manuel Duque-Suárez. El objetivo de esta investigación está centrado en aplicar una técnica de SLAM sobre una plataforma robótica móvil. Para lograr dicho objetivo se realiza un estudio de diferentes metodologías de SLAM existentes que pudieran cumplir con las expectativas resultando elegido el LSD-SLAM, la cual con una cámara adecuada se logra obtener un mapa semi-denso 3D del ambiente en el cual se trabaja. Además, se desarrolla en ROS (Robot Operating System) definido como sistema operativo, que cada vez debería ser tenido en cuenta para futuras investigaciones gracias a sus grandes prestaciones. Como Resultado con los avances en la computación, la electrónica y la robótica cada vez se realizan robots más pequeños con grandes funcionalidades, en este proyecto se usó un robot hecho en base al robot Turtlebot 2, el cual posee grandes dimensiones en cuanto a su tamaño, siendo esta una limitante y por eso se quiso direccionar hacia el fin de realizar el mismo trabajo con plataformas mucho más pequeñas que ahorren espacio con el uso de tarjetas de procesamiento como la raspberry y tele-operación con ROS. La calidad de mapeo que se obtuvo no fue excelente, pero se consiguió una gran representación de la escena real, comparada con otras técnicas de mapeo que solo logran extraer algunas características del entorno. Dichos resultados se podrían mejorar utilizando una cámara de mayor ángulo de visión, mayor velocidad de frames, y en un computador con procesamiento más rápido. [12]

En el año 2017, Universidad de Politécnica de Catalunya de Barcelona, España, se presenta la Tesis “Estación Robotizada De Paletizado” de Ignacio Romero Guillén. El proyecto se basa en la automatización de una estación de paletizado robotizada que permite gestionar el almacenamiento de unas piezas según las indicaciones configuradas por el usuario. El control de la aplicación se realiza a través de un interfaz hombre máquina y un controlador PLC, este último, es el responsable de gestionar el resto de elementos que incorpora el sistema, como son el sistema de visión artificial y el robot de ABB. Una de las partes importantes de este proyecto, ha sido la integración de todos los elementos antes mencionados en una red de comunicaciones que permita el intercambio de datos de una manera rápida, eficiente y segura, en este caso, el protocolo elegido para esta red ha sido Profinet debido a que es el estándar de SIEMENS y está muy extendido en el mundo industrial. El resultado es que este proyecto se ha realizado con la "Escola Jesuïtes el Clot" con una finalidad docente y se han utilizado dispositivos y equipos que ya existían de anteriores proyectos, como es el caso de las cintas transportadoras, este hecho tiene ventajas e inconvenientes, ya que por un lado no se ha tenido que diseñar desde cero, por otro lado, se han tenido que adaptar equipos al nuevo proyecto. [13]

2.2 Fundamentación Teórica

2.2.1 Robótica de Desarrollo

Esta es una parte de la robótica que tiene como objetivo desarrollar sistemas de control de propósito general, por medio de un extenso proceso de organización autónoma. Tenemos como respuesta que el robot tiene las características de seguir desarrollando aplicaciones más y más complejas como son las capacidades perceptuales, cognitivas y comportamentales. En resumen, es un área de investigación que junta la neurociencia del desarrollo, la psicología del desarrollo y la robótica situada. En inicio el sistema puede tener un reducido conjunto de conductas o conocimientos innatos, pero por las aplicaciones ya desarrolladas tiene la habilidad de desarrollar aplicaciones más sofisticadas a las ejecutadas con anterioridad. Esto indica de la máquina desarrolla de forma autónoma las habilidades precisas para un entorno especifico “desarrollo mental autónomo”. [14]

2.2.2 Inteligencia Artificial

La inteligencia artificial (IA), es la capacidad que tiene una computadora digital en realizar tareas comúnmente asociadas con seres inteligentes. El término se aplica con frecuencia al proyecto de desarrollo de sistemas dotados de procesos intelectuales y característicos de los seres humanos, como la capacidad de razonar, descubrir significado, generalizar o aprender de experiencias pasadas. Desde el desarrollo de la computadora digital en la década de 1940, se ha demostrado que las computadoras pueden programarse para llevar a cabo tareas muy complejas, como, por ejemplo, descubrir pruebas para teoremas matemáticos o jugar ajedrez con gran dominio, aun así, a pesar de los avances continuos en la velocidad de procesamiento de la computadora y la capacidad de memoria, todavía no hay programas que puedan igualar la flexibilidad humana en dominios más amplios o en tareas que requieren mucho conocimiento diario. Por otro lado, algunos programas han alcanzado los niveles de rendimiento de los expertos y profesionales humanos en la realización de ciertas tareas específicas, de modo que la inteligencia artificial en este sentido no se encuentra limitado en aplicaciones tan diversas como el diagnóstico médico, los motores de búsqueda por computadora y el reconocimiento de voz o escritura a mano. [15]

Aprendizaje

Hay varias formas diferentes de aprendizaje aplicadas a la inteligencia artificial. Lo más sencillo es aprender por ensayo y error, por ejemplo, un programa de computadora simple para resolver problemas de ajedrez podría intentar movimientos al azar hasta que se encuentre el mate. Luego, el programa puede almacenar la solución con la posición, de modo que la próxima vez que la computadora encuentre la misma posición, recuerde la solución. Esta simple memorización de elementos y procedimientos individuales, conocida como aprendizaje de memoria, es relativamente fácil de implementar en una computadora. Más desafiante es el problema de implementar lo que se llama “generalización” . La generalización implica aplicar la experiencia pasada a situaciones nuevas análogas, por ejemplo, un programa que aprende el tiempo pasado de los verbos regulares en inglés no podrá producir el tiempo pasado de una palabra como saltar, a menos que se haya presentado previamente saltado , mientras que un programa que sea capaz de generalizar puede

aprender la regla que se usa y así forman el tiempo pasado de salto basado en la experiencia con verbos similares. [15]

Razonamiento

Razonar es sacar inferencias apropiadas a la situación. Las inferencias se clasifican como deductivo o inductiva. Un ejemplo de lo anterior es, “Fred debe estar en el museo o en la cafetería, él no está en el café; por lo tanto, él está en el museo ", y de este último, “accidentes anteriores de este tipo fueron causados por la falla del instrumento; por lo tanto, este accidente fue causado por la falla del instrumento”. La diferencia más significativa entre estas formas de razonamiento es que, en el caso deductivo, la verdad de las premisas garantiza la verdad de la conclusión, mientras que, en el caso inductivo, la verdad de la premisa se apoya a la conclusión sin dar una seguridad absoluta. El razonamiento inductivo es común en la ciencia, donde se recopilan los datos y se desarrollan modelos tentativos para describir y predecir el comportamiento futuro, hasta que la aparición de datos anómalos obliga a revisar el modelo. El razonamiento deductivo es común en las matemáticas y la lógica, donde las estructuras elaboradas de teoremas irrefutables se construyen a partir de un pequeño conjunto de reglas y axiomas básicos. [15]

Ha habido un éxito considerable en la programación de computadoras para hacer inferencias, especialmente inferencias deductivas. Sin embargo, el verdadero razonamiento involucra más que solo hacer inferencias, implica sacar inferencias relevantes para la solución de la tarea o situación particular. Este es uno de los problemas más difíciles que enfrenta la IA. [15]

Métodos en la IA.

Enfoques simbólicos vs. conexionistas.

La investigación de la IA sigue dos métodos distintos y, hasta cierto punto, en competencia, el enfoque simbólico (o "arriba-abajo") y el enfoque conexionista (o "abajo-arriba"). El enfoque de arriba hacia abajo busca replicar la inteligencia mediante el análisis de la cognición independiente de la estructura biológica del cerebro, en términos del procesamiento de símbolos, de ahí la etiqueta simbólica. El enfoque de abajo hacia arriba, por otro lado, implica la creación de redes neuronales

artificiales en imitación de la estructura del cerebro, de ahí la etiqueta conexionista. [15]

Para ilustrar la diferencia entre estos enfoques, considere la tarea de construir un sistema, equipado con un escáner óptico, que reconozca las letras del alfabeto. Un enfoque de abajo a arriba generalmente implica entrenar una red neuronal artificial presentándole cartas una por una, mejorando gradualmente el rendimiento al "sintonizar" la red. (La afinación ajusta la capacidad de respuesta de diferentes vías neuronales a diferentes estímulos). En contraste, un enfoque de arriba a abajo generalmente implica escribir un programa de computadora que compare cada letra con descripciones geométricas. En pocas palabras, las actividades neuronales son la base del enfoque de abajo hacia arriba, mientras que las descripciones simbólicas son la base del enfoque de arriba hacia abajo. [15]

2.2.3 Visión Artificial

La visión por computadora, una tecnología de inteligencia artificial que permite a las computadoras comprender imágenes, ahora se usa en tiendas de conveniencia, pruebas de automóviles sin conductor, diagnósticos médicos diarios y en el monitoreo de cultivos y ganado. Se ha visto que las computadoras son competentes para reconocer imágenes. Hoy en día, las empresas de tecnología punta como Amazon, Google, Microsoft y Facebook están invirtiendo miles de millones de dólares en investigación de visión computarizada y desarrollo de productos. [16]

Teniendo esto en cuenta, decidimos descubrir cómo las principales empresas de tecnología global utilizan la visión por computadora y explorar qué tipo de nueva tecnología y medios podrían aparecer en los próximos años.

De nuestra investigación, hemos encontrado que muchos de los casos de uso de la visión de computadora se encuentran en los siguientes grupos: [16]

- Seguridad al por menor y al por menor
- Automotor
- Cuidado de la salud
- Agricultura
- Bancario

- Industrial

2.2.4 Detección De Objetos

La detección de objetos es una de las aplicaciones de inteligencia artificial más desarrolladas y es por eso que tiene tantas aplicaciones, se podría decir que entre todas las aplicaciones de AI es la que mayor rentabilidad tiene para los usuarios comunes que no necesitan saber cómo funciona o como trabaja. “La detección de objetos tiene la característica de identificar todas instancias o características de un objeto determinado dentro de una imagen” [17]

En la ***Figura 1*** se puede observar un ejemplo de cómo se observaría la detección de objetos usando visión artificial.

Figura 1. *Detección de Objetos. [17]*

Aplicaciones de detección de objetos.

Reconocimiento facial:

Un grupo de investigadores ha desarrollado un sistema de reconocimiento facial de aprendizaje profundo denominado " **DeepFace** " en **Facebook,** que identifica rostros humanos en una imagen digital de manera muy eficaz. **Google** utiliza su propio sistema de reconocimiento facial en Google Photos, que separa automáticamente todas las fotos según la persona que aparece en la imagen. [18]

En la ***Figura 2*** se puede observar un ejemplo de varios componentes involucrados en el reconocimiento facial como los ojos, la nariz, la boca y las cejas.

Figura 2. *Detección de reconocimiento facial. [18]*

Recuento de personas:

La detección de objetos también se puede usar para el conteo de personas, se usa para analizar el rendimiento de la tienda o las estadísticas de multitudes durante los festivales. Estos tienden a ser más difíciles a medida que las personas salen del cuadro rápidamente. [18]

En la ***Figura 3*** se observa una aplicación muy importante, ya que durante la recopilación de multitudes esta función se puede usar para múltiples propósitos.

Figura 3. *Detección y recopilación de multitudes. [18]*

Control de calidad industrial:

La detección de objetos también se utiliza en procesos industriales para identificar productos. Encontrar un objeto específico a través de la inspección visual es una tarea básica que está involucrada en múltiples procesos industriales como la clasificación, la gestión de inventario, el mecanizado, la gestión de calidad, el embalaje, etc.

La administración del inventario puede ser muy complicada ya que los artículos son difíciles de rastrear en tiempo real. El conteo y la localización automática de objetos permite mejorar la precisión del inventario. [18]

Autos de auto conducción:

Los autos autónomos son el futuro, no hay duda de eso. Pero trabajar detrás es muy complicado, ya que combina una variedad de técnicas para percibir su entorno, como el radar, la luz láser, el GPS, la odometría y la visión por computadora. Los sistemas de control avanzados interpretan la información sensorial para identificar las rutas de navegación apropiadas, así como los obstáculos y una vez que el sensor de imagen detecta cualquier signo de un ser vivo en su camino, se detiene automáticamente. Esto sucede a un ritmo muy rápido y es un gran paso hacia los coches sin conductor. [18]

En la ***Figura 4*** se puede observar la aplicación que tiene la detección de objetos en los autos.

Figura 4. *Uso de la detección de objetos en el campo automovilístico. [18]*

Seguridad:

La detección de objetos juega un papel muy importante en la seguridad. Ya sea la identificación de la cara de Apple o el escaneo de retina utilizado en todas las películas de ciencia ficción. También es utilizado por el gobierno para acceder a la fuente de seguridad y relacionarlo con su base de datos existente para encontrar delincuentes o para detectar el vehículo de los ladrones, las aplicaciones son ilimitadas.

2.2.5 Robot omnidireccional autónomo KUKA YouBot

YouBot, un robot móvil omnidireccional de KUKA está destinado a la investigación y la educación. KUKA, conocido como uno de los principales fabricantes de robots industriales del mundo, ha diseñado aquí una plataforma rica, asequible y abierta. Con un pequeño brazo KUKA con 5 grados de libertad, su API compatible con ROS y una

carga útil de 15 kg (20 kg si retiramos el brazo), el YouBot de KUKA es la plataforma ideal para experimentos de manipulación móvil o grandes concursos internacionales de robótica. El KUKA YouBot tiene una placa para alojar sensores (los buscadores de rango láser de Hokuyo o el buscador de rango de láser enfermo, Xtion Pro Live o Microsoft Kinect). [19]

Figura 5. *Características del robot móvil KUKA YouBot con brazo. [19]*

En la ***Tabla 1*** se puede observar todas las características tomadas desde el datashet del robot autónomo.

Tabla 1. *Características del robot Kuka*

Fuente: *Investigador*

Características	
Base móvil omnidireccional	Pinza paralela con 2 dedos.
Peso de la base móvil: 20 kg.	Comunicación en tiempo real EtherCAT.
Peso del brazo: 5,3 kg.	PC mini ITX integrado con CPU, 2 GB de RAM, 32 GB de SSD Flash, USB, 6xUSB 2.0, 1xVGA, 2xLAN (solo 1 disponible)
Carga útil de la base móvil con el brazo: 15kg.	Entrada DC: 12V
Carga útil del brazo: 0,5 kg.	Bateria recargable 24V, 5Ah
Controladores de motor para la gama TMCM de Trinamic	Autonomía: alrededor de 90 minutos.
Manipulador de brazo con 5 grados de libertad.	Imagen Ubuntu específica de Kuka (Disponible para descargar)
Interfaces abiertas	El brazo y la plataforma se pueden utilizar de forma independiente
Software de control	Fuente de alimentación

El protocolo de comunicación definido para el Kuka YouBot utiliza la red estándar Ethernet de tipo Simple Open Source EtherCAT Master (SOEM), el cual permite la implementación de un controlador de código abierto para el maestro.

2.2.6 Sistema Operativo Robótico

El sistema operativo de robot (ROS) no es un sistema operativo real, sino un marco y un conjunto de herramientas que proporcionan la funcionalidad de un sistema operativo en un grupo de computadoras heterogéneas. Su utilidad no se limita a los robots, sino que la mayoría de las herramientas proporcionadas se centran en trabajar con hardware periférico. ROS se divide en más de 2000 paquetes, y cada paquete proporciona una funcionalidad especializada. El número de herramientas conectadas al marco es probablemente su mayor poder. [20]

¿Por qué debería usar Robot OS?

ROS proporciona funcionalidad para la abstracción de hardware, controladores de dispositivos, comunicación entre procesos en múltiples máquinas, herramientas para prueba y visualización, y mucho más. La característica clave de ROS es la forma en que se ejecuta el software y la forma en que se comunica, lo que le permite diseñar software complejo sin saber cómo funciona cierto hardware. ROS proporciona una forma de conectar una red de procesos (nodos) con un concentrador central. Los nodos se pueden ejecutar en múltiples dispositivos y se conectan a ese hub de varias maneras. [21]

Las principales formas de crear la red son proporcionar servicios solicitados o definir conexiones de editor / suscriptor con otros nodos. Ambos métodos se comunican a través de tipos de mensajes especificados. Los paquetes principales proporcionan algunos tipos, pero los paquetes individuales pueden definir los tipos de mensajes.
Los desarrolladores pueden ensamblar un sistema complejo conectando las soluciones existentes para pequeños problemas. La forma en que se implementa el sistema, nos permite: [20]

- Reemplace los componentes con interfaces similares sobre la marcha, eliminando la necesidad de detener el sistema para varios cambios

- Multiplexación de salidas de múltiples componentes en una entrada para otro componente, permitiendo la resolución paralela de varios problemas
- Conecte componentes creados en varios lenguajes de programación simplemente implementando los conectores adecuados al sistema de mensajería, facilitando el desarrollo de software mediante la conexión de módulos existentes de varios desarrolladores.
- Cree nodos a través de una red de dispositivos, sin preocuparse por dónde se ejecuta el código e implementando sistemas de comunicación entre procesos (IPC) y llamada a procedimiento remoto (RPC)
- Conéctese directamente a las fuentes a pedido del hardware remoto sin escribir ningún código adicional, empleando los dos puntos anteriores

2.2.7 Lenguaje de programación para inteligencia artificial

La programación de AI es una elevación de la tecnología que ha brindado eficiencia y beneficios óptimos a las diferentes operaciones de la compañía y la vida de las personas. AI ha traído otro nivel de tecnología inteligente a diferentes industrias y las perspectivas de su potencial aún crecen con la expectativa de que alcanzaría la inteligencia humana. Esto se debe a que los desarrolladores están dispuestos a explorar, experimentar e implementar sus capacidades para satisfacer más necesidades humanas y organizativas. Después de todo, la necesidad es la madre de la invención.

Al igual que en el desarrollo de la mayoría de las aplicaciones de software, un desarrollador tiene una variedad de lenguajes para usar en la escritura de AI. Sin embargo, no hay un lenguaje de programación perfecto para señalar como el mejor lenguaje de programación usado en inteligencia artificial. El proceso de desarrollo depende de la funcionalidad deseada de la aplicación de AI que se está desarrollando. Hasta ahora, AI ha logrado inteligencia biométrica, pilotos automáticos para autos que conducen por sí mismos y otras aplicaciones que requerían diferentes lenguajes de codificación de inteligencia artificial para sus proyectos de desarrollo. [22]

Java, Python, Lisp, Prolog y C ++ son los principales lenguajes de programación de inteligencia artificial utilizados para la inteligencia artificial, capaces de satisfacer diferentes necesidades en el desarrollo y diseño de diferentes programas. Corresponde a un desarrollador elegir cuál de los idiomas de AI gratificará la funcionalidad y las características deseadas de los requisitos de la aplicación. [22]

Python

Se encuentra entre los lenguajes de programación favoritos de los desarrolladores en el desarrollo de IA debido a su sintaxis, simplicidad y versatilidad. Python es muy alentador para el aprendizaje automático para los desarrolladores, ya que es menos complejo en comparación con C ++ y Java. También es un lenguaje muy portátil, ya que se utiliza en plataformas que incluyen Linux, Windows, Mac OS y UNIX. También es agradable por sus características como Interactivo, interpretado, modular, dinámico, portátil y de alto nivel que lo hacen más único que Java. [22]

En la ***Tabla 2*** se puede observar todas las ventajas y desventajas de usar Python

Tabla 2. *Ventajas y desventajas de usar Python.*

Fuente: *Investigador*

Ventajas	Desventajas
Python tiene una rica y extensa variedad de bibliotecas y herramientas.	Los desarrolladores acostumbrados a usar Python enfrentan dificultades para adaptarse a una sintaxis completamente diferente cuando intentan usar otros lenguajes para la programación de IA.
Soporta pruebas de algoritmo sin necesidad de implementarlas.	A diferencia de C ++ y Java, python trabaja con la ayuda de un intérprete que hace que la compilación y la ejecución sean más lentas en el desarrollo de AI.
Python que admite el diseño orientado a objetos aumenta la productividad de un programador.	No apto para computación móvil. Para AI destinado a aplicaciones móviles, Python no es adecuado debido a su lenguaje débil para la informática móvil.
Comparado con Java y C ++, Python es más rápido en desarrollo.	

C ++

C ++ es el lenguaje informático más rápido, su velocidad es apreciada para proyectos de programación de AI que son sensibles al tiempo. Proporciona una ejecución más rápida y tiene menos tiempo de respuesta que se aplica en los motores de búsqueda y el desarrollo de juegos de computadora. Además, C ++ permite el uso extensivo de algoritmos y es eficiente en el uso de técnicas estadísticas de IA. Otro factor importante es que C ++ admite la reutilización de programas en desarrollo debido a la herencia y la ocultación de datos, por lo tanto, eficiente en ahorro de tiempo y costos. [22]

En la ***Tabla 3*** se puede observar todas las ventajas y desventajas de usar C++

Tabla 3. *Ventajas y desventajas de usar C++.*

Fuente: *Investigador*

Ventajas	Desventajas
Bueno para encontrar soluciones para problemas complejos de inteligencia artificial.	Pobre en la multitarea; C ++ es adecuado solo para implementar el núcleo o la base de sistemas o algoritmos específicos.
Rico en funciones de biblioteca y colección de herramientas de programación.	Sigue el enfoque de abajo hacia arriba, por lo que es muy complejo, lo que dificulta a los desarrolladores novatos usarlo para escribir programas de inteligencia artificial.
Programación de múltiples paradigmas que admite principios orientados a objetos, por lo que es útil para lograr datos organizados.	
Comparado con Java y C ++, Python es más rápido en desarrollo.	

JAVA

Java (sitio web oficial) es otro lenguaje de programación para responder "¿qué lenguaje de computadora se usa para la inteligencia artificial?" Java también es un lenguaje de múltiples paradigmas que sigue los principios orientados a objetos y el

principio de Once Written Read / Run Anywhere (WORA). Es un lenguaje de programación AI que puede ejecutarse en cualquier plataforma que lo admita sin la necesidad de una recompilación.

Java es uno de los más utilizados y no solo en el desarrollo de AI. Deriva una parte importante de su sintaxis de C y C ++, además de sus herramientas menores. Java no solo es apropiado para NLP y algoritmos de búsqueda, sino también para redes neuronales. [22]

En la ***Tabla 4.*** se puede observar todas las ventajas y desventajas de usar Java

Tabla 4. *Ventajas y desventajas de usar java.*

Fuente: *Investigador*

Ventajas	Desventajas
Es fácil de implementar en diferentes plataformas gracias a la tecnología de máquina virtual.	Java es más lento que C ++, tiene menos velocidad de ejecución y más tiempo de respuesta.
A diferencia de C ++, Java es fácil de usar e incluso de depuración.	Aunque es altamente portátil, en plataformas más antiguas, Java requeriría cambios dramáticos en el software y el hardware para facilitar.
Tiene un administrador de memoria automático que facilita el trabajo del desarrollador.	Java también es un lenguaje AI de programación generalmente inmaduro, ya que todavía hay algunos desarrollos en curso, como JDK 1.1 en beta.

CECEO

LISP es otro lenguaje utilizado para el desarrollo de la inteligencia artificial. Es una familia de lenguajes de programación informática y es el segundo lenguaje de programación más antiguo después de Fortran. LISP se ha desarrollado con el tiempo para convertirse en un lenguaje fuerte y dinámico en la codificación.

Algunos consideran a LISP como el mejor lenguaje de programación de inteligencia artificial debido al favor de la libertad que ofrece a los desarrolladores. LISP se usa en

la IA debido a su flexibilidad para la creación rápida de prototipos y la experimentación, lo que a su vez facilita que LISP crezca a un lenguaje de IA estándar. Por ejemplo, LISP tiene un sistema de macros único que facilita la exploración y la implementación de diferentes niveles de Inteligencia Intelectual. [22]

En la ***Tabla 5.*** se puede observar todas las ventajas y desventajas de usar CECEO

Tabla 5. *Ventajas y desventajas de usar CECEO.*
Fuente: *Investigador*

Ventajas	Desventajas
Rápido y eficiente en la codificación, ya que es compatible con compiladores en lugar de intérpretes.	Pocos desarrolladores conocen bien la programación de Lisp.
El administrador de memoria automático se inventó para LISP, por lo tanto, tiene una recolección de basura.	Al ser un lenguaje de programación antiguo de inteligencia artificial, LISP requiere la configuración de un nuevo software y hardware para adaptarse a su uso.
LISP ofrece control específico sobre los sistemas que resultan para su uso máximo.	

PROLOG

Prolog es también uno de los lenguajes de programación más antiguos, por lo que también es adecuado para el desarrollo de la programación AI. Al igual que Lisp, también es un lenguaje informático principal para la inteligencia artificial. Tiene mecanismos que facilitan la creación de marcos flexibles con los que los desarrolladores disfrutan trabajar. Es un lenguaje basado en reglas y declarativo, ya que contiene hechos y reglas que dictan su lenguaje de codificación. [22]

En la ***Tabla 6.*** se puede observar todas las ventajas y desventajas de usar Prolog

Tabla 6. *Ventajas y desventajas de usar Prolog.*

Fuente: *Investigador*

Ventajas	Desventajas
Prolog tiene un manejo de listas integrado esencial para representar estructuras de datos basadas en árboles.	A pesar de la edad avanzada del prólogo, no se ha estandarizado completamente en que algunas características difieren en la implementación, lo que hace que el trabajo del desarrollador sea engorroso.
Eficiente para la creación rápida de prototipos para los programas de AI que se lanzarán con frecuencia los módulos.	
Permite la creación simultánea de bases de datos con la ejecución del programa.	

Comparación entre los diferentes lenguajes de programación usados en IA.

Se puede observar en la ***Tabla 7.*** Los diferentes lenguajes de programación usados en inteligencia artificial, detallando el número de lanzamientos, las calificaciones y los cambios que han tenido. [23]

Cualquiera de los lenguajes de programación de esta lista podría ser perfectamente válido para desarrollar aplicaciones de IA, a excepción probablemente del lenguaje SQL, que está orientado a consultas en bases de datos, en lugar de a desarrollo de aplicaciones. [23]

Tabla 7. *Tabla comparativa de los lenguajes de programación.*

Fuente: *Investigador*

Enero 2019	Enero 2018	Lenguaje de Programación	Calificaciones	Cambio
1	1	Java	16.904%	+2.69%
2	2	C	13.337%	+2.30%
3	4	Python	8.294%	+3.62%
4	3	C++	8.158%	+2.55%
5	7	Visual Basic .NET	6.459%	+3.20%
6	6	JavaScript	3.302%	-0.16%
7	5	C#	3.284%	-0.47%
8	9	PHP	2.680%	+0.15%
9	–	SQL	2.277%	+2.28%
10	16	Objective-C	1.781%	-0.08%

2.2.8 Plataformas de Inteligencia Artificial para desarrolladores

El campo de AI está ligado esencialmente al procesamiento de lenguaje natural y el reconocimiento de voz es prometedor. En la actualidad existes diferentes placas para desarrolladores esta clase de aplicaciones. Los procesos de inteligencia artificial fueron de las áreas de más desarrollo en el 2015 y lo continúan siendo hasta el 2019, con grandes proyecciones hacía muchos años más. La verdad es que fue, es y seguirá siendo el sector que genera mayores beneficios. Haciendo la comparación imaginando

un terreno de cientos de hectáreas de tierra cultivable, pero pocas semillas plantadas. Las grandes empresas como Google, Facebook o Microsoft ya van tiempo entrando en el desarrollo de herramientas de Inteligencia Artificial, siempre aplicados a procesamiento de lenguaje natural, al reconocimiento de voz y visión artificial. [24]
En la ***Tabla 8.*** Están las principales placas para el desarrollo de IA, detallando su creador y su característica principal, esta tabla ayudara a elegir la mejor opción.

Tabla 8. *Plataformas de desarrollo con mejor respuesta en la IA.*
Fuente: *Investigador*

Nombre	Creador	Característica
Samsung ARTIK 710	Samsung	módulos de hardware y servicios en la nube
SensorTile	STMicroelectronics	Aplicaciones de IoT de baja potencia
ADALM-PLUTO	AnalogDevices	teoría de RF y la práctica de RF
AudioSmart 2-Mic	Amazon AVS y Conexant	reconocimiento de voz
Ultrahaptics	Ultrahaptics	Comunicación inalámbrica
Raspberry Pi 3 Modelo B	Raspberry Pi	propia pieza personalizada de tecnología electrónica
ThunderboardSense 2	Silicon Labs	Solución personalizada habilitada para la nube.
i.MX8	NXP	Aplicaciones multimedia.
Jetson TX2	NVIDIA	Informática periférica, inteligencia artificial y supercomputadoras
Arduino Uno	Arduino	adquisición de datos o incluso manejo de un robot
NVIDIA Jetson Nano	NVIDIA	Soluciones IA, reconocimiento de voz, integración de sensores.

CAPÍTULO III

METODOLOGÍA

3.1 Modalidad de la Investigación

El presente proyecto es una investigación aplicada ya que resolverá situaciones problemáticas, aplicando conocimientos ya existentes para la solución de un determinado problema, y proponiendo un prototipo de solución.

Se empleará esta modalidad debido a que el proyecto se fundamentará en fuentes como; libros, revistas, publicaciones científicas, proyectos de investigación, datasheets de equipos y recomendaciones técnicas, referidos a Navegación Autónoma y detección de posturas humanas, información necesaria para adquirir los conocimientos que permitan desarrollar de mejor manera la investigación.

Se empleará esta modalidad debido a que la investigación debe ser verificada experimentalmente, al inicio usando software para su simulación y posteriormente utilizando elementos reales, que nos permitirán encontrar un óptimo funcionamiento.

3.2 Recolección de Información

Para la recolección de información se optará como parámetro el uso de documentos, revistas, libros, proyectos desarrollados, por lo que se tomará en cuenta bases de datos confiables que permitirán la obtención de información para el desarrollo del proyecto.

3.3 Procesamiento y Análisis de Datos

En el proceso y análisis de datos se establecen las siguientes etapas:

- Definición del foco de estudio por medio la veracidad de los datos obtenidos, esto a través de la clasificación de datos de manera ordenada y sistemática

- Recopilación de datos que aporten información técnica por medio de tesis, libros, revistas, periódicos, encuestas relacionadas a Navegación Autónoma y detección de objetos
- Clasificación de información para almacenar datos útiles y deshacer los datos no relevantes
- Interpretación de la información procesada por medio de gráficos, tablas y esquemas estadísticos.
- Análisis de los resultados obtenidos, se presentará en un informe técnico destacando los datos requeridos en los objetivos planteados.

3.4 Desarrollo del Proyecto

El proyecto será desarrollado acorde a las siguientes actividades:

1. Estudio de Inteligencia y Visión Artificial.
2. Descripción de la detección de objetos y los métodos que esta conlleva.
3. Análisis de la posición, velocidad y aceleración del Robot autónomo KUKA YouBot.
4. Análisis de estimaciones de la posición relativa del Robot autónomo KUKA YouBot.
5. Análisis de cada uno de los algoritmos de detección de posturas humanas.
6. Selección del algoritmo que mejor se adapte a la detección de posturas humanas.
7. Desarrollo e implementación del algoritmo seleccionado.
8. Estudio de la integración y el funcionamiento de los entornos de trabajo ROS y un lenguaje de programación para inteligencia artificial
9. Simulación de la navegación autónoma del robot utilizando Gazebo con el algoritmo seleccionado.
10. Realización de pruebas, detección y corrección de errores.
11. Elaboración del informe final.

CAPÍTULO IV

DESARROLLO DE LA PROPUESTA

El presente proyecto de investigación esta enfatizado en realizar una navegación autónoma del Robot Kuka YouBot, usando comandos generados por un sistema de inteligencia artificial idónea para controlar y planear de manera eficiente las trayectorias de un robot omnidireccional autónomo, para el mejoramiento del libre movimiento en un ambiente de difícil acceso.

El sistema de visión artificial que se ejecuta dentro de la tarjeta de desarrollo permite obtener las posturas humanas del operario del robot y enviar los datos inalámbricamente hacia el robot autónomo, esto con el fin de que el operario del robot Kuka pueda decir la ruta que quiere que el robot tome para cumplir la tarea deseada, siendo este proyecto de gran aporte hacia el proyecto de investigación denominado "Plataforma Móvil Omnidireccional KUKA dotada de Inteligencia Artificial utilizando estrategias de Machine Learnig para Navegación Segura en Espacios no Controlados".

4.1 Factibilidad

4.1.1 Factibilidad Institucional

El presente proyecto de investigación posee factibilidad institucional, porque fue implementado en la oficina de asistentes de investigación de la Facultad de Tecnologías de la Información, Telecomunicaciones e Industrial, de la Universidad Técnica de Ambato.

4.1.2 Factibilidad Técnica

La ejecución del presente proyecto de investigación tiene factibilidad técnica dado que los equipos utilizados en el desarrollo de la aplicación son de fácil acceso gracias a que la investigación forma parte del proyecto de investigación denominado "Plataforma Móvil Omnidireccional KUKA dotada de Inteligencia Artificial utilizando estrategias de Machine Learnig para Navegación Segura en Espacios no Controlados".

4.1.3 Factibilidad Bibliográfica

El proyecto de investigación cuenta con esta factibilidad ya que la información necesaria es encontrada repositorios de las diferentes Universidades nacionales y extranjeras, como también libros, revistas, publicaciones científicas, proyectos de investigación, datasheets de equipos y recomendaciones técnicas, referidos al tema tratado.

4.1.4 Factibilidad Económica

Este proyecto cuenta con la factibilidad económica ya que el valor final del presente proyecto es accesible para el investigador.

4.2 Estudio de inteligencia y visión artificial.

4.2.1 Visión global de la visión artificial

El punto central de la visión artificial es poder sustituir a la visión humana, entendiéndola como una disciplina con la cual podemos entender, analizar y captar objetos o imágenes de un lugar específico con la ayuda de un de una cámara y un ordenador central. [25]

Un sistema de visión artificial constituye diferentes procesos que se los puede describir generalmente con la idea de que una imagen de recibe por medio de una cámara o algún tipo de sensor de visión que se aproximaría al de ojo humano, este sistema tiene como resultado imágenes bidimensionales de una realidad y un entorno tridimensional. [25]

4.2.2 Visión Humana vs Visión Artificial

Los sensores de visión o las cámaras emulan el funcionamiento del globo ocular en los humanos, luego de esto el ordenador central se encarga del procesado de la información aproximándose al comportamiento del cerebro humano. Si bien podemos encontrar un gran parentesco en los 2 sistemas, pero al ser imposible conocer cuál es el mecanismo sofisticado que utiliza el cerebro humano para procesar la información obtenida por el sentido de la vista, limita la posibilidad de igualar artificialmente dicha percepción, esto es algo que no se podrá lograr ahora ni en mucho tiempo. [25]

El sentido de vista humana cuanta también con un excelente sistema de reconocimiento de objetos, ya que tiene la capacidad de adaptación a situaciones imprevistas, cuenta con una excelente resolución y funciona mejor en situaciones que necesiten mayor cantidad de procesamiento. Por otra parte, el sistema de visión artificial tiene la característica de hacer tareas de un modo más efectivo que la de visión humana, como por ejemplo tener la ventaja de no solo captar el rango de luz visible sino poder detectar todo el espectro electromagnético, otro ejemplo son las velocidades de respuesta reducidas que cuenta la visión artificial en comparación a la humana, en este caso la visión humana por los tiempos alargados de trabajo le puede afectar el cansancio, la fatiga o las distracciones del entorno, a comparación que la visión artificial puede trabajar con el mismo nivel de rendimiento en toda su vida útil y además en lugares en donde el sistema de visión humano podría tener riesgo o peligro. Es por esto que el sistema de visión artificial puede realizar algunos procesos eficazmente y tiene ventajas en comparación al de visión humana. [25]

4.2.3 Visión Artificial en la Industria

En los últimos años el campo de la visión a nivel industrial ha ido creciendo a pasos agigantados estos se aplican a niveles de inspección ya que por los complejos procesos que maneja la industria nace la necesidad de aplicar un control de calidad que verifique características definidas de un producto. En dicho control se verifica el producto y se determina el cumplimiento de las especificaciones técnicas, estos controles se hacen al final o al intermedio de un proceso en donde se conoce si los procesos realizados han sido realizados correctamente para poder continuar con la producción y así evitar que el sistema siguiera trabajando con un proceso defectuoso.La creciente instalación de sistemas de visión artificial se debe al aumento competitivo de fabricantes y

proveedores que buscan entregar un buen producto a las industrias receptoras, esto ha desencadenado que las empresas mejoren sus sistemas de automatización es una inspección en línea de los procesos mediante visión artificial. Estos sistemas de inspección en línea tienen grandes ventajas en comparación o a otros sistemas de inspección, los cuales necesitan que los productos estén posicionados de maneras específicas y tener la necesidad de tener contacto directo con los productos para hacer el control, en cambio la inspección por visión artificial no necesita nada de esto evitando posibles daños o paro de procesos innecesarios. [25]

Los sistemas de inspección industrial deben tener las siguientes características

- Robustos: Tienen que ser inmunes a variaciones que se presenten en el entorno
- industrial.
- Fiables y Precisitos.: Deben hacer que la inspección obtenga un índice de error de
- reconocimiento muy bajo.
- Veloces: Deben obtener un tiempo de inspección necesario evitando que el sistema de visión se convierta en un cuello de botella del proceso de fabricación.
- Disponibles de servicio técnico: Tienen que tener la posibilidad de ante cualquier fallo o daño sean fácilmente su reparación.
- Flexibles: Esta característica es muy importante ya que deben tener la posibilidad de aplicarse en otro proceso o ayudar a fortalecer otro diferente.
- Costo: La inversión económica en un sistema de visión artificial no es fácil de pronosticar a simple vista y es necesario considerar todos los posibles costos que este conlleva como por ejemplo el mantenimiento, el arreglo o la prevención de fallos. Pero hay que tomar en cuenta que este sistema ahorra tiempo de ejecución y ayudaría a una mejora en el tiempo de entrega de productos. [25]

4.3 Descripción de la detección de objetos

La detección de objetos es el proceso de encontrar instancias de objetos del mundo real como automóviles, bicicletas, televisores, flores y personas en imágenes fijas o

videos. Permite el reconocimiento, la localización y la detección de múltiples objetos dentro de una imagen, lo que nos permite comprender mucho mejor la imagen en su conjunto. Se usa comúnmente en aplicaciones como los sistemas de recuperación de imágenes, seguridad, vigilancia y asistencia avanzada para el conductor. [16]

La detección de objetos se puede hacer de varias maneras:

- Detección de objetos basada en características
- Detección de objetos Viola Jones
- Clasificaciones SVM con características HOG
- Detección de objetos de aprendizaje profundo

4.3.1 Seguimiento de objetos

El seguimiento de objetos se refiere al proceso de seguir un objeto específico de interés, o varios objetos, en una escena determinada. Tradicionalmente, tiene aplicaciones en video e interacciones del mundo real donde las observaciones se realizan después de una detección inicial de objetos. Ahora, es crucial para los sistemas de conducción autónomos, como los vehículos de auto conducción de compañías como Uber y Tesla. [26]

En la ***Figura 6.*** se puede observar un ejemplo de de seguimiento de objetos en este caso es el seguimiento a diferentes personas.

Figura 6. *Seguimiento de Objetos con visión artificial. [26]*

4.3.2 Métodos de seguimiento de objetos

Los métodos de seguimiento de objetos se pueden dividir en 2 categorías según el modelo de observación:

- Método generativo

- Método discriminativo.

El método generativo utiliza el modelo generativo para describir las características aparentes y minimiza el error de reconstrucción para buscar el objeto. El método discriminativo se puede usar para distinguir entre el objeto y el fondo, su rendimiento es más robusto y, gradualmente, se convierte en el método principal de seguimiento. El método discriminativo también se conoce como Seguimiento por Detección, y el aprendizaje profundo pertenece a esta categoría. Para lograr el seguimiento por detección, detectamos objetos candidatos para todos los cuadros y utilizamos el aprendizaje profundo para reconocer el objeto deseado de los candidatos. Hay 2 tipos de modelos de red básicos que se pueden usar: auto encoders apilados (SAE) y redes neuronales convolucionales (CNN). [26]

La red profunda más popular para el seguimiento de tareas que utilizan SAE es Deep Learning Tracker , que propone una formación previa sin conexión y un ajuste en línea de la red. El proceso funciona así:

- Fuera de línea sin supervisión, entrene previamente el auto-codificador de eliminación de ruido apilados utilizando conjuntos de datos de imágenes naturales a gran escala para obtener la representación general del objeto. El auto-codificador de eliminación apilada puede obtener una capacidad de expresión de características más robusta al agregar ruido en las imágenes de entrada y reconstruir las imágenes originales.
- Combine la parte de codificación de la red entrenada previamente con un clasificador para obtener la red de clasificación, luego use las muestras positivas y negativas obtenidas del marco inicial para ajustar la red, lo que puede discriminar el objeto y el fondo actuales. DLT usa el filtro de partículas como modelo de movimiento para producir parches candidatos del cuadro actual. La red de clasificación genera los puntajes de probabilidad para estos parches, es decir, la confianza de sus clasificaciones, luego elige el más alto de estos parches como el objeto.
- En la actualización del modelo, DLT utiliza la forma de umbral limitado. [26]

4.4 Análisis del Robot autónomo KUKA YouBot.

4.4.1 Robots móviles Omnidireccionales

- Esta clase dé robot tiene la característica de contar con la máxima maniobrabilidad en una superficie plana, esta puede maniobrar en cualquier dirección sin la necesidad de reorientarse, esta es una gran ventaja en comparación a otra clase de robot.
- Los robots omnidireccionales pueden ser ensamblados con tres, cuatro o más ruedas omnidireccionales. Como se muestra en la ***Figura 7***. Los que cuentan con tres ruedas utilizan ruedas universales, las cuales forman un ángulo de 90˚ con el rodillo como se observa en la ***Figura 7***. ***(a)***, esto hace que esta clase de robot tengan control y dirección simple. Por el contrario, los que utilizan cuatro o más ruedas usan el tipo Mecanum y esto conlleva a que tengan mecánica y control complejo con la gran ventaja de una mayor estabilidad y tracción, como las que se muestran en la ***Figura 7***. ***(b).*** [27]

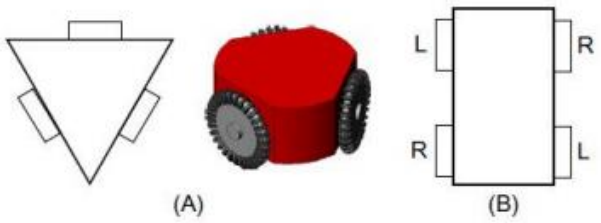


Figura 7. *Robots Omnidireccionales.*

(a) omnidireccionales de 3 ruedas, (b) omnidireccionales de 2 ruedas
Fuente: *G. Tzafestas,2014*

4.4.2 Modelo cinemático del Kuka YouBot

Modelo cinemático de la plataforma omnidireccional.

El robot Kuka YouBot consta de una plataforma omnidireccional de cuatro ruedas tipo Mecanum que permiten el libre desplazamiento del sistema en el plano Cartesiano, sin embargo, es necesario considerar la disposición de los rodillos de este tipo de ruedas para obtener un modelo cinemático fidedigno. Las medidas de la plataforma móvil se muestran en la Figura 8. y están expresadas en milímetros. [27]

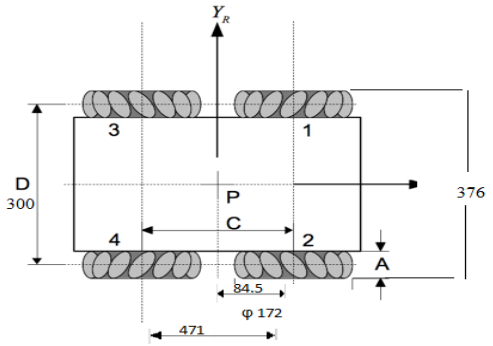


Figura 8. *Medidas de la plataforma móvil.*

Fuente: *Investigador*

De acuerdo con Nagatani et al. (2000), la configuración cinemateca de un robot móvil omnidireccional está dada por el conjunto de ecuaciones (1).

$$\dot{x} = v_l = \frac{1}{4}(d_1 + d_2 + d_3 + d_4)$$

$$\dot{y} = v_t = \frac{1}{4}(-d_1 + d_2 + d_3 - d_4)\tan(\alpha_b) \qquad (1)$$

$$\dot{\theta_b} = v_a = \omega = \frac{1}{4}(d_1 + d_2 + d_3 + d_4)\beta$$

En donde:

$\dot{x}$ = velocidad en el eje X = velocidad longitudinal.

$\dot{y}$ = velocidad en el eje Y = velocidad transversal.

$\dot{\theta_b}$= velocidad angular de la plataforma móvil omnidireccional.

α_b y β = ángulo de los rodillos de las ruedas (calculados experimentalmente). (***Figura 9***)

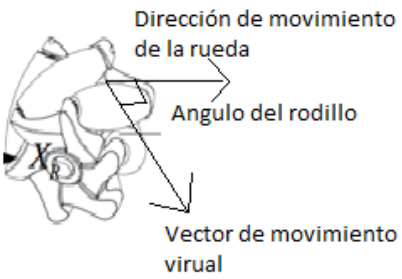


Figura 9. *Movimiento Rueda Mecanum.*

Fuente: *Investigador*

Entonces el movimiento en el plano dependerá de la combinación de las velocidades lineales de cada rueda de la plataforma. Los movimientos de traslación transversal y rotación se ilustran en la ***Figura 10.*** donde la plataforma móvil difuminada indica hacia donde se moverá la plataforma según la combinación de velocidades lineales de las 4 ruedas Mecanum. [28] [29]

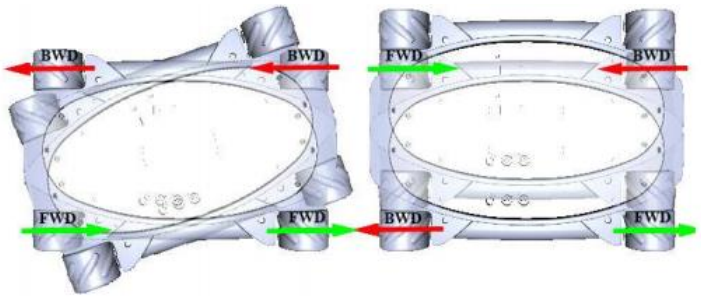


Figura 10. *Movimientos de la Plataforma móvil.*

Fuente: *Investigador*

Modelo cinemático del manipulador de 5 grados de libertad

El brazo manipulador del robot autónomo Kuka YouBot, tiene la característica de poseer 5 grados de libertad, todos estos grados son rotacionales. En la parte superior del brazo existe una pinza de 2 dedos que permite la manipulación de objetos pequeños. [28]

El modelo Cinemático del manipulador del robot autónomo Kuka YouBot se lo consigue mediante el algoritmo de Denavit-Hartenber desarrollado en el 2005, este algoritmo proporciona un procedimiento sistemático para la determinación de la posición y orientación del efecto final de un manipulador robótico de n-articulaciones.

En la ***Figura 11.*** se especifica las medidas de cada articulación, así como también sus límites articulares. [28]

Figura 11. *Medidas del manipulador del Kuka YouBot*

Fuente: *Investigador*

Los marcos de referencia para cada articulación se pueden apreciar en la ***Figura 12.*** y los parámetros Denavit-Hartenber obtenidos para el manipulador de muestran en la ***Tabla 9***, mientras que la ***Tabla 10*** se muestran las distancias y ángulos entre los marcos de referencia para el brazo manipulador.

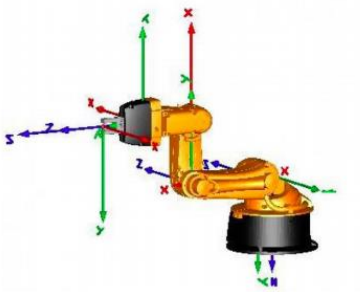


Figura 12. *Posiciones de referencia para cada articulación del manipulador Kuka*

Fuente: *Investigador*

Tabla 9. *Parámetros de Denavit-Hartenber para el manipulador Kuka.*

Fuente: *Investigador*

Eslabón	θ	d	A	α
1	q1	0.147	0.0330	$\frac{\pi}{2}$
2	q2	0	0.1550	0
3	q3	0	0.1350	0
4	$q4 + \frac{\pi}{2}$	0	0	$\frac{\pi}{2}$
5	Q5	0.2175	0	0

Tabla 10. *Cadena Cinemática del manipulador Kuka.*
Fuente: *Investigador*

	Marco anterior	Traslación (cm)			Rotación (grados)		
		x	y	Z	x	y	z
Articulación 1	Base	2.4	0	11.5	180^0	0^0	0^0
Articulación 2	Articulación 1	3.3	0	0	90^0	0^0	-90^0
Articulación 3	Articulación 2	15.5	0	0	0^0	0^0	-90^0
Articulación 4	Articulación 3	0	13.5	0	0^0	0^0	0^0
Articulación 5	Articulación 4	0	11.36	0	-90^0	0^0	0^0
Pinza	Articulación 5	0	0	5.716	90^0	0^0	180^0

Con los parámetros de las ***Tabla 8 y 9, es*** posible obtener las matrices de trasformación homogéneas de acuerdo a la ecuación (2). [29]

$$A_i^{i-1} = \begin{bmatrix} C\theta_i & -S\theta_i C\alpha_i & S\theta_i S\alpha_i & a_i C\theta_i \\ S\theta_i & C\theta_i C\alpha_i & -C\theta_i S\alpha_i & a_i S\theta_i \\ 0 & -S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

S = función sin ()

C= Función cos ()

La matriz de trasformación total se obtiene de la multiplicación sucesiva de cada una de las matrices homogéneas.

$$T_n^0 = A_1^0 \dots A_n^{n-1} \quad (3)$$

4.5 Algoritmos de detección de objetos para detección de Posturas Humanas

El reconocimiento de objetos es una metodología a partir de la cual se puede hacer la comparación de sectores o porciones que sean invariantes a cambios en la escala, la orientación y la iluminación entre imágenes para conformar la semántica de éstas y obtener su reconocimiento.

El reconocimiento de objetos se puede dividir en dos grupos:

Reconocimiento de Instancias. Implica reconocer un objeto previamente conocido dentro de una imagen, cuando es observado desde diferentes puntos de vista incluyendo la presencia de objetos extraños e incluso oclusiones.

Reconocimiento de Categorías. Consiste en reconocer un objeto que no ha sido visto y asignarle una categoría (por ejemplo “es un auto”, “es una persona”).

El reconocimiento de objetos mediante imágenes a color se viene investigando desde hace varios años. Algunos de los enfoques que se han desarrollado a lo largo del tiempo, son:

- Detección de líneas, contornos y/o superficies para luego compararlas con modelos 2D o 3D.
- Adquisición de imágenes desde diversas posiciones y orientaciones para representarlas en un espacio vectorial y realizar una descomposición en una base de datos con los valores propios más importantes (Ejemplo: Eigenfaces). [30]
- Extracción de un conjunto de características locales que tengan propiedades de invariancia frente a cambios de iluminación y punto de vista. Se reconoce una imagen comparando estas características locales con las características de las imágenes de la base conocida (debe de haber una suficiente cantidad de correspondencias y esas correspondencias deben ser coherentes con una transformación que alinee las imágenes). (Ejemplo: Matching con SIFT). [30]

Cuando la cantidad de imágenes consideradas crece, no es posible la comparación 1 a N. Las características locales son mapeadas a un conjunto de “palabras visuales”. Estas “palabras visuales” se pueden aprender mediante k-means sobre un conjunto de entrenamiento. El reconocimiento de una nueva imagen se realiza comparando las palabras visuales contra la frecuencia de aparición de éstas. Esto da un ranking de candidatos que se puede refinar teniendo en cuenta la coherencia espacial de correspondencias [30,31].

4.5.1 Redes neuronales convolucionales o CNN

En la ***Figura 13***.se presenta un ejemplo popular de ilustrar cómo funciona un algoritmo de detección de objetos. Cada objeto en la imagen, desde una persona a una cometa, ha sido localizado e identificado con un cierto nivel de precisión. [32]

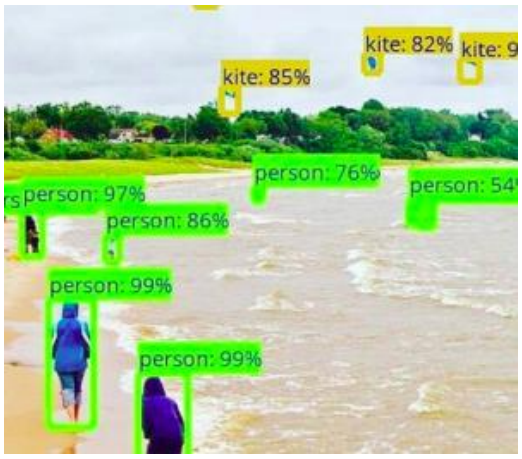


Figura 13. *Algoritmo de detección de objetos. [32]*

Comencemos con el enfoque de aprendizaje profundo más simple y ampliamente utilizado para detectar objetos en imágenes: redes neuronales convolucionales o CNN. Si su comprensión de las CNN está un poco oxidada, [32]

Pero resumiré brevemente el funcionamiento interno de una CNN para usted. Echa un vistazo a la ***Figura 14*** de abajo:

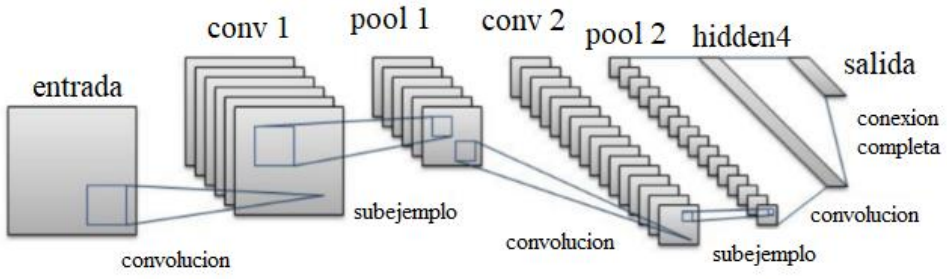


Figura 14. *Funcionamiento interno de una CNN.*
Fuente: *Investigador*

Pasamos una imagen a la red, y luego se envía a través de varias circunvoluciones y capas de agrupación. Finalmente, obtenemos la salida en la forma de la clase del objeto. Bastante sencillo, ¿no es así?

Para cada imagen de entrada, obtenemos una clase correspondiente como salida. ¿Podemos usar esta técnica para detectar varios objetos en una imagen? ¡Si podemos! Veamos cómo podemos resolver un problema general de detección de objetos utilizando una CNN. [32]

- Primero, tomamos una imagen como entrada:

Figura 15. *Imagen de entrada del CNC. [32]*

- Luego dividimos la imagen en varias regiones:

Figura 16. *División de regiones CNC. [32]*

- Luego consideraremos cada región como una imagen separada.
- Pase todas estas regiones (imágenes) a la CNN y clasifíquelas en varias clases.
- Una vez que hemos dividido cada región en su clase correspondiente, podemos combinar todas estas regiones para obtener la imagen original con los objetos detectados:

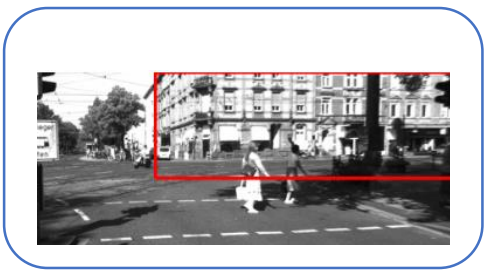


Figura 17. *Combinación de regiones CNC. [32]*

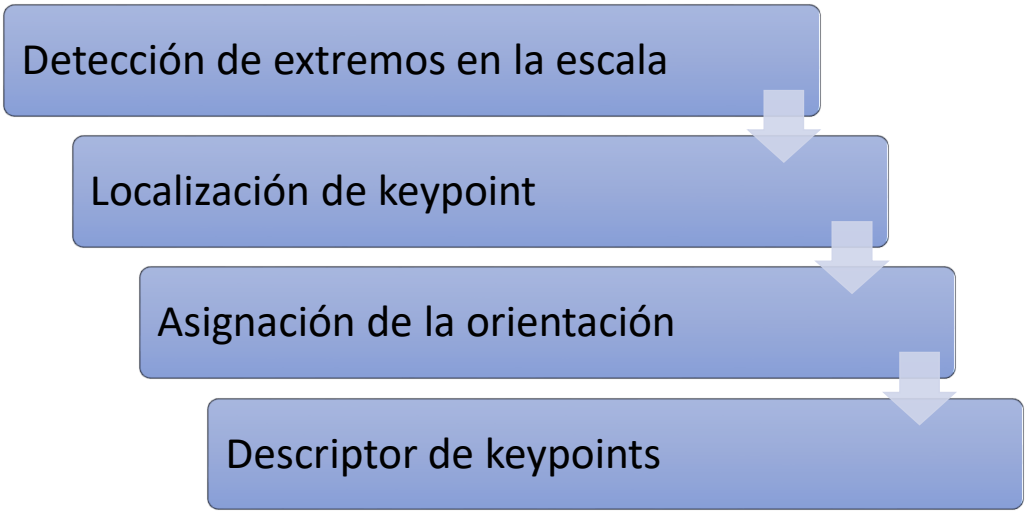
El problema con el uso de este enfoque es que los objetos en la imagen pueden tener diferentes relaciones de aspecto y ubicaciones espaciales. Por ejemplo, en algunos casos, el objeto podría estar cubriendo la mayor parte de la imagen, mientras que en otros el objeto podría estar cubriendo solo un pequeño porcentaje de la imagen. Las formas de los objetos también pueden ser diferentes (sucede mucho en los casos de uso de la vida real). [32]

Como resultado de estos factores, requeriríamos una gran cantidad de regiones, lo que resultaría en una gran cantidad de tiempo computacional. Entonces, para resolver este problema y reducir el número de regiones, podemos usar CNN basada en regiones, que selecciona las regiones mediante un método de propuesta. Entendamos lo que esta CNN basada en la región puede hacer por nosotros. [32]

4.5.2 Sift – Scale invariant feature transform

La transformación de característica invariante de escala fue publicado por David Lowe en 1999 [33] propone un algoritmo para la extracción de características de una imagen, que permitan describir los objetos contenidos en ella. [33]

El algoritmo descrito por Lowe consta de 4 etapas:



- Detección de extremos en la escala: La primera etapa del algoritmo realiza una búsqueda sobre las diferentes escalas y dimensiones de la imagen identificando posibles puntos de interés, invariantes a los cambios de orientación y escalado. Esto se lleva a cabo mediante la función DoG (Difference-of-Gaussian).

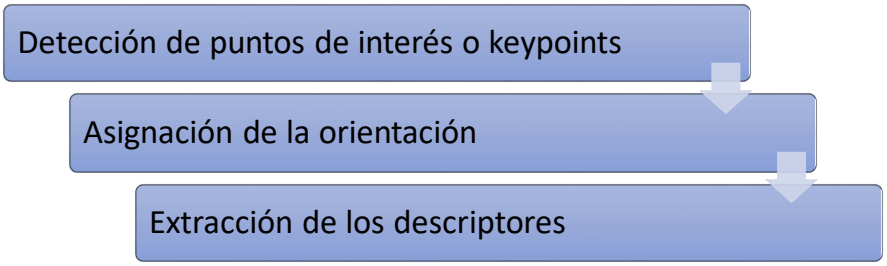
- Localización de keypoint: los puntos claves se eligen en base a medidas de estabilidad (se eliminan los puntos clave con bajo contraste o si se encuentran localizados en los bordes).
- Asignación de la orientación: La invarianza respecto de la rotación se consigue mediante la asignación a cada uno de los puntos una orientación basada en las propiedades locales de la imagen y representando el descriptor respecto de esta orientación.
- Descriptor de keypoints: Los gradientes locales de la imagen se miden en la región que rodea al punto clave. Éstos son transformados mediante una representación que permite medir niveles de distorsión y cambios en la iluminación de forma local. [33]

4.5.3 Surf- Speed Up Robust Features

Desarrollado por Herbert Bay, TinneTuytelaars y Luc Van Gool (2008), como un detector y descriptor de puntos de interés para el reconocimiento de objetos. Siendo una de sus ventajas sobre otros métodos, el mejoramiento de la velocidad de cálculo y la robustez ante transformaciones de las imágenes. [34]

Estas mejoras se consiguen mediante la reducción de la dimensionalidad y complejidad en el cálculo de los vectores de características de los puntos de interés obtenidos, mientras continúan siendo suficientemente particulares e igualmente repetitivos.

A continuación, se listan las etapas en las que se divide: [34]



4.5.4 ORB -ORIENTED FAST Y ROTATIVO BRIEF

El descriptor local ORB fue publicado en el ICCV (International Conference on Computer Vision) del 2011 como una alternativa a los descriptores SIFT y SURF, siendo una de sus ventajas sobre estos la velocidad de cómputo. [35]

ORB utiliza el algoritmo Features from Accelerated Segment Test (FAST) para la detección de puntos claves y BinaryRobust Independent Elementary Features (BRIEF) para la extracción de los descriptores.

FAST y BRIEF son algoritmos que mejoran notablemente la velocidad de cómputo, sin embargo, carecen de un operador de orientación e invariancia a la rotación respetivamente. [35]

Para solventar estos dos problemas ORB propone:

oFAST (FAST KeypointOrientation)

- Al algoritmo FAST se le agrega un componente de orientación para los keypoints detectados.

rBRIEF (Rotation-AwareBrief)

- Al algoritmo BRIEF se le agrega invarianza a la rotación y mejora la velocidad de cálculo.

4.5.5 Comparación de los algoritmos de detección de objetos

Para poder comparar los diferentes algoritmos tenemos que tomar en cuenta sus características, tanto en tiempo de ejecución, predicción de imagen, tiempo de proceso, calidad de imagen, y sobre todo tamar en cuenta toso estos parámetros para compararlos con las limitaciones que presentan.

En la ***Tabla 11*** se comparan dichos algoritmos previamente estudiados.

Tabla 11. *Comparación de los algoritmos de detección de objetos*
Fuente: *Investigador*

Algoritmo	Características	Tiempo de Predicción /imagen	Limitaciones
CNN	Divide la imagen en varias regiones Luego clasifica cada región en varias clases.	Alto	Necesita una gran cantidad de regiones para predecir con precisión. Alto tiempo de calculo
SIRF	Detecta objetos incluso con oclusión parcial, no le afecta la orientación, distorsiona y cambios de iluminación	Medio	Perdida de rendimiento, complejo cálculo de vectores de puntos de interés.
SURF	No ocasiona perdida de rendimiento. Mayor robustez	Bajo	Al analizar múltiples objetos, necesita gran cantidad de rendimiento computacional.
ORB	Coste de cálculo bajo, remplaza a los algoritmos Sift y Surf	Muy bajo	Alto consumo computacional.

4.5.6 Posturas humanas

El comportamiento humano puede ser analizado como comunicación no verbal, dos tercios de nuestra comunicación consiste en comunicación no verbal y solo un tercio de nuestra comunicación consiste en contenido verbal.

La comunicación no verbal consiste en expresiones faciales y señales corporales, la postura humana y el movimiento del cuerpo juegan un papel importante en la comunicación no verbal, los humanos usan diferentes gestos con las manos y posturas corporales para expresar su estado emocional interno en diferentes situaciones. En los humanos, las posturas proporcionan información significativa a través de señales no verbales, estudios psicológicos también han demostrado los efectos de la postura corporal sobre las emociones, esta investigación se remonta a los estudios de Charles Darwin sobre la emoción y el movimiento en humanos y animales. [36]

Se ha realizado un estudio e investigación masiva en la década de 1970 sobre la importancia del lenguaje corporal en el que la principal área de enfoque fue el cruce de piernas, postura defensiva y cruce de brazos, lo que sugiere que estos comportamientos no verbales representan sentimientos y actitud, la postura también puede depender de la situación, es decir las personas cambian sus posturas dependiendo de la situación. [36]

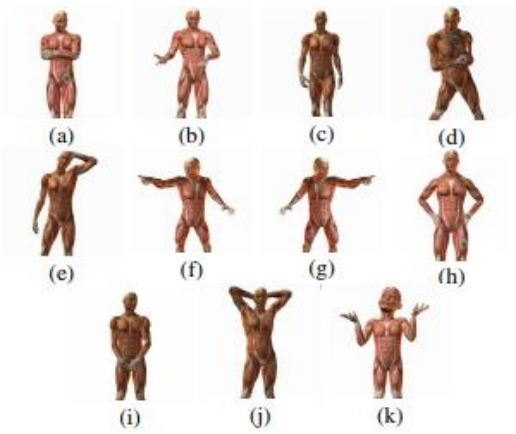


Figura 18. *Posturas humanas [36]*

Reconocimiento de la Postura humana

El reconocimiento de la postura humana es un tema de investigación activo en el campo de la interacción humano-robot. Además de ser utilizado en el contexto de la robótica humanoide, reconocimiento de humanos. [36]

Las posturas tienen muchas aplicaciones en los sistemas asistenciales de los humanos y en la industria, como por ejemplo la industria automovilística que lo que hace es permitir que los robots humanoides trabajen codo con codo con los humanos durante la vida cotidiana. [36]

Para realizar este objetivo, los sistemas robóticos deben tener la capacidad de diferenciar humanos de los entornos desordenados, además de detectar humanos, estos sistemas también deben analizar su postura, acciones, emociones, motivos y comportamiento global, esto a su vez ayuda a los robots a ser más inteligente e ingenioso al interactuar con los humanos.

4.6 Selección del algoritmo de detección de objetos para detección de Posturas Humanas

4.6.1 Criterios y métricas de evaluación

Los criterios y métricas utilizadas en este estudio para el análisis de los descriptores SIFT, SURF y ORB se encuentran dividas en dos grupos: eficiencia y eficacia. En esta Selección de deja de lado el algoritmo CNN por su alto costo de implementación.

4.6.2 Métricas de eficiencia:

Permite medir el uso de recursos de la placa de desarrollo o computador.

Numero de keypoints. Se define la variable K_t la cual representa el número promedio de keypoints extraídos de una imagen de entrada.

Tiempo de ejecución. Se define al tiempo de ejecución como el intervalo de tiempo que tarda el descriptor local en procesar una imagen hasta obtener sus características. Esta dada por la fórmula:

$$T_{total} = T_{detector} + T_{extractor} \quad (4)$$

Dónde:

- $T_{detector}$ = representa el tiempo utilizado para detectar los puntos de interés.
- $T_{extractor}$ = es el tiempo utilizado para la extracción del vector de características.

Consumo energético

Sea B_c el porcentaje de batería consumida por un descriptor local para procesar un conjunto de n imágenes de muestra, se define por la fórmula:

$$B_t \frac{B_c}{n} \quad (5)$$

Dónde:

B_t = es el porcentaje de batería que consume el descriptor local para procesar una imagen.

Pesos de los descriptores

Este aspecto es muy importante para aplicaciones que necesiten almacenar el vector de características generado por el descriptor para uso futuro. El tamaño de este vector varía dependiendo del descriptor utilizado y el número de características generadas. Esta métrica de define por la fórmula:

$P_t = KB\ necesarios\ para\ almacenar\ un\ vector\ de\ caracteristicas.$

4.6.3 Métricas de eficacia:

Permite medir el desempeño de los descriptores. Se consideran los siguientes parámetros:

Precisión vs recall

Este enfoque se basa en el número de aciertos y fallos respecto de las correspondencias establecidas entre dos imágenes. A continuación, se describe sus componentes:

- *correct mathes:* son aquellas imágenes que tras el proceso de comparación han sido identificadas como relacionadas con la imagen original de forma correcta, pues pertenecen a la misma escena.
- *false mathes*: son aquellas imágenes que tras el proceso de comparación han sido identificadas como relacionadas con la imagen original de forma incorrecta, pues no pertenecen a la misma escena.
- *correspondences*: representa el número de imágenes que están relacionadas con la imagen original, es decir, que pertenecen a la misma escena

Precisión (P)

Porcentaje de las imágenes que verdaderamente están relacionadas a la imagen original, se la calcula mediante la fórmula.

$$P = \frac{\#\ correct\ mathes}{\#\ correct\ mathes + \#\ false\ mathes} \quad (6)$$

Recall (R)

El porcentaje de imágenes que fueron identificadas como relacionadas con la imagen original respecto al número total de imágenes analizadas, se la define mediante la fórmula:

$$R = \frac{\#\ correct\ mathes}{\#\ correspondences} \tag{7}$$

Medida-F1 (F1)

Es un valor ponderado entre las dos métricas anteriores y será utilizada para determinar la precisión de los descriptores, se la calcula mediante la fórmula.

$$F1 = \frac{2PR}{P+R} \tag{8}$$

4.6.4 Resultados de los criterios y métricas:

Escenario de prueba

Como se puede observar en la **figura. 19**, para la ejecución de la prueba se desarrolló un aplicativo sobre la plataforma de Matlab que permite obtener las métricas descritas en la sección 4.6.3.

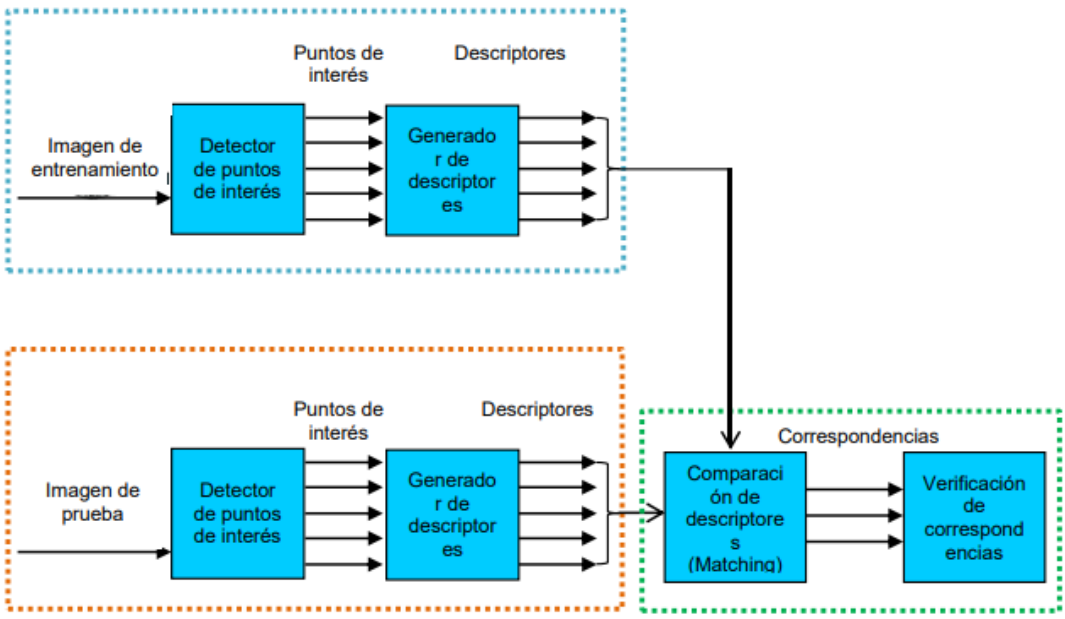


Figura 19. *Componentes del escenario de prueba*
Fuente: *Investigador*

Eficiencia

Numero de keypoints

La figura 20, resume el número promedio de keypoints detectados en una imagen de entrada por parte de los descriptores, observándose que tanto el descriptor ORB y SURF son los que detectan el mayor número de keypoints (432 y 431 respectivamente), en comparación con el descriptor SIFT que detecta 222 puntos claves.

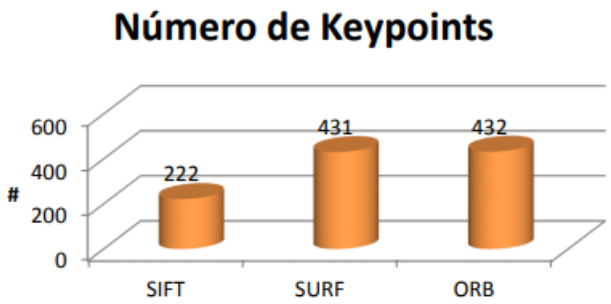


Figura 20. : *Comparación del número de keypoints.*
Fuente: *Investigador*

Tiempo de ejecución

En la figura 21. se observa que el tiempo de computo empleado por el descriptor SIFT (3.0 s) es mayor que el tiempo empleados por el descriptor SURF y ORB con aproximadamente 2.06 s y 2.84 s respectivamente. Esto conduce a determinar que el descriptor SIFT no es una buena opción para aplicaciones que realicen reconocimiento en tiempo real, como sería el caso de reconocimiento de imágenes en una trama de video.

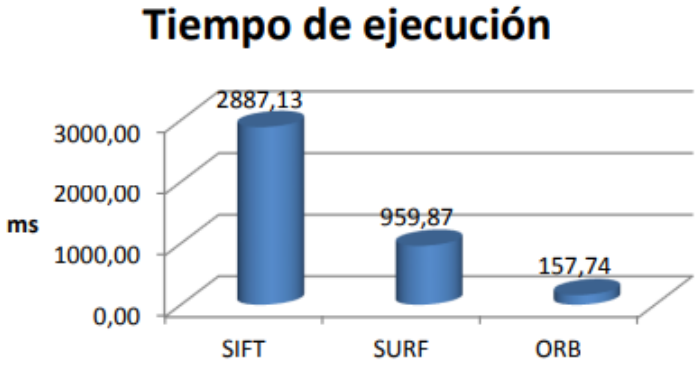


Figura 21. *Comparación de tiempo de ejecución*
Fuente: *Investigador*

Eficacia:

Al calcular el promedio entre los resultados (F1) de los grupos de imágenes analizados, se obtuvo que el descriptor ORB posee una precisión promedio del 41% siendo esta la más alta entre los descriptores evaluados. La Figura 6 muestra el promedio de las precisiones calculadas para cada descriptor.

Se pudo observar que el descriptor SURF tiene mejor desempeño con imágenes que no han sufrido transformaciones y con imágenes con cambios de escala, alcanzando un 65% y 57% respectivamente en cada grupo.

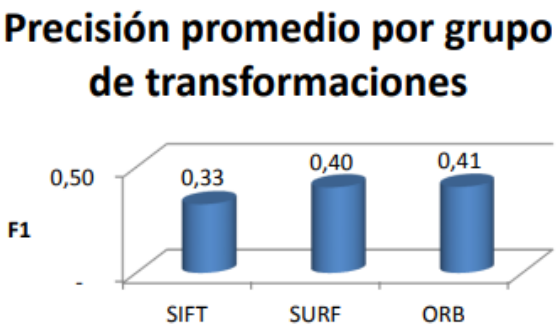


Figura 22. *Comparación entre descriptores por cada grupo de transformaciones.*
Fuente: *Investigador*

El algoritmo Elegido es Surf por la velocidad que representa y las facilidades de su aplicación.

La solución presentada inicia con la captura de la imagen del cuerpo por medio de una cámara web, este algoritmo no toma en cuenta la iluminación y el fondo las cuales son muy importantes para el resultado final.

Implementado el algoritmo se puede observar en la ***Figura 23*** el diagrama utilizado en el tema de investigación

4.7 Desarrollo e implementación del algoritmo seleccionado.

4.7.1 Configuración del sistema

Antes de que realmente entremos en la fase de construcción del modelo, debemos observar el sistema completo para iniciar con las configuraciones.

En la ***Figura 23*** se puede observar un breve diseño del sistema completo se muestra el sistema de visión artificial montado sobre la tarjeta Nvidia Jetson Nano, trabajando con el ROS-MELODIC debido a que el sistema de Jetson nano trabaja con la distribución de Linux-UBUNTU 18.04, la cámara que va recoger los datos de la persona son procesados mediante OpenCV, la comunicación con el robot autónomo Kuka es mediante el protocolo de comunicación 802.11 aplicando los nodos de ROS, tomando ROS–HYDRO instalado en el robot autónomo como nodo maestro.

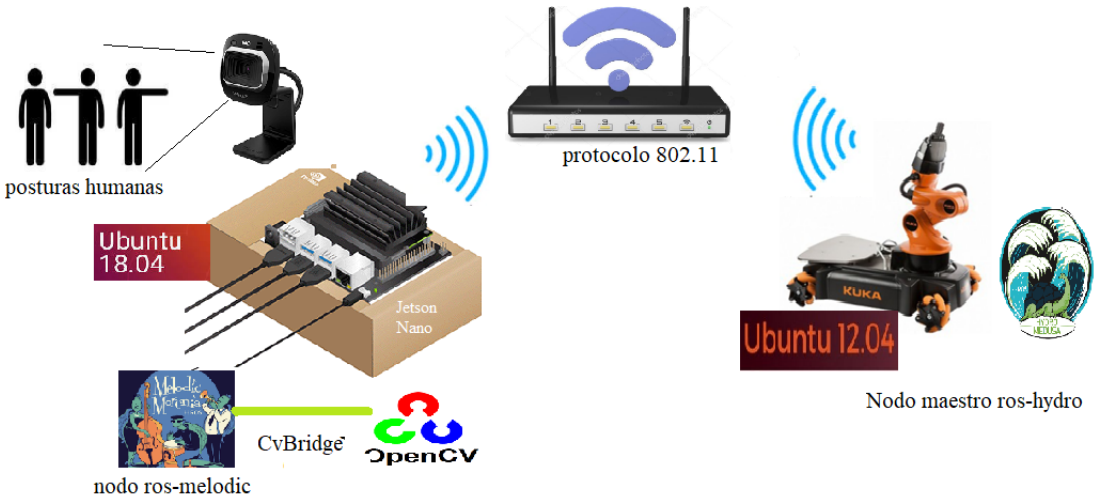


Figura 23. *Diagrama básico de solución de investigación planteada.*
Fuente: *Investigador*

Ahora el siguiente paso es asegurarnos de elegir todos los requisitos previos, adecuando los entornos para ejecutar este proyecto de investigación:

- Placa de desarrollo (Nvidia Jetson Nano)
- Ros
- Opencv
- Lenguaje de programación (Python)

4.7.2 Elección Placa de desarrollo

Después de haber revisado las diferentes placas de desarrollo buscando todas las características y el creador mostrados en la ***Tabla 8*** se ha decidido utilizar la Nvidia Jetson Nano.

4.7.3 Nvidia Jetson Nano

Es un pequeño ordenador, de gran alcance que le permite ejecutar múltiples redes neuronales en paralelo para aplicaciones como la clasificación de imágenes, detección de objetos, segmentación y procesamiento del habla. Todo en una plataforma fácil de usar que funciona en tan solo 5 vatios.

NVIDIA Jetson Nano es un sistema integrado en el módulo (SoM) y un kit de desarrollo de la familia NVIDIA Jetson, que incluye una GPU Maxwell de 128 núcleos integrada, una CPU ARM A57 de 64 bits, memoria LPDDR4 de 4GB, junto con soporte para E / S de alta velocidad MIPI CSI-2 y PCIe Gen2 se la puede observar en la ***Figura 24.***

Figura 24. *Placa física Nvidia Jetson Nano.*

Jetson Nano, que es útil para implementar la visión por computadora y el aprendizaje profundo, ejecuta Linux y proporciona 472 GFLOPS de rendimiento de cómputo FP16 con 5-10W de consumo de energía.

Tabla 12. *Características de la placa física Nvidia Jetson Nano*
Fuente: *Investigador*

ESPECIFICACIONES TÉCNICAS	
GPU	Maxwell de 128 núcleos
UPC	BRAZO DE CUATRO CORAZONES A57 @ 1.43 GHz
Memoria	4 GB de 64 bits LPDDR4 25.6 GB / s
Almacenamiento	microSD (no incluida)
Codificador de video	4K @ 30 | 4x 1080p @ 30 | 9x 720p @ 30 (H.264 / H.265)

Decodificador de video	4K @ 60 | 2x 4K @ 30 |8x 1080p @ 30 | 18x 720p @ 30 | (H.264 / H.265)
Cámara	1x carriles MIPI CSI-2 DPHY
Conectividad	Gigabit Ethernet, M.2 Key E
Monitor	HDMI 2.0 y eDP 1.4
Usb	4x USB 3.0, USB 2.0 Micro-B
medidas	100 mm x 80 mm x 29 mm

Configurando el NVIDIA Jetson Nano

Antes de poder arrancar la NVIDIA Jetson Nano, necesitamos tres cosas que no vienen incluido en el kit de desarrollo:

- Una tarjeta micro-SD en este proyecto se utilizó una tarjeta de 64gb
- Una fuente de alimentación de 5V 3A MicroUSB
- Un cable ethernet o adaptador inalámbrico USB.

Antes de que podamos comenzar a instalar cualquier paquete o ejecutar demostraciones en el Jetson Nano, primero debemos descargar la imagen de la tarjeta SD del kit de desarrollador de Jetson Nano del sitio web de NVIDIA.

NVIDIA proporciona documentación para actualizar el archivo .img en una tarjeta micro-SD para Windows, macOS y Linux; debe elegir las instrucciones flash adecuadas para su sistema operativo en particular.

Primer arranque

Una vez que haya descargado y flasheado el archivo .img en su tarjeta micro-SD, inserte la tarjeta en la ranura para tarjeta micro-SD ver ***Figura 25.***

Figura 25. *Puerto de la tarjeta SD, Jetson Nano*
Fuente: *Investigador*

Después de deslizar el hogar de la tarjeta micro-SD, conectar su fuente de alimentación y arranque como se ve en la ***Figura 26.***

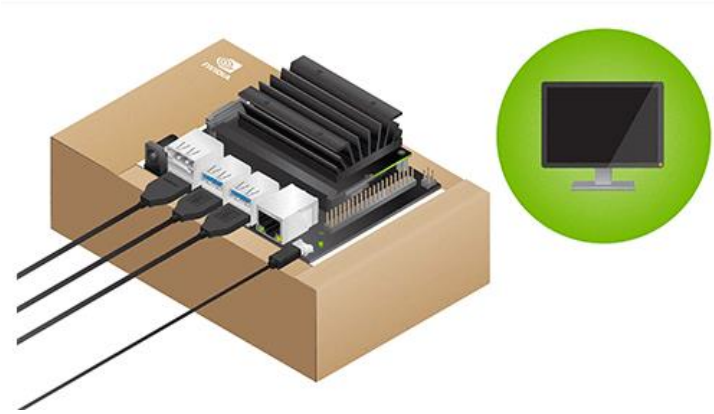


Figura 26. *Conexiones, Jetson Nano*
Fuente: *Investigador*

El Jetson Nano guía a través del proceso de instalación, incluyendo la configuración de su nombre de usuario / contraseña, zona horaria, diseño del teclado, etc. En la ***Figura 27*** se puede observar el primer arranque.

Figura 27. *Primer arranque, Jetson Nano*
Fuente: *Investigador*

4.7.4 Sistema Operativo Robótico en Jetson Nano

El Jetson Nano está diseñada para máquinas autónomas, es una plataforma pequeña, de baja potencia y asequible con un alto nivel de potencia de cómputo que permite realizar visión de computadora en tiempo real y operaciones de aprendizaje profundo a nivel móvil en el límite.

ROS es la elección natural cuando se construye un robot autónomo multisensorial.

Después de configurar el Jetson Nano con su imagen JetPack, instalaremos la última versión de ROS que se ejecuta en Ubuntu 18 Bionic Beaver: Melodic Morenia ver ***Figura 28.***

Figura 28. *Logo ROS-MELODIC*
Fuente: *Investigador*

Instalación

1. Abra un nuevo terminal presionando Ctrl + Alt + t, usando el sistema de lanzamiento de Ubuntu 18.

2. Configure el Jetson Nano para aceptar el software de packages.ros.org :

```
$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $
(lsb_release -sc) main"> /etc/apt/sources.list.d/ros-latest.list'
```

3. Añadir una nueva clave de apt:

```
$ sudo apt-key adv --keyserver 'hkp: //keyserver.ubuntu.com: 80'
--recv-key
```

4. Actualizar el índice de paquetes de Debian:

```
$ sudo apt-get update
```

5. Se instala el paquete de ROS en el escritorio, incluyendo el apoyo a rqt, rvizy otros paquetes de robótica útiles:

```
$ sudo apt install ros-melodic-desktop
```

Importante: "ROS Desktop Full" es un paquete más completo, sin embargo, no se recomienda para una plataforma integrada; Los simuladores 2D / 3D se instalarán con él y ocupan demasiado espacio en la ROM, y tienen demasiada hambre computacional para ser usados en el Nano.

6. Inicializar rosdep, rosdep le permite instalar fácilmente las dependencias del sistema para el código fuente que desea compilar y se requiere que ejecute algunos componentes centrales en ROS:

```
$ sudo rosdep init

$ rosdep update
```

7. Cargar las variables de entorno de ROS automáticamente cuando ejecuta una nueva sesión de shell. Actualice su .bashrc script:

```
$ echo "source /opt/ros/melodic/setup.bash" >> ~ / .bashrc
$ source ~ / .bashrc
```

8. Crear un espacio de trabajo ROS

```
$ mkdir -p ~ / catkin_ws / src

$ cd ~ / catkin_ws /

$ catkin_make
```

Ahora, el Jetson Nano está listo para ejecutar paquetes ROS y convertirse en el cerebro de la navegación autónoma basada en visión artificial.

Conceptos importantes de Ros

Los nodos en un entorno ROS pueden comunicarse de tres formas como: Publicando mensajes a un tema, escuchando los mensajes publicados sobre un tema y llamando a un servicio proporcionado por otro nodo[37].

Los mensajes representan estructuras que pueden ser transmitidas entre nodos, y la información que estos contienen puede ser de cualquier tipo de dato primitivo, por ejemplo: enteros, booleanos, números de coma flotante o tipo texto.

Los topics o temas de ROS representan canales con nombre sobre los cuales un tipo particular de mensaje puede ser transferido, el topic provee comunicación unidireccional entre cualquier número de publicaciones y nodos consumidores.

Un nodo que publica un topic o tema es llamado publicador o editor, estos corren continuamente para proporcionar la lectura de sensores u otra información periódica a otros nodos, sin embargo, es posible que un editor publique un mensaje una sola vez. Un nodo que escucha los mensajes de un tema es llamado suscriptor, el cual especifica que tema es el que quiere escuchar así también como el tipo de mensaje para dicho tema, además registra una función de devolución de llamada (callback) que se ejecuta siempre y cuando se reciba un mensaje[38].

Finalmente un nodo puede proporcionar un servicio a otro nodo, donde una llamada de servicio en ROS se define como un flujo de trabajo remoto, un nodo cliente envía una solicitud al nodo del proveedor de servicios, mientras que una función de devolución de llamada registrada en el nodo proveedor realiza la acción apropiada y el proveedor de servicio envía una respuesta de regreso al cliente, dado que una llamada de servicio es un intercambio entre un nodo cliente y un proveedor de servicios no se emplean temas, sin embargo los mensajes de petición de servicio y de respuesta se definen de manera similar a los mensajes estándar. Se obtiene un gran flexibilidad en las acciones ya que un nodo puede realizar cualquier combinación de las acciones de comunicación[37].

En la **figura 29**, se puede observar como el maestro ROS permite el intercambio de información dentro del protocolo publicación/suscriptor.

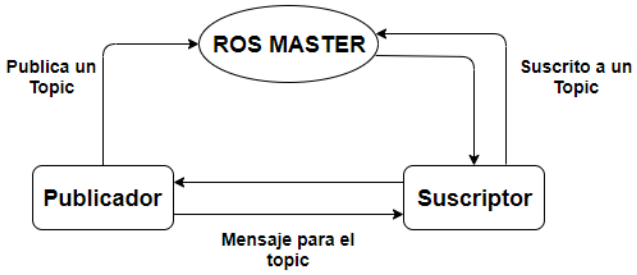


Figura 29. *Arquitectura ROS*
Fuente: *Investigador*

Creación del Nodo subscriptor

1. Primero nos dirigimos al directorio del espacio de origen del espacio de trabajo de catkin

```
$ cd ~ / catkin_ws / src
```

2. Usamos el script catkin_create_pkg para crear un nuevo paquete llamado 'beginner_tutorials' que depende de std_msgs, roscpp y rospy:

```
$ catkin_create_pkg beginner_tutorials std_msgs rospy
```

3. Construir los paquetes en el espacio de trabajo de catkin

```
$ cd ~ / catkin_ws
$ catkin_make
```

4. Para agregar el área de trabajo a su entorno ROS, necesita obtener el archivo de configuración generado:

```
$ . ~/catkin_ws/devel/setup.bash
```

Una vez realizado esto el archivo **package.xml** que se creó dentro del paquete se debería ver así.

```
<?xml version="1.0"?>
<package format="2">
  <name>beginner_tutorials</name>
  <version>0.1.0</version>
  <description>The beginner_tutorials package</description>

  <maintainer email="you@yourdomain.tld">Your Name</maintainer>
  <license>BSD</license>
  <url type="website">http://wiki.ros.org/beginner_tutorials</url>
  <author email="you@yourdomain.tld">Jane Doe</author>

  <buildtool_depend>catkin</buildtool_depend>

  <build_depend>roscpp</build_depend>
  <build_depend>rospy</build_depend>
  <build_depend>std_msgs</build_depend>

  <exec_depend>roscpp</exec_depend>
  <exec_depend>rospy</exec_depend>
  <exec_depend>std_msgs</exec_depend>

</package>
```

5. Ahora vamos a crear los archivos **msg** que son los los archivos que describen los campos de un mensaje en ros, y el archivo **srv** que es el archivo que describe un servicio. Tiene solicitud y respuesta

```
$ roscd beginner_tutorials
$ mkdir msg
$ echo "int64 num"> msg / Num.msg
```

6. Abrir packege.xml y agregar estas líneas de código, asegurarse que no estén comentadas. En este paso debemos estar seguros que estos archivos puedan ser comprendidos para códigos como pyton y C++

```
<build_depend> message_generation </build_depend>
<exec_depend> message_runtime </exec_depend>
```

message_generation= se utiliza para la creación

message_runtime = se usa en tiempo de ejecución

7. Abrimos con el editor nano el archivo **CMakeLists.txt** y agregamos las siguientes dependencias

```
find_package(catkin REQUIRED COMPONENTS
roscpp
rospy
std_msgs
message_generation
)
```

8. También nos aeguramos que estas líneas de código queden como se ven en los cuadros.

```
catkin_package(
...
CATKIN_DEPENDS message_runtime ...
...)
```

```
add_message_files(
FILES
Num.msg
)
```

```
generate_messages(
  DEPENDENCIES
  std_msgs
)
```

```
add_service_files (
  ARCHIVOS
  AddTwoInts.srv
)
```

9. Entramos a nuestro paquete y creamos el archivo srv

```
$ roscd beginner_tutorials
$ mkdir srv
```

10. Copiamos la carpeta que es propiedad de rospy_tutorials para poder tener la estructura de svr

```
$  roscp rospy_tutorials AddTwoInts.srv srv / AddTwoInts.srv
```

11. Ahora que creamos mensajes debemos compilar el archivo catkin

```
$ roscd beginner_tutorials
$ cd ../ ..
$ catkin_make install
$ cd -
```

12. En este paso ingresamos a nuestro paquete y creamos una carpeta llamada scripts, aquí va hacer donde vamos a guardar nuestro código escrito en Python

```
$ roscd beginner_tutorials
$ mkdir scripts
$ cd scripts
```

13. Creamos nuestro archivo que se va a llamar **talker.py** y lo hacemos ejecutable

```
$ mkdir talker.py
$ chmod +x talker.py
```

14. Ingresamos con nano al archivo **talker.py** y podemos ver el archivo que es el que va a enviar mensajes hacia el nodo subscriptor, el código se puede ver así:

```
#!/usr/bin/env python
# license removed for brevity
import rospy
from std_msgs.msg import String

def talker():
    pub = rospy.Publisher('chatter', String, queue_size=10)
    rospy.init_node('talker', anonymous=True)
    rate = rospy.Rate(10) # 10hz
    while not rospy.is_shutdown():
        hello_str = "hello world %s" % rospy.get_time()
        rospy.loginfo(hello_str)
        pub.publish(hello_str)
        rate.sleep()

if __name__ == '__main__':
    try:
        talker()
    except rospy.ROSInterruptException:
        pass
```

15. En el nodo subscritor creado en el Robot Kuka YouBot se realiza exactamente los mismos pasos vistos hasta aquí a excepción de ahora crear en el archivo

```
#!/usr/bin/env python
import rospy
from std_msgs.msg import String

def callback(data):
    rospy.loginfo(rospy.get_caller_id() + "I heard %s", data.data)

def listener():

    # In ROS, nodes are uniquely named. If two nodes with the same
    # name are launched, the previous one is kicked off. The
    # anonymous=True flag means that rospy will choose a unique
    # name for our 'listener' node so that multiple listeners can
    # run simultaneously.
    rospy.init_node('listener', anonymous=True)

    rospy.Subscriber("chatter", String, callback)

    # spin() simply keeps python from exiting until this node is stopped
    rospy.spin()

if __name__ == '__main__':
    listener()
```

listener.py que va hacer el que reciba los datos y pueda realizar las acciones nuestro robot Kuka. El código de listener.py se puede observar a continuación:

16. Ahora el siguiente paso va hacer configurar la comunicación entre ROS-MELODIC que corre en la Jetson Nano y ROS-HYDRO que corre en el robo Kuka YouBot. Esto es posible ya que los nodos de ROS pueden comunicarse aun si están en plataformas diferentes, un nodo bien direccionado puede tener muchas aplicaciones.

    El siguiente comando ejecuta todo el servicio de ROS y cabe destacar que en una red de nodos de ROS solo hace falta que este levantado un nodo maestro en todo el sistema.

    ```
    ssh youbot
    roscore
    ```

17. Los siguientes comandos se ejecutan en el nodo maestro que en este caso se ha configurado el Kuka YouBot como nodo maestro mediante el servicio ssh.

    ```
    ssh youbot
    export ROS_MASTER_URI=http://youbot:11311
    rosrun beginner_tutorials listener.py
    ```

18. Ahora en el nodo subscritor

    ```
    ssh santi
    export ROS_MASTER_URI=http://youbot:11311
    rosrun beginner_tutorials talker.py
    ```

19. De esta forma es como se puede ver la comunicación de los datos entre los nodos publicador y subscriptor

    En el nodo suscriptor

```
[INFO] [WallTime: 1314931831.774057] hello world 1314931831.77
[INFO] [WallTime: 1314931832.775497] hello world 1314931832.77
[INFO] [WallTime: 1314931833.778937] hello world 1314931833.78
[INFO] [WallTime: 1314931834.782059] hello world 1314931834.78
[INFO] [WallTime: 1314931835.784853] hello world 1314931835.78
[INFO] [WallTime: 1314931836.788106] hello world 1314931836.79
```

En el nodo publicador (maestro)

```
[INFO] [WallTime: 1314931969.258941] /listener_17657_13149319687951 heard hello world 13149
31969.26
[INFO] [WallTime: 1314931970.262246] /listener_17657_13149319687951 heard hello world 13149
31970.26
[INFO] [WallTime: 1314931971.266348] /listener_17657_13149319687951 heard hello world 13149
31971.26
[INFO] [WallTime: 1314931972.270429] /listener_17657_13149319687951 heard hello world 13149
31972.27
[INFO] [WallTime: 1314931973.274382] /listener_17657_13149319687951 heard hello world 13149
31973.27
[INFO] [WallTime: 1314931974.277694] /listener_17657_13149319687951 heard hello world 13149
31974.28
[INFO] [WallTime: 1314931975.283708] /listener_17657_13149319687951 heard hello world 13149
31975.28
```

4.7.5 Nodo Publicador Corriendo en Kuka YouBot

Para integrar este dispositivo en ROS se ha desarrollado una librería C++ para comunicación serie mediante términos disponible en los apéndices.

La integración con ROS de la unidad pan/tilt consistirá en la comunicación bidireccional con el dispositivo mediante topics utilizando mensajes de tipo JointState. que contienen los siguientes campos:

```
Header header
string[ ] name
float64[ ] position
float64[ ] velocity
float64[ ] effort
```

4.7.6 OpenCV

OpenCV (Open Source Computer Vision) es una biblioteca de funciones de programación orientada principalmente a la visión de computadora en tiempo real. En lenguaje simple es una biblioteca utilizada para el procesamiento de imágenes. Se utiliza principalmente para realizar todas las operaciones relacionadas con las imágenes. [39]

Tabla 13. *Características de OpenCV*

Fuente: *Investigador*

Aplicaciones
Leer y escribir imágenes.
Detección de rostros y sus características.
Detección de formas como Círculo, rectángulo, etc. en una imagen. Ej. Detección de moneda en imágenes.
Reconocimiento de texto en imágenes. por ejemplo, lectura de placas de números
Modificar la calidad de la imagen y los colores, por ejemplo, Instagram, CamScanner.
Desarrollar aplicaciones de realidad aumentada.
y muchos más.....

Tabla 14. *Idiomas que soporta OpenCV*
Fuente: *Investigador*

Idiomas que soporta	
C ++	Java
Android SDK	Python

4.7.7 Interacción entre OpenCV y ROS-MELODIC

ROS proporciona una muy simple integración con OpenCV, el open más utilizado es la fuente de la biblioteca de visión artificial. Sin embargo, no incluye una cinética específica de Debian para ROS porque eso obligaría a los usuarios a usar una única versión. En lugar de eso, le permite usar la última biblioteca de OpenCV en el sistema y proporciona herramientas de integración adicionales para usar OpenCV en ROS.

Figura 30. *ROS y OpenCV*
Fuente: *Investigador*

Integración OpenCV y ROS

Para poder obtener trabajar con ROS obteniendo las facilidades de OpenCV se necesita intslar la pila vision_opencv, que proporciona un paquete de la popular biblioteca OpenCV para ROS.

OpenCV vision_opencv proporciona varios paquetes:

- **cv_bridge :** Puente entre mensajes ROS y OpenCV.
- **image_geometry :** colección de métodos para tratar la imagen y la geometría de píxeles.

ROS transmite imágenes en su propio formato sensor_msgs / Mensaje de imagen , pero muchos usuarios querrán usar imágenes junto con OpenCV. CvBridge es una biblioteca de ROS que proporciona una interfaz entre ROS y OpenCV. CvBridge se puede encontrar en el paquete cv_bridge en la pila vision_opencv .

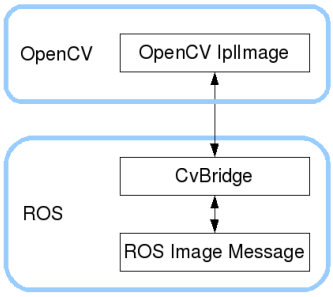


Figura 31. *Interacción ROS y OpenCV*
Fuente: *Investigador*

Instalación

Esta instrucción instala un paquete OpenCV en ROS.

```
$ sudo apt-get install ros-melodic-opencv3
```

Esta instrucción crea el puente entre OpenCV y ros

```
$ sudo apt-get install ros-melodic-vision-opencv
```

4.7.8 Algoritmo Surf en OpenCV

"SURF: Características robustas aceleradas", como su nombre lo indica, es una versión acelerada de SIFT.

OpenCV proporciona funcionalidades SURF al igual que SIFT. Usted inicia un objeto SURF con algunas condiciones opcionales como descriptores de 64/128-dim, vertical / normal SURF, etc. Todos los detalles están bien explicados en los documentos. Luego, como hicimos en SIFT, podemos usar SURF.detect (), SURF.compute (), etc. para encontrar puntos clave y descriptores.

Primero veremos una demostración simple sobre cómo encontrar puntos clave y descriptores SURF y dibujarlo. Todos los ejemplos se muestran en la terminal de Python ya que es igual que SIFT solamente.

```
>>> img  =  cv2 . imread ( 'fly.png' , 0 )

# Crear objeto SURF. Puede especificar parámetros aquí o más adelante.
# Aquí configuro el umbral de Hesse en 400
>>> surf  =  cv2 . SURF ( 400 )

# Encuentra puntos clave y descriptores directamente
>>> kp ,  des  =  surf . detectAndCompute ( img , Ninguno )

>>> len ( kp )
699
```

1199 puntos clave es demasiado para mostrar en una imagen. Lo reducimos a unos 50 para dibujarlo en una imagen. Si bien coinciden, es posible que necesitemos todas esas características, pero no ahora. Entonces aumentamos el umbral de Hesse.

```
# Comprobar el umbral actual de Hesse
>>> print surf.hessianThreshold
400,0

# Lo configuramos en unos 50000. Recuerde, es solo para representar en la imagen.
# En casos reales, es mejor tener un valor de 300-500
>>> surf.hessianThreshold = 50000

# Nuevamente calcule los puntos clave y verifique su número.
>>> kp, des = surf.detectAndCompute (img, Ninguno)

>>> imprimir len (kp)
47
```

Es menos de 50. Dibujémoslo en la imagen.

```
>>> img2  =  cv2 . drawKeypoints ( img , kp , Ninguno , ( 255 , 0 , 0 ), 4 )
>>> plt . imshow ( img2 ), plt . show ()
```

Ver el resultado a continuación. Puedes ver que SURF es más como un detector de gotas. Detecta las manchas blancas en alas de mariposa. Puedes probarlo con otras imágenes.

Figura 32. *Puntos Surf, imagen de ejemplo.*

4.7.9 Interacción Surf OpenCV y ROS-Melodic

Ros cuenta con un paquete en donde permite usar una interfaz Qt simple para probar implementaciones OpenCV de SIFT, SURF, FAST, BRIEF y otros detectores y descriptores de características. Usando una cámara web, los objetos se pueden detectar y publicar en un tema ROS con ID y posición (píxeles en la imagen). Este paquete es una integración ROS de la aplicación Find-Object .

Instalación

En la wiki de ROS se puede encontrar toda la información del paquete, puede ver si el sistema de compilación es catkin o rosbuild, quién lo creó y qué tipo de licencia de código abierto es. En la wiki, puede consultar la lista de paquetes dependientes haciendo clic en el enlace "Dependencias" en el menú del artículo, el enlace al sitio web externo del proyecto, la dirección del repositorio del paquete e instrucciones sobre cómo usa el paquete. Especialmente, asegúrese de verificar las dependencias del paquete.

Figura 33. *Wiki de ros*

Para instalar el paquete

```
$ sudo apt-get install ros-melodic-find-object-2d
```

Después de esto copiamos toda la carpeta de ejemplo.

```
$ cd ~/catkin_ws/src

$ git clone https://github.com/introlab/find-object.git

$ cd ~/catkin_ws/

$ catkin_make
```

Los siguientes paquetes no están directamente relacionados con ROS, pero las bibliotecas OpenCV y Qt son se usa en el paquete "find_object_2d", por lo que debe instalarlos.

```
sudo apt-get install libopencv-dev // Install OpenCV $
sudo apt-get install libqt4-dev // Install Qt
```

4.8 Sistema Completo

Una vez teniendo todos los componentes previos, como son la interacción entre OpenCV-ROS-Kuka YouBot, el sistema toma los datos de la cámara manejada en OpenCV y los procesa enviando datos mediante cv_bridge que es el puente de interacción con ROS este crea un nodo Subscriptor cargado en la Nvidia Jetson Nano y este se conecta a KUKA YouBot que crea el nado Publicador y este envía los datos de velocidad y posición hacia la API de Kuka.

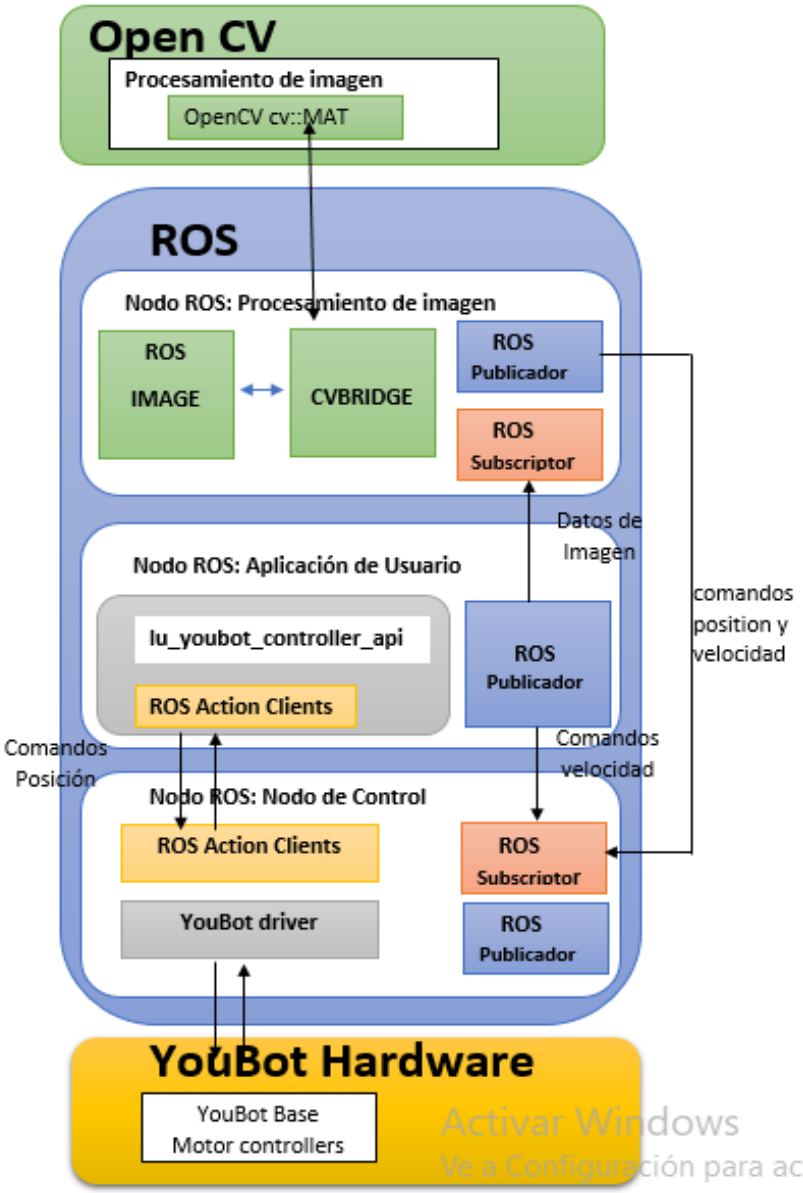


Figura 34. *Interacción de todos los sistemas*
Fuente: *Investigador*

Nodo ROS: el procesamiento de imágenes convierte las imágenes de ROS a formato OpenCV o viceversa a través de CvBridge, una biblioteca que permite enviar o recibir imágenes con el procesamiento de imágenes OpenCV. Además, este nodo obtiene imágenes con los suscriptores de los editores establecidos en la aplicación ROS Nod: User y envía diferentes comandos con su editor al suscriptor en el nodo ROS: controller_node.

Aplicación de Usuario: Ejecuta la comunicación entre el Cliente y el Servidor a través del Protocolo de Acción de ROS, que se construye sobre los mensajes de ROS. Luego, el cliente y el servidor proporcionan una API simple (interfaz de programa de aplicación, que es un conjunto de rutinas, protocolos y herramientas para crear aplicaciones de software) para que los usuarios soliciten objetivos (del lado del cliente) o ejecuten objetivos (del lado del servidor) a través de llamadas de función y devoluciones de llamada. La comunicación de User_application y los nodos del controlador proporciona al nodo del controlador los comandos lógicos para ser interpretados como acciones físicas. Los Clientes de Acción de ROS envían la información de posición y trayectoria procesada con la API y otras herramientas y protocolos al Servidor de Acción del nodo controlador. Mientras que, el editor ROS del nodo User_application envía los comandos como velocidad.

Nodo de Control: Transforma los comandos en medidas o señales que pueden ser comprendidas por los actuadores del robot.

YouBot Hardware: Es el espacio donde el sistema de robot se representa como una combinación de subsistemas funcionales desacoplados. El brazo manipulador y la plataforma base están dispuestos como la combinación de varias articulaciones. Al mismo tiempo, cada junta se define como una combinación de un motor y una caja de cambios. La comunicación con el hardware y el controlador se realiza mediante la conexión Serial Ethercat.

4.8.1 Arrancando el sistema

Antes de iniciar todo el sistema verificamos la dirección Ip, para saber si están conectados en la red, tanto la Jetson nano como el robot Kuka, para poder configurar el nodo maestro. Esto se puede verificar con el comando.

```
$ ifconfig
```

Ahora configuramos para que el robot Kuka sea el nodo maestro.

```
$ export ROS_HOSTNAME=ubuntu
$ export ROS_MASTER_URI=http://ubuntu:11311
```

El siguiente paso es configurar la Jetson nano como nodo publicador, la primera dirección ip es la que tienen nano y la segunda es la dirección ip que tiene Kuka.

```
$ export ROS_IP=192.168.137.219
$ export ROS_MASTER_URI=http://192.168.137.202:11311
```

Este paso es el más importante y va hacer que todo nuestro sistema esté controlado, arrancamos el nodo maestro

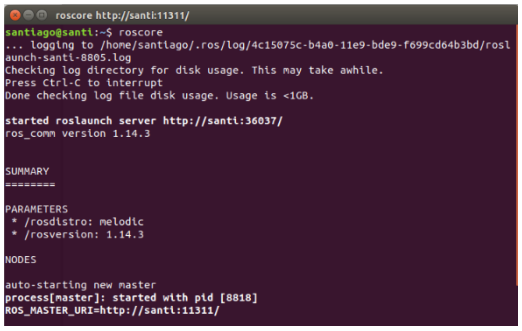


Figura 35. *Arranque del nodo maestro*
Fuente: *Investigado*

Arrancamos el nodo que controla la cámara web.

```
$ rosrun usb_cam usb_cam_node
```

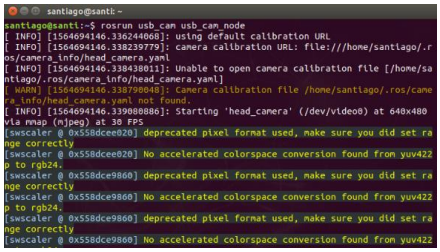


Figura 36. *Nodo usb_cam*

Fuente: *Investigado*

Se puede verificar visualmente los nodos en funcionamiento utilizando el comando rqt_graf.

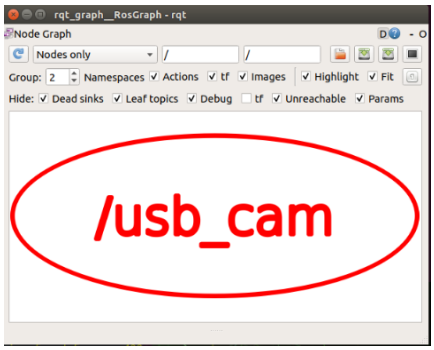


Figura 37. *Nodo usb_cam desde rqt_graf*

Fuente: *Investigado*

Ahora vamos a levantar el nodo de deteccion de posturas usando el algoritmo SURF.

```
$ rosrun find_object_2d find_object_2d image:=/usb_cam/image_raw
```

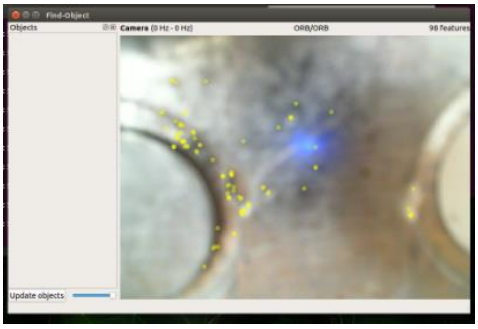


Figura 38. *Interfaz qt de deteccion de posturas*

Fuente: *Investigado*

Montamos las imágenes previamente seleccionadas y con formato jpg.

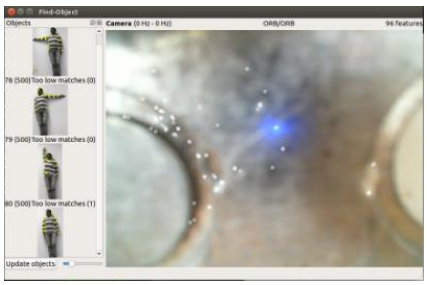


Figura 39. *Interfaz qt con las posturas, detectando los puntos SURF*

Fuente: *Investigado*

Verificamos que nuestros nodos se estén comunicando exitosamente, con rqt_graf, se puede observar que el nodo usb_cam se comunica con el nodo find_object_2d por medio del topic usb_cam/image_raw que es quien le da la imagen desde la web cam.

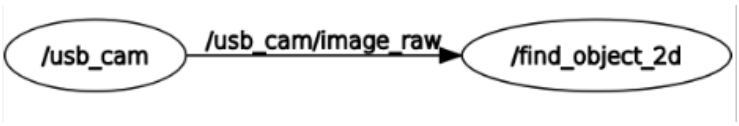


Figura 40. *Nodos entre conectados*

Fuente: *Investigado*

Iniciamos el nodo en el robot Kuka y se puede observar como tenemos conexión con el nodo maestro que es el encargado de enviar los parámetros de movimiento al Kuka.

```
$ rosrun tutorials listen
```

Probamos el reconocimiento de las posturas realizando las mismas frente a la cámara.

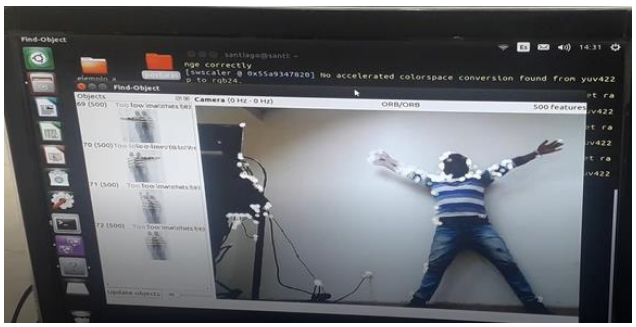


Figura 41. *Prueba de posturas*

Fuente: *Investigado*

Por ultimo verificamos la detección en el nodo del Kuka YouBot.

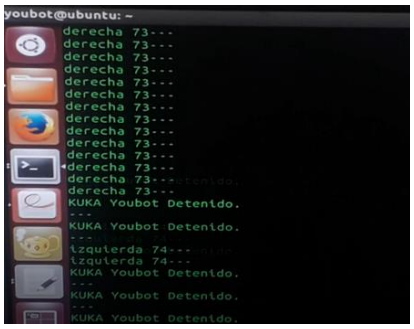


Figura 42. *Toma de datos en el nodo maestro.*

Fuente: *Investigado*

4.9 Resultados

Para el reconocimiento de postura Humana OpenCV crea un espacio un espacio de descriptores 2D, en donde se obtiene puntos de interés, eligiendo las posturas mediante algoritmo Surf, dando como respuesta valores de las coordenadas de las posturas detectadas, las mismas que representan en un Topic. Estos valores son del cuerpo ubicados en diferentes formas utilizado los brazos ubicados de forma lateral y sólo realizando movimientos que no se desprenden del plano de la imagen. El análisis total del cuero tanto la imagen cargada, los puntos de interes y las acciones del robot Kuka se detalla en la Tabla 15.

La navegación del robot autónomo Kuka se la realiza con la conexión por adaptador wifi y solo está conectado a la alimentación eléctrica debido a que las baterías se encuentran averiadas.

Tabla 15. *Reconocimiento de imagen Surf y acciones Kuka.*
Fuente: *Investigado*

Postura	Puntos de interés generados	Acciones robot Kuka
		
		
		
		

4.10 Recursos económicos

Los materiales utilizados en el proyecto de investigación fueron adquiridos adecuadamente, el Robot Kuka por su alto precio fue prestado por la facultad, de igual forma lo cámara LifeCam HD.

A continuación, en la **Tabla 16,** se detalla todos los elementos utilizados en el desarrollo del presente proyecto con su respectivo costo incluido el IVA

Tabla 16. *Recursos Económicos*
Fuente: *Investigado*

ITEM	Descripción	Unidad	Cantidad	Precio Unitario	Precio Total
1	Nvidia Jetson Nano	c/u	1	$170.00	$170.00
2	Tarjeta micro SD 64 Gb	c/u	1	$20.00	$20.00
3	Teclado	c/u	1	$12.00	$12.00
4	Mouse	c/u	1	$8.00	$8.00
5	**Memory Flash 8Gb**	c/u	1	$14.00	$14.00
6	**Cargador 3 A**	c/u	1	$12.00	$12.00
7	**Imprevistos (10%)**	—	—	—	$23.60
	Total				S 259.60

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

- El trabajo investigativo analiza el estado actual y el campo de aplicación de la detección y el seguimiento de objetos usados en visión artificial que es el campo de más aplicación de la inteligencia artificial por sus múltiples aplicaciones como son el reconocimiento facial, recuerdos de personas, control de calidad industrial, autos de auto conducción, seguridad y muchos más. Estos procesos de realizan con interacciones con el mundo real en donde el primer paso es detectar un objeto especifico de interés de una escena determinada y dar el seguimiento a este objeto, estos procesos se pueden realizar en tiempo real o a través de un video.
- El Robot Autónomo Kuka YouBot tienen características de contar con máxima maniobrabilidad en superficies planas y tiene la gran ventaja comparado con otros robots de para girar no necesita reorientación, esta clase de robot cuenta con 4 ruedas tipo Mecanum que lleva a que tenga un control más complejo, esta clase de ruedas le ofrece mejor estabilidad y tracción al robot. El modelo cinemático de YouBot consta de dos partes el Modelo de la Plataforma Omnidireccional que es basado en el modelo de Nagatani desarrollado en el 2000 y el Modelo del manipulador de 5 grados de libertad que está basado en el algoritmo de Denavit-Hartenber desarrollado en el 2005, este algoritmo da un procedimiento sistemático para la determinación de la posición y orientación del efecto final de un manipulador robótico de n-articulaciones.

- Los algoritmos de detección de objetos encajan en muchísimas aplicaciones en la industria, dentro de los algoritmos más utilizados en detección de objetos tenemos CNN, SIRF, SURF, ORB, los cuales tienen diferentes características, tiempos de predicción de imagen y limitaciones. Para poder elegir entre uno u otro algoritmo es necesario saber cuál va hacer su aplicación entre los algoritmos mencionados existes grandes diferencias.
- La Nvidia Jetson Nano es un ordenador fabricado de manera optimizada de gran alcance que permite realizar en paralelo aplicaciones de detección, segmentación y procesamiento de imágenes, la placa cuenta con una GPU Maxwell de 128 núcleos integrada, una CPU ARM A57 de 64 bits, memoria LPDDR4 de 4GB, junto con soporte para E / S de alta velocidad MIPI CSI-2 y PCIe Gen2, estas características son idóneas para poder implementar sistemas de visión artificial y aprendizaje profundo.
- OpenCV y ROS pueden trabajar en conjunto para el procesamiento de una imagen en este caso la detección de la postura humana, la imagen usa las facilidades de OpenCV y usa la pila vision_opencv, que proporciona un paquete de la biblioteca OpenCV para ROS este paquete es vision_opencv proporciona cv_bridge que es el puente entre mensajes ROS y OpenCV.
- Java, Python, Lisp, Prolog y C ++ son los principales lenguajes de programación de inteligencia artificial utilizados para la inteligencia artificial, capaces de satisfacer diferentes necesidades en el desarrollo y diseño de diferentes programas. Corresponde a un desarrollador elegir cuál de los idiomas de AI gratificará la funcionalidad y las características deseadas de los requisitos de la aplicación.
- Los nodos usados en la arquitectura de detección de posturas humanas para el robot Kuka son: Nodo Ros que es encargado del procesamiento de imágenes convierte las imágenes de ROS a formato OpenCV o viceversa a través de CvBridge, Nodo Aplicación de Usuario este Ejecuta la comunicación entre el Cliente y el Servidor a través del Protocolo de Acción de ROS, que se construye sobre los mensajes de ROS, Nodo de Control que es el que transforma los comandos en en medidas o señales que pueden ser

comprendidas por los actuadores del robot, el nodo YouBot Hardware: Es el espacio donde el Robot Kuka toma los datos y realiza las acciones, las acciones son por separado el brazo o la plataforma.

5.2 Recomendaciones

- Para una rápida comunicación con el robot Kuka es aconsejado usar EtherCAT por la velocidad de trasmisión que ofrece, en nuestro caso utilizamos el protocolo 802.11 (wifi) para evitar que los cables estén conectado al robot, el protocolo 802.11 el robot demora un tiempo en recibir las ordenes.
- Al emplear placas para desarrolladores, verificar las versiones que son compatibles con las plataformas que tiene pensado utilizar, esto ahorrara tiempo ya que una sola versión o plataforma que no esté soportada por su placa causaría conflicto en su investigación.
- Verificar todas las características de los equipos que está utilizando para evitar posibles daños, se recomienda conocer los voltajes máximos y mínimos con los que puede trabajar los dispositivos

Bibliografía

[1] Alejandro Ramirez Serrano. "NAVEGACIÓN AUTÓNOMA DE ROBOTS MÓVILES MEDIANTE LÓGICA DIFUSA", Instituto Tecnológico Y De Estudios Superiores De Monterrey Campus Ciudad De México, 1998

[2] Kawamur Phillip J. McKerrow, lntroduction to Robotics, Addison Wesley 1991, ISBN : 0-201- 18240.

[3] Anita M. Flynn & Joseph L. Jones, Mobile Robots. lnspiration to lmplementation, A.K. Peters 1993, ISBN : 1-56881-011-3.

[4] Okumura, T., Y. Shimizu, M. Furuta and Tawara T. "Design and construction of a series of compact humanoid robots and development of biped walk control strategies". Robotics and AutonomousSystems, 37:81-100 (2001).

[5] ROSAS-ARIAS, Leonel †, VALLEJO-MERAZ, Jair de Jesus, PÉREZ-BAILÓN, Waldemar, ROJAS-CID, Jesús Daniel "Robot clasificador de objetos de color utilizando técnicas de filtrado RGB". Instituto Tecnológico De Lázaro Cárdenas. .10 50-59 (2017).

[6] Robotic Industries Association. "Inteligencia artificial y machine learning y su aplicación a la robótica industrial", 2018. Available: https://prograbox.com/inteligencia-artificial-y-machine-learning-y-su-aplicacion-a-la-robotica-industrial/

[7] Patricio Javier Guaraca Medina, Jorge Leonardo Ochoa Ochoa, "Estudio de la Programación Y Operación de los Robots Industriales Kuka Kr16-2 Y Kr5-2 Arc Hw", Universidad Politécnica Salesiana Sede Cuenca, 2015. Available: https://es.scribd.com/document/346567126/PROGRAMACION-Y-OPERACION-DE-LOS-ROBOTS-INDUSTRIALES-KUKA.

[8] Rafael Aracil, Carlos Balaguer, Manuel Armada "Robots de servicio". Revista Iberoamericana de Automática e Informática Industrial RIAI (2018).

[9] Gordón C., P. Encalada, Lema H., León D., Peñaherrera C. (2019) Robot autónomo KUKA YouBot Navegación basada en la planificación de rutas y reconocimiento de señales de tráfico. En: Arai K., Bhatia R., Kapoor S. (eds)

Actas de la Conferencia de Tecnologías Futuras (FTC) 2018. FTC 2018. Avances en Sistemas Inteligentes y Computación, vol. 880. Springer, Cham

[10] Claudia Russo, Hugo Ramón, Sandra Serafino, Benjamin Cicerchia1, Mónica Sarobe, Agustín Balmer, Álvarez Eduardo, Pablo Luengo, Gustavo Useglio, Martín Faroppa"Visión artificial aplicada en Agricultura de Precisión"Instituto de Investigación y Transferencia en Tecnología (ITT) 5 Comisión de Investigaciones Científicas (CIC) Escuela de Tecnología (ET) Universidad Nacional del Noroeste de la Provincia de Buenos Aires (UNNOBA). (2018). Available: http://sedici.unlp.edu.ar/bitstream/handle/10915/68308/Documento_completo.pdf-PDFA.pdf?sequence=1&isAllowed=y

[11] Ing. César Paúl Carrillo Delgado, "Selección y manipulación de objetos a partir de su forma geométrica y color por medio de un brazo robótico"; Centro de Investigaciones en Óptica, A.C. León, GTO, 2010.

[12] L.K Garzón-Obregón, L.A Forero-Rincón, y O.M Duque-Suárez, "Diseño e implementación de un sistema de visión artificial usando una técnica de mapeo y localización simultánea (SLAM) sobre una plataforma robótica móvil" Mundo Fesc, vol. 15, no. 1, pp. 8-17, 2018.

[13] Ignacio Romero Guillén, "ESTACIÓN ROBOTIZADA DE PALETIZADO", Universidad de Politécnica de Catalunya de Barcelona, España, 2017. Available: https://upcommons.upc.edu/bitstream/handle/2117/118106/TFG%20Estaci%C3%B3n%20robotizada%20de%20paletizado%20IGNACIO%20ROMERO.pdf?sequence=1&isAllowed=y

[14] Kovács, A. y Ueno, H. (2005): Cognitive Substrates: What They Are and What They Learn. Página personal del autor. 4 de marzo. 2016.

[15] B.J. Copeland, "Artificial intelligence" [Online]. Available: https://www.britannica.com/technology/artificial-intelligence/Evolutionary-computing. [Accessed: 29-Apr-2019].

[16] Ayn de Jesus, "Aplicaciones de visión artificial - Compras, conducción y más"

[Online]. Available: https://emerj.com/ai-sector-overviews/computer-vision-applications-shopping-driving-and-more/, [Accessed: 10-Mayo-2019].

[17] Morgan Quigley, Eric Berger, Andrew Y. Ng (2007), STAIR: Hardware and Software Architecture, AAAI 2007 Robotics Workshop.

[18] Kurt, "Object Detection Tutorial in TensorFlow: Real-Time Object Detection" 2019. [Online]. Available: https://www.edureka.co/blog/tensorflow-object-detection-tutorial/. [Accessed: 22-Mayo-2019].

[19] Max Motor, Robot omnidirectional autónomo KUKA YouBot, Available:https://www.maxonmotor.es/medias/sys_master/root/8815739797534/2014-10-story-es-kuka-research-robot.pdf?attachment=true

[20] Daniel Ortego Delgado, "¿Qué es ROS?" Openwebinar2017. Available: https://openwebinars.net/blog/que-es-ros/.

[21] Adnan Ademovic, "C Una introducción al sistema operativo de robots: el último marco de aplicación de robots" [Online]. Available: https://www.toptal.com/robotics/introduction-to-robot-operating-system

[22] Existek, "Programación de IA: 5 lenguajes de programación de IA más populares" [Online]. Available: https://existek.com/blog/ai-programming-and-ai-programming-languages/. 2018

[23] José Vicente Carratala, Revista Jocarsa tecnología Humana, "¿Cuál es el mejor lenguaje de programación para inteligencia artificial?". Enero 6, 2019. [En línea]. Available:https://jocarsa.com/cual-es-el-mejor-lenguaje-de-programacion-para-inteligencia-artificial.

[24] BBVA API_Market, INTELIGENCIA ARTIFICIAL, "Plataformas de Inteligencia Artificial para desarrolladores: reconocimiento de voz" 19 ENE.2016, [Enínea]. Available: https://bbvaopen4u.com/es/actualidad/plataformas-de-inteligencia-artificial-para-desarrolladores-reconocimiento-de-voz.

[25] Ainara Estaun Gabás, Rafael Cabeza Laguna. T:T:S de Ingenieria Industrial, Informatica y de Telecomunicaciones. "¿Adecucaion de un sistema de vision

artificial para control de calidad?". Enero 07, 2016. [En línea]. Available: https://academicanavarra.es/bitstream/handle/2454/21702/TFG_AinaraEstaun-1.pdf?sequence=1&isAllowed=y.

[26] James Le en Heartbeat ,Hearbeat "The 5 Computer Vision Techniques That Will Change How You See The World" 2018. [Online]. Available: https://heartbeat.fritz.ai/the-5-computer-vision-techniques-that-will-change-how-you-see-the-world-1ee19334354b

[27] C. López-Franco, J. Hernández-Barragán, A. Y. Alanis, N. Arana-Daniel, and M. López-Franco, "Inverse kinematics of mobile manipulators based on differential evolution," Int. J. Adv. Robot. Syst., vol. 15, no. 1, pp. 1–22, 2018.

[28] F. G. Sales, "Control de trayectoria de la simulación de un brazo robot de 5 grados de libertad, controlado mediante la plataforma C2000 Piccolo

[29] J. Guadalupe, Z. Villalpando, S. Alfonso, G. Blanco, J. José, and G. Mandujano, "Cinemática inversa , Fanuc LR Mate 200ic," no. 103, pp. 202–227, 2013.

[30] Lowe, D. G., "Object recognition from local scale-invariant features", International Conference on Computer Vision, Corfu, Greece, September 1999.

[31] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce, Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene categories, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, New York, June 2006, vol. II, pp. 2169-2178

[32] L. Jon and R. Martinez, "Tema 6. dinámica de robots y control," *Open Course Ware*, p. 46, 2018.

[33] Lowe, David G. (1999). "Reconocimiento de objetos locales de escala invariante características". Actas de la Conferencia Internacional sobre la Visión por Computador [consultada 30 de Marzo del 2013]

[34] Bay. H, Tuytelaars. T, Van Gool L. (2008) "SURF: Speeded Up Robust Features", disponible en URL: http://www.vision.ee.ethz.ch/~surf/eccv06.pdf, [consultada 31 de Marzo del 2013]

[35] Rublee. E, Rabaud. V, Konolige. K, Bradski, K. (2012) "ORB: anefficientalternativeto SIFT or SURF", disponible en URL: https://willowgarage.com/sites/default/files/orb_final.pdf, [consultada 31 de Marzo del 2013]

[36] Tatiana Aparicio Vergara y Cristóbal Sebastián González Dixon, "Sistema de Reconocimiento de Posturas del Cuerpo Humano" 2005. [Online]. Available: http://www.setianworks.net/publicaciones/Sistema_de_reconocimiento_de_posturas_del_cuerpo_humano.pdf

37] J. G. Pérez, "Introducción a ROS en Raspberry Pi," Universidad Abierta de Cataluña, 2017.

[38] A. Araújo, D. Portugal, M. S. Couceiro, and R. P. Rocha, "Integrating Arduino-Based Educational Mobile Robots in ROS," *J. Intell. Robot. Syst.*

[39] GitHuba, "OpenCV: Biblioteca de visión de computadora de código abierto" [Online]. Available: https://github.com/opencv/opencv

ANEXOS

Anexo A.

DEVELOPER KIT SETUP AND HARDWARE

The NVIDIA® Jetson Nano™ Developer Kit is an AI computer for makers, learners, and developers that brings the power of modern artificial intelligence to a low-power, easy-to-use platform. Get started quickly with out-of-the-box support for many popular peripherals, add-ons, and ready-to-use projects.

Jetson Nano is supported by the comprehensive NVIDIA® JetPack™ SDK, and has the performance and capabilities needed to run modern AI workloads. JetPack includes:

- Full desktop Linux with NVIDIA drivers
- AI and Computer Vision libraries and APIs
- Developer tools
- Documentation and sample code

DEVELOPER KIT SETUP

Before using your developer kit, you need to set up a microSD card with the operating system and JetPack components. The simplest method is to download the microSD card image and follow instructions found in Getting Started with Jetson Nano Developer Kit.

In summary:

- You need a 16 GB or larger UHS-1 microSD card, HDMI or DP monitor, USB keyboard and mouse, and 5V⎓2A Micro-USB power supply.
- Download the image and write it to the microSD card.
- Insert the microSD card into the slot under the Jetson Nano module, then attach the display, keyboard, mouse, and Ethernet cable or wireless networking adapter.
- Connect the Micro-USB power supply. The developer kit powers on automatically.

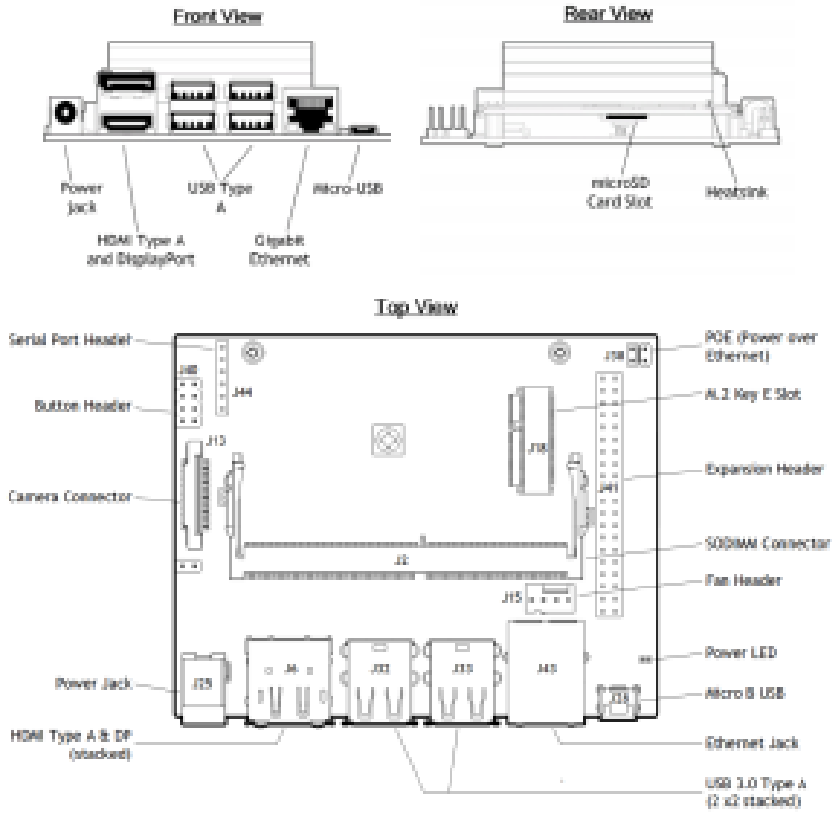
For alternative methods, see How to Install JetPack, below.

INCLUDED IN THE BOX

- Jetson Nano module (P3448-0000)
- Reference carrier board (P3449)
- Small paper card with quick start and support information
- Folded paper stand for the developer kit

DEVELOPER KIT INTERFACES

Developer kit module and carrier board



Anexo B.

KUKA youBot

Operating and Assembly Instructions

Issued: 29.11.2012

Version: MA KUKA youBot V4 en (PDF)

Product description

Overview of KUKA youBot

Mobile manipulation is a key technology for future service robots in private and industrial environments. KUKA offers the modular robot system KUKA youBot for research into this key technology.

The KUKA youBot system consists of the following hardware components:

- youBot arm (5-axis robot arm)
- youBot platform (omnidirectional mobile platform)
- youBot gripper module (two-finger gripper) (optional)
- youBot loading area (optional)

Fig. 3-1: KUKA youBot, overview

Item	Description
1	Gripper (optional)
2	youBot arm
3	youBot platform
4	youBot loading area (optional)

The components youBot arm and youBot platform can be operated together or separately. The youBot platform is operated with the youBot battery. The youBot arm can be supplied with power either via the youBot platform or separately using the youBot power supply unit.

The drives of the youBot platform and the youBot arm have an EtherCAT interface whose protocol is described in the supplier documentation from Trinamic. Both components can be controlled by a PC integrated into the youBot platform or by an external PC (not included in the scope of supply) using EtherCAT telegrams.

The software required for controlling the system is not part of the scope of supply and is either created in the context of research projects or can be obtained

Anexo C.

Código OpenCV-ROS (find_object)

```
#ifndef FIND_OBJECT_2D_MESSAGE_OBJECTSSTAMPED_H
#define FIND_OBJECT_2D_MESSAGE_OBJECTSSTAMPED_H


#include <string>
#include <vector>
#include <map>

#include <ros/types.h>
#include <ros/serialization.h>
#include <ros/builtin_message_traits.h>
#include <ros/message_operations.h>

#include <std_msgs/Header.h>
#include <std_msgs/Float32MultiArray.h>

namespace find_object_2d
{
template <class ContainerAllocator>
struct ObjectsStamped_
{
  typedef ObjectsStamped_<ContainerAllocator> Type;

  ObjectsStamped_()
    : header()
    , objects()  {
    }
  ObjectsStamped_(const ContainerAllocator& _alloc)
    : header(_alloc)
    , objects(_alloc)  {
  (void)_alloc;
    }



   typedef  ::std_msgs::Header_<ContainerAllocator>
_header_type;
  _header_type header;

   typedef  ::std_msgs::Float32MultiArray_<ContainerAllocator>
_objects_type;
  _objects_type objects;




  typedef boost::shared_ptr<
::find_object_2d::ObjectsStamped_<ContainerAllocator> > Ptr;
```

```
  typedef boost::shared_ptr<
::find_object_2d::ObjectsStamped_<ContainerAllocator> const>
ConstPtr;

}; // struct ObjectsStamped_

typedef ::find_object_2d::ObjectsStamped_<std::allocator<void> >
ObjectsStamped;

typedef boost::shared_ptr< ::find_object_2d::ObjectsStamped >
ObjectsStampedPtr;
typedef boost::shared_ptr< ::find_object_2d::ObjectsStamped
const> ObjectsStampedConstPtr;

// constants requiring out of line definition



template<typename ContainerAllocator>
std::ostream& operator<<(std::ostream& s, const
::find_object_2d::ObjectsStamped_<ContainerAllocator> & v)
{
ros::message_operations::Printer<
::find_object_2d::ObjectsStamped_<ContainerAllocator>
>::stream(s, "", v);
return s;
}

} // namespace find_object_2d

namespace ros
{
namespace message_traits
{



// BOOLTRAITS {'IsFixedSize': False, 'IsMessage': True,
'HasHeader': True}
// {'find_object_2d': ['/tmp/binarydeb/ros-melodic-find-object-
2d-0.6.2/msg'], 'sensor_msgs':
['/opt/ros/melodic/share/sensor_msgs/cmake/../msg'], 'std_msgs':
['/opt/ros/melodic/share/std_msgs/cmake/../msg'],
'geometry_msgs':
['/opt/ros/melodic/share/geometry_msgs/cmake/../msg']}

// !!!!!!!!!!! ['__class__', '__delattr__', '__dict__',
'__doc__', '__eq__', '__format__', '__getattribute__',
'__hash__', '__init__', '__module__', '__ne__', '__new__',
'__reduce__', '__reduce_ex__', '__repr__', '__setattr__',
'__sizeof__', '__str__', '__subclasshook__', '__weakref__',
'_parsed_fields', 'constants', 'fields', 'full_name',
'has_header', 'header_present', 'names', 'package',
'parsed_fields', 'short_name', 'text', 'types']
```

```
template <class ContainerAllocator>
struct IsFixedSize<
::find_object_2d::ObjectsStamped_<ContainerAllocator> >
  : FalseType
  { };

template <class ContainerAllocator>
struct IsFixedSize<
::find_object_2d::ObjectsStamped_<ContainerAllocator> const>
  : FalseType
  { };

template <class ContainerAllocator>
struct IsMessage<
::find_object_2d::ObjectsStamped_<ContainerAllocator> >
  : TrueType
  { };

template <class ContainerAllocator>
struct IsMessage<
::find_object_2d::ObjectsStamped_<ContainerAllocator> const>
  : TrueType
  { };

template <class ContainerAllocator>
struct HasHeader<
::find_object_2d::ObjectsStamped_<ContainerAllocator> >
  : TrueType
  { };

template <class ContainerAllocator>
struct HasHeader<
::find_object_2d::ObjectsStamped_<ContainerAllocator> const>
  : TrueType
  { };


template<class ContainerAllocator>
struct MD5Sum<
::find_object_2d::ObjectsStamped_<ContainerAllocator> >
{
  static const char* value()
  {
    return "5ec2736b62b92d101276c97e8db387b1";
  }

  static const char* value(const
::find_object_2d::ObjectsStamped_<ContainerAllocator>&) { return
value(); }
  static const uint64_t static_value1 = 0x5ec2736b62b92d10ULL;
  static const uint64_t static_value2 = 0x1276c97e8db387b1ULL;
};

template<class ContainerAllocator>
struct DataType<
::find_object_2d::ObjectsStamped_<ContainerAllocator> >
```

```
{
  static const char* value()
  {
    return "find_object_2d/ObjectsStamped";
  }

  static const char* value(const
::find_object_2d::ObjectsStamped_<ContainerAllocator>&) { return
value(); }
};

template<class ContainerAllocator>
struct Definition<
::find_object_2d::ObjectsStamped_<ContainerAllocator> >
{
  static const char* value()
  {
    return "# objects format: \n"
"# [ObjectId1, objectWidth, objectHeight, h11, h12, h13, h21,
h22, h23, h31, h32, h33, ObjectId2...] \n"
"# where h## is a 3x3 homography matrix (h31 = dx and h32 = dy,
see QTransform)\n"
"Header header\n"
"std_msgs/Float32MultiArray objects \n"
"================================================================
===============\n"
"MSG: std_msgs/Header\n"
"# Standard metadata for higher-level stamped data types.\n"
"# This is generally used to communicate timestamped data \n"
"# in a particular coordinate frame.\n"
"# \n"
"# sequence ID: consecutively increasing ID \n"
"uint32 seq\n"
"#Two-integer timestamp that is expressed as:\n"
"# * stamp.sec: seconds (stamp_secs) since epoch (in Python the
variable is called 'secs')\n"
"# * stamp.nsec: nanoseconds since stamp_secs (in Python the
variable is called 'nsecs')\n"
"# time-handling sugar is provided by the client library\n"
"time stamp\n"
"#Frame this data is associated with\n"
"string frame_id\n"
"\n"
"================================================================
===============\n"
"MSG: std_msgs/Float32MultiArray\n"
"# Please look at the MultiArrayLayout message definition for\n"
"# documentation on all multiarrays.\n"
"\n"
"MultiArrayLayout  layout        # specification of data
layout\n"
"float32[]         data          # array of data\n"
"\n"
"\n"
"================================================================
===============\n"
"MSG: std_msgs/MultiArrayLayout\n"
```

```
"# The multiarray declares a generic multi-dimensional array of
a\n"
"# particular data type.  Dimensions are ordered from outer
most\n"
"# to inner most.\n"
"\n"
"MultiArrayDimension[] dim # Array of dimension properties\n"
"uint32 data_offset        # padding elements at front of data\n"
"\n"
"# Accessors should ALWAYS be written in terms of dimension
stride\n"
"# and specified outer-most dimension first.\n"
"# \n"
"# multiarray(i,j,k) = data[data_offset + dim_stride[1]*i +
dim_stride[2]*j + k]\n"
"#\n"
"# A standard, 3-channel 640x480 image with interleaved color
channels\n"
"# would be specified as:\n"
"#\n"
"# dim[0].label  = \"height\"\n"
"# dim[0].size   = 480\n"
"# dim[0].stride = 3*640*480 = 921600  (note dim[0] stride is
just size of image)\n"
"# dim[1].label  = \"width\"\n"
"# dim[1].size   = 640\n"
"# dim[1].stride = 3*640 = 1920\n"
"# dim[2].label  = \"channel\"\n"
"# dim[2].size   = 3\n"
"# dim[2].stride = 3\n"
"#\n"
"# multiarray(i,j,k) refers to the ith row, jth column, and kth
channel.\n"
"\n"
"================================================================
================\n"
"MSG: std_msgs/MultiArrayDimension\n"
"string label   # label of given dimension\n"
"uint32 size    # size of given dimension (in type units)\n"
"uint32 stride  # stride of given dimension\n"
;
  }

  static const char* value(const
::find_object_2d::ObjectsStamped_<ContainerAllocator>&) { return
value(); }
};

} // namespace message_traits
} // namespace ros

namespace ros
{
namespace serialization
{
```

```
  template<class ContainerAllocator> struct Serializer<
::find_object_2d::ObjectsStamped_<ContainerAllocator> >
  {
    template<typename Stream, typename T> inline static void
allInOne(Stream& stream, T m)
    {
      stream.next(m.header);
      stream.next(m.objects);
    }
```