Universidad Complutense de Madrid

Universidad Nacional de Educación a Distancia

Proyecto Final del Máster

Máster en Ingeniería de Sistemas y de Control

Título:

# "Navegación por visión y control de un robot bípedo"

Autor: Orlando Barcia Ayala

Director: Dictino Chaos García

Curso Académico 2016/2017 – Convocatoria ordinaria de septiembre

Universidad Complutense de Madrid

Universidad Nacional de Educación a Distancia

Proyecto Final del Máster

Máster en Ingeniería de Sistemas y de Control

Título:

# "Navegación por visión y control de un robot bípedo"

Tipo B: Proyecto específico propuesto por el alumno

Autor: Orlando Barcia Ayala

Director: Dictino Chaos García

**Autorización**

Autorizo a la Universidad Complutense de Madrid UCM y a la Universidad Nacional de Educación a Distancia UNED a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria de este Trabajo Fin de Máster, cómo el código, la documentación y/o el prototipo desarrollado.

Firmado:

Orlando Barcia Ayala

**Resumen**

Hay investigaciones que trabajan con robots autónomos de ruedas, otros con robots humanoides ayudados con sensores externos para percepción del entorno. Sin embargo, en la realidad ecuatoriana existen muy pocas investigaciones en robots bípedos que tratan el problema de la locomoción, localización, navegación de forma autónoma y teleoperado.

Realizar la navegación y control de un robot autónomo bípedo con sensores como el de visión artificial y sonar sin necesidad de sensores externos para localizarse en un entorno y obtener datos del comportamiento del robot en su locomoción es todavía un problema de estudio actual.

La meta es integrar en el robot bípedo NAO técnicas de visión artificial para efectuar la navegación en entornos de interiores, detectando obstáculos, realizando la planificación y seguimiento de una trayectoria en un mapa conocido.

Para lo cual se utiliza visión artificial con OpenCv para percibir la detección del entorno de líneas, contornos, áreas mediante la segmentación y descripción de la imagen extrayendo características como colores, bordes, aplicando la detección de esquinas Harris, Transformadas de Hough, descriptor SIFT, SURF, ORB y FAST para encontrar landmark o puntos de refencia visuales. El emparejamiento se realiza mediante emparejamiento o matching con SIFT y para visualización con homografía con las características encontradas. La localización se efectúa aplicando SLAM Simultaneous Localization And Mapping. (Simultanea Localización y Mapeo) mediante el método probabilístico de filtro de partículas.

Se utiliza un robot virtual Webots y dos robots bípedos NAO, experimentando en un entorno de interiores y se limita a utilizar el movimiento de las piernas utilizando solo los elementos propios del robot sin añadir ningún dispositivo adicional.

El robot se desplaza a un punto de referencia de forma autónoma, se mueve de forma teleoperada, describe el objeto mediante visión artificial y con la posición absoluta almacenada del objeto se procede a mover al punto en el plano bidimensional. El filtro de partículas recibe la posición y localiza al objeto mientras se mueve.

**Palabras claves:**

Visual Landmark, visión artificial, navegación, SLAM, filtro de partículas, control.

**Abstract**

There are researches that work with autonomous robots of wheels, others with humanoid robots aided by external sensors for perception of the surroundings. However, in Ecuadorian reality there is very little research on bipedal robots that deal with the problem of locomotion, localization, autonomous and teleoperated navigation.

To perform the navigation and control of a bipedal autonomous robot with sensors such as artificial vision and sound without the need of external sensors to locate in an environment and to obtain data of the behavior of the robot in its locomotion is still a problem of current study.

The goal of this project is to integrate in the biped robot NAO techniques of artificial vision to carry out navigation in interior environments, detecting obstacles, performing the planning and monitoring of a trajectory in a known map.

For this purpose, artificial vision with OpenCv is used to perceive the detection of the surroundings of lines, contours, areas through the segmentation and description of the image extracting characteristics like colors, borders, applying the detection of Harris corners, Hough transformations, SIFT descriptor, SURF, ORB and FAST to find visual landmarks. The matching is done by SIFT and with homography with the characteristics found for display. The localization is carried out by applying Simultaneous Localization and Mapping (SLAM) using the probabilistic method of a particle filter.

This project uses a Webots virtual robot and two NAO biped robots, experimenting in an indoor environment and is limited to use the movement of the legs using only the robot's own elements without adding any additional device.

The robot moves to a reference point autonomously, moves in a teleoperated way, describes the object through artificial vision and with the absolute stored position of the object moves to the point in the two-dimensional plane. The particle filter receives the position and locates the object as it moves.

**Keywords:**

Visual Landmark, artificial vision, navigation, SLAM, Particle filter, control.

**Agradecimientos**

De forma infinita a Jehová Dios, por mantenerme con vida y a pesar de los diferentes problemas de salud y operaciones, poder continuar para proteger y cuidar a mis hijos.

A mi esposa Bélgica por su tiempo y comprensión en estos momentos de trabajo, que involucra sacrificios, pero son la semilla para recibir en el futuro buenos frutos.

A mi país Ecuador y mi provincia Manabí por recuperarse del terremoto del 2016, estamos en deuda por no tener dispositivos robóticos que hubiesen ayudado a salvar vidas, vamos a trabajar en realizarlos.

A las autoridades de la Universidad Politécnica Salesiana, por el apoyo en la maestría y al permitir el acceso a los laboratorios y Robots NAO.

A mi tutor Profesor Dr. Dictino Chaos, por su paciencia, comprensión y guía en los avances del trabajo de fin de Máster.

**Dedicatoria**

Dedico este trabajo a mis hijos Giovannita y Orlandito que son el motivo y fuerzas para concluirlo. A mi esposa Bélgica, mi madre Carmen Amelia, a mi hermano Israel y mi Padre Orlando.

Espero no decepcionarlos.

## Índice General

**Lista de símbolos y nomenclatura**

| Símbolo | Unidad | Descripción |
|---|---|---|
| $A^*$ | | Planificador de trayectoria A start. |
| t | s | Tiempo en segundos. |
| ${}^{A}T_{B}$ | | Transformada homogénea que representa el marco o frame {B} respecto al frame {A}, o desde A hacia B. Si {A} no se indica, se considera que es relativo al frame de coordenadas mundo o frame 0. $${}^{A}T_{B} = \left( {}^{B}T_{A} \right)^{-1}$$ |
| $\theta$ | | Ángulo, en este documento respecto al eje Z (yaw) |
| $\theta_x, \theta_y, \theta_z$ | | Ángulos de vectores roll, pitch, yaw. En este documento, respectivamente giro sobre eje X, eje Y, eje Z. |
| $x_t = (x_t, y_t, \theta_t)$ | | Pose en el tiempo t. X, Y, ángulo. |
| $\oplus$ | | Dilatación morfológica. |
| $\ominus$ | | Erosión morfológica. |
| $\circ$ | | Apertura morfológica / Opening. Erosión seguida dilatación. |
| $\bullet$ | | Cierre morfológico / Closing. Dilatación seguida de erosión. |
| $\{F\}$ | | F frame o marco de coordenadas. |
| m | m | Metros (medida de longitud) |

## Índice de figuras

## Índice de tablas

# 1. Introducción.

## 1.1. Marco del Proyecto

El proyecto es la parte final del Master Inter Universitario en Ingeniería de Sistemas y de Control de la UNED y UCM. Se ha desarrollado en los Laboratorios de automatización y fabricación flexible de la Carrera de Ingeniería Electrónica de la Universidad Politécnica Salesiana de Ecuador, sede Guayaquil.

## 1.2. Motivación del Proyecto

Existen muchos trabajos en robótica que presentan de forma aislada o en conjunto de las actividades que debe realizar un robot autónomo como son la locomoción, percepción, planeación y navegación. Sin embargo, a pesar de la amplia literatura a nivel mundial, La aplicación de estos algoritmos avanzados se utiliza muy poco en la realidad ecuatoriana encontrando trabajos académicos muy puntuales de visión artificial o armado de robots, pero no de localización, visión artificial para descripción del entorno, mapeo y navegación. De este modo hay una línea base muy limitada y un campo amplio para investigar.

Esto se fundamenta en los resultados de las encuestas aplicadas a estudiantes de Ingeniería Electrónica de la Universidad Politécnica Salesiana sede Guayaquil en el cual de los 39 estudiantes de últimos niveles el 92% no aplican en sus proyectos la visión artificial. Un 5% utiliza Python, Un 1% utiliza OpenCv para visión artificial. Un 98% desconoce el funcionamiento de algoritmos de localización, mapeo y navegación.

Los estudiantes se han limitado al manejo de robots manipuladores, matrices de rotación, uso del sensor ultrasónico, uso de sensores de contacto para seguimiento, a robots de batallas y a crear prototipos de robots ensamblados por partes, teleoperados por palancas o controles remotos de tipo infrarrojo. El instrumento de preguntas está en el anexo A.9.

El proyecto propone iniciar en el grupo de investigación en control y robótica la línea de visión artificial, localización, mapeo y navegación.

## 1.3. Objetivos

El proyecto tiene un objetivo general y varios específicos como se detalla a continuación.

**Objetivo general**

Integrar en el robot bípedo NAO técnicas de visión artificial para efectuar la navegación en entornos de interiores, detectando obstáculos que serán mencionados mediante señales audibles, realizando la planificación y seguimiento de una trayectoria o camino en un mapa conocido.

**Objetivos específicos.**

- Realizar el control del robot, sincronizando las actividades requeridas para posicionarlo en un punto dado y realizar el seguimiento de una trayectoria o camino.
- Aplicar visión artificial para efectuar la navegación en entornos de interiores (indoor) mediante balizas pasivas naturales.
- Realizar la detección de proximidad y obstáculos aplicando técnicas de visión artificial y ultrasónicas siendo mencionados mediante señales audibles del robot.
- Describir en el robot bípedo los errores sistemáticos y no sistemáticos detectados entre la ubicación estimada y la real.

**1.4. Alcance del proyecto**

El trabajo se delimita en la república del Ecuador, es de tipo académico, aplicado en el aula como parte de una propuesta de investigación de la línea de investigación del Grupo GISCOR Grupo de Investigación Control y robótica de la Universidad Politécnica Salesiana sede Guayaquil en el área de robótica móvil. El trabajo es de tipo innovación tecnológica, es experimental, de alcance descriptivo y enfoque cuantitativo. Se consideran condiciones ideales en un entorno controlado indoor (laboratorios y pasillos de los laboratorios). La Universidad Politécnica Salesiana tienen dos robots NAO blue y red de acceso limitado el laboratorio. Para disminuir este inconveniente se utiliza un software simulador antes de ejecutarlo en el robot real. Al robot seleccionado, no se añade ningún sensor adicional, por lo cual se utiliza el sensor de visión la cámara monocular (superior o inferior) y el sensor ultrasónico.

Los dos robots antes de la implementación del prototipo fueron reparados en el exterior (Francia). Una vez reintegrado los servomotores tienen inconveniente al caminar no han quedado en buen comportamiento.

El producto es un sistema teleoperado, de locomoción y navegación de un robot bípedo. La implementación se realizó en Python y OpenCv. La simulación del entorno y el robot se utiliza con el software Webots. Se ha escogido estas herramientas con el afán de conocer otras tecnologías, aunque el autor utiliza en sus actividades diarias Java y Matlab.

Debido a la naturaleza y objetivos del trabajo que es la locomoción, localización y navegación, se considera el movimiento en el espacio en el plano de dos dimensiones. Existen muchas investigaciones para optimizar el movimiento del robot, velocidad, estabilidad del torso al caminar, movimientos de manos. Así como el manejo en el espacio de 3 dimensiones. Estas y otras características no son consideradas y quedan fuera del alcance del trabajo.

### 1.5. Resultados alcanzados e Impacto

El impacto o cambio esperado al implementar el trabajo es la creación un grupo de investigación en la línea relativas a locomoción, visión artificial, navegación. Así también sea la línea base de propuestas e implementaciones para futuros proyectos

Los resultados alcanzados del trabajo son:

- Realizar el control del robot para efectuar el movimiento a un punto dado.
- Localizar el robot en entorno conocido y fijo
- Navegar en el entorno mediante visión artificial y sensores de ultrasonido
- Detectar proximidad por visión artificial y sensores de ultrasonido
- Que el robot emita señales audibles indicando su estado
- Efectuar simulaciones en el robot virtual e interactuar con el robot real con la mayor compatibilidad posible

### 1.6. Organización del documento

El documento está estructurado en cinco apartados:

1. La introducción donde se estable la motivación, objetivos del proyecto, e impacto del proyecto.
2. Los antecedentes y estado del arte, donde se establecen los fundamentos teóricos e investigaciones relacionadas al trabajo.
3. El desarrollo donde se efectúa la selección del robot bípedo a utilizar, las herramientas de software y hardware requeridas, el diseño y estrategias

establecidas para la solución. Se realizan las pruebas tanto en el simulador como el robot real.

4. El diseño de la aplicación, donde se describe las interfaces para controlar el robot, localizar, visualizar lo que el robot percibe y observar los procesos de visión artificial efectuados en la imagen.
5. Se presentan los Resultados obtenidos por el trabajo y videos de los experimentos.
6. Se exponen por último las conclusiones, trabajos futuros y problemas detectados.

En los anexos se sintetiza la información referente a los videos de los experimentos, características técnicas del hardware y otros relativo a algunos puntos del proyecto.

## 2. Antecedentes y Estado del arte

La robótica es de naturaleza multidisciplinaria, debido a la aplicación de varias áreas del conocimiento como matemáticas, física, electrónica, computación, inteligencia artificial y control [1]. Los robots son agentes físicos que realizan tareas de manipulación física en el entorno que lo rodea. Están equipados con efectores como piernas, ruedas, articulaciones y o pinzas de agarre. Los robots están equipados con sensores que permiten percibir el entorno. La mayoría de robots en la actualidad están en las siguientes categorías: manipulador industrial, robots móviles y robots móviles manipuladores [2][3]. Los robots manipuladores industriales tienen una gran desventaja que es la falta de gran movilidad, debido a que tiene un limitado rango de movimiento [4]. De acuerdo a ISO 8373 se define al Robot Móvil como el que contiene lo necesario para el pilotaje y movimiento tanto en potencia, control y sistema de navegación [3][4]. Un medio habitual de locomoción natural es la tierra. Los robots terrestres de acuerdo sus locomociones se clasifican en robots de ruedas o patas. Entre los robots de patas hay los bípedos (de dos patas), cuadrúpedos, hexápodos y de muchas más [1]. Los robots humanoides son máquinas antropomórficas que tiene el fin de imitar al ser humano al realizar las funciones de caminar, ver, hablar, recoger, limpiar y trasladar elementos u objetos. Estos robots de locomoción bípeda son sistemas muy complejos en la parte mecánica, electrónica, de percepción por sensores, servomotores, programación, control y modelado matemático [3].

En un robot móvil, una de las metas es que se mueva sin supervisión en entornos reales para cumplir una actividad o tarea. Por lo cual se deben realizar algoritmos para que el robot pueda resolver estar interrogantes:

¿Cómo se mueve de forma física y mecánica a un punto determinado? Locomoción.

¿Cómo detecta o identifico a un objeto? ¿Qué objetos están frente al robot? Percepción visual y percepción ultrasónica.

¿Dónde se encuentra? ¿Dónde está? Localización.

¿Cómo es el entorno que lo rodea? Mapping.

¿Cuál es la mejor manera de llegar a un determinado lugar? ¿Cómo hace el robot para evitar obstáculos? Mapping y Navegación.

En la actualidad estas interrogantes y soluciones son aún estudiadas. Por lo cual se considera que es un campo aún joven, abierto a futuras propuestas, enfoques y soluciones.

Un robot móvil requiere mecanismos de locomoción que le permita moverse en un entorno controlado o no controlado. En [5] se describe el modelo de lazo cerrado para la caminada ominidireccional del robot NAO. El robot NAO basa su funcionamiento en [6], donde se describe el péndulo invertido lineal para controlar un robot bípedo.

Referente al modelo cinemático y obtención de los parámetros DH del robot NAO se encuentra un estudio en [7].

El modelamiento y control de un robot se detalla en [8]. Para localizar el robot y efectuar una mejora en la caminada [9] utiliza un láser, cámara y un controlador de lazo cerrado. Mientras que [10] presenta una estrategia para caminar directamente de un punto a otro utilizando sensores ultrasónico comparada la referencia con la distancia de la pared.

Relativo a la percepción por sonar [11] realiza un estudio comparativo de la distancia con el sensor ultrasónico de NAO y otros sensores de profundidad.

Para detectar y describir objetos [12] expone los algoritmos y aplicaciones de la visión artificial. En [13] se detallan las métricas y análisis de la visión por computador. En [14] [15] [16], se orienta la visión por computador con Python y OpenCV, se describe el modelo de las cámaras pinhole, mapas de colores, morfología

matemática como la erosión, dilatación, opening, closing. Se presentan algoritmos para encontrar bordes, líneas por la transformada de Hough, regiones, extracción de características con SIFT, SURF, FAST, BRIEF, ORB, homografías, calibración de la cámara, procesamiento de video, seguimiento o tracking de objetos, Matching.

Uno de los proyectos europeos que trabajó con robots humanoides con habilidades visuales y auditivas en espacios poblados fue HUMAVIPS (HUManoids with Auditory and Visual abilities In Populated Spaces) [17]. Para Detección de objetivos y localización con el robot NAO, [18] presenta una propuesta para localizar y reconocer un objetivo con el sensor de visión. Las imágenes son segmentadas en base a un umbral, se calcula su centroide y se localiza por estructura geométrica el objeto.

Un landmark o punto de referencia es una característica del entorno fácilmente detectable por el robot. En [19] se utilizan landmark visuales no artificiales detectados mediante una cámara. Aplica funciones visuales dedicadas a la extracción de características y reconocimiento de landmark visuales, tal como figuras planas.

Un componente clave de un robot móvil es la habilidad de localizarse de manera exacta y construir un mapa del entorno de manera simultánea.

En [20] se formaliza y describe la robótica probabilística, mediante distribuciones de probabilidad de creencia o belief , filtros de bayes, filtros gaussianos como el Kalman, filtro de partículas. Todas aplicadas a los robots tanto en el movimiento, percepción, localización, mapping y SLAM Simultaneous Localization and Mapping.

Respecto a la localización, mapeo y navegación utilizando como fuente la visión artificial, [21] utiliza una sola cámara para detectar características visuales del piso. En [22] y [23] se propone la localización y mapeo de un robot utilizando características invariantes de la imagen como landmark en un entorno fijo. Estos landmark naturales son estimados realizando un matching o emparejamiento en base a las características de variación del punto de vista, aunque utiliza cámaras estereoscópicas y SIFT para el seguimiento de las características de los landmark. De forma similar [24] lo realiza con SURF. Mientras [25] ha utilizado localización por visión mediante marcadores visuales artificiales de colores y códigos. Con la posición detectada por los marcadores, se utiliza filtro de partículas para obtener la posición de un robot, pero de ruedas.

Utilizando Struct From Motion SFM para crear un mapa detallado para localizar el robot que será la entrada para SLAM se encuentra en [26] con una aplicación de modelado para el robot NAO. De similar forma en [17]. En [27] se realiza una propuesta para detectar distancia en una posición fija del robot reconstruyendo el entorno desde dos imágenes, efectuando que el robot se agache tome una imagen y se levante. Esta opción no es adecuada para un robot en movimiento.

Para planificación de la ruta y navegación se ha utilizado [25]. Referente a la calibración de la cámara lo presentado en [28] que realiza el proceso añadiendo en el pie del robot la tabla de ajedrez para obtener los datos intrínsecos.

Para la construcción del mapa mediante sonar, en [29] se utilizan técnicas perceptuales para identificar, localizar características del entorno , crear su mapa mediante el sonar. [30] utiliza occupancy grid para la percepción del robot y navegación. [10] propone una metodología para representar una representación del entorno utilizando el sonar y mapas de ocupación con la teoría bayesiana.

# 3. Desarrollo

## 3.1. Herramientas y plataformas utilizadas

En el desarrollo del proyecto se utiliza dos Robots reales NAO y lenguaje de programación, herramientas de desarrollo para codificación, frameworks de comunicación, librería de visión artificial, simuladores webots NAO para interacción del robot, comunicación, framework software que se describen a continuación:

**Lenguaje de programación:**

Python versión 2.7.13 de 32 bits. No se selecciona la versión 3 ni la versión de 64 bits por compatibilidad con el robot.

Python es un lenguaje interpretado, similar en estructura a C, C++ y java, pero con menos código para realizar una tarea. A percepción del autor más compacto y fácil de leer.

A pesar de que C o C++ son más eficientes en bajo nivel que Python y manipulación de memoria, se elige Python porque es un lenguaje moderno, tiene acceso y compatibilidad a librerías de visión y procesamiento científico.

**Librería de Visión artificial**

OpenCV (Open Source Computer Vision) tanto en la versión 2.4.13 y versión 3.2.0.7 con módulos contrib para utilización de algoritmos de detección y descripción por visión [31].

**IDE de Desarrollo:**

IDE PyCharm community edition 2016.3 para la codificación de Python y Opencv.

**SDK de Desarrollo para NAO**

Framework NAOqi de 32 bits (que se describe posteriormente).

Pynaoqi 2.1.4.13 para comunicación Python con el framework. Soporte opencv 2.4 y 3.2. con Python 2.7 de 32 bits solamente. NAOqi o imagen del robot real. Versión 2.1.4 (image Atom) utilizada para actualización del robot [32].

**Paquetes para computación científica con Python**

Distribución miniconda 4.3.2.1 con un entorno que contiene librerías y paquetes que pueden ser exportados para usar en diferentes máquinas [33].

SciPy 0.19.1 y Numpy 1.12.1. Librería Numerical Python para procesamiento científica y utilizado para las imágenes mediante un ndarray [34], Matplotlib 2.0.2 librería para realizar plot o gráficas con usabilidad similar a Matlab. Requiere de Numpy [35], pyQt 5.6 para la creación de la interfaz gráfica y para reproducción y grabación de video se utiliza ffmpeg 2.4.13.

**Software gráfico para programación y comunicación con el robot**

Se utiliza el software Choregraphe 2.1.413 de 32 bits para pruebas del robot y generación de rutinas de movimientos. El software lo provee la empresa creadora del robot. Permite trabajar con el robot real y algunas acciones con el robot simulado. A pesar de que se puede realizar un comportamiento del robot desde este programa por Diagrama, Línea de tiempo, Custom Box vía script de Python y también desde Python. El autor no lo utiliza por no poder aplicar comportamientos más avanzados directamente desde el lenguaje de programación.

**Software de simulación Webots**

Debido a políticas de la Universidad y al alto costo del robot. El acceso es limitado. Se utiliza el software comercial de simulación Webots versión Educativa 8.2.1 de 32 bits. A pesar de que existen versiones de 64 bits Por compatibilidad con el simulador y framework del robot se utilizan las versiones de 32 bits.

Los requerimientos mínimos son: Computador 2 GHz, 2 GB de RAM, Tarjeta de Video NVidea o AMD de mínimo 128 MB. OpenGL mínimo 2.1 [32].

### 3.2. Equipo informático y de comunicación

Se han utilizado dos computadores para pruebas:

PC desarrollo: con Windows 10, Intel Core i7 2 GHz de 64 bits. 8 GB de RAM. tarjeta NVIDEA, 1 Terabyte de Disco Duro.

PC laboratorio: Pc Windows 8, Intel Core i3. 1.7GHz de 64 bits, 4 GB RAM. 600GB de disco duro.

Además se ha utilizado un Router Linksys E900 de hasta 300 Mbps, fast Ethernet 10/100 Mbps. 802.11g.

Respecto a la parte documental se utiliza Mendeley para manejo bibliográfico.

### 3.2.1. Robot NAO, breve descripción de sensores y actuadores

El robot NAO tiene las versiones NAO-H25, NAO-H21, NAO-T14 (parte superior), NAO-T2 (torso y cabeza). El robot utilizado es el NAO Versión 5 con 25 grados de libertad. Es creado por la empresa Aldebaran del grupo SoftBank Robotic[36]. Tiene 27.4 cm de alto, 31.1 cm de profundidad con los brazos extendidos y 27.5 cm de ancho y un peso de 5.3 Kg. Una presentación con audio y movimientos del robot NAO se encuentra en el video A.1.1.

Los sensores del robot están formados por dos cámaras CMOS tipo Webcam HD de máximo 1280x960 pixeles a 30 fps, dos sensores ultrasónicos, sensor inercial, infrarojos, sensores resistivos de fuerza FSR. Tiene micrófonos, altavoces (parlantes) y LED para obtener retroalimentación del robot.

Existen cámaras monoculares, estéreo y omnidireccional, panorámica, RGB-D. Las cámaras que tiene el robot nao son de tipo monocular. Estas se encuentran en la cabeza. Una con vista hacia al frente y otra hacia el piso. La percepción mediante visión se detalla en las siguientes unidades. Las características de la cámara se describen en el anexo A.10.

El motheboard es un ATOM Z530 1.6 GHz CPU con 1 GB RAM y para acceso externo 2 GB Flash memory con 8 GB Micro SDHC. El Robot NAO es compatible con Windows, Linux y Mac Os. Se puede programar de manera sencilla con Choregraphe y de forma avanzada con los lenguajes Python, C++, Java y Matlab desde un computador externo al robot. La programación dentro el robot es con Python y C++. Además, se puede manejar por el software Choregraphe. La batería tiene una autonomía de 60 minutos con uso activo y de 45 minutos con sensores de visión y ultrasonido. Para uso normal es de 90 minutos (1 hora y media). Con capacidad de 48.6 Wh. La conectividad para un PC y el robot para configuración y o ejecución es con un puerto ethernet 1×RJ45 - 10/100/1000 base T y por WIFI con especificación IEEE 802.11 a/b/g/n. Hay además un puerto USB para adaptar otros dispositivos.

El Robot NAO en la 12 campeonato internacional RoboCup (2008) se ha utilizado como robot oficial del evento.

### 3.2.2. Robot NAO vs simulador Webots

Un robot simulado es el que no tiene existencia en la vida real y se ejecuta en el computador efectuando similares actividades que el robot real. Su función es probar

el código antes de ejecutarlo en un robot físico y observar el comportamiento. Se pueden utilizar tres formas con el robot simulado:

- Choregraphe para uso local NAOqi (software de la empresa creadora del robot).
- NAOqi binary (independiente de Choregraphe).
- Mundo Virtual (virtual world) tal como webots para NAO.

El software Webots simula la cámara 1 y 2, unidad inercial, sonar, masa, colisiones. Además de otros sensores no utilizados en el proyecto. El acceso al audio en la simulación no está disponible, así como otros elementos físicos tales como batería, disco duro real del robot y otros.

El simulador recreará un entorno muy parecido al real, donde se encuentra el robot con objetos, pisos, elementos. Posteriormente se efectúa la comunicación con el robot real el cual tiene similar distribución que el entorno físico simulado.

#### 3.2.2.1. Transferencia al robot real

La metodología utilizada en el proyecto es que se programa en Python con OpenCv tal como se fuera con el robot real. Se utiliza el framework de comunicación NAOqi entre el robot y el lenguaje de programación. En sí la comunicación es Python con NAOqi. El código generado es en la mayoría de casos el mismo que se utiliza en el robot real. Esto se logra efectuando envío de comandos desde el computador al robot mediante una dirección ip y puerto del robot.

### 3.3. Entorno Real de trabajo y simulación del entorno real

Se utilizan tres entornos para pruebas en lugares internos y controlados.

Entornos reales de trabajo ERx y Entornos virtuales EVx, donde x=1…3

1. ER1. Laboratorio de Fabricación Flexible.
2. ER2. Laboratorio de automatización.
3. ER3. Pasillo del laboratorio.

Cada entorno real, tienen su par de Entornos Virtuales.

En las siguientes figuras se muestran los diseños de ER1 y EV1. Los diseños de ER2 y ER3 se encuentran en los anexos en el video A.1.2

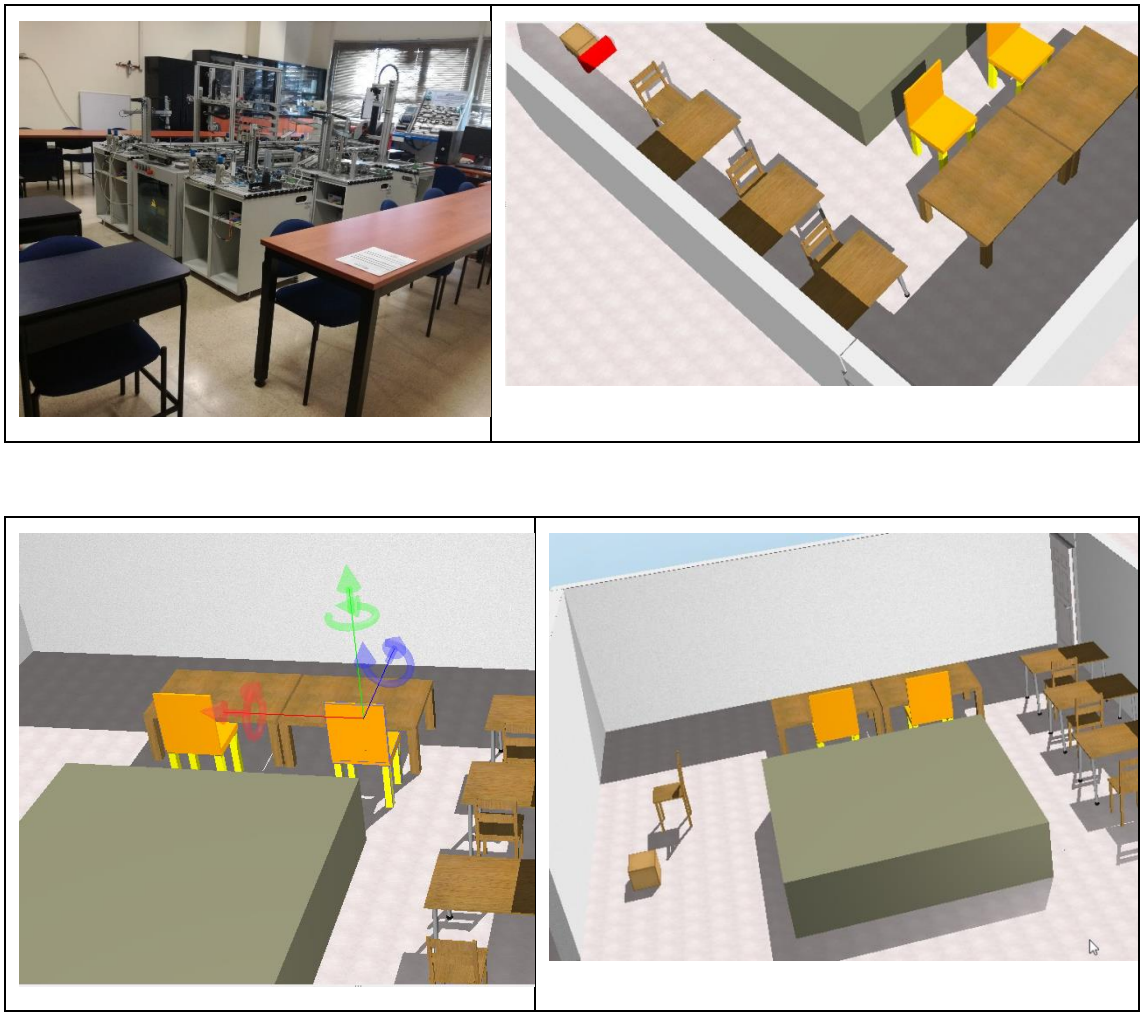


Figura 1. Entorno 1. EV1 Laboratorio de Fabricación Flexible.

Figura 2. Entorno de desarrollo ER1 como ambiente de pruebas.

### 3.4. Propuesta del diseño del proyecto

Se diagrama la propuesta general del proyecto.

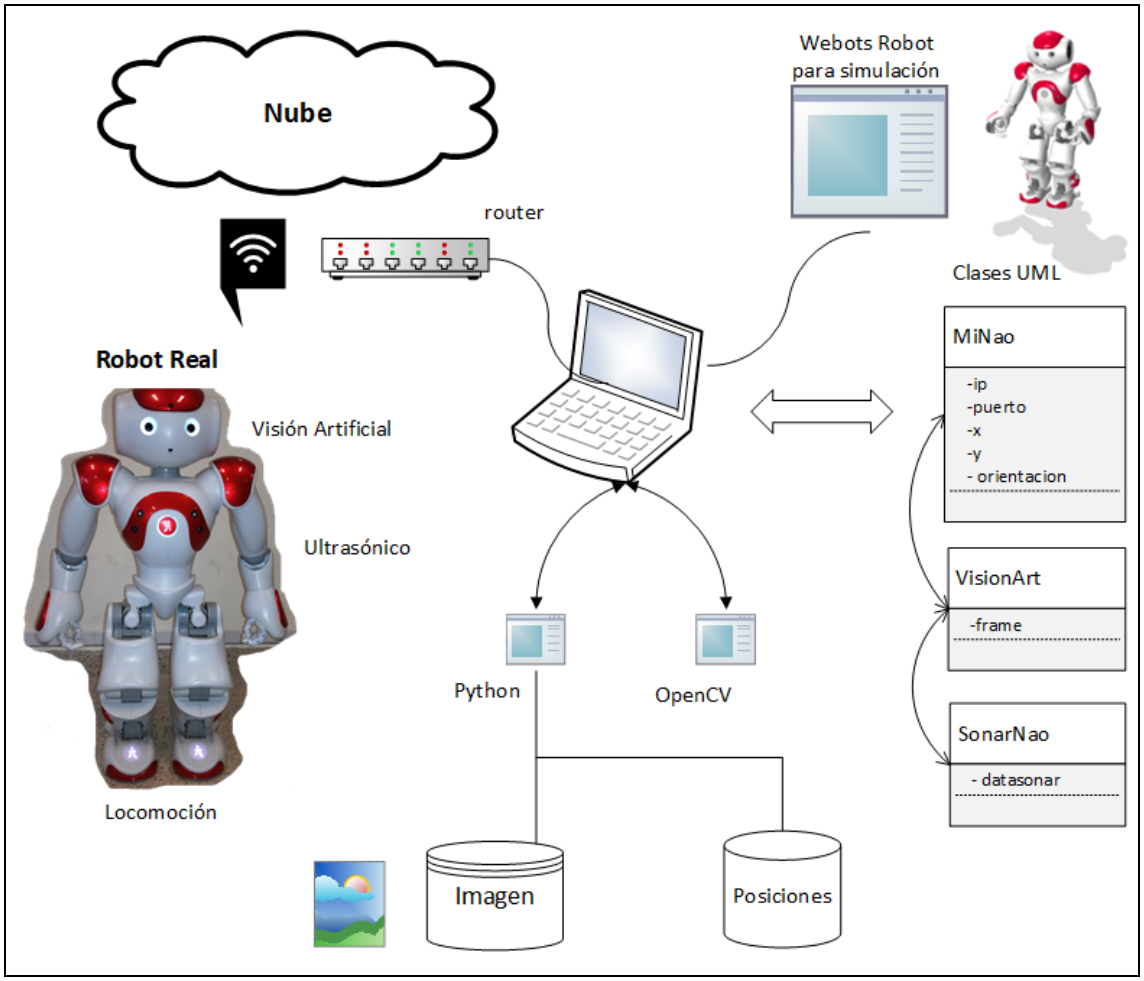



Figura 3. propuesta general del proyecto.

Se describe la propuesta del diseño de la localización y actualización de la posición.

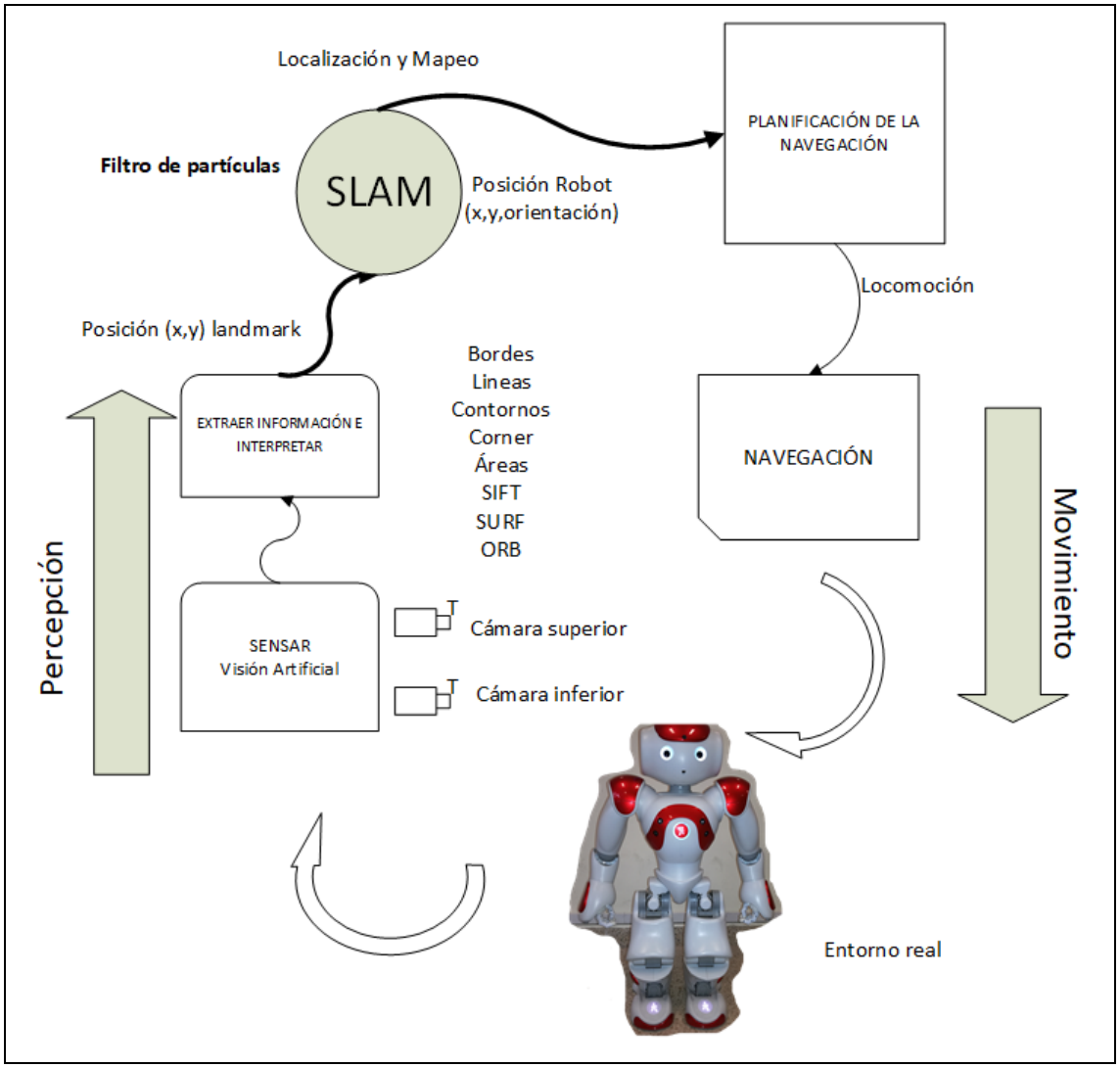



Figura 4. Propuesta del diseño de la localización y actualización de la posición.

Propuesta del diseño integral de percepción, descripción, localización y navegación.

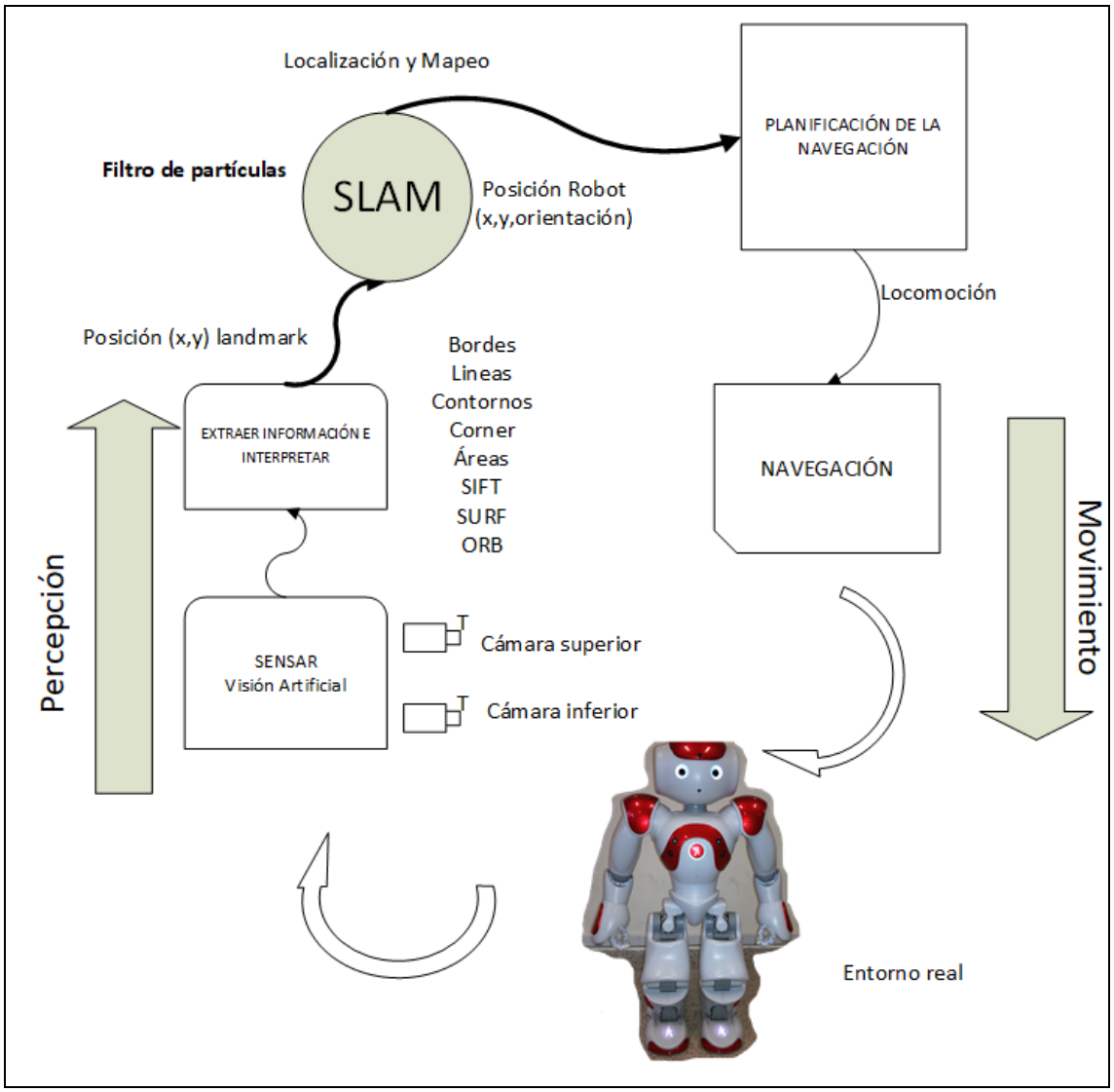



Figura 5. Diseño de la solución: percepción, descripción, localización, navegación.

### 3.5. Comunicación con el Robot. Framework NAOqi

El framework NAOqi, es el nombre del software principal que se ejecuta en el robot para controlarlo. Está formado por el Kit de Desarrollo de Software con el cual se accede a la API para programar y comunicarse con el robot[32]. El framework de programación se ejecuta bajo la distribución de OpenNAO, responde a las necesidades de la robótica entre ellas: paralelismo, recursos, sincronización y eventos. Este framework puede utilizarse tanto en un robot real como en un robot simulado. Se ha establecido que en los dos robots este la versión del NAOqi 2.14 así como en el robot simulado de Webots (a partir de la versión 8). Esto para evitar problemas de compatibilidad entre el robot real y simulado.

El framework tiene como características:

- Es cross-platform. Por lo cual es posible programar en Windows, Linux o Mac.
- Es cross-language, con un API idéntico tanto para C++ como Python.

**Bróker y proxy.** El NAOqi que se ejecuta en el robot se denomina bróker o agente-corredor. Este bróker provee acceso a los módulos del framework a través de la red y ejecuta los métodos disponibles en el método. El bróker es un objeto que provee acceso al directorio de servicio y a la red. El bróker realiza las tareas de forma transparente. En sí el bróker es un ejecutable y servidor que está atento a escuchar los comandos remotos en base a una dirección ip y un puerto. Tal como la estrategia establecida en el diseño del proyecto.

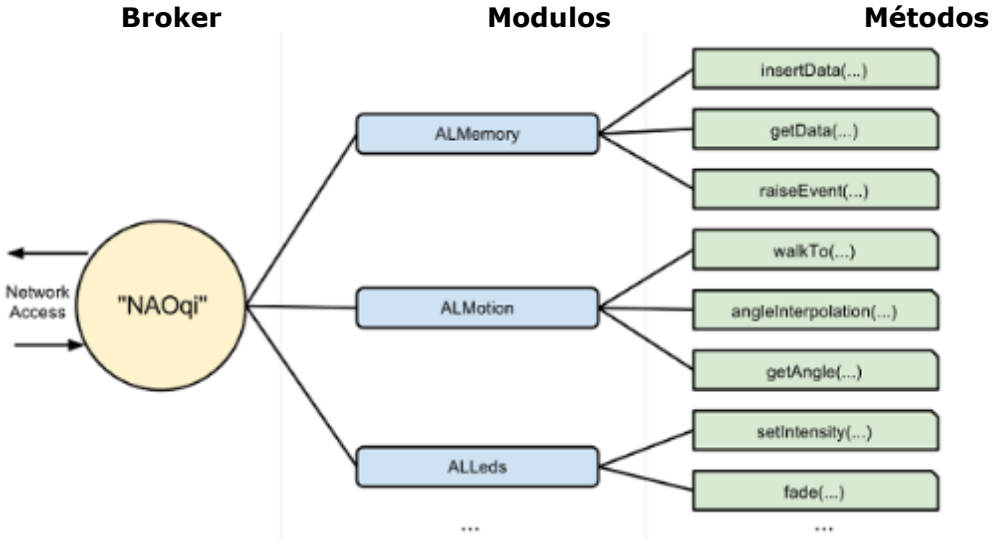



Figura 6. Proceso de comunicación Framework NAOqi [32].

Un proxy es un objeto que se comporta como el módulo que desea utilizar. Es decir, llama al módulo y se comporta como tal. Por ejemplo, si el proxy llama al módulo ALMotion que es para control del movimiento, puede utilizar todos los métodos relativos a ese módulo.

En el framework NAOqi al proxy se denomina ALProxy y es el objeto principal. En la Tabla 1 se describe a continuación los módulos utilizados en la investigación.

Tabla 1. Módulos de NAOqi utilizados en la investigación.

| Módulo | Operación a realizar |
|---|---|
| ALMotion | Control del movimiento caminar, parar, girar |
| ALRobotPosture | Control de la postura del robot |
| ALVideoDevice | Acceso a la cámara |
| ALSonar | Acceso al sonar |
| ALMemory | Acceso a los datos del sensor inercial y a los datos del sonar |
| ALTextToSpeech | Acceso para que el robot hable |

En la Tabla 2 se muestra una llamada a los módulos ALMotion y ALRobotPosture, ambos mediante ALProxy. Con la referencia a la variable se pueden utilizar los métodos de los módulos correspondientes. Para la comunicación se requiere la dirección IP y puerto del robot. Se utiliza la clase creada para el proyecto MiNao.

Tabla 2. Instrucciones básicas para el movimiento del robot.

```
motionProxy = ALProxy("ALMotion", ip, puerto)
postureProxy = ALProxy("ALRobotPosture", ip, puerto)
tts = ALProxy("ALTextToSpeech", ip, puerto)

motionProxy.wakeUp() # despertar al robot
postureProxy.goToPosture("StandInit", 1)#ir a una posición de inicio del robot

motionProxy.moveTo(0.5,0,0) # mover a una posición de 0.5 metros en X

tts.setLanguage("Spanish")
tts.post.say("Hola UCM UNED")

motionProxy.rest()   # descansar
```

Se indica que framework oficial es NAOqi, pero varios investigadores han efectuado otros frameworks para comunicarse, entre ellos uno utilizando ROS (Robot Operating System) o sistema operativo del Robot.

### 3.6. Control externo del robot

Como se ha mencionado, el framework NAOqi requiere una dirección ip y un puerto para comunicarse entre el robot y el programa origen. Esta configuración permite

trabajar con el Robot Real y Robot virtual sin muchos cambios. Es así que se puede utilizar Python con el robot virtual y el mismo código de Python con el robot real.

Esto se describe en la siguiente figura. La comunicación es por wifi Como se describe mediante choregraphe o software como Python, la comunicación es wifi.

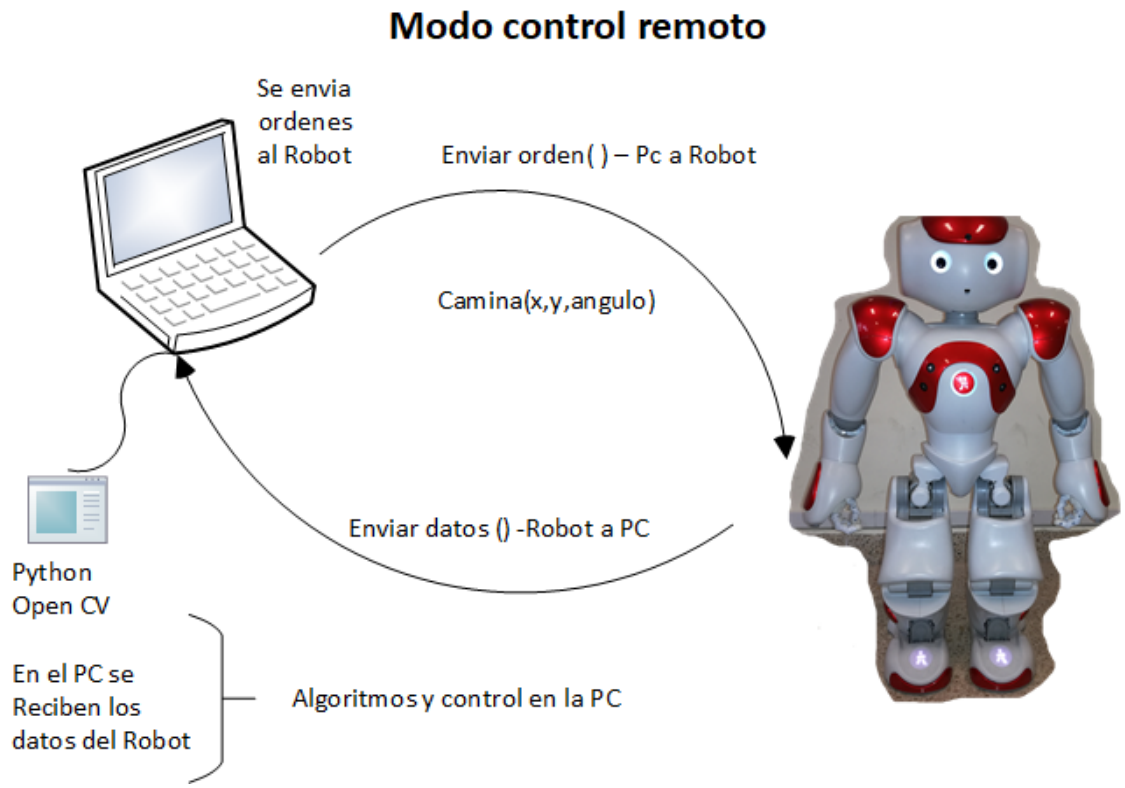



Figura 7. Modo de control del robot.

Cada robot tiene una dirección ip y un puerto. Se utilizan dos robots para pruebas reales. Se detalla las direcciones ip con sus puertos.

Tabla 3. Direcciones IP y puerto de los robots Real y virtual.

| Robot | Ip | Puerto |
|---|---|---|
| NAO1.Obarcia rojo | 192.168.0.100 | 9559 |
| NAO2.Nao azul | 192.168.0.103 | 9559 |
| Robot simulado | 127.0.0.1 | 9559 |

Estas direcciones IP son dinámicas y locales. Se puede configurar también para que tenga direcciones ip públicas. Una vez que se establece la ip y puerto se procede a comunicar mediante NAOqi.

```
[I] 8092 qimessaging.session: Session listener created on tcp://0.0.0.0:0
[I] 8092 qimessaging.transportserver: TransportServer will listen on: tcp://192.168.1.146:50614
[I] 8092 qimessaging.transportserver: TransportServer will listen on: tcp://127.0.0.1:50614
```

### 3.7. Definición de ejes de referencia

Un eje de coordenadas o frame se representa como {F}. Utilizando los ejes x, y, z, respecto al frame se denominan $X_F$, $Y_F$, $Z_F$. Una pose es la posición y orientación de frame de coordenadas. Un frame puede estar desplazado y rotado respecto a otro frame. El primer frame de coordenadas es el de referencia o world (mundo) [37].

Un punto del robot en el espacio tridimensional 3D (pose world), requiere de 6 coordenadas en el frame, 3 para la orientación y 3 para la posición. Para el espacio bidimensional se requiere de 3 coordenadas en el frame, dos para la posición x , y y una para la orientación. Se efectúa la siguiente restricción. A pesar de que el robot en su movimiento las piernas se levantan en el eje 3D. Por motivos prácticos se considera el espacio del plano de 2D para cada desplazamiento en posición y orientación de un punto específico del robot. Tal como lo muestra la Figura 8.

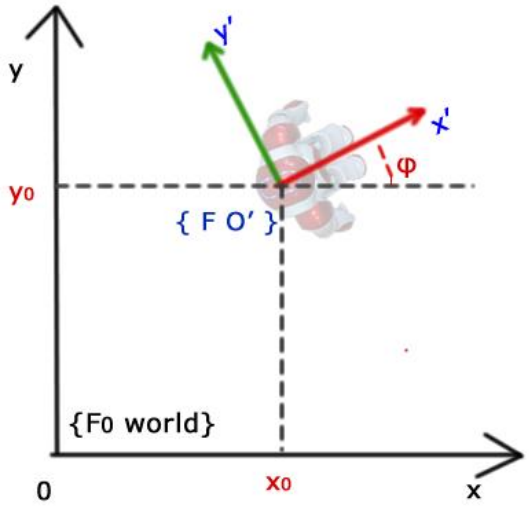



Figura 8. Pose del robot en el Frame 0, coodenadas referencia inicial o mundo (world).

En la figura se observa el Robot en el Frame 0 o coordenadas de referencia inicial o mundo (world) con puntos x0, y0 y orientación φ. Cada vez que el robot se mueve el pasa a ser un nuevo Frame {F O'}, toda nueva observación que realice el robot se lo puede efectuar desde su posición que será una observación relativa al robot. Posteriormente se puede obtener la posición absoluta relacionada al Frame 0 aplicando operaciones denominadas composición de transformaciones homogéneas.

Respecto a la definición de los ejes y giros se detalla a continuación. Eje x, hacia delante del robot. Eje Y hacia la izquierda. Eje Z es vertical. Esta orientación en base

a la convención ENU (Este - X, Norte – Y Up – arriba-Z). Los ángulos de giros son Roll, Pitch, Yaw. El giro roll es respecto al eje X. El giro pitch es referente a Y. El giro Yaw es referente al Eje Z.

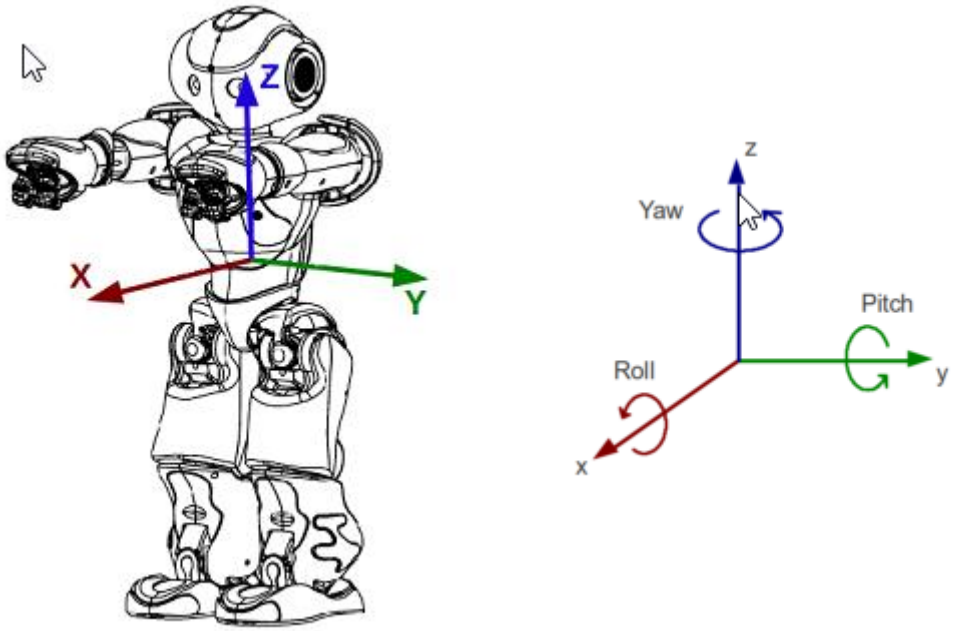



Figura 9. Definición de los ejes de referencia.

**Matriz de transformación homogénea**

Es una matriz T que representa la transformación de un vector de coordenadas de un sistema a otro. Tiene dimensión de 4x4.

$$T = \begin{bmatrix} R_{3x3} & p_{3x1} \\ f_{1x3} & w_{1x1} \end{bmatrix} = \begin{bmatrix} Rotación & Traslación \\ Perspectiva & Escalado \end{bmatrix} \quad (1)$$

La matriz homogénea T, está compuesta por cuatro submatrices de diferentes tamaños. En robótica se utiliza solo $R_{3x3}$ y $p_{3x1}$ considerando los otros elementos nulos.

$$T = \begin{bmatrix} R_{3x3} & p_{3x1} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} Rotación & Traslación \\ 0 & 1 \end{bmatrix} \quad (2)$$

Se indica que la Matriz de rotación R es ortonormal $R^{-1}=R^{T}$.

En muchos problemas de robótica como es obtener la posición en base a otras posiciones relativas, así como en el cálculo de parámetros Denavit-Hartenverg se utilizan la composición de matrices homogéneas. La notación para una transformación desde A hacia B, o B relativo a es ${}^{A}T_{B}$ .

La obtención cuando hay composición de varias transformaciones, se puede representar con un gráfico de transformación o direct grahp, para obtener la relación entre el elemento A y B, con lo cual se requiere ir multiplicando las matrices de transformación relativas al arco del gráfico, desde el punto A y buscar los arcos que lleguen a B. En el caso de que la relación sea en sentido contrapuesto a las flechas debe utilizarse una matriz inversa o cambiando el orden del superscript y subscript en la transformación. Toda relación se puede obtener del gráfico.

En la Figura 10 se encuentran los elementos F, B y R respecto al Frame 0 o mundo. Estos a su vez tiene posiciones relativas a otros elementos como B y C. Se puede obtener las posiciones con respecto a otros Frames utilizando composición de matrices de transformación.

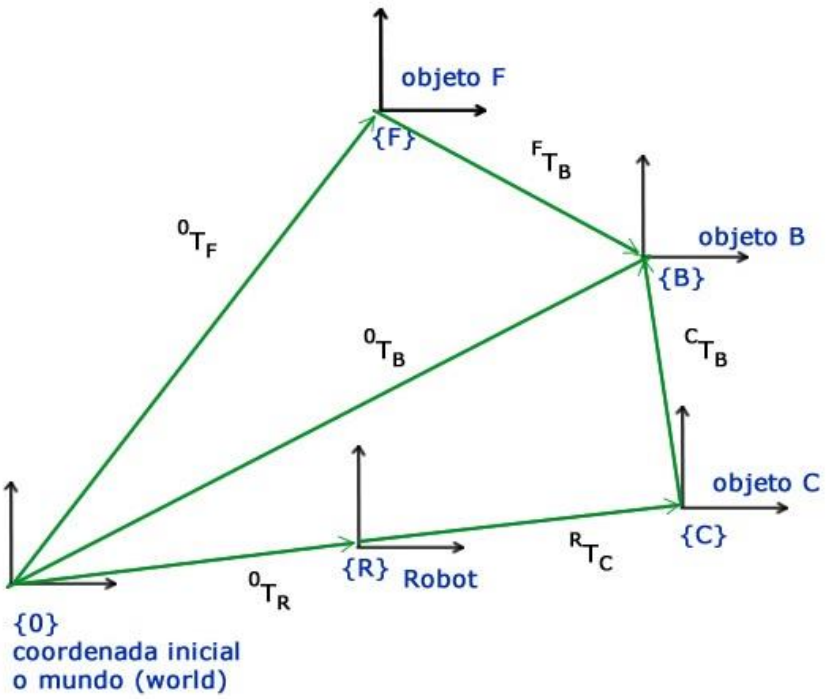



Figura 10. Posiciones absolutas a F0 y relativas a otros objetos.

Se suele utilizar un gráfico de transformación o gráfico directo para representar el mismo comportamiento. Por ejemplo, la ruta desde el frame O mundo de F a R, se realiza por una multiplicación de Matrices Homogéneas. Se debe considerar que las operaciones no son conmutativas.

$$ {}^{F}T_{R} = \left( {}^{O}T_{F} \right)^{-1} {}^{O}T_{R} \tag{3} $$

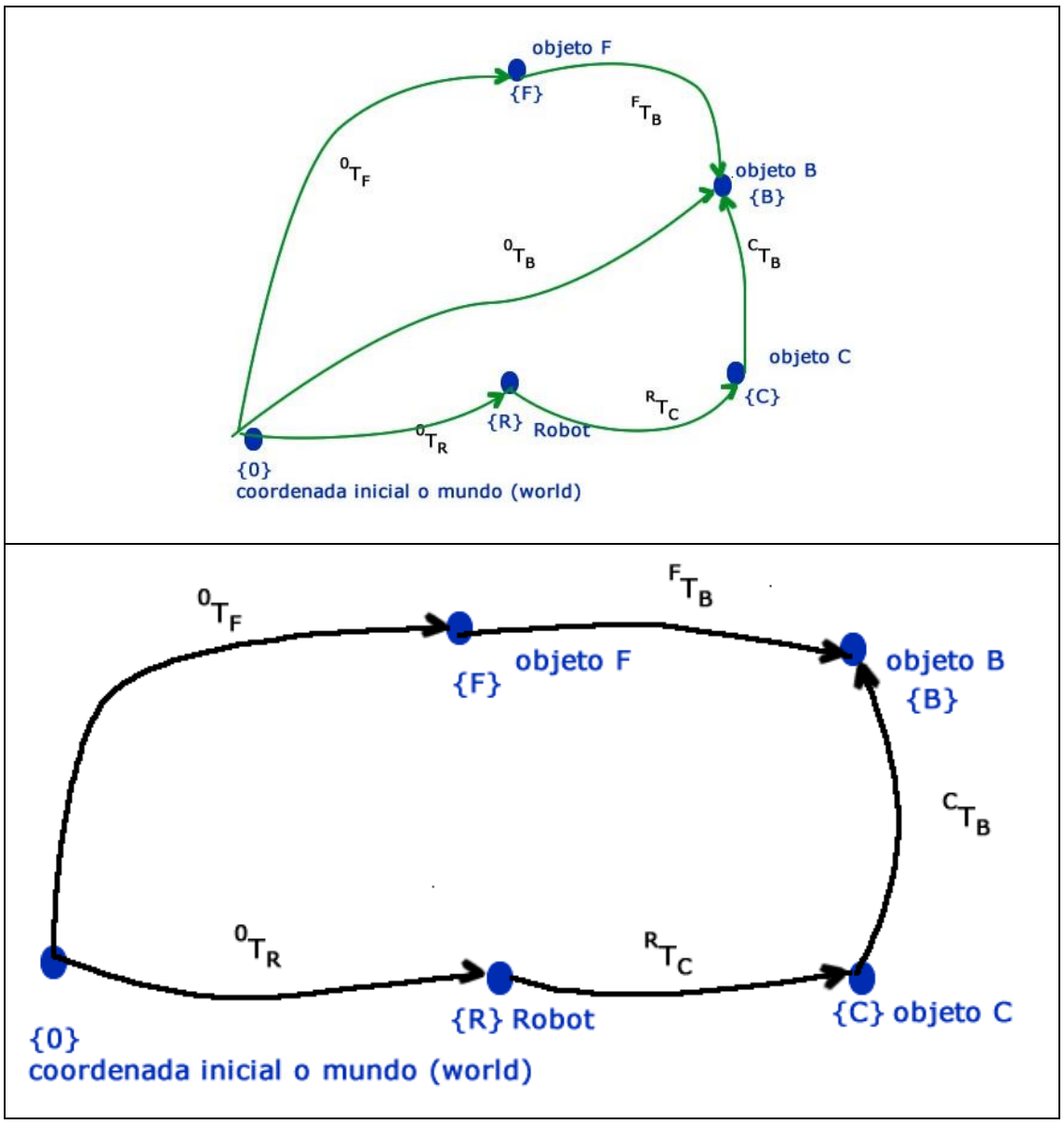



Figura 11. Gráfico de transformación o directo [37][3].

### 3.7.1. Coordenadas de los frames respecto a la cámara

Se requiere las siguientes coordenadas para identificación:

Coordenadas del frame mundo $S_o$ como $[X,Y,Z]^T$. Posteriormente el frame de la cámara $S_k$ como $\left[{}^kX,\ {}^kY,\ {}^kZ\right]^T$. El eje del plano al frame $S_c$ como $[\ {}^cx,\ {}^cy]^T$. Por último, el sensor de coordenadas $S_s$ como $[\ {}^sx,\ {}^sy]^T$. La transformación requerida para efectuar el mapa utilizando matrices homogéneas es:

$$\begin{bmatrix} {}^sx \\ {}^sy \\ 1 \end{bmatrix} = {}^sT_c \quad {}^cT_k \quad {}^kH_0 \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{4}$$

En el frame sensor = imagen del plano al sensor *cámara a la imagen *mundo a la cámara * frame de 3D en el mundo. Tal como lo expresa la Figura 12:

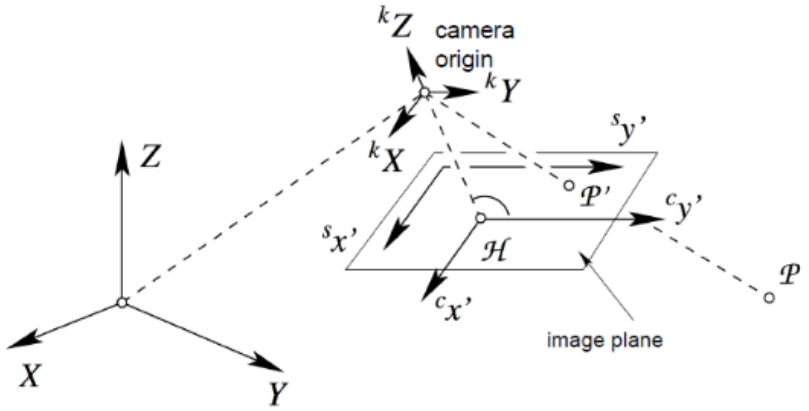



Figura 12. Coordenadas de los frames respecto a la cámara y el mundo [38].

### 3.8. Percepción del robot

Una de las más importantes tareas de los robots autónomos es adquirir conocimiento del entorno. Los sensores se los clasifica en propioceptivos PC, exteroceptivos EC y pasivos/activos [4].

Los propioceptivos son sensores que miden valores internos como el ángulo de la articulación, velocidad del motor y voltaje de la batería, intensidad de luz. Los exteroceptivos son sensores que adquieren información del entorno. Con estos datos medidos el robot interpreta lo que ocurre en el entorno. Los sensores pasivos miden

energía del entorno ingresada por el sensor tal como medidor de temperatura, micrófonos y cámaras CMOS. Los sensores activos emiten energía al entorno, y mide la reacción que provoca. En la Tabla 4 se muestran algunos sensores que tiene el robot NAO.

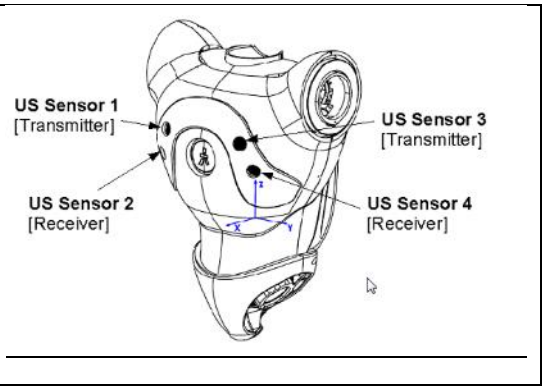
Tabla 4 Clasificación de sensores utilizados en robótica móvil NAO.

| Tipo de uso | Sensor | PC o EC | P Pasivo o A Activo |
|---|---|---|---|
| Orientación del robot en relación a un eje de referencia | Giroscopio | PC Propioceptivo | P Pasivo |
| Sensor de aceleración | Acelerómetro | PC Propioceptivo | P Pasivo |
| Detector de rango por tiempo de vuelo (time of flight TOF) | Sensor ultrasónico | EC exteroceptivos | A Activo |
| Sensor de visión Distancia visual, análisis de una imagen, segmentación, reconocimiento de objetos | Cámara CMOS | EC exteroceptivos | P Pasivo |

De los sensores indicados se utiliza solo la cámara para visión artificial y el sensor ultrasónico. Se descarta el acelerómetro y el giroscopio debido a que se desea medir el ángulo del eje Z para corrección en el desplazamiento, pero no está implementado.

### 3.8.1. Sensores ultrasónicos

El sensor ultrasónico es un sensor EC exteroceptivos y Activo que está formado por 2 emisores y 2 receptores, cuya meta es estimar la distancia del robot a un objeto en entorno y así evitar obstáculos. Para detección de distancias se obtiene mediante la obtención del tiempo de (time of flight TOF). Las características del sensor son:

| | |
|---|---|
| Frecuencia: 40kHz<br>Resolución: 1cm-4cm<br>Cono Efectivo: 60°<br>Rango de detección: 0.20 m - 0.80 m<br>Sonar lee valores cada 10 ms<br><br>Cuando el robot esta inferior de los 20 cm, no se detecta distancia, pero se identifica que hay un objeto |  |

El sensor ultrasónico tiene su principio en que un transmiso envía ondas de presión (ultrasónicas) y medir el tiempo que toma en reflejarse y volver a el receptor.

La distancia del objeto en base a la reflexión, se obtiene en base a la velocidad del sonido c y el Time Of Flight TOF o tiempo de vuelo [4]:

$$d = \frac{ct}{2} \tag{5}$$

La velocidad del sonido en el aire es:

$$c = \sqrt{\gamma RT} \tag{6}$$

Donde $\gamma$ es la relación de calores específicos. R, es la constante del gas. T es la temperatura en grados Kelvin. En presión estándar de 20° C, la velocidad del sonido es $c = 343\ m/s$.

Para conectar con el sensor de visión se requiere el proxy ALSonar. El robot tiene 2 sensores, el izquierdo y derecho. Se procede a obtener un muestreo del promedio de 3 medidas. Se utiliza la clase creada para el proyecto SonarNao.

```
sonarProxy = ALProxy("ALSonar", ip, puerto)
sonarProxy.subscribe("sonarmiupsapp") #es el nombre con el cual se identifica al sonar
# promedio de tres medidas:
        izq= memoryProxy.getData("Device/SubDeviceList/US/Left/Sensor/Value")
        der= memoryProxy.getData("Device/SubDeviceList/US/Right/Sensor/Value")
```

### 3.8.2. Sensor de Visión. Cámara monocular

Es un sensor EC exteroceptivos, pasivo. El utilizado en el robot NAO está formado por dos cámaras monoculares con un formato óptico de 1/6 inch con rango de 60.97° grados (1.06 radianes), situadas en la parte superior e inferior de la cabeza del robot. Por su ubicación no se pueden utilizar para visión estereoscópica.

El campo de visión FOV (Field Of View) es el ángulo lineal o área observable en un instante dado. En la Figura 13 se muestra la localización de la cámara superior e inferior, sus ángulos de campo de visión.

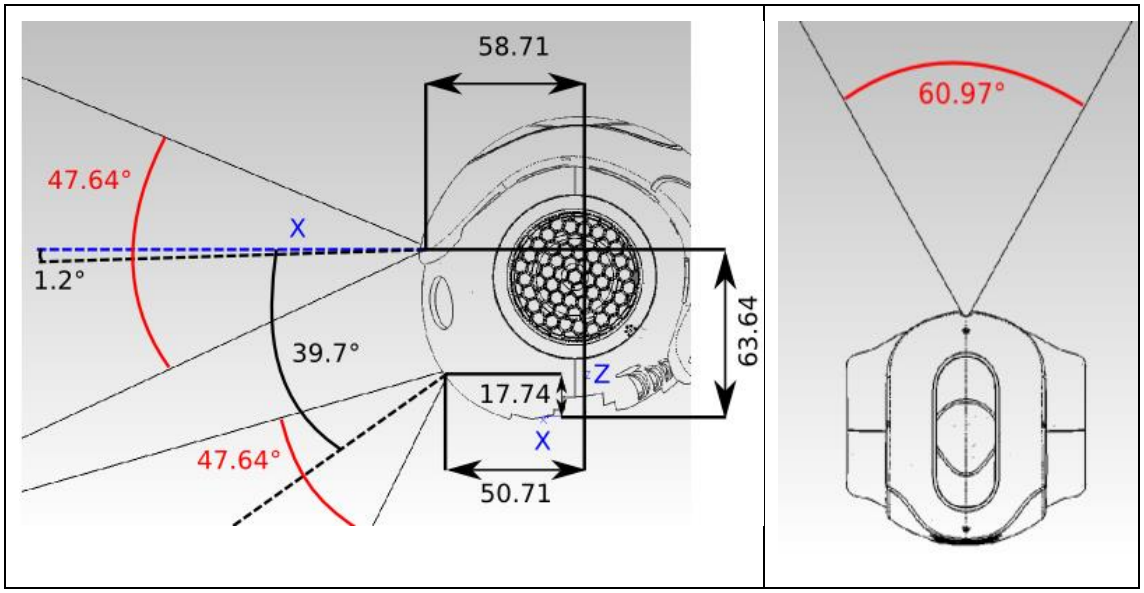



Figura 13. Localización de las cámaras superior e inferior del robot NAO.

A pesar de que la resolución máxima de ambas cámaras es de 1280x960 a 30 fps (frames per second) y debido a la limitación de la red por la transmisión de la imagen entre el computador y el robot, se establece el siguiente diseño para la captura de imágenes del robot:

Tabla 5. Diseño de la resolución, frame y modo para captura de imágenes.

| Resolución | FPS | Modo |
|---|---|---|
| 320 x 240 pixeles | 20 | QVGA |

El modo QVGA soporta a modo local 30 fps, a 100Mb Ethernet 30 fps y a Wifig a 11 fps. El modelo de color original del robot es YUV422 que se procede a transformar a BGR Blue Green Red para ser procesado en OpenCv. Las características técnicas de la cámara se detallan en el anexo A.10. La posición de las cámaras con respecto al punto de inicio alineado al torso del robot es:

| **Cámara** | **Posición Desplazamiento en x** | **Posición Desplazamiento en z** |
|---|---|---|
| Superior | 0.058 m | 0.06364 m |
| Inferior | 0.05071 | 0.017 m |

Figura 14. Posición de la cámara respecto al centro del robot.

Respecto al campo de visión FOVh: horizontal es 60.97° y el FOVv: vertical es 47.64°

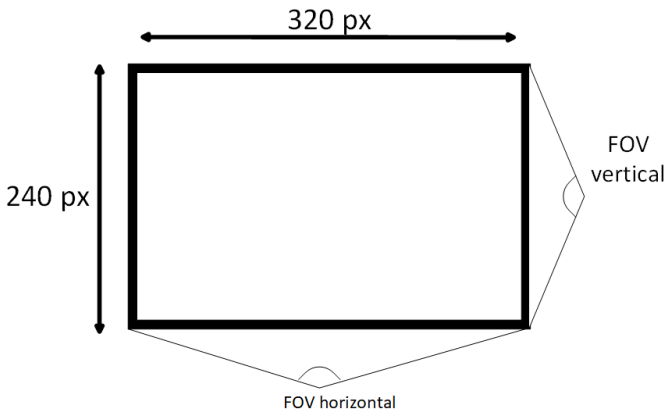



Figura 15. Campo de visión horizontal y vertical.

Para establecer el sensor de visión se utiliza la clase VisionArt con el proxy ALVideoDevice. Se establecen los valores requeridos de la imagen según lo diseñado.

```
videoDevice = ALProxy('ALVideoDevice', ip, puerto)
#parametros de cámara
# tipo 0 es cámara superior   . tipo 1 es cámara inferior
AL_kTopCamera = tipo/  AL_kQVGA = 1  # 320x240/  fps = 20
       ancho = 320 /      alto = 240
 captureDevice = videoDevice.subscribeCamera("naocamara", parámetros)
# al finalizar de obtener los datos dejar la referencia
videoDevice.unsubscribe("naocamara")
```

Para obtener los datos de la cámara el traspaso no es directo. Primero debe estar conectada la cámara. Posteriormente se extrae cáda imagen y se transforma al modo BGR (Blue, Green, Red) tratados en Python. Este mapa de color se explica con detalle más adelante.

```
# obtener parámetros de la cámara
result = videoDevice.getImageRemote(captureDevice)
# se extrae la imagen de result .   imagen= result[índice=6]
#obtener ancho, alto
Recorrer cada imagen , cambiando el orden de pixeles.
#Los pixeles  X se transforma a Y.  los pixeles Y se ubican en X

#El resultado es la imagen para procesar
```

En la Figura 16, se muestra el proceso para la percepción del robot.

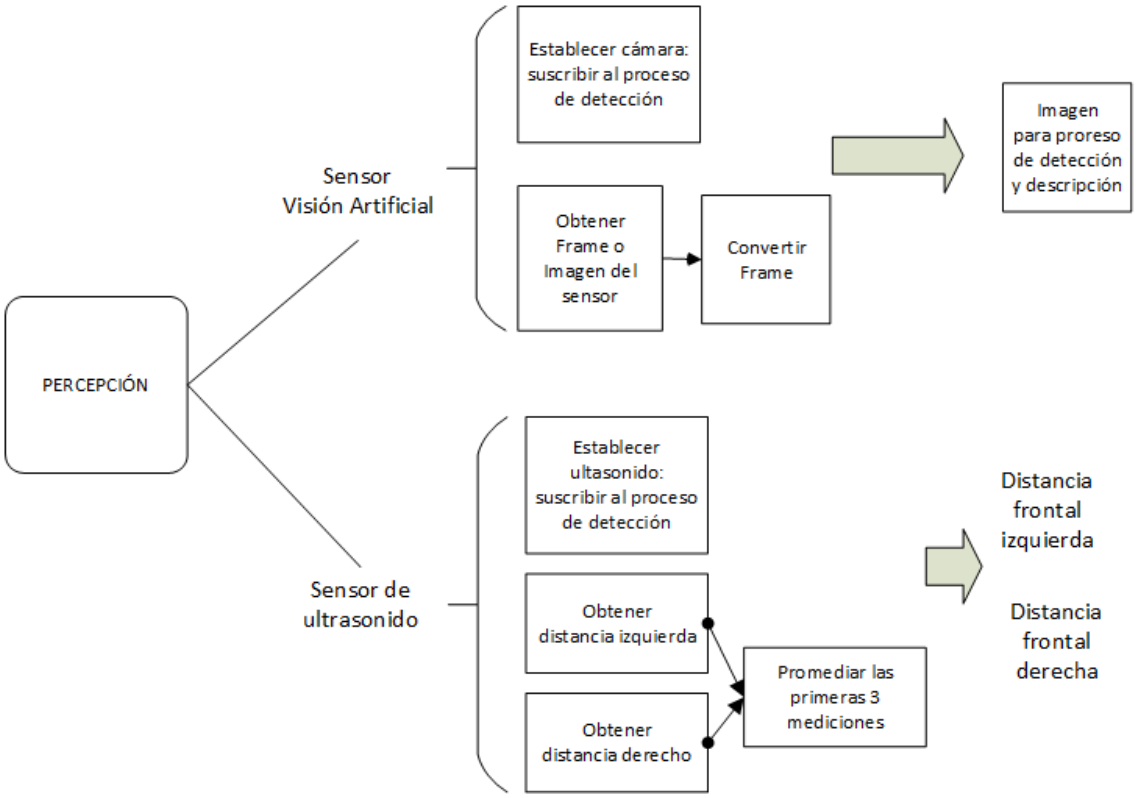



Figura 16. Proceso de percepción del robot.

### 3.9. Cinemática del robot móvil

Se requiere comprender el comportamiento mecánico del robot para diseñar y/o efectuar tareas y entender el control. Un entorno o espacio de trabajo de un robot móvil es importante debido a que define el rango de posibles poses que el robot móvil puede lograr. El robot móvil debe moverse respecto a este entorno.

La cinemática es el estudio del movimiento de sistemas mecánicos sin considerar las fuerzas que originan el movimiento, por lo cual no tiene ecuaciones diferenciales como en el caso de la dinámica [1]. La cinemática es la base del control del robot. En el diseño de un robot bípedo en el cual se debe caminar, debido a las dos piernas hay problemas de estabilidad.

#### 3.9.1. Obtención de datos cinemáticos

Un robot está formado de manera mecánica por elementos denominados eslabones o links, que están unidos mediante articulaciones o joints [8]. Las articulaciones proveen un movimiento entre cada dos eslabones contiguos. Un Grado de libertad (DOF Degree of Freedom) es cada uno de los movimientos independientes que cada

articulación puede realizar referente a la articulación anterior [3]. El robot NAO sus articulaciones son de rotación.

Para la obtención de datos cinemáticos se debe conocer que el robot está formado por: eslabones (links), articulaciones (joints), cadenas (chains), masa. Tiene 15 motores de cuatro tipos de acuerdo a la articulación, el torque nominal es desde los 4.9mN/m a 16.1 mN/m. Los detalles de las características se mencionan en los anexos.

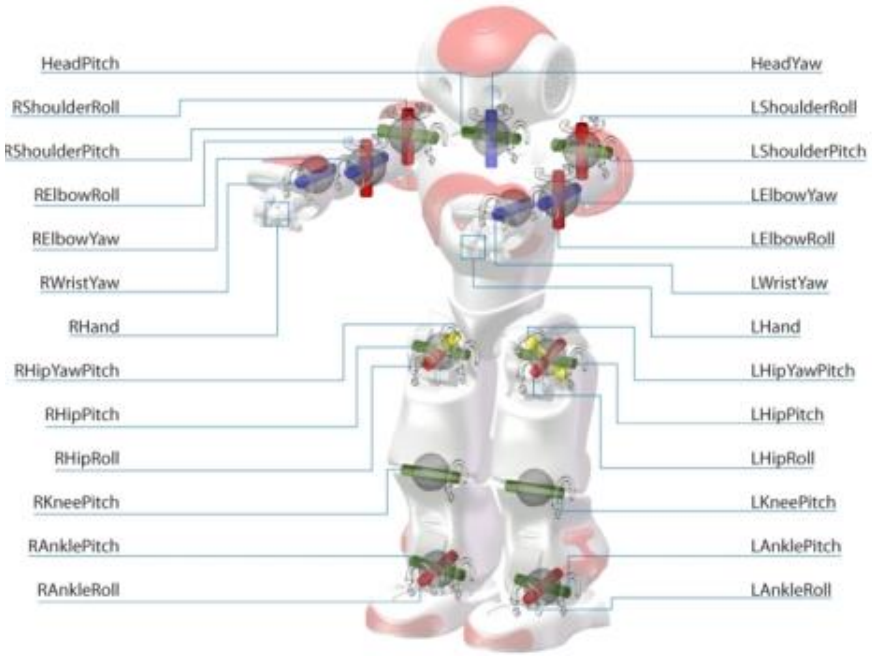



Figura 17. 25 Articulaciones Joints del Robot [39].

Es importante determinar para el control del robot, obtener las características o información del robot real, los sensores disponibles, las articulaciones y sus nombres, así como los valores o rango de los mismos:

```
Model Type :  naoH25
Head Version :  VERSION_50
Body Version :  VERSION_50
Laser :  False
Legs :  True
Arms :  True
Extended Arms :  True
Hands :  True
Arm Version :  VERSION_50
Number of Legs :  2
Number of Arms :  2
Number of Hands :  2
```

El robot NAO tiene configurado comportamientos para caminar hacia adelante, atrás, hacia la izquierda o izquierda, así como una cantidad de grados, que permite caminar en círculos. Para utilizar alguna articulación, cadena o partes del cuerpo se debe conocer sus identificadores tanto en: Nombre de la articulación, Grupo de articulaciones y partes del cuerpo. Por lo cual:

- La cadena (chain) está formada por articulaciones (joints), eslabones con los actuadores los 2 actuadores que se añaden al final del eslabón brazo.
- El cuerpo body está formada por todas las cadenas (articulaciones y actuadores).
- Los efectores para la cabeza(head), brazo (arm), pierna (leg) y torso.

Se describe en la siguiente tabla cada elemento. Los giros roll, pitch, yaw se efectúan sobre los ejes establecidos como referencia.

| **Cadena (chain)** | **Articulación (joint)** | **Movimiento** | **Actuadores** | **Efectores** |
|---|---|---|---|---|
| | **Body (cuerpo)** | | | |
| Head(cabeza) | HeadYaw<br>HeadPitch | | | Head |
| LArm (brazo izquierdo) | LShoulderPitch<br>LShoulderRoll<br>LEIbowYaw<br>LEIbowRoll<br>LWristYaw | Hombro<br><br>codo | <br><br><br><br>LHand | LArm |
| RArm (brazo derecho) | RShoulderPitch<br>RShoulderRoll<br>REIbowYaw<br>REIbowRoll<br>RWristYaw | Hombro<br><br><br>codo | <br><br><br><br>RHand | RArm |
| LLeg (pierna izquierda) | LHipYawPitch<br>LHipRoll<br>LHipPitch<br>LKneePitch<br>LAnklePitch<br>LAnkleRoll | Cadera<br><br><br>rodilla<br>tobillo | | LLeg |
| RLeg (pierna derecha) | RHipYawPitch<br>RHipRoll<br>RHipPitch<br>RKneePitch<br>RAnklePitch<br>RAnkleRoll | Cadera<br><br><br>rodilla<br><br>tobillo | | RLeg |
| | | | | Torso |

Los detalles de las características técnicas del robot se encuentran en el anexo A.10 en el están las posiciones de longitud de los eslabones.

En [7] se realiza un estudio detallado de la cinemática directa e inversa. Se utiliza el método de Denavit-Hartengber [3][37] para obtener las ecuaciones que transforman las coordenadas articulares con las cartesianas. Se realizará la obtención de los

parámetros DH de las cadenas Head(cabeza) y Leg (pierna) que se utilizan en el proyecto.

Sean las matrices de rotación en x denominada Rx, en y denominada Ry, en z denominada Rz se tiene:

$$Rot\ x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\text{sen}(\alpha) \\ 0 & \text{sen}(\alpha) & \cos(\alpha) \end{bmatrix} \tag{7}$$

$$Rot\ y(\emptyset) = \begin{bmatrix} cos(\emptyset) & 0 & sen(\emptyset) \\ 0 & 1 & 0 \\ -sen(\emptyset) & 0 & cos(\emptyset) \end{bmatrix} \tag{8}$$

$$Rot\ z(\theta) = \begin{bmatrix} cos(\theta) & -sen(\theta) & 0 \\ sen(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{9}$$

El torso del robot es el punto donde todas las cadenas cinemáticas inician y son localizadas en el centro del cuerpo del robot. La base es el inicio de la cadena y fin es el efector de la cadena. El desplazamiento o traslación se indica con la matriz A.

Se realiza el proceso de la extracción de los puntos en el espacio tridimensional $(p_x, p_y, p_z)$ y las orientaciones del punto final $(a_x, a_y, a_z)$.Se tiene que:

$$\begin{aligned} p_x &= T_{(1,4)} \\ p_y &= T_{(2,4)} \\ p_x &= T_{(3,4)} \end{aligned} \tag{10}$$

La rotación final de la transformación se obtiene por $R_z R_y R_x$, con lo cual se obtiene las orientaciones del punto final $(a_x, a_y, a_z)$:

$$\begin{aligned} a_x &= arctan2\left(T_{(3,2)}, T_{(3,3)}\right) \\ a_y &= arctan2\left(-T_{(3,1)}, \sqrt{{T_{(3,2)}}^2 + {T_{(3,3)}}^2}\right) \\ a_x &= arctan2\left(T_{(2,1)}, T_{(1,1)}\right) \end{aligned} \tag{11}$$

Se obtendrán los parámetros DH y Se muestra los parámetros DH para la cadena Head del robot NAO.

Tabla 6 . Parámetros DH para la cadena Head.

| **Frame (articulación)** | **a** | **$\alpha$** | **d** | **θ** |
|---|---|---|---|---|
| Base | A(0,0, offset de la nuca Z) | | | |
| HeadYaw | 0 | 0 | 0 | $\theta_1$ |
| HeadPitch | 0 | $-\frac{\pi}{2}$ | 0 | $\theta_2 - \frac{\pi}{2}$ |
| Rotación | $R_x\left(\frac{\pi}{2}\right) R_y\left(\frac{\pi}{2}\right)$ | | | |
| Cámara superior | A(cámara superiorX, 0, cámara superior Z) | | | |
| Cámara inferior | A(cámara inferiorX, 0, cámara inferior Z) | | | |
| Cámara superior X=53.9mm. Cámara superior Z =67.9mm.<br>Cámara inferior X=48.8mm . Cámara inferior Z=23.8 mm | | | | |

Combinando las matrices de transformación, se encuentra el punto del efector:

$$^{Base}T_{End} = {}^{Base}A_0 \ {}^0T_1 \ {}^1T_2 \ R_x\left(\frac{\pi}{2}\right) R_y\left(\frac{\pi}{2}\right) \ {}^2A_{End} \qquad (12)$$

Donde ${}^0T_1$ es la matriz de transformación DH de la articulación HeadYaw, y ${}^1T_2$ es de HeadPitch. ${}^2A_{End}$ es una de las dos matrices de translación dadas. En este ejercicio para la cámara superior e inferior. El punto del efector final se obtiene de $^{Base}T_{End}$.

Para la pierna derecha se tienen los siguientes parámetros DH:

Tabla 7. Parámetros DH para la pierna derecha.

| **Frame (articulación)** | **a** | **$\alpha$** | **d** | **θ** |
|---|---|---|---|---|
| Base | A(0,-hip offsetY, -hip offsetZ) | | | |
| RHipYawPitch | 0 | $-\frac{\pi}{4}$ | 0 | $\theta_1 - \frac{\pi}{2}$ |
| RHipRoll | 0 | $-\frac{\pi}{2}$ | 0 | $\theta_2 - \frac{\pi}{4}$ |
| RHipPitch | 0 | $\frac{\pi}{2}$ | 0 | $\theta_3$ |
| RKneePitch | -ThighLength | 0 | 0 | $\theta_4$ |
| RAnklePitch | -TibiaLenght | 0 | 0 | $\theta_5$ |
| RAnkleRoll | 0 | $-\frac{\pi}{2}$ | 0 | $\theta_6$ |
| Rotation | $R_z(\pi) R_y\left(-\frac{\pi}{2}\right)$ | | | |
| End efector | A(0,0, -FootHeight) | | | |

La cinemática para la pierna derecha que es simétrica la pierna izquierda es:

$$^{Base}T_{End} = {}^{Base}A_0 \ {}^0T_1 \ {}^1T_2 \ {}^2T_3 \ {}^3T_4 \ {}^4T_5 \ {}^5T_6 \ R_z(\pi) R_y\left(-\frac{\pi}{2}\right) \ {}^6A_{End} \qquad (13)$$

La diferencia entre las dos piernas es la distancia entre el eje y, así como las articulaciones que rotan sobre este eje.

### 3.9.2. Obtención de la masa del robot

Se obtiene la masa de cada cadena y cada articulación con el siguiente algoritmo

```
Para todas las cadenas:

        cadenamasa= motionProxy.getMass(nombre)
        lista articulaciones =  motionProxy.getBodyNames(nombre)
Para todas las articulaciones
            jointmasa= motionProxy.getMass(njoints)
```

Con lo cual se describe la masa de cada uno. A continuación, se muestra la masa de la cadena cabeza, brazo izquierdo y pierna izquierda.

```
Body mass 5.30535030365
Com mass 5.30535030365
cadenas  ['Head', 'LArm', 'LLeg', 'RLeg', 'RArm']
Torso mass  1.04955995083
Es cadena  Head
cadena:  Head  mass  0.683749973774
------ lista joints  ['HeadYaw', 'HeadPitch']
--------- joint:  HeadYaw  mass =  0.0784199982882
--------- joint:  HeadPitch  mass =  0.605329990387
Verificacion masa de esta cadena Head suma de los joints  0.683749988675
cadena:  LArm  mass  0.578580021858
------ lista joints  ['LShoulderPitch', 'LShoulderRoll', 'LElbowYaw', 'LElbowRoll', 'LWristYaw', 'LHand']
--------- joint:  LShoulderPitch  mass =  0.093039996922
--------- joint:  LShoulderRoll  mass =  0.157769992948
--------- joint:  LElbowYaw  mass =  0.0648299977183
--------- joint:  LElbowRoll  mass =  0.077610000968
--------- joint:  LWristYaw  mass =  0.1853300035
Verificacion masa de esta cadena LArm suma de los joints  0.578579992056
cadena:  LLeg  mass  1.20744001865
------ lista joints  ['LHipYawPitch', 'LHipRoll', 'LHipPitch', 'LKneePitch', 'LAnklePitch', 'LAnkleRoll']
--------- joint:  LHipYawPitch  mass =  0.0698100030422
--------- joint:  LHipRoll  mass =  0.140530005097
--------- joint:  LHipPitch  mass =  0.389679998159
--------- joint:  LKneePitch  mass =  0.301420003176
--------- joint:  LAnklePitch  mass =  0.134159997106
--------- joint:  LAnkleRoll  mass =  0.171839997172
Verificacion masa de esta cadena LLeg suma de los joints  1.20744000375
```

Figura 18. Masa de las cadenas y articulaciones del Robot NAO (Kg).

### 3.9.3. Obtención del centro de masa

Se obtiene el Centro de Masa COM (Center Of Mass) del solido (S) en metros. Se requiere la matriz Inercial con valores relativos al eje coordenado (o,R).

$$COM = \begin{bmatrix} X_G \\ Y_G \\ Z_G \end{bmatrix}_{(o,R)} \tag{14}$$

$$[I_o S]_R = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}_R$$

Se procede a obtener el COM. Utilizando las siguientes métodos de del NAOqi.

```
nombreCadenas = motionProxy.getBodyNames(name)
combody = motionProxy.getCOM("Body",frame, useSensors)
comtorso = motionProxy.getCOM("Torso",frame, useSensors)
# recorrido por cada cadena y cada joint
        comjoint = motionProxy.getCOM(njoints,frame, useSensors)
```

Figura 19. Comandos para obtener el COM del robot NAO.

Para la cadena Body y Head el centro de masa es:

```
COM Body ---------- x = 0.0129609489813 y = 0.00105522957165 z = -0.0328637994826
cadenas  ['Head', 'LArm', 'LLeg', 'RLeg', 'RArm']
COM Torso ---------- x = -0.00412999978289 y = 0.0 z = 0.0434199981391
Es cadena  Head
COM cadena Head x = 0.0119333053008 y = -0.000421705743065 z = 0.168357789516
------ lista joints  ['HeadYaw', 'HeadPitch']
---------COM joint HeadYaw x = -9.99376152322e-06 y = 3.53165120259e-07 z = 0.0990799963474
---------COM joint HeadPitch x = 0.0134805496782 y = -0.000476383138448 z = 0.177332654595
```

Figura 20. Centro de Masa COM de la cadena Body y Head.

### 3.9.4. Control de los efectores.

Se controla los efectores usando la cinemática inversa.

En base a [39] del modelo geométrico del robot  y dada la posición de los efectores:

$$X = \left[P_x, P_y, P_z, P_{wx,} P_{wy}, P_{wz}\right] \tag{15}$$

Estos efectores relativos al espacio absoluto en función de todas las articulaciones:

$$q = [q_1, q_2, \ldots\ldots, q_n] \tag{16}$$

$$X = f(q) \tag{17}$$

Este modelo cinemático es derivado respecto al tiempo.

$$\dot{X} = \frac{\delta}{\delta t} f(q)\dot{q} \tag{18}$$

$$\dot{X} = J(q)\dot{q}$$

Donde $J(q)$ es la matriz Jacobiana. Por lo tanto, para controlar un efector y estimar la posición de la articulación se utiliza el modelo cinemático inverso:

$$\dot{q} = J^{-1}\dot{X} \tag{19}$$

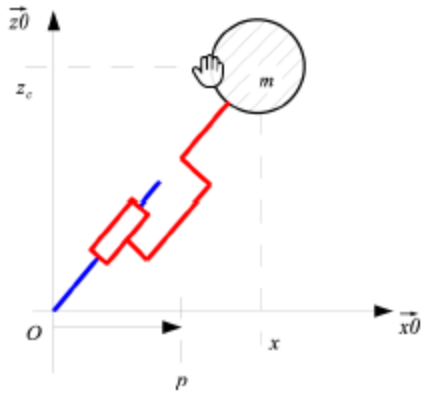
### 3.10. Locomoción del robot bípedo y control del movimiento – Motion Control

Para trabajar con la locomoción, se debe conocer el mecanismo del robot, la cinemática, dinámica y teoría de control. El robot utilizado es de tipo humanoide y los experimentos efectuados en el laboratorio son que el robot camine de manera suave. La locomoción con patas, requiere mayores grados de libertad y más complejidad en lo mecánico que la locomoción con ruedas.

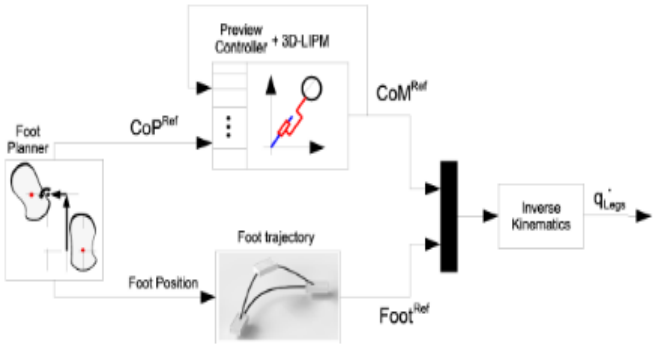
El robot NAO tiene tanto en su pierna izquierda y derecha 5 DOF, más 1 DOF compartido. En total para la locomoción en la parte inferior tiene 11 DOF.

La eficiencia de locomoción bípeda depende de la masa de las piernas y la masa del cuerpo, debido a que es lo que tiene que soportar para realizar el movimiento. Una de las desventajas de la locomoción bípeda es la potencia requerida y la complejidad.

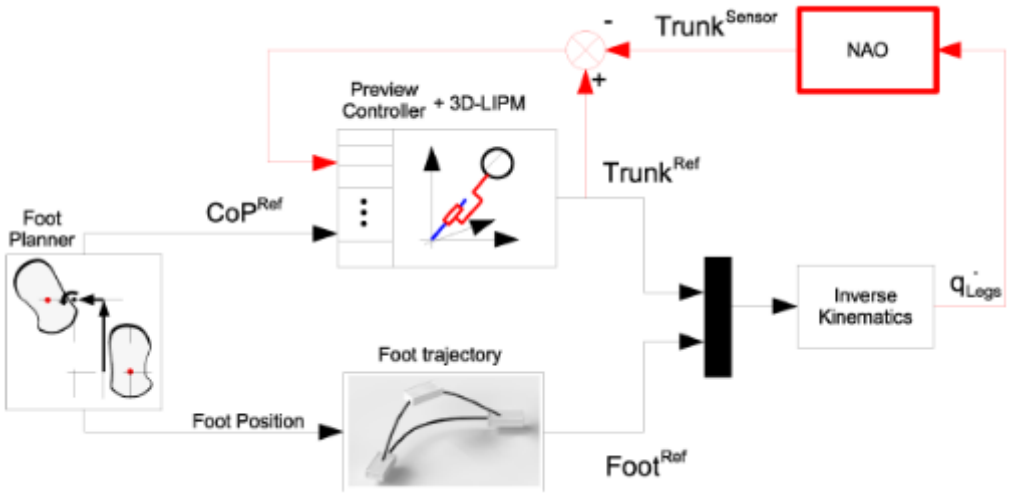
A pesar de que el robot no se implementa, se experimenta que consume mucha potencia y para efectuar actividades o tareas, al consumir energía se descarga con rapidez. Realizar el movimiento del robot a un punto exacto es complicado, pero el robot NAO tiene comportamientos para caminar pre configurados. Al caminar el robot, de forma interna crea patrones de pasos. De esos se obtiene el punto de momento cero o ZMP (Zero Moment Point) mediante muestreo, el cual posteriormente se transforma en una trayectoria del COM centro de masa y a los ángulos de las articulaciones [21]. La caminada del robot NAO [40] utiliza un péndulo lineal invertido. El modelo a  lazo abierto y cerrado para la caminada lo expone [41]




a) péndulo invertido lineal.




b) Modelo a lazo abierto del motor de caminado.




C) Modelo a lazo cerrado del motor de caminado.

Figura 21. Modelo de caminada a lazo abierto y cerrado[41].

### 3.10.1. Control de la locomoción

Aunque aparentemente se puede realizar el movimiento de caminada. La mayoría de las veces el robot no finaliza en la posición estimada. Los comandos para efectuar el movimiento se realizan con mucha imprecisión debido al deslizamiento del pie y holgura de las articulaciones.

Antes de que el robot inicie el proceso de caminar, se deben especificar acciones de configuración en el robot para disminuir una caída o ubicación de posición no correcta. Se requiere establecer adicional al eje de referencia los frames de ubicación del robot. Denominados Frame_Robot {FR}, Frame_World {FW}. El {FR} es el promedio de dos posiciones del pie, proyectados sobre el eje Z. El {FW} es el mundo del robot que se toma como referencia cuando inicia el primer movimiento. Será diferente cuando el robot realice una rotación en el eje Z.

Una vez establecido se efectúa el procedimiento para la locomoción que se muestra en la Figura 22.

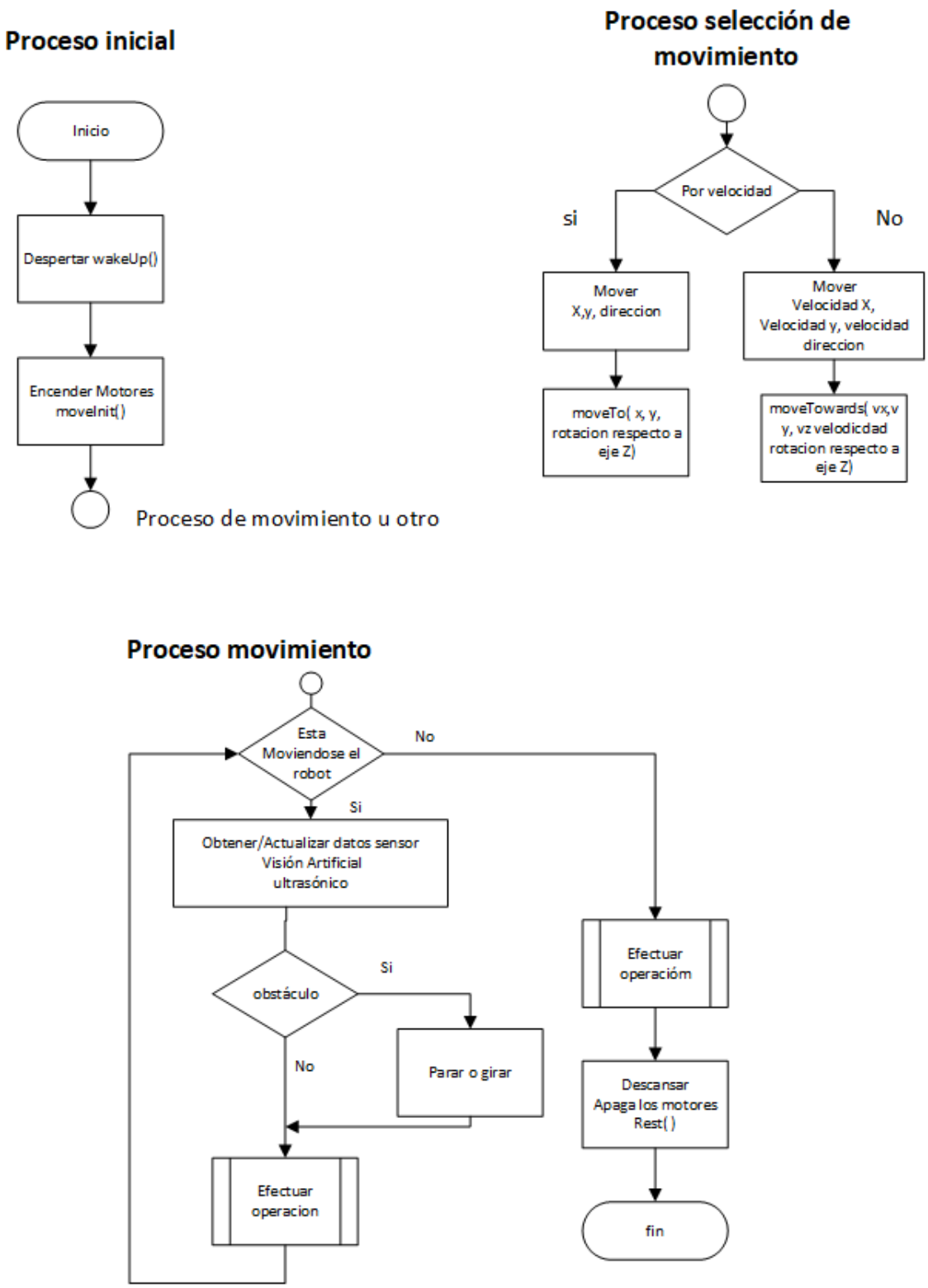



Figura 22. Proceso para la locomoción.

El método despertar wakeUp () acciona los motores y ubica al robot en una posición inicial para que los pies pueden moverse sin inconvenientes.

```
motionProxy  = ALProxy("ALMotion","127.0.0.1", 9559)
postureProxy = ALProxy("ALRobotPosture", ","127.0.0.1", 9559)

motionProxy.wakeUp() #despierta al robot

postureProxy.goToPosture("StandInit", 0.5) #

motionProxy.moveInit() # inicializa el proceso de movimiento, chequea la pose del robot
```

Figura 23. Proceso de inicialización del robot antes del movimiento. Ejm con robot virtual

Con wakeUp () enciende los motores e intenta ir a una posición inicial. Con moveInit () se verifica y establece que la posición del robot antes del movimiento es la correcta. Pero no realiza el movimiento. Con goToPosture () se establece una posición adecuada ideal para iniciar el movimiento. Observar que se utilizan dos proxys, uno para el movimiento con ALMotion y otro con la postura que es ALRobotPosture. La descripción de las posturas utilizadas en el proyecto se indica en el anexo A.8.

Se utilizan en el trabajo las siguientes dos formas de movimiento para la locomoción de Caminar y mover. Para efectuarlo se deben utilizar los siguientes métodos:

moverPosicion(x, y,θ).

moverPosicion(velocidadX, velocidadY, velocidad θ).

**Movimimiento a posición estimada en x, y y ang en radianes**

```
# primero sentencias para obtener datos del sonar y vision
motionProxy.moveTo(x,y,angr)

    while motionProxy.moveIsActive():  # Este proceso se ejecuta en otro hilo, si es en el mismo
                                       #debe ubicar post
            # realizar acción
            motionProxy.getRobotVelocity()
            motionProxy.getRobotPosition(False)
```

**Movimiento en velocidad**

```
    frecuencia = 1.0
    motionProxy.moveToward(vx, vy, vang, [["Frequency", frecuencia]])
```

**# detección de obstáculos o parada de emergencia**

```
    motionProxy.Killmove()  # No se considera el estado de balance del robot el cual puede caer
```

Figura 24. Comandos para moverse, caminar y parar.

Se debe tener presente la configuración del movimiento para los pies, con esto se disminuye los errores de posición al caminar y a la vez se evita caídas del robot.

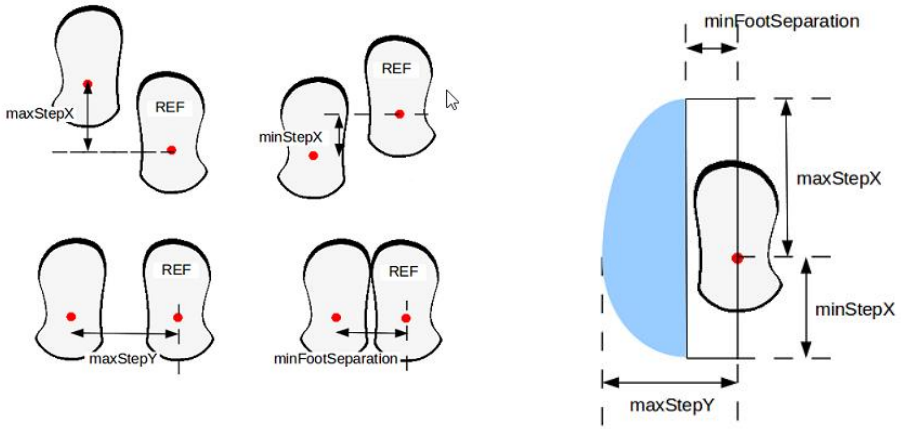



Figura 25. Parámetros de configuración para la caminada [39].

Debido a experimentación, entre mayor es el paso en x, y que se envía al robot, y entre mayor es el ángulo de rotación, el robot es bastante impreciso. Por lo tanto se establecen los siguientes parámetros para una locomoción correcta, disminuyendo el error de desplazamiento.

Tabla 8. Parámetros de configuración para una locomoción que evite caídas.

| | |
|---|---|
| Dx: desplazamiento x | 0.10 m |
| Dy: desplazamiento y | 0.10 m |
| Ángulo de rotación | 15° o $\frac{\pi}{12}$ |
| MinFootSeparation | 0.088 m (no editable) |
| maxStepX= | 0.06 m |
| minStepX | -0.04 m (no editable) |
| maxStepY | 0.14 m |

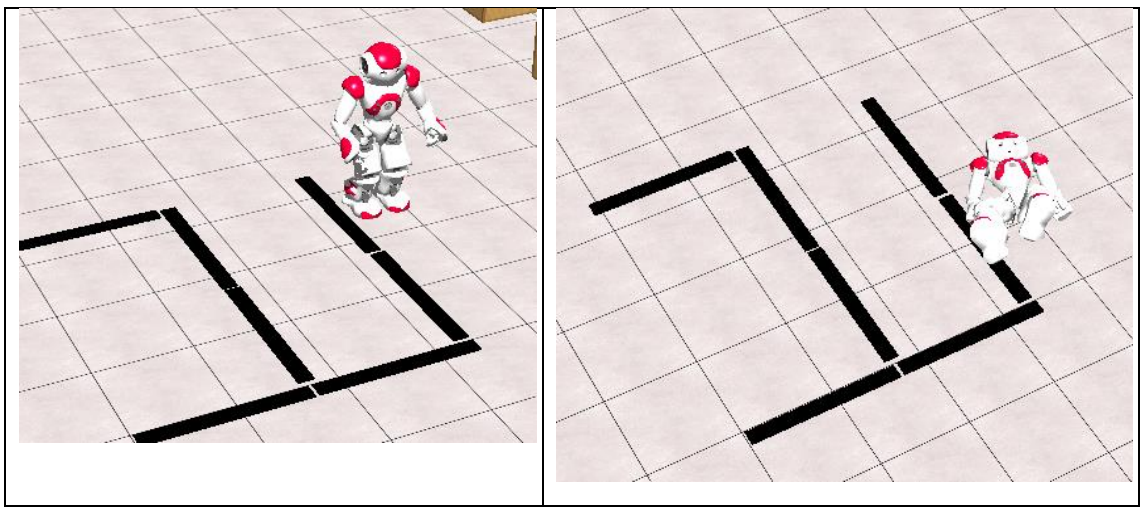


Figura 26. Caída del robot al no controlar la locomoción.

El robot NAO al realizar un movimiento bloquea de forma nativa a otras funciones. Por tal motivo se reestructura el mismo mediante la creación de un hilo y que en cada ciclo se realice un procesamiento de consulta del estado de sensor de video y ultrasónico. Se establece el diseño en la siguiente sección de control de hilos.

### 3.10.2. Control de hilos para locomoción y programación paralela

El robot se programa de forma paralela en ciertas actividades tal como caminar y hablar a la vez.

```
Esto es ideal para caminar y a la vez hablar. Ambas son estructuras bloqueantes
salida = motionProxy.post.moveTo(x, y, angr)


Si se desea esperar hasta que finalice el movimiento
salida = motionProxy.post.moveTo(x, y, angr)
motionProxy.wait(salida, 0) # por el momento para no bloquear el enviado

Si se desea controlar que ocurre en cada movimiento
salida = motionProxy.post.moveTo(x, y, angr)
  while motionProxy.moveIsActive():
        # realizar el control

time.sleep(2)
Hay que esperar un tiempo para que se realice la orden
```

Figura 27. Control de hilos para programación paralela

### 3.10.3. Sentencias básicas de comunicación para programación paralela

Las instrucciones mínimas para efectuar tareas paralelas se describen a continuación

```
Haciendo que el robot hable y se mueva a la vez. Se utiliza una estructura post.El método
moveTo() es bloqueante, por lo cual es requerido un hilo

from naoqi import ALProxy
motion = ALProxy("ALMotion", ip, puerto)
tts    = ALProxy("ALTextToSpeech", ip, puerto)
motion.moveInit()
motion.post.moveTo(0.5, 0, 0)
tts.say("Estoy hablando")

Si se desea que espera hasta que se ejecute una acción, se establece una variable que espera
con el método espere o wait() hasta que finalice la operación

from naoqi import ALProxy
motion = ALProxy("ALMotion", "127.0.0.1" puerto)
motion.moveInit()
espera = motion.post.moveTo(0.5, 0, 0)
motion.wait(espera, 0)
```

Figura 28. Sentencias básicas para comunicación paralela

### 3.10.4. Control de la locomoción con un lazo cerrado

A lazo abierto el robot por lo general se desvía de una meta establecida debido a que no hay control de lazo cerrado para compensar el ruido de los sensores. El error se lo puede reducir con un controlador PI. El lazo cerrado solo se basa en el error entre la posición real y la posición [9].

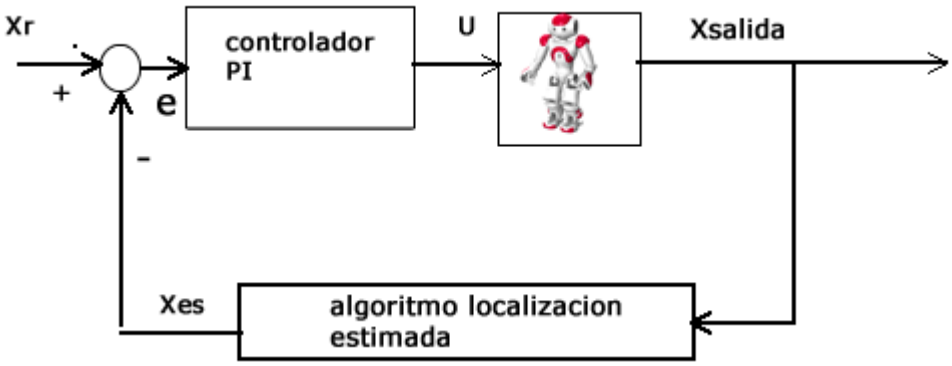



Figura 29. Controlador de lazo Cerrado.

Sea $x_R$ la posición real y $x_{es}$ la posición estimada que se obtiene por un algoritmo de localización. $\varepsilon$ es el umbral o threshold deseado y $k_p$ la constante de proporcionalidad. El proceso para el control es:

Definir un valor inicial al controlador U = constante.

Si el error $e = x_{R-}x_{es} > \varepsilon$ entonces $U = k_p\left(e + \frac{T}{T_i}\sum e\right)$

Sino U= constante.

De esta manera la desviación del robot al caminar se reduce.

Se procede a establecer las coordenadas absolutas del robot real y del virtual para el control de la locomoción. El algoritmo de localización para obtener la posición estimada es el filtro de partículas que se explica posteriormente.

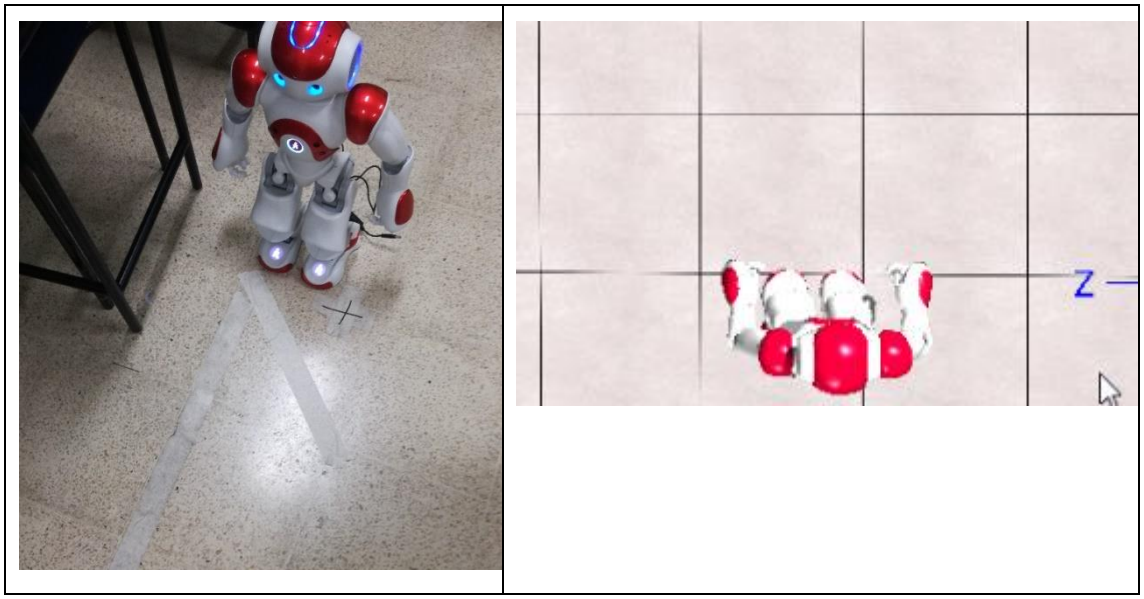



Figura 30. Ubicación de referencia para coordenadas absolutas – robot Real.

Figura 31. Medición de punto de referencia y distancia.

En la sección resultados 5.4 locomoción y localización se muestran los experimentos.

### 3.11. Visión artificial en el robot NAO y reconocimiento de objetos

Se utiliza visión artificial con la libreria OpenCv y se adapta al reconocimiento para que el robot identifique los diferentes elementos de una imagen, describirla y realizar una decisión en base a esta descripción.

Se complementa el procesamiento con el uso de la librería numpy de Python para la operación a nivel matricial de las imágenes.

En el anexo A.4 se describe la representación en el modelo de Color y en el anexo A.5 de forma breve el proceso segmentación y descripción de la imagen.

En el computador la imagen se representa en forma de matriz de dimensión mxn elementos, m filas y n columnas. Cada elemento o contenido de esta matriz está formada por valores enteros ubicado en localizaciones espaciales (i,j) denominadas pixel de picture element. Por lo tanto una imagen es una función de intensidad bidimensional I(i,j) donde i,j representan las coordenadas espaciales y el valor de I es proporcional a la intensidad o nivel de gris. La resolución espacial es representada por la cantidad de filas y columnas (pixel). Cada pixel tiene un valor relacionado con el nivel de luminosidad que se obtiene por la cuantización del nivel de gris o intensidad términos usados forma indistinta. Este rango de valores de niveles de intensidad se denomina resolución en amplitud Resolución de niveles de gris o intensidad de 0 a 255, el cual el 0 representa el negro absoluto (intensidades bajas) y el 255 el blanco absoluto [42]. Esta imagen se representa con modelo o espacio de color. Existen varios espacios de colores. En el trabajo se utiliza el Gray, BGR, RGB, HSV y para obtener la imagen desde el robot el YUV.

La segmentación es extraer información de la imagen para detectar elementos de la imagen fundamentados en la similitud y discontinuidad [44].

Por técnicas similitud se detecta regiones que son áreas de la imagen en la que los pixeles que la forman tienen propiedades similares (en color, en intensidad entre otros). Utilizando técnicas de discontinuidad se detectan bordes que son líneas que separan dos regiones de diferentes propiedades, son pixeles en los cuales la imagen presenta una brusca variación en los niveles de gris.

Point Of Interest POI: Es cualquier punto donde la imagen cambia bidimensionalmente. Los puntos característicos, son puntos o pixeles que destacan en la imagen por su diferencia de nivel de intensidad con referencia a los píxeles vecinos. Estos por lo general se usan como entrada para entrenar o recuperar información de un clasificador.

La descripción tiene como fin extraer propiedades, características o atributos reconociendo estructuras de la imagen para identificar de forma única a la misma y utilizarlos en fases posteriores.

La detección de características y matching es utilizada en el proyecto [13] [12]. Estos tipos de características se pueden combinar en función de su orientación y la apariencia local (perfiles de borde) y también pueden ser buenos indicadores de los límites de los objetos. Después de detectar características se de realizar un emparejamiento o match.

A continuación, se presenta el diseño para obtener detectar y describir la imagen, paso previo para localizar el objeto.

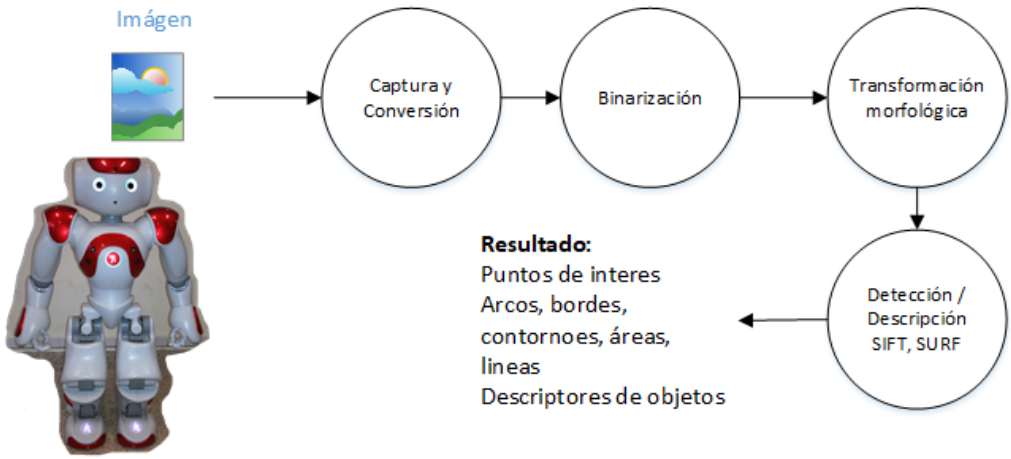



Figura 32. Diseño de la detección y descripción del objeto.

Capturada la imagen se procede a efectuar la detección o descripción de acuerdo a lo requerido: color, líneas, bordes, esquinas, áreas, centroides, puntos de interés.

En la Figura 33 se presenta el diseño para procesamiento específico del color, bordes, esquinas, líneas.

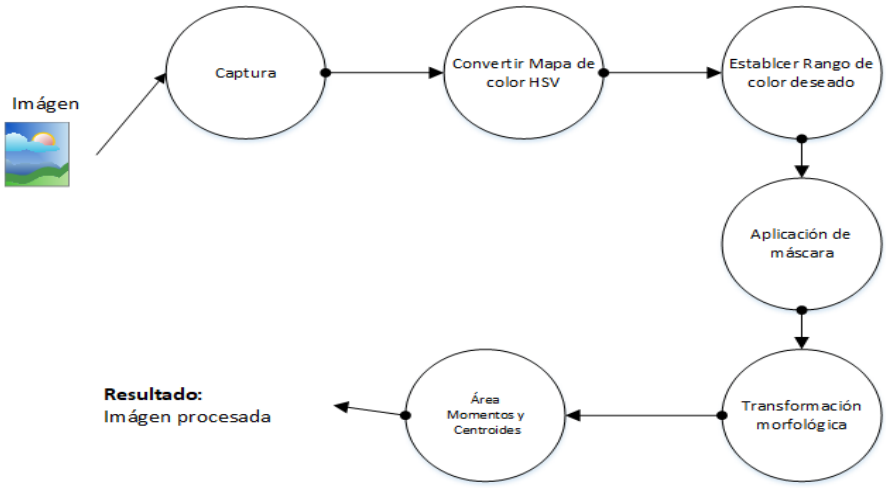
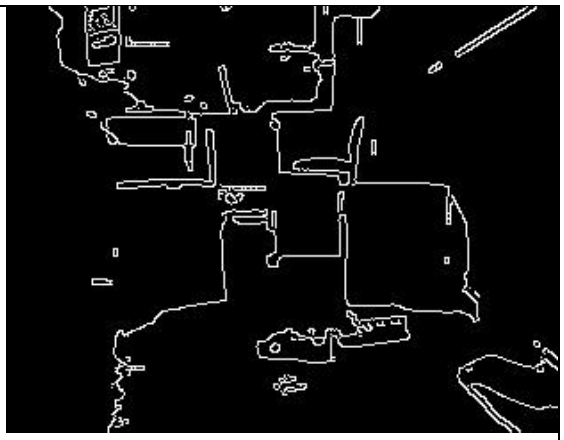


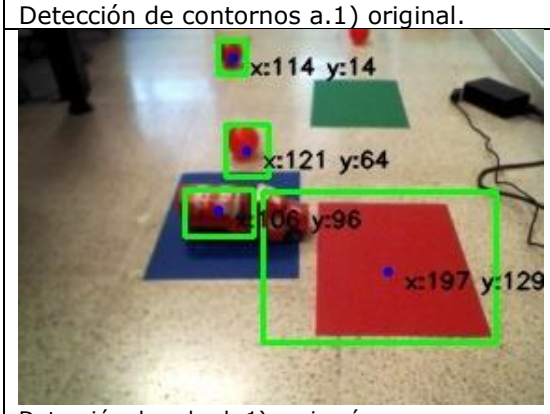

Figura 33. Procesamiento para detección de líneas, contornos, colores, regiones.

De forma breve esto se indica en la Figura 34 donde se segmenta los bordes en a) y después se detectan colores en b en base a un contorno, obteniendo los centroides.

Detección de contornos a.1) original.



a.2) contornos imagen procesada.




Detección de color b.1) varias áreas.


Detección de contornos b.2) en un área pequeña.

Figura 34. Aplicación de la visión artificial, bordes, contornos y detección de colores.

### 3.12. Calibración de la cámara

Se requiere calibrar la cámara para obtener la longitud focal f en $f_x$ y $f_x$ .Los parámetros intrínsicos [38] de una cámara son:

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (20)$$

La longitud focal f, es la distancia entre la imagen del plano y el centro de la cámara. S se utiliza si es sesgado. $f_x = \alpha f_y$. El $\alpha$ es el aspecto de radio para pixeles no cuadrados. Por lo general es 1.

Se obtienen los parámetros en el Robot NAO realizando el siguiente procesamiento con las imágenes que se muestran en la Figura 35.

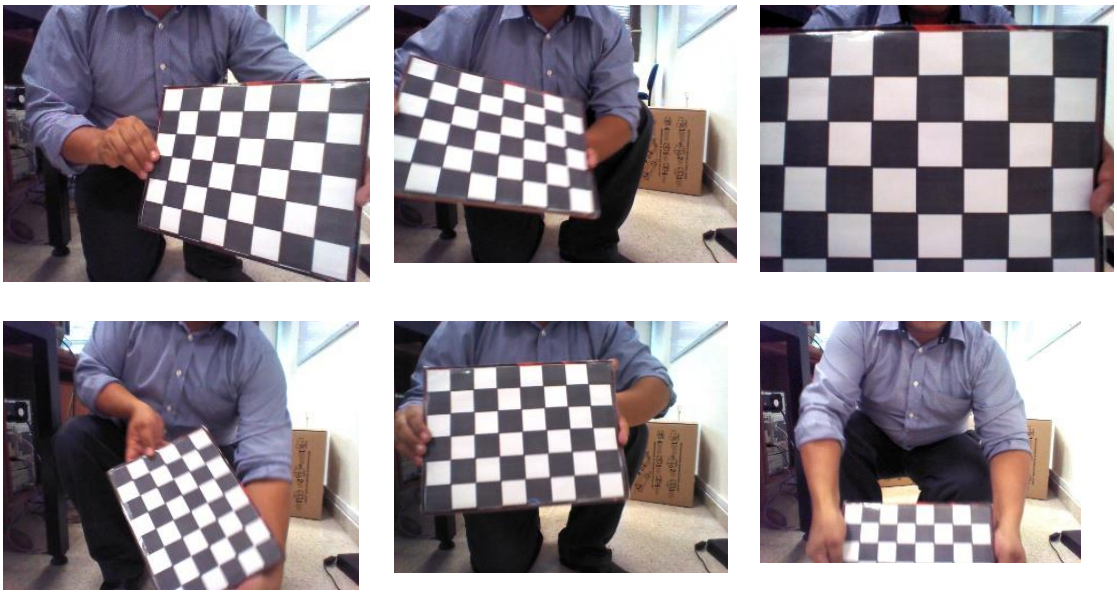


Figura 35. Figuras para calibrar la cámara

Los resultados son:

```
Matriz
 [[ 536.39559881          0.          334.03199264]
 [   0.          536.43966202          239.43393198]
 [   0.            0.                    1.        ]]
distorsion [[ -3.10623609e-01   2.22524864e-01   6.87554500e-04  -2.27913827e-04
   -1.94685072e-01]]
```

Con $f_x = 536.39$ y $f_y = 536.4$

### 3.13. Detección de distancias desde la cámara a un punto del suelo.

Para determinar una estimación de la distancia entre el robot NAO y el objetivo detectados se establecen una serie de relaciones entre el sistema de visión de la cámara y el sistema de coordenadas del mundo. El fin es obtener el centroide calculado del objeto con sus coordenadas espaciales. De esta manera se localiza por estructura geométrica el objeto [18].

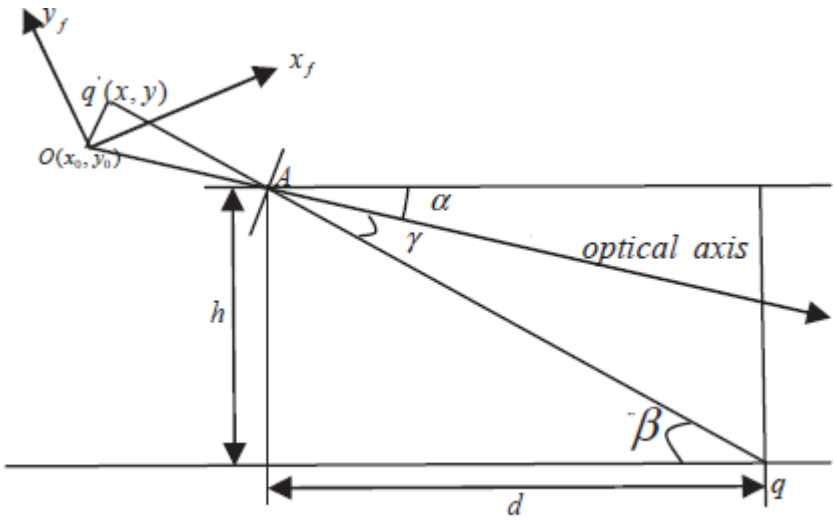



Figura 36. Relación Geométrica entre la cámara y un punto en el piso.

En la Figura 36 se describe a la altura de la cámara h. El ángulo entre la cámara y el plano horizontal es α. es La distancia sobre el plano horizontal de q y la proyección del centro del lente es d.

El punto $O(x_0, y_0)$ es el origen del centro de coordenadas $q'(x_0, y_0)$ es el plano de proyección para el punto q. En las siguientes ecuaciones se expresan las relaciones geométricas:

$$\beta = \alpha + \gamma \tag{21}$$

$$\tan\beta = \frac{h}{d} \tag{22}$$

$$tan\gamma = \frac{oq'}{f} \tag{23}$$

$$d = \frac{h}{tan(\alpha+\gamma)} = \frac{h}{tan\left(\alpha + \tan^{-1}\left(\frac{oq'}{f}\right)\right)} \tag{24}$$

Se obtienen por medición α y h.

$$oq' = (x^2 + y^2) \tag{25}$$

Las coordenadas (u,v) son las del sistema de coordenadas en pixeles de la imagen $O(u_0, v_0)$ relativas a $O(x_0, y_0)$. $q'(u,v)$ es el punto $q'(x,y)$ en coordenadas de pixeles.

Realizando la operación geométrica entre el plano de coordenadas del mundo y el plano de coordenadas en pixeles se tiene:

$$\frac{X}{Z} = \frac{x}{z} = \frac{x}{f} \tag{26}$$

$$\frac{Y}{Z} = \frac{y}{z} = \frac{y}{f} \tag{27}$$

$$x = \frac{(u - u_0)}{a_x} \tag{28}$$

$$y = \frac{(v - v_0)}{a_y} \tag{29}$$

Donde ax y ay son las direcciones en x y dirección en y del coeficiente de amplificación de la proyección de la imagen del plano. Con esto se encuentran los parámetros intrínsecos de la cámara y la calibración $k_x = a_x f$ y $k_y = a_y f$.

Con (28)(29) y (25) se obtiene:

$$\left(\frac{oq'}{f}\right)^2 = \left[\frac{(u - u_0)}{a_x}\right]^2 + \left[\frac{(v - v_0)}{a_y}\right]^2 \tag{30}$$

Para encontrar la distancia al punto q:

$$d = \frac{h}{tan(\alpha + \gamma)} = \frac{h}{tan\left(\alpha + \tan^{-1}\left(\sqrt{\left[\frac{(u - u_0)}{a_x}\right]^2 + \left[\frac{(v - v_0)}{a_y}\right]^2}\right)\right)} \tag{31}$$

Para obtener los puntos $q'(u,v)$ se encuentra de una región, área o descripción de la imagen el pixel que está en su centroide. Esto se lo realiza con los momentos de una figura.

Mediante datos experimentales y relaciones geométricas, se obtiene las posiciones en el espacio 2D que observan ambas cámaras.

$$\text{angulo} = tan^{-1}\left(\frac{vertical}{horizontal}\right) \quad (32)$$

Con lo cual los ángulos son de θ para la cámara inferior de 1.1069 radianes o 63.42 °. Mientras que para la cámara superior φ de 0.51138 o 29.3° . Esto se muestra en la Figura 37. Por lo tanto para detectar un objeto al ras del piso en la cámara superior debe estar a 82 cm frente al robot, para la cámara inferior a 23 cm.

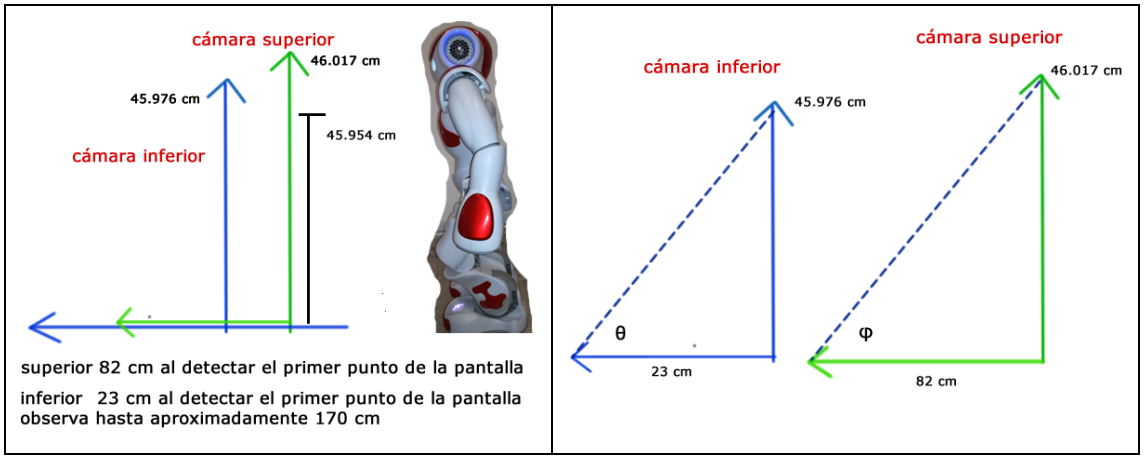



Figura 37. Posición en el eje x, del cual se observan los primeros elementos en pantalla.

### 3.14. Métodos de detección, descripción

La segmentación es extraer información de una imagen para detectar elementos en la imagen basado en la similitud y discontinuidad [44]. La descripción tiene como meta extraer propiedades, características o atributos reconociendo estructuras de la imagen para identificar de forma única a la misma y utilizarlos en fases posteriores.

#### 3.14.1. Obtención de los centroides por momentos

Los centroides en base a un área, son ideales para detectar el punto central del objeto. Se debe tener presente las características de la imagen para establecer los valores adecuados. Los momentos son medidas de la distribución de la forma que pueden utilizarse como características para el reconocimiento de patrones estadístico.

$$M_{xy} = \sum_i \sum_j i^x i^y f(i,j) \quad (33)$$

Se utiliza para describir objetos después de la segmentación. Se obtiene el área o intensidad total y a partir de ellos los centroides.

$$C_x = \frac{M_{10}}{M_{00}} \tag{34}$$

$$C_y = \frac{M_{01}}{M_{00}}$$

En el proyecto se utilizan para obtener el punto central de la imagen.

Antes de aplicar los métodos de Canny, Harris, transformadas de Hough para líneas, contornos se debe binarizar el objeto. Se pueden aplicar operaciones o transformaciones morfológicas tal como se indica en el anexo A.7.

### 3.14.2. Detección de bordes -Canny

Es un detector de bordes, uno de sus usos es cuando el kernel Laplaciano genera mucho ruido. Se trabaja con dos umbrales, el superior y el inferior. Este detector combina el borde de la primera y segunda derivada. Fue diseñado para optimizar la detección de bordes [16][15][14][12].

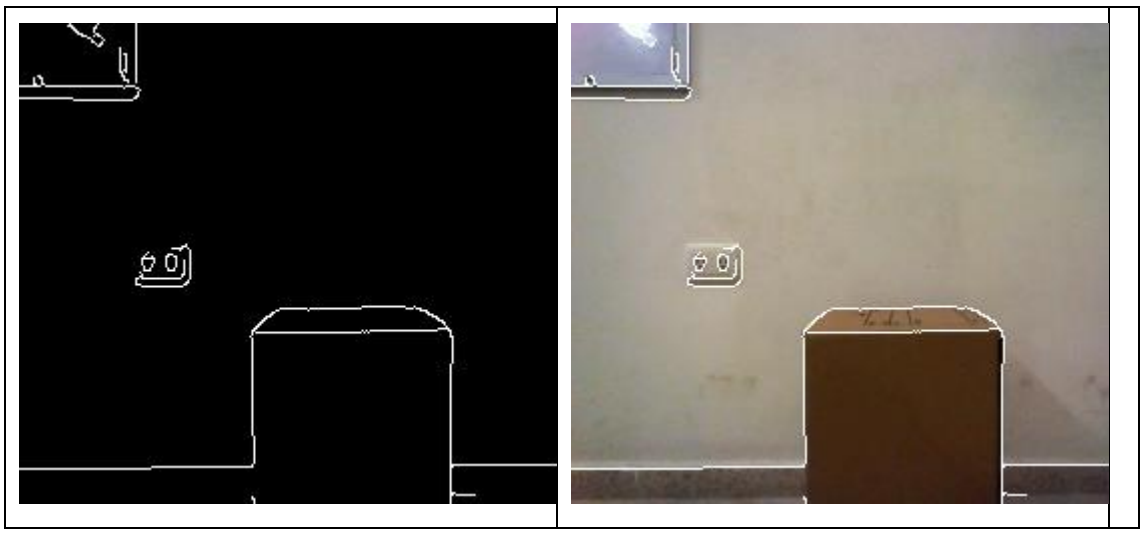


Figura 38. Aplicación de Canny en la imagen original.

### 3.14.3. Detección de contornos

En un trabajo de procesamiento de imágenes, no solo los bordes son necesarios. Por lo que se requiere los datos que hay en el borde y así procesar mejor la imagen.

El contorno se genera en una imagen binaria (en gris), con lo cual la imagen estará formada por líneas y contornos. Primero se debe detectar cuales puntos son bordes y se procede a obtener estos puntos. Un contorno se almacena como un vector de puntos, y todos los contornos se almacenan como un vector de contornos. Se muestra la imagen original, la máscara y la imagen procesada (máscara y contorno) [16][15][14][12].

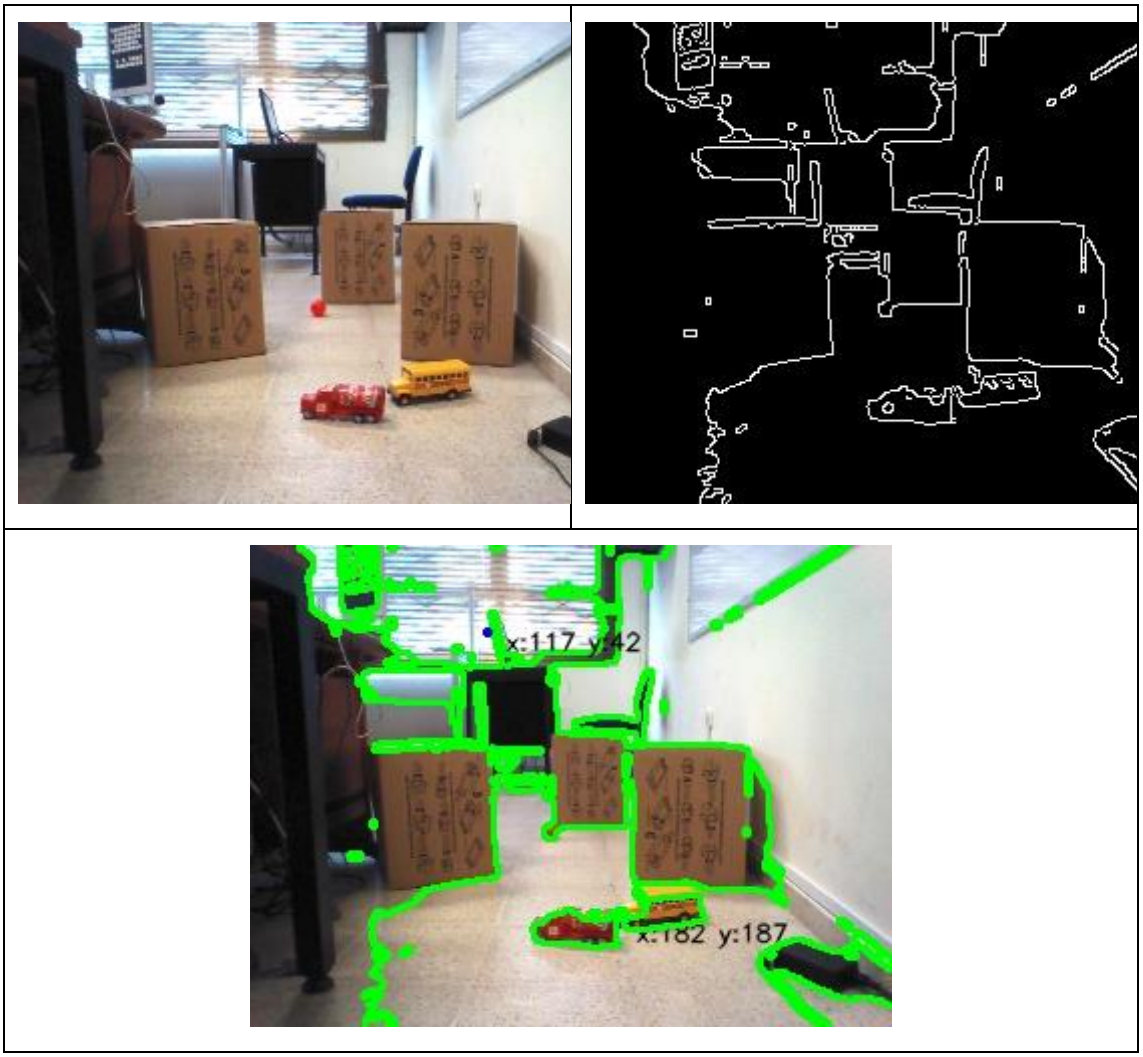



Figura 39. Aplicación de detección de contornos en imagen original.

### 3.14.4. Detección de esquinas Harris

Extracción de puntos de interés o esquinas (corners). Se utiliza Harris o método de auto valores se trabaja con una imagen binaria. El proceso es utilizar una matriz de 2x2 formada por las derivadas parciales de la imagen y analizar sus eigenvalores o

valores propios. Un puntos de esquina es un punto en los cuales ambos eigenvalores tienen altos valores [16][15][14][12].

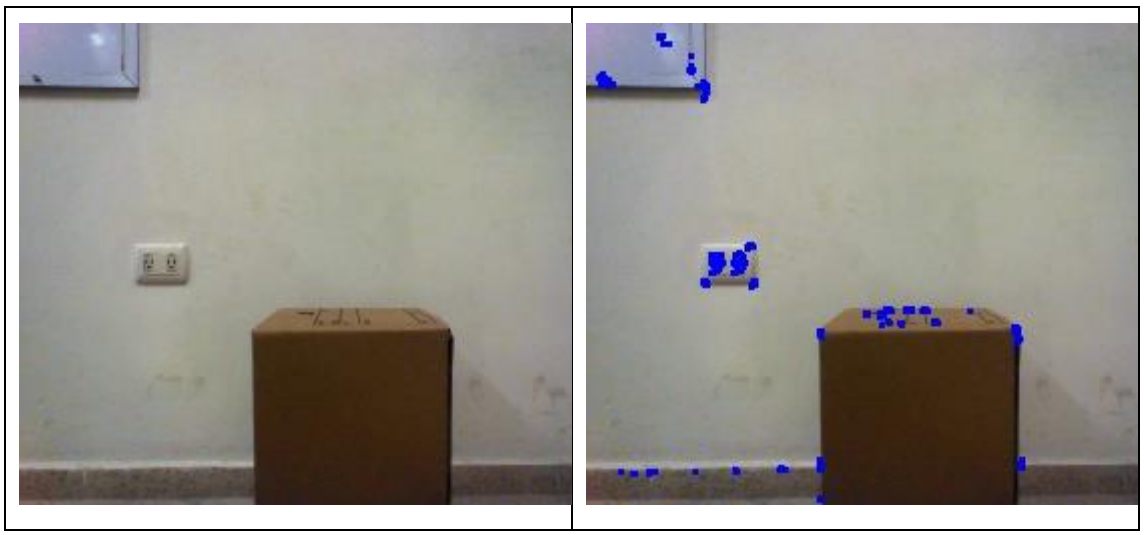


Figura 40. Aplicación de Harris en la imagen original.

### 3.14.5. Detección de colores

En la Figura 33 se indicó el proceso para realizar la detección de colores. Una de las propuestas es convertir al modo de color HSV, establecer que rango de valores incluyen el color rojo, verde, azul, amarillo, aplicar una transformación morfológica para reducir el ruido y aplicar la mascará con la cual se logra solo mostrar el color requerido. En la tabla siguiente se muestran unos rangos de valores de los colores a detectar. Según HSV el color rojo está en dos rangos [16][15][14][12].

Tabla 9. Rango de valores mínimos y máximos para detectar colores.

| |
|---|
| # Verdes:<br>verde_bajos = [49, 50, 50]<br>verde_altos = 100, 255, 210] |
| # Amarillos:<br>amarillo_bajos = [16, 76, 72]<br>amarillo_altos = [30, 255, 210] |
| # Azules:<br>azul_bajos = [100, 65, 75]<br>azul_altos = [130, 255, 255] |
| # Rojos:<br>rojo_bajos1 = [0, 65, 75]<br>rojo_altos1 = [12, 255, 255]<br>rojo_bajos2 = [240, 65, 75]<br>rojo_altos2 = [256, 255, 255] |

El robot frente a los objetos

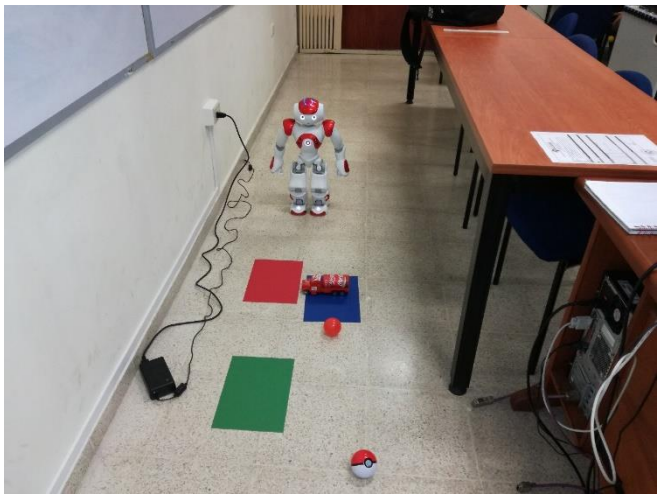


Figura 41. Robot frente a los objetos a detectar color.

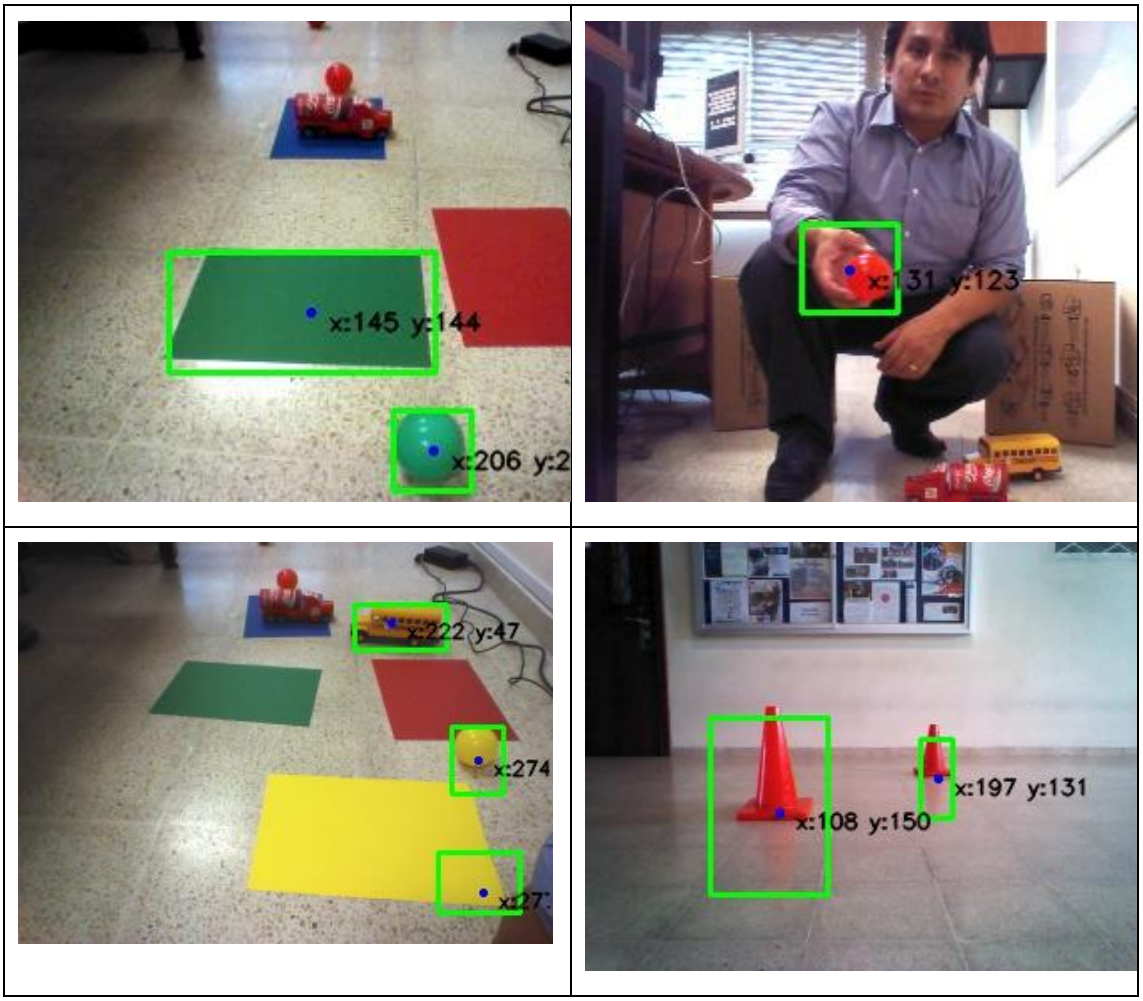



Figura 42. Detección de colores.

### 3.14.6. Detector y descriptor SIFT (Scale-Invariant Feature Transform)

Es un detector y descriptor patentado, utilizado para el seguimiento y reconocimiento [43] [16][15][14][12]. Se utiliza para extraer puntos clave y formar los descriptores de características asociados a la imagen. Este descriptor es invariante a la escala y rotación y en gran parte invariable a la iluminación y cambios de puntos de vista. SIFT utiliza la diferencia Gaussiana por lo cual es un proceso lento.

Mediante experimentación se debe tener la destreza para emparejar (matching) las características de la imagen y así evitar algún emparejamiento no adecuado. Una vez que se encuentra en la fase final de la extracción del punto clave, se debe describir la región alrededor del punto clave para que este se compare con otros puntos clave.

Para efectuar el emparejamiento de puntos clave (Matching Keypoints), se debe encontrar una coincidencia para el punto clave k y se la compara con el descriptor que está almacenado en una base de datos. El mejor punto clave será el que tiene la menor distancia euclidea al punto clave k. De esta forma todos los puntos clave serán igualados, por lo cual hay que identificar los puntos clave que coinciden con los que no coinciden en la base de datos.

Para efectuar el proceso de reconocimiento de objetos, deben utilizarse pocas características.

Figura 43. Aplicación de SIFT en la imagen original.

### 3.14.7. Descriptor SURF (Speeded-Up Robust Features)

SURF es una versión rápida de SIFT, está patentada y no disponible de forma libre para uso comercial [44] [16][15][14][12]. SIFT es un buen descriptor, pero utiliza bastantes recursos. Por lo cual no es apropiado para aplicaciones en tiempo real. Por lo cual SURF realiza el procesamiento mediante la aproximación al gaussiano por un filtro de caja simple. Este detalle permite que el algoritmo sea rápido.

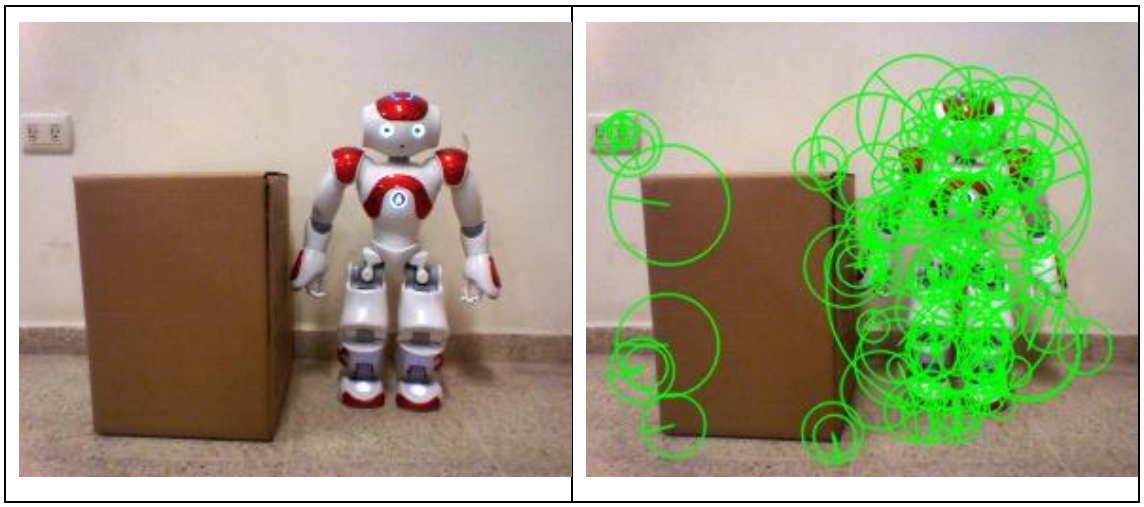


Figura 44. Aplicación de SURF en la imagen original.

### 3.14.8. Detector FAST (Features from Accelerated Segment Test)

FAST es un detector de esquinas [45][16][15][14][12]. Es mucho más rápido que otros detectores. Comparando Harris es de 5 a 10 veces más lento que FAST. Mientras que SIFT es 50 veces más lento que FAST. Aunque SURF es más rápido que SIFT, no lo es tanto para aplicaciones en tiempo real con limitante en recursos computacionales. Mediante aprendizaje automático se determina si el punto actual es un punto clave potencial. FAST es solo para detección de puntos claves, es un detector. Cuando se tienen los puntos claves, se debe utilizar para obtener los descriptores a SIFT o SURF.

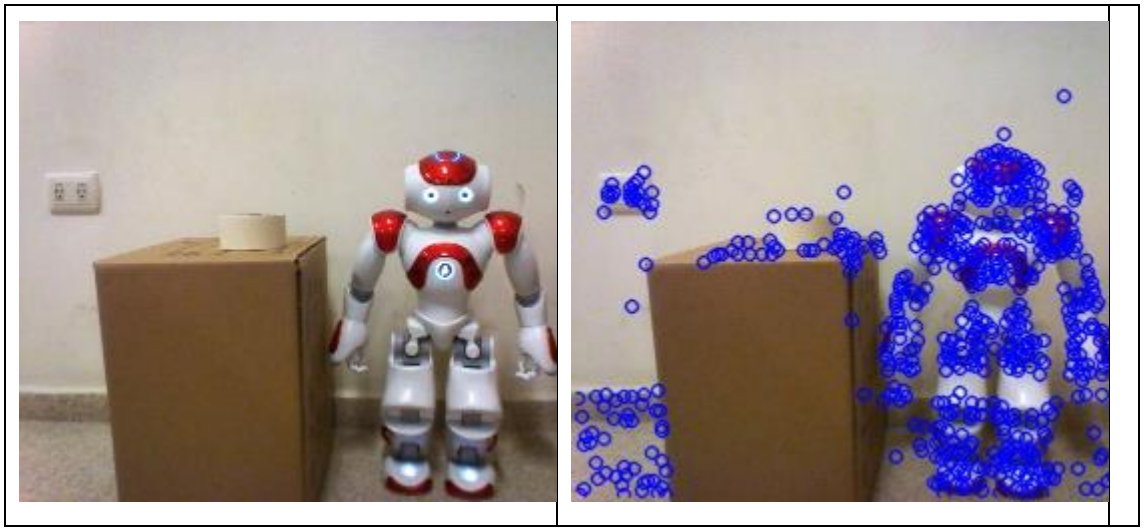


Figura 45. Aplicación de FAST en la imagen original.

### 3.14.9. Descriptor ORB (Oriented FAST and Rotated BRIEF)

Es un algoritmo que surge los laboratorios de OpenCv como alternativa a los algoritmos de SIFT y SURF que son patentados [15].

Figura 46. Aplicación de ORB en la imagen original.

### 3.14.10. Matching y Transformación homográfica

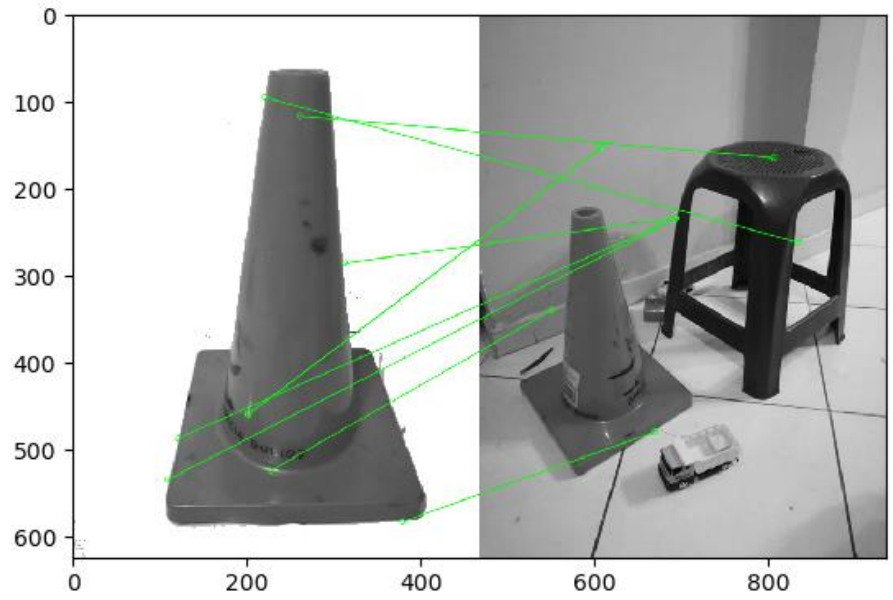
Una homografía es una proyección de transformación 2D que traza puntos de un plano a otro. Se utilizará para registro de imágenes.

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} \tag{35}$$

$$x' = Hx$$

En los descriptores mencionados, uno que se utiliza en el trabajo para reconocimiento y matching de puntos claves es SIFT. Se pueden también obtener puntos claves (Keypoints) con SURF u ORB.

En los experimentos efectuados, se describe el matching entre el cono de seguridad almacenado en una base de datos y el cono en el entorno.




**matching con valores con calibrados**

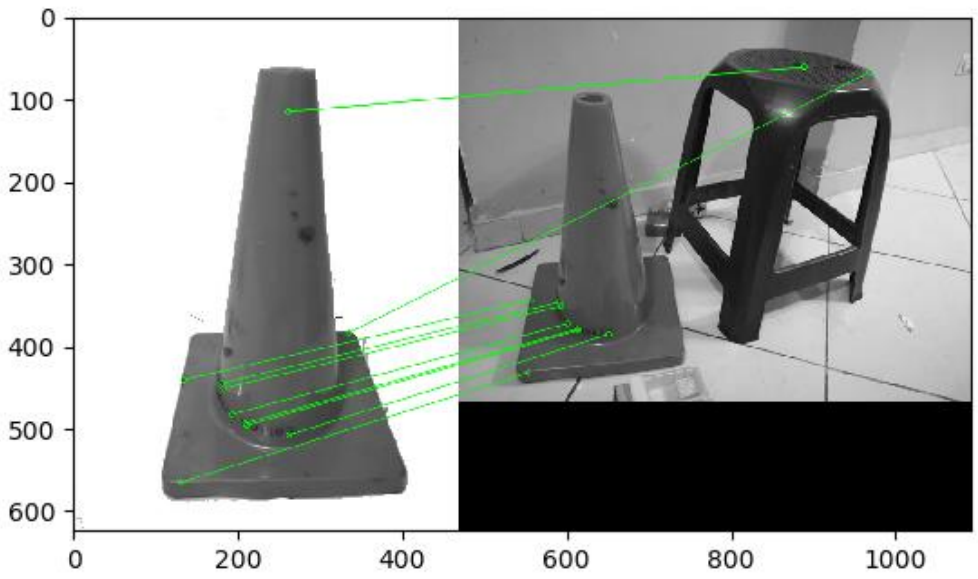



Figura 47 Homografía para matching.

**Métrica para medir el error**

Se utiliza la siguiente métrica para medir si los objetos fueron detectados en la posición esperada.

$$E = \frac{\sum_{i=1}^{C} \alpha_i}{C} \tag{36}$$

$$\alpha_i = \frac{\left| |\bar{d}| - |\bar{p_i}| \right|}{|\bar{d}|}$$

La ecuación detectará por lo general distancias, al punto $\bar{p_i}$ y la distancia del centroide.

### 3.15. Localización del robot Móvil

En La localización se estima la pose del robot con referencia a un mapa. El robot responde a la interrogante. ¿Dónde estoy?

Se utilizan por lo general tres tipos de localización: absoluta, incremental, seguimiento de marcas. En la incremental, se realiza un seguimiento de la pose, conociendo la posición inicial del robot. En la localización absoluta o localización global, no se conoce a priori la posición inicial. Y la Localización por marcas naturales o pasiva se utiliza como un landmark para generar puntos de referencia y localizar al robot en un entorno.

En el proyecto se utilizan pasivas naturales: objetos con formas y colores, y localización absoluta marcadores naturales.

En el robot NAO, se utiliza la cámara para obtener elementos del entorno. El problema es que cuando camina el Robot, la Localización Visual de las observaciones que se obtienen por la cámara tienen mucho ruido debido al movimiento del robot. Debido a la imprecisión al caminar y el ruido en las imágenes por el movimiento "agitación" del robot, la efectividad de la localización se reduce. Al moverse el robot la incertidumbre sobre su pose incrementa. La navegación no podría efectuarse correctamente debido a que la estimación de la pose del robot no correspondería a la ubicación real. [21]. Para mejorar la descripción se establece el siguiente diseño.

El robot camine cada $\Delta x = 0.10\, m$ .

Parar y volver a caminar.

Con esto se disminuye el inconveniente del ruido al caminar.

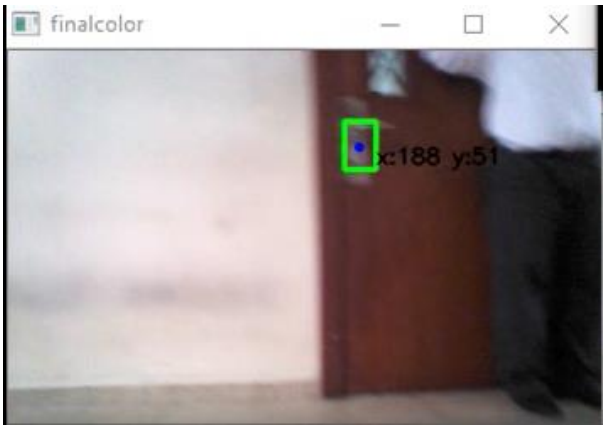



Figura 48. Imagen con ruido al caminar el robot.

En los resultados 5.4 se muestra los valores del experimento.

### 3.16. Localización por vision y movimiento a un punto del espacio planar

La localización basada en sistemas de visión artificial, tiene como meta encontrar puntos de interés o características en las imágenes capturadas por el sensor. De esta manera sirve para localizar al robot. Por ejemplo, detectar por colores, por regiones, por áreas, por puntos de interés en SIFT, SURF por matching y homografía.

Las imágenes en el escenario deben ser únicas. Sea por sus descriptores e inclusive posición.

En los experimentos se utilizan:

- landmark pasivos artificial. Ejm cartulinas de color.
- Landmark pasivos naturales: cartones, carros de colores, triángulos de seguridad de colores, líneas del piso.

Se muestra el proceso referente a landmarks pasivos naturales con triángulos de seguridad frontales al robot.

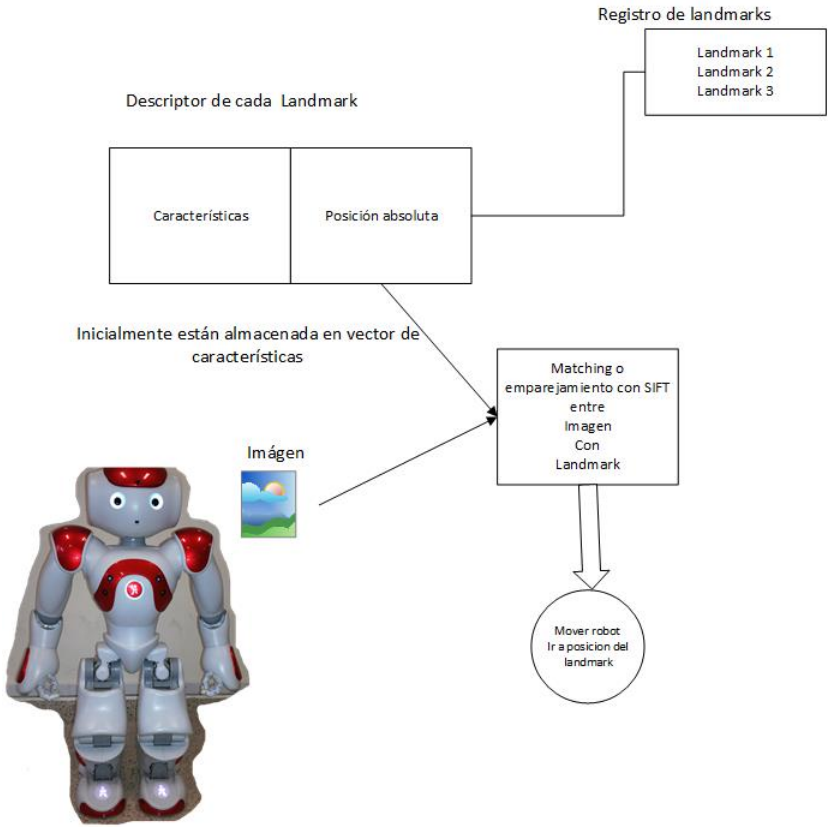



Figura 49. Emparejamiento del landmark y registro de vectores de características.

Una vez segmentada y descrita la imagen, está se compara a elementos almacenados con características de colores, regiones, puntos de interés.

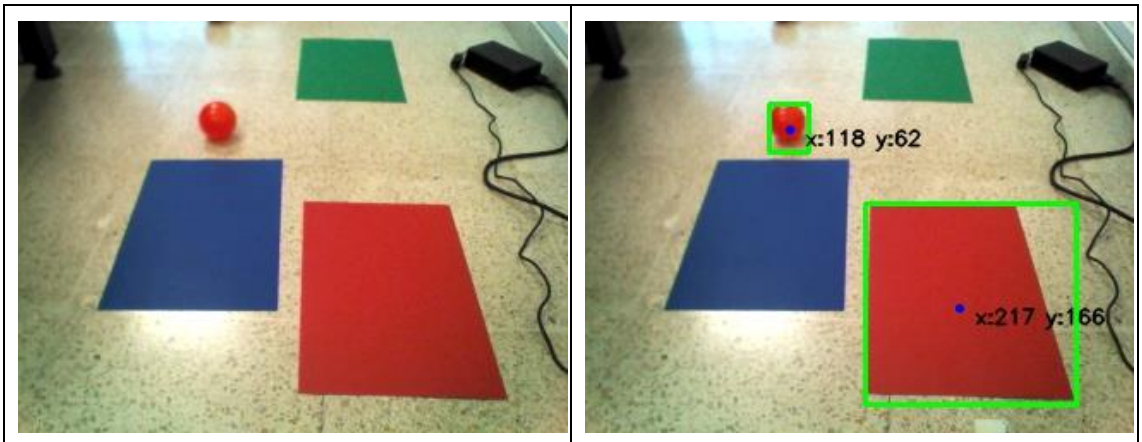
Por lo cual se utiliza las siguientes fases:

- Fase 1 Segmentación (Detección bordes-discontinuidad y regiones- similitud, puntos de interés).
- Fase 2 Descripción (Extraer propiedades de los elementos detectados).
- Fase 3. Realización de matching o emparejamiento.
- Fase 4 Mapeo y navegación.

### 3.16.1. Localización por color

El matching es por color de un área o región. Se utilizan cartulinas de colores ubicadas en el piso. El robot detecta la cartulina, estima la posición relación pixeles a coordenadas 2D con el siguiente proceso.

1. Se toma la cámara inferior del robot.
2. Se obtienen el centroide del elemento a detectar.
3. Los pixeles se transforman a coordenadas espaciales en 2D.
4. El robot se mueve al punto establecido.




a) Figura original.

b) Figura detecta para caminar e ir a la posición indicada.

Figura 50. Imagen a detectar por el robot (cámara inferior).

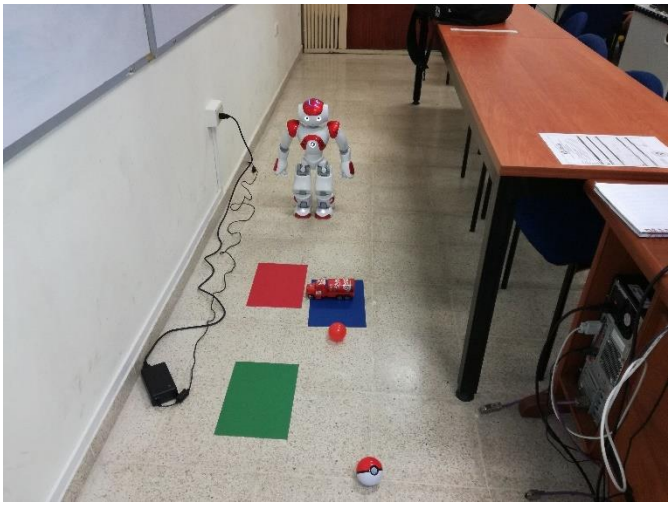


Figura 51. Robot detectando landmark de colores.

#### 3.16.1.1. Coordenadas espaciales. Para la cámara inferior del Robot

De acuerdo al proceso indicado en 3.13 Detección de distancias desde la cámara a un punto del suelo.

Cada 120 pixeles en altura. genera de forma aproximada 0.53 metros. Este valor es bastante cercano. Si un objeto está sobre los 120 pixeles. Se utiliza proyección geométrica y no se puede estimar correctamente el punto.

Tabla 10. Relación pixeles unidades de frame.

| Pixeles | Posición |
|---|---|
| Alto 120 | 0.53 en X |
| Ancho 116 | 0.33 en Y |

Estos valores son de bastante aproximación referente a los primeros X, Y pixeles.

### 3.16.2. Localización por Matching o emparejamiento

El matching o emparejamiento se realiza por las características del objeto. Primero se construye un vector de características con referencia a la apariencia visual de un punto:

1. Está almacenado un registro de características de landmark su posición absoluta en el plano x,y del robot frente al objeto.
2. El robot se mueve al punto adecuado.

Se obtiene la figura en escena del robot, frente a los landmarks.

Figura 52. Robot frente a los landmarks.

Figura 53. Landmark utilizado en las pruebas vistas por el robot.

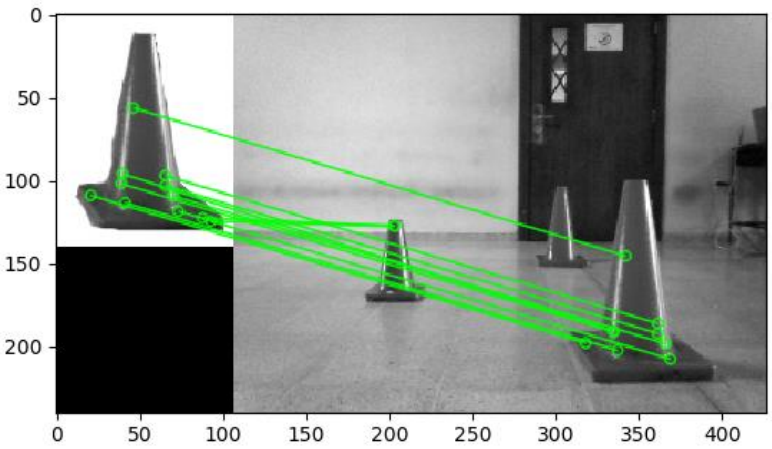



Figura 54. Figura realizada descripción y matching.

#### 3.16.2.1. Coordenadas espaciales. Para la cámara superior o frontal

Conociendo los landmark y su ubicación en el frame mundo se aplica el algoritmo de localización de filtro de partículas para determinar la localización del robot.

Este proceso [21] se describe con los siguientes requisitos previos:

- Se requiere un mapa conocido.
- El mapa contiene la posición absoluta de los landmark y sus descriptores.
- Se ha almacenado las características que debe tener un landmark para que sea reconocido, emparejado o igualado (matching) con los detectados en la captura de la imagen.

Para estimar la pose del robot en el plano 2D se requiere:

$$x_t = (x_t, y_t, \theta_t) \tag{37}$$

Se utiliza el algoritmo de localización (filtro de partículas) con un control $u_t$ y se necesita observaciones $z_t$.

Con la cámara, se obtiene un Landmark Visual mediante SIFT extrayendo sus características.

Cuando el robot "empareja" con una característica percibida del landmark, procesa la posición relativa en el plano xy como una observación $z_t$ para el filtro de partículas

$$z_t = (r_t, \varphi_t) \tag{38}$$

Se seleccionan los elementos que cumplan con una región de área del punto estimado.

El robot se mueve al punto establecido.

### 3.17. Actualización de la posición de objetos por visión.

El robot es fijo hasta localizar el elemento.

Los landmark son fijos y se conocen la posición absoluta referente al frame {W} mundo.

Se segmenta por contornos, colores, momentos, descriptores SIFT, SURF.

Se obtiene los pixeles de estos elementos y se realiza el movimiento por la relación pie del robot y baldosa del piso el cual se conoce sus dimensiones. Así se tiene una relación para obtener el desplazamiento en el eje.

| | |
|---|---|
| ancho del pie del robot | 15 cm |
| ancho de la baldosa del piso | 30x30 cm |

En caso de que se desee conocer la posición relativa al mundo o con respecto a otro landmark. Se aplica transformaciones homogéneas. Tal como lo indica la Figura 55. El robot se mueve a punto P con referencia a la coordenada absoluta, este es el marco de referencia para el robot. Aplicando composición de transformaciones se obtiene las posiciones relativas al landmark B, A o C respecto al robot.

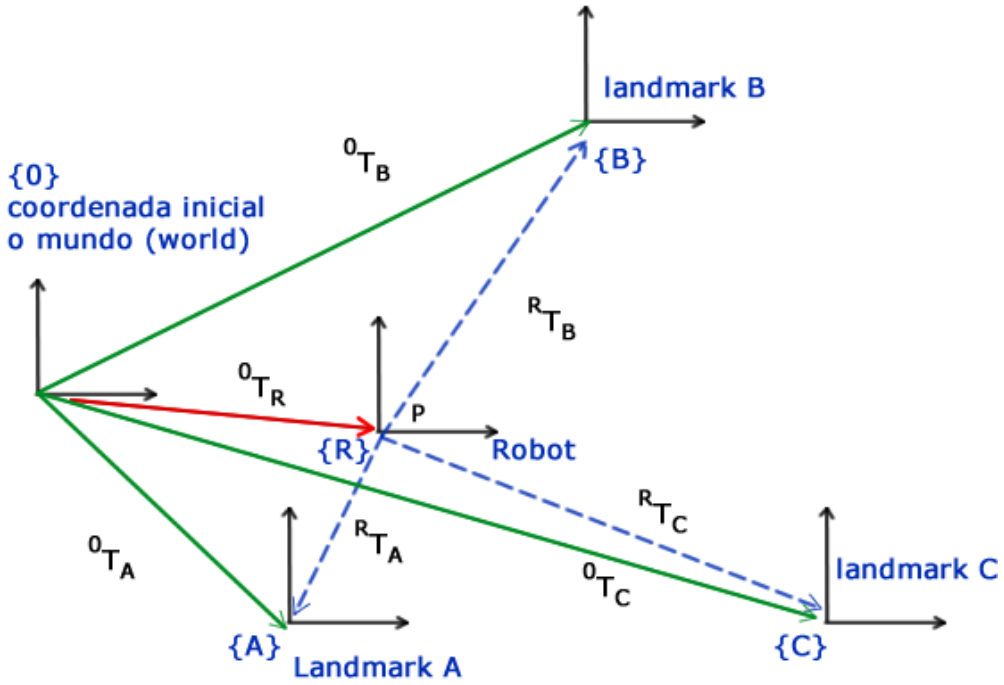



Figura 55. Transformaciones homogéneas para la posición relativa del robot

Por ejemplo, la posición relativa de {B} respecto a {R} es:

$$^{R}T_{B} = \left( {}^{A}T_{R} \right)^{-1} {}^{0}T_{B} \quad (39)$$

### 3.18. SLAM

SLAM es la abreviatura de Simultaneous Localization And Mapping.[46] también conocida como concurrente Mapeo y Localización CML Concurrent Mapping and Localization. ¿Qué se requiere primero el Mapping o la Localización?

SLAM soluciona uno de los problemas de la robótica. La localización con precisión es un requisito para construir un buen mapa Mapping, y teniendo un buen mapa es esencial para una buena localización. El denominado problema de quién fue primero el huevo o la gallina.

Antes de que un robot pueda responder sobre el entorno que lo rodea en base a un conjunto de observaciones, se debe conocer desde que ubicaciones se han realizado estas observaciones. Por lo cual es difícil estimar la posición del robot sin un mapa.

En la práctica estos dos problemas no pueden resolverse de forma independiente. SLAM es el proceso de simultáneamente seguir la posición del robot móvil relativo a su entorno y construir a la vez un mapa.

Cuando el SLAM usa sensores visuales para obtener información del entorno (observaciones) que lo rodea es llamado Visual SLAM. En el proyecto no se usa visual SLAM debido a que no se procesa las imágenes cronológicamente.

El sensor está sujeto a ruido, por lo cual es una implementación muy sencilla del SLAM, que es describir el objeto, realizar un matching y obtener la posición absoluta. Obtener su posición en pixeles, ingresar los datos de posición x, y al algoritmo de SLAM que mueve el robot al punto x, y estimado.

Los datos que obtenga el robot por la percepción, se entregarán al algoritmo de SLAM tal como lo indica la Figura 5.

Existen diferentes alternativas para solucionar entre ellas EKF Filtro extendido de Kalman, filtro de partículas, GraphSLAM y otras.

### 3.19. Filtro de partículas

En esta fase que recibe el SLAM son la posición absoluta en el plano del landmark. Se va a conocer la posición en base a los landmark detectados [20][47][48].

Los métodos probabilísticos se denominan filtros. Los Filtros de partículas son una secuencia de algoritmos para estimar el estado de un sistema. Este filtro es para espacio de estados continuos, es con una distribución belief multimodal y su eficiencia

es aproximada. La ventaja del filtro de partículas es estimar donde el robot podría estar moviéndose. Estás partículas son una creencia o belief de muestras aleatorias. El filtro de partículas es una alternativa no paramétrica de la implementación del filtro de bayes. La siguiente figura muestra la aplicación en un gaussiano.

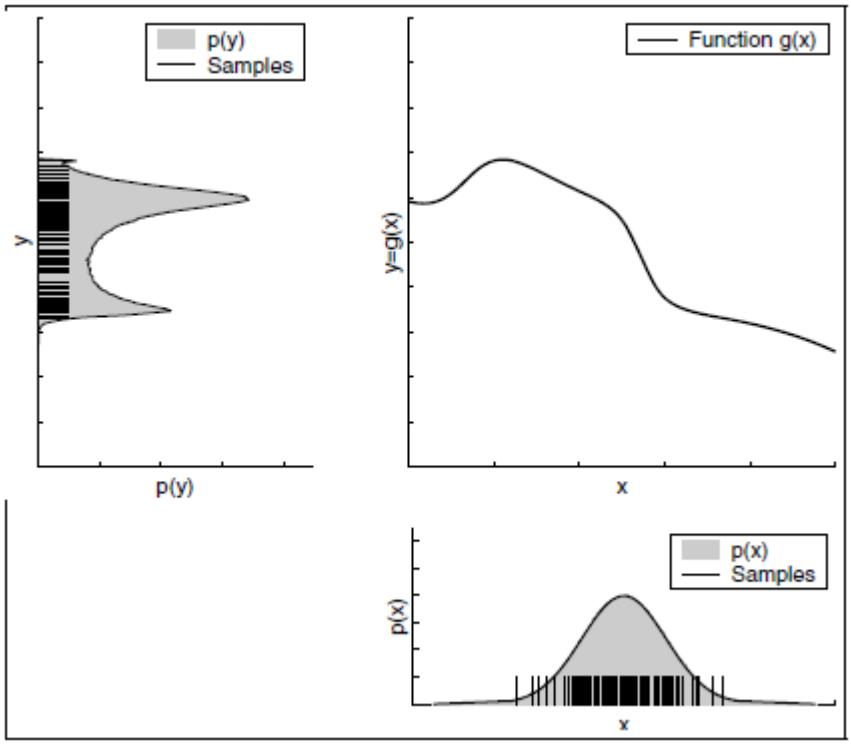



Figura 56. Representación del filtro de partículas[20]

La clave del filtro de partículas es representar la posterior belief $bel(x_t)$ por un conjunto de muestras aleatorias extraidas de la belief.

En robótica probabilística cada vez que el robot se mueve pierde información y cada vez que obtiene datos del sensor gana información. El proceso es obtener información mediante el sensor y mover para actualizar las distribuciones de probabilidad[20]

En el filtro de partículas se debe muestrear (sampling) y remuestrear (resampling). El filtro es la estimación del movimiento, mediante observación y actualización de Bayes. Se debe tener en cuenta que las partículas se deben hacer mover de forma aleatoria.

Se utiliza para el algoritmo una distribución objetivo f:

$$\mathrm{f} : p(x \mid z_1, z_2, \ldots, z_n) = \frac{\prod_k p(z_k \mid x) \quad p(x)}{p(z_1, z_2, \ldots, z_n)} \tag{40}$$

Distribución g de sampling

$$\mathrm{g} : p(x \mid z_l) = \frac{p(z_l \mid x) p(x)}{p(z_l)} \tag{41}$$

Importancia de los pesos w

$$\mathrm{w} : \frac{f}{g} = \frac{p(x \mid z_1, z_2, \ldots, z_n)}{p(x \mid z_l)} = \frac{p(z_l) \prod_{k \neq l} p(z_k \mid x)}{p(z_1, z_2, \ldots, z_n)} \tag{42}$$

Cuando un landmark está detectado, se tomará como posición del robot la partícula con más peso.

Si en cambio no hay detectado ningún landmark, la posición estimada será la media de las partículas

El entorno para efectuar la práctica se indica en la siguiente tabla

| **Objetos en el entorno** | **Posición en el entorno** |
|---|---|
| Área del espacio de trabajo | 4 m x 4 m |
| Robot | Posición 0.3 , 16 |
| Landmark A | (0.13, 10), 0.30x0.30 |
| Landmark B | (0.21,2.4), 0.15 x0.15 |
| Landmark C | (0.27,10), ancho =0.30x0.30 |
| | |

La prueba se realizará con el robot caminando en línea recta, teniendo los obstáculos a un lado. Los colores para interpretación se indican a continuación

| Elemento | color |
|---|---|
| Landmark | naranja |
| Robot | Celeste azul |

| | |
|---|---|
| Orientación del robot | Negro con dirección a la punta de la flecha |
| Partícula en el sampling | Violeta |
| Orientación de la partícula | Café |
| Partícula al resampling | Verde |
| Orientación al resampling | Verde fuerte |

El proceso inicia con el sampling como lo indica la Figura 57.

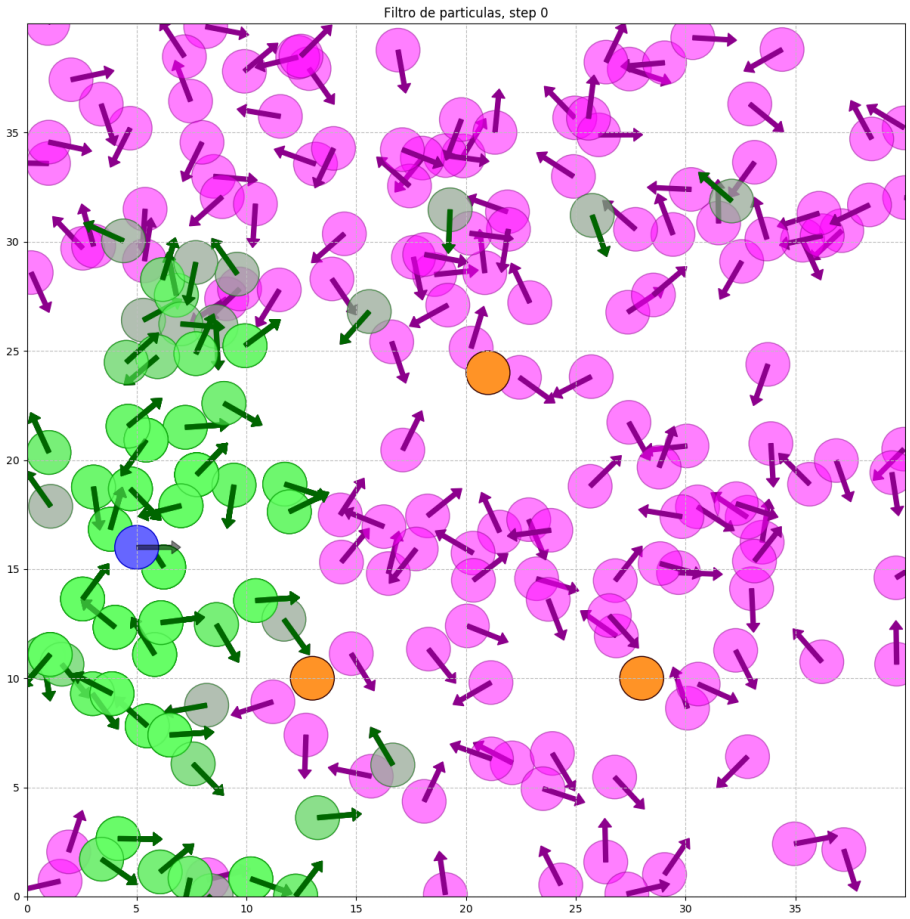



Figura 57. Filtro de partícula inicial. Generación de sampling.

Después se efectúa el resampling. Observar que al ir moviendo el robot las partículas se agrupan donde hay mayor peso.

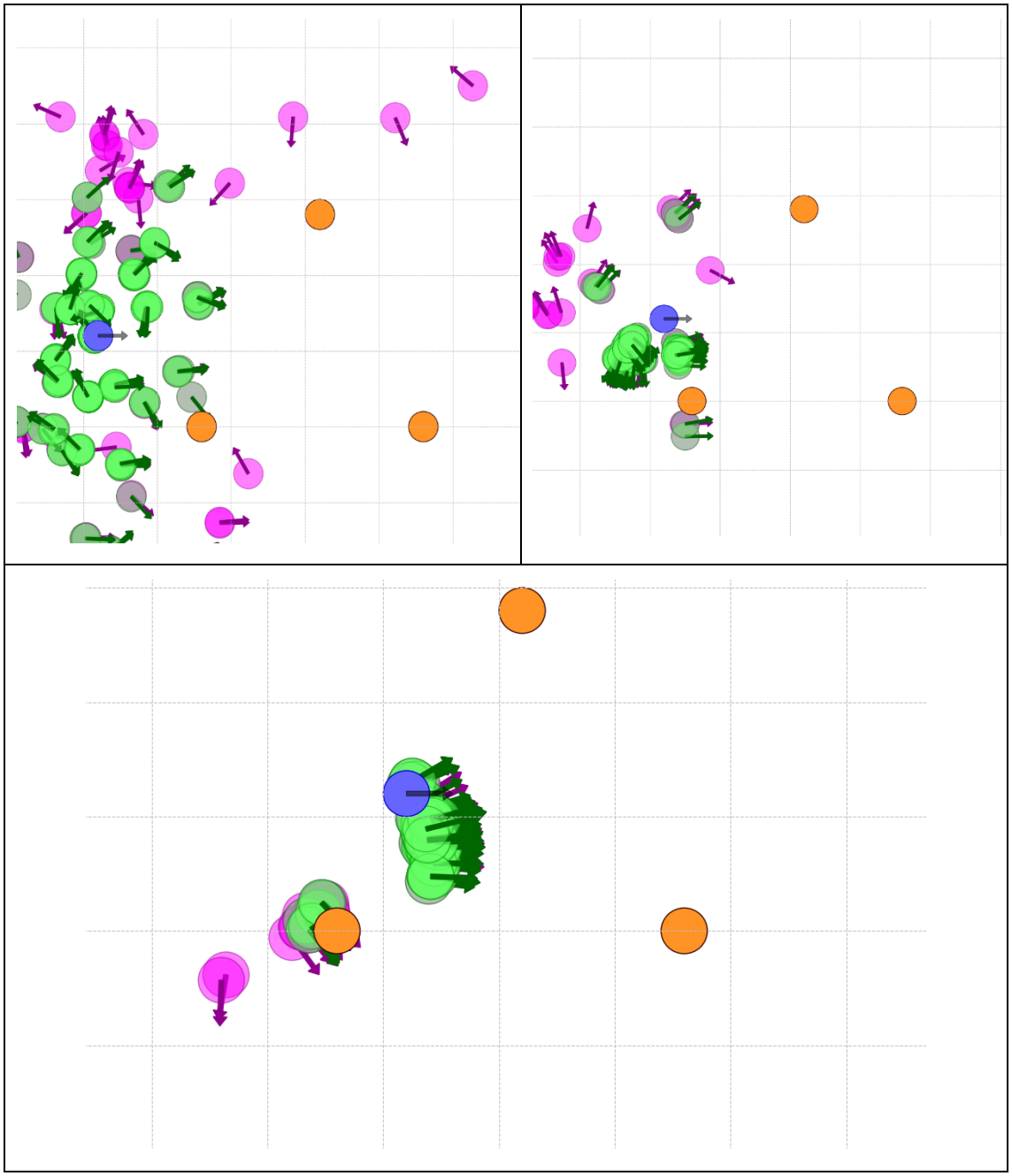


Figura 58. Filtro de partículas. Robot va moviendo de 0.10 m.

Después de recorrer 3 metros.

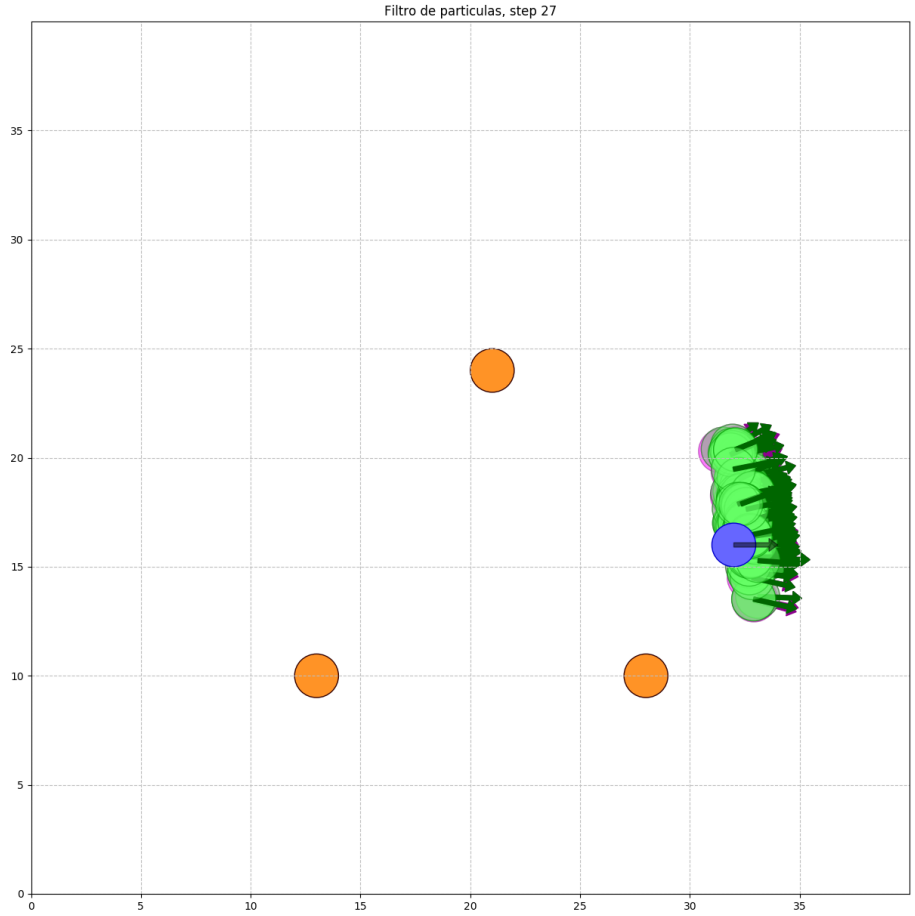



Figura 59.Filtro de partícula con resampling.

## 3.20. Construcción del mapa

Para conocer el entorno que rodea al robot, se requiere identificar: el espacio libre o los obstáculos, reconocer objetos y/o regiones en el entorno. Esto se lo efectúa con una representación de un mapa. El mapping o mapeo es el requerimiento de integrar los datos recuperados por los sensores del robot en una representación del entorno. [4]. Permite contestar la interrogante de ¿Cómo se observa el mundo el robot? Entre los tipos de mapas se tiene a los mapas métricos y los topológicos. Los métricos establecen los elementos del mapa respecto a un sistema de referencia. Entre las formas para elaborar un mapa se tiene a: mapas basados en rejillas (grid o celdas), mapas geométricos (características), mapas topológicos (jerárquicos).

### 3.20.1. Construcción del mapa basado en rejilla

Para construir el mapa se representa el entorno como un grid o rejilla, que tendrán en cada celda un estado de ocupado o no ocupado. El contenido de la celda se actualiza mientras el robot explora el terreno de acuerdo a la estrategia escogida entre ellas: occupancy grids, probabilistic occupancy grid.

Estas estrategias difieren de la manera en cómo se actualiza la celda. Inicialmente lo propuso [30], posteriormente se estableció un enfoque bayesiano, y actualmente [20] realiza el mapeo en referencia a una ocupación máxima posteriori.

Entre las ventajas de estos mapas es que se construyen de forma accesible al recorrer el entorno, debido a que se establecen las celdas de acuerdo al terreno, por lo cual el robot puede determinar su posición y orientación en el entorno.

Se utiliza en el proyecto teorías bayesianas en los mapas de rejillas. Para lo cual utilizamos el teorema de Bayes que utiliza la probabilidad condicional, teorema de la probabilidad total.

### 3.20.2. Generación del mapa con el sonar

El occupancy grid se puede construir por varios tipos de sensores entre ellos el ultrasónico. A pesar de que otros tienen mejores resultados como el láser.

Se divide el entorno en un conjunto de celdas uniformes

El robot NAO con sus dos sensores ultrasónicos, caminará horizontalmente sobre el entorno o paredes. Este sensor detecta de que hay algún obstáculo en cierta distancia El movimiento se hace paso a paso por el entorno. Los mapas de celdas de ocupación requieren mucha memoria, pero se utiliza en el proyecto debido a que el entorno es pequeño, controlado y con objetos simples de detectar. A pesar de los efectos de múltiples reflexiones con un objeto y de que varias lecturas no corresponden a las reales se lo selecciona por ser el disponible en el robot y por los objetivos del proyecto de no utilizar sensores externos.

Detalle del diseño

| Elemento | Dimensiones |
|---|---|
| Baldosas | de 0.3 x 0.3 m |
| Obstáculos (cartones) | 0.3 ancho x 0.4 largo x 0.40 alto |
| Área de mapeo | 4 x 4 |

El lado de 0.4 será horizontal a la pared. El lado de 0.3 será vertical. Los ejes se establecen X hacia arriba x, a la derecha y. Esto se muestre en el diseño.

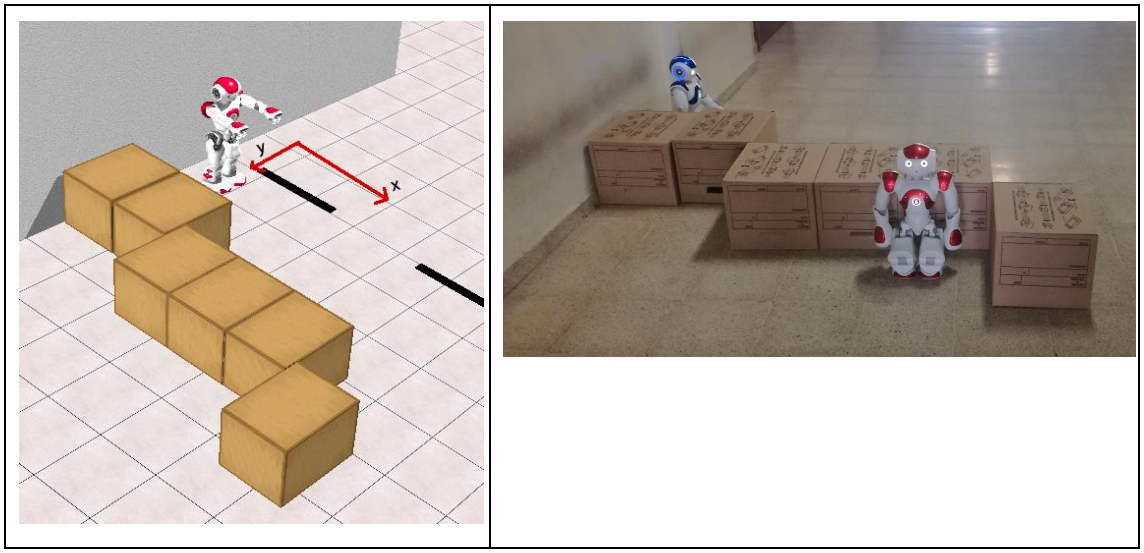



Figura 60. Diseño para la detección con el sensor ultrasónico

El modelo se indica a continuación [49]:

Sea $H_1, H_2, H_3, \dots H_k$ hipótesis mutuamente exclusivas que explican un suceso E. Para obtener la probabilidad condicional de una hipótesis $H_i$ se aplica el teorema de bayes

$$\mathrm{P(H_i|E)}=\frac{\mathrm{P(E|H_i)\,P(H_i)}}{\sum_{n=1}^{k}\mathrm{P(E|H_n)P(H_n)}} \text{ donde } \sum_{i=1}^{k}\mathrm{P(H_i)}=1 \quad (43)$$

Recordando que la probabilidad máxima es 1, para utilizar el teorema de bayes se debe cumplir

$$P(H|ocupada)+P(H|libre)=1 \quad (44)$$

Aplicado al sensor de ultrasonidos, con el cual medimos una distancia r al detectar un obstáculo, con un ángulo de apertura $\theta$ en el rango $[\theta_1, \theta_2\,]$ tal como lo muestra la y considerando que las celdas desconocidas son de 0.5, 0 las no detectadas y las que tienen un alto probabilidad de ocupación se tiene que es

$$P(H|ocupada)=\begin{cases}0.5+0.5k_d & si\ r\epsilon[r_0, r_0+\varepsilon\,]\ y\ si\ \ \theta\epsilon[\theta_1,\theta_2\,]\\ 0 & si\ \ r<r_0\ y\ \ \theta\epsilon[\theta_1,\theta_2\,]\\ 0.5 & otro\ caso\end{cases} \quad (45)$$

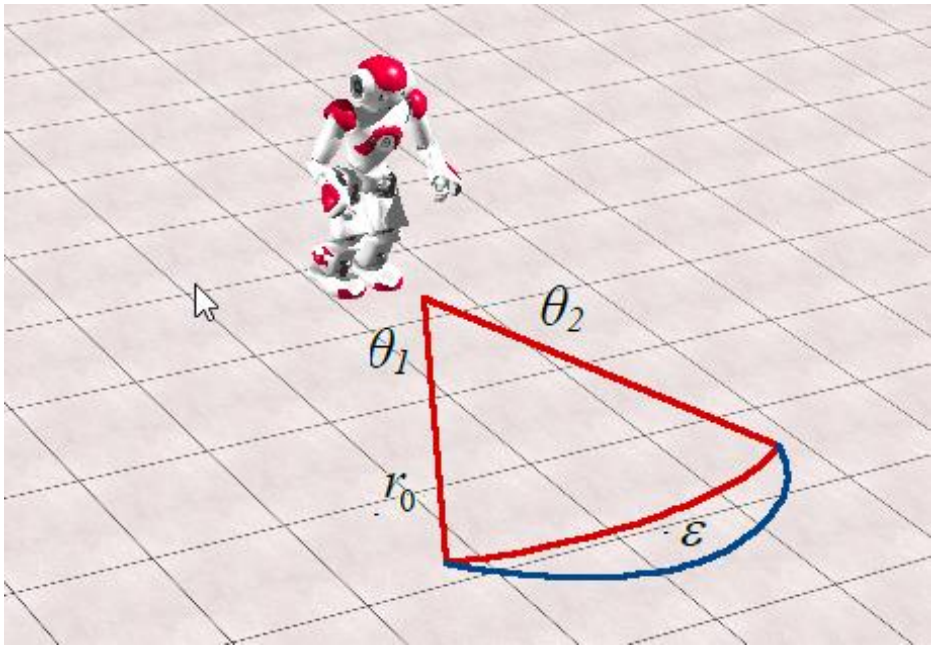



Figura 19 Mapa de rejilla y probabilidad de ocupación

Aplicando el teorema de Bayes de forma recursiva suponiendo que se conoce la probabilidad a Priori, o que se tiene antes de la medida. Con esto se actualiza el mapa

$$P(ocupada)^t = P(ocupada|H)^{t-1} \tag{46}$$

Se obtiene

$$\mathrm{P}(ocupada|\mathrm{H})^t = \frac{\mathrm{P}(\mathrm{H}|ocupada)\mathrm{P}(ocupada|H)^{t-1}}{\sum_{\mathrm{n}=1}^{\mathrm{k}} \mathrm{P}(\mathrm{H}|\mathrm{E_n})\mathrm{P}(\mathrm{E_n}|H)^{t-1}} \tag{47}$$

Con valores iniciales de $\mathrm{P}(E_j|\mathrm{H})^0 = \mathrm{P}(ocupada|\mathrm{H})^0 = \mathrm{P}(libre|\mathrm{H})^0 = 0.5$

### 3.21. Planificación de trayectorias-cognición Path Planning

Existen diferentes algoritmos para planificar la trayectoria, BreathFirhst, D*, Se utiliza el A* A start. La meta es minimizar el costo estimado total de la solución. Es una forma ampliamente conocida de la búsqueda primero el mejor.

**Navegación**

Se establecen las siguientes formas de navegar:

- Manejo teleoperado -avanzar retroceder girar
- Navegación autónoma por el mapa fijo
- Seguimiento de personas y objetos
- Navegar en un lazo o bucle y evitar colisionar con el objeto.

**Navegación a través de un mapa. navegación autónoma**.

Para mover el robot a una localización determinada el robot necesita: planear caminos sin obstáculos y seguir esos caminos. Del proceso de mapping se generó un mapa de celdas de ocupación o se utiliza un mapa ya cargado. Debido a que el robot opera en un entorno dinámico, durante la navegación el robot constantemente debe adquirir información del entorno por los sensores (visión y ultrasónico), actualizando el mapa y evitando colisiones.

En la navegación de robots se utiliza la ruta generada mediante algoritmos de búsquedas. Un robot puede moverse en un espacio continuo con un infinito posibles de acciones y estados. Existen estrategias de búsqueda no informada e informadas. Las no informadas o búsqueda a ciegas (blind search), no tienen información adicional sobre los estados más que los que proporciona el problema. En cambio, las búsquedas informadas o búsquedas heurísticas conocen si un estado no objetivo es mejor que otro. Estas estrategias de búsqueda se distinguen por el orden de cómo se expanden los nodos. Las búsquedas no informadas son muy ineficientes. Las estrategias de búsquedas informadas pueden encontrar soluciones con más eficiencia. [2]. La clasificación de estas técnicas de búsquedas se indica en el anexo A.6

En el proyecto se utiliza en la solución A* (A start o A estrella) para minimizar el costo estimado total de la solución. Es una estrategia de búsqueda heurística. La solución es del tipo best-first search (búsqueda primero el mejor). La función heurística permite transmitir datos adicionales a la búsqueda.

Se indica que es un caso particular del algoritmo de búsqueda de árboles o búsqueda de grafos.

La selección se la efectúa debido a que es un algoritmo muy utilizada y ampliamente conocido en robótica.

El Proceso consiste en:

- En base a una función de evaluación f(n) seleccionar un nodo para la expansión,
  - Se utiliza por lo general el nodo con la evaluación más bajo.
  - Esta evaluación mide la distancia al objetivo.
  - Será primero el mejor siempre y cuando la función f(n) evalúe al mejor.
  - Se implementa con una lista que mantiene el orden ascendente de f-valores.
- Se tiene una función heurística h(n) que tiene el costo estimado del camino más barato desde el nodo n a un objetivo.
- Evaluar los nodos combinando: g(n): el costo para alcanzar el nodo. h(n): costo de ir al nodo objetivo. Se obtiene

$$f(n) = g(n)+h(n) \tag{48}$$

- En referencia a que g(n) es el costo del camino desde el nodo de inicio al nodo n. Y que h(n) es el costo estimado del camino más barato desde n al objetivo, se tiene
  - f(n): costo más barato estimado de la solución a través de n

De esta manera se intenta encontrar la solución más barata. Por eso se efectúa primero con el nodo más bajo de g(n) + h(n). Si la función heurística satisface ciertas condiciones, la búsqueda de A* es completa y óptima.

Una vez planeado el camino se procede a enviar los comandos de locomoción. Se indica que se establece un umbral de seguridad seguridad= 50 cm, para que el robot no colisione con el objeto.

Se establece un nivel de celdas de 30 cm x 30 cm. Donde cada paso del robot está entre maxStepX y minStepX. De forma aproximada cada movimiento de 10 cm se genera por dos pasos de cada pie. primero el pie izquierdo por 5 cm maxStep, luego derecho con minStep.que es la referencia, luego el izquierdo otro 5cm y por último el derecho que iguala al nivel

Se efectuaron dos pruebas con los siguientes entornos

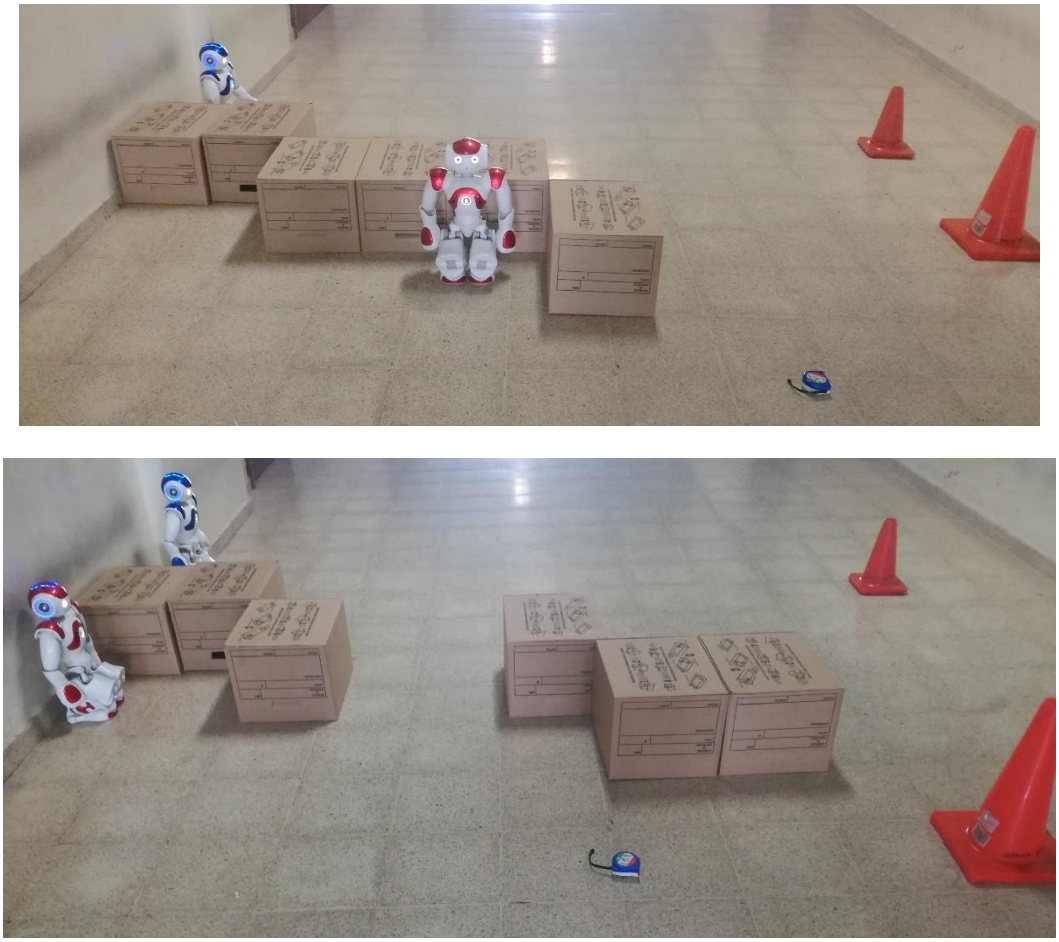


Figura 61. Navegación con mapa

### 3.21.1. Navegación sin mapas

Se realiza la navegación sin mapas con el manejo teleoperado -avanzar retroceder girar, la navegación autónoma por el mapa fijo, el seguimiento de personas y objetos, así como el Navegar en un lazo o bucle y evitar colisionar con el objeto.

### 3.21.2. Navegación a un punto objetivo

Se requiere ir a un punto objetivo. Se utiliza el planificador de trayectoria A*, con el camino a seguir al punto solicitado.

## 3.22. Seguimiento o Tracking

El tracking o seguimiento es Localizar un objeto en sucesivos frames de un video. [15]. En el trabajo se realizan pruebas con los algoritmos de tracking BOOSTING, MIL, KCF. Hay varios algoritmos que permiten seguir o detectar objetos.

Flujo Óptico Denso. Estima el vector de movimiento de cada pixel en el frame.

Flujo Óptico Escaso. Usa el seguidor de características Kanade-Lucas-Tomashi (KLT). Sigue la ubicación de pocos puntos característicos en la imagen.

Filtro de Kalman.Utilizado para predecir posiciones de un objeto que se mueve en base a información a priori.

Meanshift and Camshift. Localizan el máximo de una función de densidad.

Single object trackers. El primer frame es marcado usando un rectángulo que indica la ubicación del objeto que desea seguir.

Se describen los utilizados como prueba en el proyecto. Boosting Tracker. Utiliza una versión online de AdaBoost. Este clasificador requiere entrenar con positivos y negativos del objeto. MIL. Es similar a Boosting, pero con la diferencia que considera solo la actual ubicación del objeto como un positivo. KCF Tracker. Significa Kernelized Correlation Filters. Utiliza las ideas de MIL y Boosting tracker. Considera que las múltiples muestras positivas utilizadas en MIL tienen grandes regiones sobrepuestas (overlapping regions)

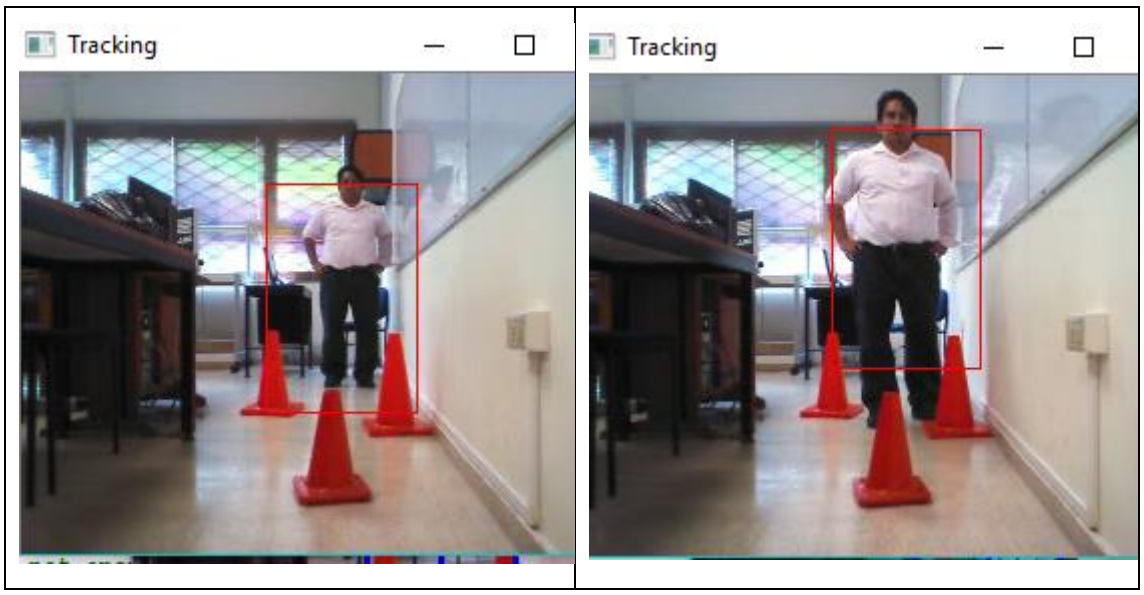



Figura 62. Tracking a un video, desde la cámara de NAO

## 4. Diseño de la aplicación

La aplicación se realiza con Python, pyqT5, OpenCv, matplotlib, numpy mediante la distribución miniconda.

**Prerrequisito para el funcionamiento**: mínimo el robot virtual para efectuar la conexión.

El diseño de la aplicación denominada Control NAO V1.0 es:

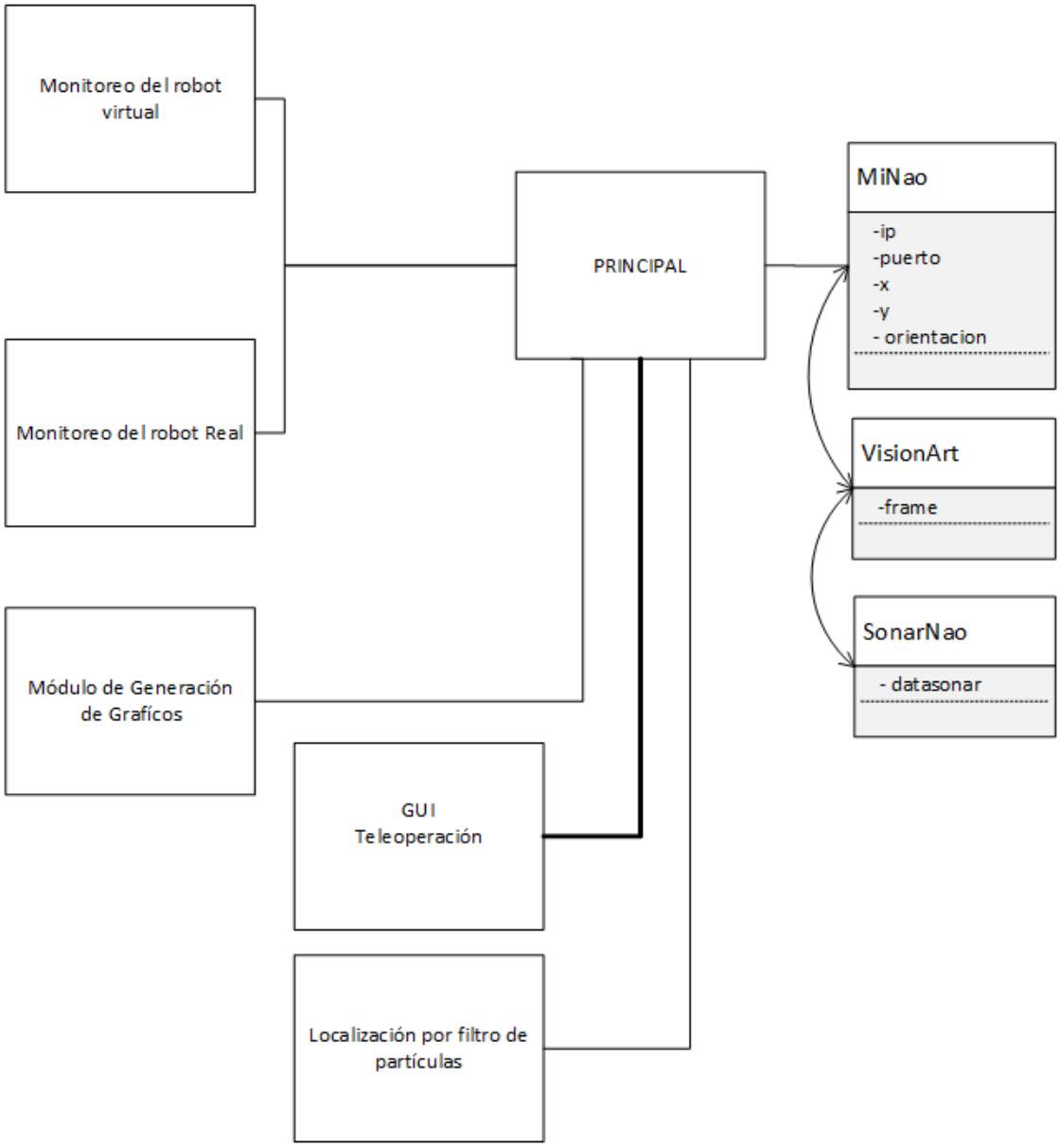



Figura 63. Diseño de la aplicación desarrollada

Está estructurada en las siguientes GUI (Graphics User Interfaces):

- De Teleoperación.
- Localización mediante filtro de partículas.
- Monitoreo del robot virtual.
- Monitoreo del Robot Real la imagen original, detección y descriptores.
- Módulo de generación de gráficos.

El usuario para ingresar a algunos de estos módulos debe iniciar desde Python, o en su defecto el IDE.

```
Ir a la ruta donde está el proyecto
Consola de comandos
Python Principal.py
```

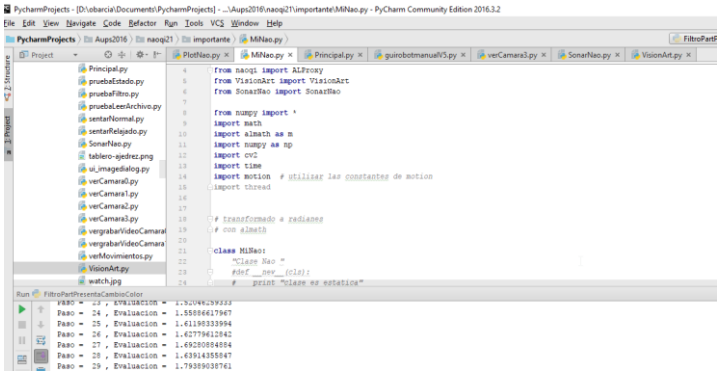


Figura 64. Ejecución de la aplicación.

Se describe en la sección siguiente:

### 4.1. GUI de Teleoperación.

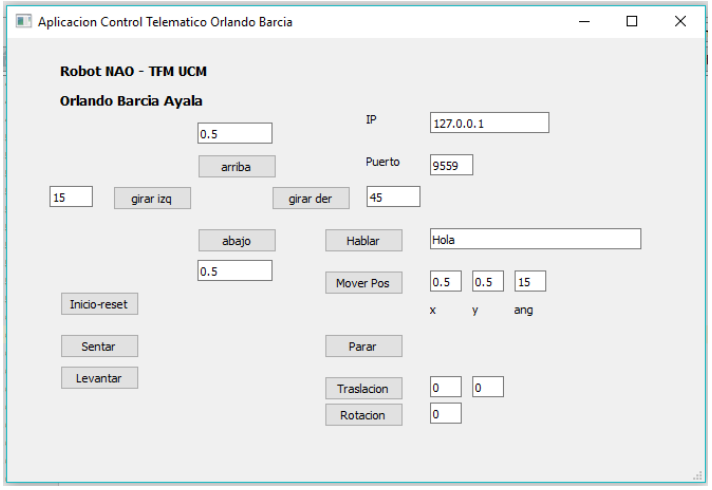



Figura 65. GUI de la aplicación de control Telemática.

## 4.2. Interfaz de visualización del objeto Real

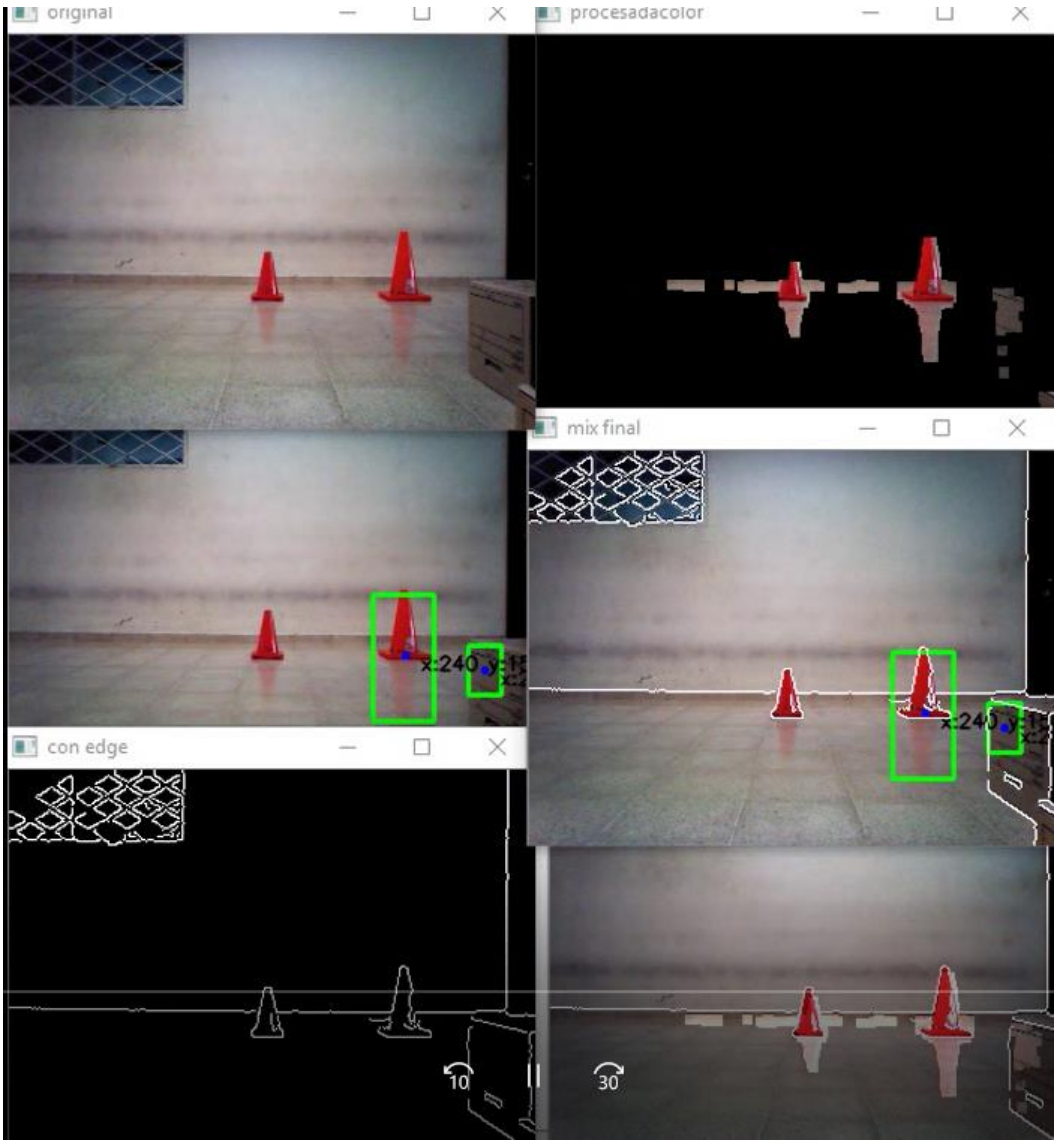
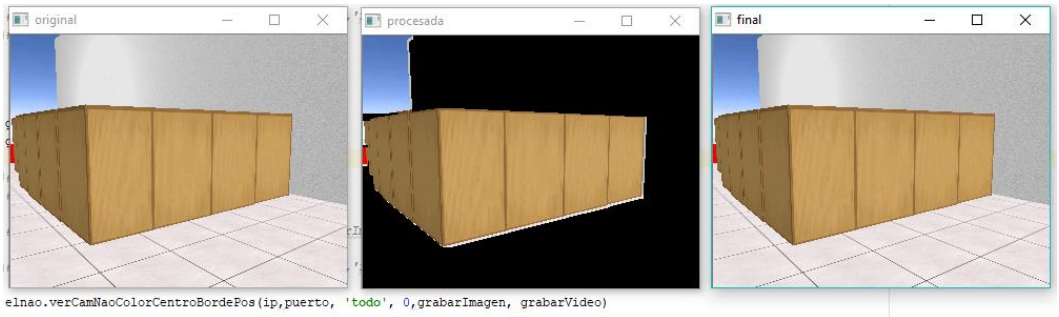



Figura 66. Visualización del procesamiento de las capturas realizadas por el robot.

## 4.3. Interfaz de visualización del robot Virtual



### 4.4. Módulo de visualización del filtro de partículas

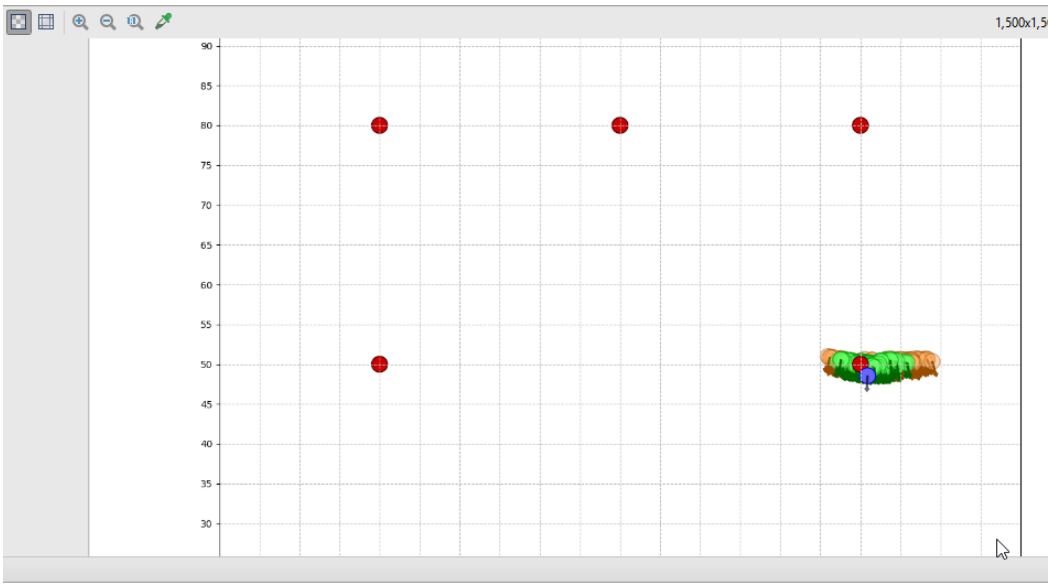



Figura 67. Visualización del filtro de partículas.

### 4.5. Módulo de generación de gráficos.

Cada procesamiento de imagen o pasos realizados por el robot se almacenan en el computador para posterior análisis. Cada una se guarda en una carpeta de acuerdo al procesamiento a efectuar son Imágenes originales, máscara, descriptores, posiciones de movimiento, sonar. Este módulo realiza la lectura de los datos con Python, procesa con numpy y muestra con matplotlib.

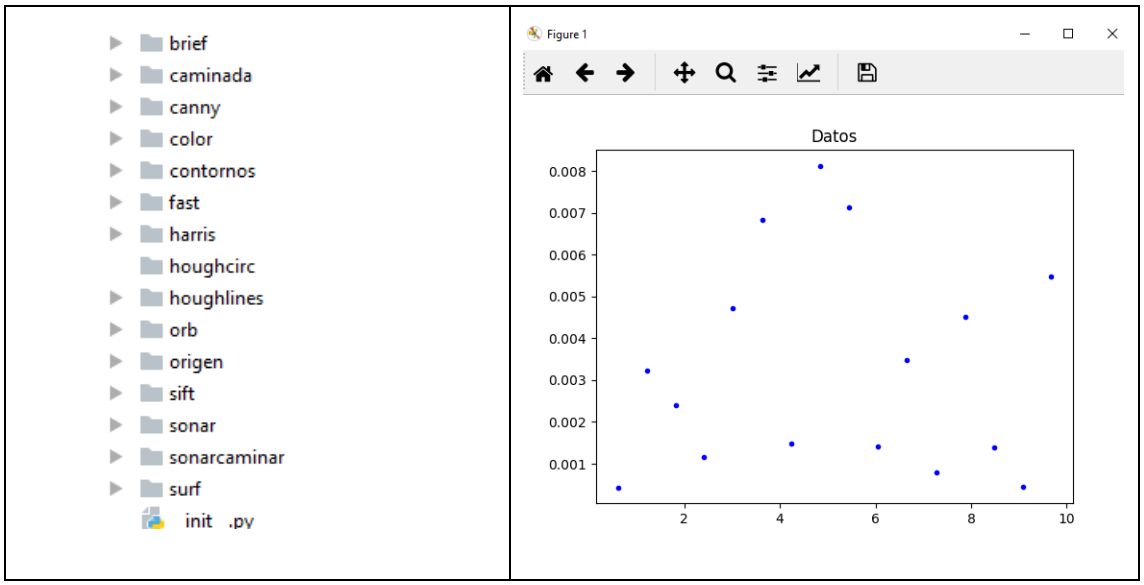



Figura 68. Visualización de posiciones x-y almacenadas.

### 4.6. Diagramas de clases

Las clases creadas son: MiNao,VisionArt, SonarNao, PlotNao.

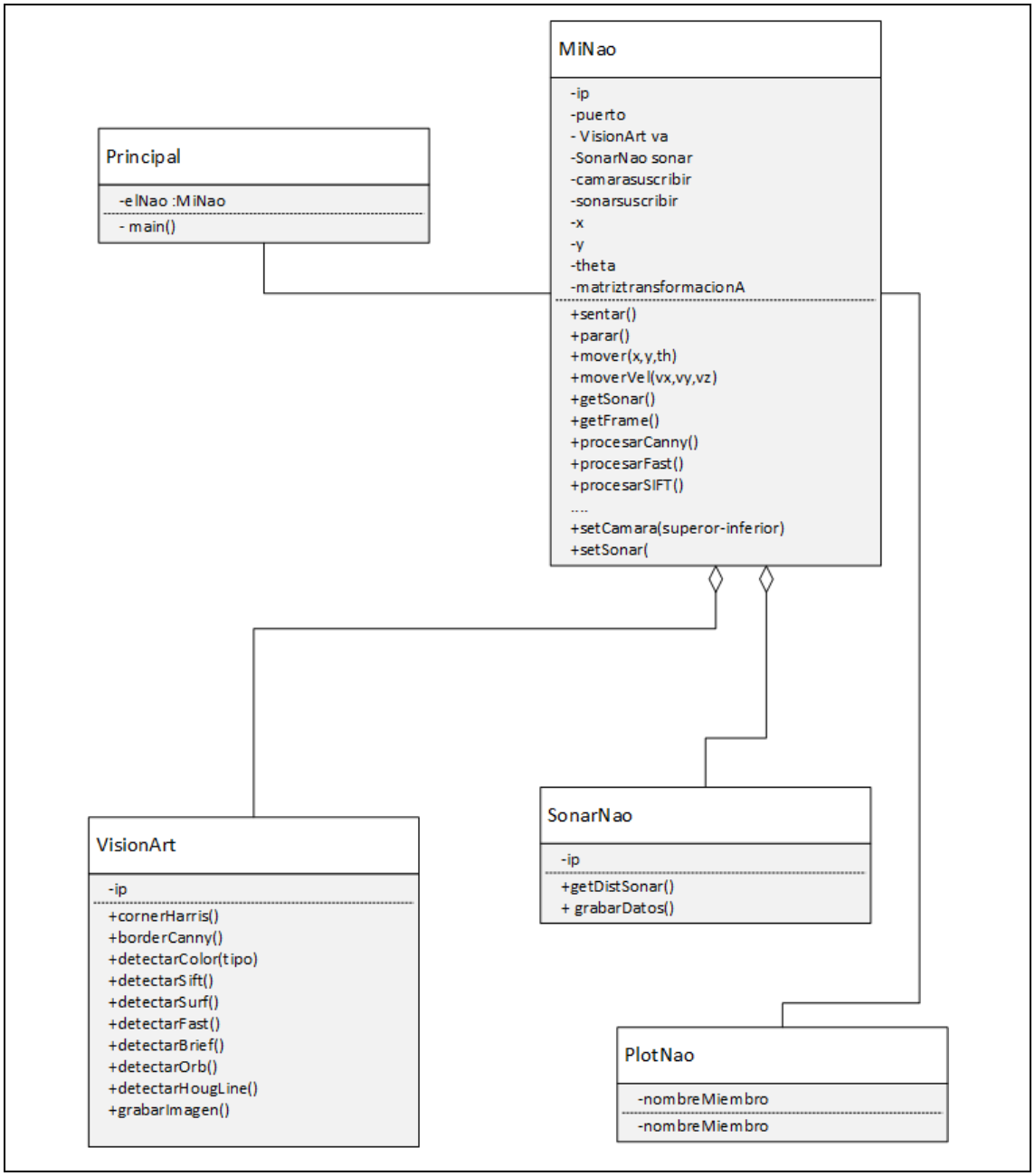



Figura 69. Diagramas de clase de la aplicación

### 4.7. Diagramas de caso de uso.

Los Diagramas de caso de uso se muestran en la Figura 70

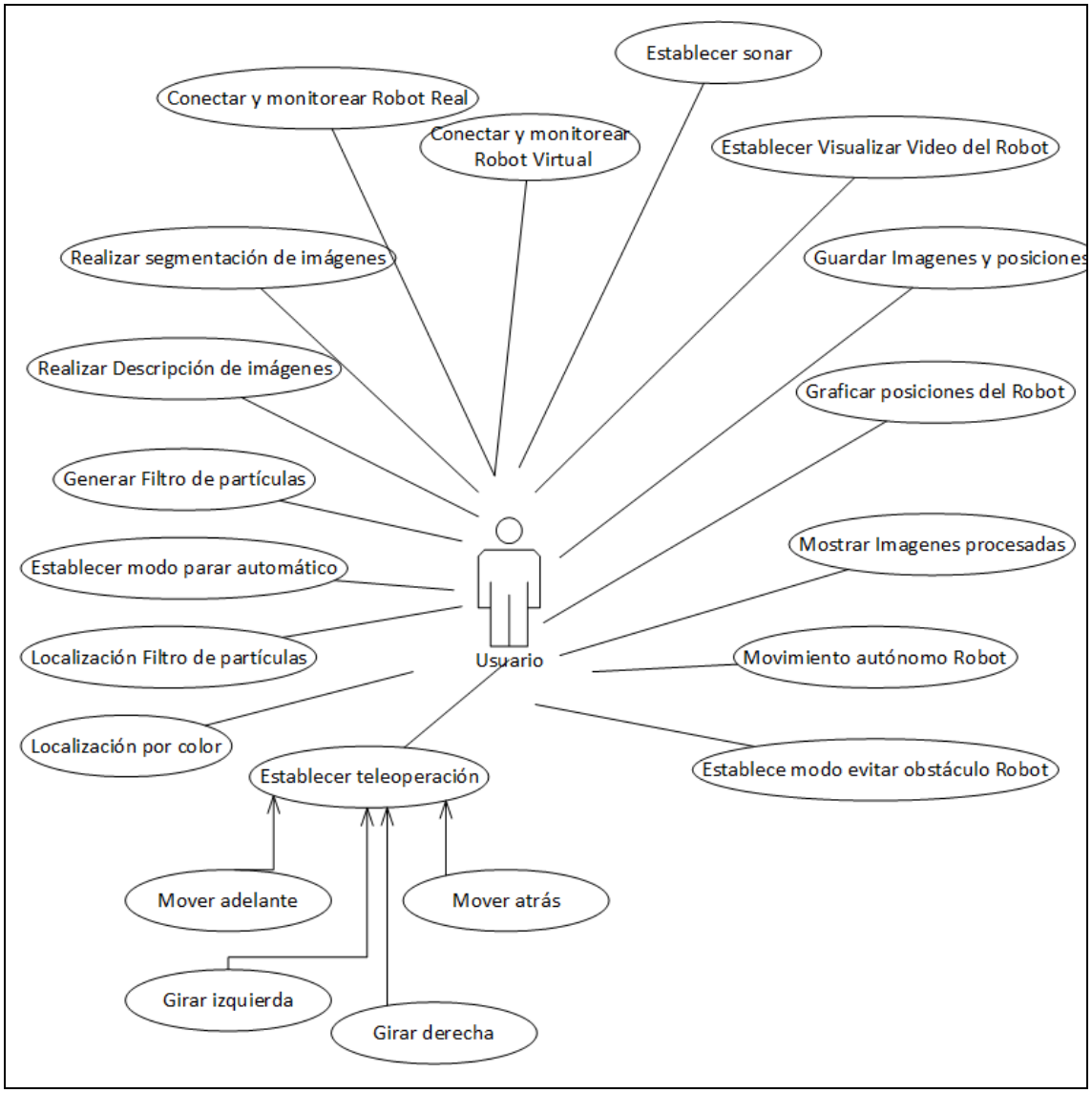



Figura 70. Diagrama de caso de uso

# 5. Resultados

Los resultados están acompañados de los siguientes videos:

## 5.1. Robot Nao, características y simulador Webots

**Escenario 1:** Presentación con audio y movimiento del robot NAO.

**Resultados:** El robot inicia en una posición fija, se levanta, procede a saludar utilizando las articulaciones hombro (shoulder), codo (elbow) y muñeca (wrist) derecha, mueve la articulación (head) mientras emite sonidos audibles de la presentación del TFM a los docentes de la UCM-UNED. Posteriormente se procede a sentar y parar.

**Video**: www.orlandobarcia.com/presentacion-nao-ucm-uned-hablado

---

**Escenario 2**: Proceso detallado del ambiente de pruebas por simulación.

**Resultados:** Se diseña el entorno de pruebas (laboratorios y pasillo de la Universidad Politécnica Salesiana) mediante el software simulador Webots. En este entorno, el robot virtual será sometido a pruebas.

**Video**: http://www.orlandobarcia.com/simulacion-de-un-entorno-real-para-nao-en-webots/

---

**Escenario 3**: Descripción del robot NAO y Webots entorno de modelado.

**Resultados:** Se realiza una descripción del robot y muestran los movimientos del robot real y robot simulado utilizando el entorno Webots.

**Video**: http://www.orlandobarcia.com/robot-bipedo-nao-y-webots-como-entorno-de-modelado/

---

## 5.2. Navegación

**Escenario 4**: Locomoción y navegación a lazo abierto con detección de sonar.

**Resultados:** El robot se mueve en el pasillo de los laboratorios de la Universidad Politécnica Salesiana. Navega en un lazo infinito, en el caso de detectar obstáculo que aparece de improviso, lo esquiva.

Figura 71. Robot frente a obstáculos que logra evitar.

En el robot se establece una zona de seguridad de 0.20 m. En el caso de detectar al caminar, el robot se detiene. Posteriormente gira 15º y con el sonar verifica si no hay obstáculo, con el cual continua 0.10 m hacia adelante. Sino hay obstáculos el robot continúa caminando. Debido a que el robot mientras camina tiene una función de bloqueo, dentro del bucle de detección de caminada si llega a tener uno de los sensores menor a 0.20 m el detiene según las especificiaciones indicadas. El cono del sonar es de 60º.

**Video**: www.orlandobarcia.com/video-sonar-deteccion

---

**Escenario 5:** El robot detecta el movimiento de la cara de una persona y lo sigue por un tiempo de 30 s.

**Resultados**: Navegación sin mapa. Una persona está frente al robot. Una vez detectado por contorno y área del elemento lo procede a seguir. Tiempo de seguimiento 30 segundos. Al inicio el robot debe capturar el área del contorno de la figura a detectar por un tiempo de 5 segundos. En caso de no detectar el robot espera por 10 segundos, sino finaliza el proceso. De 10 intentos, 9 fueron positivos y procedió a efectuar el seguimiento.

**Video**: www.orlandobarcia.com/seguidor-cara

---

**Escenario 6**: El robot detecta el área, contorno y centroides de una bola de color rojo de radio 6 cm. Posteriormente realiza el seguimiento por un tiempo de 30 s.

**Resultados**: Navegación sin mapa. Una persona tiene una bola roja de diámetro 6 cm. Se realiza el seguimiento de una persona que contenga la bola en base a detección de contorno, área y color. Se realiza el seguimiento durante 30 segundos. Al inicio el robot debe capturar el área del contorno y color de la figura a detectar por un tiempo de 5 segundos. En caso de no detectar el robot espera por 10 segundos, sino finaliza el proceso. De 10 intentos, 10 fueron positivos y procedió a efectuar el seguimiento.

**Video**: www.orlandobarcia.com/video-seguidor-bola-roja

---

**Escenario 7:** El robot se mueve en base a un mapa ya cargado (Mapa 1).

**Resultados**: Se efectuaron pruebas en tres entornos con mapas cargados en memoria, pero de diferente distribución. En el mapa 1 inicia el movimiento en línea recta tomando como referencia el cono de seguridad naranja, gira a la derecha, y continúa en línea recta hasta el punto final (un flexómetro en el piso). Utiliza el algoritmo A*.

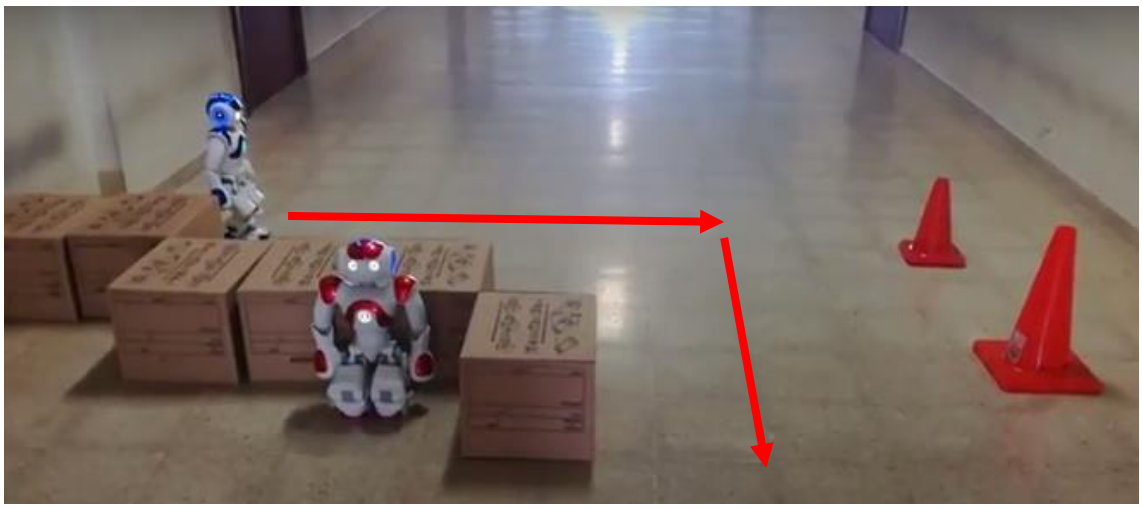


Figura 72. Entorno para navegación mapa 1.

**Video** Navegación con mapa 1. www.orlandobarcia.com/navegacion-pasillo-1

---

**Escenario 8:** El robot se mueve en base a un mapa ya cargado (Mapa 2).

En el mapa 2, se realiza similar proceso. El landmark continúa como referencia, pero mas distante de la posición de giro, el obstáculo tiene un pasillo interno. El robot gira a la derecha y continúa en línea recta hasta el objetivo. Utiliza A*.

Figura 73. Entorno para navegación mapa 2.

**Video**: Navegación con mapa 2. www.orlandobarcia.com/navegacion-pasillo-2

---

**Escenario 9:** El robot se mueve en base a un mapa ya cargado (Mapa 3).

**Resultados**: En el mapa 3, hay 3 landmark visuales (conos de seguridad). El mapa está cargado, al ir al segundo cono de seguridad gira a la derecha.

**Video**: Mapa 3 www.orlandobarcia.com/girar-derecha

---

**Escenario 10:** Navegación en entorno de interiores (Laboratorio de automatización).

**Resultados**: En un entorno interior, sin landmark y con el mapa cargado, el robot navega en U.

**Video**: www.orlandobarcia.com/navegacion-lab-auto

**Escenario 11:** Navegación en entorno de interiores (Laboratorio de automatización) real y simulada.

**Resultados**: En un entorno interior, sin landmark y con el mapa cargado, el robot navega en L invertida. Se compara la navegación con el simulador webots, de forma detallada.

**Video:** www.orlandobarcia.com/navegacion-lab-flexible

### 5.3. Detección y descripción de imágenes y/o video

**Escenario 12:** Detección de colores y formas en el piso por áreas, momentos y rango de color y movimiento en base a su posición.

**Resultados**: El robot detecta varias figuras de colores en base a umbral y va a un punto determinado. Se describe en el video la detección y movimiento en el color verde y las figuras por color rojo en el robot real y el simulado. El algoritmo requiere el color que desea detectar. Posteriormente se selecciona el rango de colores de acuerdo al requerimiento, se efectúa la binarización y segmentación de la imagen. Se obtiene el centroide de la primera imagen más cercana. De acuerdo a la relación establecida en 3.16.1 Coordenadas espaciales. Para la cámara inferior del Robot, el robot camina al punto dado. Marca el landmark y espera hasta detectar otro punto.

**Video**: www.orlandobarcia.com/deteccion-colores-real

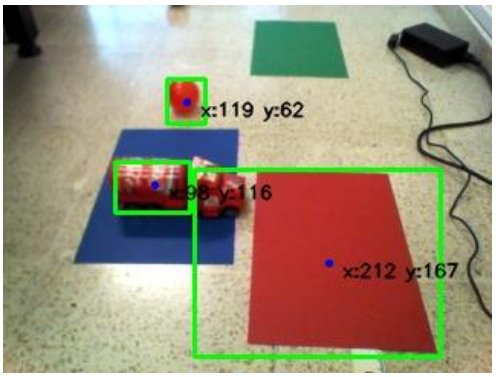



Figura 74. Detección de tres figuras rojas. Se mueve al primer punto encontrado.

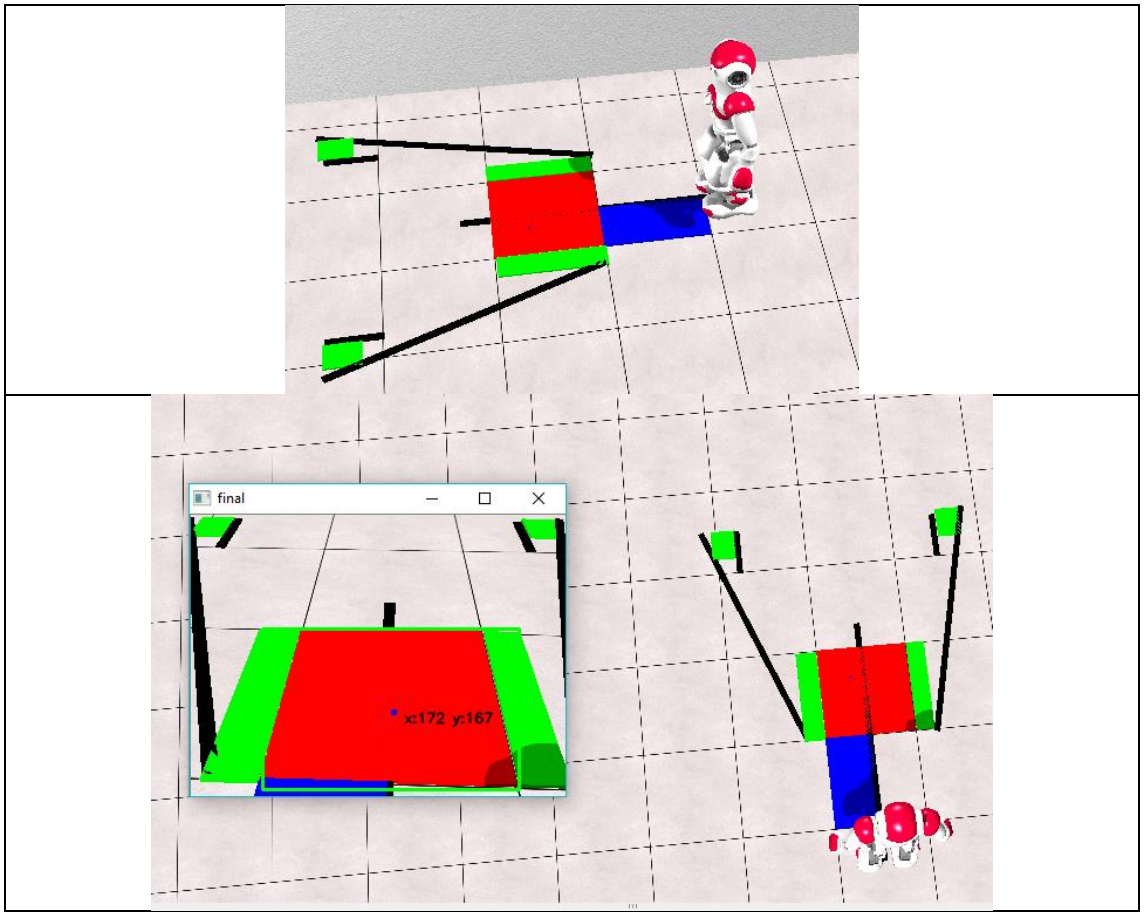
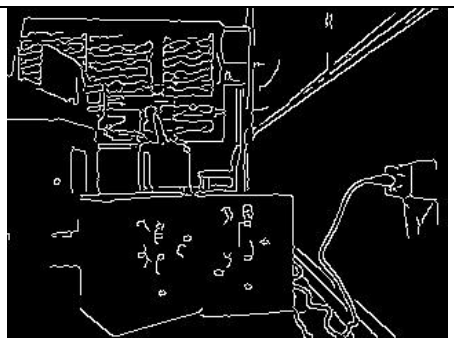


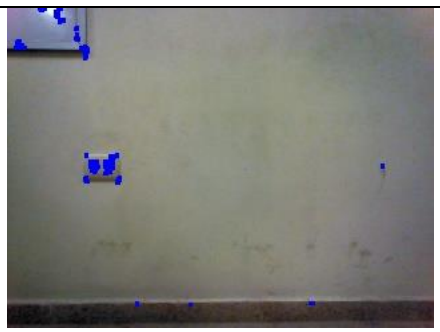

Figura 75. Detección de colores por simulación.

**Escenario 13:** El robot frente a los objetos, detecta y describe objetos como bordes y esquinas.

**Resultados**: Se realiza la detección y descripción de esquinas (corners), bordes.



bordes



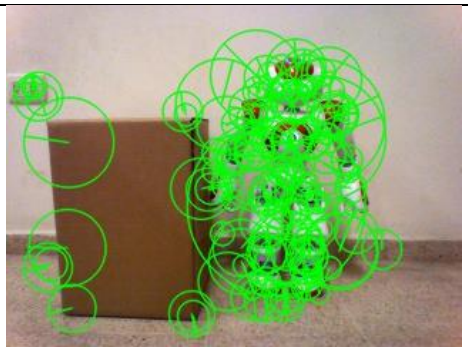
esquinas (corners)

Figura 76. Detección de esquinas y bordes.

**Escenario 14.** Descripción de objetos uso de SIFT, SURF, FAST, BRIEF, ORB.

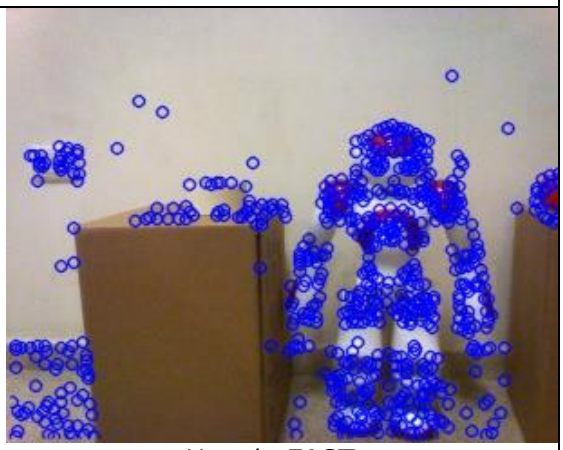
**Resultados:** El robot frente a los objetos, aplica SIFT, SURF, FAST, BRIEF y ORB para descripción de objetos.

Uso de SIFT.



Uso de SURF.

Uso de BRIEF.



Uso de FAST.

Uso de ORB.

Figura 77. Aplicación de SIFT, SURF, FAST, BRIEN, ORB

### 5.4. Locomoción y localización

**Escenario 15:** Localización exacta, no siempre posible.

**Resultados**: Al mover el robot en lazo abierto, por ruido en las articulaciones no se ubica en la posición estimada.

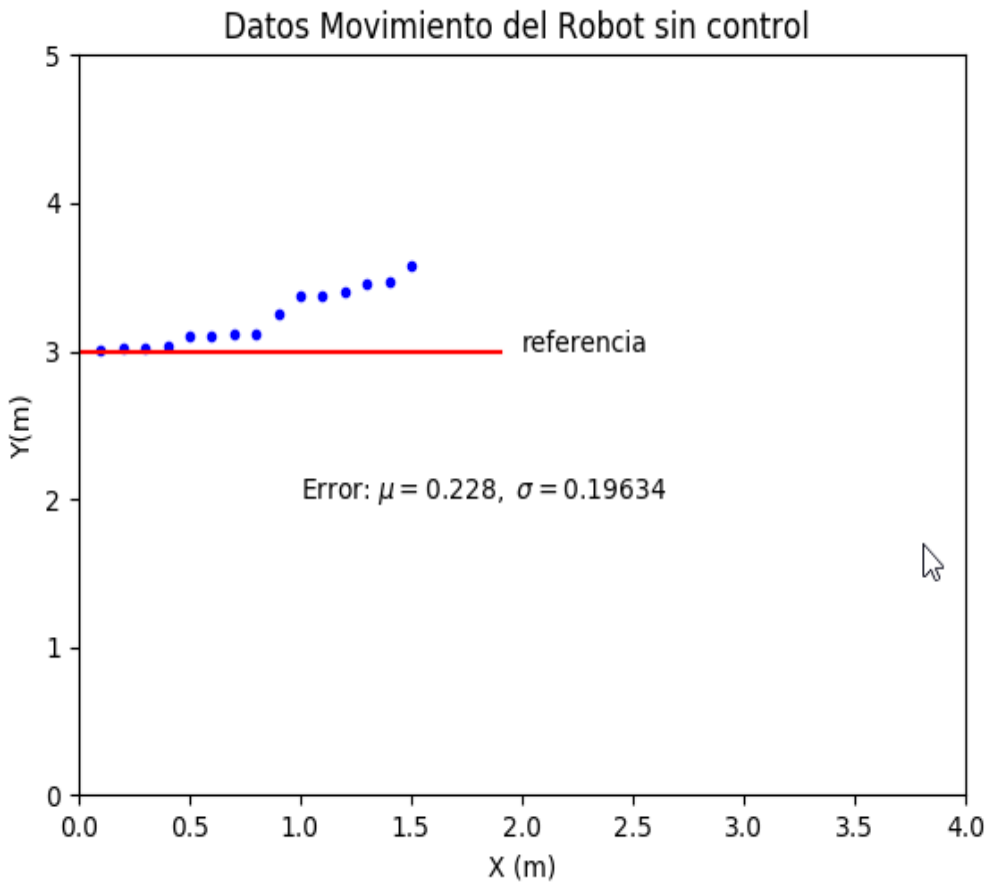



Figura 78. Movimiento x vs y del robot sin control.

Realizando una trayectoria recta x=0 y a paso de 0.1 m, con y =3. Se interpreta los resultados que el robot se mueve por el ruido en las articulaciones al lado izquierdo en aproximadamente 0.33 m. El error es de $e = 0.228 \mp 0.19634$.

**Escenario 16:** Control a lazo cerrado.

**Resultados**: Controla la locomoción a una posición establecida con los datos de la estimación por filtro de particulas en un espacio de 1.5 m en línea recta.

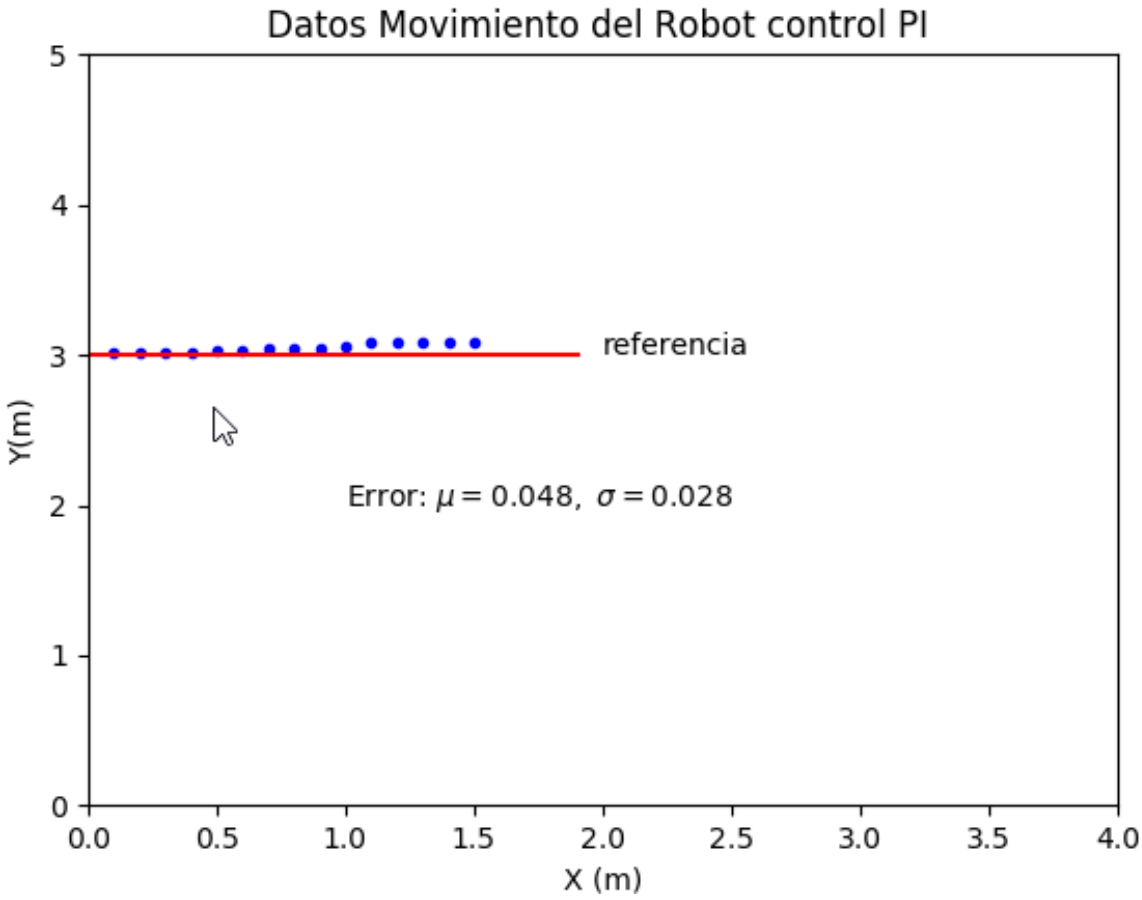



Figura 79. Movimiento x vs y del robot con control.

Se interpreta que el valor de error es $e = 0.048 \mp 0.028$**.** Se utiliza en el control un yr de 3. Si el error $e = y_{R-}y_{es} > \varepsilon$ entonces $U = k_p\left(e + \frac{T}{T_i}\sum e\right)$

PI $K_p = 0.78$ , $T_i = 5.4$ y el control U de 0.1

**Escenario 17**: Detección de distancias de sonar frente a una pared.

**Resultados**: Se procede a medir la distancia del sonar del robot, frente a una pared frontal.

| sonar derecho | distancia real | diferencia |
|---|---|---|
| 0.34 | 0.3 | 0.04 |
| 0.64 | 0.6 | 0.04 |
| 0.94 | 0.9 | 0.04 |
| 1.23 | 1.2 | 0.03 |
| 1.53 | 1.5 | 0.03 |
| | promedio | **0.036** |
| | Desviación Estandar $\sigma$ | **0.00547723** |

Se interpreta que el valor de error es $e = 0.036 \mp 0.0054$

Figura 80. Escenario para medición de la distancia generada por el sonar.

**Escenario 18**: El robot se mueve por un entorno con el primer descriptor, localiza y navega.

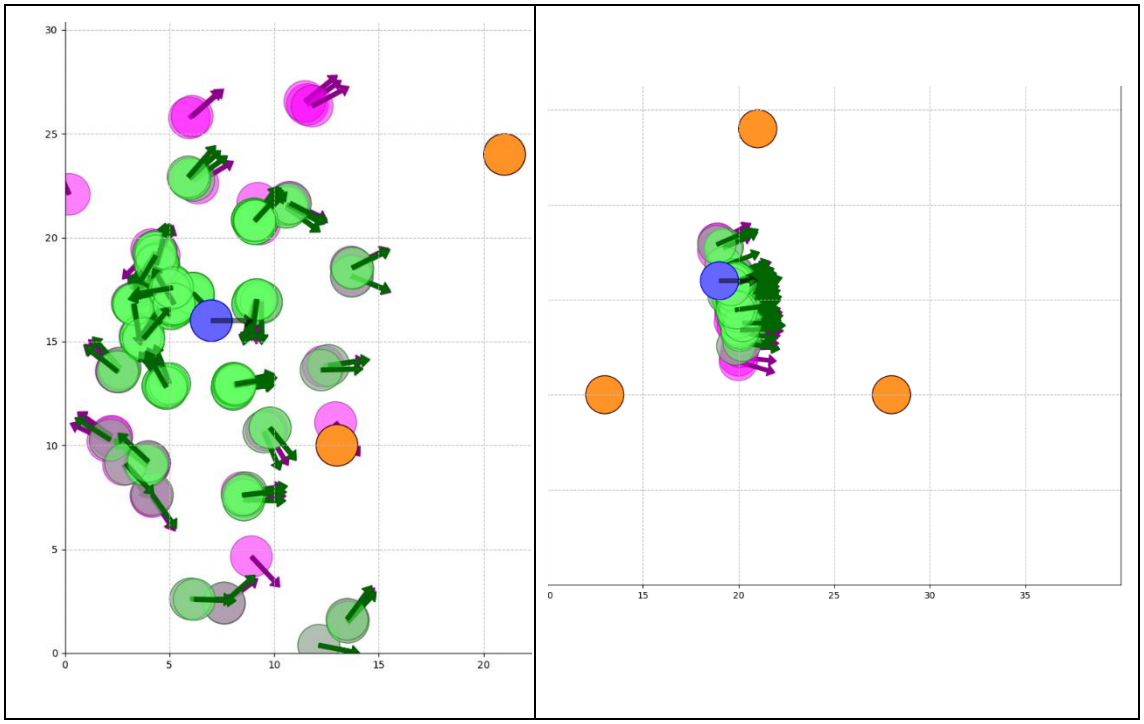
**Resultados**: Se utiliza el procedimiento de 3.16.2 Localización por Matching o emparejamiento. En 5 pruebas realizadas el 90% aproximadamente cumplia las localizaciones esperadas al pasar por los obstáculos de forma lateral (por ejemplo a 1.25 m calculado el robot estaba a 1.13 m ). Se consideran condiciones ideales, entre ellos tres balizas naturales fijas. Sus posiciones absolutas conocidas y dimensión del piso para estimar movimiento, que se actualiza por la localización del filtro de partículas.

**video**: http://www.orlandobarcia.com/deteccion-obj-vision/

**video**: www.orlandobarcia.com/localizar-descriptor

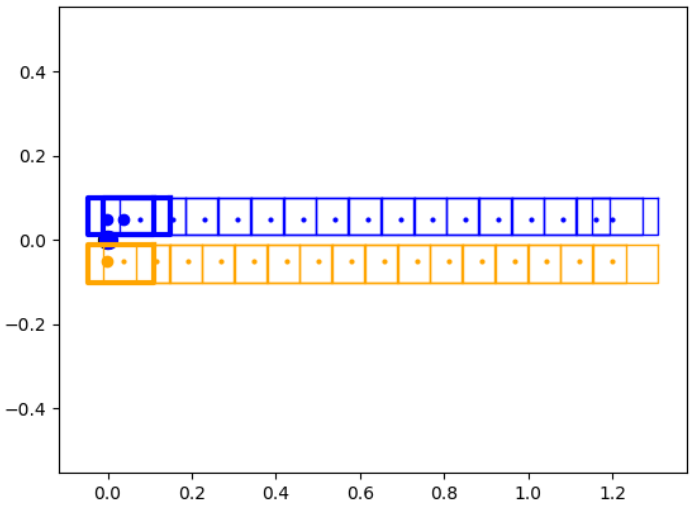
**Escenario 19**: localización mediante filtro de partículas.

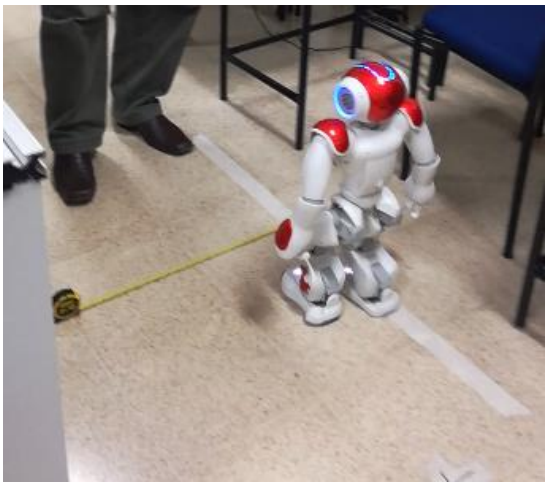
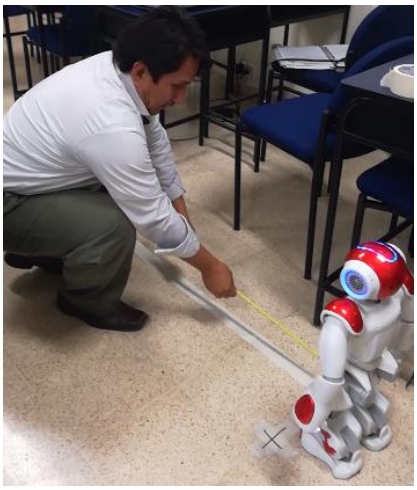
**Resultados**: En un área de 4x4 m. El robot camina en línea recta a lado de los obstáculos. La aproximación a la posición real es del 90%, aproximadamente 10 cm de diferencia entre la posición real y la calculada.



**Escenario 20:** Locomoción a diferentes puntos del entorno de prueba.

**Resultados**. El robot se mueve de forma frontal a 1.2 m. Este comportamiento se da al inicio de encendido del robot. X=1.2, y = 0 m.



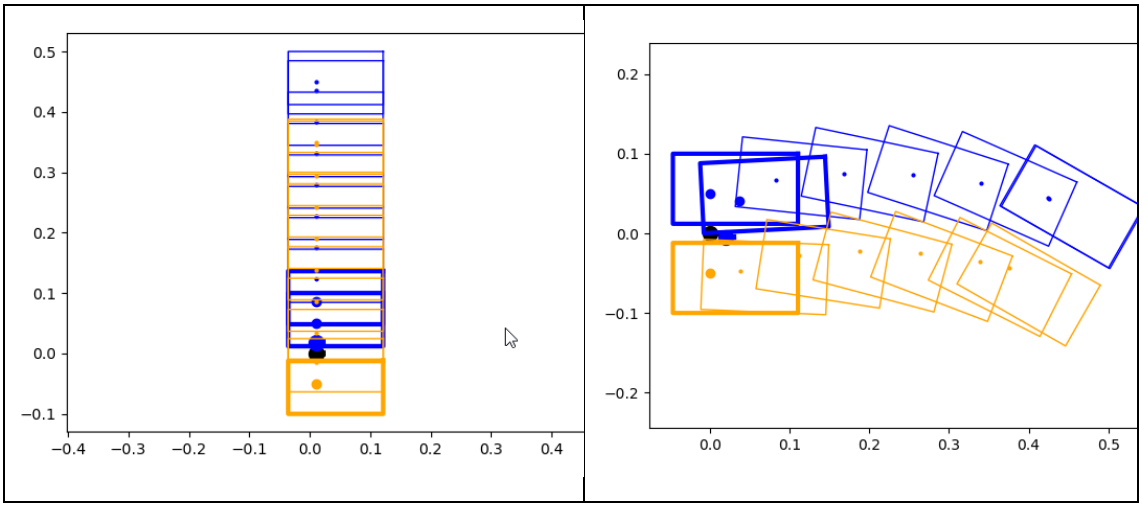
| | |
|---|---|
|  |  |
| Experimento movimiento y – línea recta | Medición de la coordenada para verificación de posición |

**Escenario 21:** Movimiento en eje y, movimiento rotacional.

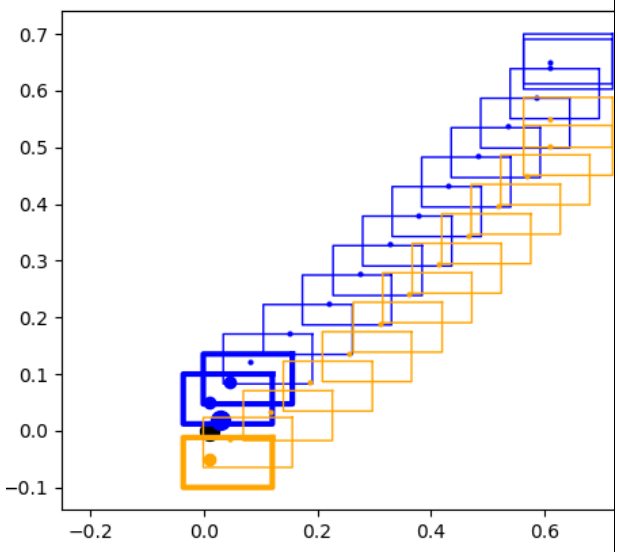
**Resultados**: Se procede a efectuar tres movimientos. El primero solo en el eje y. El segundo en x, rotando con movimiento angular de – 30 grados. El tercero movimiento lateral.

Movimiento x = 0, y =0.5, Angulo 0. Ángulos en radianes, distancia en metros.

Movimiento x = 0.4, y =0 , ángulo de -30 grados o $-\frac{\pi}{6}$ radianes

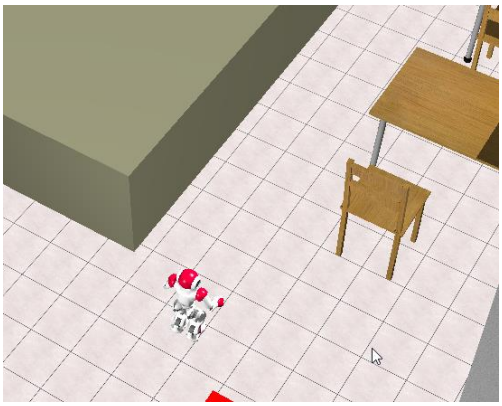



Movimiento lateral. x = 0.6, y =0.6, ángulo 0. Ángulos en radianes, distancia en metros

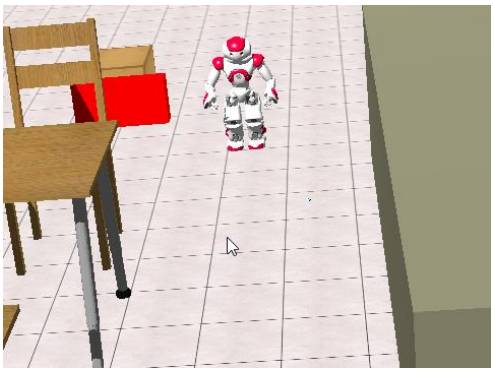



**Escenario 22.** Locomoción a Lazo abierto

**Resultados**: en un entorno de interiores, a distancias pequeñas el robot navega a lazo abierto y sin sensor sonar para protección



Vista superior.



Vista frontal.

La locomoción sencilla y movimiento de caminada.

**Video**: www.orlandobarcia.com/locomocion-lazo-abierto
**Video**: www.orlandobarcia.com/lazo-abierto-caminada

**Escenario 23**: Determinación del centro de masa.

**Resultados**: Se obtiene el centro de masea de todas las cadenas.

```
COM Body ---------- x = 0.0129609489813 y = 0.00105522957165 z = -0.0328637994826
cadenas  ['Head', 'LArm', 'LLeg', 'RLeg', 'RArm']
COM Torso ---------- x = -0.00412999978289 y = 0.0 z = 0.0434199981391
Es cadena  Head
COM cadena Head x = 0.0119333053008 y = -0.000421705743065 z = 0.168357789516
------ lista joints  ['HeadYaw', 'HeadPitch']
---------COM joint HeadYaw x = -9.99376152322e-06 y = 3.53165120259e-07 z = 0.0990799963474
---------COM joint HeadPitch x = 0.0134805496782 y = -0.000476383138448 z = 0.177332654595
Es cadena  LArm
COM cadena LArm x = 0.0203453078866 y = 0.113401390612 z = 0.0106005743146
------ lista joints  ['LShoulderPitch', 'LShoulderRoll', 'LElbowYaw', 'LElbowRoll', 'LWristYaw', 'LHand']
---------COM joint LShoulderPitch x = -5.29154058313e-05 y = 0.071369998157 z = 0.101655080914
---------COM joint LShoulderRoll x = 0.00597279984504 y = 0.107885628939 z = 0.0773750320077
---------COM joint LElbowYaw x = 0.00848005898297 y = 0.126387417316 z = 0.0267898309976
---------COM joint LElbowRoll x = 0.0196246467531 y = 0.1322298944 z = -0.0244673900306
---------COM joint LWristYaw x = 0.0474647209048 y = 0.126860797405 z = -0.0829038619995
Es cadena  LLeg
COM cadena LLeg x = 0.0192683786154 y = 0.0511901527643 z = -0.144779905677
------ lista joints  ['LHipYawPitch', 'LHipRoll', 'LHipPitch', 'LKneePitch', 'LAnklePitch', 'LAnkleRoll']
---------COM joint LHipYawPitch x = -0.010193339549 y = 0.0373198613524 z = -0.0599301718175
```

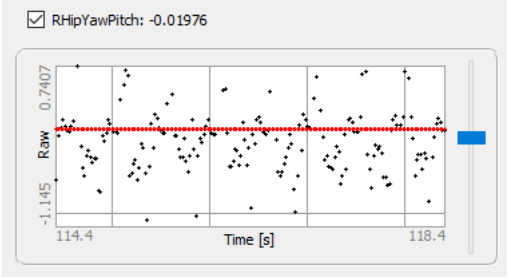
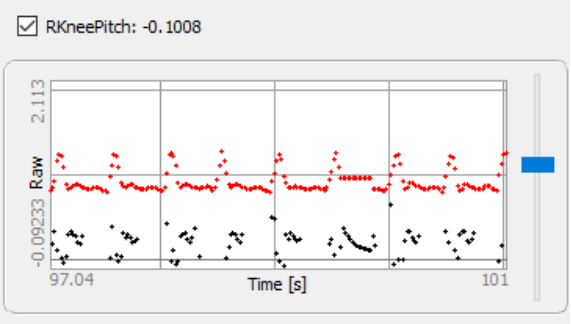
```
---------COM joint LHipRoll x = -0.0141863133758 y = 0.0475379750133 z = -0.0926963463426
---------COM joint LHipPitch x = 0.0427853390574 y = 0.0574786886573 z = -0.116736203432
---------COM joint LKneePitch x = 0.0349669866264 y = 0.0533800460398 z = -0.168149232864
---------COM joint LAnklePitch x = -0.0138547187671 y = 0.0428548566997 z = -0.176721706986
---------COM joint LAnkleRoll x = 0.00353434123099 y = 0.0480801835656 z = -0.21960195899
Es cadena  RLeg
COM cadena RLeg x = 0.00961730163544 y = -0.0466678701341 z = -0.147100076079
------ lista joints  ['RHipYawPitch', 'RHipRoll', 'RHipPitch', 'RKneePitch', 'RAnklePitch', 'RAnkleRoll']
---------COM joint RHipYawPitch x = -0.0102067664266 y = -0.0373087972403 z = -0.059941239655
---------COM joint RHipRoll x = -0.0141365621239 y = -0.0472960546613 z = -0.0927066877484
---------COM joint RHipPitch x = 0.0370712988079 y = -0.054165519774 z = -0.123756557703
---------COM joint RKneePitch x = 0.0207378230989 y = -0.0466940514743 z = -0.172789081931
---------COM joint RAnklePitch x = -0.0300139244646 y = -0.0367956198752 z = -0.172079339623
---------COM joint RAnkleRoll x = -0.0130533091724 y = -0.0406831279397 z = -0.215246081352
Es cadena  RArm
COM cadena RArm x = 0.0314230360091 y = -0.11263871938 z = 0.0194252934307
------ lista joints  ['RShoulderPitch', 'RShoulderRoll', 'RElbowYaw', 'RElbowRoll', 'RWristYaw', 'RHand']
---------COM joint RShoulderPitch x = -0.000184491655091 y = -0.071369998157 z = 0.101645618677
---------COM joint RShoulderRoll x = 0.00751454290003 y = -0.110380701721 z = 0.0791304260492
---------COM joint RElbowYaw x = 0.0135670322925 y = -0.134358853102 z = 0.0312109980732
---------COM joint RElbowRoll x = 0.0329979732633 y = -0.137520968914 z = -0.0154699683189
---------COM joint RWristYaw x = 0.0732797160745 y = -0.117162376642 z = -0.0621113181114
```

Figura 81 Centro de Masa COM de las cadenas y articulaciones del robot.

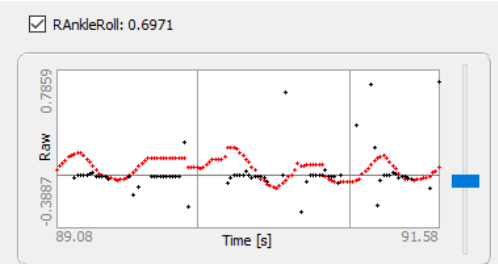
## 5.5. Monitoreo de articulaciones

**Escenario 24**: obtención de datos de articulaciones y sensores para procesamiento.

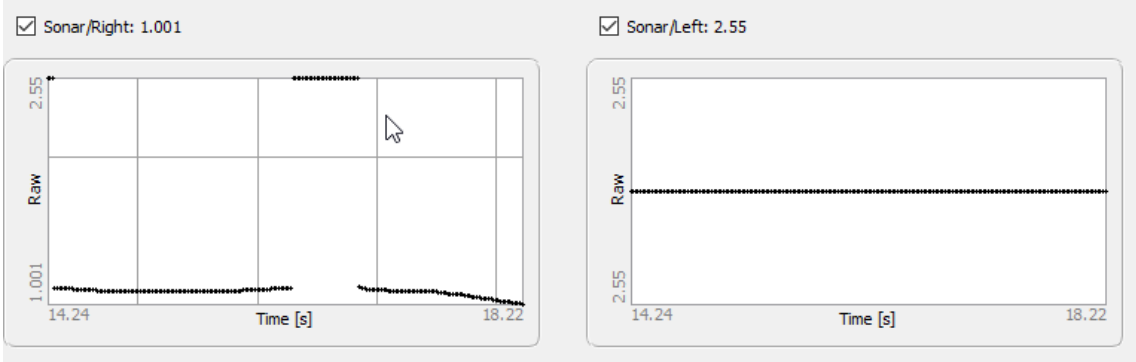
**Resultados**: Se monitorean las articulaciones al movimiento, para observar el comportamiento al mover el robot una serie de pasos.

| Cadera | Rodilla |
|---|---|
| RHipYawPitch | Articulación RKneePitch |
|  |  |

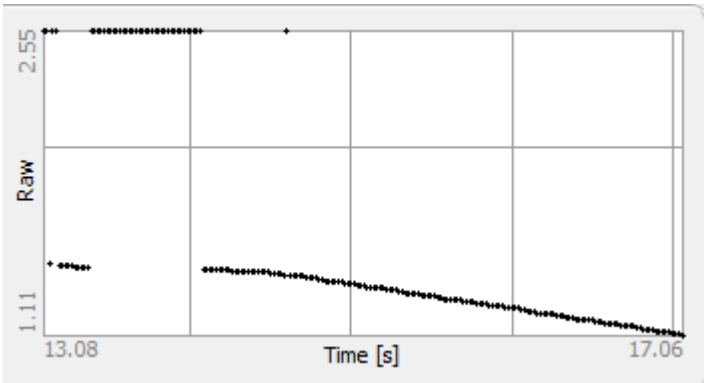
RAnkleRoll. Tobillo




Detección del sonar




Distancia del sensor a 11 cm



### 5.6. Feria de ciencias

**Escenario 25**: Participación en eventos académicos.

**Resultados**: En Agosto 2016 y Febrero 2017 el robot partició en la feria de Ciencias. El evento consiste en mostrar prototipos y productos a estudiantes de instituciones secundarias y Universitarias. El robot participó con los movimientos de saludar, hablar, caminar, laberinto. Aún no estaban en aplicación la vision artificial en el robot.


Figura 82. Feria de Ciencias

## 6. Conclusiones y líneas futuras

**Conclusiones.**

El diseño de la propuesta y los resultados obtenidos verifican que los objetivos planteados se han cumplido:

El robot mediante la locomoción se ubica en una posición determinada en el plano.

Se han utilizado los sensores de visión artificial (cámara) y ultrasónico para la solución planteada, determinando que usando únicamente dos sensores es insuficiente, por lo tanto se requiere adicionar más sensores.

En base a un mapa conocido se realiza la navegación siguiendo la trayectoria establecida.

Se ha utilizado la cámara de visión incluida en el robot para detectar objetos mediante segmentación y descripción con OpenCV.

Las balizas naturales tienen puntos de interes que se encuentran almacenados en la aplicación para comparación al momento de percibir la imagen, y así detectar el objeto. Se utilizan tres conos con marcas identificables, y colores en el piso.

Se utiliza visión por computador aplicada al robot para detectar y describir objetos con SURF y SIFT por puntos de interes. A la vez también se determinan las áreas, contornos, centroides, bordes de la imagen. Al tener pocos objetos en el mapa se describen de forma única.

Las características recuperadas de SIFT son buenos landmark visuales naturales que son ideales para realizar un seguimiento.

En la localización por visión cuando el robot está parado en su momento inicial es bastante buena, debido a que conoce la descripción del objeto y su distancia absoluta. Una vez localizado, obtiene la posición absoluta almacenada y se mueve al punto en el plano 2D. Una vez finaliza el movimiento emite una señal audible.

En la localización por visión al moverse no siempre es exacta y su eficacia se reduce. Esto debibo a que los datos que se obtienen por la cámara tienen mucho ruido por el movimiento del robot al caminar provocanco un incremento de la incertidumbre en la pose del robot. La navegación puede no ser correcta debido a que estima una pose del robot irreal. Por lo cual se hacen movimientos pequeños de 10 cm, se para y en caso de girar se lo hace parcialmente en rotación de 15°.

Para localización se utiliza el filtro de partículas como una implementación de SLAM, sus datos de entrada son las coordenadas espaciales del objeto detectado. Para motivos prácticos en en el mapa se añaden pocos marcadores.

La propuesta de solución es válida en condiciones ideales en entornos de interiores, tales como superficie lisa, balizas naturales fijas en el mapa y conocidas su posición absoluta. Esto por la limitante de determinar en tiempo real y con precisión la distancia entre los objetos que están en el mapa. Por lo cual se debe incluir más sensores tales como profundidad, láser e inclusive sensores externos.

Se realiza la detección de proximidad con éxito utilizando el sensor ultrasónico en lazo abierto al caminar libre en un pasillo.

El movimiento del robot a desplazamientos largos se realiza con bastante imprecisión, por las holguras en las articulaciones y/o el desplazamiento del pie, por lo cual se utilizan técnicas de control tomando como referencia de puntos en el piso (líneas), líneas de la pared, distancia de la pared y de este modo se resuelve el problema.

Con el mapping por sonar, se tiene una estimación del entorno. Se hicieron pruebas con objetos rectangulares y el resultado fue positivo.

En la aplicación desarrollada se utilizan tecnologías que permiten efectuar el control del robot e integrar la visión artificial.

El robot es teleoperado y navega de forma segura detectando landmark visuales y por sonar a una zona de seguridad de 0.5 m.

En la aplicación desarrollada se pueden observar los procesamientos a la imagen efectuada, y así poder corregir parámetros de detección.

Se almacena todo el procesamiento: imagen original, la imagen procesada, la imagen antes de filtro. Todo movimiento del robot almacena la posición en el plano x, y, permitiendo su análisis posterior.

La odometria propia del robot es inexacta. Además, no se pueden utilizar el sensor inercial para corregir debido a que los angulos requeridos de movimiento en el eje Z, no están implementados.

### 6.1. Posibles líneas futuras de investigación

Se tiene como motivación la creación del grupo de investigación de visión artificial, localización por visión artificial, mapeo, navegación de un robot. La línea base de

investigaciones en la Universidad Politécnica Salesiana referente a estos campos de estudio es casi nula, y en el país de manera muy similar. Por lo cual toda propuesta presentada es una línea de investigación que requiere urgente desarrollo. Entre ellas:

- Integrar al robot elementos de SLAM como Visual SLAM, Graph Slam y otros.
- Aplicar Struct From Motion SFM para reconstruir el entorno o mapping y utilizarlo en la fase de navegación.
- Mejorar la localización utilizando cámaras profundas, cámaras estereoscópicas o sensores externos.
- Incrementar el desarrallo en otros lenguajes de programación tal como lenguaje. C++ y Matlab, así como ampliar al uso de ROS.
- Utilizar el Visual Odometría y Visión estereoscopia.
- Mejorar el control del robot bípedo en la locomoción, incluyendo sensores externos y así poder obtener precisión en la localización.
- Mejorar el emparejamiento o matching para detección de objetos más complejos que ya se ha utilizado en objetos simples.
- Mejorar la interfaz de usuario, porque está a un nivel de usuario técnico.
- Ampliar investigaciones referentes a Interacción Humana Robots HRI, donse se estudie el comportamiento y relación entre el robot y niños, o el robot y adultos mayores.

### 6.2. Problemas detectados

Los problemas detectados al efectar el trabajo son:

Errores al no efectuarse la conexión entre el robot y el computador.

Haber logrado el primer movimiento del robot. La locomoción inicialmente se efectuaba vía choregraphe, hasta que se logró hacer mover el robot desde un software externo. Fue muy difícil encontrar la forma de realizarlo de forma externa.

Al efectuarse órdenes de movimiento a largos tramos, el robot no siempre las efectúa. Entre ellos la distancia. Un valor de 0.10 m lo realiza de forma correcta.

La odometría es totalmente inexacta al utilizar las matrices de transformaciones, debido a la acumulación de errores de las medidas previas.

Fue necesario mejorar el sistema de giros para que el robot gire con mayor precisión al ángulo establecido.

La no compatibilidad con versiones nuevas de Python, webots y el Naoqi.

Otro error se produce cuando hay sobrecarga del controlador naoqi, especialmente al trabajar con imágenes.

## Listado de Referencias y bibliografía


[1] F. R. Cortes, *Robótica. Control de robots manipuladores*. México, 2011.

[2] S. Russell and P. Norvig, *Artifical Intelligence: A Modern Approach (Third Edition)*. 2009.

[3] A. Barrientos, L. F. Peñín, C. Balaguer, and R. Aracil, *Fundamentos de robótica*, 2nd ed. Madrid: McGrawHill, 2007.

[4] R. Siegwart, I. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*, 2nd ed. Cambridge: MIT Press, 2011.

[5] D. Goualilier, C. Collette, and K. C, "Omni-directional closed loop walk for nao," *IEEE-RAS Int. Conf. Humanoid Robot.*, 2010.

[6] S. Kajita and K. Tani, "Experimental Study of biped dynamic Walking in the linear inverted pendulum Mode," *Mech. Eng. Lab.*

[7] N. Kofinas, "Forward and Inverse Kinematics for the NAO Humanoid Robot," 2012.

[8] M. Spont, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and control*. 2006.

[9] S. Wen, K. Mohammed, A. Rad, Y. Zhang, and Y. Zhao, "Indoor SLAM Using Laser and Camera with Closed-Loop Controller for NAO Humanoid Robot," *Hindawi Publ. Corp.*, 2014.

[10] A. Pandey, K. Krishna, and M. Nath, "Feature Based Occupancy Grid Maps for Sonar Based Safe-Mapping," *Robot. Res. Cent.*, 2012.

[11] J. Guan and M. Max, "Study on Distance Measurement for NAO Humanoid Robot," *Int. Conf. Robot. Biomimetics*, 2012.

[12] R. Szeliski, *Computer Vision. Algorithms and Applications*. London: Springer, 2011.

[13] S. Krig, *Computer Vision Metrics Survey, taxonomy, and analysis*. Apress Media, 2014.

[14] J. E. Solem, *Programming Computer Vision with Python*. O'Really Media, 2012.

[15] J. Howse, P. Joshi, and M. Beyeler, *OpenCV: Computer Vision Projects with Python*. 2016.

[16] K. Dawson-Howe, *A practical introduction to computer vision with OpenCV*. John Wiley & Sons Ltd, 2014.

[17] S. Fotju, M. Bresler, and D. Prusa, "Nao Robot Navigation Based on a Single VGA," Praga, 2012.

[18] S. Zhao, X. Wang, W. Su, D. Gaoyan, and F. Ziyi, "Target Recognition and Localization of International Standard Platform Humanoid Robot," *Coll. Inf. Sci. Eng.*

[19] J. Hayet, F. Lerasle, and M. Devy, "A visual landmark framework for mobile robot navigation," *LAAS-CNRS,* 2006.

[20] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, 2nd ed. MIT Press, 2005.

[21] S. Osswald, A. Hornung, and M. Bennewitz, "Learning Reliable and Efficient Navigation with a Humanoid."

[22] S. Se, D. Lowe, and J. Little, "Vision-based Mobile Robot Localization And Mapping using Scale-Invariant Features," *Univ. Br. Columbia*.

[23] S. Frintrop, "Visual Robot Localization and Mapping based on Attentional Landmarks," *Comp. Sci. III, Univ. Bonn*.

[24] P. Anderson, Y. Yusmanthia, B. Hengst, and A. Sowmya, "Robot Localisation Using Natural Landmarks," *Rob. 2012 Robot Soccer World Cup XVI*, pp. 118–129, 2013.

[25] J. Forcén, "Navegación de un robot autónomo utilizando balizas y luz estructurada," 2015.

[26] S. Tasse, M. Hofmann, and O. Urbann, "SLAM in the Dynamic Context of Robot Soccer Games," *Rob. 2012 Robot Soccer World Cup XVI*, pp. 368–379, 2013.

[27] M. Havlena, D. Prusa, and PajdlaTomas, "Towards Robot Localization and Obstacle Avoidance from Nao Camera," Prague, 2010.

[28] T. Kastner, T. Rofer, and T. Laue, "Automatic Robot Calibration for the NAO," *Rob. 2014 Robot World Cup*, pp. 233–244, 2015.

[29] J. Tardos, J. Neira, P. Newman, and J. Leonard, "Robust Mapping and Localization in Indoor Environments Using Sonar Data."

[30] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *IEEE*, 1989.

[31] Python, "Sitio oficial de Python," 2016. [Online]. Available: https://www.python.org/.

[32] Softbank, "Framework NAOqi." [Online]. Available: http://doc.aldebaran.com/2-1/dev/naoqi/index.html#naoqi-framework-overview. [Accessed: 01-Feb-2016].

[33] "Sitio oficial Miniconda," 2016. [Online]. Available: https://conda.io/miniconda.html.

[34] Scipy, "Sitio Web Scipy Numpy." [Online]. Available: https://scipy.org/.

[35] Matplotlib, "Sitio Oficial Matplotlib." [Online]. Available: http://matplotlib.org/.

[36] SoftBank, "Aldebaran Robotics." [Online]. Available: https://www.ald.softbankrobotics.com/en.

[37] P. Corke, *Robotics, Vision and Control*, vol. 73. 2011.

[38] R. Hartley and A. Zizzerman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge: Cambridge, University Press, 2003.

[39] Aldebaran, "NAOqi Documentación," 2016. [Online]. Available: http://doc.aldebaran.com/2-1/index.html. [Accessed: 01-Feb-2016].

[40] S. Kajita and K. Tani, "Experimental study of biped dynamic walking in the linear inverted pendulum mode," in *IEEE Int. Conf. on Robotics and Automation*, 1995, pp. 2885–2891.

[41] P.-B. Wieber, "Trajectory Free Linear Model Predictive Control for Stable Walking in the Presence of Strong Perturbations Trajectory Free Linear Model Predictive Control for Stable Walking in the Presence of Strong Perturbations," *IEEE-RAS Int. Conf. Humanoid Robot.*, 2006.

[42] G. Pajares and J. De La Cruz, *Visión por computador imágenes, digitales y aplicaciones*, 2nd ed. Mexico: Alfaomega Grupo Editor, SA de CV, 2008.

[43] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *Comput. Sci. Dep.*, 2004.

[44] H. LBay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded Up Robust Features," *ETH Zurich*, 2008.

[45] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection.," *ECCV*, 2006.

[46] O. Slam, "Open SLAM." [Online]. Available: http://openslam.org/slam.html. [Accessed: 10-Feb-2017].

[47] Udacity, "Udacity Artificial Intelligence for Robotics." .

[48] Alexey Abramov, "On the way to vision computer vision." [Online]. Available: https://salzis.wordpress.com/.

[49] J. Forcen and D. Chaos, "Proyecto Final de Máster Máster en Ingeniería de Sistemas y de Control. Navegación de un robot autónomo utilizando balizas y luz estructurada," 2015.

## Listado de siglas, abreviaturas y acrónimos

| Sigla, abreviatura y acrónimo | Descripción |
|---|---|
| CML | Concurrent Mapping and Localization. (Nombre alternativo al SLAM de concurrente mapeo y localización) |
| COM | Center Of Mass. (Centro de masa) |
| EKF | Filtro extendido de Kalman |
| FAST | Features from Accelerated Segment Test |
| FOV | Field OF View |
| fps | Frame per seconds. Frames por segundo |
| FSR | Force Sensitive Resistors. Sensores resistivos de fuerza |
| ORB | Oriented FAST and Rotated BRIEF |
| POI | Point of interest |
| RAM | Random Access File / Memoria de Acceso Aleatorio |
| RGB | Red, Green, Blue |
| RIA | obot Institute of America |
| ROI | Regions Of Interest. Regiones de interés |
| ROS | Robot Operating System |
| SFM | Struct From Motion. Estructura desde movimiento |
| SIFT | Scale-Invariant Feature Transform |
| SLAM | Simultaneous Localization And Mapping. (Simultanea Localización y Mapeo) |
| SURF | Speeded-Up Robust Features |
| TOF | Time Of Flight |
| UCM | Universidad Complutense de Madrid |
| UNED | Universidad Nacional de Educación a Distancia |
| UPS | Universidad Politécnica Salesiana de Ecuador |
| ZMP | Zero Moment Point. ( Punto de momento cero) |

# Anexos

## A.1. Experimentos - Lista de videos

La web del autor: Orlando Barcia www.orlandobarcia.com muestra el conjunto de videos realizados para creación del Proyecto. Cada entrada o página está enlazada internamente al sitio youtube.com/orlandobarciaweb.

| Descripción | Referencia |
|---|---|
| A.1.1 Presentación de NAO a UCM/UNED | www.orlandobarcia.com/presentacion-nao-ucm-uned-hablado |
| A.1.2 proceso detallado de la simulación del entorno real | http://www.orlandobarcia.com/simulacion-de-un-entorno-real-para-nao-en-webots/ |
| A.1.3 síntesis del Robot NAO y el entorno de modelado | http://www.orlandobarcia.com/robot-bipedo-nao-y-webots-como-entorno-de-modelado/ |
| A.1.4 Locomoción detección de sonar | www.orlandobarcia.com/video-sonar-deteccion |
| A.1.5 Ejecución de software simulador webots y choregraphe | www.orlandobarcia.com/webosts-choreographe |
| A.1.6 Seguidor de objeto a persona con bola roja | www.orlandobarcia.com/Video-seguidor-bola-roja |
| A.1.7 Seguidor de cara a una persona | www.orlandobarcia.com/seguidor-cara |
| A.1.8 . Movimiento en línea recta | www.orlandobarcia.com/mov-linea-recta-x |
| A.1.9 Movimiento a lazo abierto | www.orlandobarcia.com/locomocion-lazo-abierto |
| A.1.10 Caminada a lazo abierto real y virtual | www.orlandobarcia.com/lazo-abierto-caminada |
| A.1.11 Se muestra en el E1 escenario del Laboratorio de Fabricación Flexible | www.orlandobarcia.com/navegacion-interna |
| A.1.12 Navegación en el laboratorio de automatización | www.orlandobarcia.com/Navegación-lab-auto |
| A.1.13 Navegación en el laboratorio de fabricación flexible | www.orlandobarcia.com/navegacion-lab-flexible |
| A.1.14 Navegación 1 en el pasillo | www.orlandobarcia.com/navegacion-pasillo-1 |
| A.1.15 Navegación 2 en el pasillo | www.orlandobarcia.com/navegación-pasillo-2 |
| A.1.16 Detección de colores y formas en tiempo real | www.orlandobarcia.com/deteccion-colores-real |
| A.1.17 Localización por descriptor | www.orlandobarcia.com/localizar-descriptor |
| A.1.18 Locomoción y giro a la derecha | www.orlandobarcia.com/girar-derecha |

## A.2. Cronograma

El proyecto se presentó en septiembre del 2015. El desarrollo se ha iniciado desde diciembre 2016 a septiembre 2017.

| Actividades/Fecha | Diciembre y anterior | Enero | Febrero | Marzo | Abril | Mayo | Junio | Julio | Agosto |
|---|---|---|---|---|---|---|---|---|---|
| Estado del arte | x | x | x | x | x | x | x | | |
| python/robot | x | x | x | x | x | x | x | x | |
| manejo básico del robot | x | x | x | | | | | | |
| locomoción del robot | x | x | x | x | x | | | | |
| navegación del robot telemática | | | x | x | x | | x | x | |
| open cv / segmentación | x | | | x | x | x | x | | |
| open cv / descripción | | | | | | x | x | | |
| integración vision/robot | | | | | | | x | x | x |
| robot segmentación | | | | | | | | x | x |
| robot descripción | | | | | | | | x | x |
| creación mapa del robot | | | | | | | | x | x |
| navegación del robot mapa | | | | | | | | x | x |
| localización filtro de partículas | | | | | | | | x | x |
| aplicacion creación de clases | | | | | | | | x | x |
| aplicación creación GUI | | | | | | | | x | x |
| redacción de la memoria | | | | | | | | | x |

## A.3. Presupuesto

El presupuesto del proyecto, que es una inversión para futuras investigaciones. Por tal motivo se requiere recuperar la inversión efectuando varios trabajos.

| Descripción | Cant Unitaria | Precio Unitario | Precio Total (dólares) |
|---|---|---|---|
| Robot NAO | 2 | $ 15,000.00 | $ 30,000.00 |
| Licencia webots NAO Educativa | 1 | $ 450.00 | $ 450.00 |
| Portatil | 1 | $ 1,200.00 | $ 1,200.00 |
| Uso de Laboratorio | 120 | $ 10.00 | $ 1,200.00 |
| Desarrollador del proyecto | 500 | $ 23.00 | $ 11,500.00 |
| Director del proyecto | 100 | $ 60.00 | $ 6,000.00 |
| | | **subtotal** | $ 50,350.00 |
| imprevistos 10% | | | |

Neto: El robot y el laboratorio lo asume la Universidad

| | |
|---|---|
| Valor Universidad | $ 28,800.00 |
| Monto Neto | $ 21,550.00 |

El valor que asume la Universidad es de $USD 28800 y el presupesto del autor del trabajo es de $21550.

## A.4. Representación del modelo de Color (Color Space)

Un modelo muy utilizado es a nivel de gris Gray. Solo tiene un canal con valores de intensidad de 0 a 255. Cuando la imagen es a color, para cada localización espacial (pixel) hay valores de intensidad por cada canal. El modelo de color RGB. (Red, Green, Blue) utiliza tres canales: canal Red/Rojo; canal Green/Verde; canal Blue/Azul. Los colores se forman en combinación de los tres. Estos colores varían bastante con la iluminación. RGB es aditivo, entre más se agregue intensidad a cada canal, se acercará más al blanco absoluto.

En OpenCv se cambia el orden de los bits, y se denomina BGR. Esto hay que considerarlo debido al manejo y operaciones matriciales que se realizan en ellos.

Otro espacio de color utilizado es el HSV de Hue (matiz), S (saturación) y V (valor). Es una conversión no lineal del espacio de colores RGB. Tanto H y S no varían mucho como consecuencia de los cambios de luz.

Esto en OpenCV hay que realizar una conversión debido a que HSV se maneja con valores de 0 a 255.

El matiz (Hue) indica que color está presente en un determinado canal. La representación es mediante un ángulo y su rango de valores es [0, 360], siendo 0 el rojo, 60 el amarillo y 120 el verde. La "saturación" expresa la cantidad de color en un canal. Su representación es la distancia al eje del brillo negro/blanco. Sus rangos son de [0, 100]. A mayor saturación, más vida tiene un color. Entre menos saturación es más gris. El "Valor" representa la altura en el eje blanco/negro y su rango es de 0, 100], donde 0 es siempre negro y 100 puede variar con la saturación.

El modelo de color YUV utilizado por el robot NAO codifica la imagen teniendo en cuenta la percepción humana. Está formado por tres componentes: luminancia y dos componentes de crominancia.

En el robot se obtiene la imagen y se procede a convertirla para procesarla en modo BGR.

## A.5. Fundamentos de segmentación y descripción de las imágenes

Como lo manifiesta [44] La segmentación es extraer información de la imagen para detectar elementos de la imagen fundamentados en la similitud y discontinuidad. Por técnicas similitud se detecta regiones que son área de la imagen en la que los pixeles que la forman tienen propiedades similares (en color, en intensidad entre otros). Utilizando técnicas de discontinuidad se detectan bordes que son líneas que separan dos regiones de diferentes propiedades, son pixeles en los cuales la imagen presenta una brusca variación en los niveles de gris. Para ambos casos se requiere procesar la imagen original aplicando operadores o funciones de transformación. En

caso de requerirse en la segmentación s como procedimiento previo se aplica técnicas de mejora de la imagen original como suavizado y realzado. El suavizado tiene como fin la supresión de ruido por la acción de la captura de la imagen. El realzado su fin es eliminar falsos reflejos y sombras. Para la detección de bordes (por discontinuidad) se utilizan operadores de primera derivada, segunda derivada y morfológicos). Para la detección de regiones (por similitud) se aplica binarización basada en umbral, crecimiento de regiones, división de regiones, similitud de textura, color o nivel de gris.

Los puntos característicos, son puntos o pixeles que destacan en la imagen por su diferencia de nivel de intensidad con referencia a los píxeles vecinos. Estos por lo general se usan como entrada para entrenar o recuperar información de un clasificador.

La descripción tiene como fin extraer propiedades, características o atributos reconociendo estructuras de la imagen para identificar de forma única a la misma y utilizarlos en fases posteriores. Previamente se Detecta los bordes y/o regiones como elementos de interés. Descriptores de bordes, se identifica los bordes mediante ajuste de rectas, funciones polinómicas, curvas, códigos encadenados entre otras. Los Descriptores de regiones para obtener propiedades de color, textura, superficie, entre otros.

Referente a la segmentación de bordes, en la extracción de bordes se utilizan los operadores de primera derivada y segunda derivada. Los operadores de primera derivada: gradiente, operador sobel, operador prewitt, operador Roberts, máscaras de Kirsch, máscaras de robinson, máscaras de Frei-Chen, Canny'. Los operadores de segunda derivada: operador laplaciana, operador laplaciana de la Gaussiana.

De forma general el operador de gradiente magnifica el ruido en la imagen. Sobel y resto de operadores tienen la propiedad adicional de suavizar la imagen, eliminando parte del ruido. Canny extrae bordes y cierra contornos evitando posibles rupturas de los mismos. Proceso es obtención del gradiente, adelgazamiento del ancho de los bordes hasta lograr bordes un ancho de 1 pixel. Histeresis de umbral. Cierre de contornos abiertos.

Referente a la segmentación de regiones tenemos: binarización basada en umbral threshold, crecimiento de regiones, división de regiones, similitud de textura, color o nivel de gris.

En binarización basada en umbral Respecto al threshold o umbral es una de las técnicas utilizadas para detección de objetos. Por método de otsu o kapur.

En la descripción de líneas y contornos. Por lo general los descriptores deben ser independientes del tamaño, la localización y orientación del objeto, deben tener información para distinguir un objeto de otro. Entre las técnicas se utiliza segmentos rectos mediante códigos de cadena, el ajuste de líneas mediante mínimos cuadrados, ajuste de línea mediante autovector, la transformada de Hough.

La transformada de hough permite encontrar líneas rectas. Otros descriptores como código de cadenas, descriptor de Fourier, momentos, funciones splines.

Referente a la descripción de regiones una de ellas son los momentos invariantes de Hu.

### A.6. Clasificación de las técnicas de búsqueda

Entre las búsquedas no informadas se mencionan las siguientes estrategias [2]:

- Breadth-First search (búsqueda primero en anchura), Uniform-cost search ( búsqueda de costo uniforme),
- Depth-first search( búsqueda primero en profundidad),
- Depth-limited search (búsqueda de profundidad limitada),
- Iterative deepening depth-first search ( búsqueda primero en profundidad con profundidad iterativa ),
- Bidirectional search (Búsqueda bidireccional).

Las estrategias de búsqueda o heurística utilizan en la solución el tipo best-first search (búsqueda primero el mejor). Esta es un caso particular del algoritmo de búsqueda de árboles o búsqueda de grafos.

Entre los algoritmos con la técnica de la búsqueda primero el mejor se mencionan:

- Greedy best-first search (Búsqueda voraz primero el mejor),
- A* ,
- Memory-bounded heuristic search (Búsqueda heurística con memoria acotada),
- iterative-deepening A$*$ (IDA$*$) o A* de profundidad iterativa (A*PI),
- Recursive best-first search (RBFS) o búsqueda recursiva del primero mejor (BRPM) y
- MA$*$ (A*M)

### A.7. Transformaciones morfológicas para disminuir el ruido

Para efectuar una transformación morfológica se usa la erosión, dilatación. Estas se aplican en operaciones binarias que se puede realizar sobre una máscara Mask, que es la imagen a procesar ya en formato binario. Será la parte blanca, con nivel de intensidad 255. Lo que no se desea procesar es en negro 0.

Para efectuar la transformación morfológica. Se requiere un kernel y una máscara.

El Kernel es una matriz de convolución, que debe ser impar de M3x3 por ejemplo la convolución es un proceso de superposición entre el kernel y la imagen.

Erosión o Erode, se aplica para eliminar y reducir las regiones blancas

Dilatación Dilate incrementa la parte blanca de la máscara.

Un opening o apertura, es una erosión seguido de una dilatación. Elimina los elementos de ruido blanco sobre los elementos negros.

Un closing o cierre. Es una dilatación seguida de una erosión Elimina el ruido negro de las partes blancas. Se aplica para quitar los elementos negros dentro de la parte blanca de la máscara.

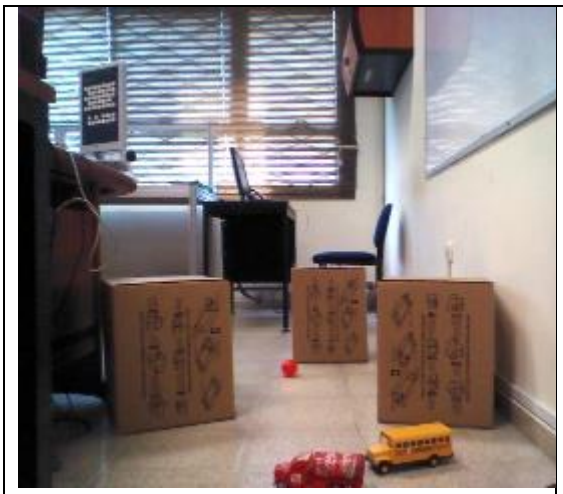
| | | |
|---|---|---|
| **Apertura** | $X \circ B = (X \otimes B) \oplus B$ | erosión seguida de dilatación |
| **Cierre** | $X \bullet B = (X \oplus B) \otimes B$ | dilatación seguida de erosión |
| **Bordes** | $Bordes = X \circ B - X \otimes B$ | apertura menos erosión |

**Dilatación**

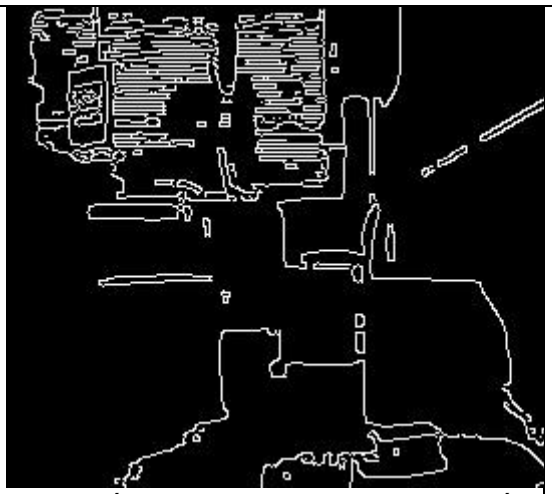
$$X \oplus B = \left\{ \mathbf{d} \in E^2 : \mathbf{d} = \mathbf{x} + \mathbf{b} \ para \ cada \ \mathbf{x} \in X \ y \ \mathbf{b} \in B \right\}$$

**Erosión**

$$X \otimes B = \left\{ \mathbf{d} \in E^2 : \mathbf{d} + \mathbf{b} \ \in X \ para \ cada \ \mathbf{b} \in B \right\}$$



a) Figura original.

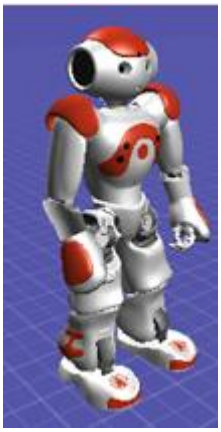


b) Máscara aplicada operación morfológica


c)detección de contornos

Figura 83. Aplicación de operaciones morofológicas.

## A.8. Posturas del robot



Stand

StandInit

Standzero

Crouch

Figura 84. Posturas del robot

## A.9. Encuesta. Recopilación de datos - estudiantes de niveles superiores.

**CONFIDENCIAL**.

Estimado estudiante en proceso de titulación, con el fin de conocer su percepción como integrantes activos de la comunidad universitaria referente a diferentes algoritmos de robótica móvil, visión artificial, localización, mapeo y navegación. Se manifiesta que las respuestas serán administradas con profesionalismo y discreción. Toda información proporcionada es de confidencialidad absoluta. Se garantiza que será utilizada para fines académicos de trabajo de fin de master Navegación por visión y control de un robot bípedo de la Universidad Complutense de Madrid UCM, Universidad Nacional de Educación a Distancia UNED y para la Universidad Politécnica Salesiana como insumo de futuros grupos de investigación y trabajos de titulación. Seleccione con una X las alternativas que considere adecuadas. Gracias por su tiempo y colaboración

---

**1. ¿Cuánto conoce de robótica?**

a) A nivel Experto ( )

b) Bastante ( )
c) Poco ( )
d) Muy poco ( )
e) Desconozco ( )

**2. ¿Cuánto conoce de robótica móvil? (no se refiere a robótica industrial)**

a) A nivel Experto ( )
b) Bastante ( )
c) Poco ( )
d) Muy poco ( )
e) Desconozco ( )

**3. ¿Aplica visión artificial en sus proyectos?**

a) A nivel Experto ( )
b) Bastante ( )
c) Poco ( )
d) Muy poco ( )
e) Desconozco ( )

**4. ¿Aplica matrices de rotación y traslación para el posicionamiento del extremo de un robot?**

a) A nivel Experto ( )
b) Bastante ( )
c) Poco ( )
d) Muy poco ( )
e) No he aplicado ( )

**5. ¿Aplica sensores ultrasónicos para detección de distancia en sus proyectos de robótica?**

a) Si ( )
b) No ( )
c) Desconozco ( )

**6. ¿Qué tipo de Locomoción ha aplicado en sus proyectos de robótica?**

| **Tipo de locomoción** | **SI** | **NO** | **Desconozco** |
|---|---|---|---|
| Locomoción con ruedas | | | |
| Locomoción con patas: bípeda humanoides | | | |
| Locomoción con patas : araña | | | |
| Robots aeros -drone o similares | | | |

**7. ¿Qué tipo de cámaras ha utilizado en sus proyectos de robótica o similares?**

| **Tipo de cámara** | **SI** | **NO** |
|---|---|---|
| Cámara monocular (1 sola cámara) | | |
| Cámaras stereo ( dos cámaras) | | |
| Cámara omnidireccional | | |

**8. ¿Qué tipo de proyectos y/o aplicaciones ha realizado con visión artificial?**

| **Método utilizado** | **SI** | **NO** | **Desconozco** |
|---|---|---|---|
| Detección de objetos por visión | | | |
| Determinación de colores | | | |
| Detección de distancia por visión | | | |
| Odometria visual | | | |

9. **¿Cuál de los algoritmos de visión de artificial ha utilizado para seguimiento de un objeto o Tracking?**

| **Método utilizado** | **SI** | **NO** | **Desconozco** |
|---|---|---|---|
| BOOSTING, | | | |
| KCF, | | | |
| TLD, | | | |
| MEDIANFLOW | | | |
| CMT | | | |

10. **¿Qué mapa de color ha utilizado en el procesamiento de una imagen?**

| **Método utilizado** | **SI** | **NO** | **Desconozco** |
|---|---|---|---|
| Mapa RGB | | | |
| Mapa BGR | | | |
| Mapa de color HSV | | | |

11. **¿Cuáles algoritmos de visión por computador referente a detección y descripción de objetos ha utilizado?**

| **Método utilizado** | SI | NO | Desconozco |
|---|---|---|---|
| Detección de líneas transformada de Hough | | | |
| Detección de esquinas – Harris | | | |
| Descriptor SIFT | | | |
| Descriptor SURF | | | |
| Detector FAST | | | |
| Descriptor BRIEF | | | |
| Desriptor ORB | | | |

12. **¿Cuáles algoritmos de tipo probabilístico ha utilizado para localizar a un robot?**

| **Método utilizado** | SI | NO | Desconozco |
|---|---|---|---|
| Filtro de montecarlo | | | |
| Filtro Kalman | | | |
| Localización de partículas | | | |

13. **¿Utiliza SLAM para Localización de un robot y generar el mapa de forma simultanea?**
    a) Si ( )
    b) No ( )
    c) Desconozco ( )

14. **¿Utiliza algoritmos A * (A start) para navegación de un robot?**
    a) Si ( )
    b) No ( )
    c) Desconozco ( )

15. **¿Qué tipo de control automático ha utilizado para el movimiento y locomoción en un robot?**

| | SI | NO | Desconozco |
|---|---|---|---|
| Control P | | | |
| Control PI | | | |
| Control PID | | | |

16. **¿Ha utilizado Python para desarrollo de algún proyecto de robótica o similares?**
    a) A nivel Experto ( )
    b) Bastante ( )
    c) Poco ( )
    d) Muy poco ( )
    e) No he aplicado ( )

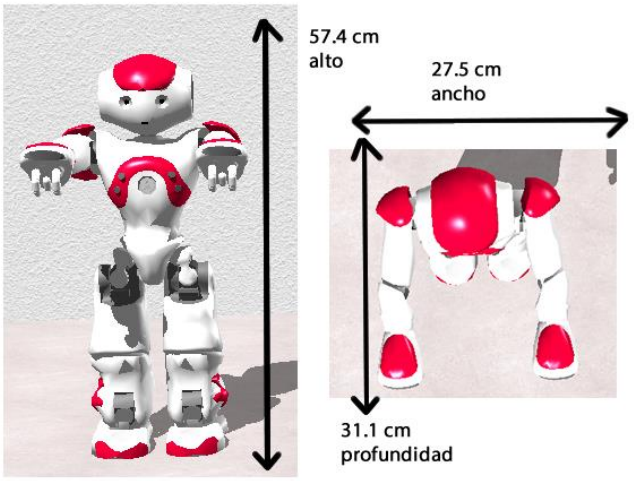
17. **¿Ha utilizado OpenCV para desarrollo de proyectos de visión artificial o similar?**
    a) A nivel Experto ( )
    b) Bastante ( )
    c) Poco ( )
    d) Muy poco ( )
    e) No he aplicado ( )

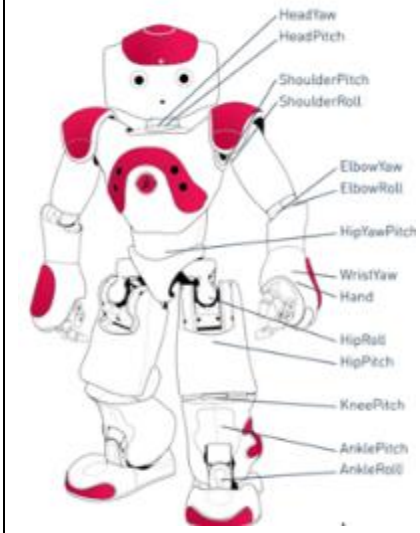
18. **¿Ha utilizado el robot NAO para desarrollo de proyectos de robótica, navegación y visión artificial?. ( No solo mover con choregraphe)**
    a) A nivel Experto ( )
    b) Bastante ( )
    c) Poco ( )
    d) Muy poco ( )
    e) No he aplicado ( )

Muchas gracias por su colaboración

## A.10. Características técnicas del Robot NAO

### Ubicación de los 14 motores

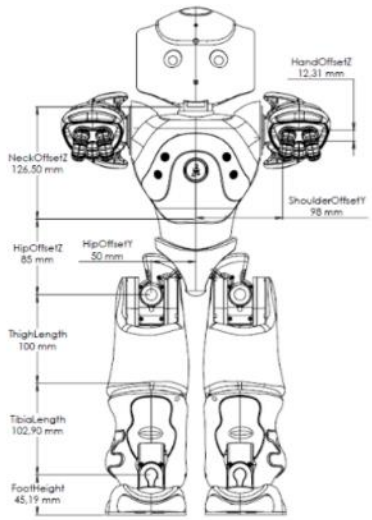


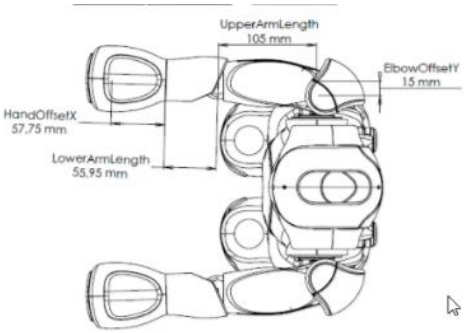




Motors

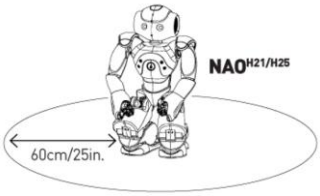
| | Motor type 1 | Motor type 2 | Motor type 3 | Motor type 4 |
|---|---|---|---|---|
| Model | 22NT82213P | 17N88208E | 16GT83210E | DCX16S01GBKL651 |
| No load speed | 8 300 rpm ±10% | 8 400 rpm ±12% | 10 700 rpm ±10% | 12 700 rpm ±10% |
| Stall torque | 68 mNm ±8% | 9.4 mNm ±8% | 14.3 mNm ±8% | 22.4 mNm ±8% |
| Nominal torque | 16.1 mNm | 4.9 mNm | 6.2 mNm | 5.53 mNm |

### Dimensiones de anchura y altura del robot




Espacio de trabajo del robot



Características técnicas del sensor de visión

**Especificiaciones**

| | | |
|---|---|---|
| Camera | Model | MT9M114 |
| | Type | SOC Image Sensor |
| Imaging Array | Resolution | 1.22 Mp |
| | Optical format | 1/6 inch |
| | Active Pixels (HxV) | 1288x968 |
| Output | Camera output | 1280*960@30fps |
| | Data Format | (YUV422 color space) |
| | Shutter type | Electronic Rolling shutter (ERS) |
| View | Field of view | 72.6°DFOV (60.9°HFOV,47.6°VFOV) |
| | Focus range | 30cm ~ infinity |
| | Focus type | Fixed focus |

**Resoluciones soportadas**

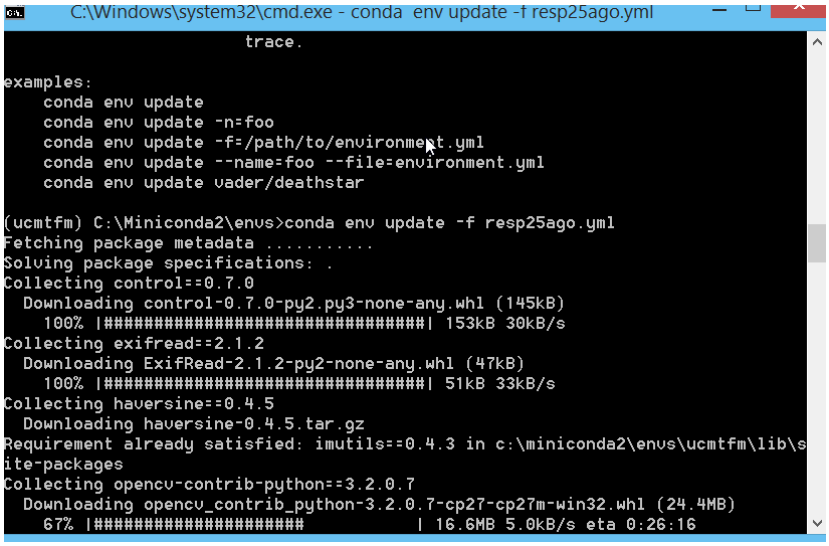
| **Id value** | | **local** | **Gb Ethernet** | **100Mb Ethernet** | **WiFi g** |
|---|---|---|---|---|---|
| 8 | 40x30 (QQQQVGA) | 30fps | 30fps | 30fps | 30fps |
| 7 | 80x60 (QQQVGA) | 30fps | 30fps | 30fps | 30fps |
| 0 | 160x120 (QQVGA) | 30fps | 30fps | 30fps | 30fps |
| 1 | 320x240 (QVGA) | 30fps | 30fps | 30fps | 11fps |
| 2 | 640x480 (VGA) | 30fps | 30fps | 12fps | 2.5fps |
| 3 | 1280x960 (4VGA) | 29 fps | 10 fps | 3 fps | 0.5fps |

## A.11. Valores máximos y mínimos de las articulaciones

HeadYaw:
minAngle -2.08566856384 rad  maxAngle 2.08566856384 rad  maxVelocity 8.26797389984 rad/s maxTorque 1.20000004768 Nm
HeadPitch:
minAngle -0.671951770782 rad  maxAngle 0.514872133732 rad  maxVelocity 7.19407272339 rad/s maxTorque 1.20000004768 Nm
LShoulderPitch:
minAngle -2.08566856384 rad  maxAngle 2.08566856384 rad  maxVelocity 8.26797389984 rad/s maxTorque 1.20000004768 Nm
LShoulderRoll:
minAngle -0.314159274101 rad  maxAngle 1.32645022869 rad  maxVelocity 7.19407272339 rad/s maxTorque 1.20000004768 Nm
LElbowYaw:
minAngle -2.08566856384 rad  maxAngle 2.08566856384 rad  maxVelocity 8.26797389984 rad/s maxTorque 1.20000004768 Nm
LElbowRoll:
minAngle -1.54461634159 rad  maxAngle -0.0349065847695 rad  maxVelocity 7.19407272339 rad/s maxTorque 1.20000004768 Nm
LWristYaw:
minAngle -1.82386910915 rad  maxAngle 1.82386910915 rad  maxVelocity 24.6229305267 rad/s maxTorque 0.759999990463 Nm
LHand:
minAngle 0.0 rad  maxAngle 1.0 rad  maxVelocity 8.32999992371 rad/s maxTorque 0.550000011921 Nm
LHipYawPitch:
minAngle -1.14528501034 rad  maxAngle 0.740717709064 rad  maxVelocity 4.16173744202 rad/s maxTorque 3.20000004768 Nm
LHipRoll:
minAngle -0.379434585571 rad  maxAngle 0.790459632874 rad  maxVelocity 4.16173744202 rad/s maxTorque 3.20000004768 Nm
LHipPitch:
minAngle -1.53588974476 rad  maxAngle 0.483979791403 rad  maxVelocity 6.40239143372 rad/s maxTorque 3.20000004768 Nm
LKneePitch:
minAngle -0.092327915132 rad  maxAngle 2.11254644394 rad  maxVelocity 6.40239143372 rad/s maxTorque 3.20000004768 Nm
LAnklePitch:
minAngle -1.18944191933 rad  maxAngle 0.922581017017 rad  maxVelocity 6.40239143372 rad/s maxTorque 3.20000004768 Nm
LAnkleRoll:
minAngle -0.397760540247 rad  maxAngle 0.768992066383 rad  maxVelocity 4.16173744202 rad/s maxTorque 3.20000004768 Nm
RHipYawPitch:
minAngle -1.14528501034 rad  maxAngle 0.740717709064 rad  maxVelocity 4.16173744202 rad/s maxTorque 3.20000004768 Nm
RHipRoll:
minAngle -0.790459632874 rad  maxAngle 0.379434585571 rad  maxVelocity 4.16173744202 rad/s maxTorque 3.20000004768 Nm
RHipPitch:
minAngle -1.53588974476 rad  maxAngle 0.483979791403 rad  maxVelocity 6.40239143372 rad/s maxTorque 3.20000004768 Nm
RKneePitch:
minAngle -0.092327915132 rad  maxAngle 2.11254644394 rad  maxVelocity 6.40239143372 rad/s maxTorque 3.20000004768 Nm
RAnklePitch:
minAngle -1.18630027771 rad  maxAngle 0.932005822659 rad  maxVelocity 6.40239143372 rad/s maxTorque 3.20000004768 Nm
RAnkleRoll:
minAngle -0.768992066383 rad  maxAngle 0.397935062647 rad  maxVelocity 4.16173744202 rad/s maxTorque 3.20000004768 Nm
RShoulderPitch:
minAngle -2.08566856384 rad  maxAngle 2.08566856384 rad  maxVelocity 8.26797389984 rad/s maxTorque 1.20000004768 Nm
RShoulderRoll:
minAngle -1.32645022869 rad  maxAngle 0.314159274101 rad  maxVelocity 7.19407272339 rad/s maxTorque 1.20000004768 Nm
RElbowYaw:

minAngle -2.08566856384 rad  maxAngle 2.08566856384 rad  maxVelocity 8.26797389984 rad/s maxTorque 1.20000004768 Nm
RElbowRoll:
minAngle 0.0349065847695 rad  maxAngle 1.54461634159 rad  maxVelocity 7.19407272339 rad/s maxTorque 1.20000004768 Nm
RWristYaw:
minAngle -1.82386910915 rad  maxAngle 1.82386910915 rad  maxVelocity 24.6229305267 rad/s maxTorque 0.759999990463 Nm
RHand:
minAngle 0.0 rad  maxAngle 1.0 rad  maxVelocity 8.32999992371 rad/s maxTorque 0.550000011921 Nm

## A.12. Actualización de entornos con miniconda



**Fotos adicionales relativas a experimentos**

Some Webots users

Filter by country: Ecuador

| Name | Organization | Country |
| --- | --- | --- |
| Prof Orlando Barcia Ayala | Universidad Politécnica Salesiana | |