UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA
División de Estudios de Posgrado

"Implementación de un sistema modular para el reconocimiento de objetos en un contexto de robótica de servicio"

TESIS PARA OBTENER EL GRADO DE:

MAESTRO EN ROBÓTICA

PRESENTA:

ING. KAREN LIZBETH FLORES RODRÍGUEZ

DIRECTORES DE TESIS:

Dr. Felipe de Jesús Trujillo Romero
Universidad Tecnológica de la Mixteca
Dra. Claudia Esteves Jaramillo
Universidad de Guanajuato

HUAJUAPAN DE LEÓN, OAXACA, MÉXICO, MARZO DEL 2017

Dedicado a mi familia.

Agradecimientos

Le agradezco plenamente a mis padres, hermanos, a mi pareja y su familia por el apoyo que me brindaron en todo momento para llevar a cabo este proyecto de vida.

Le agradezco a CONACyT por la beca económica 571271/584190 tanto nacional como extranjera recibida durante la *Maestría en Robótica*.

De igual manera quiero agradecer a mi director de tesis el Dr. Felipe de Jesús Trujillo Romero, por todo el apoyo, los consejos y la confianza que recibí de su parte durante todo el desarrollo de mi trabajo de tesis.

Le agradezco a la Dra. Claudia Esteves Jaramillo de la Universidad de Guanajuato por haber aceptado ser parte de este trabajo de tesis y al Dr. Jean Bernard Hayet de CIMAT por el apoyo brindado.

Agradezco al Dr. Ricardo Pérez Aguila y al Dr. Rosebet Miranda Luna por el tiempo que han invertido en la revisión de esta tesis como sinodales.

Gracias al Dr. Wael Suleiman de la Universidad de Sherbrooke por recibirme en estancia bajo su tutela.

Gracias al Dr. José Victor Nuñes Nalda de la UPSIN por facilitar el laboratorio de esta universidad para realizar ciertas pruebas de este tesis.

Publicaciones Derivadas

[1] Flores-Rodríguez, K. L.and Trujillo-Romero, F., "Desarrollo de un sistema lógico difuso para el control de la locomoción bípeda de un robot humanoide NAO", Research in Computing Science 113, (2016), pp. 181–194

[2] Flores-Rodríguez, K. L. and Trujillo-Romero, F., "Free form object recognition module using A-KAZE and GCS", Research in Computing Science 118, ISSN 1870-4069, (2016), pp. 19–29

[3] Flores-Rodríguez, K. L., Trujillo-Romero, F. and Suleiman, W. "Object recognition modular system implementation in a service robotics context", CONIELECOMP 2017, 27th International Conference on Electronics, Communications and Computers, accepted : dec–16–2016

Índice

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Crear un ser artificial ha sido el sueño del hombre desde que nació la ciencia. Ya que siempre ha tratado de imitar, igualar y mecanizar la inteligencia humana en máquinas que puedan ejecutar tareas para cumplir un propósito. Esta idea se plasma en el desarrollo de autómatas humanoides el cual intenta crear a un ser idéntico al humano con capacidad de pensar y servir. Estas máquinas cuentan con la forma antropomórfica y articulaciones necesarias para realizar movimientos parecidos a los humanos. Se considera que sus raíces se hallan en la historia misma de la humanidad, en cuentos, mitos y leyendas de diversas culturas. Por mencionar alguno, el mito griego popular *"Talos: Gigante de Bronce"* [1], trata sobre un mítico gigante de bronce llamado Talos que custodiaba la isla de Creta. Según se cuenta, el gigante había sido fabricado por el dios Hefesto para donárselo al rey cretense, con la misión de proteger la isla contra ataques enemigos.

El desarrollo de autómatas data de algún tiempo atrás según la línea del tiempo publicada en [2]. Se sabe que en el año 1495, Leonardo Da Vinci contaba con bocetos sobre autómatas humanoides. En los siglos XVI y XVII se presentan los autómatas *Ningyo Karakuri*, un sistema mecanizado de títeres. En el año 1920 Karel Capek presenta su obra *"Rossum's Universal Robots"*. En esta obra fue utilizada por primera vez la palabra *robot* para referirse a autómatas humanoides que realizaban trabajos que les correspondían a los humanos. Esta palabra se refiere al término *trabajo duro* en checo el cual fuera el idioma natal del autor. Ocho años después se presentó el primer robot con capacidad de interactuar proveniente de Japón, *Gakutensoku*, diseñado y construido por el biólogo Makoto Nishimura. En 1930, Westinghouse Electric Corporation, creó un robot humanoide conocido como *Elektro*, capaz de interactuar con fines de exposición. Sin embargo, el único propósito de estos primeros autómatas era el de entretener.

En el año 1941, Isaac Asimov acuña el término robótica como la ciencia o tecnología de diseñar, construir, operar y usar robots o dispositivos automáticos similares. Curiosamente fue publicado entre sus historia de ciencia ficción de la revista titulada *"Astounding. Stories of Super Science"*, [3]. Tiempo después, la comunidad científica adoptó el término sin gran problema. Fue hasta 1954 cuando se presentó el *UNIMATE*, el cual fue el primer robot autónomo, digital y programable capaz de hacer una tarea funcional, inventado por George Devol. Este robot era un autómata que levantaba y apilaba piezas metálicas en una fundición, [4]. No obstante, tuvieron que pasar 20 años para que se desarrollara el que se considera el primer robot humanoide. En

1973, en Japón, el primer robot antropomórfico *WABOT-1*, [5], fue demostrado por Kato en la universidad de Waseda. Este robot podía reconocer objetos mediante visión, entender lenguaje, hablar con voz artificial, manipular objetos con dos manos y caminar en dos piernas. Usando un simple esquema de control, fue capaz de realizar unos cuantos pasos de estable. Este logro fue el punto inicial de una generación prolífera de bípedos en Japón.

En la actualidad un robot no está pensado para proteger e intenta ser más que una máquina de entretenimiento. En [6], se menciona que las aplicaciones de los robots tienen el fin de cubrir cuatro características particulares, deben ser capaces de cumplir con tareas peligrosas, sucias, tediosas y/o pesadas. Además, la robótica es considerada una sinergia entre diversas disciplinas, tales como: mecánica, electrónica, informática, inteligencia artificial, ingeniería de control, física, etc. Según [7], un robot es un sistema complejo, compuesto de: (1) sensores que le permiten recibir información acerca de su estado propio y del estado de su entorno, (2) mecanismos efectores para actuar sobre su entorno, (3) sistema de planificación de la actividad y de control para asegurar el logro de sus objetivos en un entorno cambiante, (4) medios de comunicación con el usuario y otros sistemas técnicos.

La robótica industrial ha demostrado ser una tecnología madura al cubrir con la mayoría de las características comentadas con anterioridad. Los robots industriales son capaces de levantar cargas muy pesadas, trabajar con gran exactitud y son relativamente de bajo costo. Pero aún existe una desventaja notable, este tipo de robots trabajan en un ambiente estructurado, ambientes controlados en los que las tareas están coordinadas y no existe gran interacción con el ser humano. Para cubrir esa notable desventaja, las investigaciones en robótica están siendo encaminadas hacia la robótica de servicio donde robots y humanos cooperan en un ambiente no estructurado. Según la Federación Internacional de Robótica (IFR), un robot de servicio es un robot que opera de forma parcial o totalmente autónoma, para realizar servicios útiles para el bienestar de los humanos y del equipamiento, excluyendo operaciones de manufactura. Entendiendo como servicio a una actividad que se realiza para el beneficio de otros.

Aunque aún no exista una *"Robotina"* o un *"Wall-e"* en nuestros hogares o centros de trabajo ayudándonos con tareas domésticas así como lo muestran algunas historias de ciencia ficción, hoy en día se pueden encontrar sistemas robóticos casi en cualquier área. Algunas de las áreas donde la robótica tiene aplicación son: (1) la producción masiva de cualquier producto, (2) en el sector económico (agricultura, minería, etc.), (3) en medicina con robots que realizan cirugías, (4) en exploración espacial, (5) en la industria bélica. Según la IFR, en el 2013 las adquisiciones en robots industriales fueron de 178,132 unidades, en el 2014 205,000 unidades y se estima que del 2015 al 2017 aumentará un 12 %, [8]. Esto indica que, según los números, para el 2017 existirán 1,946,000 robots industriales en el mundo. En cuanto a los robots de servicio, en el 2013 fueron adquiridas 21,000 unidades, se proyecta que para el periodo 2014-2017 cerca de 134,500 nuevos robots de servicio para uso profesional y 31 millones de unidades de robots de servicio para uso personal sean adquiridos. Lo anterior sabiendo que la demanda de robots industriales se considera para un mercado reducido, además, los robots industriales cuentan con un tiempo mayor de vida útil que los robots de servicio que se encuentran en el mercado.

Se han desarrollado robots de servicio para diversas áreas que cubren funciones parciales en las actividades humanas. Entre estos robots de servicio se pueden mencionar algunos juguetes robóticos como el *MiP* [9], que es un pequeño robot con capacidad de transportar una carga chica, por ejemplo una bebida, balanceándola en su base. También se cuenta con mascotas robóticas como *PARO* [10], este es un robot en forma de bebé foca la cual es utilizada como herramienta terapéutica. Igualmente se han desarrollado robots de limpieza como *ROOMBA* [11], el cual es un robot aspiradora que recorre toda la superficie en una habitación evadiendo obstáculos. Además existen desarrollos en robótica de asistencia para hospitales y hogares como el robot *Care-O-bot 4* [12], el cual es un robot móvil de estatura media que tiene la capacidad de monitorear los signos vitales de una persona. De igual forma se cuentan con robots cirujanos como *da Vinci Surgery* [13], el cual es un sistema robotizado que manipula herramientas de cirugía con mucha precisión y exactitud. Otros desarrollos más son los robots para transporte de cargas en terrenos difíciles como *BigDog* [14], este es un robot cuadrúpedo que balancea cargas pesadas en terrenos accidentados. Por otro lado, existen algunos robots humanoides de carácter comercial que han surgido para desarrollo científico en centros de investigación y escuelas. Entre estos se encuentra: el robot *QRIO* [15], *HOP* [16], *Bioloid* [17], *NAO* [18]. Siendo el robot *NAO*, el de mayor capacidad computacional.

Si bien existe gran diversidad de robots de servicio, aun no hay en el mercado un robot humanoide con inteligencia artificial colaborando en tareas domésticas. Existen varias razones del porqué de la afirmación anterior, en [19], se hace una breve discusión sobre robots domésticos, donde se menciona que las tareas que deben cumplir este tipo de robots son muy difíciles. Tareas como por ejemplo: limpiar ventanas, lavar vajilla, doblar y ordenar ropa, ordenar habitaciones, planchar, etc. Para realizar dichas tareas es necesario resolver problemas como la visión 3D bajo variabilidad de iluminación, manipulación de objetos de configuración variada e inteligencia artificial para la toma de decisiones. Este tipo de sistemas deben realizar toda una serie de tareas para cumplir un objetivo. En [20], se enfatiza los desafíos "futuros" de la robótica, los cuales se considera deben cumplir los robots de servicio de uso general:

- ✓ Manipulación e interacción física con el mundo real.
- ✓ Percepción de ambientes no estructurados.
- ✓ Seguridad para la operación cerca de humanos.
- ✓ Interacción hombre-robot.
- ✓ Redes de trabajo entre robots, sensores y usuarios.

La solución de dichos desafíos han sido parcialmente desarrollados en un contexto de investigación donde se busca resolver problemas a situaciones más específicas. Por ejemplo, se conoce gran diversidad de investigaciones en robótica que dan solución parcial a problemas como, aspirar y barrer, envíos y entregas en oficina, guía de turista en hoteles o museos, entretenimiento y exposición, etc. En Japón se ha invertido en robótica humanoide obteniendo resultados impresionantes en diversas áreas como: visión, locomoción, entretenimiento e interacción con humanos. Tal es el caso del robot *ASIMO* de Honda [21], que cuenta con comportamiento autónomo. Este robot es considerado uno de los robots humanoides más avanzados en la actualidad, pretende ser de ayuda para personas que carecen de movilidad completa en sus cuerpos. De igual forma,

el Instituto Nacional de Ciencia Industrial Avanzada y Tecnología (AIST), cuentan con el desarrollo de robots ayudantes domésticos con propósitos generales, *The Humanoid Robotics Project* (HRP) una serie de robots humanoides de mediana estatura [22]. Por otro lado, para motivar a la comunidad científica en lo referente a la investigación en robótica, existen varios eventos y torneos. Entre las competencias robóticas donde los robots de servicios son los protagonistas se encuentra el torneo de *RoboCup* [23]. Este torneo es una iniciativa científica internacional con el propósito de impulsar el desarrollo en inteligencia artificial y robótica. Cuando se inició en 1997, se propuso el objetivo principal de conseguir un equipo de jugadores de fútbol robótico capaces de vencer en una copa del mundo a un equipo de jugadores humanos para el 2050. Mientras esa misión se cumple, *RoboCup* se ha expandido a otros dominios de aplicación relevantes basados en las necesidades de la sociedad moderna. Las 4 categorías principales son: *RoboCupRescue*, *RoboCupJunior*, *RoboCupSoccer* y *RoboCup@Home*. De las anteriores, *RoboCupRescue* se dedica a robots de búsqueda y rescate en caso de desastres mientras considera aplicaciones del mundo real y la interacción hombre-máquina con robots autónomos. Mientras que *RoboCup@Home* [24], trata del desarrollo de robots de servicio y asistencia con alta relevancia para las futuras aplicaciones domésticas. Otra competencia de la cual se considera debe hacerse mención fue *Darpa Robotics Challenge* [25]. Esta competencia fue realizada por la agencia del departamento de defensa de Estados Unidos *Defense Advanced Research Projects Agency* (DARPA). Se llevó a cabo del 2012 al 2015 y tuvo como objetivo el desarrollo de robots semi-autónomos humanoides los cuales debían contar con la capacidad de realizar tareas complejas en ambientes peligrosos.

Una de las razones por las que el desarrollo en robótica de servicio ha despertado tanto interés en la comunidad científica es el envejecimiento poblacional. Según [26], la población mundial está creciendo en proporciones jamás antes conocidas. Aunque en casi todas partes están bajando los índices de fecundidad, el crecimiento de la población mundial se mantendrá hasta muy avanzado el siglo XXI. Según proyecciones actuales, la población mundial alcanzará aproximadamente los 9 000 millones de habitantes para el año 2050. Como consecuencia de los bajos índices de natalidad, unidos a mejoras en la esperanza de vida, muchos países desarrollados han registrado un aumento rápido de la proporción de personas ancianas y, en algunos casos, incluso una baja en su población total. De igual manera en [27] se menciona que para el 2050, más de la cuarta parte de la población en México será vieja.

El aumento de personas ancianas le preocupa a países primer mundistas pues tiene como consecuencia menor mano de obra en la mayoría de las áreas de producción. Entre estas áreas se encuentra por supuesto la dedicada al cuidado de la salud. Es por esto que se invierte en el desarrollo de robots humanoides. El hecho de considerar un robot humanoide como un ideal de robot para servir a la humanidad es por el parecido con el hombre, ya que en algún punto se deja de ver como una máquina. Es más amigable interactuar con una entidad parecida a uno mismo que con un simple artefacto. Además de la clara ventaja de interacción en ambientes humanos donde la infraestructura considera rampas, escaleras y desniveles.

1.1. Estado del Arte

En este trabajo de tesis se tocan temas relacionados al reconocimiento de objetos y la robótica de servicio. Por este motivo, en esta sección se estudian trabajos de visión, interacción,

navegación, manipulación y locomoción referentes a robótica de servicio. De igual manera se revisaron y estudiaron diversos métodos de reconocimiento de objetos con el objetivo de comparar su desempeño y resultados.

1.1.1. Robótica de Servicio

La investigación en robótica de servicio no deja de despertar el interés en la comunidad científica. Estas investigaciones comprenden desde el desarrollo de aplicaciones o algoritmos para cubrir alguna de las tareas que deben cumplir este tipo de robots hasta la construcción de nuevas plataformas robóticas. A partir de la búsqueda bibliográfica sobre robótica de servicio se encontraron algunos trabajos sobre construcción y desarrollo de estos robots.

Una de las habilidades con las que un robot de servicio debe contar es con visión artificial. Esta habilidad le permite interactuar mejor con el ambiente donde se encuentra. Por ejemplo en [28], se muestra la construcción de una plataforma desde el diseño hasta la implementación, considerando el uso de inteligencia artificial para la autonomía del mismo. Esta plataforma puede detectar y reconocer rostros utilizando herramientas como el clasificador en cascada basado en *Wavelets de Haar* [29] junto con el algoritmo *Fisherfaces* [30]. En el trabajo en [31], se diseñó e implementó un sistema para robots humanoides de entretenimiento y competición. Este sistema cuenta con un algoritmo capaz de reconocer colores y con base a estos planear movimientos en tiempo real mediante sistemas expertos.

Navascués [32] desarrolló funciones primitivas mediante el software *Robot Operating System* (ROS) [33] que le permiten a un robot humanoide la localización e interacción con objetos. Morante [34] desarrolló una interfaz y una librería para visión artificial, navegación y seguimiento multiplataforma. En el seguimiento se utiliza la técnica *SURF* [35], *Meanshift* o *Templates Matching* [36]. La detección de movimiento la realiza mediante *Optical Flow* [37]. Lorencik *et al.* [38] se enfocaron en la realización de un sistema en la nube para el reconocimiento de objetos. Utilizan técnicas como *SIFT* [39] y *SURF* para la extracción de características. Además, hacen uso del método basado en redes neuronales *MF ArtMap* [40] para la clasificación de objetos. En los trabajos anteriores se menciona que los resultados obtenidos son aceptables pero todos mostraron una clasificación correcta debajo del 90 %.

En los trabajos [41], [42], [43], [44] y [45], se implementaron sistemas de reconocimiento de objetos para ciertos robots que les permite resolver tareas específicas como manipulación de objetos simples. Los trabajos anteriores hicieron uso de algoritmos que se basan en las forma de los objetos poliédricos mediante el uso de diversas herramientas de procesamiento de imágenes. Estos trabajos detectan los objetos en movimiento con la ayuda de estimadores como el *Filtro de Kalman Lineal* [46] para seguimiento de estos. En [47], Peña presenta el desarrollo de un módulo de visión para un robot humanoide que se asimila a los sistemas de reconocimiento de objetos descritos con anterioridad. A diferencia de los anteriores, este sistema utiliza la red neuronal *Perceptrón* [48] para la clasificación y el reconocimiento de objetos. Otros trabajos que hacen uso de las redes neuronales para el reconocimiento de objetos, pero en robots manipuladores, se describen a continuación. En [49] se utilizó el modelo clásico de red neuronal *multilayer perceptron* (MLP) [50]. Mientras que en [51] se utilizó la técnica *FuzzyARTMAP* [52].

En [53] por su parte, se utilizó la red neuronal MLP con *Backpropagation* [48]. En [54], se basa en la forma de los objetos y utiliza los enfoques de red neuronal *MLP* con *Backpropagation* y el método no supervisado *Mapa Auto-Organizado de Kohonen* [48]. Estos trabajos tienen en común la obtención de resultados arriba del 60 % de reconocimiento correcto.

Otra de las habilidades de los robots de servicio es el trabajo colaborativo. Esta habilidad le permite realizar actividades ya sea con otros robots o con humanos. Por ejemplo, Battiston [55], desarrolló un sistema que le permite a un robot humanoide interactuar con otro robot o con un humano con el fin de intercambiar objetos. La detección de los objetos a intercambiar se realiza utilizando técnicas de procesamiento de imágenes como *filtros HSV* [56], conectividad entre componentes y uniones. En el trabajo de Ramanathan *et al.* [57], se desarrolló un sistema de visión que consiste en la categorización de objetos. Este sistema se basa en la técnica *Bag of Words* [58] junto con *SIFT*. Este trabajo es muy interesante ya que un robot humanoide aprende a distinguir entre cuatro clases de objetos. El robot inspecciona cada uno de los objetos para el aprendizaje y reconocimiento. Los resultados de este trabajo tuvieron un porcentaje de categorización correcta en un rango de 50-80 por ciento. Krause *et al.* [59], se desarrollaron un método de aprendizaje y reconocimiento de objetos para un robot usando lenguaje natural. El enfoque se basa en describirle al robot el objeto a reconocer. El robot se apoya en la descripción dada mediante un sistema de lenguaje natural integrado y un sistema de visión. Este sistema de visión utiliza un sensor *RGB-d* [60] para la captura de la información visual y el método *SIFT* para la clasificación y reconocimiento de objetos.

Con el fin de cubrir la tarea de localización en el ambiente por parte de robots móviles se investigaron trabajos referentes a navegación espacial. Se estudiaron algunos artículos referentes a localización espacial para robots en los cuales se desarrollan algoritmos para *RoboCupSoccer* de plataforma estándar. Trabajos como [61] [62] [63], presentan sistemas de visión para la localización de objetos y de ellos mismos en la cancha de juego. Estos sistemas están basados en segmentación de color y herramientas como las redes neuronales, *Filtros de Kalman* y *Filtro de Partículas* [64]. El robot *NAO* es utilizado por Llofriu *et al.* [65], para realizar un sistema que reproduce la navegación realizada por parte de animales. El sistema permite desarrollar enfoques bio-inspirados de navegación para la resolución del problema de *SLAM visual* [66].

1.1.2. Reconocimiento de Objetos

El reconocimiento de objetos es una tarea importante en este trabajo de tesis, por esta razón se realizó una revisión extensa sobre métodos de reconocimiento de objetos para encontrar el que mejor se adapte al problema a atacar. Artículos como [67] [68] [69] [70] [71], presentan desarrollos de modelos de detección visual en base a descripción de forma o características (*SIFT*, *SURF*, etc), utilizando técnicas de clasificación como *Lógica Difusa* [72] y *Adaboost* [73]. Gómez [74], desarrolló un modelo de reconocimiento de objetos basado en la forma de los mismos. Así mismo, realizó un estudio comparativo entre algunas técnicas de inteligencia artificial. Estas técnicas comprenden las técnicas de *Growing Neural Gas* [75]. Para la extracción de características utilizó métodos similares a los de los artículos descritos con anterioridad. Sin embargo, es importante mencionar que realizó una comparativa entre *Mapa Auto-Organizado de Kohonen*, *Growing Cell Structures* [76], *Neural gas* [77] y *Growing Neural Gas*. Los resultados muestran

que el método *Growing Cell Structures* fue el más rápido, mientras que el método *Growing Neural Gas* está totalmente balanceado y lo consideran óptimo. Ya que el segundo método fue el que mejor preservó la topología, consumió menos tiempo, tuvo buenos resultados en ejecución de tiempo real y ejecución con figuras complejas.

Basado en el tipo de características utilizadas, los métodos de reconocimiento de objetos, en general puede ser divididos en dos categorías, global o local. Considerando sólo los métodos de características de superficie local, estos pueden ser comprimidos en tres fases: detección de *keypoint* (elementos clave), descripción de características de superficie local y *matching* (coincidencia) de superficie. A continuación se mencionan algunas investigaciones relacionadas con reconocimiento de objetos 2D y 3D. Tomando en cuenta que los métodos de reconocimiento de objetos en 2D y 3D se diferencian en los datos a procesar. Para los métodos que consideran objetos 2D los datos provienen de sistemas monoculares. Mientras que los datos para el reconocimiento 3D pueden provenir de sensores de profundidad *RGB-d*, imágenes de rango, nubes de puntos o mallas poligonales.

Alcantarilla *et al.* [78], desarrollaron un algoritmo de detección y descripción de características 2D multiescala llamado *KAZE*. Este método es comparable a *SIFT* pero muestra la desventaja de un incremento moderado en costo computacional comparado a *SURF* o *CenSurE* [79]. Sin embargo, este método se encuentra un paso adelante en la ejecución de la detección y descripción en comparación a los existentes, ya que, incrementa la repetitividad y distinción gracias a la implementación del *filtrado de difusión no lineal* [80] que permite considerar los bordes. Más tarde, Alcantarilla *et al.* en [81], muestran el desarrollo del algoritmo *A-KAZE*, en el que utilizan el esquema numérico *Fast Explicit Diffusion* (FED) [82], embebido en un marco de trabajo piramidal para acelerar dramáticamente la detección de características en espacio de escala no lineal. Además, se introduce el descriptor *Modified-Local Difference Binary* (M-LDB) [83] que es altamente eficiente, explota información general de gradiente desde el espacio de escala no lineal, es invariante a rotación y escala, aparte de que requiere menor capacidad de almacenamiento. En este trabajo se presentó una evolución extensiva que muestra el excelente compromiso entre rapidez y ejecución del enfoque comparado a los métodos *BRISK* [84], *ORB* [85], *SURF*, *SIFT* y *KAZE*.

Salamanca *et al.* [86], desarrollaron un método de reconocimiento de objetos de forma libre a partir de los datos de rango de una vista parcial usando *Cono Curvaturas Ponderadas* determinada a partir de una representación previa llamada *Cono-Curvatura* (CC) [87]. El método emplea un conjunto de características determinadas sobre mallas esféricas y los datos de rango 3D de las vistas parciales de los objetos que se requieren conocer. La tasa de reconocimiento de este método es del 90 %, sin embargo, se menciona que tiene la desventaja de no poder ser implementado en tiempo real pues conlleva un coste de tiempo alto. Chen *et al.* [88], desarrollaron un método de reconocimiento de objetos de forma libre 3D usando *Local Superficial Patches* (LSP) [89]. Este método se compara a los métodos de *Spin Image* [90] y *Spherical Spin Image* [91], con un costo de tiempo de 89.42 segundos, siendo la mitad de tiempo que los métodos mencionados.

Ya que en este trabajo de tesis se utilizará reconocimiento de objetos en un contexto de robótica de servicio es necesario elegir un método para la búsqueda de objetos. Se propone el uso de un método que, basado en lo que el robot observa en el ambiente, sea capaz de decidir hacia donde es mas conveniente avanzar. Este método toma decisiones basadas en el reconocimiento del objeto que está buscando y en la distancia que le hace falta para llegar a él. Por lo anterior, se propone la revisión del enfoque *Next Best View* (NBV) [92] para la planeación de la búsqueda de los objetos a reconocer. Este enfoque tiene diferentes variantes que parten de la idea de la adquisición de datos y exploración automática por parte de sistemas de visión computacional y robots, estimando la siguiente posición para el sensor. Potthast y Sukhatme [93], presentaron una variante del método NBV para ambientes ocluidos. Se presentan evaluaciones experimentales mediante una plataforma robótica. La plataforma robótica realiza una inspección del ambiente donde se encuentra y mediante estimaciones de los nuevos datos va cambiando de posición para cubrir todo el ambiente en el que pueda navegar.

Krainin *et al.* [94], desarrollaron un sistema que le permite a un robot tomar un objeto y moverlo enfrente de su cámara para reconstruir un modelo 3D. Se utiliza una variante del algoritmo NBV basado en la ganancia de información para determinar cómo manipular el objeto de manera probabilística. Kriegel *et al.* [95], desarrollaron un enfoque basado en superficie de NBV para la creación de modelos 3D de objetos nuevos. Se basa en la detección de bordes en la superficie escaneada. Dunn *et al.* [96], presentaron un enfoque para la adquisición de modelos 3D en vehículos autónomos y robots. Este enfoque es incremental, además alterna entre el enfoque NBV para maniobrar y planeación de trayectorias recursiva. Los trabajos de Vasquez-Gomez *et al.* en [97] y [98], plantean el uso de NBV para reconstrucción de objetos tridimensionales enfocado a aplicaciones robóticas. En su desarrollo utilizan un robot manipulador móvil con sensor *eye-hand* o cámara en mano para explorar el objeto a reconstruir.

1.2. Planteamiento del Problema

La investigación propuesta se basa en desarrollar un sistema modular. Este sistema debe permitirle a un robot móvil cumplir tareas de reconocimiento de objetos en un contexto de robótica de servicio. El sistema modular debe tener la capacidad de ser multiplataforma. Lo anterior se refiere a que debe poder ser utilizado en cualquier plataforma robótica sin grandes modificaciones en el código. El desarrollo del mismo debe llevarse a cabo mediante técnicas óptimas de visión y navegación; bajo coste computacional y alto grado de eficiencia al resolver las tareas en un entorno semiestructurado.

Se trabaja con un entorno semiestructurado considerando que, un robot de servicio en el hogar, oficina u hospital, primero debe conocer el lugar y los elementos no dinámicos con los que va a interactuar. Por ejemplo, en la competencia *RoboCup@Home* [24], los competidores cuentan con un día y medio para conocer los escenarios en los que van a interactuar y realizar las calibraciones necesarias para el cumplimiento de las tareas propuestas. La tarea principal que se busca cumplir con este sistema es que el robot sea capaz de reconocer y entregar un objeto que el usuario requiera.

Para lograr que el sistema realice esta tarea se consideran los siguientes puntos:

1. Reconocimiento de objetos
2. Manipulación de Objetos
3. Navegación
4. Interacción Hombre-Robot

Cada uno de los puntos anteriores es considerado un módulo del sistema. El sistema a desarrollar se resume en el diagrama de bloques de la Figura 1.1. Se planea que este sistema sea mínimamente supervisado por el usuario. Por esto se considera que la tarea del usuario será dar ciertas órdenes:

- ✓ Empezar nuevo aprendizaje de objetos,
- ✓ Apoyar con la etiqueta de los objetos entrenados,
- ✓ Empezar nueva navegación y construcción de mapa,
- ✓ Comenzar búsqueda de objeto.

Para cumplir con las órdenes especificadas por el usuario, el robot debe hacer uso de los módulos correspondientes. El sistema debe ser capaz de:

1. Crear un mapa del lugar,
2. Aprender y etiquetar objetos,
3. Encontrar objeto indicado por el usuario,
4. Manipular el objeto para entregarlo.

Esta tesis se enfoca en el módulo de reconocimiento de objetos. El cual es considerado como la tarea principal del sistema modular. El resto de los módulos tienen el fin de apoyar a la pronta detección y manipulación de los objetos requeridos por el usuario mediante técnicas computacionales eficientes. El desarrollo de los módulos se enlista a continuación.

El módulo de ***Reconocimiento de Objetos*** desarrollado se divide en dos fases: *Aprendizaje* y *Reconocimiento*. En la fase de *Aprendizaje*, se considera como entrada diferentes vistas o tomas de un objeto. De estas vistas se extraen características mediante una técnica basada en parches. Se utiliza el algoritmo de detección y descripción de características multiescala *A-KAZE*[81]. La información obtenida del objeto pasa a la etapa de entrenamiento. La etapa de entrenamiento se lleva a cabo mediante una red neuronal que reconoce patrones de información para la determinación y clasificación de los objetos.

Después de la etapa de entrenamiento, se identifican las neuronas pertenecientes a los objetos aprendidos de la red neuronal entrenada mediante una etiqueta. En la fase de *Reconocimiento*, se considera como entrada una sola vista del objeto. De esta se extraen las características

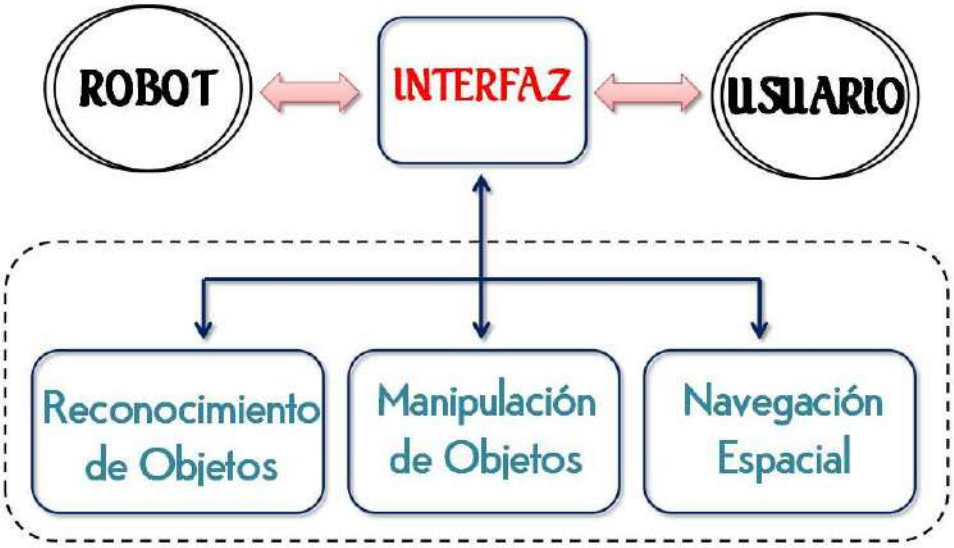


Figura 1.1: Diagrama de bloques del sistema modular.

mediante el algoritmo *A-KAZE*. Las características extraídas se envían a la etapa de evaluación. La etapa de evaluación considera la red neuronal entrenada para clasificar los patrones de información reconocidos. Se identifica la neurona ganadora y se recupera la etiqueta que se le asignó para identificar a qué objeto pertenece.

Se ha realizado una comparativa entre las redes neuronales artificiales: *Growing Cell Structure* (GCS) [76] y *Adaptative Resonance Theory* (ART) [99]. De la comparativa se ha determinado que la red neuronal que mejor desempeño ha presentado es GCS.

El módulo de ***Manipulación de Objetos*** permite localizar objetos espacialmente y manipularlos. El módulo considera que el objeto ya ha sido reconocido y que el robot se encuentra cerca del objeto como para alcanzarlo. El robot utiliza su(s) cámara(s) para detectar el objeto, estimar una posición tridimensional en el espacio real y utilizar uno o dos manipuladores para tomarlo. Se toman en cuenta aspectos como tamaño del objeto y posición la cámara para obtener una distancia aproximada. Se pretende que los objetos se encuentren a cualquier distancia dentro de su área de trabajo. Las dimensiones de los objetos no se conocen, sin embargo, estos deben estar dentro de un rango tal que el robot a utilizar pueda manipular. Se utilizan algoritmos de procesamiento de imagen 2D para estimar la posición y tamaño de los objetos. Con los contornos y las dimensiones conocidas del objeto es posible usar relaciones de proyección para determinar sus coordenadas 3D en el espacio. La imagen de la cámara y el tamaño estimado del objeto dan información acerca de la dirección y distancia desde la cámara al centro del objeto.

El módulo de ***Navegación Espacial*** se divide en dos tareas: Localización Espacial y Locomoción Bípeda. Ambas tareas deben permitirle al robot:

1. construir un mapa bidimensional del lugar donde se encuentra,

2. estimar su ubicación con respecto al mapa bidimensional,

3. buscar y ubicar el objeto pedido por el usuario,

4. mantener una buena locomoción para que no caiga.

La localización espacial se cubre mediante *Odometría* [100] apoyada por elementos visuales. Con ambos, el robot tiene la tarea de construir el mapa de la habitación donde debe buscar los objetos aprendidos. El robot considera el marco de referencia global como el punto del cual parte para realizar un recorrido alrededor de la habitación. Teniendo en cuenta su propio marco de referencia, este conoce su ubicación con respecto al marco de referencia global. La idea es que el robot guarde la relación espacial de su ubicación y lo que está observando en ese punto mientras realiza el recorrido. Con la información recabada de la habitación se construye un mapa bidimensional.

Cuando el robot deba iniciar una búsqueda del objeto requerido por el usuario, este se apoya de un enfoque de planeación basado en NBV [92]. El enfoque le permite reconocer y aproximarse lo más posible al objeto requerido por el usuario. Una vez que el robot tome el objeto, se apoya del mapa bidimensional para conocer su ubicación en la habitación y regresar al punto de referencia global para entregarle el objeto al usuario.

El módulo de ***Interacción Hombre-Robot*** consta de dos partes: *Entorno Gráfico* y *Sincronización de Tareas*. El *Entorno Gráfico* se refiere a una interfaz gráfica contenida en una aplicación desde la computadora. La interfaz es muy intuitiva, le permite al usuario dar las órdenes necesarias para que el robot le entregue un objeto. En la fase de entrenamiento, el usuario debe indicar el nombre de los objetos a aprender y en la fase de reconocimiento, el usuario le indicará al robot que objeto requiere. La interfaz se ha diseñado mediante el lenguaje de programación *Python* [101] y las librerías de procesamiento de imagen*OpenCV* [102]. La *Sincronización de Tareas* se refiere al uso del módulo de *Interacción Hombre-Robot* como el intermediario o esqueleto de la aplicación que sirve para comunicar y sincronizar cada una de las tareas. Los módulos de todo el sistema se comunican mediante este módulo, en el cual se recogen y entregan resultados que comprenden la información necesaria para ejecutar sus tareas correspondientes.

Con base a lo expuesto en las secciones anteriores y al desarrollo propuesto del sistema se plantea el siguiente escenario: teniendo un ambiente semiestructurado, se utiliza un robot servicio que realice tareas específicas. Se establece un ambiente limitado, al ser un robot de servicio, se le indica en qué tipo de entorno va a trabajar. El espacio de trabajo del robot es de 3x3 metros, en él se encontraran distribuidos los objetos en mesa de un tamaño apropiado para el robot. La primer tarea del robot es el aprendizaje de los objetos, la segunda tarea es la creación de un mapa del lugar en el que debe buscar los objetos. Así mismo, debe ser capaz de manipularlos y entregarlos al usuario.

El robot se basa en el siguiente enfoque de planeación para buscar los objetos a reconocer y aproximarse (lo más posible) de tal manera que pueda realizar manipulación del mismo. A continuación se presenta el algoritmo del enfoque de planeación:

1. Cuando el robot reciba la orden de buscar cierto objeto, este comenzará una búsqueda del objeto desde el momento en que comience a navegar.

2. Una vez identificado el objeto se procede a determinar la caja o cuadro mínimo que contiene el perímetro o borde de un objeto.

3. Con base a información de tamaño estimado del objeto e información de la cámara se procede a calcular la distancia a este.

4. Con la distancia, se le indica al robot hacia donde debe desplazarse para llegar al objeto reconocido tratando de no perderlo de vista.

5. Se realiza un bucle cerrado del algoritmo que se verificará mediante la distancia al objeto, este se detendrá cuando la distancia se encuentre dentro de un umbral de error.

6. Cuando el robot se encuentre lo más cerca posible del objeto se procede a estimar las coordenadas tridimensionales.

7. Teniendo la posición tridimensional del objeto se procede a manipularlo y a entregarlo al usuario en el punto de referencia del cual ha partido.

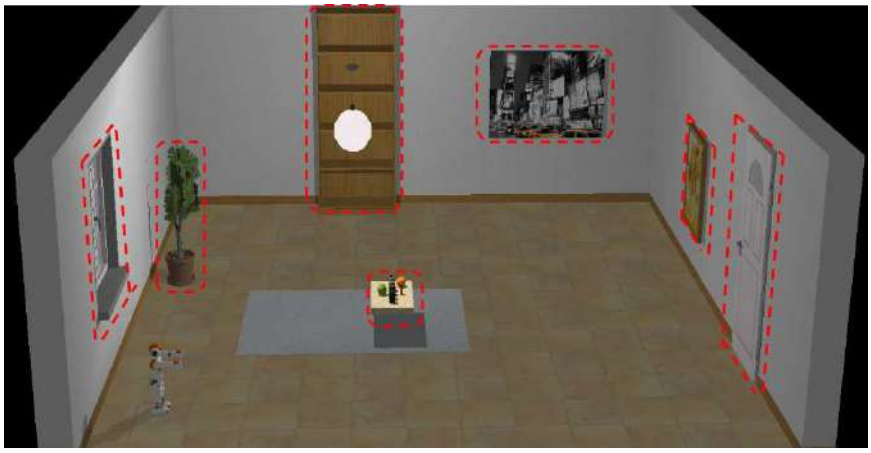


Figura 1.2: Representación del entorno semiestructurado y los elementos visuales que el robot puede tomar como puntos de referencia para la interacción.

En la Figura 1.2 se muestra un posible escenario diseñado en el software de simulación *Webots* [103] con el robot humanoide *NAO* [18]. El escenario presenta una habitación con una puerta y una ventana, además se agregaron elementos en las paredes como dos cuadros, una planta, un estante y un tapete. En el centro de la habitación se ubicaron siete objetos a reconocer en una base o mesita con una altura de 300 mm. El tamaño de la habitación es de aproximadamente 4x4 metros. El robot se ubica en una esquina de la habitación la cual puede ser su marco de referencia global si comienza a navegar desde ese punto.

En la Figura 1.3 se muestra una imagen en la que el robot se encuentra cerca de los objetos. En la esquina superior derecha se observa lo que el robot percibe con una de sus cámaras. En la imagen se aprecian tanto los elementos visuales de la habitación y los objetos a reconocer,

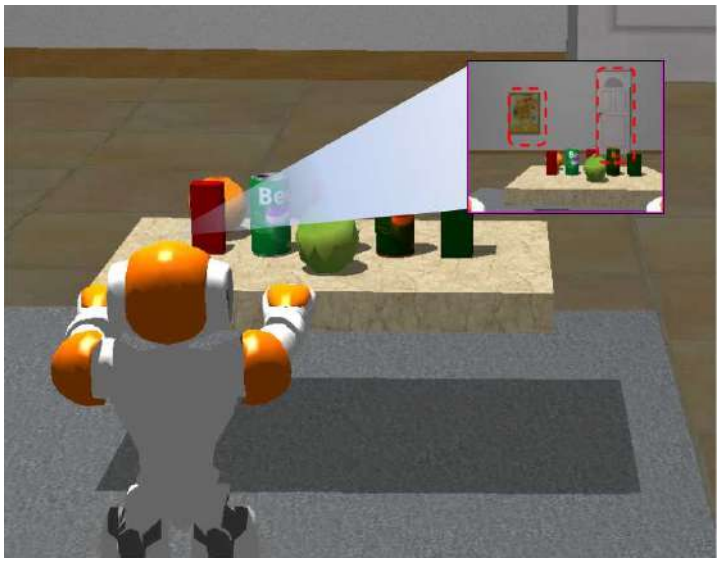


Figura 1.3: Representación de lo que el robot observa frente a una mesa con objetos.

algunos de los objetos se encuentran obstruidos. Desde este punto el robot debe empezar a estimar la posición tridimensional del objeto reconocido para su posterior manipulación.

1.2.1. Delimitaciones

Los alcances de esta tesis están definidos por el desarrollo del sistema modular. Para asegurar la capacidad del sistema a ser utilizado en diferentes plataformas robóticas se delimitarán los módulos a la resolución del problema que plantea y no a la implementación. Se definieron las siguientes delimitaciones del trabajo:

1. La estimación de posición 3D y la tarea de agarre de objetos para la manipulación es básica.

2. El módulo de manipulación se desarrollan en base a la plataforma robótica a utilizar, sin embargo, se especifican los cambios que se pueden realizar para implementarlo en otras.

3. Se realiza una validación en simulación que depende de las restricciones del mismo simulador *Webots* respecto a elementos en el entorno.

4. El módulo de reconocimiento de objetos se aplica a una base de datos propia de 20 objetos.

5. Los objetos se encuentran en cierto rango de tamaño para que el robot a usar pueda levantarlos.

6. Se establece un ambiente limitado libre de obstáculos que el robot debe explorar.

7. El espacio de trabajo del robot debe ser no mayor a 4x4 metros, en él se encontrarán distribuidos los objetos en una mesa de un tamaño apropiado para el robot en el centro de la habitación.

1.3. Aportaciones

El sistema modular que se propone desarrollar tendrá algunas aportaciones que parten de la aplicación de los métodos propuestos para la resolución de los problemas planteados. Entre las aportaciones más significativas se enumeran las siguientes:

1. Realización de un sistema modular para el control de autonomía de un robot en un contexto de robótica de servicio.
2. Aplicación del método *A-KAZE* para el reconocimiento de objetos junto con redes neuronales no supervisadas.
3. Uso del enfoque *Next Best View* en la búsqueda y aproximación a objetos en base al módulo de reconocimiento de objetos.
4. Desarrollo del módulo de *Interacción Hombre-Robot* para la sincronización de las tareas.

1.4. Justificación

Al realizar la búsqueda bibliográfica para la investigación de este tema, se encontró que se está trabajando en diversos desarrollos en robótica de servicio y se considera que el proyecto propuesto incluye temas de actualidad.

- ✓ Países de primer mundo han invertido gran cantidad de recursos en investigación sobre robótica de servicio.
- ✓ Por otro lado, se encontraron numerosos trabajos pertenecientes a universidades variadas alrededor del mundo.
- ✓ En México se han realizado investigaciones sobre algoritmos y desarrollo en robótica de servicio. Centros de investigación y universidades como el *Tecnológico de Monterrey*, UNAM, INAOE y CINVESTAV, han participado en torneos robóticos como *RoboCupSoccer* o *RoboCup@Home* con resultados favorables.

El sistema modular obtenido puede servir de base para:

- ✓ La realización de investigación en robótica humanoide.
- ✓ El desarrollo de sistemas más complejos.
- ✓ La aplicación en las asignaturas de cursos en posgrado y licenciatura.
- ✓ En la Universidad Tecnológica de la Mixteca no se han encontrado trabajos similares. Sin embargo, sí se han realizado trabajos involucrados en el área de robótica: robótica de rehabilitación por parte de manipuladores [104], robótica recreativa mediante el desarrollo de mascotas robots [105], robótica orientada a la enseñanza pedagógica [106] [107] [108], robots de exploración [109], etc. De igual manera se han realizado diversas aportaciones en inteligencia artificial mediante algoritmos de clasificación, reconocimiento de objetos, reconocimiento de patrones [110][111], control inteligente de sistemas [112][113][114], redes neuronales [115][116], etc.

1.5. Hipótesis

Con base a la propuesta de tesis definida en este trabajo se plante lo siguiente: será posible realizar un sistema modular para la autonomía de un robot, incluyendo reconocimiento de objetos, en un contexto de robótica de servicio.

1.6. Objetivos

1.6.1. General

Desarrollar un sistema modular que le permita a una plataforma robótica realizar tareas de reconocimiento y manipulación de objetos considerado dentro de un contexto de robótica de servicio, conociendo su localización espacial respecto a un marco de referencia global.

1.6.2. Específico

El proyecto de tesis comprendido contempla como objetivos específicos el desarrollo de cada uno de los módulos del sistema siendo estos los enlistados a continuación:

1. Implementar el módulo de *Reconocimiento de Objetos* mediante el uso del enfoque *A-KAZE* y redes neuronales no supervisadas.

2. Realizar el módulo de *Manipulación de Objetos* mediante procesamiento de imagen y cinemática de manipuladores.

3. Desarrollar el módulo de *Navegación Espacial* mediante un enfoque NBV considerando su posición con respecto a un punto de referencia.

4. Diseñar y construir el módulo de *Interacción Hombre-Robot* mediante el lenguaje de programación *Python* y la librería de procesamiento de imagen *OpenCV*.

1.7. Estructura de la Tesis

La tesis *Implementación de un sistema modular para el reconocimiento de objetos en un contexto de robótica de servicio* está comprendida por 8 capítulos en los cuales se distribuye la realización de los módulos propuestos.

Para continuar con el contenido de la tesis, el Capítulo 2 se enfoca al ***Marco Teórico*** donde se explican las herramientas a utilizar para el desarrollo de los módulos.

El desarrollo del módulo de ***Reconocimiento de Objetos*** se presenta en el Capítulo 3. Este contiene la estructura del algoritmo de extracción de características, la implementación del módulo con la base de datos de objetos reales y los resultados experimentales del módulo.

El desarrollo del módulo de ***Manipulación de Objetos*** se plasma en el Capítulo 4. El análisis cinemático y dinámico con la plataforma robótica. Además, contiene el proceso de extracción de características, determinación de tamaño y estimación de la posición 3D del objeto a manipular. Además de la implementación y resultados del módulo.

El Capítulo 5 comprende el módulo de ***Navegación Espacial***. Se plasma el desarrollo del algoritmo de navegación espacial apoyado por elementos visuales. También, comprende la proposición del algoritmo Next Best View para seguimiento de objeto. De igual forma se incluye la implementación y resultados del módulo.

La implementación del módulo de ***Interfaz Hombre-Robot*** se describe en el Capítulo 6. En este se plasma el desarrollo de la interfaz gráfica mediante Python para la sincronización del sistema modular.

El Capítulo 7 se ha destinado para mostrar la implementación del sistema modular tanto en simulación como con la plataforma robótica NAO. Se detallan experimentos realizados y los resultados obtenidos con estos.

Las conclusiones de este trabajo de tesis se detallan en el Capítulo 8. En este también se incluyen discusiones sobre el sistema modular, dificultades y trabajos futuros.

Capítulo 2

Marco Teórico

Este capítulo comprende la descripción teórica de las herramientas y algoritmos utilizados para el desarrollo de cada uno de los módulos del sistema. En la sección 2.1, se abordan una introducción concisa a visión computacional, la cual continua con la descripción del método de extracción de características *A-KAZE* en la sección 2.2. En la sección 2.3, se da una explicación breve sobre redes neuronales, se especifican los modelos de redes neuronales no supervisadas y el desarrollo de su comparativa. Además, en la sección 2.4, se detallan las herramientas a utilizar en la planificación de trayectorias de manipuladores, en la sección 2.5, se aborda la odometría inercial para la navegación espacial junto con el método NBV y la información referente a locomoción bípeda. Asimismo, en la sección A.1, se describe la plataforma robótica *NAO* junto con el software de simulación *Webots*.

2.1. Visión Computacional

En las plataformas robóticas, la tarea de recibir información sobre el estado de su entorno puede ser abordada con diferente tipo de sensores tales como: ultrasónicos, sensores de fuerza, infrarrojos, cámaras de vídeo, etc. Siendo la cámara más utilizada por su bajo costo y el gran contenido de información que se puede adquirir de estas. La información que se adquiere de las cámaras son imágenes que se procesan y analizan para ser utilizadas en la toma de decisiones. Al procesamiento de imágenes y extracción de información para el uso en una aplicación se le conoce como visión computacional. El propósito de la visión computacional es realizar algoritmos con el fin de "entender" una escena o las características de una imagen. Según [56], se basa en el uso de la computadora para simular la visión humana, incluyendo el aprendizaje, la capacidad de tomar decisiones y acciones basado en entradas visuales. Según [117], los objetivos típicos de la visión computacional son:

- ✓ La detección, segmentación, localización y reconocimiento de objetos(e.g.: caras humanas).
- ✓ La evaluación de los resultados (e.g.: segmentación, registro).
- ✓ Registro en diferentes imágenes de una misma escena, i.e.: coincidencia de objeto en diversas imágenes.
- ✓ Seguimiento de un objeto en una secuencia de imágenes.

- ✓ Mapeo de una escena para generar un modelo tridimensional (e.g.: navegación robótica).
- ✓ Estimación de posturas tridimensionales de humanos.
- ✓ Búsqueda de imágenes digitales por su contenido.

La visión computacional, requiere de ciertos procedimientos realizados secuencialmente para llegar a los objetivos mencionado anteriormente. Estos procedimientos se encuentran definidos como *Procesamiento de Imágenes* y *Análisis de Imágenes*. El *Procesamiento de Imágenes* realizado mediante cómputo digital tiene como entrada y salida una imagen. Entre los procesos se incluyen: operaciones primitivas (de nivel bajo), como el pre-procesamiento para reducción de ruido, realce de contraste y agudizamiento de imagen; operaciones de segmentación (nivel medio), descripción de imágenes, reducción a formas manejables para el procesamiento computacional y la clasificación (reconocimiento) de objetos individuales (bordes, contornos, etc); interpretación (nivel alto), dar sentido a las características descritas [56].

El *Análisis de Imágenes* se encarga de darle sentido a las descripciones de las imágenes después del *Procesamiento de Imágenes*. Este análisis es llevado a cabo por diferentes métodos que utilizan características externas (longitud, orientación) e internas (color y textura) para la representación de las regiones [118]. Las características son la esencia de una imagen o escena. Estas son típicamente escalares o vectores cortos, que representan un resumen de la información presente en los píxeles. Entre los métodos de características externas se encuentran: los códigos de cadenas, aproximación poligonal, firmas, segmentación de bordes, esqueletos, etc. De igual manera, existen descriptores de bordes como la longitud, diámetro, eje mayor y menor para formar el mínimo rectángulo que lo contiene, esquinas, curvatura, descriptores de *Fourier*, momentos estáticos, media y mediana de niveles de gris, etc. Mientras que entre los métodos que utilizan características internas se encuentra la segmentación de imágenes mediante discontinuidades, transformada de *Hough*, umbralización multiescala, procesamiento de histogramas, regiones crecientes, segmentación en diferentes espacios de color, descriptores de puntos de interés, etc.

Específicamente hablando, en esta tesis se hará uso de un método *A-KAZE* para la detección de características tipo puntos en espacio de escala. Estos puntos son visualmente distintivos en la imagen y son llamados puntos de interés, puntos salientes, *keypoints*, o puntos esquina. Estos métodos determinan la escala y la orientación de los puntos salientes, además, son conocidos como detectores y descriptores de características invariantes locales. Los métodos más populares de este tipo de detección de características son: el método *Scale Invariant Feature Transform* (SIFT)[39], el cual, se basa en el uso de una secuencia de *Gaussianas* para encontrar el valor máximo de la diferencia de esta; el método *Speeded Up Robust Feature* (SURF)[35], basado en el uso de una secuencia *Gaussiana* para encontrar el valor máximo de la diferencia de esta pero haciendo uso de una aproximación de la determinante *Hessiana*. El resultado de ambas son diversos elementos con sus respectivas características de coordenadas: escala, orientación y descripción. Estas características son utilizadas para realizar una correspondencia de características. La correspondencia se refiere al problema de encontrar las coordenadas de un pixel en dos imágenes diferentes, lo cual se refiere al mismo punto pero en el mundo real.

2.2. A-KAZE

El método *A-KAZE* [81], que se resume en la Figura 2.1, se divide en tres tareas principales: la construcción de un espacio de escala no lineal, la detección de características y la descripción de características. En la construcción del espacio de escala no lineal se considera una imagen de entrada, de la cual se construye el espacio de escala no lineal utilizando el método numérico *Fast Explicit Diffusion* (FED) [82] con un enfoque piramidal.

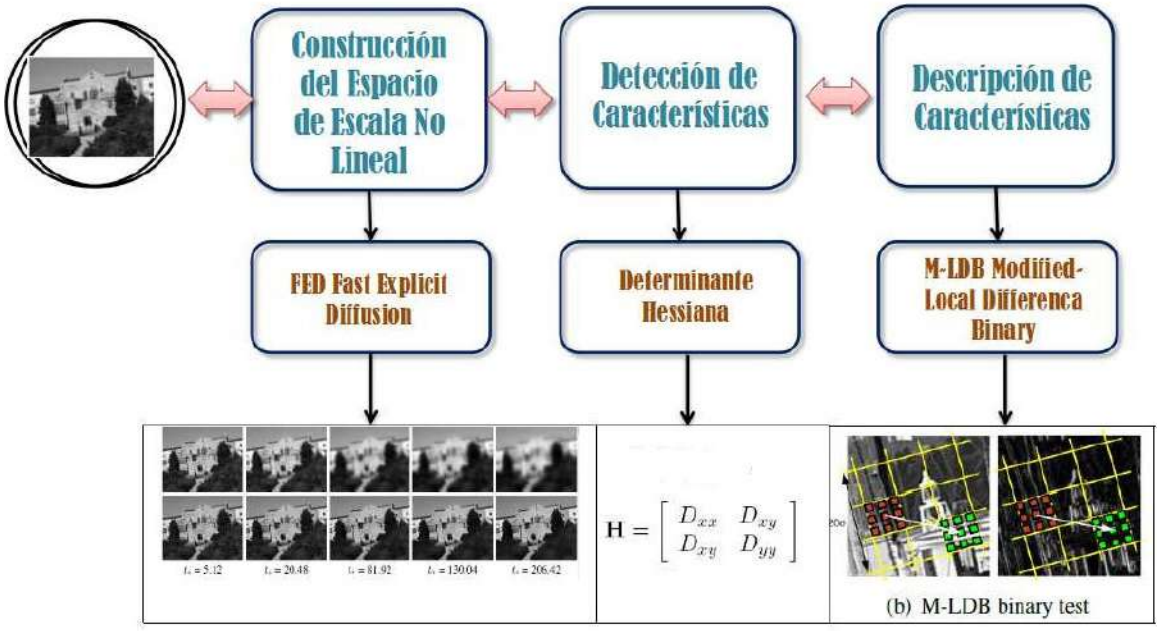


Figura 2.1: Descripción general del algoritmo de detección y descripción de características multiescala A-KAZE.

Primero se define un conjunto de tiempo de evolución. El espacio de escala se discretiza en una serie de O octavas y S subniveles. El conjunto de octavas y subniveles son identificados por un índice discreto (o y s respectivamente). La octava y el subnivel son mapeados a su correspondiente escala σ (pixel) mediante (1).

$$\sigma_i(o,s) = 2^{(o+s)/S}, o \ \in [0...O-1], \ s \ \in [0...S-1], i \ \in [0...M] \tag{1}$$

donde M es el número total de imágenes filtradas. Después, se convierten a unidades de tiempo t_i mediante (2).

$$t_i = \frac{1}{2}\sigma_i^2, \ i = 0...M \tag{2}$$

Adicionalmente, la imagen de entrada es convolucionada con una *Gaussiana* de desviación estándar σ_0 para reducir ruido y posibles artefactos. Desde la imagen de entrada suavizada se calcula el factor de contraste λ de manera automática como el 70 % del histograma gradiente. Dada la imagen de entrada y el factor de contraste se inicia el esquema FED. La idea principal es ejecutar M ciclos de n pasos explícitos de difusion variando el tamaño de paso τ_j originado

de la factorización de la descomposición en cajas de filtros en términos del esquema explícito (3):

$$\tau_j = \frac{\tau_{max}}{2\cos^2(\pi\frac{2j+1}{4n+2})} \tag{3}$$

donde τ_{max} es el paso máximo que no viola la condición de estabilidad del esquema explícito. El tiempo de ciclo FED θ_n definido en la ecuación (4) se refiere al tiempo de parada del ciclo obtenido de la siguiente manera:

$$\theta_n = \sum_{j=0}^{n-1} \tau_j = \tau_{max}\frac{n^2+n}{3} \tag{4}$$

Algunos de los tamaños de paso τ_j de la ecuación (3) pueden violar la condición de estabilidad. Así que se obtiene un esquema estable al final de cada ciclo FED. La discretización de la ecuación (3) usando un esquema explícito se puede expresar como notación vector-matriz:

$$\frac{L^{i+1} - L^i}{\tau} = A(L^i)L^i \tag{5}$$

donde $A(L^i)$ es una matriz que codifica las conductividades de la imagen y τ es la constante de tiempo, tal que $\tau < \tau_{max}$, para respetar la condición de estabilidad. En el esquema explícito, la solución L^{i+1} se calcula de manera directa desde la solución de la evolución de nivel de L^i y la conductividad de la imagen $A(L^i)$:

$$L^{i+1} = (I + \tau A(L^i))L^i \tag{6}$$

donde I es la matriz de identidad. Considerando la estimación a priori $L^{i+1,0} = L^i$, un ciclo FED con n variables de tamaño de paso τ_j se obtiene como:

$$L^{i+1,j+1} = (I + \tau_j A(L^i))L^{i+1,j},\ j = 0, ..., n-1 \tag{7}$$

Las no linealidades de la matriz $A(L^i)$ permanecen constantes durante todo el ciclo FED. Una vez terminado el ciclo FED, se calculan los nuevos valores de la matriz $A(L^i)$.

Entonces, se usan $M-1$ ciclos FED exteriores y por cada uno se calcula el número mínimo de pasos interiores n. Para imágenes 2D, el tamaño de paso máximo que no viola las condiciones de estabilidad es $\tau_{max} = 0{,}25$ considerando un tamaño de rejilla de un pixel por derivadas de imágenes. Se considera que cada ciclo externo FED cubre un tiempo de ciclo $T = t_{i+1} - t_i$. El tiempo de ciclo FED θ_n definido en la ecuación (4), cubre sólo un conjunto de valores discreto. Para evadir tiempos arbitrarios se propone T, calculando la longitud mínima del ciclo n con $\theta_n > T$ y multiplicando el tamaño de paso τ_j de la ecuación (3) con el factor $q = \frac{T}{\theta_n}$. Cuando se alcanza el último subnivel de cada octava, se muestrea la imagen con un factor de 2 usando una máscara de suavizado $(\frac{1}{4}, \frac{1}{2}, \frac{1}{4})$ y se utiliza como la imagen inicial para el siguiente ciclo FED en la siguiente octava. Después, se modifica el factor de contraste λ y la máscara suavizante reduce el contraste un 25 %, haciendo necesario multiplicar por 0.75. Los algoritmos se presentan en 1 y 2.

Después se detectan características 2D de interés que exhiban un determinante de escala normalizada de la respuesta *Hessiana* [119] a través del espacio de escala no lineal para cada

Algoritmo 1: Enfoque piramidal FED para filtro de difusión no lineal.

Datos: L_0 Imagen, λ parámetro de contraste, τ_{max} y conjunto de evolución en el tiempo t_i

Resultado: Conjunto de imágenes filtradas $L_i,\ i = 0...M$

para $i = 0 \rightarrow M - 1$ **hacer**
- 1. Calcular matriz de difusividad $A(L^i)$ ec.(5)
- 2. Establecer tiempo de ciclo externo FED $T = t_{i+1} - t_i$
- 3. Calcular número de pasos internos n
- 4. Calcular tamaño de paso τ_j
- 5. establecer prioridad $L^{i+1,0} = L^i$ ec.(6)
- $L^{i+1} =$ **Ciclo FED**$(L^{i+1,0}, A(L^i), \tau_j)$
- **si** $o_{i+1} > o_i$ **entonces**
  - Muestreo L_{i+1} con máscara $(\frac{1}{4}, \frac{1}{2}, \frac{1}{4})$
  - $\lambda = \lambda 0{,}75$

Algoritmo 2: Ciclo FED.

Función Ciclo FED $(L^{i+1,0}, A(L^i), \tau_j)$
para $j = 0 \rightarrow n - 1$ **hacer**
- $L^{i+1,j+1} = (I + \tau_j A(L^i))L^{i+1,j}$ ec.(7)

return $L^{i+1,n}$

imagen filtrada. La normalización se realiza utilizando un factor que toma en cuenta la octava de cada imagen, en particular, en el espacio de escala no lineal como se muestra en la ecuación (8).

$$L^i_{Hessiana} = \sigma^2_{i,norm}(L^i_{xx}L^i_{yy} - L^i_{xy}L^i_{yx}), \quad \sigma_{i,norm} = \sigma_i/2^{o^i} \tag{8}$$

Para el cálculo de las derivadas de segundo orden se utiliza el *Filtro Concatenado Scharr* [120] con tamaño de paso $\sigma_{i,norm}$. El *Filtro Scharr* aproxima la invarianza de rotación mejor que otros filtros o la diferenciación central. Primero se busca la respuesta máxima del detector en la locación espacial. En cada nivel de evolución i, se puede observar que la respuesta del detector es más alta que un umbral predefinido y es una máxima en una ventana de píxeles 3x3. Lo anterior se hace descartando respuestas máximas. Por cada máxima potencial, se observa que la respuesta es una máxima con respecto a otros puntos clave o *keypoints* del nivel $i + 1$ e $i - 1$, respectivamente por encima y por debajo directamente en una ventana de tamaño $\sigma_i \times \sigma_i$ píxeles. Finalmente, la posición 2D del punto clave es estimada con precisión de sub-pixel adecuando una función cuadrática a la respuesta de la determinante *Hessiana* en un vecindario 3x3 y encontrando su máxima.

Finalmente, se calcula la orientación principal del *keypoint* mediante el descriptor *Modified-Local Difference Binary* (M-LDB)[83]. Este método utiliza información de gradientes e intensidad del espacio de escala no lineal. Se propone usar una rejilla de pasos finos dividiendo los parches en 2x2, 3x3 y 4x4, etc. El promedio de esas subdivisiones son calculadas rápidamente

utilizando imágenes integrales si el descriptor es correcto. Cuando se considera la rotación de los puntos clave no se utilizan las imágenes integrales y visitar todos los puntos en una subdivisión rotada puede ser relativamente caro en tiempo de cómputo. Se sub prueban las rejillas en pasos en función de la escala característica σ. El muestreo de escala en turno convierte el descriptor robusto a cambios de escala. El descriptor M-LDB usa el cálculo derivativo en el paso de detección de características, reduciendo el número de operaciones requeridas para construir el descriptor. Además, calcula una aproximación del promedio de las mismas áreas de las imágenes de intensidad y gradiente. Los valores booleanos que resultan de la comparación no son independientes unas de otras.

2.3. Redes Neuronales Artificiales

Las *Redes Neuronales Artificiales* (RNA) parten de la neurona biológica (ver Figura 2.2 (a)) la cual es una célula nerviosa que se encuentra en el cerebro. Esta neurona está formada por:

- ✓ Un núcleo, que es el cuerpo donde se realiza la mayor parte de la computación neuronal.
- ✓ Dendritas, que son las ramificaciones a través de las cuales una neurona recibe señales de entrada.
- ✓ Axón, que es una ramificación cuyo objetivo es propagar la señal.
- ✓ Región presináptica, que es la parte final del axón donde este se ramifica.
- ✓ Una región denominada unión sináptica la cual es donde la neurona transmite su señal a otras neuronas.

El sistema nervioso de los animales se estructura con ese tipo de neuronas las cuales forman redes para el procesamiento de información generada por los diversos estímulos de entrada de los sistemas sensoriales. Dichas redes, han servido también de inspiración para modelar las RNA, las cuales son interconexiones de neuronas artificiales [121]. McCulloch y Pitts [48], fueron los precursores de estas redes, argumentando que éstas funcionan como un mecanismo booleano. Gracias a esa afirmación se generó una neurona como un modelo lineal seguido de una función de activación booleana.

Según [48], el modelo de la neurona lineal se visualiza como una entidad que cuenta con n entradas subíndice $i(i = 1, 2, ..., n)$, las cuales se identifican como las dendritas de una neurona biológica. En cada entrada se recibe un valor proveniente de otra neurona, o bien, del mundo exterior. El conjunto de éstas, puede ser representado como un vector con n componentes que tendrán asociados un peso w_i el cual tiene la función de neuro-transmitir.

El peso se asocia al valor recibido en la entrada i. Además, la combinación de ambas características se refiere a la conexión como excitadora o como inhibidora. Todo par (entrada i y peso w_i) se lleva al núcleo en donde se efectúa el cómputo neuronal. El resultado es un valor distribuido en m salidas, que sería la sinapsis en una neurona biológica. Estas salidas pueden

conectarse con el mundo exterior o a otra neurona. En la Figura 2.2 (b) se representa el modelo de RNA en el que se representa a la sinapsis entre neuronas como función lineal y el procesamiento que hace la neurona como la función de tipo sigmoide.

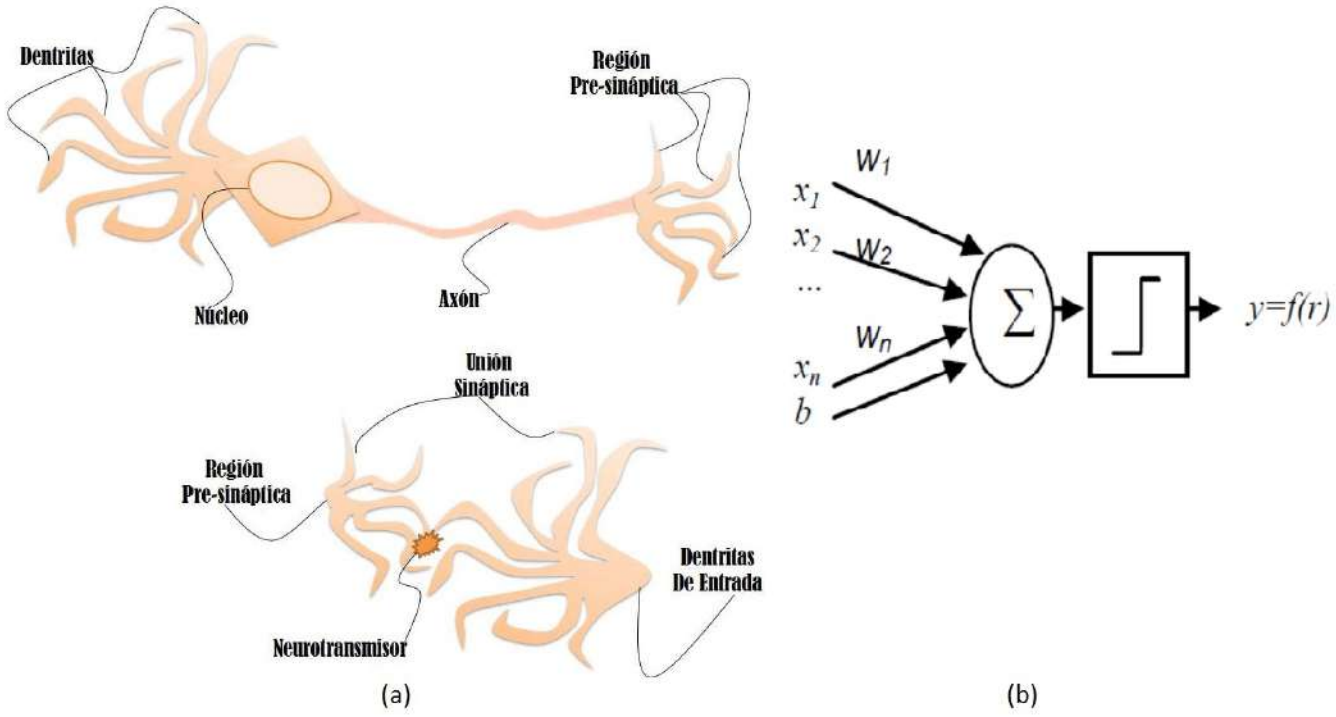


Figura 2.2: (a) Representación de una *Neurona Biológica* con sus respectivos componentes, (b) Modelo lineal con función de activación booleana de neurona artificial.

Las neuronas que forman parte de una red están interconectadas y agrupadas en capas que pueden ser de tres tipos:

1. Capa de Entrada, conjunto de neuronas que recibe información proveniente del exterior.
2. Capa de Salida, conjunto de neuronas que proporciona la salida final en la red.
3. Capas Ocultas, conjunto de neuronas ubicadas entre las dos capas anteriores.

Así mismo, las RNA pueden ser clasificadas como *redes neuronales supervisadas* o *redes neuronales no supervisadas*. En las *redes neuronales supervisadas* el cómputo neuronal se basa en un entrenamiento que busca el ajuste de pesos para la obtención de salidas deseadas. En una RNA se considera a los conjuntos de vectores de entrada y los de salida esperada como el conjunto de entrenamiento de la red. Este conjunto se utiliza para ajustar a las neuronas que forman una red. El procedimiento de ajuste trata de modificar los pesos de las neuronas de manera que para cualquier entrada, el vector de salida proporcionado por la red sea el esperado.

El primer modelo de *redes neuronales supervisado* utilizado en el área de procesamiento de imágenes fue el *Perceptrón* [122], el cual fue probado como un detector de caracteres ópticos. El

Perceptrón simple es un clasificador lineal de patrones en dos clases. Según la ecuación (9), éste toma un vector de características $x = (x_1, x_2, ..., x_n)$ como entrada y produce una salida escalar simple. El proceso de clasificación se completa mediante una función umbral b. Esta función es una entrada x_0 que se mantiene como una constante de unidad. La salida final del clasificador se presenta en la ecuación (10).

$$r = \Sigma_{i-1}^{n} x_i w_i + b \tag{9}$$

donde, x_i son los datos de entrada, w_i son los pesos sinápticos y b es un factor de polarización o función umbral. Esto es procesado por una función binaria que da uno o cero a la salida según r.

$$y = f(r) = f(\Sigma_{i=0}^{n} x_i w_i) \tag{10}$$

En las *redes neuronales no supervisadas*, se conocen solamente las entradas del sistema. Estas se utilizan en situaciones donde el objetivo es el de clasificar sin supervisión. Estos procesos tratan de determinar la estructura que forman los vectores de entrada por sus similitudes. También conocidas como redes neuronales auto-organizadas. El conjunto de vectores de entrada constituye lo que se llama el espacio o ambiente de la red. Los datos de entrada permiten ajustar los pesos de las neuronas de manera apropiada. Tal adaptación producirá una representación interna del espacio o ambiente.

Uno de los modelos más populares sobre las *redes neuronales no supervisadas* es el propuesto por el finlandés Teuvo Kohonen, también llamado *Mapa Auto-Organizado de Kohonen* (SOM) [121]. Este modelo se basa en un proceso competitivo donde cada neurona en una red neuronal se vuelve sensitiva a las categorías de los patrones de entrada. Generalmente, tiene una arquitectura bidimensional y utiliza el aprendizaje competitivo para que las neuronas sin una distribución de datos específica se vayan aproximando a la distribución de los datos de entrada. Esta red se ha utilizado mucho en reconocimiento de patrones y en análisis de texto. Las neuronas pueden reorganizar grupos de vectores de entradas similares generando un mapa topográfico de los vectores de entrada a los de salida resultando una reducción del espacio de entradas. El algoritmo de aprendizaje permite la clusterización de la información de entrada en un conjunto de elementos más pequeño con las mismas características. Está basado en técnicas de aprendizaje competitivo conocido como la estrategia del ganador toma todo. Según [123], entre las aplicaciones más comunes para este modelo de RNA se encuentra el reconocimiento del habla, codificación de vectores, segmentación de textura y diseño de controladores no lineales.

2.3.1. Growing Cell Structures

Conforme se van generando más conocimiento en la forma en cómo funciona el sistema nervioso de los animales, nuevos modelos de RNA surgen. Uno de ellos es *Growing Cell Structures* (GCS). El modelo de RNA GCS [76] cuenta con dos variantes: supervisado y no supervisado. La variante que realiza aprendizaje no supervisado puede ser utilizada para la visualización de datos, agrupamiento y cuantización vectorial. Su principal ventaja es la habilidad del modelo de automáticamente encontrar una estructura y tamaño de red adecuada. Esto se realiza mediante un proceso de crecimiento controlado que incluye eliminación ocasional de unidades. Este mo-

delo es una extensión del trabajo de Kohonen [121], sobre los mapas auto-organizados.

El modelo se basa en la resolución de cierto tipo de problemas. Se tiene un número n–dimensional de señales de entrada que obedecen a una probabilidad de distribución $P(\xi)$ desconocida. Con $V = \mathbb{R}^n$ como el espacio vectorial del cual se derivan las señales de entrada. El objetivo principal es generar un mapeo desde V hasta una estructura topológica discreta k-dimensional A. La topología inicial de la red A es un simplex k-dimensional donde para $k = 1$ representa un segmento de línea, $k = 2$ un triángulo, $k = 3$ o mayor se representan como un tetraedro o hipertetraedro. Los vértices $(k + 1)$ del simplex son las células o neuronas c y los bordes $(k + 1)k/2$ denotan las relaciones topológicas de los vecindarios.

El modelo GCS utiliza hipertetraedros ya que cuentan con una complejidad mínima y pueden ser combinados en grandes estructuras. El número de vértices de un hipertetraedro k-dimensional crece linealmente con k. Cada célula c tiene un vector sináptico n-dimensional W_c adjunto. Este se considera como la posición de c en el espacio vectorial. Se denota como W a todo el conjunto de vectores sinápticos $W_i, i \in A$. Un mapeo ϕ_w desde el espacio vectorial de entradas V hacia la red A se define como el mapeo de cada señal de entrada a la célula con la posición más cercana. De manera formal:

$$\phi_W : V \rightarrow A, \ (\xi \in V) -> (\phi_W(\xi) \, \epsilon \, A) \tag{11}$$

con $\phi_W(\xi)$ como la unidad de mejor resultado definida como:

$$\|W_{\phi_{W(\xi)}} - \xi\| = \min_{r \epsilon A} \|W_r - \xi\| \tag{12}$$

De esa manera $\| \cdot \|$ denota la norma Euclidiana del vector . Con esto V se particiona en un número de renglones $F_i(i \in A)$, siendo cada uno la localización de los vectores sinápticos W_i más cercanos en común. Esto se conoce como mosaico de Voronoi y las regiones se denotan como regiones de Voronoi. La adaptación del vector sináptico se realiza como la actualización de los mapas auto-organizados:

1. Se determina la unidad de mejor resultado para la señal de entrada actual.
2. Se incrementa la coincidencia en la unidad y su vecindario topológico.

En el modelo de Kohonen la fuerza de adaptación decrementa de acuerdo a un itinerario estricto mientras que en GCS:

- ✓ La adaptación de fuerzas es constante sobre el tiempo. Se utilizan parámetros de adaptación constantes ξ_b y ξ_n para la unidad de mejor resultado y el vecindario de la célula.
- ✓ Sólo la unidad de mejor resultado y su vecindario topológico directo son adaptados.

El conjunto del vecindario topológico directo de una célula c se denota mediante N_c. Para cada célula c, se define una variable contable τ_c básicamente conteniendo el número de señales de entrada para la cual la célula tiene una unidad de mejor resultado. Ya que el método dispersa las células, las señales recientes deben volverse más fuertes. Entonces, después de cada adaptación, la variable contable decrementa todas las variables por cierta fracción. El paso de adaptación se realiza de la siguiente forma:

1. Se elige una señal de entrada ξ de acuerdo a la probabilidad de distribución $P(\xi)$.
2. Se localiza la unidad de mejor resultado $s = \phi_{W(\xi)}$.
3. Se incrementa la coincidencia para s y su vecindario topológico directo.

$$\Delta W_s = \xi_b(\xi - W_s) \tag{13}$$

$$\Delta W_c = \xi_n(\xi - W_c) \; \forall \; c \; \in \; N_s \tag{14}$$

4. Se incrementa la señal del contador para s:

$$\Delta \tau_s = 1 \tag{15}$$

5. Se decrementan todas las señales contables por una fracción α:

$$\Delta \tau_c = -\alpha \tau_c \; \forall \; i \; \in \; A \tag{16}$$

Si se eligen valores ξ_n y ξ_b pequeños, entonces las células se mueven desde sus posiciones iniciales aleatorias a localizaciones con un equilibrio dinámico entre cambios de todas direcciones. El objetivo es conseguir una estructura en la que los vectores sinápticos W_c estén distribuidos de acuerdo a $P(\xi)$. Esto se logra cuando todas las células tengan la misma probabilidad de ser la mejor unidad para el vector de entrada actual. La frecuencia relativa $P(\xi)$ de las señales de entrada recibidas por una célula en particular está dada por:

$$h_c = \tau_c / \Sigma_{j \in A} \; \tau_j \tag{17}$$

Todas las células deben tener frecuencias de señales relativas similares. Un valor alto h_c indica una buena posición para insertar una célula nueva. Después de un cierto número λ de pasos de adaptación se determina la célula q con la siguiente propiedad:

$$h_q \geq h_c \; \forall \; c \; \in \; A \tag{18}$$

Luego se busca el vecindario directo de q con la distancia mas grande en el espacio de entrada. Esta es una célula f que satisface:

$$\|W_f - W_q\| \geq \|W_c - W_q\| \; \forall \; c \in \; N_q \tag{19}$$

Se inserta una nueva célula en r entre q y f. Esta nueva célula se conecta a otras células de tal forma que se tenga de nuevo una estructura simple k–dimensional. Se conecta r a q, f y el vecindario común de q y f se remueven. El vector sináptico r se inicializa como:

$$W_r = 0{,}5(W_q + W_f) \tag{20}$$

La inserción de r lidia con la nueva región de Voronoi F_r en el espacio de entrada. Al mismo tiempo las regiones de Voronoi de los vecindarios topológicos de r son disminuidos. Este cambio se refleja por una redistribución acorde de las variables contables τ_c. Se calculan los cambios de las señales contables como:

$$\Delta \tau_c = \frac{|F_c^{new}| - |F_c^{old}|}{|F_c^{old}|} \tau_c \; \forall \; c \; \in \; N_r \tag{21}$$

$|F_c|$ es el volumen n–dimensional de F_c. Finalmente el valor inicial de las nuevas células se definen como:

$$\tau_r = -\Sigma_{c \in N_r} \Delta\tau_c \tag{22}$$

Algoritmo 3: Algoritmo Principal GCS:

```
1 Start: simplex k–dimensional con posiciones aleatorias en V = ℝ^n
2 mientras Tamaño deseado de la red∼ hacer
3 |   Ejecución de número constante λ de pasos de adaptación
4 |   Inserción de célula nueva
5 |   Distribución de variables contables según ec. (21) y (22)
```

La característica principal del modelo es que cada cierto número de pasos de adaptación se realiza una inserción simple. El Algoritmo 3 describe el modelo GCS en la que se puede notar la siguiente relación de retroalimentación entre los dos tipos de acción:

- ✓ Cada paso de adaptación incrementa la señal contable de la unidad de mejor resultado y esta incrementa la posibilidad de que otra célula pueda ser insertada cerca de esta célula.
- ✓ La inserción cerca de una célula c decrementa el tamaño del campo de Voronoi F_c y el valor de la señal contable τ_c. La reducción del campo de Voronoi hace menos probable que c sea la mejor unidad de resultado para señales futuras de entrada.
- ✓ Inserción de neurona: Un nuevo nodo r siempre se inserta dividiendo el borde existente $\bar{qf}$. El nodo r debe ser conectado con q, f y con todos los vectores en común de q y f.
- ✓ El hipertetraedro también debe ser actualizado. Cada hipertetraedro h que contiene q y f, se reemplaza por dos hipertetraedros que contienen el mismo conjunto de nodos que h excepto q. Entonces, f es reemplazado por el nuevo nodo r. Finalmente, el borde original $\bar{qf}$ es removido. El nuevo hipertetraedro debe ser insertado en los conjuntos asociados con sus nodos participantes.
- ✓ Eliminación de neurona: Para eliminar un nodo, es necesario y suficiente eliminar todos los hipertetraedros que son parte del nodo. Se remueve el hipertetraedro desde el conjunto asociado con sus nodos. Los bordes de los cuales el nodo final respectivo no comparte al menos un hipertetraedro es removido, así, la estructura resultante k–dimensional es consistente.

El proceso de crecimiento y eliminación se muestra en la Figura 2.3. En la parte superior se observa que la red parte de una estructura simplex k-dimensional de dos o tres neuronas. Ya que este método es una variante de los mapa auto-organizados de Kohonen, utiliza la estrategia competitiva donde el vencedor gana todo. Mediante esa estrategia, todos los datos de entrada se distribuyen en las neuronas iniciales. Después de cierto número de pasos de adaptación se verifica entre cuales neuronas es necesario insertar una nueva. La red insertará neuronas formando tetraedros hasta alcanzar el tamaño máximo de neuronas indicado. En la misma Figura 2.3, en la parte inferior, se observa el proceso de eliminación de una neurona y su vecindario. La eliminación de neurona se lleva a cabo después de cierto número de pasos de adaptación,

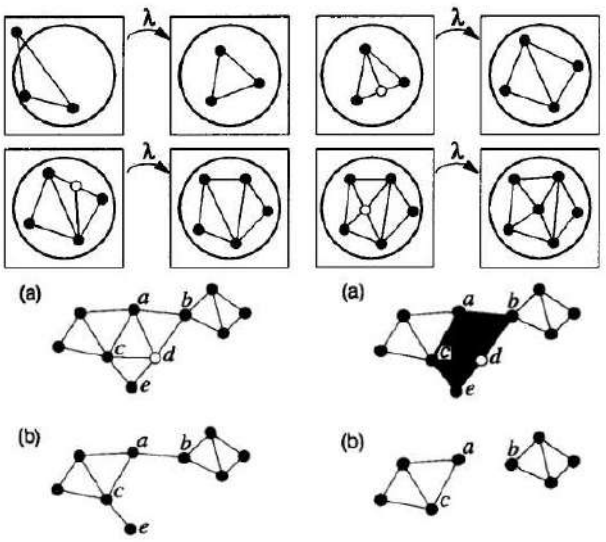


Figura 2.3: Crecimiento de neuronas en la red GCS (parte superior), eliminación de neurona (parte inferior izquierda) y eliminación de su vecindario (parte inferior derecha), (Imagen extraída de [76]).

al igual que la inserción. La red verifica si existen neuronas muertas o con pesos despreciables para eliminar ésta y su vecindario.

Para este trabajo de tesis, se ha optado por el uso de una red neuronal no supervisada por sus características particulares. Este tipo de redes trabajan como asociadores de datos, distribuyen las neuronas de acuerdo con la atracción o repulsión entre ellas, manejan estrategias competitivas y son fáciles de implementar. Se eligen las redes GCS y la *Adaptative Resonance Theory* (ART 2) (Apéndice B), ya que ambas cuentan con inserción de clases o neuronas. Esta inserción la realizan sólo si es necesario contar con otra neurona para clasificar la entrada que no encaja en las ya existentes. Por esta razón se decidió realizar un estudio comparativo entre estos modelos. Este estudio comparativo se presenta en el Apéndice A. En este estudio se demuestra, mediante algunos experimentos, que la red GCS cuenta con mejor desempeño que la red ART 2.

2.4. Manipulación de Objetos

El sistema modular desarrollado en esta tesis cuenta con un módulo de manipulación de objetos. Este módulo pretende ser utilizado para cualquier plataforma robótica móvil con al menos un brazo manipulador y una pinza como efector final para que cumpla tareas de robótica de servicio.

Se sabe que la manipulación de objetos por brazos robóticos en la industria está extensamente estudiada y resuelta con precisión. En cambio, la manipulación en un entorno semiestructurado en comparación con un entorno estructurado en su totalidad representa algunas dificultades. A lo anterior se le suma que se va a considerar manipulación por parte de una plata-

forma robótica de móvil lo que vuelve al problema más interesante. Un robot móvil de servicio, por lo general, puede llegar a contar con uno o dos manipuladores, los cuales se consideran sus brazos. Estos manipuladores pueden ser de tres a seis grados de libertad en la mayoría de los casos. Para que un robot pueda manipular un objeto, éste debe calcular la posición tridimensional del lugar donde se encuentra el objeto, la cinemática inversa y planificar movimientos para poder tomarlo. En esta sección se presentan algunos conceptos sobre planificación de movimientos de manipuladores en un robot humanoide. Se considera un robot humanoide para explicar los conceptos de manipulación ya que es fácil migrar estos conceptos a cualquier plataforma móvil.

2.4.1. Planificación de Movimientos de Manipuladores

En un robot humanoide se tienen cuatro cadenas de eslabones y articulaciones los cuales son dos brazos y dos piernas. Estas cadenas de eslabones son las extremidades del robot las cuales están unidas por medio del torso del humanoide. Para poder realizar trayectorias de movimiento en un robot humanoide primero que nada es necesario definir un marco de referencia global y una posición inicial. A partir del marco de referencia global se definen marcos de referencia local para cada extremidad. Por ejemplo, en la Figura 2.4 (a), se muestra un robot humanoide con su marco de referencia global establecido sobre la superficie que se encuentra entre ambas plantas de sus pies estando en una posición inicial. La posición definida a partir del marco de referencia global se denomina posición absoluta y se puede describir mediante el vector en (23).

$$P_h = \begin{bmatrix} P_{hx} \\ P_{hy} \\ P_{hz} \end{bmatrix} \tag{23}$$

Los marcos de referencia locales pueden estar establecidos ya sea en la primera articulación de una extremidad o en el mismo torso. Este marco de referencia se moverá con la extremidad. Por ejemplo, en la Figura 2.4 (b) [124], se establece el marco de referencia local de un brazo en la articulación del hombro.

Los vectores unitarios se definen para ser paralelos al marco de referencia global. La rotación que realice el brazo se verá reflejada mediante un ángulo. La relación entre los vectores unitarios, el marco de referencia local y el ángulo de rotación se muestran en la ecuación (24).

$$e_{ax} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix},\ e_{ay} = \begin{bmatrix} 0 \\ \cos\phi \\ \sin\phi \end{bmatrix},\ e_{az} = \begin{bmatrix} 0 \\ -\sin\phi \\ \cos\phi \end{bmatrix}, \tag{24}$$

Con el marco de referencia local del brazo definido es posible calcular la cinemática y dinámica del manipulador para definir trayectorias de movimiento. Según [118], la cinemática del manipulador trata del estudio analítico de la geometría de movimiento de un brazo con respecto a su marco de referencia como función del tiempo sin considerar fuerzas que lo causan. Considera dos problemas:

1. Cinemática directa, que determina la posición y orientación del efector final mediante los ángulos de las articulaciones.

2. Cinemática inversa, el cual considera la posición del efector final para calcular la posición de las articulaciones y eslabones del manipulador.

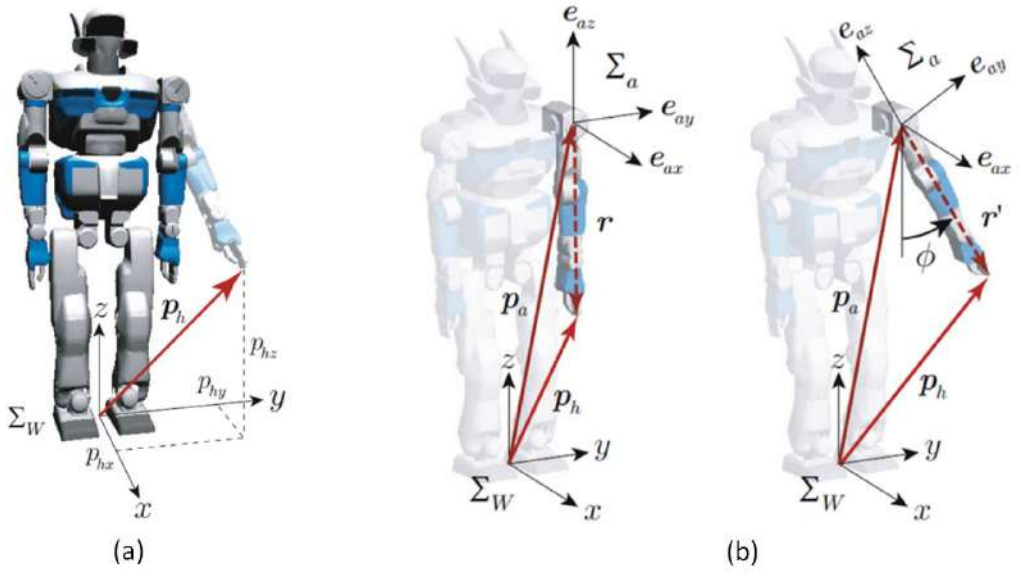


Figura 2.4: Posición del marco de referencia global y marco de referencia local para un brazo. (a) Marco de referencia global definido entre ambas plantas de sus pies, a su vez, estando en una posición inicial. Donde ph es la localización del final de la mano considerado desde el marco de referencia global, (b) Posición inicial del brazo y marco de referencia local del brazo rotado junto con el brazo (Imagen extraída de [124]).

Por otro lado la dinámica trata de la obtención de las ecuaciones matemáticas de movimiento del manipulador con el fin del desarrollo de sistemas de control. El propósito del desarrollo de estos sistemas es el de mantener la respuesta dinámica esperada del manipulador. Denavit y Hartenberg (D-H) [20], propusieron un método sistemático y generalizado de utilizar álgebra matricial para describir y representar la geometría espacial de los elementos de un brazo con respecto a un sistema de coordenadas de referencia fijo. La representación D-H tiene el objetivo de obtener la posición y orientación del extremo final del robot teniendo que tomar en cuenta únicamente 4 parámetros implícitos en la estructura del mismo. Este método facilita la obtención de matrices de transformación que son útiles para derivar las ecuaciones de movimiento dinámico del manipulador.

La planificación de las trayectorias del manipulador se refiere a un historial en el tiempo de la posición, la velocidad y la aceleración para cada grado de libertad. Para la planificación sólo es necesario especificar la posición y la orientación de destino deseadas del efector final. El sistema determina la forma exacta de la ruta para llegar ahí, la duración, el perfil de velocidad y otros detalles. La descripción de ruta requiere una secuencia de puntos vía deseados (puntos intermedios entre las posiciones inicial y final). Generalmente es conveniente que el movimiento del manipulador sea uniforme. La función uniforme es una función continua al igual que su primera derivada. Puede usarse cualquier función uniforme de tiempo que pase a través de los puntos vía para especificar la forma exacta de la ruta. Entre estas funciones se encuentran los polinomios cúbicos, polinomios de mayor orden y funciones lineales con mezclas parabólicas [20].

2.5. Navegación Espacial

Los robots de servicios deben tomar decisiones complejas, como identificar el medio con el que interactúa, detectar su objetivo (objetos) y cumplir órdenes (reconocimiento, manipulación) etc. Para la toma de esas decisiones es necesario que estos robots deban tener la capacidad de ser autónomos. Como se mencionó en la sección de *Introducción*, la autonomía se logra mediante un sistema de planificación de la actividad y de control para asegurar el logro de sus objetivos. Este sistema debe conseguir que los robots sean capaces de interactuar con el medio en el que se encuentran para tomar decisiones correctas y cumplir metas concretas. Una de las cualidades de estos sistemas es la navegación espacial, la cual permite calcular la pose (posición más orientación) de un robot en función de mediciones, ya sea, incrementales, inerciales y/o visuales. En esta sección se expone el concepto de odometría utilizado como parte del módulo de navegación espacial.

2.5.1. Odometría

La palabra odometría derivada de la palabra "odometró", se compone por las palabras griegas hodos ("viajar", "trayecto") y metron ("medida"), [125]. La odometría es el uso de datos de sensores de movimiento, para estimar los cambios en la posición del tiempo. Ésta es utilizada en la robótica móvil, para estimar su posición relativa a un punto de partida. Este método es sensible a los errores debidos a la integración de las mediciones de velocidad en el tiempo, para obtener estimaciones de posición, [126]. La posición obtenida se basa en el plano donde el robot está navegando, ésta se representa por el vector (x, y, θ).

La odometría se puede estimar mediante encoders, éstos son sensores basados en distintas tecnologías, [100]. Con éstos es posible medir las revoluciones de un eje. Es decir, un encoder dice cuantas vueltas ha dado un eje en un intervalo de tiempo dado. En un robot móvil, los encoders se suelen situar en los ejes de las ruedas, en cada uno de los grados de libertad de movimiento que pueda tener cada una de sus articulaciones.

Uno de los criterios con el que se puede calcular el desplazamiento se basa en el movimiento rectilíneo uniformemente acelerado. El movimiento rectilíneo uniformemente acelerado se da cuando la línea es recta y el vector de aceleración es constante e igual a la aceleración tangencial. Como el movimiento de un dispositivo llega a ser un ciclo, la aceleración en un punto comienza a ser constante. Se denomina movimiento rectilíneo uniformemente acelerado (m.r.u.a) al que tiene aceleración normal a cero y aceleración tangencial constante no nula. Al ser cero la aceleración normal, el radio de curvatura es infinito, por lo que el movimiento es rectilíneo, y además la aceleración tangencial es igual a la total. La velocidad 5.4 y las coordenadas de posición 5.5 vienen dados por:

$$v(t) = v_0 + at \tag{25}$$

$$s(t) = s_0 + v_0 t + \frac{at^2}{2} \tag{26}$$

Donde el tiempo será la variación $\triangle t$ que transcurrió al capturar los datos de aceleración.

La odometría inercial utiliza mediciones generadas por sensores inerciales como los giroscopios y/o los acelerómetros [100]. Los giroscopios son dispositivos que permiten medir y/o mantener la orientación, basados en los principios de la conservación de movimiento angular. Desde el giroscopio se permite el cálculo de la orientación y la rotación. Por ejemplo, en la

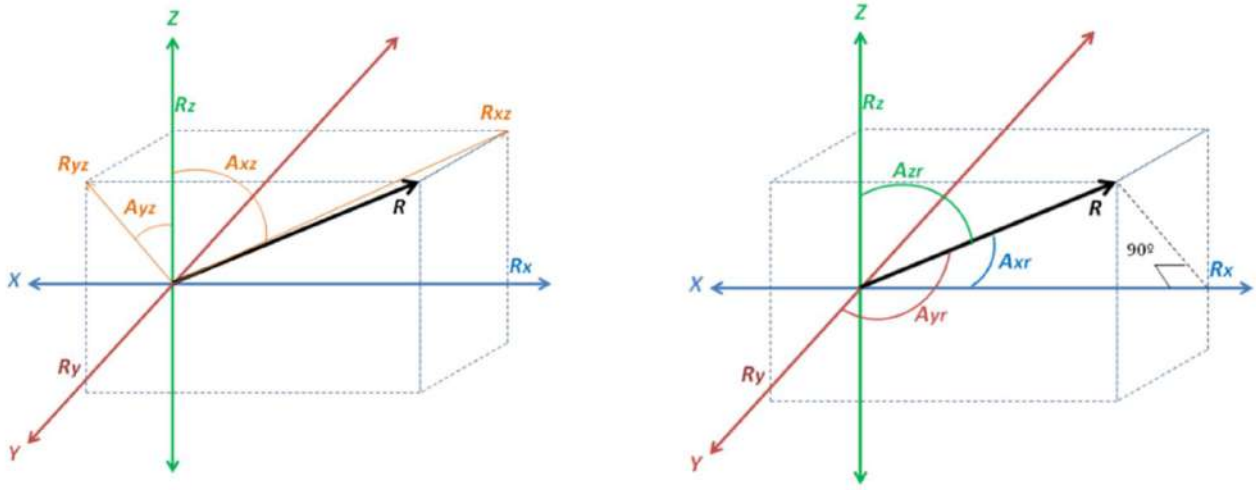


Figura 2.5: (a) Modelo del giroscopio con sistema coordenado de los ejes fijo. (b) Modelo del acelerómetro que muestra los ángulos que se deben encontrar para conocer la dirección del eje de inclinación.

Figura 2.5 (a) se muestra el modelo de un giroscopio de 2 ejes que medirá la rotación referente a los ejes X y al Y. R_{xz} será la proyección del vector fuerza inercial R en el plano XZ. Ryz será la proyección del vector fuerza inercial R en el plano YZ. De los triángulos formados por Rxz y Rz así como de Ryz y Rz usando el teorema de Pitágoras se obtiene lo siguiente:

$$R_{xz}^2 = R_x^2 + R_z^2 \tag{27}$$

$$R_{yz}^2 = R_y^2 + R_z^2 \tag{28}$$

Los ángulos entre el eje Z y los vectores R_{xz} y R_{yz} son definidos como A_{xz} (ángulo entre R_{xz} y el eje Z) y A_{yz} (ángulo entre R_{yz} y el eje Z). El giroscopio mide la tasa de cambio de los ángulos definidos anteriormente. Se obtendrá el valor relacionado linealmente con la velocidad de cambio de estos ángulos.

Mientras que los acelerómetros miden la aceleración lineal a lo largo de los tres ejes (x, y, z). Al medir la cantidad de aceleración de la gravedad, un acelerómetro puede averiguar el ángulo al que está inclinado en relación con la tierra. Mediante la detección de la cantidad de aceleración dinámica, el acelerómetro puede encontrar a qué velocidad y en qué dirección se está moviendo cualquier dispositivo. Por ejemplo, en la Figura 2.5 (b) se muestran que los ángulos que se deben encontrar son los comprendidos entre vector fuerza y los ejes X, Y, Z. Estos ángulos serán definidos como A_{xr}, A_{yr} y A_{zr}.

$$A_{xr} = \arccos(\frac{R_x}{R}) \tag{29}$$

$$A_{yr} = \arccos(\frac{R_y}{R}) \tag{30}$$

$$A_{zr} = \arccos(\frac{R_z}{R}) \tag{31}$$

De modo que la posición de un robot equipado con este tipo de sensores se obtiene mediante la adquisición de los datos: aceleraciones y cambios de orientación, y la integración de esta información incremental en el tiempo, permitiendo medir la distancia que se ha desplazado y la orientación que llevo.

El desplazamiento de un dispositivo consta de una serie de aceleraciones que dependen del tiempo en el que fueron capturadas. Estas aceleraciones se integran respecto al tiempo, para obtener velocidad instantánea (ecuación 32), y a su vez la velocidad se integra para obtener posición (ecuación 33) que basta con ir incrementando bajo cierto criterio para así obtener el desplazamiento.

$$v = \int adt \tag{32}$$

$$p = \int vdt \tag{33}$$

2.6. Next Best View (NBV)

Después de conocer el medio en el que se encuentra, el robot debe ser capaz de ubicar su objetivo. El objetivo en cuestión es el objeto que el usuario le requiere al robot. Entonces, la tarea a resolver es la búsqueda de objeto. La búsqueda de objetos requiere del módulo de reconocimiento de objetos para encontrar el objeto que el usuario está pidiendo. A su vez requiere que el robot esté capturando imágenes para evaluarlas. Por esta razón, se utilizará el enfoque *Next Best View* (NBV) [92]–[98] para apoyar a la planificación de los movimientos requeridos para que el robot encuentre un objeto y se aproxime a él para tomarlo.

Originalmente, el enfoque NBV se refiere a la siguiente mejor vista para el proceso de reconstrucción de un objeto 3D por parte de sistema automáticos. Tomando en cuenta un objeto desconocido, la planificación de todas las posiciones que debe alcanzar un sensor es imposible. En el contexto de robots móviles, el problema de NBV consiste en determinar la acción de sensado futura más favorable a ejecutar por el robot en un esfuerzo por alcanzar objetivos de tareas específicas. En otras palabras, mover al robot a una posición deseada para adquirir imágenes automáticamente.

Entre las aplicaciones que tiene NBV en robótica son la manipulación, estimación de pose, rastreo o *tracking*, reconocimiento y reconstrucción de objetos. Además, es utilizado para la detección de colisiones. El algoritmo, en general, busca tomar gran cantidad de escaneos de un objeto desde diferentes posiciones cuidando que siempre se tenga información nueva tomando en cuenta restricciones de posición tanto del robot como del sensor.

NBV tiene un objetivo particular, buscar la próxima mejor vista del objeto. Sin embargo, el método para determinar esta próxima mejor vista ha sido atacado de diferentes formas. Por ejemplo, en [96], se basan en la intensidad de las imágenes. La geometría de la escena es representada por un conjunto de primitivas de escala variada como superficies planas del tamaño de un pixel como distancia en la escena donde cada primitiva se parametriza por:

$$P_i = [X_i, \sum_i, S_{i,j}] : \{X_i \in R^3, \sum_i \in R^{3\times 3}, S_{i,j} \in R^{p\times p}\}, \tag{34}$$

donde X_i es la posición 3D primitiva, $\sum_i$ es la matriz de covarianza 3D y $S_{i,j}$ el conjunto cuadrado de $p \times p$ (p valor entero definido por el usuario) píxeles vecindario en la imagen de la proyección P_i en la imagen j. Además, las configuraciones de punto de vista se parametrizan en función de los ángulos de posición y orientación del sensor como,

$$v_j = [x_j, \theta_j] : \{x_j \in R^3, \theta_j \in SO(3)\}, \tag{35}$$

La estimación de estructura 3D y la incertidumbre de asociación geométrica de cada primitiva se realiza mediante un Filtro de Kalman Extendido (EKF) [46] individual. Entonces, para un punto de vista v y una primitiva 3D P, se utilizan las ecuaciones de colinealidad como función de observación no-lineal $\phi(X, v)$ y para el cálculo del efecto del sensado visual incremental para cada primitiva. El uso de EKF es motivado por la naturaleza no incremental de las estimaciones de incertidumbre.

Por otro lado en [95], utilizando un escáner láser se construye una malla triangular con puntos 3D que definen el objeto. Esta malla consiste en vértices v_i con posición p_i, normal n_i, bordes e_i con vértice inicial v_i y final v_j, dos vértices adicionales v_l y v_r, cerca del triángulo adyacente de izquierda y derecha.

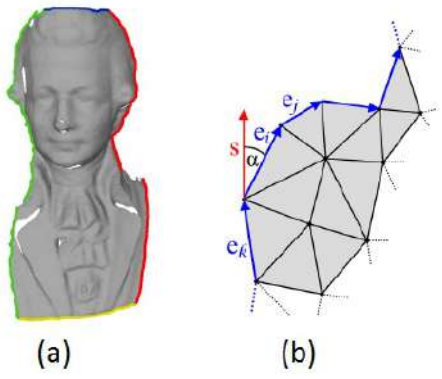


Figura 2.6: La clasificación de bordes de un busto de mozart, (a) bordes: izquierdo (verde), derecho (rojo), arriba (azul), abajo (amarillo), (b) clasificación de borde izquierdo (Extraído de [95]).

Después del escaneo del objeto, se efectúa un estimador de punto de vista que consiste en clasificación de bordes, estimación de las curvas de los bordes y determinación del punto de vista. El clasificador de bordes detecta y clasifica todos los bordes del objeto basado en la

información de superficie. La malla global del objeto se transforma en sistemas coordenados del sensor para determinar los bordes derecho, izquierdo, arriba y abajo de los puntos de vista del sensor. Por ejemplo, en la Figura 2.7 (a) se muestra la clasificación de los bordes de un busto de Mozart. En la Figura 2.7 (b) se muestra la clasificación del borde izquierdo, la cual se realiza iterando en todos los bordes e_i de la malla. El ángulo α entre la dirección escaneada s y el borde actual e_i se calcula cómo:

$$\alpha = arccos(\frac{s \cdot e_i}{|s||e_i|}) \tag{36}$$

Debiendo ser α mayor a cierto umbral para seguir siendo considerando parte de ese borde. Una vez detectados y clasificados los bordes, la estimación de las curvas de los bordes ejecuta un inicio de crecimiento de región desde los diferentes bordes para encontrar vértices y ajustarlos a parches cuadrados. Se utiliza un enfoque simple para ajustar los vértices v_i usando solo la posición $p_i = [x_{p_i}, y_{p_i}, z_{p_i}]$ de un parche cuadrático,

$$z_{p_i} = f(x_{p_i}, y_{p_i}) = ax_{p_i}^2 + bx_{p_i}y_{p_i} + cy_{p_i}^2 + dx_{p_i} + ey_{p_i} + f \tag{37}$$

Durante la determinación del punto de vista, las vistas del sensor se determinan usando una curva estimada con la restricción de que el sensor los observe perpendicular a cierta distancia d de la superficie estimada y exista sobre posición con el escaneo anterior, (ver Figura).

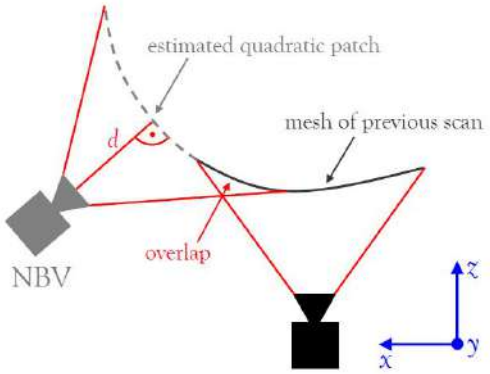


Figura 2.7: Campo de vista traslapado con la malla desde el escaneo anterior y el sensor visto en perpendicular al parche cuadrado estimado a cierta distancia d (Extraído de [95]).

Iniciando desde los bordes detectados, se determinan candidatos de puntos de vista calculando puntos y normales del parche cuadrado estimado. Desde un punto a lo largo del borde se calculan posibles puntos de la superficie p_i en la dirección del área desconocida insertando x_{p_i} y y_{p_i} en (37). Para el borde izquierdo y_{p_i} se considera constante y se incrementa x_{p_i}. Luego la normal n_i de este punto en la superficie se calcula de las derivadas de (37):

$$n_i = \begin{pmatrix} x_{p_i} \\ y_{p_i} \\ z_{p_i} \end{pmatrix} = \begin{pmatrix} \frac{\partial f}{\partial x_{p_i}} \\ \frac{\partial f}{\partial y_{p_i}} \\ -1 \end{pmatrix} = \begin{pmatrix} 2ax_{p_i} + by_{p_i} + d \\ bx_{p_i} + 2cy_{p_i} + e \\ -1 \end{pmatrix} \tag{38}$$

donde z_{p_i} se establece como -1 ya que la dirección de la vista del escáner se describe por el eje z positivo y los puntos en la superficie están en dirección opuesta. Para este punto en la

superficie una vista candidata se calcula a cierta distancia d desde la curva y en dirección a la normal.

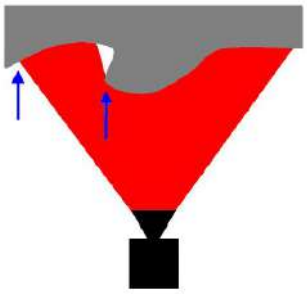


Figura 2.8: Objeto con oclusión: dos bordes izquierdos se detectan. (Extraído de [95]).

Después, se generan y almacenan varios puntos de vista para posteriormente ejecutar la selección NBV. En la Figura 2.8, se muestra un ejemplo donde dos bordes izquierdos se han detectado debido a la oclusión. Esta se realiza simplemente eliminando la oclusión de los puntos de vista candidatos y se selecciona al candidato que se encuentre más a la derecha del borde izquierdo.

En el trabajo [98], para la reconstrucción de un objeto 3D, se utiliza un robot móvil que cuenta con un brazo manipulador de 8 grados de libertad (GDL). Este brazo a su vez tiene un sensor *eye in hand* u ojo en mano, que se refiere a un sensor al final del brazo manipulador en vez de contar con un efector final. NBV no solo es utilizado para la reconstrucción y obtención de pose (posición y orientación), si no también para calcular la trayectoria del robot.

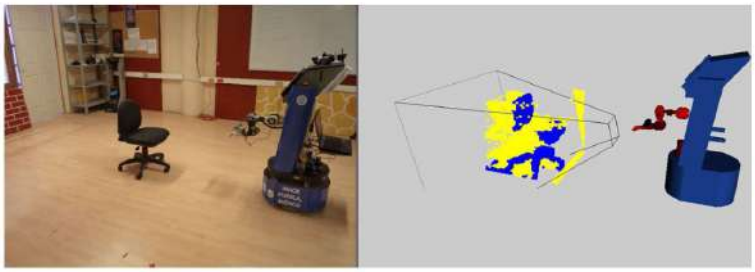


Figura 2.9: Robot móvil manipulador de 8 GDL para reconstruir objetos con un sensor kinect como efector final, modelo reconstruido de la escena parcial, los *voxels* se muestran en amarillo los ocupados en azul, (Extraído de [98]).

El proceso en el que se basa el trabajo se divide en dos tareas principales: reconstrucción 3D y planificación de estado mediante NBV. La reconstrucción 3D consiste en integrar una serie de escaneos del objeto tomadas desde diferentes posiciones de sensado. La primer vista o estado se establece arbitrariamente, y los siguientes se determinan mediante el algoritmo de planificación NBV. El robot corrige su pose después de cada escaneo usando el pareo entre la nube de puntos

reconstruido y la nueva superficie escaneada. Antes del primer movimiento del robot, se asume que esas posiciones y orientaciones de la base móvil y el brazo son conocidas con precisión con respecto a un marco de referencia definido por la *bounding box* o caja contenedora del objeto.

Después de cada escaneo, las lecturas del sensor se integran en un octaedro que representa el *boundig box* del objeto. La representación de la *bounding box* W_{box}, utiliza un mapa de ocupación probabilística basado en una estructura *octomap*. Después de cada escaneo, las lecturas del sensor se integran en un octaedro que representa el *boundig box* del objeto. Este octaedro se compone de *voxels*, de éstos los ocupados que representan superficie de objetos se integran con la superficie del modelo actual mediante un proceso de registro llevado a cabo por el traslape de las vistas previas. La fase NBV inicia generando un conjunto de puntos de vistas o estados en el espacio de configuraciones clasificados de acuerdo a una función de utilidad.

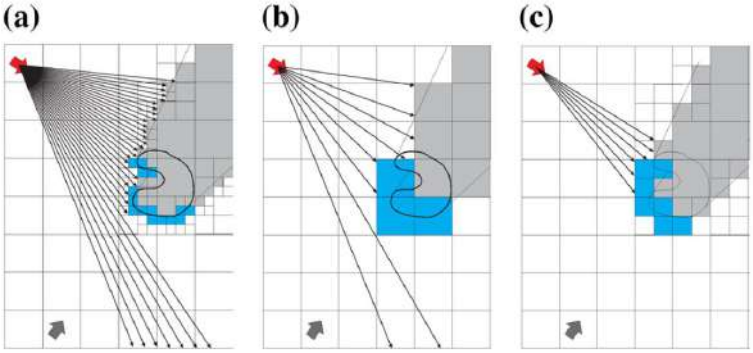


Figura 2.10: Ejemplo de trazo uniforme de rayos o líneas y trazo jerárquico. (a) los rayos se trazan en el octaedro, (b) rayos trazados en un octaedro grueso, (c) rayos trazados en un octaedro más fino, (Extraído de [98]).

En la Figura 2.9 se observa la representación donde a cada *voxel* se le asocia una probabilidad de ser ocupado. Estos *voxels* puede tener tres clasificaciones con su probabilidad: ocupado (0,55), libre ($<$ 0,45) o desconocido (0,45a0,55). La función de utilidad toma en cuenta cuatro factores: posición, registro, superficie y distancia. La posición permite eliminar a los candidatos que colisionan. Solo los candidatos que garanticen un traslape mínimo con previas vistas se mantienen y registran.

Lo siguiente es evaluar el monto de *voxels* desconocidos observados, este factor se normaliza por el total de números de *voxels* desconocidos restantes. En la Figura 2.10 (a) se muestra un trazo uniforme de líneas o rayos dentro del mapa simulando el rango de un sensor. (b) y (c) representan trazos más finos hacia los *voxels*. Los estados candidatos son evaluados de acuerdo a su distancia al estado actual del robot considerando el camino seguido por el robot. Finalmente, los mejores candidatos de acuerdo a lo obtenido con la función de utilidad se selecciona como NBV. El ciclo de reconstrucción 3D se repite hasta que el factor superficie sea mínimo a un umbral o no se encuentre nuevo camino.

Capítulo 3

Módulo de Reconocimiento de Objetos

En un entorno humano como lo puede ser una oficina, una habitación o cualquier otro espacio donde el hombre realiza sus actividades diarias, se encuentran los más variados objetos. Sin embargo, no todos los objetos cuentan con una forma definida tal como la poseen los objetos poliédricos (esferas, prismas, etc). A los objetos poliédricos es relativamente sencillo describirlos mediante su ecuación característica y con ello un sistema robótico puede fácilmente aprender a diferenciarlos. Por el contrario, aquellos objetos comunes para el ser humano que no tienen una forma definida, son difíciles de modelar matemáticamente. Estos objetos son llamados objetos de forma libre debido a que pueden tener formas variadas lejos de la sencillez de los objetos poliédricos. Ejemplos de este tipo de objetos se pueden encontrar con tan solo ver un poco alrededor. En una oficina, por ejemplo, hay engrapadoras, tijeras, clips, teléfono, etc. Así que definir una metodología para describir este tipo de objetos no es tarea sencilla sobre todo si se considera que son objetos tridimensionales, lo cual añade un grado de complejidad extra a los métodos que se encuentran en la literatura sobre el reconocimiento de objetos, que usualmente usan información fotométrica para poder diferenciarlos unos de otros.
Entonces, si el objetivo es reconocer ese tipo de objetos conocidos como objetos de forma libre, la mejor solución es el uso de un método de reconocimiento de objetos que considere la textura de estos. Esta textura representará aquellas marcas que los diferencian unos de otros.

Por lo anterior, se plantea el desarrollo de un módulo de reconocimiento de objetos de forma libre, tal cual se describe en la Figura 3.1. Este módulo se divide en dos fases: *Aprendizaje* y *Reconocimiento*. En la fase de *Aprendizaje*, se considera como entrada diferentes vistas o capturas de un objeto. De estas vistas se extraen características mediante una técnica que detecta visualmente puntos distintivos en una imagen llamados puntos de interés, puntos salientes, puntos esquinas, *keypoints* o puntos claves. La detección de estos puntos incluyen su escala, orientación y descripción haciéndolos una fuerte característica para aprender y reconocer un objeto. El algoritmo de detección y descripción de características multiescala A-KAZE [81] es el utilizado para realizar esta tarea. Estas características representan el modelo del objeto el cual pasa a la etapa de entrenamiento. La etapa de entrenamiento es llevada a cabo mediante una red neuronal no supervisada. Se utiliza la variante de los mapas auto-organizados de Kohonen: *Growing Cell Structure* (GCS) [76]. Esta red se encarga de agrupar las características similares en clases o neuronas más cercanas. Esta red cuenta con la ventaja de incrementar y decrementar dinámicamente su forma durante la fase de entrenamiento. Cada objeto es representado por las neuronas asignadas en el entrenamiento. Después de la etapa de entrenamiento, cada grupo de

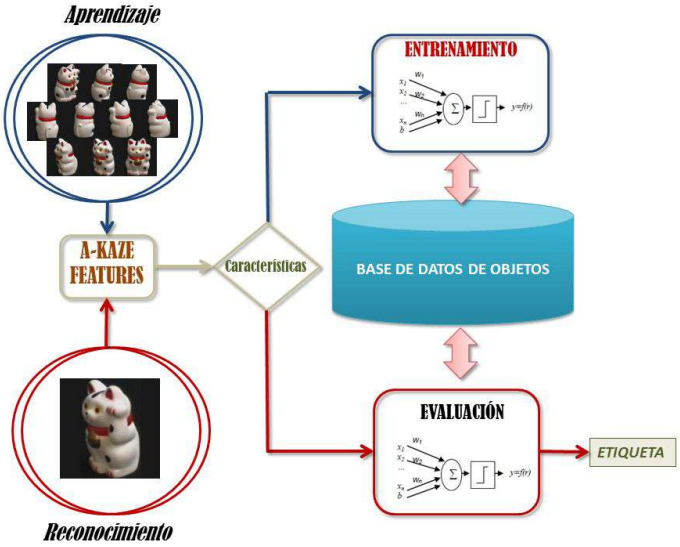


Figura 3.1: Módulo de Reconocimiento de Objetos.

neuronas asignadas es identificado por una etiqueta con el nombre del objeto que representan y se reserva en la base de datos. En la fase de *Reconocimiento*, se considera como entrada una sola vista del objeto. De esta se extraen características mediante el algoritmo A-KAZE. Las características extraídas son enviadas a la etapa de evaluación. La etapa de evaluación considera la red neuronal entrenada para clasificar los patrones de información reconocidos. Se considera la base de datos de objetos aprendidos para la clasificación. Del grupo de neuronas al que pertenezca el reconocimiento se recupera la etiqueta. Esta etiqueta es el objeto nombre del objeto reconocido.

En la sección 3.1, se describe la implementación del método de extracción de características de superficie local A-KAZE, presentando la información extraída de las imágenes de los objetos que se quieren reconocer. La implementación de la red GCS, utilizada para la asociación de los datos extraídos por el método A-KAZE se describe en la sección 3.2. En la sección 3.3, se desarrolla el algoritmo de clasificación y evaluación del módulo de reconocimiento de objetos. Aquí, se detalla el procesamiento de la información para cada una de las tareas del módulo y su construcción en seudo código. Posteriormente, la sección de experimentos y resultados muestra la implementación del módulo para validar su ejecución. Por último, se incluye una sección de conclusiones del capítulo.

3.1. Implementación de A-KAZE

El módulo de reconocimiento de objetos utiliza el método de extracción de características de superficie local A-KAZE. La elección de este método se debe a diversas razones. En la sección de *Marco Teórico*, se mencionó su excelente compromiso entre rapidez y ejecución comparados a BRISK, ORB, SURF, SIFT y KAZE. Al ser un método relativamente nuevo, éste no ha sido explotado lo suficiente para demostrar su capacidad de ejecución. Ya que el objetivo del módulo de reconocimiento de objetos es el de aprender objetos de forma libre, el descriptor A-KAZE per-

mitirá extraer aquellas características distintivas que los diferencian unos de otros. Además, este método toma en cuenta los bordes de los objetos, punto importante para estos objetos de forma libre que son difíciles de modelar. Sin olvidar que este extractor es invariante a rotación y escala.

Estas últimas características, permiten que el entrenamiento del módulo de reconocimiento de objetos sea más sencillo. La invarianza a rotación y escala es necesaria debido a que, en las imágenes capturadas de un objeto real, los objetos pueden tener diferente tamaño, posición y orientación, como se observa en la Figura 3.2. Esta ventaja permite que, en la etapa de aprendizaje, A-KAZE extraiga *keypoints* y descriptores de cada una de las imágenes que ingresen al módulo independientemente su orientación y posición. Posteriormente, los datos extraídos de las imágenes son procesados para enviarlos al entrenamiento de la red neuronal no supervisada con el fin de asociar la información más cercana.

Figura 3.2: Imágenes de prueba de un objeto real para la implementación de A-KAZE.

La implementación del método A-KAZE se lleva a cabo mediante la clase *AKAZE* de OpenCV 3 [102]. Según esta clase, es necesario especificar algunos parámetros como: tipo, tamaño y número de canales de descriptor. La información que entrega la clase *AKAZE* son: posición del punto (x,y), diámetro del vecindario, orientación del *keypoint* [0-360], índice de la octava a la que corresponde y descriptores para cada *keypoint*.

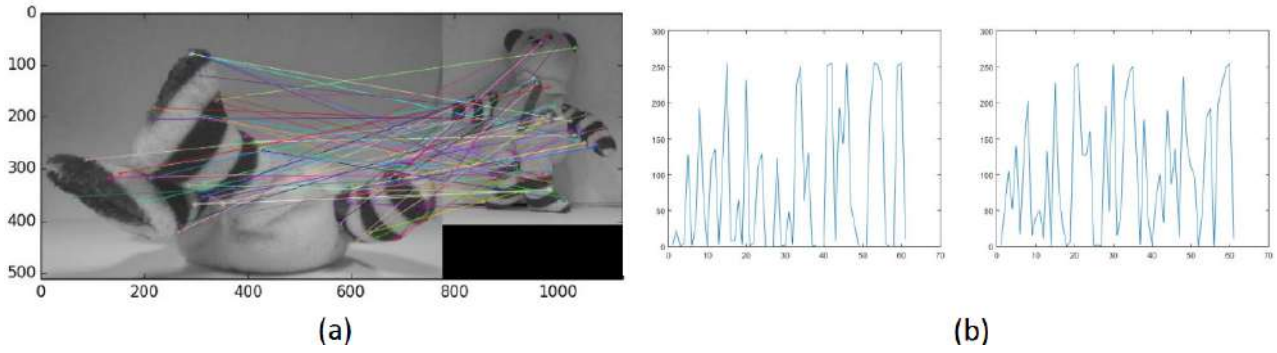


Figura 3.3: Dos imágenes de prueba del objeto real para la implementación de la clase *AKAZE*. (a) Correspondencia entre *keypoints* de ambas imágenes. (b) Representación de dos de los *keypoints* más cercanos entre sí como histogramas.

Visualmente se puede observar la información entregada de la implementación de A-KAZE como puntos de interés resaltados, si se utiliza la correspondencia en una imagen y otra, se pueden distinguir donde se encuentran los descriptores más parecidos. En la Figura 3.3 (a) se presentan dos imágenes de un objeto. Para estas imágenes se obtuvieron los siguientes resultados: de la imagen en la izquierda se obtuvo un total de 144 *keypoints*, la imagen de la derecha

obtuvo 128 *keypoints*. La mayoría de los *keypoints* encontrados tuvieron coincidencia con la otra imagen. Sin embargo, sólo se muestran gráficamente los 50 *keypoints* más parecidos. En la Figura 3.3 (b) se muestra dos de los *keypoints* más cercanos entre sí. Estos son descriptores representados por histogramas de intensidad de gradientes alrededor del *keypoint* con tamaño de 61 valores de 0 a 255.

Estos descriptores son utilizados en muchas aplicaciones para encontrar paridad entre imágenes o buscar un objeto en una escena. Este módulo, por su parte, utiliza la información de los descriptores de los objetos a reconocer para almacenar un modelo tridimensional de estos. El modelo se forma utilizando la gran cantidad de *keypoints* obtenidos por A-KAZE, considerando que las vistas de un objeto comparten gran cantidad de información entre sí. En la Figura 3.4, se observan cuatro imágenes de una botella de corrector líquido.

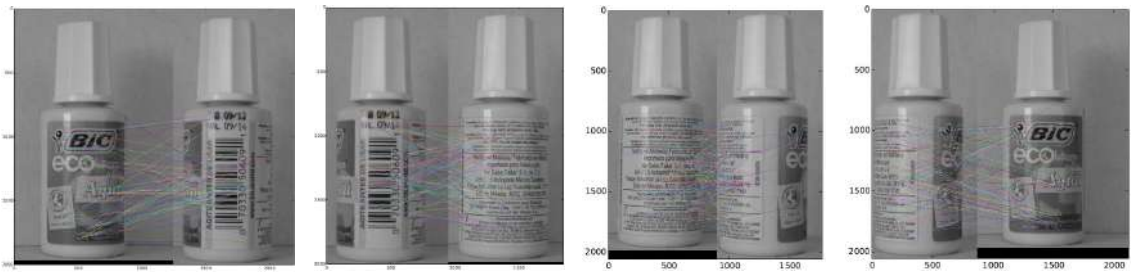


Figura 3.4: Se muestran cuatro capturas de un corrector líquido: de frente, atrás, izquierda y derecha. Las capturas se comparan y se observa gráficamente las 50 mejores coincidencias unidas por líneas.

A este objeto se le han tomado cuatro capturas: de frente, atrás, izquierda y derecha. A cada captura se le han extraído todos los *keypoints*. Enseguida, se compara la captura de frente contra la de la izquierda, izquierda con atrás, atrás con derecha y derecha con frente. Entre cada captura se obtuvo una gran cantidad de coincidencias, algunas de ellas erróneas. Estas coincidencias erróneas disminuirán si se toman más capturas entre cada una de ellas. Sin embargo, el número de descriptores (histogramas representados por vectores de 61 valores) será muy grande. Dando como resultado una gran cantidad de datos a almacenar y clasificar. Además, muchos de estos datos, entre capturas, son muy similares tanto en cantidad como en descripción. Considerando una *bag of words* o bolsa de palabras poco convencional.

Para disminuir la cantidad de datos a clasificar se decide construir un solo histograma que unifique a todos aquellos *keypoints* encontrados en una sola imagen. Esta idea da a entender que mucha de la información de esta imagen se perderá. Sin embargo, lo que el módulo de reconocimiento de objetos aprende son varias imágenes de un mismo objeto. Si estas imágenes comparten muchas características en común, entonces, tendrán una cantidad de *keypoints* muy aproximada, los descriptores serán muy similares y su histograma descriptor será muy cercano.

La construcción del histograma de *keypoints*, se realiza con la ecuación (1). Éste es el promedio de la sumatoria de todos los descriptores entre el número total de *keypoints* de una sola imagen.

$$Histograma(vista) = \frac{\sum Descriptores}{TotalKeypoints} \tag{1}$$

Para demostrar que este método es suficiente para describir la vista de un objeto, se construyen los histogramas de las cuatro imágenes del corrector líquido presentados en la Figura 3.4. Los cuatro histogramas se sobreponen en la Figura 3.5 (a) para mostrar que estos cuentan con gran similitud entre ellos. Para llevar a cabo la comparación de una vista a otra, se calculó la distancia Euclidiana entre cada uno de los valores y se construyó la gráfica de la Figura 3.5 (b). En esta gráfica se sobreponen de igual forma los resultados de las comparaciones. Se observa que los valores obtenidos están por debajo de 30 que es un 15 % del valor máximo que pueden llegar a representar.

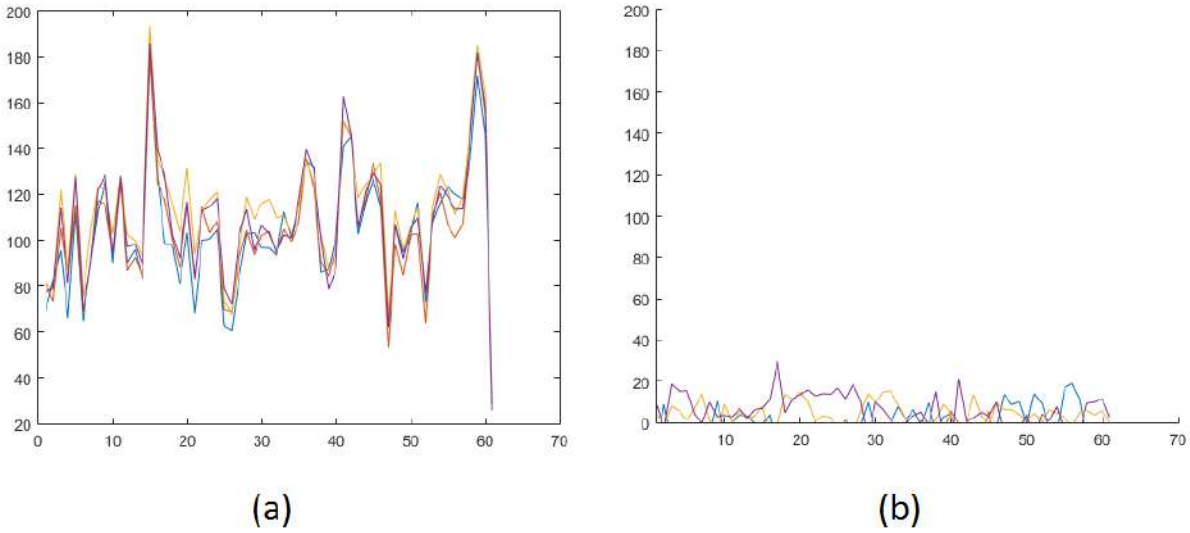


Figura 3.5: (a) Histogramas por vista, (b) Error entre vistas.

Para reforzar que este método es suficiente, ahora se compara la vista de un objeto con la de otro. En la Figura 3.6, se observan los dos objetos a comparar, la vista de frente del corrector líquido y la vista de frente de una caja de regalo con un moño rojo. De ambos se construyen los histogramas y se sobreponen en la gráfica de la Figura 3.7 (a). En esta superposición se observa que estos histogramas difieren en mayor proporción. De igual manera se obtuvo la distancia Euclidiana entre cada uno de los valores de los histogramas y se representa gráficamente en la Figura 3.7 (b). En esta gráfica se observa que el error entre ellos es mayor. Muchos de los valores obtenidos suben hasta 60 que es un 30 % del valor máximo que pueden llegar a representar.

Claro que el análisis anterior es mas cualitativo que cuantitativo, así que de estas comparaciones se obtiene también el error cuadrático medio. Este error se calcula mediante la ecuación (2),

$$\varepsilon = \frac{1}{n}\Sigma_{i=1}^{n}(x_i - y_i)^2 \tag{2}$$

donde x_i y y_i son los valores de los histogramas. En este, se obtiene el promedio de la sumatoria de todas las distancias Euclidianas de los valores en los histogramas. Los resultados

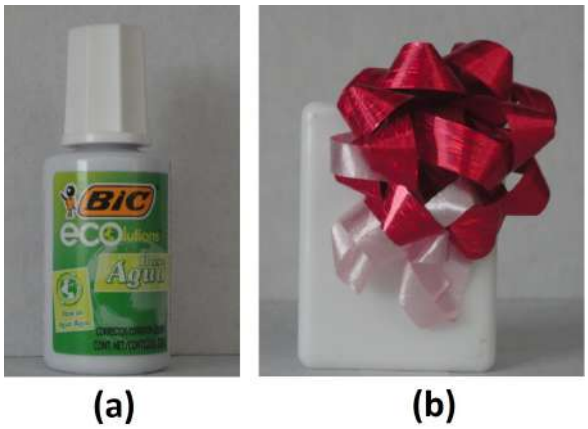


Figura 3.6: Objetos a comparar (a) Objeto1, corrector líquido, (b) Objeto2, regalo.

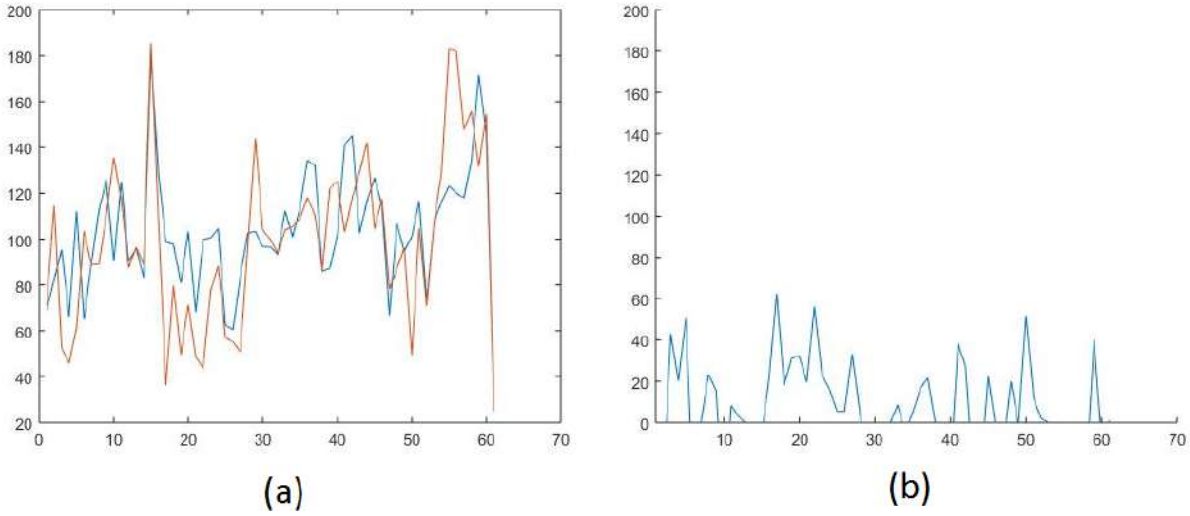


Figura 3.7: (a) Dos histogramas de objetos diferente, (b) Gráfica de error.

se muestra en la Tabla 3.1. La evaluación de cada una de las comparaciones del corrector líquido entregan un error menor a 200, mientras que la comparación entre objetos diferentes arroja un error mayor a 1000.

La representación de los histogramas es suficiente cuando las imágenes para el entrenamiento de los objetos muestran características similares. En ocasiones con solo cuatro capturas es suficiente para poder aprender un objeto. Pero que pasa cuando un objeto difiere en gran medida de una vista a otra. Los objetos de forma libre pueden ser de cierta forma por enfrente y cuando se les da la vuelta son completamente diferentes por atrás. Simplemente, a este objeto lo representarán dos histogramas o más.

Tabla 3.1: Error cuadrático medio entre histogramas.

Imágenes	ε
Frente-Izquierda	123.17
Izquierda-Atrás	93.34
Atrás-Derecha	115.84
Derecha-Frente	161.25
Objeto1-Objeto2	1068

3.2. Implementación de GCS

Una vez obtenidos los histogramas de cada objeto a reconocer, estos serán la entrada a la red neuronal no supervisada GCS. Se ha elegido esta red neuronal para que realice las funciones de algoritmo clasificador. Esta elección se decidió ya que estas redes funcionan como asociadores de datos. La red neuronal automáticamente asocia a los vecinos más cercanos, si la información de las imágenes del objeto son muy parecidas éstas estarán clasificadas en la misma neurona.

En el módulo de reconocimiento de objetos el número de objetos a aprender e imágenes para la construcción de la base de datos pueden ser tantos como el usuario desee. Si el usuario decide tomar 100 imágenes de cada objeto, la base de datos será de gran tamaño. Los histogramas de cada imagen reducirán los datos, sin embargo, muchas de las imágenes compartirán información. Entonces, a manera de homogeneizar la información, la red neuronal no supervisada automáticamente agrupará esos datos. Por ejemplo, las 20 imágenes del corrector líquido que se muestra en la Figura 3.8, comparten muchas características. Estas imágenes son de diferentes tamaños, posiciones y vistas, con el fin de cubrir toda su forma. Los histogramas obtenidos de estas imágenes serán muy cercanos. La red neuronal GCS permitirá que en vez de tener 20 histogramas para definir a ese objeto, se tendrán solo unas cuantas neuronas.

Figura 3.8: 20 imágenes capturadas para el entrenamiento del corrector líquido.

Otra de las razones por la que se elige GCS no supervisada para la tarea de clasificación es la ventaja que tiene de incrementar y decrementar dinámicamente su forma durante la fase de entrenamiento. En esta fase, GCS requiere que se especifiquen ciertos parámetros como: (i) número máximo de neuronas, el cual se refiere al tamaño máximo que tendrá la red neuronal, (ii) número de épocas o pasos adaptación λ, que es el número de repeticiones que se realizarán para la distribución de datos antes de agregar otra neurona a la red, (iii) constante de adaptación de la neurona ganador ε_b, (iv) constante de adaptación del vecindario de la neurona ganador ε_n, (v) decremento α y (vi) umbral de eliminación θ.

La red entrenada tendrá como resultado la asociación de los histogramas en ciertas neuronas por objeto. Si se quieren aprender 10 objetos, lo ideal sería tener 10 neuronas máximo en la red. Sin embargo, como se mencionó anteriormente, los histogramas de un objeto pueden diferir entre ellos. Así que, es lógico pensar que un objeto tendrá una o más neuronas para su clasificación. El número máximo de neuronas de la cual estará conformada la red permite elegir que tan precisa se quiere la distribución de los histogramas. Por ejemplo, las 20 imágenes de la base de datos del corrector líquido mostradas en esta sección se entrenaron mediante el módulo de reconocimiento de objetos junto a otros 10 objetos con 20 imágenes cada uno. Entonces la red debe clasificar 200 histogramas, así que a manera de visualización, se entrenaron 2 redes neuronales, con 10 y 100 como números máximos de neuronas en cada red.

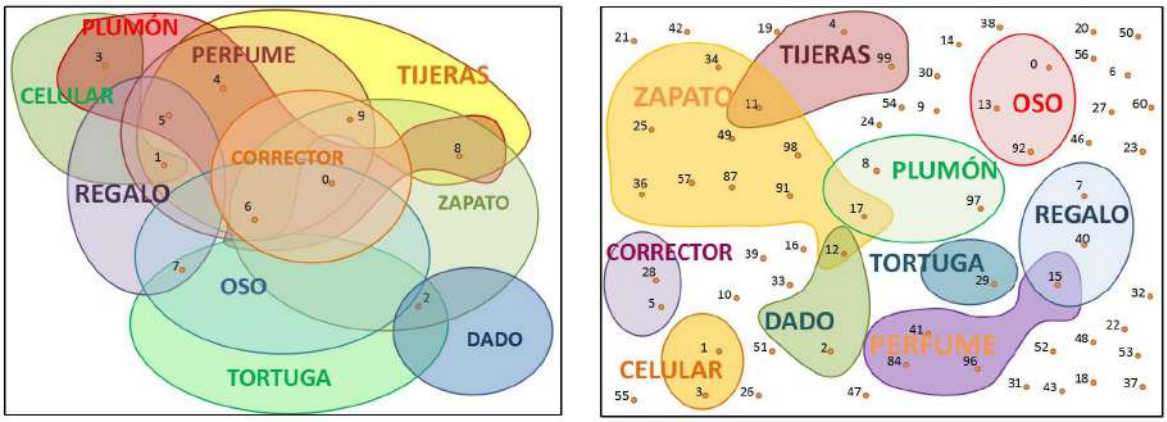


Figura 3.9: Ejemplificación de visualización gráfica de la distribución de los histogramas en el entrenamiento de la red neuronal, (1) 10 neuronas, (2) 100 neuronas.

El resultado del entrenamiento de las dos redes neuronales se muestra de manera gráfica en la Figura 3.9. En éstas se observa la distribución de los histogramas representada como regiones de diferente color para cada objeto. En la Figura 3.9 (1) se muestra la red entrenada con 10 neuronas. En este entrenamiento la discriminación de los objetos tuvo un resultado de 25 % ya que varios de los histogramas fueron erróneamente clasificados. En la Figura 3.9 (2) se muestra la red entrenada con 100 neuronas. Para ésta la distribución de los objetos es mejor teniendo como porcentaje un 98.66 % de discriminación. De la distribución obtenida en la red neuronal de 100 se visualiza que los histogramas de los objetos tienden a agruparse en un número menor de neuronas si estos comparten características similares. Si no comparten características entonces

los histogramas estarán clasificados en más de una neurona como es el caso del objeto zapato. Para este objeto la red neuronal clasificó sus histogramas en 11 neuronas, confundiendo algunos con los de otros objetos. Para obtener una discriminación con un porcentaje de 100 % solo basta con aumentar el número máximo de neuronas.

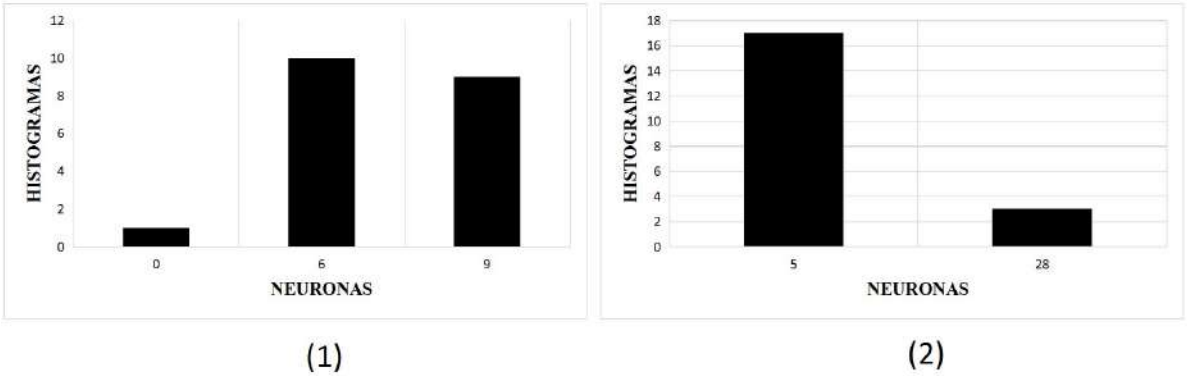


Figura 3.10: Gráfica de barras del conteo de histogramas por neurona del corrector líquido en el entrenamiento con: (1) 10 neuronas, (2) 100 neuronas.

Entonces, los histogramas del corrector líquido quedaron clasificado en 3 neuronas de la red de 10 y en 2 neuronas de la red de 100. Para simplificar los resultados se construyeron las gráficas de barra de la Figura 3.10 (1) y (2). En éstas se representa el conteo de número de histogramas por neurona para el objeto corrector líquido de las redes de 10 y 100 neuronas. En la red neuronal de 10 el corrector se clasificó en las neuronas 0, 6 y 9 con un número respectivo de 1, 10 y 9 histogramas. En la red de 100 se clasificó en la 5 y 28, con un número respectivo de 17 y 3 histogramas. Estos resultados demuestran que efectivamente varios de los histogramas tenían características en común y es por eso que se han clasificado en un número pequeño de neuronas.

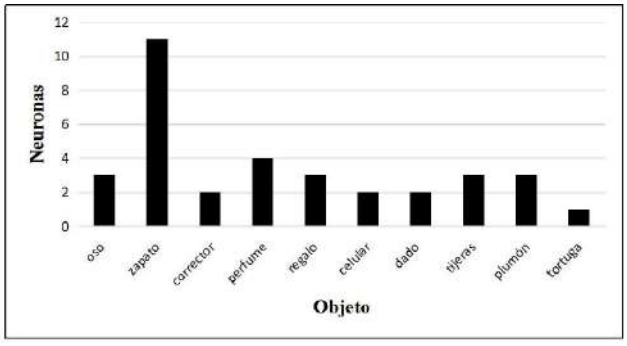


Figura 3.11: Gráfica de barras del conteo de histogramas por neurona de 10 objetos con 100 neuronas.

A manera de simplificación, en la Figura 3.11 se observa una gráfica de barras del conteo de cada objeto por neuronas asociadas a él de la red entrenada con 100 neuronas. Se observa que de las 100 neuronas indicadas solo 34 de ellas han sido utilizadas para clasificar los 200 histogramas de los 10 objetos. La red entrenada se almacena identificando las neuronas que

pertenecen a cada objeto. De esta manera, para la evaluación del histograma de una nueva imagen de algún objeto entrenado, bastará con utilizar la estrategia del vencedor gana todo. El resultado entregará la neurona a la que pertenece el objeto que se le ha pedido identificar al sistema.

Este método cuenta con diversas ventajas, como la asociación automática de información, requiere poco espacio de almacenamiento, es fácil de ser implementado, es rápido y tiene buena discriminación entre objetos en la etapa de entrenamiento. La implementación de GCS y A-KAZE en el módulo de reconocimiento de objetos se describe en la siguiente sección.

3.3. Algoritmo de clasificación y evaluación

A continuación se resumen las principales tareas realizadas por el módulo para la clasificación y evaluación de diferentes objetos:

Clasificación:

1. Para cierto número de objetos obtener imágenes de diferentes perspectivas. Es necesario trabajar con el mismo número de imágenes por objeto.

2. Extraer todos los *keypoints* por cada imagen.

3. Construir el histograma para cada imagen obteniendo el promedio de todos los *keypoints*. Se hace esto para cada imagen.

4. Ingresar como entrada a la red neuronal los histogramas obtenidos. La red neuronal se encargará de clusterizar la información.

5. Al finalizar la clusterización, se identifican las neuronas pertenecientes a cada objeto.

Evaluación:

1. Obtener una o más imágenes de uno o más objetos.

2. Extraer todos los *keypoints* de cada imagen.

3. Construir el histograma obteniendo el promedio de todos los descriptores de los *keypoints* por imagen. Hacer esto para cada imagen.

4. Evaluar cada histograma con la red neuronal entrenada para obtener las neuronas mas cercanas.

5. Identificar a que objeto pertenece las neuronas más cercanas.

3.3.1. Módulo de reconocimiento de objetos

En base a lo descrito en la sección anterior se construye el módulo de reconocimiento de objetos. El módulo se divide en dos tareas: *Entrenamiento* y *Reconocimiento*. En la tarea de *Entrenamiento*, como lo describe el Algoritmo 4, se recibe como entrada el número de objetos y el número de imágenes por objeto, así como las imágenes de las diferentes vistas de este. Utilizando A-KAZE se extraen los *keypoints* y se construyen los histogramas de descriptores por cada imagen. Todos los histogramas se envían a la red neuronal no supervisada GCS la cual cumple la función de clasificación. Le red neuronal obtenida se reserva para la etapa de evaluación.

Algoritmo 4: Módulo de reconocimiento de objetos. *Entrenamiento*

Datos: $\boldsymbol{I}$ imágenes, $\boldsymbol{N}$ objetos, $\boldsymbol{L}$ imágenes por objeto.
Resultado: ***classes(E)*** Clases por etiqueta.

```
1 para n ← 1 to N hacer
2     para l ← 1 to L hacer
3         keypoints =A-KAZE(I(n,l))
4         H(n,l)=Build-Histos(keypoint)
5 ANN_trained=GCS(H)
```

La tarea de *Reconocimiento*, descrita en el Algoritmo 5, usa solamente una imagen del objeto a reconocer como entrada. En esta tarea, como en la de *Entrenamiento*, se utiliza A-KAZE para extraer los *keypoints* y se construye el histograma de la imagen. Después, este histograma se envía a la evaluación utilizando la red neuronal ya entrenada. La red neuronal entregará la neurona a la cual pertenece este histograma. La neurona que resulte ganadora se utilizará para identificar a cual objeto pertenece.

Algoritmo 5: Módulo de reconocimiento de objetos. *Reconocimiento*

Datos: $\boldsymbol{I}$ image
Resultado: ***class(I)*** Object class.

```
1 keypoints =A-KAZE(I)
2 H(I)=Build-Histos(keypoint)
3 neuron=ANN_trained(H)
```

3.4. Experimentos y Resultados

Para la validación del módulo de reconocimiento de objetos se realizaron tres experimentos: i) Base de datos propia, ii) Simulación en Webots y iii) Implementación en robot real.

Figura 3.12: 20 objetos reales considerados para la base de datos.

El experimento con base de datos propia, consiste en el uso de imágenes de objetos reales. Para esto se capturaron 20 imágenes de los 20 objetos reales mostrados en la Figura 3.12. En el experimento de simulación en Webots, se eligieron 10 de los objetos de la base de datos propia y se construyeron sus modelos simulados para poderlos colocar en el ambiente virtual. En el experimento de la implementación en robot real, se utilizaron 7 cajas de cereal diferentes, las cuales se muestran en la Figura 3.13. Éstas tienen forma, peso y tamaño similares. Sin embargo, se diferencian por la información que proporciona el producto que contienen.

Figura 3.13: Cajas de cereales diferentes utilizadas para los experimentos con el robot real.

Las dimensiones tanto de las cajas de cereal como los objetos de la base de datos propia se encuentran entre 20x20x20 a 150x150x150 mm aproximadamente y con un peso menor a 300 gramos. Estas especificaciones son necesarias ya que se hace uso del robot humanoide NAO, ver Apéndice A para la implementación del sistema.

3.4.1. Base de datos propia

La validación con la base de datos propia utiliza los objetos de la Figura 3.14 y los objetos de la Figura 3.15 para realizar diversos experimentos. Para los primero objetos se utilizaron 10 imágenes de diferentes tamaños. Mientras que para los segundos objetos se utilizaron 20 imágenes diferentes.

Figura 3.14: Imágenes reales de la base de datos, tomadas con fondo blanco, para el primer experimento. (1) Corrector, (2) Caja de regalo, (3) Teléfono celular, (4) Dado, (5) Tortuga, (6) Bolsa de regalo, (7) Resistol en barra, (8) Caja de cereal, (9) Bote de medicamento, (10) Libro.

Para los 10 objetos se realizó una prueba donde se utilizan 5 imágenes de entrenamiento y 5 de evaluación. Para los 5 objetos se realizaron diferentes pruebas que incluyen de 5 a 15 imágenes para entrenamiento y el resto para evaluación.

Figura 3.15: Imágenes reales de la base de datos, tomadas con fondo blanco, para el segundo experimento. (1) Corrector, (2) Caja de regalo, (3) Teléfono celular, (4) Plumón, (5) Tortuga.

En la Tabla 3.2 se describen los parámetros principales de cada prueba para cada experimento tales como: el número de imágenes de entrenamiento, imágenes de evaluación, número máximo de neuronas de la red neuronal y pasos de entrenamiento. El número de neuronas y épocas son seleccionadas para ser el doble o más que el número de imágenes totales, de este

manera conseguir una mejor distribución de la red, de un valor de 100 hasta 500 neuronas.

Tabla 3.2: Parámetros principales por experimento base de datos propia.

Experimento	**Prueba**	**Entrenamiento**	**Evaluación**	**Neuronas**	**Epocas**
1	Primera	5	5	500	500
2	Primera	5	5	100	100
2	Segunda	5	1	100	100
2	Tercera	5	15	200	200
2	Cuarta	10	10	200	500
2	Quinta	15	5	200	500

Cada una de las pruebas tomó imágenes aleatorias de la base de datos para cada objeto. En estas pruebas, la red neuronal utiliza un número grande de neuronas y de épocas, ya que permite agrupar y aprender cada objeto con mayor detalle. Además, como algunos de los objetos no lucen de la misma manera en diferente puntos de vista, el incremento de este parámetro facilita el aprendizaje de los histogramas.

Los resultados de los experimentos se muestran en las respectivas matrices de confusión. De igual manera, se presenta un análisis de curvas RoC para la clasificación de cada experimento. En la Tabla 3.3, se muestra la matriz de confusión del primer experimento. Se puede observar que sólo tres de los diez objetos registraron un solo falso positivo en la clasificación. El resto de los objetos obtuvo clasificación exitosa. Estos falsos positivos pueden deberse a imágenes borrosas con histogramas que son muy diferentes a los de su objeto y buscan el más cercano.

Tabla 3.3: Matriz de confusión para el primer experimento.

	Primera									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	5	0	0	0	0	0	0	0	0	0
2	0	5	0	0	0	0	0	0	0	0
3	0	0	5	0	0	0	0	0	0	0
4	0	0	0	5	0	0	0	0	0	0
5	0	0	0	0	4	0	0	1	0	0
6	0	0	0	0	0	5	0	0	0	0
7	0	0	0	0	0	0	5	0	0	0
8	1	0	0	0	0	0	0	4	0	0
9	0	0	0	0	0	0	0	0	4	1
10	0	0	0	0	0	0	0	0	0	5

En la Tabla 3.4, se muestran las matrices de confusión de las pruebas para el segundo experimento. A diferencia del primer experimento, en este, se observa que en la primera y en la tercera prueba solo un objeto obtuvo falsos positivos. En la segunda cuarta y quinta prueba

se observa que dos de los objetos han sido clasificados erróneamente en algunas de sus imágenes, sin embargo, el porcentaje de clasificación correcta es mayor. La mayoría de estos falsos positivos son clasificados en el objeto 1. Esto puede deberse a que el objeto 1 es el primero en clasificarse en la red neuronal lo que le da más peso.

Tabla 3.4: Matrices de confusión del segundo experimento.

Primero						**Segundo**						**Tercero**					
	1	**2**	**3**	**4**	**5**		**1**	**2**	**3**	**4**	**5**		**1**	**2**	**3**	**4**	**5**
1	5	0	0	0	0	**1**	10	0	0	0	0	**1**	5	0	0	0	0
2	0	5	0	0	0	**2**	0	10	0	0	0	**2**	0	5	0	0	0
3	0	0	5	0	0	**3**	0	0	10	0	0	**3**	0	0	5	0	0
4	2	0	0	3	0	**4**	4	0	0	6	0	**4**	2	0	0	3	0
5	0	0	0	0	5	**5**	1	0	0	0	9	**5**	0	0	0	0	5

Cuarto						**Quinto**					
	1	**2**	**3**	**4**	**5**		**1**	**2**	**3**	**4**	**5**
1	1	0	0	0	0	**1**	13	0	0	1	1
2	0	1	0	0	0	**2**	0	15	0	0	0
3	0	0	1	0	0	**3**	0	0	15	0	0
4	1	0	0	0	0	**4**	4	0	0	10	1
5	0	0	0	1	0	**5**	1	1	2	0	11

El porcentaje obtenido de clasificación correcta de cada experimento y el tiempo de entrenamiento para cada uno, se presentan en la Tabla 3.5. El primer experimento obtuvo un porcentaje de clasificación correcta del 94 % , para el segundo experimento los dos mejores obtuvieron un porcentaje de clasificación correcta obtuvieron 92 % y el resto obtuvo resultados arriba del 80 % de clasificación correcta.

Tabla 3.5: Resultado de los experimentos.

Experimento	**Prueba**	**Porcentaje (%)**	**Tiempo (seg)**
1	Primera	94	19.639
2	Primera	92	0.351
2	Segunda	80	0.426
2	Tercera	85.33	1.246
2	Cuarta	90	2.344
2	Quinta	92	3.457

Los tiempos de ejecución dependen de los parámetros seleccionados para el entrenamiento de la red. Los principales parámetros de los que depende son las épocas y la longitud del vector de entrada. A manera de análisis se toma el primer tiempo, el último tiempo, él número de épocas correspondientes del experimento 2 y se construye una gráfica. La gráfica se presenta en

la Figura 3.16, en ésta se observa el comportamiento de la cantidad de épocas contra el tiempo en segundos. De ésta gráfica se calculó la ecuación de aproximación lineal que la describe:

$$time = 0{,}007765(epochs) - 0{,}4255 \tag{3}$$

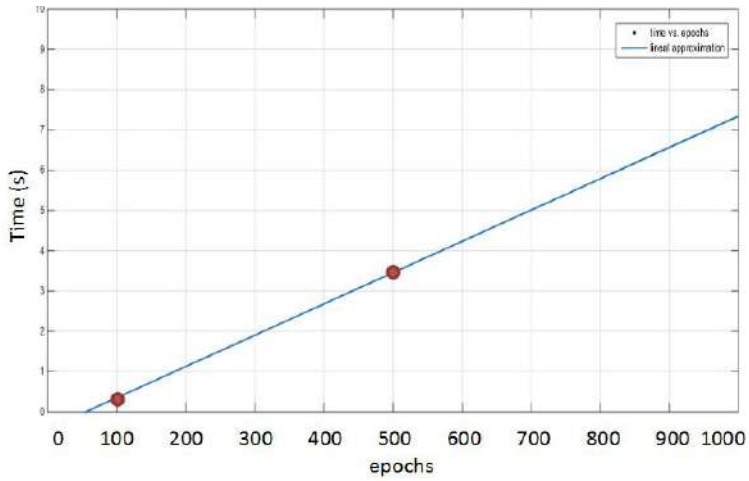


Figura 3.16: Gráfica de épocas contra tiempo, aproximación lineal.

En la gráfica de épocas contra tiempo, se observa que el incremento en tiempo permanece pequeño de una cantidad de épocas a otra. Por ejemplo, si las imágenes de entrenamiento incrementaran a 20 por cada uno de los 5 objetos, lo mejor sería incrementar el número de neuronas a 700 dando un tiempo de 5.01 s aproximadamente. Entonces, el módulo es considerado rápido ya que se siguen hablando de segundos y no de minutos ni horas como en otros modelos de reconocimiento de objetos.

Adicionalmente, se obtuvo un análisis de la sensibilidad y la especificidad de los experimentos haciendo uso de los resultados obtenidos en sus matrices de confusión mediante los verdaderos positivos (TP), verdaderos negativos (TN), falsos positivos (FP) y falsos negativos (FN). Los valores para cada prueba se muestran en la Tabla 3.6 para el primer experimento y en la Tabla 3.7 para el segundo. En particular, para el experimento dos se toman en cuenta todos los datos de clasificación de las cinco pruebas, esto corresponde a 180 imágenes en total.

Tabla 3.6: Características operativas de la prueba del experimento 1.

Objeto	TP	TN	FP	FN	Sensibilidad	Especificidad	1-Especificidad
(1)	5	44	1	0	1	0.9778	0.0222
(2)	5	45	0	0	1	1	0
(3)	5	45	0	0	1	1	0
(4)	5	45	0	0	1	1	0
(5)	4	45	0	1	0.8	0.986	0.014
(6)	5	45	0	0	1	1	0
(7)	5	45	0	0	1	1	0
(8)	4	44	1	1	0.8	0.986	0.014
(9)	4	45	0	1	0.8	1	0
(10)	5	45	1	0	1	0.9783	0.0217

Tabla 3.7: Características operativas de las pruebas del experimento 2.

Objeto	TP	TN	FP	FN	Sensibilidad	Especificidad	1-Especificidad
(1)	34	129	15	2	0.944	0.896	0.104
(2)	36	143	1	0	1	0.993	0.007
(3)	36	142	2	0	1	0.986	0.014
(4)	22	142	2	14	0.611	0.986	0.014
(5)	30	142	2	6	0.833	0.986	0.014

Estas características operativas pueden reformularse y representarse gráficamente, ver Figuras 3.17, 3.18 y 3.19 para el experimento 1 y la Figura 3.20 para el experimento 2. Este tipo de gráfica se llama curva ROC (*Receiver Operating Characteristic*). Se gráfica el valor de verdaderos positivos contra el valor de 1-falsos positivos para las diferentes clases resultantes.

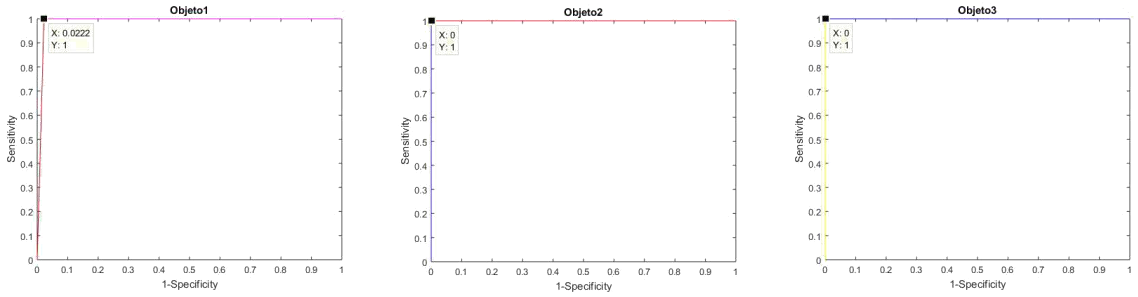


Figura 3.17: Curvas de RoC para los objetos del 1 al 3 del experimento 1.

Esta curva demuestra diferentes puntos: (1) Muestra la compensación entre sensibilidad y especificidad (cualquier incremento en sensibilidad se acompaña con un decremento en especificidad), (2) Entre mas cerca este la curva de la esquina izquierda el clasificador es más preciso, (3) Entre más cerca esté la curva de la diagonal de 45 grados, menos preciso es el clasificador.

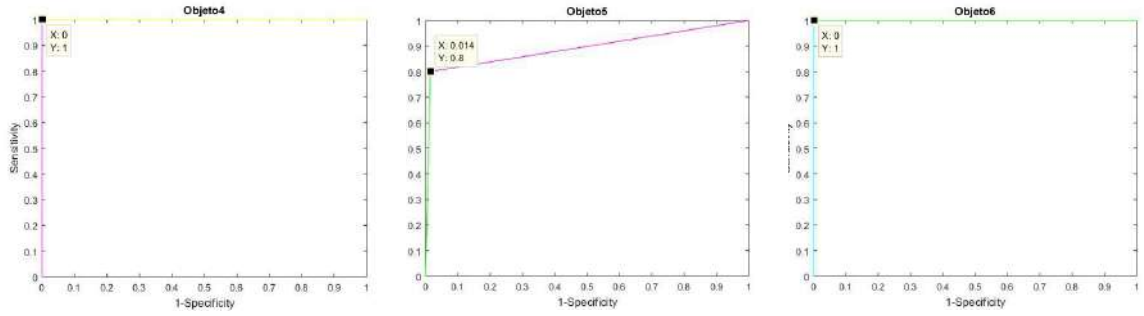


Figura 3.18: Curvas de RoC para los objetos del 4 al 6 del experimento 1.

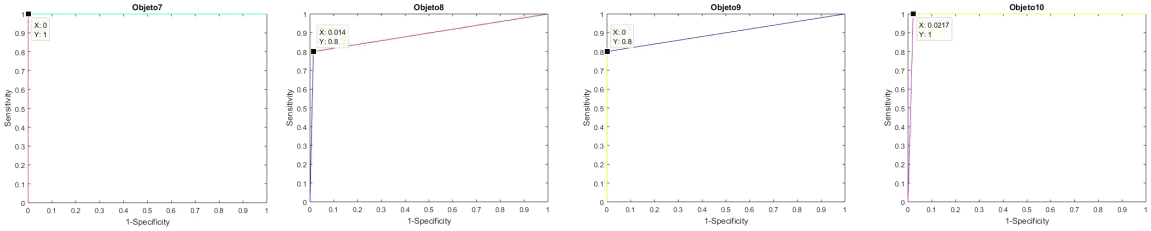


Figura 3.19: Curvas de RoC para los objetos del 7 al 10 del experimento 1.

De las gráficas de los 10 objetos se observa que las curvas se acercan más a la esquina superior izquierda lo cual representa una buena precisión para la clasificación de los objetos en el experimento 1. A pesar de que el objeto 5, 8 y 9 obtuvieron algunos falsos positivos, estos aun presentan una buena precisión de clasificación. De igual manera para el experimento 2, en las gráficas, se observan curvas con buena precisión. Sin embargo, el objeto 4 y 5, obtuvieron algunos falsos positivos que afectaron un poco la precisión del objeto 1 ya que fueron confundidos con este último.

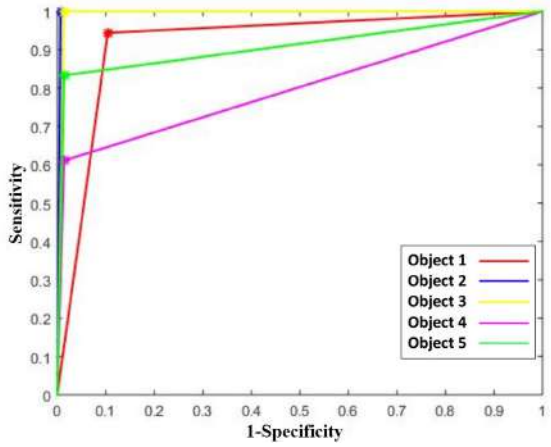


Figura 3.20: Curvas de RoC para cada objeto del experimento 2.

3.4.2. Simulación en Webots

El objetivo del módulo de reconocimiento de objetos, además de formar parte del sistema modular, es el de ser usado por un robot humanoide. Es por esto que se realizó la simulación en Webots de un ambiente semi-controlado y el robot humanoide NAO para la validación del módulo. Para llevar a cabo la simulación, en la fase de entrenamiento se utilizaron 10 de los objetos de la base de datos propia, estos son lo mostrados en la Figura 3.21.

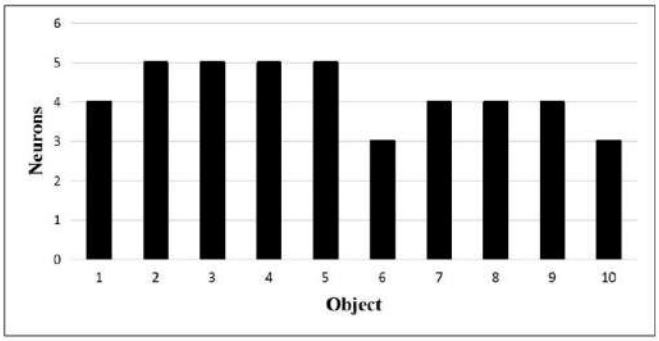
Figura 3.21: Objetos seleccionados para el entrenamiento: (1) corrector, (2) caja de regalo, (3) celular, (4) dado, (5) tortuga, (6) bolsa de regalo, (7) pegamento, (8) cereal, (9) medicina y (10) libro.

Con estos objetos se llevaron a cabo dos experimentos usando 5 y 10 imágenes diferentes para el entrenamiento de la red neuronal. En la Tabla 3.8, se muestran los parámetros utilizados para la red neuronal GCS en cada experimento. Los parámetros incluyen: número de imágenes de entrenamiento, número máximo de neuronas en la red, pasos de entrenamiento y tiempo total que tardó la red en entrenar. El número de neuronas y épocas son seleccionadas para ser el doble o un poco más que el número de imágenes totales, de este manera conseguir una mejor distribución de la red. El tiempo de entrenamiento se obtuvo con la implementación del módulo en MATLAB 2015 en una computadora con procesador Intel(R) Core(TM) i7-4770 3.40GHz, con memoria RAM de 12 GB y sistema operativo Windows 8 de 64 bits.

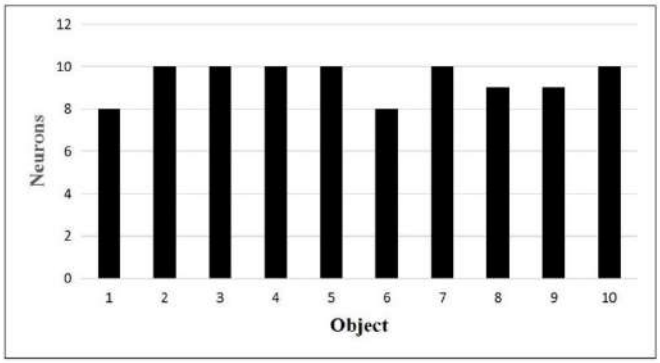
Tabla 3.8: Parámetros para la red neuronal: Entrenamiento de Objetos

Experimento	**Imágenes de Entrenamiento**	**Número de Neuronas**	**Número de Épocas**	**Tiempo Total Obtenido (seg)**
1	5	100	100	1.193
2	10	250	250	13.536

El resultado del entrenamiento de la red neuronal en ambos experimentos se exhibe en la Figura 3.22 (a) para el primer experimento y (b) para el segundo experimento. Estas gráficas representan el número de objetos entrenados contra la cantidad de neuronas que le corresponden. Como se puede observar, los objetos han sido asociados a más de una neurona. Cuando el reconocimiento de alguno de los objetos se lleve a cabo, el módulo buscará por el grupo donde se encuentre la neurona más cercana.



(a)



(b)

Figura 3.22: Número de neuronas asociadas a cada objeto por experimento: (a) Primero (b) Segundo

Una vez realizado el entrenamiento de las redes neuronales, lo siguiente es probar el reco-

nocimiento de este. El reconocimiento se apoya de la simulación en Webots, donde se realiza la búsqueda de objetos.

Figura 3.23: Los 10 objetos reales simulados en Webots.

Para éstos se construyeron los modelos simulados de los objetos reales mostrados en la Figura 3.23. Estos objetos se ubicaron con posición aleatoria en el centro del ambiente simulado sobre una mesa. Para la evaluación del reconocimiento de objetos se propuso realizar la búsqueda de cuatro de los 10 objetos utilizando ambas redes neuronales entrenadas del primer y segundo experimento. Los objetos a buscar son: corrector líquido, cereal, tortuga y libro.

Figura 3.24: Capturas de la búsqueda de los objetos por el robot NAO.

La búsqueda del objeto se realiza de la siguiente manera. El robot camina alrededor de la mesa donde se encuentran los 10 objetos, como se observa en la Figura 3.24. Mientras está haciendo su recorrido, el robot toma varias capturas con diferente perspectiva de cada uno de los objetos que encuentre en ella. El robot toma imágenes durante su recorrido, segmenta los objetos de cada una y los envía como imágenes individuales a la evaluación del módulo de reconocimiento de objetos. El algoritmo de segmentación de imágenes se describe en el siguiente capítulo junto con el *Módulo de Manipulación de Objetos*.

La evaluación de la búsqueda de los objetos se muestra en la matriz de confusión de la Tabla 3.9. Para la búsqueda de los dos primeros objetos se utilizó la red entrenada del primer experimento. Mientras que para la búsqueda de los otros dos se utiliza la segunda. En la matriz de confusión se observa que para el primer experimento se obtuvo un porcentaje de clasificación correcta de 80 %. Confundiendo el corrector con la tortuga y la medicina con la bolsa de regalo, ambos solo en una ocasión. Por otro lado, el segundo experimento obtuvo el 100 % de clasificación correcta. Estos resultados son concluyentes para afirmar que el módulo muestra eficiencia en un

Tabla 3.9: Matriz de confusión de los objetos a buscar.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
corrector	4	0	0	0	1	0	0	0	0	0
cereal	0	0	0	0	0	1	0	0	4	0
tortuga	0	0	0	0	5	0	0	0	0	0
libro	0	0	0	0	0	0	0	0	0	5

ambiente simulado. Además, se corrobora que para algunos objetos sólo bastan cinco imágenes para tener un buen entrenamiento. Pero para un mejor reconocimiento se necesitan más de cinco imágenes por objeto y un número grande de neuronas.

3.4.3. Implementación en robot real

Después de realizar la validación del módulo de reconocimiento de objetos en simulación, se procede a realizar la implementación del módulo con el robot real. Esta implementación consiste en que el robot aprenda las 7 cajas de cereales diferentes mostradas en la Figura 3.25 y posteriormente las identifique.

Figura 3.25: Siete cajas de cereales diferentes utilizadas.

De estas 7 cajas de cereales el robot toma 20 imágenes de cada una en diferentes vistas. La captura de las imágenes es asistida por el usuario, el cual debe cambiar de vista el objeto a cada captura. En la Figura 3.26 se muestra una secuencia de imágenes donde se observa cómo el usuario apoya al robot con el entrenamiento de los objetos cambiando la posición de estos.

Figura 3.26: Usuario apoyando al robot NAO en la captura de imágenes.

Una vez terminada la captura de las imágenes, la base de datos se envía al módulo de reconocimiento de objetos para su entrenamiento. Los parámetros para el entrenamiento de la red neuronal GCS se muestra en la Tabla 3.10. Esto parámetros incluyen: número de objetos, número de imágenes de entrenamiento, número máximo de neuronas, pasos de adaptación, etc. Se entrenaron los 7 objetos con las 20 imágenes cada uno. Ya que se tienen 140 imágenes a entrenar, el número de neuronas y épocas se eligen a 400 para tener una buena distribución de los datos. Los valores para los parámetros de adaptación, decremento y umbral son los mismo que se han utilizado a lo largo de este trabajo de tesis.

Tabla 3.10: Parámetros para el módulo de reconocimiento de objetos

Parámetros	
Número de Objetos	7
Imágenes de Entrenamiento	20
Número de Neuronas	400
Número de Épocas	400
Adaptación (neurona ganadora) ε_b	0.05
Adaptación (vecindario neurona ganadora) ε_n	0.005
Decremento α	0.999
Umbral de Eliminación θ	0.001

El resultado del entrenamiento de la red neuronal se exhibe en el gráfico de la Figura 3.27. Este gráfico muestra la clasificación de las 7 cajas de cereal por número de neuronas. Los cereales han sido asociados a más de una neurona para incluir cada una de las vistas de estos. Como en secciones anteriores se mencionó, cuando el reconocimiento se lleve a cabo, el módulo buscará por el grupo de neuronas donde se encuentre la más cercana.

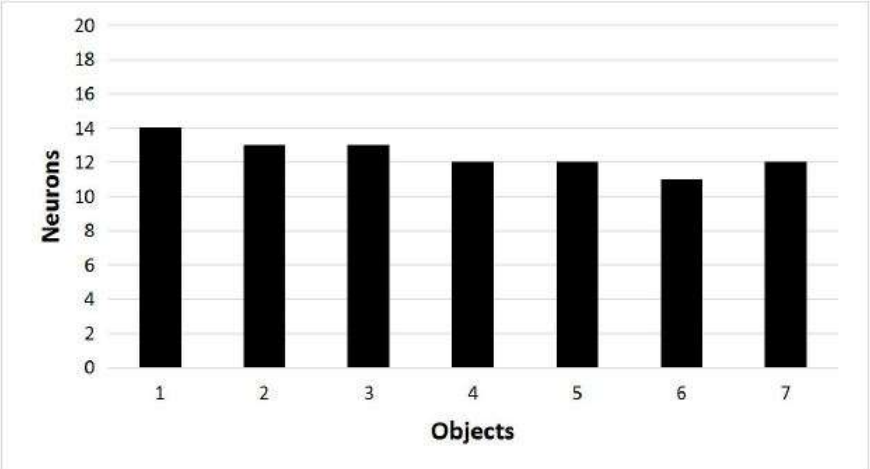


Figura 3.27: Gráfico de barras de objetos contra neuronas del entrenamiento de las 7 cajas de cereal. Neuronas asociadas a cada objeto donde el objeto es cada cereal.

La evaluación del módulo se realizó de dos formas: (i) objeto por objeto y (ii) recorrido alrededor de una plataforma. La primera se refiere a la presentación de los siete objetos frente al robot NAO para la captura de 5 imágenes más. Con esta nueva base de datos se procedió a identificar a los objetos. Cada una de las imágenes se enviaron al reconocimiento del módulo y con la neurona resultante se procedió a identificar el objeto al que pertenecía. El resultado de la identificación se presenta en la matriz de confusión 3.11. El porcentaje de clasificación fue de 74.25 %. En ésta se observa cómo hubo más confusión entre objetos, ya que éstos comparten más características en común. Por ejemplo, el objeto 3 es la caja de zucaritas, el cual es confundido con el chocokrispis y el choco zucaritas en una ocasión. Las tres cajas de cereal son de la empresa kellogg's y en estas cajas la información nutrimental tiene el mismo diseño. En general todas las cajas de cereal tienen ciertas características en común. Las de la misma compañía tienen el mismo diseño del empaque y el tamaño de la caja, con la diferencia de los personajes, el color y el nombre.

Tabla 3.11: Matriz de confusión de la identificación de las 7 cajas de cereal.

	1	**2**	**3**	**4**	**5**	**6**	**7**
1	5	0	0	0	0	0	0
2	0	5	0	0	0	0	0
3	0	0	3	0	1	0	1
4	2	0	0	3	0	0	0
5	1	0	1	0	3	0	0
6	0	1	1	0	0	3	0
7	0	0	1	0	0	0	4

En la segunda evaluación, el robot NAO ejecutó un recorrido alrededor de una plataforma de 30 cm de alto. En esta plataforma se colocaron 5 de las 7 cajas de cereal. En la Figura 3.28 se observa al robot NAO comenzando el recorrido alrededor de la plataforma con las 5 cajas de cereal sobre ésta. El robot NAO recorrió la plataforma caminando hacia su izquierda girando su cabeza 45° para capturar imágenes cada dos paso.

Figura 3.28: Robot NAO comenzando el recorrido alrededor de la plataforma con las 5 cajas de cereal.

El robot capturó un total de 10 imágenes, de esas imágenes se procedió a segmentar los objetos e identificarlos. Las imágenes capturadas durante el recorrido alrededor de la plataforma se muestran en la Figura 3.29. En algunas imágenes se observa que algunos objetos de atrás de los que se pretenden capturar se visualizan. En otras se observa que los objetos aparecen muy juntos y esto podría evitar una segmentación de objetos correcta. Es por esto que se utiliza el enfoque *Next Best View* para la búsqueda de objetos. Este enfoque se describe en el capítulo del *Módulo de Navegación Espacial*.

Figura 3.29: Las 10 imágenes capturadas por el robot NAO en su recorrido alrededor de la plataforma.

La segmentación de los objetos de este experimento se realizó de forma manual recortando cada uno de los objetos observados en las capturas. De la segmentación se obtuvieron un total de 25 imágenes, 5 por objeto. El resultado de la evaluación se muestra en la matriz de confusión de la Tabla 3.12. Los resultados muestran una clasificación de 56 %, con los 5 objetos confundidos en algunas ocasiones. Como ya se mencionó, los objetos se confunden por que sus diseños son muy parecidos. Se concluye que para poder obtener una mejor búsqueda de objeto no basta con ir capturando imágenes alrededor de la plataforma que los soporta. Es necesario un análisis de imágenes para detectar la ubicación del objeto buscado. Si el sistema es capaz de detectar por donde es que puede estar el objeto, el robot tendrá la capacidad de acercarse a él y capturar imágenes donde se aprecie mejor. Como ya se ha mencionado, este análisis de imágenes y los resultados se describe en la sección de búsqueda de objeto en la capítulo 5 de este trabajo de tesis.

Tabla 3.12: Matriz de confusión de los 5 objetos a buscar contra los 7 entrenados.

	1	**2**	**3**	**4**	**5**	**6**	**7**
1	3	2	0	0	0	0	0
2	0	4	0	0	1	0	0
3	2	0	3	0	1	0	1
5	2	0	1	0	2	0	0
6	2	1	0	0	0	2	0

3.5. Conclusión

En este capítulo se presentó el desarrollo y validación del módulo de reconocimiento de objetos utilizando el extractor de características tipo parche y la red neuronal GCS ya descritos en la sección de *Marco Teórico* de este documento. El módulo se validó usando una base de datos propia tomada de objetos reales. El sistema se entrenó con diferente número de objetos e imágenes por objeto. Los resultados obtenidos son suficientemente buenos con porcentajes de clasificación correcta arriba de 80 %. Gracias a los experimentos, se considera que los falsos negativos pueden evitarse entregando al módulo mejores vistas de los objetos a reconocer para que se cubra cada detalle de estos.

La validación del módulo mediante un robot móvil se realizó utilizando la plataforma humanoide NAO tanto en simulación como con plataforma real. Se buscaba con ésta observar si la cámara del robot afecta los porcentajes de reconocimiento del módulo y conocer el tiempo que tarda el robot en identificar correctamente los objetos que el usuario requiera. Los resultados muestran que la cámara afectan poco el aprendizaje, mientras que el robot realice las captura de las imágenes de manera estática. Además, el aprendizaje de las cajas de cereal demuestra que el módulo puede diferenciar objetos con características similares. La identificación de los objetos de manera dinámica requiere del apoyo de un algoritmo de búsqueda de objeto. Este algoritmo se basa en el enfoque *Next Best View* el cual se describe y evalúa en el capítulo 5.

Una observación del módulo de reconocimiento de objetos es que la clase nula no puede ser considerada ya que si el robot trata de reconocer un objeto que no ha aprendido, éste lo asociará o a un objeto ya existente o a una neurona libre que no haya sido utilizada en el entrenamiento. Si lo segundo llega a pasar entonces el robot habrá aprendido un objeto nuevo.

Algunas desventajas del módulo son que el uso del método A-KAZE, al ser un método de extracción de características tipo parche, requiere que los objetos de forma libre deban de tener algunas texturas de otra manera no encontraría la cantidad de *keypoints* necesarios para poder describir al objeto. Otra observación es que si se quiere entrenar la red con gran cantidad de objetos e imágenes, el tiempo de entrenamiento aumentará dependiendo la cantidad de estos. Como trabajo futuro para este módulo se propone reforzar el aprendizaje de los objetos utilizando otras características extras como color, forma, etc.

Capítulo 4

Módulo de Manipulación de Objetos

La manipulación de objetos por brazos robóticos en la industria está extensamente estudiada y resuelta con precisión. La manipulación en un entorno semiestructurado, en comparación con un entorno estructurado en su totalidad, representa más dificultades. Considerar manipulación por parte de una plataforma robótica móvil de servicio convierte el problema más interesante. Una robot móvil de servicio por lo general cuenta con uno o dos manipuladores, los cuales se consideran sus brazos. Estos manipuladores pueden ser de tres a seis grados de libertad en la mayoría de los casos. El problema de manipulación en un ambiente semiestructurado debe tener en cuenta diversos factores. Un robot debe reconocer y manipular objetos ya sea para cambiarlos de lugar o entregarlos a una persona. También, debe calcular las coordenadas tridimensional del lugar donde se encuentra el objeto, la cinemática y la planificación de movimientos necesarios para poder tomarlo.

Figura 4.1: Tareas que debe cumplir el módulo de Manipulación de Objetos.

Tomando en cuenta las consideraciones descritas anteriormente, se plantea el desarrollo del módulo de manipulación de objetos descrito en el diagrama de bloques de la Figura 4.1. El diagrama resume las tareas secuenciales que se requieren para localizar objetos espacialmente y manipularlos. El módulo considera que el objeto ya ha sido reconocido y que se encuentra al alcance del robot, como se muestra en el ejemplo de la Figura 4.2. Lo primero es la detección del objeto en la imagen con el fin de estimar sus coordenadas 2D encontrando el *BoundingBox* que lo contiene. Teniendo la información 2D del objeto, lo siguiente es la estimación de las coordenadas 3D obteniendo la relación de proyección de la imagen que captura la cámara del robot al mundo real. Con el objeto identificado, lo siguiente es realizar los movimientos necesarios para tomarlo con el efector final de uno o ambos brazos robóticos mediante cinemática inversa.

El módulo está pensado para poder ser utilizado en cualquier plataforma robótica móvil que cuente con cámara y al menos un brazo manipulador con efector final de pinza. Para fines prácticos, en esta sección se desarrollan las tareas de estimación de la posición y manipulación del objeto para el robot humanoide NAO, ver Apéndice A. Sin embargo, se indican las modificaciones que se deben realizar para poder implementarlo en otra plataforma.

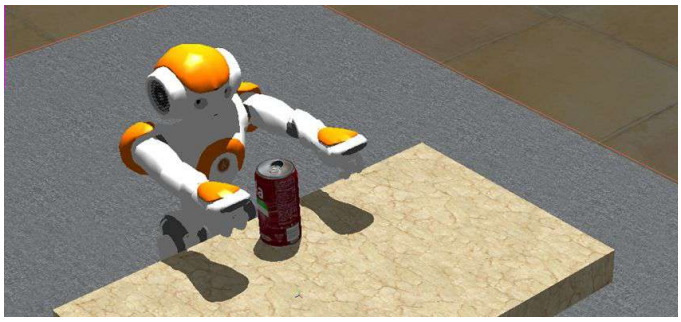


Figura 4.2: Robot humanoide NAO en entorno simulado Webots, aproximación al objeto a manipular.

El desarrollo del módulo se describe en las siguientes secciones. La sección 4.1 describe las consideraciones para el desarrollo del algoritmo de manipulación de objetos. Esta sección incluye: el procedimiento realizado para la detección del objeto a manipular, el proceso utilizado para estimar las coordenadas 3D del objeto y el calculo de movimientos en los manipuladores. En la sección 4.2 se muestra la estructura del módulo de manipulación de objetos en seudocódigo. La implementación del módulo se presenta mediante experimentos y resultados en la sección 4.3. Al final se plasma una breve conclusión del capítulo.

4.1. Algoritmo de manipulación de objetos

La tarea de manipulación de objetos depende en gran medida de la plataforma robótica en la cual se implementa el sistema. Los brazos manipuladores en los robots de servicio pueden tener diferente configuración. En esta tesis, el algoritmo de manipulación de objetos está diseñado para que un robot humanoide NAO pueda tomar objetos de una plataforma con altura conocida. El robot debe utilizar su cámara para detectar el objeto, estimar las coordenadas tridimensionales en el espacio real y utilizar los dos manipuladores para tomarlo. Los objetos a manipular podrán localizarse en cualquier posición dentro del área de trabajo de los manipuladores del robot. No será necesario conocer las dimensiones de los objetos pero estos deben estar dentro de un rango tal que el robot pueda manipular (10x10x10 a 100x100x100 mm aproximadamente) y un peso menor que 300 g. Para la detección del objeto en la imagen, se utilizan algoritmos de procesamiento de imagen 2D como binarización, detección de bordes y componentes conectados. Con la detección del objeto se estiman las dimensiones y la posición de éste en la imagen. Posteriormente, se usa relación de proyección para determinar las coordenadas 3D mediante las características de la cámara. La imagen de la cámara da información acerca de la dirección desde ésta al centro del objeto en el mundo real tomando en cuenta su posición con respecto a la plataforma con altura conocida. Para la manipulación de objetos se utiliza un modelo que

incluye cinemática. El modelo debe permitir que el robot presione con los dos efectores finales el objeto contra las palmas de sus manos para sostenerlo.

4.1.1. Detección de Objetos

En el módulo de manipulación de objetos, la detección permite estimar las coordenadas 2D del objeto en la imagen mediante ciertas funciones de procesamiento de imagen. La estimación de posición se lleva a cabo encontrando el contorno mayor del objeto más cercano al centro de la imagen. El proceso parte de una imagen de entrada como la que se muestra la Figura 4.3 (a). La imagen se procesa para cambiar sus colores a escala de grises y posteriormente se binariza como se muestra en la Figura 4.3 (b). Después, es utilizado un algoritmo de detección de contornos para determinar el tamaño de las regiones encontradas en la imagen, como se muestra en la Figura 4.3 (c). De todos los contornos encontrados es seleccionado el más cercano al centro de la imagen de mayor longitud. A esta región se procede a calcular su *BoundingBox*. El resultado es la posición y el tamaño estimados del objeto en la imagen, como se observa en la Figura 4.3 (d).

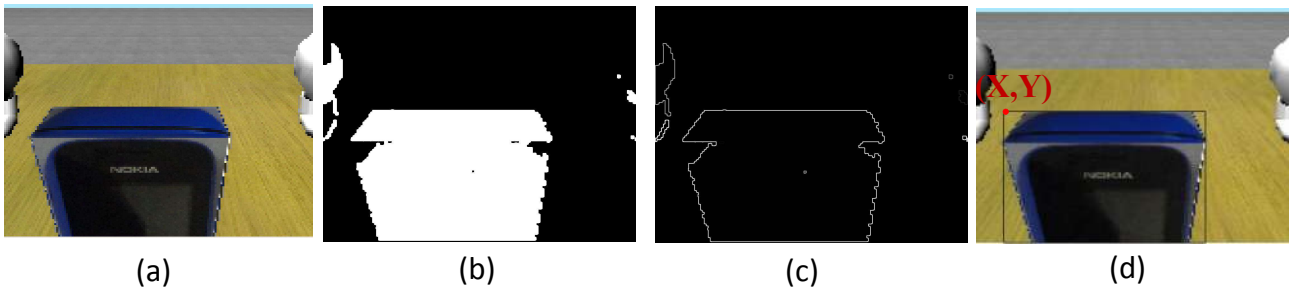


Figura 4.3: Función de procesamiento de imagen para la detección del objeto. (a) Imagen de entrada, (b) Binarización, (c) Detección de contornos, (d) Estimación de *BoundingBox* que lo contiene.

4.1.2. Estimación de Posición

La información obtenida en la detección del objeto se utiliza como aproximación para determinar las coordenadas 2D del objeto en el mundo real. El cálculo de la tercer coordenada del objeto, el cual es la profundidad, se lleva a cabo gracias a ciertas restricciones del sistema. Esta coordenada está relacionada con la plataforma de altura conocida donde se encuentran los objetos a manipular por parte del robot. Además, de la ventaja que tiene el robot NAO de llevar su cámara inferior a una posición casi paralela a la plataforma pudiendo observar fácilmente esta superficie. Si se va a utilizar otra plataforma robótica, la restricción mínima es que la cámara se encuentre posicionada por arriba del brazo o brazos y ésta tenga al menos un grado de libertad que le permita girar en ángulo *pitch* hacia abajo.

Para estimar las coordenadas 3D del objeto de la imagen al mundo real, es necesario conocer las características físicas de la(s) cámara(s), además de su posición y su orientación con respecto a un marco de referencia global en la plataforma robótica. Primero, se debe definir el área de trabajo del(los) manipulador(es) tal que le sea posible tomar objetos desde una plataforma de

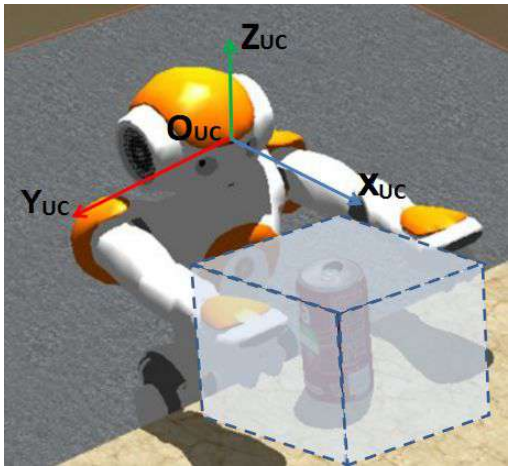


Figura 4.4: Representación del espacio de trabajo de los manipuladores del robot NAO.

altura conocida. Ya que se utiliza al robot humanoide NAO como plataforma de desarrollo, se ha determinado el área de trabajo de los manipuladores considerando la posición de sus cámaras y la plataforma de altura conocida. El espacio de trabajo de los brazos del robot se encuentra delimitado a cierta altura y a lo largo del alcance de los manipuladores. Lo que restringe al objeto a estar dentro del espacio de trabajo que se define en la Figura 4.4. En ésta se observa al robot lo más cerca posible que le permite la plataforma que soporta al objeto a manipular y se define un marco de referencia local en base a su cámara superior (UC).

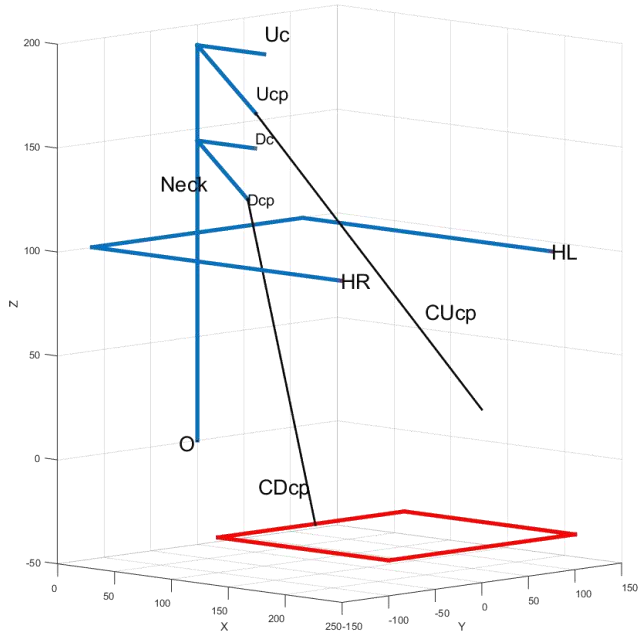


Figura 4.5: Representación esquemática de robot NAO omitiendo sus piernas.

El robot NAO tiene su centro de masa cerca de su torso. Este es considerado como marco de referencia para todo su cuerpo. Partiendo de este punto se ha esquematizado al robot humanoide

NAO desde su posición cero y omitiendo sus piernas. En la Figura 4.5, se muestra un esquemático en el cual se observa la posición de las cámaras y manipuladores con respecto al marco de referencia del robot que incluyen la representación de la superficie de una mesa a una altura conocida. Las cámaras se han definido como **Uc** para la cámara superior y **Dc** para la cámara inferior. A estas se le agrega el sufijo **p** indicando el giro en el ángulo *pitch* que se realiza para poder observar la superficie de la plataforma. **CUcp** y **CDcp** refieren al eje óptico de la cámara superior e inferior, respectivamente, hacia la superficie de la plataforma, la cual se indica en rojo. **HR** y **HL** son los efectores finales de los manipuladores del robot, mano derecha y mano izquierda respectivamente.

Entonces, el robot utiliza su cámara inferior para encontrar la información de profundidad. Las coordenadas 3D se estiman con información acerca de la escena ya que las posiciones de las cámaras y posición de la superficie de la plataforma donde se encuentren los objetos son conocidas. De acuerdo a las especificaciones del robot NAO mostradas en la Figura 4.6, de (a) se sabe que el robot puede girar su cabeza hasta 29.5° en ángulo *pitch* lo que le permite observar la superficie de la plataforma. De (b), la cámara inferior tiene una desviación de 39.7° relativa a la cámara superior, esta a su vez tiene una desviación de 1.2° desde su localización ambas respecto al ángulo *pitch*. De (c) se conoce el campo de visión de la cámara en el ángulo *yaw* de la cabeza. Con esa información el tamaño desde una imagen en píxeles al mundo real en milímetros puede ser calculado.

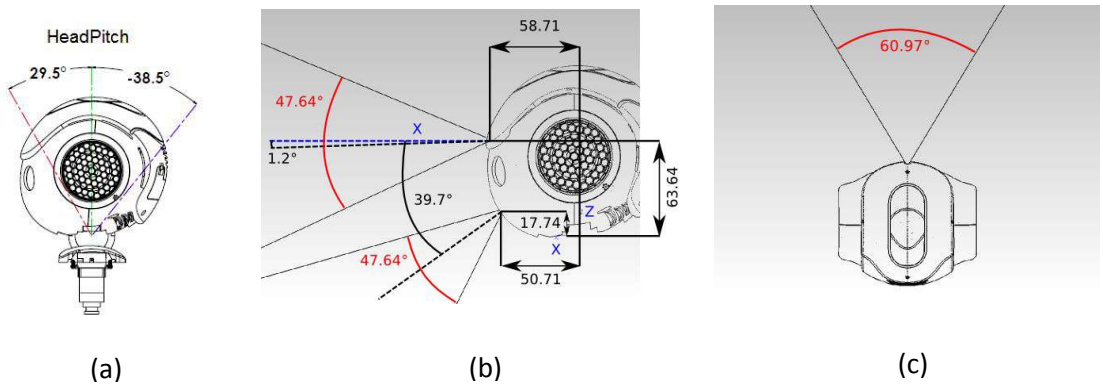


Figura 4.6: Especificaciones de la cabeza del robot NAO. (a) Rango del ángulo *pitch*; Campo de visión de las cámaras (b) ángulo *pitch*, (c) ángulo *yaw*, (Imagen extraída de [18]).

Para la realización del cálculo del tamaño de la imagen de píxeles a milímetros, primero se utilizan los ejes ópticos de cada una de las cámaras (**CUcp** y **CDcp**) para calcular su proyección a la superficie de la plataforma. En la Figura 4.7 se presenta la esquematización de los ángulos de campo de visión de cada cámara para realizar el análisis de posición. La esquematización incluye: i) la distancia del marco de referencia central en el eje x del robot a la posición de ambas cámaras **Ucx** y **Dcx**, ii) la distancia de la superficie de la plataforma al centro de masa $\overline{PO}$, iii) la distancia del centro de masa a la cámara inferior $\overline{ODc}$, iv) la distancia de la cámara inferior a la cámara superior $\overline{DcUc}$, v) el ángulo de giro en *pitch* realizado para observar la superficie, vi) el ángulo de campo de visión de la cámara, vii) el ángulo de desviación de la cámara inferior con respecto a la superior. Se observa que el cálculo de ambas proyecciones del eje óptico se resume a la resolución de su respectivo triángulo rectángulo. Así como, para encontrar el tamaño de

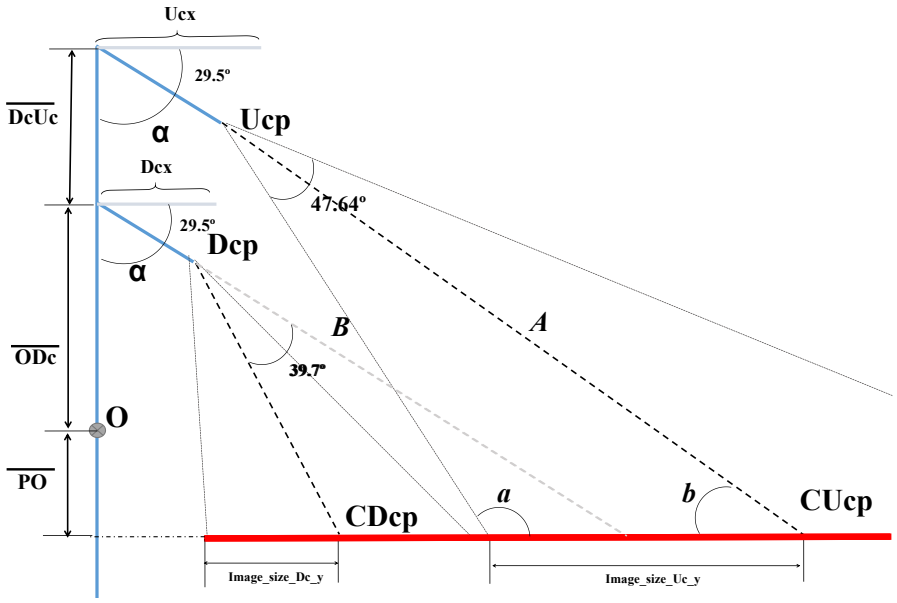


Figura 4.7: Esquemático de los ángulos de visión *pitch* de ambas cámaras del robot NAO.

cada una de las imágenes proyectadas en la superficie se resume a la resolución de un triángulo escaleno. Ambos formados por la proyección de los ángulos de campo de visión de las cámaras. Las constantes del sistema corresponden a los valores de las especificaciones técnicas del robot NAO [18]:

- ✓ Ucx = 58.71 mm
- ✓ Dcx = 50.71 mm
- ✓ $\overline{PO} = AlturaPlataforma - 333{,}09mm$ (distancia de las piernas al torso)
- ✓ $\overline{ODc}$ = 126.56 mm + 17.74 mm (distancia del torso-cuello-cámara inferior)
- ✓ $\overline{DcUc}$ = 45.9 mm (distancia cámara inferior a superior)

El cálculo del punto donde se proyecta el eje óptico **CUcp** y el tamaño de la proyección de la imagen en la superficie **Image_size_Uc_(x/y)**, (ancho, alto) se lleva a cabo de la siguiente forma:

Se determina el ángulo α complementario al giro de 29,5° considerando la desviación de 1.2°.

$$\alpha = 90° - (29{,}5° + 1{,}2°) \tag{1}$$

Se obtiene la hipotenusa del triángulo rectángulo formado por la distancia de la superficie de la plataforma a la cámara superior y por la proyección del eje óptico de la cámara a la superficie.

$$Hipotenusa = \frac{\overline{PO} + \overline{ODc} + \overline{DcUc}}{\cos\alpha} \tag{2}$$

Se calcula la distancia del eje x del robot a la proyección del eje óptico de la cámara **CUcp** sobre la plataforma.

$$distancia\_x = (\sin\alpha)(Hipotenusa) \tag{3}$$

El punto **CUcp** tendrá las coordenadas correspondientes a (x= $distancia\_x$, y = 0, z= $\overline{PO}$) respecto al marco de referencia del robot NAO. La distancia en el eje y es 0 mm ya que las cámaras se encuentran alineadas en este mismo eje. Se obtiene el lado A del triángulo escaleno correspondiente a la proyección del eje óptico en la plataforma:

$$A = Hipotenusa - Ucx \tag{4}$$

Se calcula el ángulo b como complementario al ángulo α:

$$b = 90 - \alpha \tag{5}$$

Se determina el ángulo a como complementario del triángulo formado con al ángulo b y la mitad del ángulo del campo de visión proyectado de la cámara:

$$a = 180 - (b + \frac{47{,}64}{2}) \tag{6}$$

El tamaño de la proyección de la imagen en la superficie **Image_size_Uc_y** que corresponde a la altura (**Y**) es:

$$Image\_size\_Uc\_y = \frac{(\sin\frac{47{,}64}{2})(A)}{\sin a} = Y_{image} \tag{7}$$

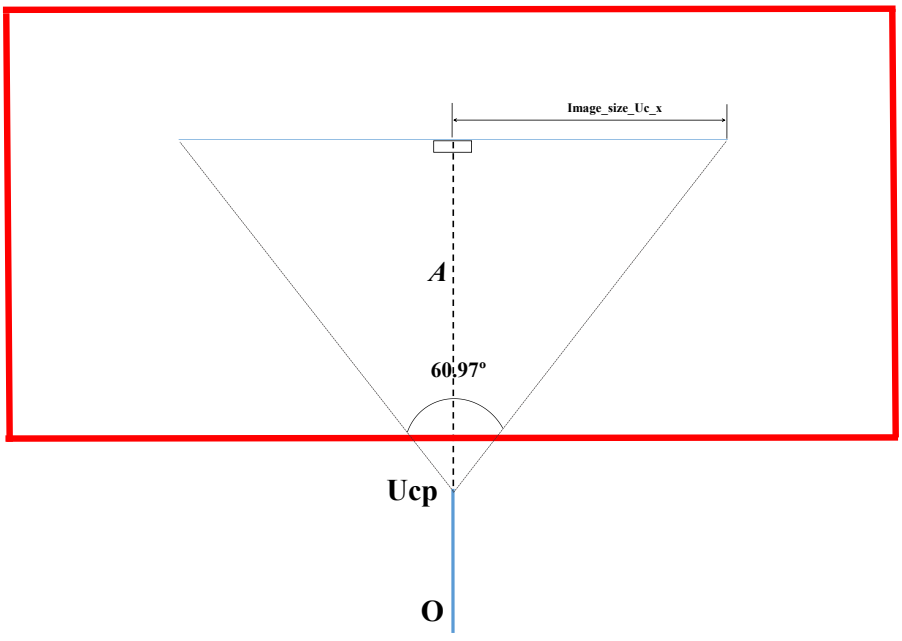


Figura 4.8: Esquemático del ángulo *yaw* del campo de visión de la cámara superior del robot NAO.

Para el cálculo del tamaño de la proyección de la imagen en la superficie **Image_size_Uc_x**, correspondiente al ancho (**X**), se considera el ángulo *yaw* del campo de visión de la cámara.

En la Figura 4.8 se muestra el esquema donde se plasma el triángulo rectángulo formado por el eje óptico de proyección desde la cámara superior a la superficie de la mesa y el ángulo del campo de visión de 60.97°. El cateto **A** de este rectángulo es el mismo que se ha calculado para el triángulo escaleno correspondiente al centro del eje óptico proyectado. Entonces el cálculo de **Image_size_Uc_x** se reduce a encontrar el cateto opuesto del triángulo mediante la función trigonométrica de tangente, la mitad del ángulo del campo de visión *yaw* y el cateto **A**:

$$Image\_size\_Uc\_x = (\tan\frac{60{,}97}{2})(A) = X_{image} \tag{8}$$

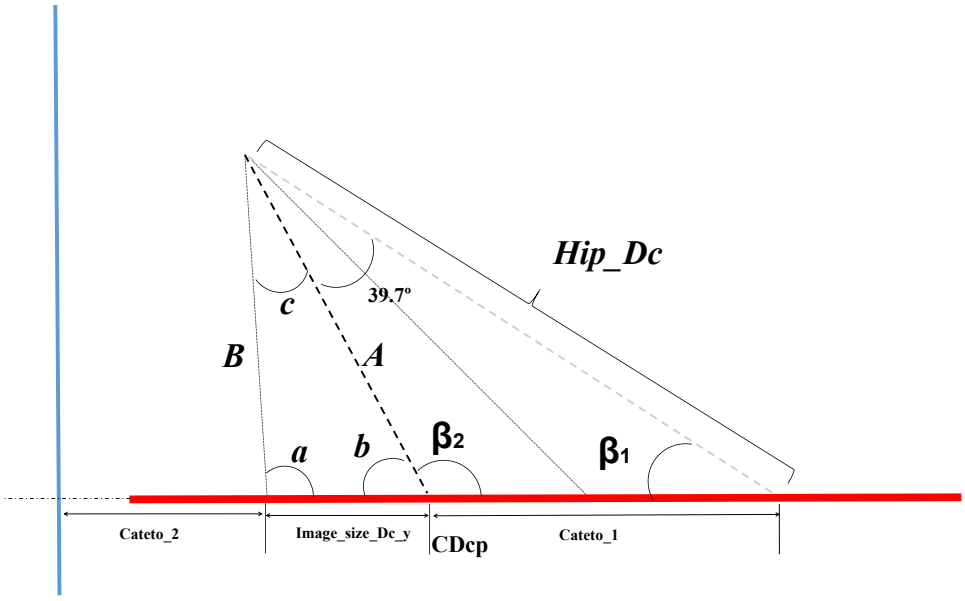


Figura 4.9: Esquemático del ángulo de visión *pitch* de la cámara inferior del robot NAO.

El cálculo de la proyección del eje óptico de la cámara inferior **CDcp** y el tamaño de la proyección de la imagen en la superficie **Image_size_Dc_y** es un poco más complejo que el anterior. Al igual que el cálculo de **CUcp**, este parte de la proyección del eje óptico de la cámara inferior a la superficie mostrado en la Figura 4.7. Es necesario encontrar el ángulo α el cual es el mismo que para **CDcp** ya que se realiza el mismo giro en el ángulo *pitch*. De igual manera se obtiene la hipotenusa del triángulo rectángulo formado por la distancia de la superficie de la plataforma a la cámara inferior y por la proyección del eje óptico de la cámara inferior a la superficie.

$$Hipotenusa2 = \frac{\overline{PO} + \overline{ODc}}{\cos\alpha} \tag{9}$$

Ya que existe una desviación de 39.7° de la cámara inferior con respecto a la cámara superior, es necesario encontrar unos valores extras. La resolución se basa en el esquemático de la Figura 4.9. En éste se observa cómo es necesario resolver dos triángulos escalenos más para encontrar el tamaño **Image_size_Dc_y**. Entonces, se calcula el ángulo β_1 como complementario al ángulo α:

$$\beta_1 = 90 - \alpha \tag{10}$$

Se calcula el lado Hip_Dc del triángulo escaleno correspondiente a la proyección de la posición de la cámara inferior y la proyección del eje óptico de la misma:

$$Hip\_Dc = Hipotenusa2 - Dcx \tag{11}$$

Se obtiene el ángulo β_2 como complementario del triángulo formado con al ángulo β_1 y el ángulo de desviación:

$$\beta_2 = 180 - (\beta_1 + 39{,}7) \tag{12}$$

Se determina el lado A del triángulo escaleno correspondiente a la proyección del eje óptico de la cámara y su ángulo del campo visión:

$$A = \frac{(\sin\beta_1)(Hip\_Dc)}{\sin\beta_2} \tag{13}$$

Se encuentra el ángulo b como suplementario al ángulo β_2:

$$b = 180 - \beta_2 \tag{14}$$

Se obtiene el ángulo a como complementario del triángulo formado con al ángulo b y $c = \frac{47{,}64}{2}$:

$$a = 180 - (b + c) \tag{15}$$

El tamaño de la proyección de la imagen en la superficie **Image_size_Dc_y** es:

$$Image\_size\_Dc\_y = \frac{(\sin c)(A)}{\sin a} = Y_{image} \tag{16}$$

La longitud del cateto opuesto del triángulo rectángulo formado por la distancia de la superficie de la plataforma a la cámara inferior y por la proyección a la superficie es igual a:

$$CatetoOpuesto = (\sin\alpha)(Hipotenusa2) = Cateto\_2 + Image\_size\_Dc + Cateto\_1 \tag{17}$$

Con lo anterior se sabe que el Cateto_1 es igual a:

$$Cateto\_1 = \frac{(\sin 39{,}7)(A)}{\sin\beta_1} \tag{18}$$

Entonces, se calcula la distancia del eje x del robot a la proyección del eje óptico de la cámara **CDcp**.

$$distancia\_x1 = CatetoOpuesto - Cateto\_1 \tag{19}$$

Finalmente, el punto **CDcp** tendrá las coordenadas correspondientes a (x= $distancia\_x1$, y = 0, z= $\overline{PO}$) respecto al marco de referencia del robot NAO. Al igual que para **CUcp**, la distancia en el eje y es 0 mm ya que las cámaras se encuentran alineados a este mismo eje.

Al igual que con la cámara superior, el cálculo del tamaño de la proyección de la imagen en la superficie **Image_size_Dc_x**, correspondiente al ancho (**X**), se debe considerar el ángulo *yaw* del campo de visión la cámara. En la Figura 4.10 se muestra un esquema similar al de la cámara superior. El cateto **A** de este rectángulo es el mismo que se ha calculado para el triángulo escaleno correspondiente a la proyección del eje óptico. Entonces el cálculo de **Image_size_Dc_x** será similar al cálculo de **Image_size_Uc_x**:

$$Image\_size\_Dc\_x = (\tan\frac{60{,}97}{2})(A) = X_{image} \tag{20}$$

Entonces, para fines prácticos, se establece que la proyección de la imagen de la cámara superior e inferior a la superficie de la plataforma es (en mm):

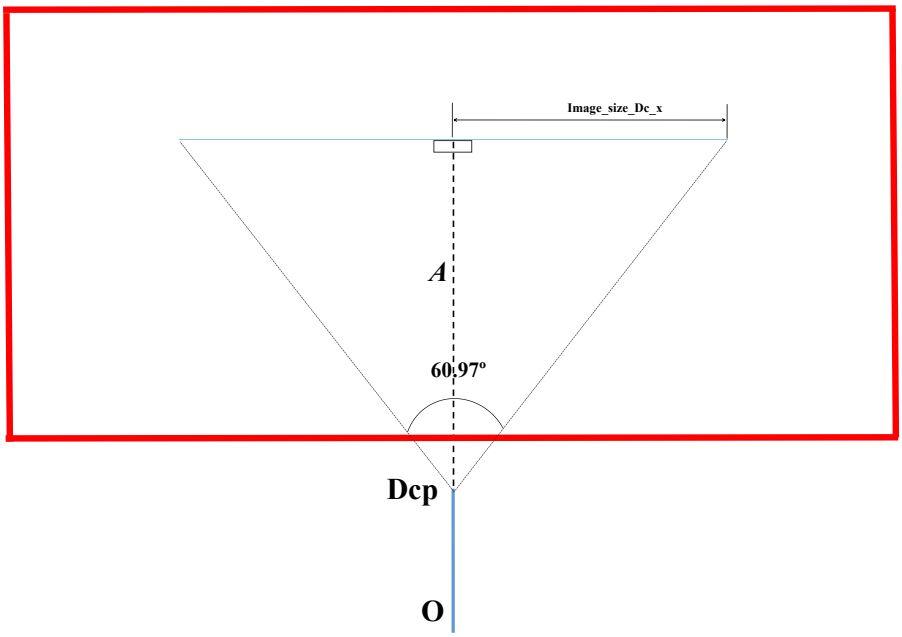


Figura 4.10: Esquemático del ángulo *yaw* del campo de visión de la cámara inferior del robot NAO.

- ✓ $Ancho_{Ucp} = 2 * Image\_size\_Uc\_x$
- ✓ $Alto_{Ucp} = 2 * Image\_size\_Uc\_y$
- ✓ $CUcp = [distancia_x, 0, \overline{PO}]$
- ✓ $Ancho_{Dcp} = 2 * Image\_size\_Dc\_x$
- ✓ $Alto_{Dcp} = 2 * Image\_size\_Uc\_y$
- ✓ $CDcp = [distancia_x 1, 0, \overline{PO}]$

Una vez obtenidos los tamaños de la proyección de la imagen en la superficie de la plataforma, se procede a determinar su correspondiente valor en píxeles. Sólo hace falta realizar una equivalencia entre distancia en píxeles a distancia en mm para calcular la posición del objeto en la superficie de la plataforma, equivalente a $Ancho_{Ucp} \times Alto_{Ucp}$ y $Ancho_{Dcp} \times Alto_{Dcp}$ respectivamente. Cuando el robot toma las imágenes desde sus cámaras, como las de la Figura 4.11, la proyección del eje óptico al mundo real ya es conocida.

De ambas imágenes se obtiene la posición en **Y** del objeto a manipular. En dado caso de que la cámara superior no vea el objeto se toman los datos de la inferior. Esta posición es determinada para cada manipulador, **Y_HL** y **Y_HR**, obteniendo las distancia hacia los bordes desde el punto **CDcp** al objeto detectado. Para fines prácticos se decide estimar la posición en **Z** asumiendo al objeto como prisma rectangular, entonces se promedian ambas distancia de **Y** para obtener una posición en **Z** ligeramente arriba de la superficie de la plataforma. La posición en **X**, correspondiente a la profundidad con respecto al robot, se obtiene con la imagen capturada sólo de la cámara inferior. Si está cámara no lo ve, se suma la coordenada de **CUcp** el tamaño de la imagen y se estima con la cámara superior. Entonces el procedimiento es sencillo,

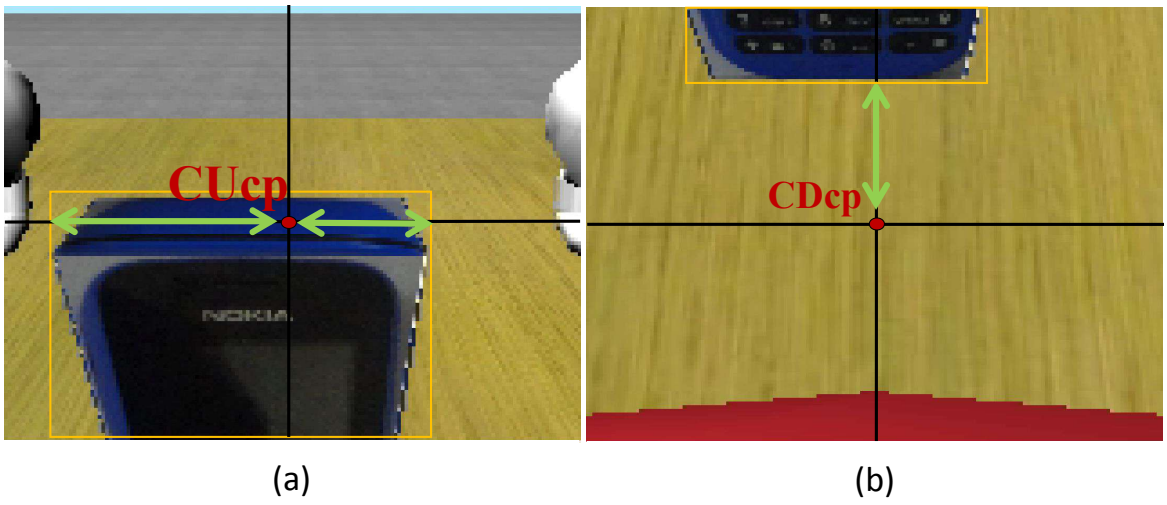


Figura 4.11: Cálculo de la información de profundidad del objeto imagen-mundo real.

solo basta calcular la distancia del punto **CDcp**, al borde inferior del objeto detectado. Una vez obtenido el tamaño en milímetros se determina su posición tomando en cuenta la posición del punto **CDcp** o **CUcp**, según sea el caso. Éstos están definidos desde el marco de referencia del robot NAO en su posición cero. Las coordenadas 3D del objeto en la superficie de la plataforma queda definido de la siguiente manera:

- ✓ X = $CDcp_x$ + distancia al borde inferior del objeto
- ✓ Y_HL = $CUcp_y$ + distancia al borde izquierdo del objeto
- ✓ Y_HR = $CUcp_y$ + distancia al borde derecho del objeto
- ✓ Z = $CUcp_z$ + (Y_HL + Y_HR)/2 (se asume un objeto cuadrado)

4.1.3. Manipulación del Objeto

En el apéndice A se presenta el análisis cinemático de los manipuladores del robot NAO. Este análisis puede ser simplificado para fines prácticos de esta aplicación, ya que el robot sólo realizará manipulación de objetos con sus brazos en un espacio de trabajo limitado. La función de manipulación de objeto utiliza las coordenadas 3D obtenidas en la estimación de posición para calcular la cinemática necesaria y así llevar a los manipuladores del robot a tomar el objeto identificado.

Considerando que el robot realiza la manipulación en una posición de pie donde las piernas están fijas, (posición cero del robot NAO), la cinemática inversa se restringe a encontrar ciertos ángulos. Estos ángulos son los mostrados en la Figura 4.12, donde primero se calcula *ShoulderPitch* para la coordenada **X** y **Z** del objeto. Mientras que el cálculo de *ShoulderRoll* y *ElbowRoll* se utiliza la coordenada en **Y** y se toma en cuenta el análisis del ángulo anterior. En cuanto al ángulo *ElbowYaw* se define constante para que las manos se encuentren hacia adentro listas para tomar el objeto. La función de agarre le permite al robot tomar cualquier objeto partiendo de la posición cero con los brazos extendidos. Ya que la articulación *ShoulderPitch*

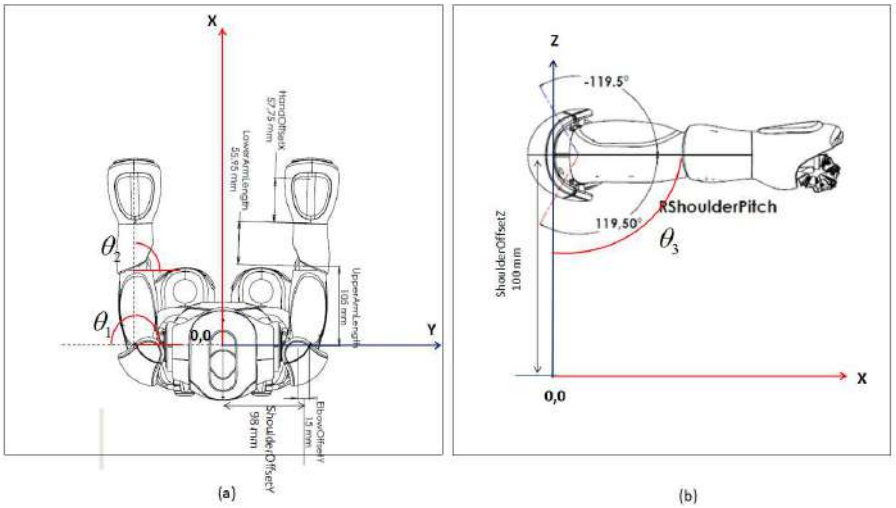


Figura 4.12: Ángulos de los brazos considerados para determinar la cinemática inversa. (a) vista superior, (b) vista de brazo derecho lateral.

cuenta con un espacio de trabajo de -119.5° a 119.5°, es fácil ver que esta se restringe de 0° a 31.32° donde para el primero basta con subir a la posición cero y el segundo es limitado por la altura de la superficie de la mesa conocida.

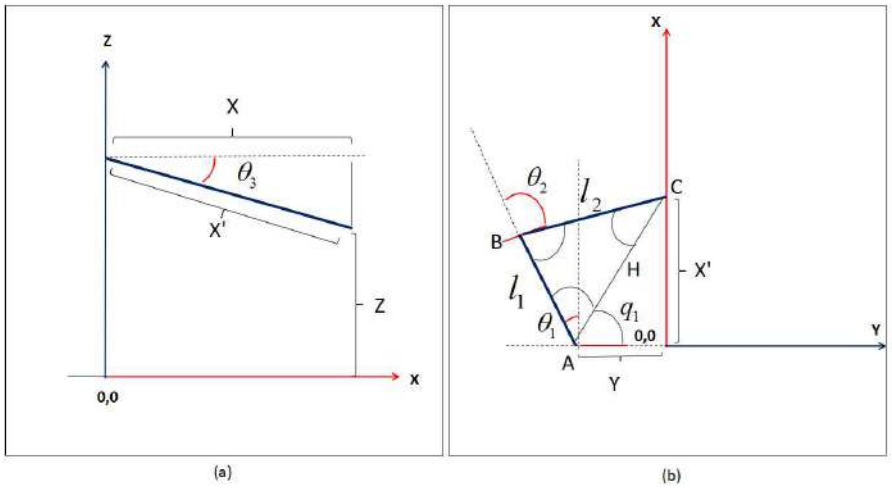


Figura 4.13: Esquemáticos de los brazos considerados para determinar la cinemática inversa. (a) vista de brazo derecho lateral, (b) vista superior.

Entonces, el cálculo de la posición de los ángulos de los brazos se desarrolla a continuación basado en los esquemáticos de la Figura 4.13. Primero se obtiene **X'** que es la hipotenusa del triángulo que forma la coordenada en **X** del objeto, la diferencia entre la coordenada en **Z** y la posición del brazo con respecto al marco de referencia del robot (OffSetZ=100 mm de los brazos del robot NAO).

$$X'^2 = X^2 + (OffSetZ - Z)^2 \tag{21}$$

El ángulo *ShoulderPitch* se obtiene mediante una relación trigonométrica entre la dimensión de **X** y **X'**:

$$ShoulderPitch = \arccos(\frac{X}{X'}) \tag{22}$$

Con la dimensión de **X'** y **Y** del objeto se determina la hipotenusa H y el ángulo q_1 del triángulo que forman desde el origen del brazo a éste:

$$H^2 = X'^2 + Y^2 \tag{23}$$

$$q_1 = \arctan(\frac{X'}{Y}) \tag{24}$$

Utilizando la ley de cosenos se determinan los ángulos del triángulo escaleno que forman ambos eslabones y la distancia del origen del brazo a la posición **X'** y la coordenada en **Y** del objeto:

$$A = \arccos(\frac{l_2^2 - H^2 - l_1^2}{-2Hl_1}) \tag{25}$$

$$B = \arccos(\frac{H^2 - l_2^2 - l_1^2}{-2l_2l_1}) \tag{26}$$

El ángulo *ShoulderRoll* y el *ElbowRoll* se calculan substrayendo los respectivos ángulos al ángulo cero de cada articulación:

$$ShoulderRoll = 90 - (q_1 + A) \tag{27}$$

$$ElbowRoll = 180 - B \tag{28}$$

4.2. Módulo de Manipulación de objetos

En base a lo descrito en la sección anterior se construye el módulo de manipulación de objetos. El módulo se divide en tres tareas: *Detección*, *Estimación* y *Manipulación*.

Algoritmo 6: Módulo de manipulación de objetos. *Detección*

Datos: ***I(2)*** Dos imágenes: cámara superior e inferior.
Resultado: ***P[(x1,y1),(x2,y2)]*** Posición del objeto en ambas imágenes.

```
1 para i ← 1 a I(2) hacer
2     temp=LeerImagen(I)
3     I=EscalaGrises(temp)
4     BW = Binarizacion(I, 0.4)
5     cc=RegionDetection(BW,8)
6     BIG= Max(cc)
7     BB=inf.BoundingBox.BIG
8     P(xi)=BB(1)
9     P(yi)=BB(2)
```

En la tarea de *Detección*, como lo describe el Algoritmo 6, se recibe como entrada dos imágenes de la escena tomada con la cámara superior e inferior del robot. Se utiliza procesamiento

de imagen para detectar el objeto en ambas: cambio a escala de gises, binarización, dilatación y detección de regiones. Se detecta la región mayor cercana al centro y se recupera su tamaño para estimar la posición del objeto en la imagen. El resultado es una posición estimada del objeto en cada imagen.

Algoritmo 7: Módulo de manipulación de objetos. *Estimación*

Datos: ***P[(x1,y1),(x2,y2)]*** Coordenadas 2D del objeto en ambas imágenes,
ImageSize tamaño de la imagen
Resultado: ***P(X,Y,Z)*** coordenadas 3D

1 ***RealSize***
2 (Xo, Yo, Zo)
3 $x = \frac{(Px2)(RealSizex)}{ImageSizey}$
4 $y = \frac{(Py1)(RealSizey)}{ImageSizex}$
5 $z = CONSTANTE$
6 $P(X) = Xo - x$
7 $P(Y) = Yo + y$
8 $P(Z) = Zo + z$

La tarea de *Estimación*, descrita en el Algoritmo 7, toma como entrada la información obtenida en la tarea de *Detección* para ambas imágenes. Esta información corresponde a las coordenadas 2D del objeto en ambas imágenes *P[(x1,y1),(x2,y2)]* y al tamaño de cada imagen en píxeles ***ImageSize***. El cálculo de las coordenadas se apoya del conocimiento del espacio de trabajo de los brazos del robot NAO ***RealSize***. Además del marco de referencia de la proyección de la imagen establecido en la superficie de la plataforma como (Xo, Yo, Zo). Con las proyecciones de los ejes ópticos CUcp y CDcp en la superficie de la plataforma hacia el centro de masa del robot NAO, las coordenadas 3D del objeto en la imagen se convierte de escala de píxeles a escala en mm.

Algoritmo 8: Módulo de manipulación de objetos. *Manipulación*

Datos: ***P(X,Y,Z)*** Posición 3D

1 $OffSetZ$
2 $X'^2 = X^2 + OffSetZ^2$
3 $ShoulderPitch = \arccos(\frac{X}{X'})$
4 $H^2 = X'^2 + Y^2$
5 $q_1 = \arctan(\frac{X}{Y})$
6 $A = \arccos(\frac{l_2^2 - H^2 - l_1^2}{-2Hl_1})$
7 $B = \arccos(\frac{H^2 - l_2^2 - l_1^2}{-2l_2l_1})$
8 $ShoulderRoll = 90 - (q_1 + A)$
9 $ElbowRoll = 180 - B$
10 CinemáticaDirecta(***ShoulderPitch,ShoulderRoll,ElbowRoll***)

La tarea de *Manipulación*, descrita en el Algoritmo 8, toma como entrada las coordenadas 3D del objeto para calcular el punto final al cual deben llegar cada uno de los manipuladores

del robot NAO. Para que cada manipulador se coloque los más cerca que pueda del objeto, se calculan los límites de este con la información obtenida en la detección de imagen y las especificaciones técnicas del robot. Una vez obtenidos los puntos se procede la cinemática directa de ambos manipuladores para realizar los movimientos necesarios que lleven a tomar el objeto partiendo y regresando de la posición cero.

4.3. Experimentos y resultados

Se realizaron dos experimentos para la validación del módulo de manipulación de objetos. El primero se lleva a cabo en el ambiente simulado Webots con 10 objetos. Una vez validado se realizó el segundo experimento. Éste se realiza con un robot NAO real, en un ambiente controlado y con 7 objetos.

4.3.1. Simulación en Webots

Para los experimentos de esta sección se utiliza el simulador Webots con el fin de validar el módulo de manipulación de objetos implementándolo en un robot humanoide NAO. Para fines prácticos, se construyeron los 10 objetos virtuales presentados en la Figura 4.14. El peso de estos es menor de 300 g lo que le permite al robot tomarlos sin ningún problema. Además, los objetos son más grandes que su tamaño real para permitirle al robot capturar mejores imágenes.

Figura 4.14: Objetos virtuales en Webots.

Los 10 objetos se distribuyeron en una mesa de 300 mm de altura en el centro del ambiente virtual. La tarea del robot es tomar cada uno de estos objetos para validar el módulo. La posición de éstos es desconocida pero se encuentran dentro del área de trabajo de los brazos del robot NAO. Para cada uno de los 10 objetos el robot es colocado de frente con posición

Tabla 4.1: Coordenadas X,Y,Z para 10 objetos, mano derecha e izquierda.

Objeto	**Posición (X,Y,Z) mano derecha**	**Posición (X,Y,Z) mano izquierda**
Corrector	(140.82,86.72,74.47)	(140.82,-28.91,74.47)
Regalo	(154.79,68.366,79.16)	(154.79,-47.04,79.16)
Celular	(144.89,41.626,65.39)	(144.89,-76.15,65.39)
Dado	(160.08,-0.611,0)	(160.08,1.96,0)
Tortuga	(129.64,53.7984,78.7)	(129.64,-69.12,78.7)
Bolsa	(163.6,71.9349,0)	(163.6,-70.24,0)
Resistol	(155.93,58.8084,25.66)	(155.93,-34.71,25.66)
Cereal	(162.18,88.3365,11.97)	(162.18,-71.58,11.97)
Medicina	(142.56,47,36.93)	(142.56,-62.92,36.93)
Libro	(147.1,53.166,78.29)	(147.1,-57.56,78.29)

cero mostrado en la Figura 4.15. Entonces, se ejecuta el módulo de manipulación de objetos para detectar el objeto, sus coordenadas 3D y la cinemática directa para tomarlo. El módulo mueve la cabeza del robot 29.5° en ángulo *pitch* para observar la superficie de la mesa, captura una imagen con cada cámara y las analiza. El análisis consiste en el cálculo de las coordenadas 3D del objeto con referencia al marco base del robot que se encuentra en su torso. Una vez obtenidas estas coordenadas, el módulo calcula los puntos que deben alcanzar cada uno de los manipuladores del robot NAO para tomar el objeto. Posteriormente, calcula su cinemática directa y la ejecuta.

Figura 4.15: Secuencia de movimientos que realiza el robot NAO para tomar la bolsa.

A manera de visualización en la Tabla 4.1, se muestran los resultados que arroja el módulo referentes al cálculo de las coordenadas 3D para cada uno de los objetos en este experimento. En ésta se identifica al objeto por su nombre y se despliegan las coordenadas en (X,Y,Z) obtenidas por el módulo tanto para el brazo derecho como para el izquierdo. Después de obtener las coordenadas del objeto, el módulo calcula la cinemática directa para cada una de las articulaciones de los manipuladores del robot.

A manera de visualización, en la Tabla 4.2 se presentan los ángulos calculados por el módulo para la ejecución de la cinemática directa de los manipuladores del robot NAO. Para cada uno el módulo de manipulación de objetos calcula los ángulos LShoulderPitch, RShoulderRoll, LShoulderRoll, RElbowRoll y LElbowRoll. De igual manera, la secuencia en la Figura 4.15 ejemplifica cómo se lleva a cabo la manipulación por parte del robot. El robot se colocó enfrente de este objeto, giró su cabeza para observar la mesa, capturó una imagen con cada cámara y utilizó el módulo de manipulación de objetos para detectar el objeto, determinar su posición

Tabla 4.2: Ángulos en grados encontrados para las posiciones de los 10 objetos.

Objeto	R/LShoulderPitch	RShoulderRoll	LShoulderRoll	RElbowRoll	LElbowRoll
Corrector	10.27	-10.82	39.29	80.23	-88.5
Regalo	7.66	-17	27.08	77.62	-83.616
Celular	13.43	-31.70	14.97	88.5	-80.26
Dado	31.99	-31.88	31.09	60.72	-60.71
Tortuga	9.32	-29.98	22.01	88.5	-88.5
Bolsa	31.43	-0.84	1.74	41.11	-41.98
Resistol	25.48	-16.18	26.67	66.95	-72.72
Cereal	28.49	4.00	5.06	41.42	-50.38
Medicina	23.86	-27.19	19.71	83.84	-79.60
Libro	8.39	-26.33	24.21	87.63	-86.46

3D y calcular la cinemática directa de sus brazos para levantarlo.

Para estos experimentos los resultados son positivos consiguiendo 9 de los 10 objetos levantados por el robot correctamente. El objeto que no se levantó correctamente fue el dado, ya que en la sección de detección, el dado tiende a umbralizarse ya que presenta una textura similar a la mesa. Para fines prácticos, se propone evitar los colores claros en los objetos y tener un color claro en la mesa.

4.3.2. Ejecución con robot real

Después de validar el módulo en el ambiente de simulación Webots, se realiza la validación con el robot NAO real. Esta validación se lleva a cabo en un ambiente controlado. Los objetos a manipular son las cajas de cereal utilizadas para la validación del *Módulo de Reconocimiento de Objetos* en el capítulo anterior. El sistema modular ejecuta el módulo de manipulación de objetos una vez que el robot se encuentre lo más cerca del objeto a manipular. Por esta razón, las pruebas experimentales en esta sección se ejecutan con el robot en posición cero, de pie frente a una plataforma de 300 mm de altura, con los brazos estirados hacia el frente.

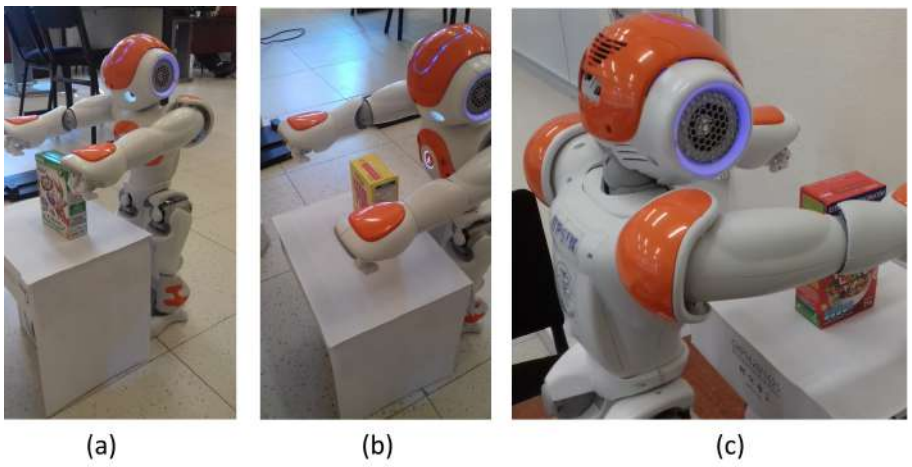


Figura 4.16: Robot NAO en posición cero frente a caja de cereal a manipular, (a) cookie crisp, (b) corn pops y (c) froot loops.

En la Figura 4.16 (a), (b) y (c), se observan tres imágenes donde el robot se encuentra frente a un objeto para comenzar el módulo de manipulación de objetos. Éstas son cajas de cereales colocadas en posición aleatoria dentro del área de trabajo de los brazos del robot.

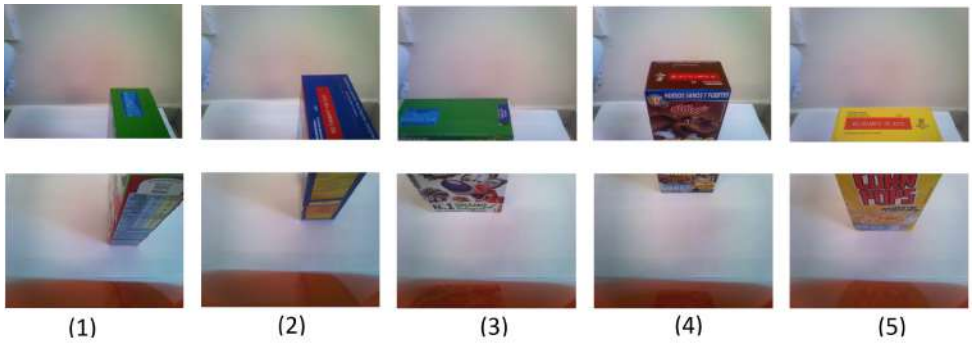


Figura 4.17: Capturas del robot NAO de dos imágenes correspondientes a 5 cinco cajas de cereal diferentes, (1) trix, (2) zucaritas, (3) cookie crisp, (4) choco krispis, (5) corn pops.

Como se ha mencionado en la descripción del módulo de manipulación de objetos, la entrada a éste son dos imágenes capturadas por el robot. Después de girar su cabeza hacia abajo para observar la superficie de la plataforma, estas imágenes se capturan con la cámara superior e inferior. En la Figura 4.17 se observan las dos imágenes correspondientes a 5 cinco cajas de cereal diferentes. Las cajas de cereal (1) y (2) se giraron, mientras que (3), (4) y (5) se encuentran de frente al robot, todas con posición aleatoria en la superficie.

Una vez capturadas las dos imágenes del objeto sobre la plataforma, la siguiente tarea del módulo es la detección del objeto y estimación de su posición en la imagen. Esta tarea se realiza mediante algoritmos de procesamiento de imágenes que incluyen binarización, componentes conectados, etc.

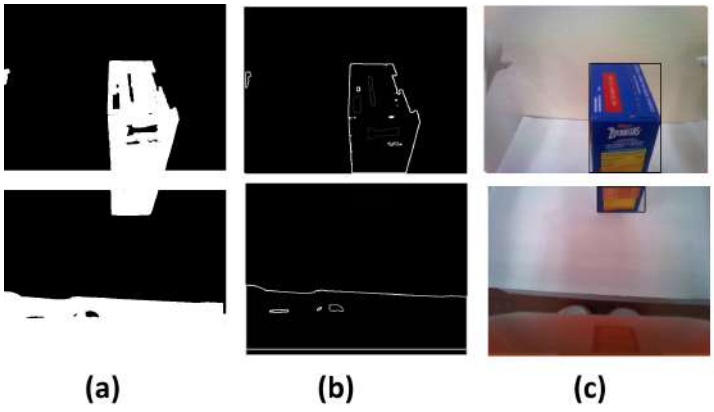


Figura 4.18: Segmentación del objeto, (a) binarización, (b) detección de bordes, (c) Identificación de *bounding box*.

En la Figura 4.18 se presenta el procesamiento de una de las cajas de cereal. En 4.18 (a) se observa la binarización de ambas imágenes. En 4.18 (b) se visualizan los bordes de todos los objetos encontrados, sin embargo, como se observa en la imagen inferior el borde del objeto no

aparece. Por esta razón se utiliza el algoritmo de componentes conectados partiendo de la binarización. Identificando el componente mayor más cercano al centro, inferior o superior, según sea el caso de la imagen. Con el tamaño de la región se procede a construir la *bounding box* del objeto en ambas imágenes.

Figura 4.19: Imágenes donde la detección de *bounding box* no se llevo a cabo correctamente, (1) inferior, (2) superior, (3) superior, (4) superior, (5) superior.

En ocasiones, la segmentación no resulta correcta y no se encuentra objeto. No obstante, solo basta que en una de las dos imágenes se detecte el objeto para que el módulo continué con el cálculo. En la Figura 4.19 se pueden observar algunos ejemplos de esta situación. De las imágenes en (1), la inferior segmenta un pequeño cuadro en medio de la caja de cereal. En la superior, logra segmentar una mayor parte de la caja suficiente para estimar la posición del objeto en la imagen. De las imágenes en (2), la segmentación de la imagen superior no se llevó a cabo correctamente, mientras que la de la inferior detectó un cuadro mediano considerado suficiente. De las imágenes en (3) y (4) en todas se segmentó la caja, sin embargo, las imágenes superiores realizaron segmentaciones erróneas. Como esta última situación es muy común, el módulo de manipulación primero verifica si con la imagen inferior se obtuvieron datos. De lo contrario se verifica la superior, si no hay datos se manda el mensaje de que no es posible la manipulación.

Tabla 4.3: Ángulos en grados encontrados para las posiciones de 5 objetos.

Objeto	**R/LShoulderPitch**	**RShoulderRoll**	**LShoulderRoll**	**RElbowRoll**	**LElbowRoll**
Zucaritas 4.18	35.04	16.53	-15.38	30.44	-31.98
Trix 4.19-1	24.47	13.93	-3.33	36.53	-49.02
Froot Loops 4.19-2	18.08	4.43	9.55	62.85	-76.22
Trix 4.19-3	8.94	1.56	14.55	74.03	-88
Trix 4.19-4	25.53	9.36	10.47	48.16	-67.78

De las cajas de cereal de 4.18 y 4.19 se obtuvieron los ángulos mediante el módulo de manipulación. Los resultados se presentan en la Tabla 4.3 a manera de visualización de los resultados

calculados por el módulo. La caja de cereal se identifica por su nombre y la figura donde se presentan las dos imágenes utilizadas para el cálculo de los ángulos de los manipuladores del robot NAO. Los resultados se demostraron de manera práctica ejecutándolos en el robot.

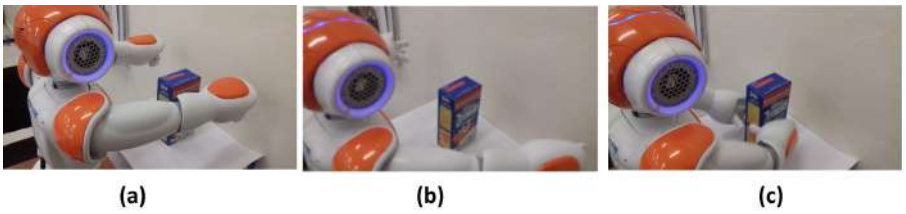


Figura 4.20: Ejecución de la cinemática directa por parte del robot NAO para tomar la caja de Zucaritas.

En la Figura 4.20 se observa la ejecución de la cinemática directa por parte del robot NAO para tomar la caja de Zucaritas. En (a) el robot se encuentra en posición cero, desde donde comienza la ejecución de movimientos. En (b) el robot separa sus brazos para poder tomar el objeto con mayor facilidad. En (c) el robot ya ha realizado la cinemática directa y como se observa, los brazos han llegado a la caja de cereal. Sin embargo, por la configuración de sus dedos no es posible tomar todo el objeto. Para el segundo objeto, Trix (4.19-1), la ejecución de la cinemática directa se observa en la Figura 4.21. De igual manera que para la caja de Zucaritas, en (a) el robot está en posición cero, en (b) el robot abre sus brazos y en (c) el robot ejecuta la cinemática directa para llegar al objeto. En esta ocasión, por la posición de la caja de cereal, el robot logra tomar el objeto con ambas manos.

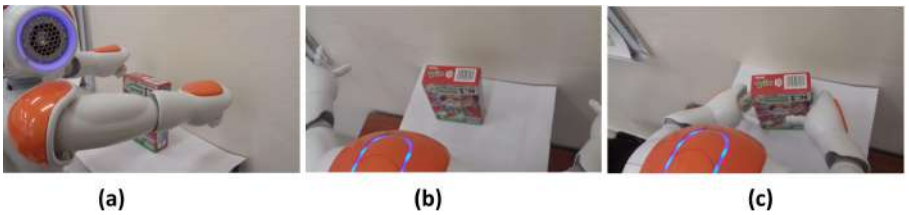


Figura 4.21: Ejecución de la cinemática directa por parte del robot NAO para tomar la caja de Trix (4.19-1).

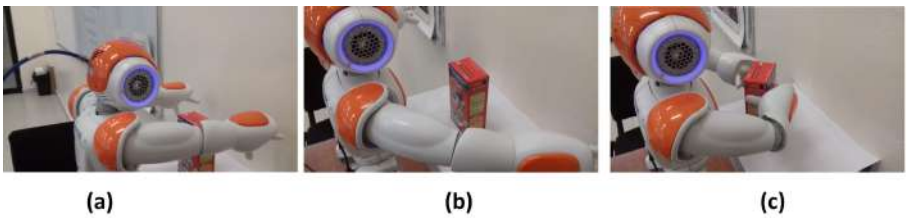


Figura 4.22: Ejecución de la cinemática directa por parte del robot NAO para tomar la caja de Froot Loops (4.19-2).

En la Figura 4.22 se observa la ejecución de la cinemática directa para tomar la caja de Froot Loops. Igual que las anteriores el robot NAO en (a) comienza en posición cero, en (b) abre

sus brazos y en (c) ejecuta la cinemática. El resultado es positivo ya que el robot logra tomar la caja de cereal. En la Figura 4.23 se realiza la ejecución de la cinemática directa y además se realiza un movimiento más para levantar el objeto. En (a) el robot está frente al objeto, (b) el robot ejecuta la cinemática directa y en (c) el robot realiza un movimiento más para levantar la caja de cereal. Los resultados son favorables ya que este se levanta sin problemas. Para la caja de Trix de la Figura (4.19-4) no se realizó la cinemática directa de manera correcta, ya que los ángulos obtenidos hacen que la caja caiga de la plataforma.

Figura 4.23: Ejecución de la cinemática directa por parte del robot NAO para tomar la caja de Trix (4.19-3).

Se realizó un experimento más para demostrar que el robot es capaz de levantar los objetos. En la Figura 4.24 se ejecutó el módulo de reconocimiento de objetos una vez mas para la caja de cereal Froot Loops. En (a) se muestra al robot abriendo los brazos, en (b) se ejecuta la cinemática directa llegando al objeto y en (c) se observa al robot levantando la caja de cereal.

4.24

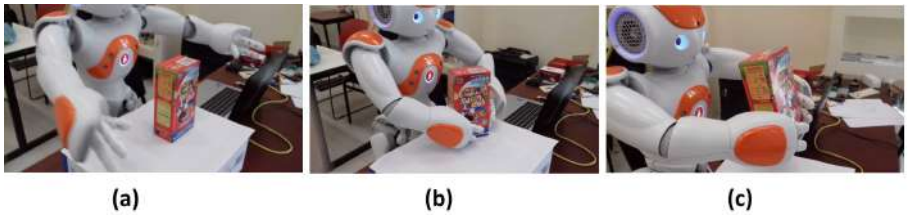


Figura 4.24: Ejecución de la cinemática directa por parte del robot NAO para tomar la caja de Froot Loops.

Los resultados de los experimentos son favorables. Gracias a estos experimentos se afirma que el módulo depende del proceso de segmentación de objetos para detectar el tamaño correcto del objeto. Por otro lado, la mayoría de los experimentos realizados obtuvieron buenos resultados. Se concluye que mientras los objetos no sean claros y se pueda realizar una buena segmentación el módulo no tiene inconvenientes de calcular la cinemática directa correcta.

4.4. Conclusión

El módulo de manipulación de objetos desarrollado en este trabajo le permite al robot humanoide NAO detectar un objeto, estimar su posición 3D y planificar una trayectoria de manipuladores para levantarlo sin conocimiento previo de las propiedades del objeto. Este módulo presenta ciertas restricciones:

- ✓ dependiendo de la plataforma a utilizar se deben realizar los ajustes necesarios
- ✓ es necesario conocer la altura de la superficie donde se encuentran los objetos
- ✓ los objetos deben estar dentro del área de trabajo de los manipuladores del robot
- ✓ los objetos deben tener un tamaño y peso tales que el robot sea capaz de levantarlo y manipularlos
- ✓ la plataforma debe ser de un tono más claro que los objetos para realizar una buena segmentación
- ✓ si el robot no puede alcanzar la posición calculada simplemente no tomará el objeto

Considerando estas restricciones, los resultados obtenidos en los experimentos son aceptables pues el objetivo de levantar los objetos se ha cumplido. Los experimentos realizados con el módulo demuestran que éste presenta un buen desempeño ya que tanto en el experimento en simulación como en la ejecución con el robot real, de un 1 objeto del total no se logró calcular su cinemática adecuadamente.

Capítulo 5

Módulo de Navegación Espacial

Figura 5.1: Tareas que debe cumplir el módulo de Navegación Espacial.

Tal como se mencionó en la sección de *Marco Teórico*, los robots de servicios para hogares, hospitales, restaurantes, etc. deben tomar decisiones complejas, como identificar el medio con el que interactúan, detectar su objetivo (objetos) y cumplir órdenes (reconocimiento, manipulación) etc. Estos robots deben tener la capacidad de ser autónomos. Para que un robot humanoide sea autónomo es necesario que cuente con un sistema de control. Este sistema debe lograr que el robot sea capaz de interactuar con el medio en el que se encuentra para tomar decisiones correctas y cumplir metas concretas. El módulo de Navegación Espacial, ver Figura 5.1, tiene el objetivo de brindarle al sistema modular cierta autonomía y control. Este módulo comprende las siguientes tareas:

1. Construir un mapa bidimensional del lugar donde se encuentra.

2. Estimar su ubicación con respecto al marco de referencia global del mapa construido.

3. Buscar y ubicar el objeto requerido por el usuario.

Para la construcción del mapa bidimensional se utiliza la pose (posición más orientación) del robot combinado con elementos visuales. Como se observa en la Figura 5.2, el robot realizará un recorrido de circuito cerrado alrededor de la habitación de la cual construirá el mapa. Éste considerará el marco de referencia global como el punto del cual partió, e irá capturando

Figura 5.2: Se pretende considerar el marco de referencia global como el punto del cual parte el robot, éste a su vez debe contar con su propio marco de referencia.

imágenes a cada paso que dé. Las imágenes capturadas las asociará a la posición de donde las tomó para realizar la construcción del mapa bidimensional.

La búsqueda de objetos la podrá realizar partiendo desde cualquier punto de la habitación. Esta búsqueda la realiza mediante el uso del enfoque *Next Best View* el cual parte de la idea de posicionar al robot de tal manera que ubique mejor al objeto que busca. Una vez encontrado el objeto, el robot hará uso del *Módulo de Manipulación de Objetos* para tomarlo. Ya que tomó el objeto, el mapa bidimensional de la habitación le permitirá ubicarse para regresar al punto de referencia global simulando la entrega al usuario.

El capítulo está comprendido por la sección 5.1 de Localización Espacial donde se incluye la construcción del mapa bidimensional y la búsqueda de objeto mediante el enfoque NBV. En la sección 5.2, se presentan experimentos y resultados del módulo. Finalmente en la sección 5.3, se plasman las conclusiones del capítulo.

5.1. Localización Espacial

Para que un robot pueda estimar su ubicación en el espacio, se utilizan ciertas herramientas de localización. Estas herramientas pueden ser GPS, brújulas, acelerómetros, giroscopios, encoders e imágenes capturadas por sensores como láser, ultrasónicos y cámaras desde un punto de partida. Para este trabajo se utilizan las mediciones incrementales de los encoders en las articulaciones y la cámara del propio robot para ir almacenando su ubicación con respecto a un marco de referencia global. Ambas herramientas le permitirán al robot la construcción de un mapa bidimensional del lugar en el que está navegando.

5.1.1. Construcción de Mapa Bidimensional

La construcción del mapa bidimensional se lleva a cabo mediante dos herramientas: 1) Odometría y 2) Elementos Visuales. Ambos enfoques van a permitir construir un mapa bidimensional en el plano XY (Figura 5.2) del entorno donde el robot navegue. La idea general es que antes de que el robot comience a buscar los objetos pedidos por el usuario, éste debe conocer el entorno donde está trabajando. Para conocer su entorno, el robot tendrá que realizar un recorrido de circuito cerrado cuadrangular en la habitación, capturando imágenes de lo que tiene alrededor y a su vez almacenando la pose. Se detallará el desarrollo del algoritmo utilizado para la construcción del mapa bidimensional más adelante.

Odometría

Como se mencionó en el capítulo de *Marco Teórico*, la odometría permite estimar la posición relativa de un robot o vehículo en el plano durante la navegación desde su localización inicial. En este trabajo se utiliza para determinar y guardar la ubicación del robot durante su recorrido en la habitación para la construcción del mapa bidimensional.

```
// store Nao's position before the walk
const bool useSensor = false;
const AL::Math::Pose2D worldToRobotInit =
    AL::Math::Pose2D(motionProxy->getRobotPosition(useSensor));

const float x     = 0.3f;
const float y     = 0.1f;
const float theta = AL::Math::PI/2.0f;
motionProxy->moveTo(x, y, theta);
motionProxy->waitUntilMoveIsFinished();

// store Nao's position after the walk
const AL::Math::Pose2D worldToRobotAfter =
    AL::Math::Pose2D(motionProxy->getRobotPosition(useSensor));

// compute Nao's displacement
AL::Math::Pose2D robotMove =

AL::Math::pose2DInverse(worldToRobotInit)*worldToRobotAfter;

 // return an angle between ]-PI, PI]
 robotMove.theta = AL::Math::modulo2PI(robotMove.theta);
 std::cout << "Robot Move: " << robotMove << std::endl;
```

Figura 5.3: Código de odometría inercial con robot NAO, funciones de Aldebaran.

El robot NAO cuenta con funciones que apoyan en la resolución de varios problemas, uno de ellos es la odometría. Por razones prácticas se decide utilizar estas funciones. En la Figura 5.3 se observa un pequeño código donde se hace uso de las funciones de odometría inercial de Aldebaran [18]. En esta implementación se inicializa la pose 2D del robot mediante *pose2D* (X , Y, theta) con valores explícitos recuperados con *getRobotPosition(useSensors)*. Los valores con los que se inicializa la pose se recuperan desde los encoders magnéticos rotatorios (MRE) de las articulaciones. Éstos estas contenidos en un vector con la posición absoluta del robot en el mundo (X, Y en metros) y un ángulo *theta Wz* en radianes (-pi,pi). Cada que el robot se enciende almacena una posición absoluta en el mundo.

Después de guardar la posición inicial, se le indica al robot que camine cierta distancia. Al terminar de caminar se vuelve a almacenar la pose 2D. Posteriormente se calcula el desplazamiento que realizó el robot y el ángulo. Utilizando la función *pose2DInverse* calcula la pose 2D entre la actual y la siguiente. El ángulo se obtiene mediante *modulo2PI* el cual regresa un ángulo entre -pi y pi.

Haciendo uso de las funciones anteriores, se propone el algoritmo para la implementación de la odometría:

- ✓ Lectura de datos odométricos del robot humanoide NAO.
- ✓ Control del caminado del robot.
- ✓ Procesamiento de datos odométricos.

Considerando un panorama más amplio, se establece el siguiente algoritmo generalizado:

1. Captura de posición del robot con respecto al mundo antes de comenzar a caminar.
2. Detección de inicio del caminado del robot.
3. Simultáneamente, inicio de la recolección de datos odométricos.
4. Procesamiento y acumulación de datos odométricos.
5. Detección de finalización del caminado del robot. Si no es así se repiten pasos 3 y 4.
6. Cálculo de la distancia que recorre el robot.
7. Almacenamiento de la distancia y posición del robot para la construcción del mapa bidimensional.

Elementos Visuales

El uso de elementos visuales se basa en analizar y crear una base de detalles existentes en el entorno tomando en cuenta la pose del robot en la cual se captura la imagen. Esta base de datos se crea de manera similar a la base de datos de objetos en el módulo de reconocimiento de objetos. En lugar de diferentes imágenes del objeto a reconocer como entrada del módulo, aquí son necesarias diferentes vistas de la habitación. Se considera que los elementos visuales más significativos para poder realizar una representación del entorno son los que se encuentran en las paredes o cerca a ellas. Con los elementos visuales y la ubicación estimada mediante odometría se construirá el mapa bidimensional de la habitación que recorra el robot.

Por ejemplo, en la Figura 5.4, se muestra una habitación simulada en Webots. En esta habitación se observan diversos objetos que podrían encontrarse en cualquier casa: cuadros, mesas, estantes, plantas, ventanas, puertas, etc. Estos objetos tienen una posición determinada y es sabido que no se moverán de su lugar. El robot tiene que recorrer esta habitación en un circuito cerrado, de preferencia cuadrangular, tomar imágenes y almacenar la pose estimada de donde se ha tomado la captura.

Figura 5.4: Simulación de una habitación en Webots.

Figura 5.5: Captura realizadas por el robot NAO en habitación simulada en Webots. Las capturas se almacenan junto con la pose de donde las tomó.

El robot debe enfocar la captura de imágenes a la pared más cercana por donde él vaya caminando. En la Figura 5.5, se observan tres capturas realizadas por el robot en diferentes puntos. Mientras realiza el recorrido, tomará una captura cada ciertos pasos dependiendo el número de imágenes que el usuario indique capturar de la habitación. Por ejemplo, si se requieren 20 imágenes en una habitación de 4 metros por pared, se tomará 1 imagen cada 20 cm. Además del número de capturas y dimensión de la pared de la habitación, también se puede decidir cuantas veces se realizará el recorrido. Entre más recorridos se realicen mejor se construirá la habitación.

Una vez terminado el o los recorridos, con la información almacenada por parte del robot se procede a la construcción del mapa bidimensional. Las imágenes capturadas contendrán objetos o partes de objetos de los cuales se tiene que adquirir cierta información. Se utiliza el módulo de reconocimiento de objetos, descrito en el capítulo 3, para la extracción de esta información. Utilizando este módulo se obtienen los descriptores de la imagen para aprender la información y asociarla con la pose del robot al momento de capturarla. Esta información se fusiona en una representación bidimensional la cual será el mapa de la habitación. Antes de iniciar cualquier recorrido por la habitación, el robot debe detectar la pared más cercana para saber hacia donde tiene que girar su cabeza. Mientras realiza su recorrido irá tomando imágenes solo girando su

cabeza hacia la pared detectada. Al momento de que el robot deba girar para recorrer la siguiente pared, solo basta con girar en la dirección opuesta a donde detectó la pared para continuar con el recorrido en la habitación. La detección de la pared se lleva a cabo visualmente. Antes de comenzar a caminar, el robot debe ser colocado de manera paralela a cualquier pared y en una esquina de la habitación.

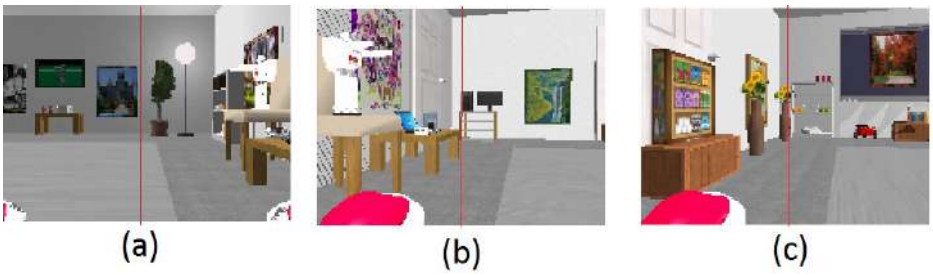


Figura 5.6: Imágenes tomadas por robot NAO antes de iniciar recorrido en la habitación para detectar la pared divididas en dos, pared más cercana: (a) lado derecho, (b) lado izquierdo, (c) lado izquierdo.

Entonces, éste capturará una imagen viendo hacia al frente para analizarla. La imagen se analiza dividiéndola en dos, por ejemplo, en la Figura 5.6, se observan tres capturas de la habitación en diferentes posiciones. En la imagen (a) la pared más cercana se encuentra a la izquierda, mientras que en las imágenes (b) y (c) están a la derecha. Cada una de las imágenes se dividen en dos de manera vertical y se obtienen los *keypoints* de cada uno de los lados. Estos *keypoint* se obtienen con el método A-KAZE utilizado en el módulo de reconocimiento de objetos. En la imagen que obtenga más *keypoints* es donde se encontrará la pared, partiendo de la restricción de que la habitación se encuentra libre de obstáculos.

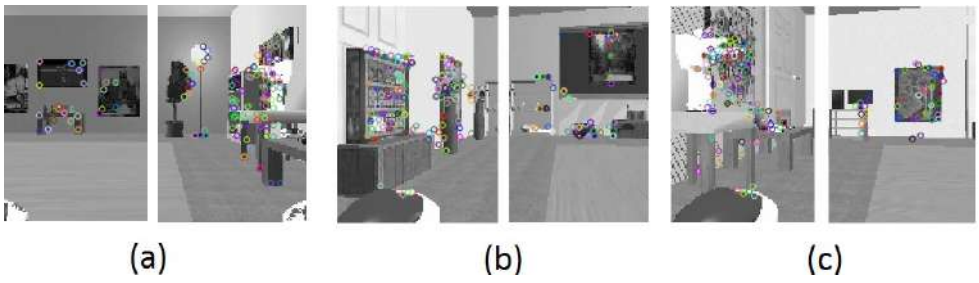


Figura 5.7: Detección de pared mediante descriptores.

En la Figura 5.7 se muestran los resultados de la evaluación de cada imagen. En éstas se observan los *keypoints* señalados con pequeños círculos de colores. En (a) la imagen con más *keypoints* es la derecha con 108 contra 36, en (b) la izquierda con 128 contra 50 y en (c) la izquierda con 119 contra 53. Entonces el robot gira su cabeza hacia esa dirección para ir aprendiendo la habitación.

5.1.2. Algoritmo para la construcción del mapa bidimensional

La construcción del mapa bidimensional se lleva a cabo de la siguiente manera. Primero, el Algoritmo 9, tiene la tarea de ejecutar un recorrido en circuito cerrado alrededor de una habitación cuadrada por parte del robot. Éste realiza el circuito en la habitación capturando y guardando imágenes con su respectiva poses. Es necesario conocer, $\boldsymbol{d}$, la dimensión de una pared a recorrer de la habitación y $\boldsymbol{p}$ el tamaño de paso al caminar. Otro parámetro que se puede elegir es cuantas veces realizará el recorrido $\boldsymbol{n}$. Si se realizan más de un recorrido el robot reforzará el aprendizaje de la habitación. Al inicio del algoritmo, el robot realiza una primer captura, *TakePicture()*, para detectar la pared más cercana *DetectNearestWall()*. De esta manera, se obtiene el ángulo, *AngleYaw*, al cual girará mientras realiza su recorrido.

Algoritmo 9: Módulo de Navegación. *Ejecución de un circuito cerrado.*

Datos: $\boldsymbol{d}$ dimensiones de la habitación, $\boldsymbol{p}$ tamaño de paso del robot al caminar, $\boldsymbol{n}$ número de recorridos
Resultado: ***data*** imágenes y poses

```
1  picture =TakePicture()
2  AngleYaw=DetectNearestWall(picture)
3  TurnHead(AngleYaw)
4  O_w=CurrentPose()
5  TotalDistanceWalked=0
6  para j = 1 a n hacer
7      mientras TotalDistanceWalked ≠ d × 4 hacer
8          DistanceWalked=0
9          mientras DistanceWalked ≠ d hacer
10             Walk(p)
11             RP=CurrentPose()
12             picture =TakePicture()
13             data = (P, I)
14             DistanceWalked=DistanceWalked+p
15         TurnBody(-AngleYaw)
16         TotalDistanceWalked=TotalDistanceWalked+DistanceWalked
```

Antes de comenzar a caminar, el robot guarda su pose actual mediante odometría *CurrentPose()*, como punto de referencia global de la habitación O_w. Entonces se comienza el ciclo de trabajo por número de recorridos $\boldsymbol{n}$. Después, se inicializa la variable, *TotalDistanceWalked*, que será el indicador de cuanto lleva recorrido el robot de la distancia total que debe recorrer en toda la habitación. Esta variable se verifica mediante un ciclo de trabajo que no se detendrá mientras la variable no sea igual a $d \times 4$ (distancia de una pared por las cuatro que conforman la habitación). Dentro de este ciclo de trabajo se encuentra otro ciclo de trabajo que verifica la variable *DistanceWalked*, (distancia que debe recorrer por pared). Este ciclo de trabajo no se detendrá mientras la variable sea igual a la distancia a recorrer, $\boldsymbol{d}$, por pared de la habitación. Dentro de este ciclo de trabajo se ejecuta la odometría inercial y la captura de imagen (**P**, **I**), cada que el robot camina $\boldsymbol{p}$ distancia por paso. Una vez terminado el ciclo, el robot gira su cuerpo, TurnBody(**-AngleYaw**), hacia el ángulo opuesto al cual giró su cabeza para seguir

recorriendo la habitación.

Después de realizado el circuito y almacenada la base de datos de la habitación, se ejecuta el Algoritmo 10. En este algoritmo, se hace uso del módulo de reconocimiento de objetos para aprender una nueva base de datos con la información de las capturas y las poses (**P**, **I**). En éste se le extraen todos los *keypoint*, se construyen los histogramas por imágenes y se entrena una red neuronal con GCS. De la red se extraen las clases por imagen obtenidas, classes($\boldsymbol{N}$). Una vez obtenidas las clases, se verifica si una clase formada por varias imágenes tiene mas de una pose asociada. Si es así se lleva a cabo un promedio entre ellas y se almacena.

Algoritmo 10: Módulo de Navegación. *Construcción de mapa bidimensional*

Datos: $\boldsymbol{I}$ imágenes, $\boldsymbol{P}$ poses.
Resultado: ***classes(N,P)*** clases del mapa bidimensional.

```
para i ← 1 to I hacer
    keypoints =AKAZE(I(i))
    H(i)=BuildHistos(keypoint)
ANN_trained=GCS(H)
classes(N)=ANN_trained
para i ← 1 to N hacer
    NewPose(i) = (1/n) Σ_{j=1}^{n} P(classes(i))
```

Posteriormente se utiliza el Algoritmo 11 para utilizar el mapa. El módulo recibe una o más imágenes, de éstas se extraen los *keypoints* y se construyen los histogramas de las imágenes. Los histogramas se envían a evaluar con la red neuronal entrenada para obtener las neuronas a las que pertenece. Una vez conocidas las clases se obtienen las poses a las cuales corresponde y se regresa la posición bidimensional en el mapa.

Algoritmo 11: Módulo de Navegación. *Utilización del mapa bidimensional*

Datos: $\boldsymbol{I}$ imágenes
Resultado: ***class(I)*** clases de objetos, ***Pose*** pose

```
keypoints =AKAZE(I)
H(I)=BuildHistos(keypoint)
classes(I)=ANN_trained
Pose=(1/n) Σ_{j=1}^{n} NewPose(I)(classes(i))
```

El algoritmo cuenta con ciertas restricciones como conocer la dimensión de la pared de la habitación. De esta manera se calcula la distancia total que va a recorrer el robot alrededor de la habitación. En la habitación no deben de haber obstáculos pues en este trabajo no se está atacando el problema de evasión de obstáculos. Otra restricción es que si algunos elementos en la habitación se han movido de lugar el robot debe volver a construir su mapa de navegación. Los objetos en el centro de la habitación serán desconocidos hasta este punto. No es necesario conocerlos ya que el robot navegará en busca de algún objeto requerido por el usuario mediante el enfoque *Next Best View* descrito en la siguiente sección. El mapa bidimensional se utiliza para que el robot conozca su posición en la habitación una vez que tome el objeto y deba entregarlo al usuario en el punto de referencia global.

5.1.3. Búsqueda de Objeto

Considerando que en este trabajo se deja de lado la evasión de obstáculos. Se establece un entorno semi-estructurado donde las dimensiones de la habitación son conocidas y los objetos a buscar estarán en el centro de la habitación sobre una mesa en posición aleatoria. Gracias al mapa bidimensional, el robot puede comenzar su búsqueda desde cualquier posición en la habitación. Es necesario solo tomar una o más imágenes de la o las paredes más cercanas para conocer su ubicación en la habitación y encaminarse hacia el centro de ésta.

Sin embargo, no es suficiente con acercarse a los objetos para encontrar el requerido por el usuario. Es necesario utilizar un algoritmo de búsqueda de objeto que facilite la manipulación acercándose lo suficiente para poder tomarlo y llevarlo consigo. En este trabajo se propone el uso del enfoque *Next Best View* que le permitirá al robot acercarse al objeto buscando la mejor posición para tomarlo. Entonces, cuando el usuario le indique al robot que busque un objeto el robot comenzará a navegar en la habitación basado en NBV.

Next Best View

Se considera una variante del enfoque *Next Best View* (NBV) para la tarea de búsqueda de objeto. Este enfoque trata con la planificación de movimientos requerida para llegar lo más cerca posible a la mesa que contiene los objetos a buscar. En un contexto de robótica móvil, NBV se dedica a determinar la acción de sensado futura más apropiada a ejecutar y alcanzar la mejor vista de un objeto.

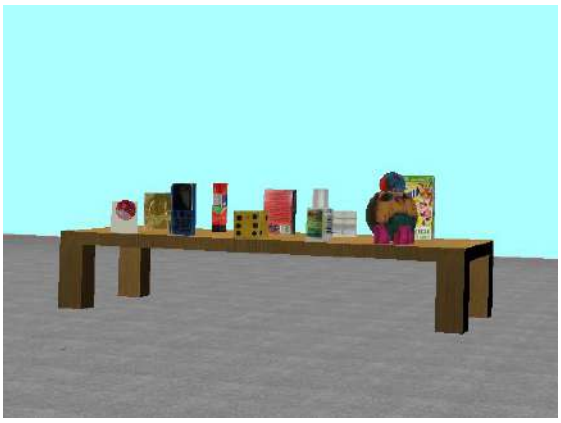


Figura 5.8: Captura de la habitación donde aparece mesa con objetos.

Entonces cuando al robot se le indique la búsqueda de un objeto éste comenzará a navegar en la habitación. Ya que el robot puede conocer su ubicación en la habitación, puede evitar las paredes e ir directamente a buscar por el centro de ésta. Una vez que el robot ubique la dirección de las paredes y se aleje de ellas, la búsqueda de objeto da inicio. El robot va a realizar captura de imágenes constantes para poder determinar hacia donde va a caminar para acercarse al objeto. En la Figura 5.8 se observa una primer captura de la habitación en la cual el robot visualiza la mesa con objetos donde buscará el objeto requerido por el usuario.

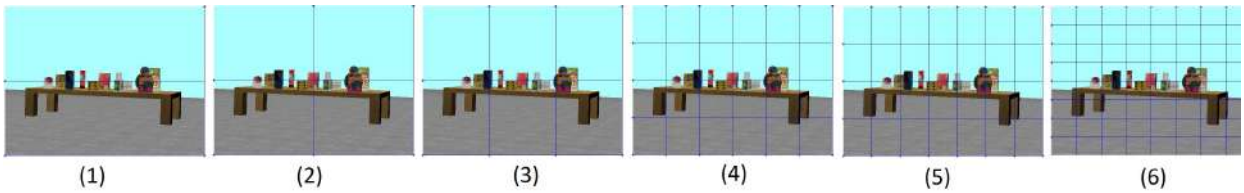


Figura 5.9: Captura de la habitación con división progresiva, (1) 2 imágenes, (2) 4 imágenes, (3) 6 imágenes, (4) 24 imágenes, (5) 28 imágenes, (6) 64 imágenes.

El análisis de los objetos en la imagen se lleva a cabo mediante una división progresiva de ésta. La imagen se divide en 10^{10} partes, cada una de éstas se almacena como imagen junto con sus coordenadas en la imagen original. Esta división se realiza recorriendo la imagen tanto horizontal como verticalmente. Por ejemplo, en la Figura 5.9, se presentan algunas de las divisiones realizadas a la imagen mostrada con anterioridad. En la primer imagen el recorrido es horizontal y se divide en dos. En la segunda imagen se divide la anterior en 2 partes para formar 4 imágenes. En la tercer imagen se divide la primera en 3 para formar 6. La cuarta imagen está dividida en 4 de manera horizontal y 6 en vertical. La quinta imagen también está dividida en 4 horizontalmente pero en 7 vertical. La sexta imagen se dividió en 8 por 8 y se formaron 64 imágenes. Cada una de las imágenes se evalúa mediante el *Módulo de Reconocimiento de Objetos* para identificar al objeto que se está buscando. Por ejemplo, si se quiere buscar la caja de cereal Nesquik, el módulo verificará por todas las imágenes en las que aparece ese objeto.

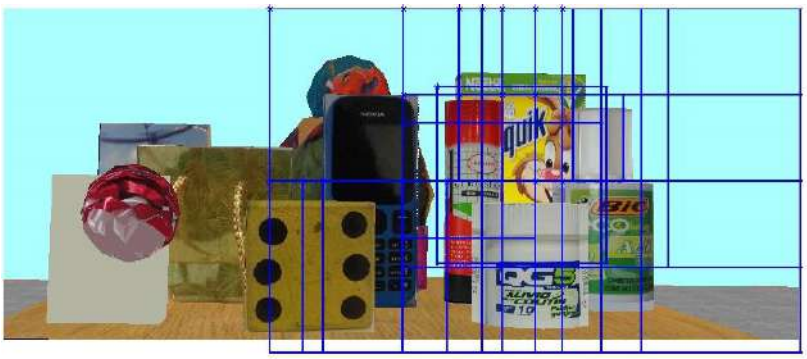


Figura 5.10: Identificación de las coordenadas donde aparece la caja de cereal.

Una vez identificadas todas las imágenes donde aparece el objeto, se procede a localizarlas en la imagen original mediante las coordenadas que se han almacenado. En la Figura 5.10 las coordenadas de las imágenes se han marcado para indicar en qué lugar de la imagen se encontró el objeto. En ésta se observa cómo las coordenadas se han marcado cerca del objeto, con algunos falsos positivos despreciables. Entonces, se obtiene la tasa de positivos hacia donde se identifica el objeto para que el robot avance. Como el robot debe centrar el objeto para poderlo manipular, éste se moverá hacia donde tienda el objeto a ubicarse. Como no se conoce el tamaño de los objetos ni de la mesa se establece un umbral mínimo de positivos a alcanzar que indicará que el objeto aun está muy lejos como para poder ser identificado. Cuando cruce ese umbral dejará de avanzar hacia la mesa y comenzará a rodearla. En este punto el robot no se acercará más ya que puede chocar con la mesa y caer, así que, solo avanzará hacia los lados sabiendo que

el objeto debe estar centrado y que debe rodear la mesa que tiene enfrente. Cuando la tasa de positivos sea mayor a cierto umbral, entonces se acercará un poco a la mesa para poder ejecutar el *Módulo de Manipulación de Objetos.*

En el Algoritmo 12, se describe en grandes rasgos las tareas que apoyan a llevar a cabo la búsqueda de objeto. La entrada del sistema es la etiqueta o nombre del objeto a buscar: *label*. Considerando como restricción que la posición del objeto a buscar es en algún lugar en el centro de la habitación donde el único obstáculo es la mesa, el robot comenzará a avanzar hacia el centro de la habitación *MoveRobot(ahead)*. La búsqueda del objeto se lleva a cabo mediante un análisis cíclico de imágenes. Después de avanzar hacia enfrente, el robot captura una imagen, corta las imágenes y las analiza en busca del objeto con el *Módulo de reconocimiento de objetos*. De las imágenes en donde se reconoció el objeto se almacenan las coordenadas y se verifica si la tasa de positivos cumple el umbral mínimo, $I_{object}(x, y) > MinimumThreshold$. Si la cantidad es menor, el robot continuará realizando ese ciclo para acercarse a la mesa. Si la cantidad es mayor, se inicia un nuevo ciclo donde el robot rodeará la mesa. Este ciclo es igual al anterior con la diferencia de que el robot se moverá hacia los lados y verificará la tasa de positivos contra un umbral máximo. Este umbral permite determinar si el objeto buscado se encuentra en la mejor posición para ejecutar el *Módulo de Manipulación de Objetos.*

Algoritmo 12: Módulo de Navegación. *Next Best View, análisis cíclico de imágenes*

Datos: ***label*** etiqueta del objeto
Resultado: **FLAG**=TRUE objeto encontrado

```
1  FLAG=FALSE
2  ObjectFound=FALSE
3  mientras ObjectFound==FALSE hacer
4      MoveRobot(ahead)
5      image=CaptureImage()
6      I(10^10)=CutImage(image)
7      class=ObjectRecognitioModule(I(10^10))
8      si class==label entonces
9          I_object(x,y)=I(class)
10         si I_object(x,y) > MinimumThreshold entonces
11             ObjectFound=TRUE
12 mientras FLAG==FALSE hacer
13     side=NBV(I_object(x,y))
14     MoveRobot(side)
15     image=CaptureImage()
16     I(10^10)=CutImage(image)
17     class=ObjectRecognitioModule(I(10^10))
18     si class==label entonces
19         I_object(x,y)=I(class)
20         si I_object(x,y) > MaxiThreshold entonces
21             FLAG=TRUE
```

5.2. Experimentos y resultados

Los experimentos de esta sección se dividieron en tres partes: (1) construcción de mapa bidimensional, (2) búsqueda de objetos, (3) Ubicación en el mapa bidimensional. En un ambiente semiestructurado libre de obstáculos, donde las dimensiones de una habitación son conocidas un robot debe navegar para encontrar ciertos objetos. Los objetos se han colocado de manera aleatoria en el centro de la habitación sobre una mesa. En este punto el robot ya ha aprendido y etiquetado 10 objetos. Para iniciar una búsqueda de objeto, primero se necesita realizar la construcción de el mapa bidimensional de la habitación. Este mapa ayuda al robot a conocer en que parte de la habitación se encuentra para poder ir a buscar un objeto o regresar al punto del cual partió. Los experimentos se han realizado mediante el simulador Webots y el robot virtual NAO.

5.2.1. Construcción del mapa bidimiensional

Se construyó la habitación simulada en Webots mostrada en la Figura 5.4, de 6×6 metros. En esta habitación se colocaron diversos objetos: sillas, mesas, retratos, etc. Se utilizó un robot NAO virtual para la construcción del mapa bidimensional de esa habitación. El robot inició su recorrido desde la esquina inferior izquierda de la imagen quedando está como posición global 0 de la habitación. El recorrido lo realizó en circuito cerrado cuadrangular girando su cabeza hacia la pared para poder capturar las imágenes mientras avanzaba. Para mostrar los resultados de la construcción del mapa bidimensional de una manera sencilla, las paredes se han enumerado de 1 al 4 en contra de las manecillas del reloj.

Tabla 5.1: Parámetros principales de los experimentos: Construcción del mapa bidimensional.

Circuito	**Imágenes**	**Pared1**	**Pared2**	**Pared3**	**Pared4**
1	89	20	21	31	17
2	75	18	20	24	13

El robot realizó dos circuitos cerrados de 4×4 alrededor de la habitación en contra de las manecillas del reloj tomando imágenes y guardando su relación espacial. El número de capturas que realizó por pared se plasman en la Tabla 5.1. Se almacenaron un total de 164 imágenes y poses utilizada para la construcción del mapa bidimensional.

Tabla 5.2: Parámetros del módulo de reconocimiento de objetos para la construcción del mapa bidimensional.

Experimento	**Entrenamiento**	**Neuronas**	**Epocas**	**Tiempo (seg)**
1	164	100	100	4.063
2	164	200	200	14.287
3	164	300	300	33.347

Ya que se utiliza el *Módulo de Reconocimiento de Objetos*, los parámetros correspondientes a los experimentos para este módulo se presentan en la Tabla 5.2. Se realizaron tres construcciones de mapa bidimensional, con 100, 200 y 300 neuronas. La idea es observar el desempeño del módulo para la construcción de un mapa bidimensional relacionando lo que el robot observó en su recorrido por la habitación. Los tiempos obtenidos de los entrenamientos son relativamente pequeños ya que permanecen por debajo del minuto.

Después del entrenamiento se obtuvieron mapas bidimensionales con 72, 111 y 132 poses, en las Figura 5.11, 5.12 y 5.13 se presentan los mapas construidos. Para verificar que tan buena fue la construcción del mapa basta con observar que las neuronas quedaron bien distribuidas a lo largo del éste. Cada neurona tiene una poses asociada las cuales son los puntos observados en cada uno de los mapas, es fácil deducir que mientras se incrementa el número de neuronas la distribución de las poses mejora. Es importante mencionar que las poses fueron homogeneizadas en la coordenada que permanecía constante al realizar el recorrido para mostrar una distribución limpia.

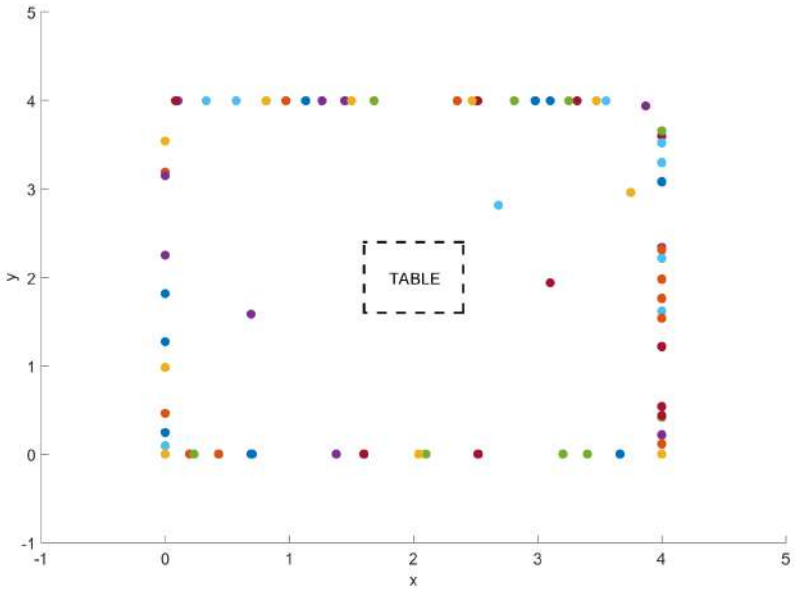


Figura 5.11: Mapa bidimensional experimento 1, distribución de las neuronas por pose en la habitación.

En el primer mapa construido con 100 neuronas la distribución no es muy buena ya que se observan algunas poses dentro del área donde el robot no ha recorrido, además, en algunas partes hay amontonamiento de éstas. En el mapa construido con 200 neuronas la distribución mejor considerablemente en comparación con el mapa anterior. Sin embargo, en este mapa aun se observan algunos amontonamientos de poses. El mapa construido con 300 neuronas tiene una mejor distribución, se cubren más espacios y aunque aun se observan algunas poses sobre otras estas son mínimas. Gracias al mapa bidimensional el robot conocerá donde están las paredes para no ir hacia ellas mientras realiza una búsqueda de objeto. El mapa también apoya al robot al momento de regresar al punto del cual partió considerado la posición global O para simular

que entrega el objeto requerido por el usuario.

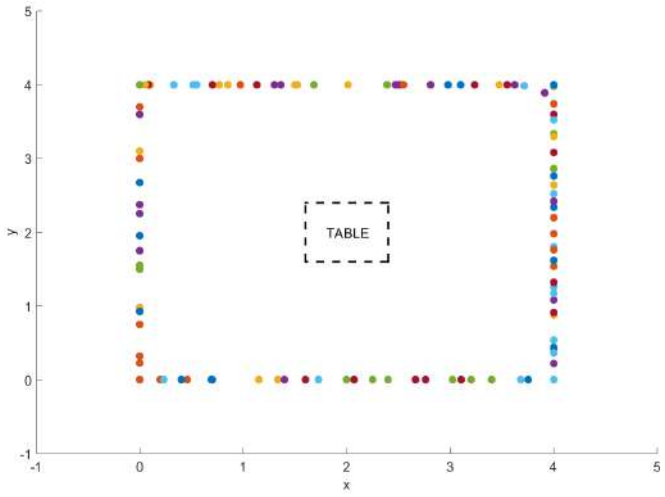


Figura 5.12: Mapa bidimensional experimento 2, distribución de las neuronas por pose en la habitación.

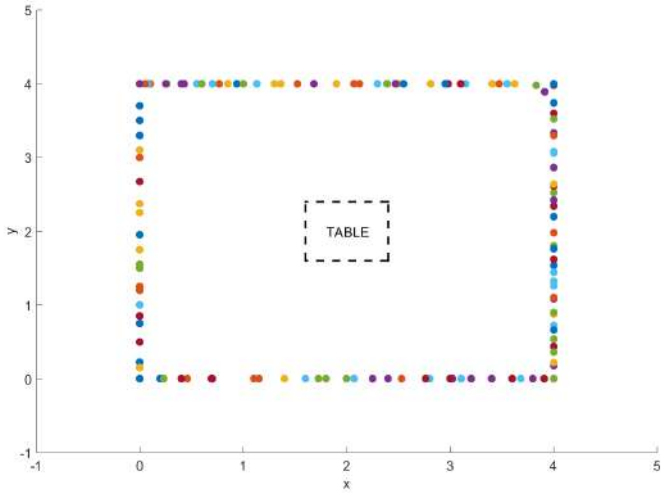


Figura 5.13: Mapa bidimensional experimento 3, distribución de las neuronas por pose en la habitación.

5.2.2. Búsqueda de Objeto

Para la ejecución de una búsqueda de objeto se construyeron 10 objetos simulados en el entorno Webots, mostrados en la Figura 5.14. Los objetos se han colocado de manera aleatoria sobre una mesa en el centro de la habitación. En esta sección se da por hecho que el robot ya ha aprendido los objetos con anterioridad, está listo para iniciar una búsqueda de objeto. Se han propuesto la búsqueda de cada uno de los objetos: (1) corrector líquido, (2) caja de cereal, (3) tortuga, (4) libro, (5) resistol, (6) celular, (7) medicina, (8) bolsa, (9) dado, (10) regalo. Se busca identificar a cada uno de los objetos entre cuatro imágenes mediante el método NBV propuesto en este capítulo.

Figura 5.14: 10 objetos simulados en Webots.

El robot NAO virtual tomó las 4 capturas de la Figura 5.15, éstas son de diferentes posiciones alrededor de la mesa que contiene a los objetos. De estas capturas se procede a identificar a los objetos eligiendo una imagen por objeto. La imagen elegida por objeto se señala en la Tabla 5.3, donde también se muestran los resultados obtenidos del análisis. Estos resultados comprenden tasa de verdaderos positivos (TP), tasa de falsos positivos (FP) y porcentaje total obtenido. La tasa TP al conteo total de veces acertadas donde el objeto aparece, la tasa FP es el conteo total de veces que confundió al objeto y el porcentaje es el resultado obtenido considerando los anteriores.

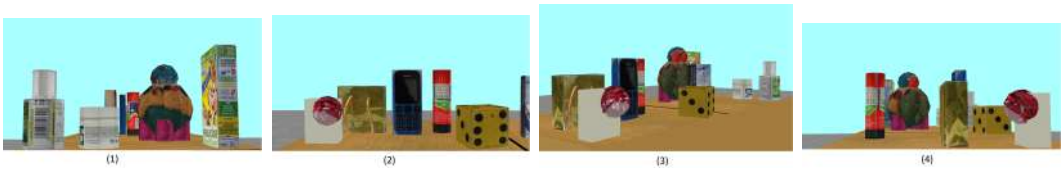


Figura 5.15: Capturas de los objetos sobre la mesa realizadas por el robot NAO virtual.

Los resultados del análisis para cada objeto se muestran en la Figura 5.16. Según el conteo de verdaderos positivos, los primeros 5 objetos ya pudieran ser tomados, solo es cuestión de centrarlos en la imagen para que se encuentren dentro del área de trabajo de los manipuladores del robot. Para los objetos 6, 7 y 8 es necesario buscar una mejor vista que pueda contener más verdaderos positivos. Mientras que los objeto 9 y 10 han sido confundidos y es muy probable que no se encuentre una mejor vista. Este experimento es útil ya que en base a los resultados el robot decidirá hacia donde moverse para poder llegar al objeto requerido. Los resultados muestran un desempeño aceptable con un porcentaje de reconocimiento del 68.19 %. Se considera

Tabla 5.3: Tasa de positivos por objeto identificados en las imágenes.

	Imagen	Tasa (TP)	Tasa (FP)	Porcentaje (%)
1	1	49	16	75.38
2	1	48	24	66.67
3	1	80	28	74.07
4	3	33	14	70.21
5	4	26	6	81.25
6	2	5	0	100
7	3	13	6	68.42
8	4	1	0	100
9	2	5	54	8.47
10	3	6	10	37.5

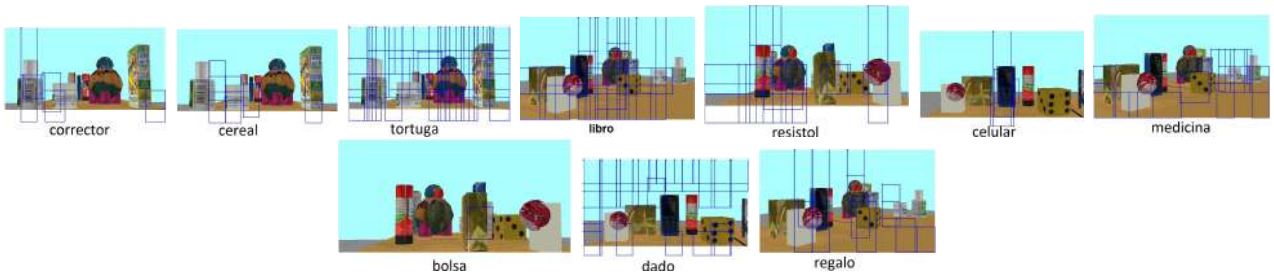


Figura 5.16: Detección de los 10 objetos mediante NBV en las capturas realizadas por el robot NAO virtual.

que el porcentaje ha salido bajo por que las imágenes en simulación tienden a perder calidad y si el aprendizaje se realizó con imágenes de mejor calidad los objetos se confundirán. Los resultados demuestran que el método NBV propuesto en este capítulo cuenta con buenas bases para ejecutar una búsqueda de objeto. Para poder visualizar mejor el uso de este método es necesario realizar la implementación con un robot real.

5.2.3. Ubicación en el mapa bidimensional

La idea del uso del mapa bidimensional es que una vez que el robot haya tomado el objeto requerido por el usuario, éste debe ir a entregarlo en la posición global 0 del mapa. Con el mapa construido el robot es capaz de ubicarse en la habitación con una o dos imágenes de las paredes más cercanas. Se realizaron 4 experimentos, en la Tabla 5.4 se encuentran los parámetros que incluyen número de experimento, mapa bidimensional construido en la sección anterior y la posición real que se pretende calcular (x,y) en metros.

Para la evaluación de la precisión de la construcción de los mapas bidimensionales, el robot virtual capturó 2 imágenes 5 veces desde las 4 posiciones a evaluar de las paredes más cercanas en diferente perspectiva. Ejemplos de las capturas realizadas por el robot se observan en la Figura 5.17, para cada posición se capturaron dos imágenes correspondientes a las paredes más

Tabla 5.4: Parámetros principales de los experimentos para la ubicación en el mapa.

No.	Mapa	(x,y) m
1	1	(3.5,0.5)
2	1	(0,0)
3	2	(0.5,3.5)
4	3	(4,4)

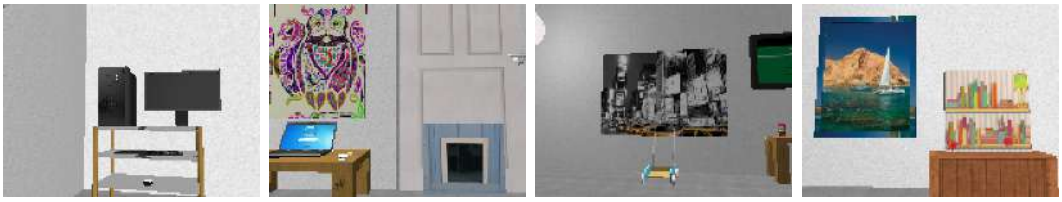


Figura 5.17: Ejemplos de capturas realizadas por el robot NAO virtual.

cercanas. Las dos primeras imágenes pertenecen a la posición (0,0), mientras que las otras dos pertenecen a la posición (4,4) de la habitación. Los resultados se muestran en la Tabla 5.5, para los cuatro experimentos de las imágenes capturadas se obtuvieron sus poses. Las poses obtenidas son cercanas a las reales, algunas llegan a ser casi precisas. Los resultados obtenidos arrojaron la siguiente precisión: primer experimento $\pm(0{,}12, 1{,}2)$, segundo experimento $\pm(0{,}3, 0{,}16)$, tercer experimento $\pm(0{,}42, 1{,}42)$ y cuarto experimento $\pm(0{,}06, 0{,}1)$. Como se esperaba, el tercer mapa bidimensional es el que tiene mejor precisión. Si se desea tener más precisión en la construcción del mapa bidimensional es necesario tomar más capturas mientras se realiza el aprendizaje de la habitación y considerar un número elevado de neuronas para entrenar ala red neuronal.

Tabla 5.5: Resultado de las evaluaciones de poses para cada experimento.

	1	2	3	4	5
1	(3.7,1.7)	(3.6,1.6)	(3.7,1.7)	(3.8,1.8)	(3.3,1.7)
2	(0.1,0.2)	(0.2,0.2)	(0.3,0.1)	(0.3,0.0)	(0.3,0.2)
3	(0.0,2.0)	(0.0,2.0)	(0.2,2.2)	(0.0,2.0)	(0.2,2.2)
4	(4.0,3.8)	(3.9,3.9)	(4,4)	(3.9,3.9)	(3.9,3.9)

5.3. Conclusión

En este capítulo se ha presentado el desarrollo del *Módulo de Navegación Espacial*. Este módulo comprende la localización espacial mediante la construcción de un mapa bidimensional con odometría y elementos visuales. La construcción del mapa bidimensional se realiza haciendo uso del *Módulo de Reconocimiento de Objetos* del capítulo 3 de este trabajo de tesis. Este módulo se utiliza para aprender la habitación y asociarle una pose a cada neurona que comprende la red entrenada para la representación del mapa bidimensional. El *Módulo de Navegación*

Espacial también comprende la búsqueda de objetos ya que se utiliza el enfoque NBV para navegar por una habitación analizando imágenes y tomando decisiones de lo observado en ella.

Los experimentos plasmados en esta sección se realizaron mediante simulación en Webots y con un robot NAO virtual. Al trabajar con imágenes provenientes de un robot virtual y de un ambiente virtual, estas disminuyen considerablemente su calidad. Sin embargo, los resultados son aceptables ya que se logró la construcción de un mapa bidimensional de la habitación y se realizaron experimento de ubicación con una precisión de hasta $\pm(0{,}06, 0{,}1)$. De igual manera se realizaron búsqueda de objetos con un porcentaje de 68.19 %, se considera que estos mejoran en un ambiente real y con una plataforma robótica real ya que la calidad de las imágenes es un punto clave para la experimentación.

Capítulo 6

Módulo de Interacción Hombre-Robot

La comunicación entre un hombre y un robot es un aspecto importante para los robots de servicio que pretenden formar parte de la vida cotidiana del hombre. Esta comunicación debe ser lo más sencillo para ambas partes. La comunicación oral o gestual se considera más sencilla para el ser humano, pero es complejo para el robot. Por medio de comandos por computadora sería la opción más conveniente para el robot, pero para un ser humano podría ser tedioso y complicado. Pensando en un ambiente neutral, se decide que la interacción Hombre-Robot se lleve a cabo por medio de una aplicación. La aplicación, debe ser muy intuitiva y permitir al usuario dar las órdenes necesarias para que el robot aprenda objetos y le entregue alguno que requiera, esto mediante un entorno gráfico.

Figura 6.1: Diagrama de bloques del módulo de interfaz hombre-robot.

Además de contener la aplicación para la comunicación con el usuario, el módulo de interacción hombre-robot, debe permitir la comunicación de todos sus módulos y la plataforma robótica utilizada. Esta comunicación debe sincronizar tareas básicas del robot, como movimiento y captura de imágenes, con tareas del sistema modular. De tal manera que la plataforma robótica pueda sustituirse sin grandes modificaciones del sistema.

Considerando lo descrito anteriormente, se diseña el diagrama de bloques del módulo presentado en la Figura 6.1, el cual encapsula la interfaz entre el usuario y el robot en dos divisiones: (i) Entorno Gráfico y (ii) Sincronización de Tareas. El entorno gráfico es una aplicación intuitiva y amigable con el usuario. Este le permite dar las ordenes necesarias al robot para aprender

objetos nuevos, crear mapa de la habitación y buscar un objeto. Mientras que, la sincronización de tareas es el intermediario o esqueleto del sistema. Este comunica y sincroniza las funciones del robot con el módulo de reconocimiento de objetos, manipulación y navegación, así como las órdenes recibidas del usuario.

En este capítulo se presenta el desarrollo del módulo de interacción hombre-robot. El diseño y desarrollo del entorno gráfico se presenta en la sección 6.1. Posteriormente, se plasma la planificación de la sincronización de tareas en la sección 6.2.

6.1. Entorno Gráfico

Como se menciona en la sección anterior, la interfaz Hombre-Robot incluye una aplicación que muestra un entorno gráfico. Este entorno gráfico debe ser sencillo e intuitivo, de esta manera, permitir que cualquier usuario pueda dar las ordenes necesarias para que el robot aprenda y busque objetos. Para el desarrollo del entorno gráfico del módulo de interacción hombre-robot, se considera el diagrama de flujo del la Figura 6.2.

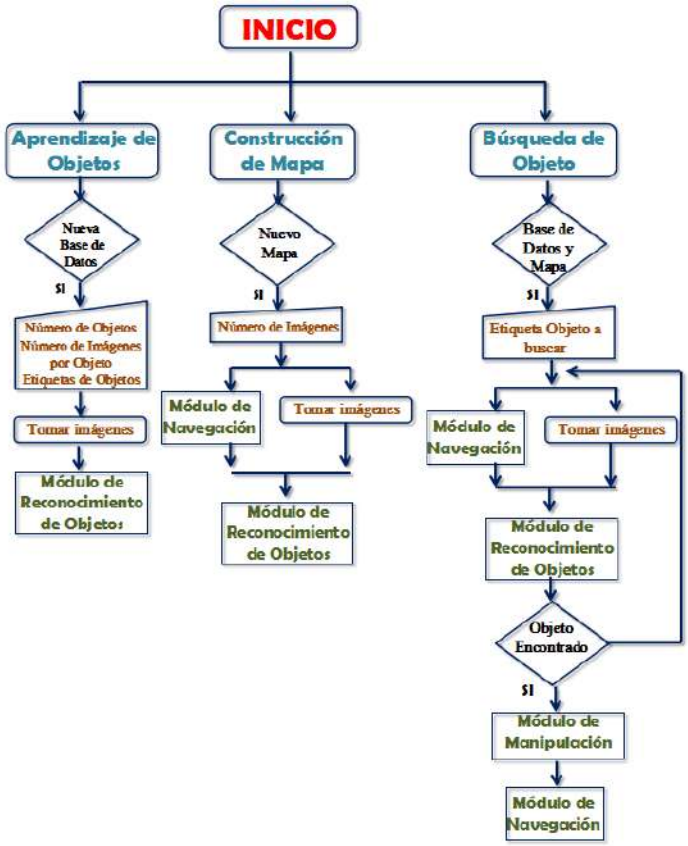


Figura 6.2: Diagrama de flujo de la interfaz Hombre-Robot.

Este diagrama de flujo muestra, de manera general, el proceso del sistema modular. Este proceso implica desde las opciones que tiene el usuario en la aplicación, así como la presencia de los módulos de reconocimiento de objetos, navegación y manipulación para cumplir las ta-

reas requeridas. El usuario puede elegir entre tres opciones: (i) “Aprendizaje de Objetos”, (ii) “Construcción de Mapa” y (iii) “Búsqueda de Objeto”.

La opción “Aprendizaje de Objetos” permite crear una nueva base de datos de objetos capturando las imágenes de cada uno y llevando acabo el proceso de aprendizaje. Este proceso lo realiza enviando la base de datos de objetos al “Módulo de Reconocimiento de Objetos”.

La opción “Construcción de Mapa” facilita la obtención del mapa bidimensional del entorno donde tabajará el robot. Esto se realiza solicitando el número de imágenes que capturará el robot y ejecutando el “Módulo de Navegación” para la captura estas y los respectivos datos inerciales. Cuando ha terminado la recolección de datos, estos son enviados al “Módulo de Reconocimiento de Objetos” para la construcción de dicho mapa.

La “Búsqueda de Objeto” no podrá realizarse sin antes elaborar un aprendizaje de objetos y una construcción de mapa. Cumpliendo lo anterior, el usuario solo debe indicar la etiqueta de uno de los objetos aprendidos. Entonces, el “Módulo de Navegación” es lanzado para que el robot camine por la habitación capturando imágenes. Esta imágenes son analizadas por el “Módulo de Reconocimiento de Objetos” para detectar el objeto requerido. Al detectar el objeto, el “Módulo de Manipulación de Objetos” se ejecuta para tomarlo, posteriormente, el “Módulo de Navegación” planificará la trayectoria de regreso al punto de partida para entregar el objeto.

Figura 6.3: Aplicación del sistema modular, (a) ventana principal, (b) advertencia para ingresar I.P. (c) advertencia para ingresar puerto.

El diseño de la aplicación que realiza el proceso del sistema modular basado en la descripción anterior, se describe a continuación. En la Figura 6.3 (a), se visualiza la ventana principal de la aplicación. En esta se observa que lo primero que se debe realizar es ingresar la “I.P.” del robot y el “Puerto”. Si estos no se ingresan, se desplegaran los mensajes de la Figura 6.3 (b) y 6.3 (c) respectivamente.

Figura 6.4: Aplicación del sistema modular, (a) ventana principal opciones habilitadas, (b) advertencia para editar datos ingresado.

Ingresados correctamente los datos, el botón "Aceptar" habilitará las opciones: "Editar", "Aprendizaje de Objetos", "Construcción de Mapa" y "Búsqueda de Objeto", (véase 6.4 (a)). El primero permite editar la I.P. y el puerto antes de avanzar, (véase 6.4 (b)). Mientras que las otras opciones, son botones que permiten ingresar a las respectivas funciones del sistema, a los cuales se acceden uno a la vez.

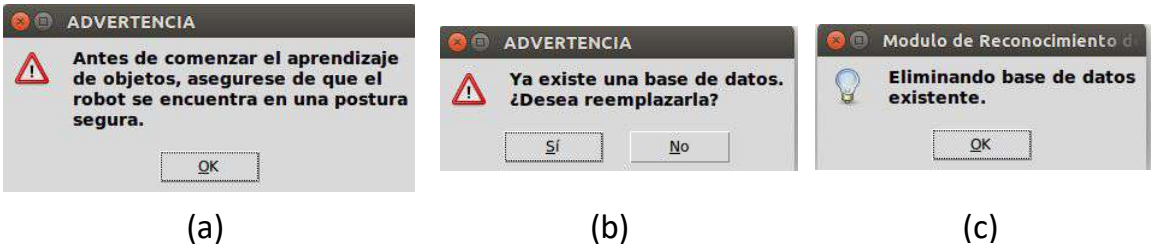


Figura 6.5: Ventanas que se despliegan antes de comenzar aprendizaje de objetos, (a) Advertencia de postura segura, (b) Reemplazar base de datos existente, (c) Eliminación de base de datos concluida.

Tal como se mencionó anteriormente, la primer opción permite el aprendizaje de cierto número de objetos indicado por el usuario. Antes de ingresar a esta opción, se desplegará una ventana (véase Figura 6.5 (a)), donde se advierte que es necesario que el robot se encuentre en una postura segura. Además, en caso de que exista una base de datos, se desplegará la ventana de la Figura 6.5 (b). Esta ventana preguntará si se desea reemplazar la base de datos de objetos. Posteriormente se desplegará la ventana de la Figura 6.5 (c), comunicando que se ha eliminado la base de datos.

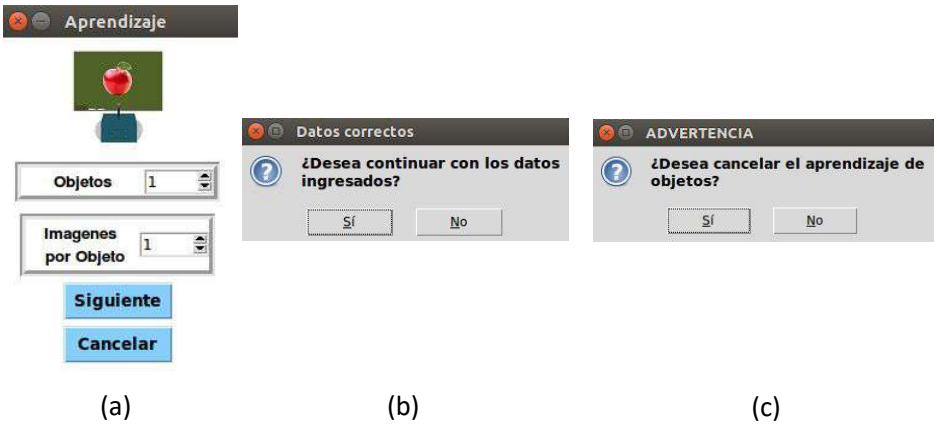


Figura 6.6: Aprendizaje de Objetos, (a) Ventana para ingresar los datos: número de objetos y número de imágenes, (b) Aceptar datos ingresados, (c) Cancelar aprendizaje de objetos.

Una vez seleccionada la opción de comenzar aprendizaje de objetos, se abrirá la ventana que muestra la Figura 6.6 (a). En esta ventana, el usuario debe indicar el número de objetos y número de imágenes por objeto. Ya que se indicaron los parámetros requeridos, se presiona el botón siguiente y se confirman los datos ingresados con la ventana de la Figura 6.6 (b).

Figura 6.7: Módulo de reconocimiento de objeto: entrenamiento, (a) ventana principal, (b) advertencia etiqueta correcta.

Para la captura de imágenes, la ventana en la Figura 6.7 (a) se despliega. En el panel derecho de esta ventana, se puede observar lo que el robot está visualizando con su cámara superior. En el panel izquierdo, se observan los indicadores de "Objeto" e "Imagen", estos marcan el número respectivo al cual pertenece la captura que se realizará. Enseguida de estos indicadores se encuentra un espacio en blanco para ingresar la etiqueta o nombre con el que será identificado el objeto a entrenar en turno. La ventana también cuenta con cinco botones, "Aceptar", "Capturar Imagen", "Comenzar Entrenamiento", "Reiniciar" y "Cancelar". Los botones de "Capturar Imagen" y "Comenzar Entrenamiento" se encuentran desactivados por el momento. Al ingresar la primer etiqueta y presionar el botón "Aceptar", aparecerá una ventana preguntado si la etiqueta es correcta (véase Figura 6.7 (b)). Si es así, el botón "Capturar Imagen" se habilita y el de "Aceptar" se inhabilita.

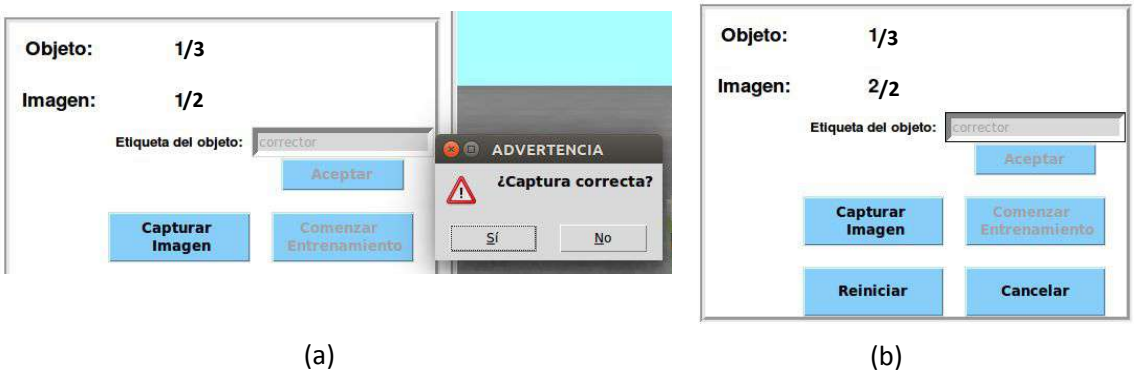


Figura 6.8: Módulo de reconocimiento de objeto: entrenamiento, (a) advertencia de captura correcta, (b) Indicador "Imagen" incrementando.

La captura de imágenes de los objetos a entrenar se apoya de lo que observa el robot con la intención de que el usuario colabore. Esto quiere decir que, el usuario debe colocar los objetos cerca del ángulo de visión del robot e ir cambiándolo de vista a cada imagen capturada. Cada que se captura una imagen, se despliega una advertencia preguntado si la captura es correcta, (véase Figura 6.8 (a)). Si es así, el indicador de "Imagen" va a incrementar para capturar la siguiente (véase Figura 6.8 (b)).

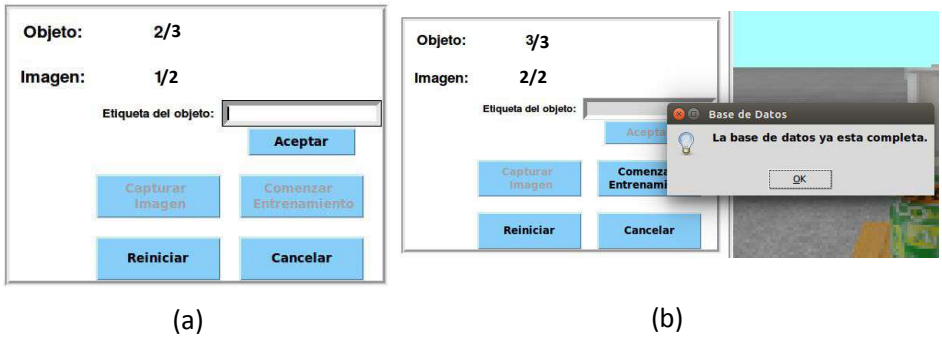


Figura 6.9: Módulo de reconocimiento de objeto: entrenamiento, (a) indicador "Objeto" incrementando (b) base de datos completa.

Cuando se detecta que la captura de imágenes para el objeto en turno ha terminado. El botón "Capturar Imagen" se inhabilita y el de "Aceptar" se habilita para ingresar la etiqueta del nuevo objeto. Además, el indicador "Objeto" se incrementa, (véase Figura 6.9 (a)). La captura de imágenes para este objeto se realiza de la misma manera que el anterior. Una vez capturadas todas las imágenes, el sistema detectará que la base de datos ya está completa y habilita el botón "Comenzar Entrenamiento" (véase Figura 6.9 (b)).

Figura 6.10: Módulo de reconocimiento de objeto: entrenamiento, (a) advertencia para comenzar el entrenamiento, (b) advertencia para cancelar aprendizaje, (c) advertencia para reiniciar la captura de imágenes.

Al selecciona el botón "Comenzar Entrenamiento", se desplegará una advertencia preguntado si se desea continuar, (véase Figura 6.10 (a)). Además, los botones "Cancelar" y "Reiniciar" siempre se encuentran habilitados lo que permite su selección en todo momento. El botón "Cancelar" cancelará la captura de imágenes y regresará a la ventana principal del sistema, (véase Figura 6.10 (b)). Mientras que el botón "Reiniciar", permite reiniciar toda la captura de objetos e imágenes, (véase Figura 6.10 (c)).

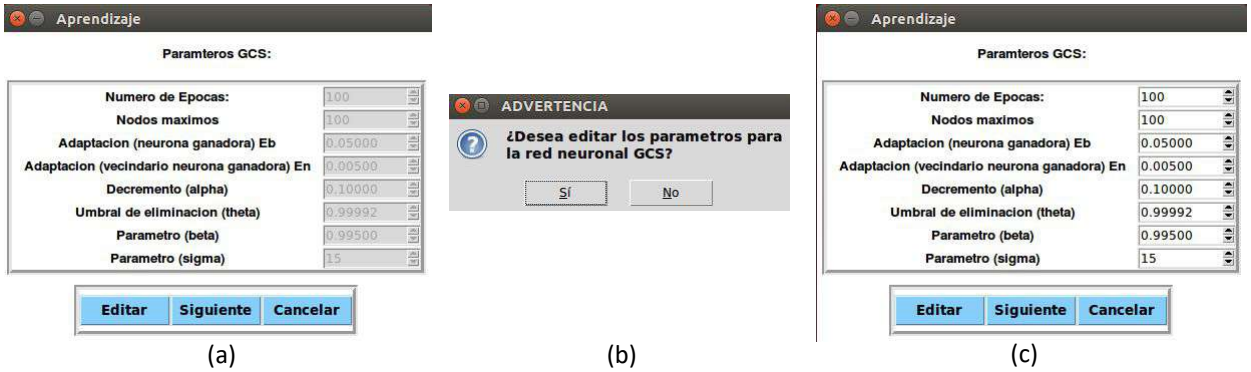


Figura 6.11: Aprendizaje, (a) Parámetros GCS asignados, (b) Advertencia para editar parámetros, (c) Parámetros GCS campos habilitados.

Después de seleccionar "Comenzar Entrenamiento", una nueva ventana se desplegará. Esta es la ventana para ingresar los datos de la red neuronal GCS. Como se observar en la Figura 6.11 (a), se deben ingresar los valores para: número de épocas, número máximo de nodos, valor de adaptación para el vecindario de la neurona ganadora, valor de adaptación para la neurona ganadora ε_b, valor de adaptación para el vecindario de la neurona ganador ε_n, decremento α, umbral de eliminación θ, parámetro β y parámetro σ. Estos parámetros tienen ciertos valores ya asignados (recomendados), solo se pueden editar cuando se presiona el botón que lo indica, (véase Figura 6.11 (b)). Al aceptar la edición, todos los campos se habilitarán para poder ser cambiados, (véase Figura 6.11 (c)).

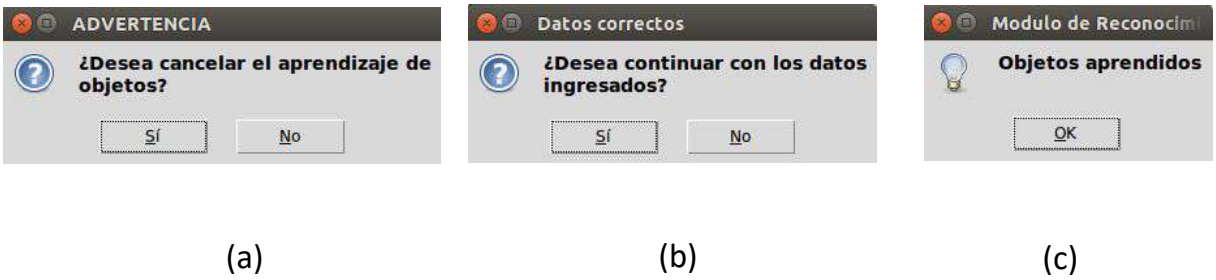


Figura 6.12: Aprendizaje, parámetros GCS, (a) Advertencia para cancelar aprendizaje de objetos, (b) Confirmar datos ingresados, (c) Objetos aprendidos.

El botón "Cancelar", permite cancelar el aprendizaje y regresar a la ventana principal del sistema, (véase Figura 6.12 (a)). Mientras que el botón "Siguiente", permite comenzar el entrenamiento de la red neuronal, (véase Figura 6.12 (b)), e indica que los objetos ya han sido aprendidos, (véase Figura 6.12 (c)).

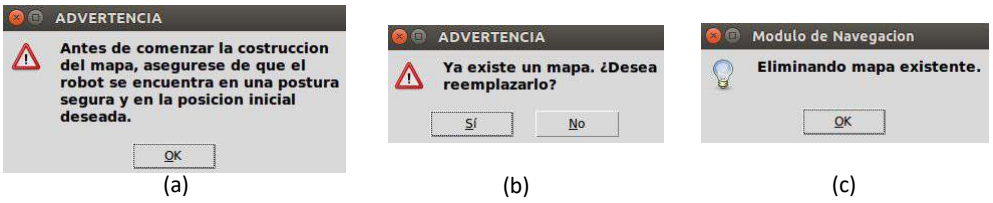


Figura 6.13: Ventanas que se despliegan antes de comenzar la construcción del mapa bidimensional, (a) Advertencia de postura segura, (b) Reemplazar mapa existente, (c) Eliminación de mapa concluida.

Al igual que la opción de comenzar aprendizaje, para la construcción del mapa bidimensional el sistema desplegará la advertencia de seguridad para el robot. Además, es importante saber que, antes de comenzar la construcción del mapa, el robot debe encontrarse en la ubicación de la habitación donde será el marco de referencia o punto origen del mapa, (véase Figura 6.13 (a)). Otro punto importante es que si existen un mapa ya construido, el sistema preguntará si se desea reemplazar el existente, (véase Figura 6.13 (b)). Al aceptar se indicará que la eliminación del mapa se ha realizado, (véase Figura 6.13 (c)).

Una vez ingresado a la opción de construcción de mapa, se solicitará el número de imágenes que tomará el robot durante su recorrido en la habitación. Así como las dimensiones de largo y ancho de esta en metros, (véase Figura 6.14 (a)). En esta ventana, el botón "Siguiente" permite confirmar los datos ingresados y continuar a la construcción del mapa. Mientras que el botón de "Cancelar", terminará el proceso y regresará a la ventana principal del sistema.

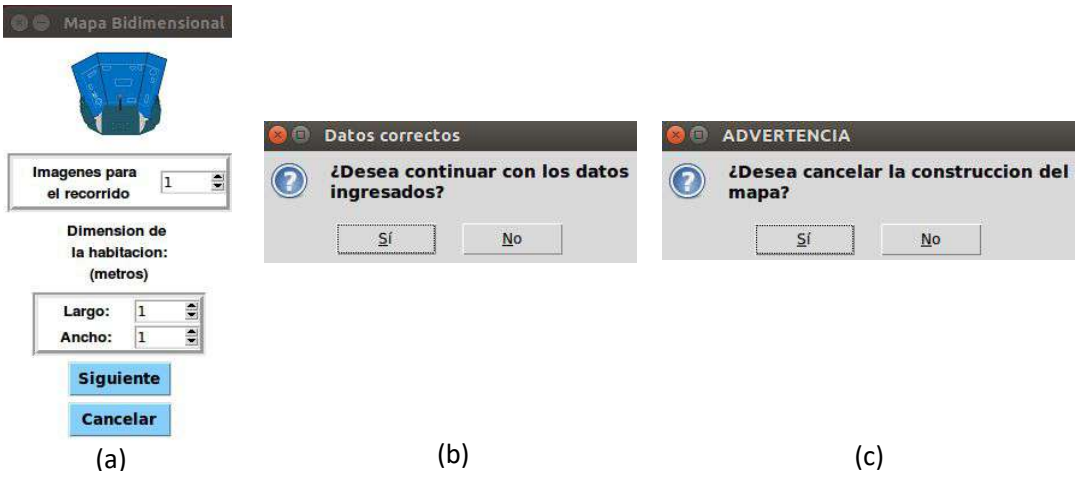


Figura 6.14: Mapa bidimensional, (a) ingresar número de imágenes para el recorrido y dimensiones de la habitación, (b) confirmación de datos ingresados, (c) cancelación de construcción del mapa.

Después de confirmar los datos, una nueva ventana se despliega para continuar con la construcción del mapa bidimensional, (véase Figura 6.15 (a)). En la parte superior de esta ventana se observa lo que el robot ve con su cámara y las dimensiones de la habitación que recorrerá. En la parte inferior, se observa el indicador "Imagen", y los botones "Reiniciar","Comenzar Recorrido" y "Cancelar".

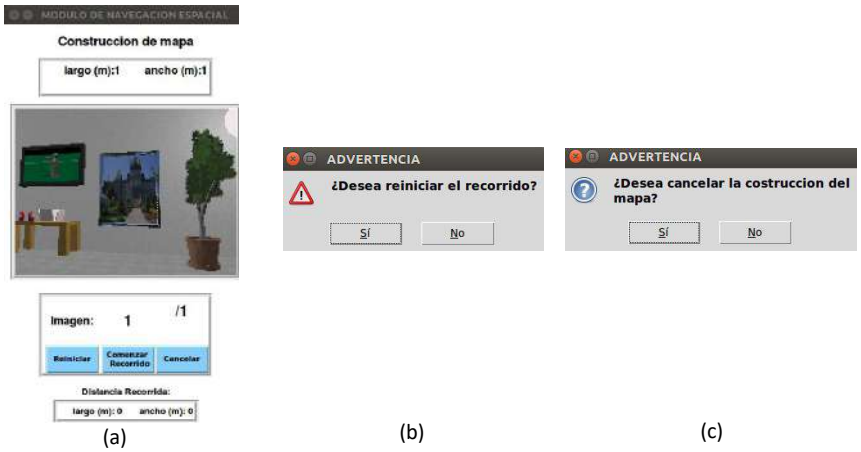


Figura 6.15: Construcción de mapa bidimensional, (a, (b) reiniciar el recorrido, (c) cancelación de construcción del mapa.

Al seleccionar “Comenzar Recorrido” el robot empezará a moverse y capturar imágenes alrededor de la habitación, guardando su ubicación. La captura de imágenes se realiza cada cierta distancia, dependiendo del número de estas indicado. Mientras se avanza con la captura de imágenes alrededor de la habitación, el indicador “Imagen” incrementa, en la parte inferior se observa la distancia recorrida por el robot y se detendrá cuando regrese al punto donde partió.

El botón de “Reiniciar” permitirá volver a comenzar el recorrido, (véase Figura 6.15 (b)). Mientras que el botón “Cancelar” permite cancelar la construcción del mapa y volver a la ventana principal del sistema, (véase Figura 6.15 (c)).

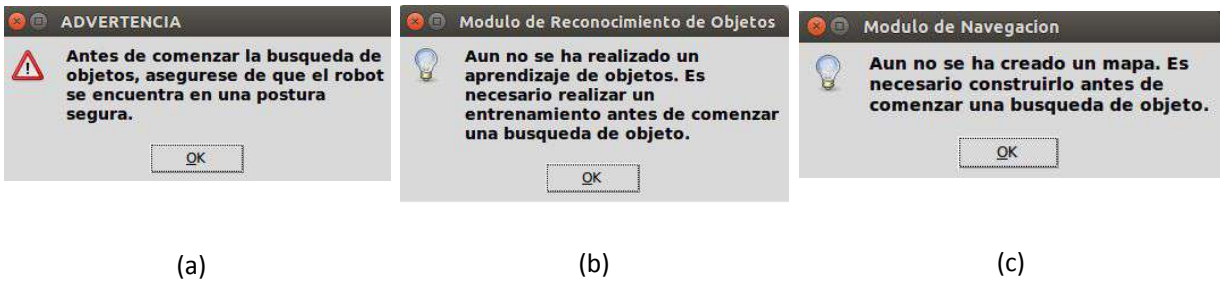


Figura 6.16: Ventanas que se despliegan antes de comenzar la búsqueda de objetos, (a) Advertencia de postura segura, (b) No existe base de datos de objetos, (c) No existe mapa bidimensional de habitación.

Al seleccionar la tercer opción del sistema modular, el cual es la búsqueda de objetos, se desplegará el mensaje de la Figura 6.16 (a). Este mensaje advierte que el robot debe estar en una postura segura antes de comenzar la búsqueda. Es importante saber que solo se podrá ingresar a la opción de comenzar una búsqueda de objetos una vez que se hayan realizado el aprendizaje de objetos y la construcción del mapa bidimensional de la habitación. Si no se ha realizado un aprendizaje de objetos, se desplegará un mensaje indicando que es necesario realizarlo, (véase Figura 6.16 (b)). De igual manera, si no se ha realizado una construcción de mapa, se desplegará un mensaje indicando que es necesario realizarlo, (véase Figura 6.16 (c)).

Habiendo ingresado a la opción búsqueda de objeto, se abrirá una ventana donde el usuario debe elegir, de una lista, el objeto que desea que el robot le entregue. Además, debe ingresar la altura de la plataforma que soporta los objetos (véase Figura 6.17 (a)). Esta misma ventana tiene los botones “Siguiente” y “Cancelar”. El primer botón confirma el objeto seleccionado, (véase Figura 6.17 (b)). Mientras que el segundo cancela la búsqueda y regresa a la ventana principal del sistema, (véase Figura 6.17 (c)).

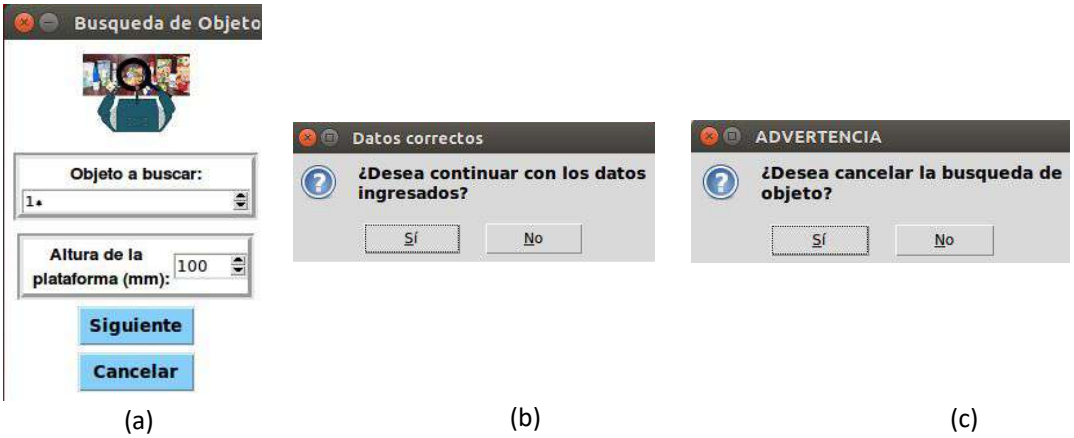


Figura 6.17: Búsqueda de objetos, (a) selección de objeto y altura de plataforma, (b) confirmar datos ingresados, (c) cancelar búsqueda de objeto.

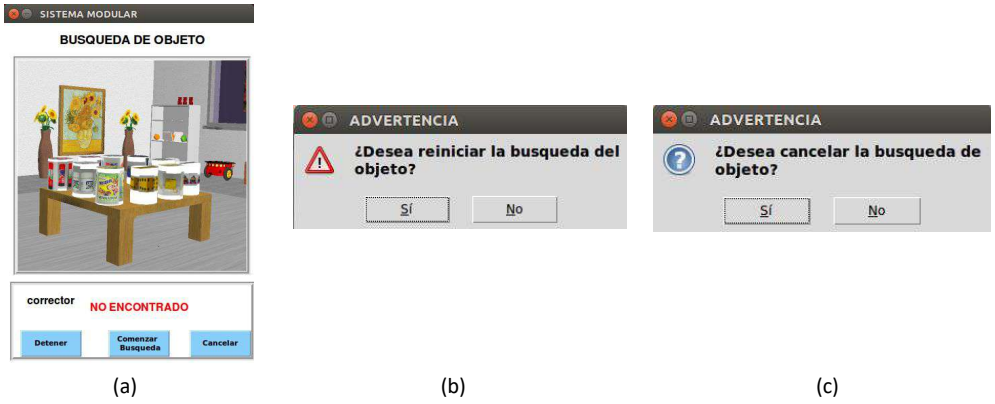


Figura 6.18: Búsqueda de objetos, (a) ventana principal para comenzar búsqueda, (b) reiniciar búsqueda de objetos, (c) cancelar búsqueda de objetos.

Con el objeto ha buscar seleccionado, se ingresa a la ventana de la Figura 6.18 (a), donde se observa lo que el robot esta viendo en la parte superior. En la parte inferior, se muestra el objeto que se está buscando y un indicador que muestra el estado de la búsqueda, el cual comienza en "NO ENCONTRADO". De igual manera, se encuentran los botones "Reiniciar", "Comenzar Búsqueda" y "Cancelar". El botón "Reiniciar" permite comenzar nuevamente la búsqueda del objeto desplegando el mensaje de advertencia de la Figura 6.18 (b). El botón de "Cancelar", cancela la búsqueda de objeto, (véase Figura 6.18 (c)), y regresa a la ventana principal del sistema.

La opción "Comenzar Búsqueda", comienza la búsqueda del objeto, (véase Figura 6.19 (a)). La búsqueda se realiza mediante un proceso cíclico donde el robot navega por la habitación tomando imágenes y evaluándolas para encontrar el objeto. El proceso cíclico se detiene una vez encontrado el objeto, el cual se indica cambiando el estado a "ENCONTRADO", (véase Figura 6.19 (b)). Una vez encontrado el objeto, se inicia el módulo de manipulación de objetos para levantarlo. Ya que el robot levantó el objeto, el módulo de navegación realiza la planeación de la trayectoria para regresar al punto de origen de la habitación.

Figura 6.19: Búsqueda de objetos, (a) comenzar búsqueda de objetos, (b) objeto encontrado.

6.2. Sincronización de Tareas

Además de la aplicación, el módulo de interacción hombre-robot permitir la comunicación de todos sus módulos y la plataforma robótica utilizada. Esta comunicación sincroniza tareas básicas del robot como movimiento, adquisición de información de sensores y captura de imágenes, con tareas del sistema modular. De tal manera que si se quisiera sustituir la plataforma robótica, la re-codificación del sistema se realice sin grandes modificaciones.

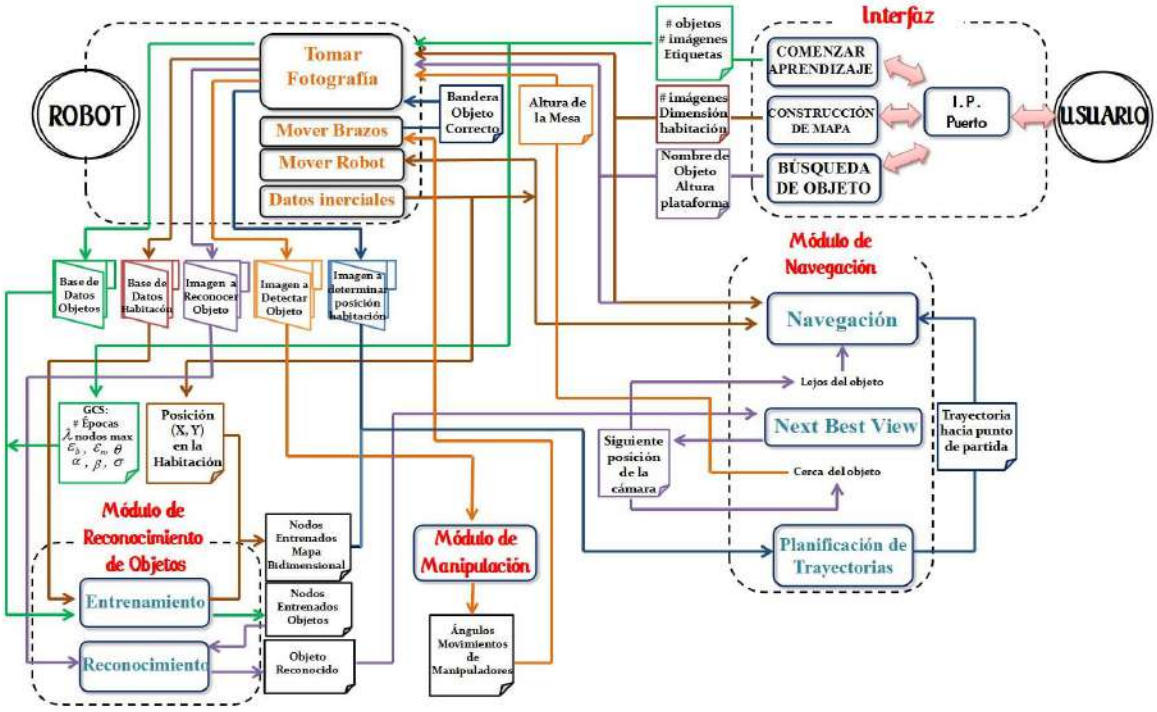


Figura 6.20: Módulo de Interacción Hombre-Robot: sincronización de tareas.

La sincronización de tareas, considerada el intermediario o esqueleto del sistema, se detalla en la Figura 6.20. En esta se observa, de manera independiente encerrados por medio de líneas

punteadas, cada uno de los módulos del sistema y la plataforma robótica. Este intermediario comunica y sincroniza las funciones del robot con el módulo de reconocimiento de objetos, manipulación y navegación, así como las órdenes recibidas del usuario. El proceso que sigue cada orden, sus carpetas y elementos que interviene, se identifica con un mismo color.

El sistema parte del usuario. Después de que este ingrese la I.P. y el puerto del robot, podrá iniciar el aprendizaje de objetos nuevos, creación de mapa de la habitación y búsqueda de un objeto. Para el aprendizaje de objetos, los datos de número de imágenes, objetos y etiquetas se almacenan en un archivo en formato de texto. Posteriormente, se van capturando las imágenes llamando a la función de "Tomar Fotografía" del robot. Estas imágenes se almacenan en su propia carpeta. Una vez terminada la captura de imágenes, se almacenan los parámetros para la red neuronal (número de épocas, neuronas, etc) en un archivo de texto y se lanza la función de "Entrenamiento" del módulo de reconocimiento de objetos. Esta función creará un archivo de texto con la información de la red entrenada.

La construcción del mapa de la habitación, crea un archivo de texto donde almacena el número de imágenes que tomará el robot y las dimensiones de la habitación donde realizará la recopilación de información. De manera paralela se lanza la función de "Tomar Fotografía" del robot y la de "Navegación". Esta última, a su vez, llama a la función de "Mover Robot" y "Datos Inerciales" del robot. Así es como se lleva acabo la recopilación de imágenes y odometría. Los datos de odometría se almacenan en un archivo de texto. Cuando la recopilación de datos e imágenes finaliza, se ejecuta la función de "Entrenamiento" del módulo de reconocimiento de objetos y se crea el archivo de texto de la red entrenada tomando en cuenta la información de los datos almacenados mediante odometría.

Para realizar una búsqueda de objetos, se debió haber realizado el aprendizaje de estos y la construcción del mapa de la habitación, de otra manera el sistema no permitirá avanzar. Si se cumple con lo anterior, el usuario deberá indicar el nombre del objeto y la altura de la plataforma que soporta los objetos. Estos se almacenan en un archivo de texto y se procede a ejecutar el análisis cíclico de imágenes. Para esto, la función de "Tomar Fotografía" se lanza junto con la "Navegación" que incluye las funciones "Mover Robot" y "Datos Odométricos". Este análisis cíclico envía las imágenes capturadas a analizar con la función de "Reconocimiento" del módulo de reconocimiento de objetos.
Si el objeto se reconoce, se lanza la función de "Next Best View" del módulo de navegación. Esta función entrega la siguiente posición recomendada para la cámara, la cual se ejecuta moviendo al robot completo. Si se detecta que el objeto está lejos, se lanza de nuevo el análisis cíclico del imágenes con el fin de acercarse al objeto. Si se detecta que el objeto está cerca, se lanza la función de "Tomar Fotografía" del robot. Esta imagen capturada se envía al módulo de manipulación junto con la altura de la plataforma que soporta a los objetos. Este módulo entregará los ángulos para el movimiento de manipuladores, los cuales se envían a la función de "Mover Brazos" del robot.

Una vez realizados los movimientos se detecta si el objeto se levantó correctamente si no fue así se realiza de nuevo el cálculo de ángulos ejecutando el módulo de manipulación de objetos. Si el objeto se levantó correctamente, se ejecuta la función de "Tomar Fotografía" para determinar

la posición del robot en la habitación enviando esta a la función de "Mover robot" del módulo de navegación. Esta función utiliza el mapa construido para determinar donde se encuentra el robot y calcula la trayectoria más cercana al punto del cual partió. Una vez calculada la trayectoria, ésta se envía a la función de "Mover robot" del mismo módulo y se procede a ejecutarla.

Capítulo 7

Implementación del sistema

En este capítulo se presenta la implementación del sistema modular para el reconocimiento de objetos en un contexto de robótica de servicio. La implementación se ha realizado con el robot humanoide NAO, en un ambiente semiestructurado. Se ha apoyado al robot a aprender la base de datos propia presentada en el capítulo del *Módulo de Reconocimiento de Objetos*. Los objetos se han colocado de manera aleatoria en el centro de una habitación en una plataforma de 30 cm. La habitación mide 4×3 metros, sin embargo, se la ha indicado al robot realizar la construcción del mapa bidimensional en un espacio de 3×3 metros. La habitación tiene diversos elementos, entre ellos posters con diferente información. Los experimentos realizados son: (1) reconocimiento de objetos individualmente, (2) construcción de mapa bidimensional, (3) búsqueda de diferentes objetos en la habitación, (4) manipulación de objetos, (5) ubicación en el mapa bidimensional. Se asume el uso de la interfaz hombre-robot en la implementación del sistema modular.

Cada uno de los experimentos forma una sección de este capítulo e incluyen parámetros seleccionados para la implementación y resultados. En la sección 7.1 Reconocimiento de Objetos se muestra la base de datos propia, los resultados del entrenamiento y de la evaluación de los objetos individualmente. La construcción del mapa bidimensional de la habitación donde se ejecutan los experimentos se muestra en la sección 7.2. La sección 7.3 presenta la búsqueda de cada uno de los objetos de la base de datos propia en la habitación. La manipulación de objetos de la sección A.4 se realiza de igual manera para cada uno de los objetos de la base de datos, considerando que el robot ya ha llegado a ellos. La ubicación en el mapa bidimensional de la sección 7.5 se realiza para comprobar que el robot es capaz de saber en qué lugar de la habitación se encuentra para poder regresar al punto del cual partió.

7.1. Reconocimiento de Objetos

La implementación del módulo parte del entrenamiento de los objetos a buscar. Se ha utilizado la base de datos propia presentada en el capítulo 3 *Módulo de Reconocimiento de Objetos*. Los objetos se muestran en la Figura 7.1, estos son 20 objetos de diferentes formas, tamaños y texturas. Para el entrenamiento, cada uno de ellos recibe una etiqueta diferente para identificarlos: (1) Mndonald, (2) Zapato, (3) Corrector, (4) Perfume, (5) Celular, (6) Tortuga, (7) Dado, (8) Resistol, (9) QG5, (10) Libro, (11) Cereal, (12) Regalo, (13) Bolsa, (14) Gel, (15)

Medicina, (16) Plumón Azul, (17) Plumón Rojo, (18) Spray, (19) Osito, (20) Libreta.

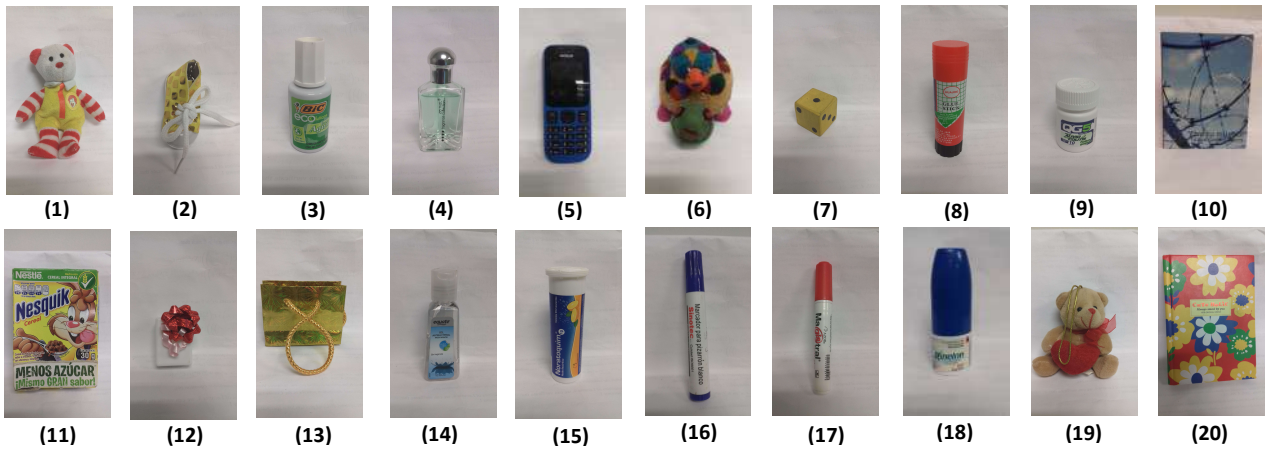


Figura 7.1: Base de datos de 20 objetos, (1) Mndonald, (2) Zapato, (3) Corrector, (4) Perfume, (5) Celular, (6) Tortuga, (7) Dado, (8) Resistol, (9) QG5, (10) Libro, (11) Cereal, (12) Regalo, (13) Bolsa, (14) Gel, (15) Medicina, (16) Plumón Azul, (17) Plumón Rojo, (18) Spray, (19) Osito, (20) Libreta.

Se ha utilizado la robot humanoide NAO para la captura de 20 imágenes de cada uno de los 20 objetos. El usuario apoyó con la captura de la base de datos, colocando los objetos frente al robot en una base de color homogéneo y claro. En la Figura 7.2, se pueden observar tres capturas realizadas al robot NAO durante el entrenamiento de los objetos: (a) Mcdonald, (b) Zapato y (c) Corrector. El entrenamiento se realizó sobre una plataforma de color blanco para evitar texturas que pudieran confundir el aprendizaje del objeto.

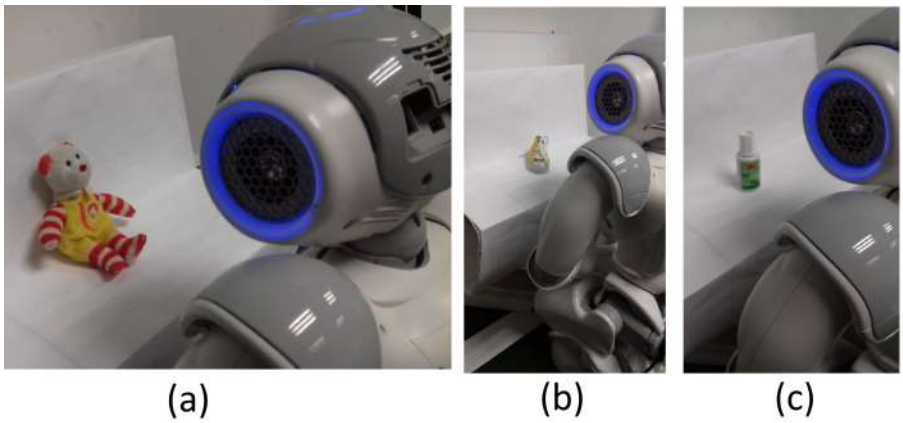


Figura 7.2: Captura de la base de datos realizada por el robot NAO, entrenamiento del objeto: (a) Mcdonald, (c) Zapato, (c) entrenamiento del objeto corrector.

Los parámetros elegidos para el entrenamiento de la red neuronal son los mostrados en la Tabla 7.1. Ya que se tienen 20 objetos y 20 imágenes por objeto, el número máximo de neuronas y de épocas se eligen para ser el doble que el total de histogramas a entrenar. El resto de

parámetros permanecen siendo los mismos utilizados en los experimentos del capítulo 3.

Tabla 7.1: Parámetros para el módulo de reconocimiento de objetos.

Parámetros	
Número de Objetos	20
Imágenes de Entrenamiento	20
Número de Neuronas	800
Número de Épocas	800
Adaptación (neurona ganadora) ε_b	0.05
Adaptación (vecindario neurona ganadora) ε_n	0.005
Decremento α	0.999
Umbral de Eliminación θ	0.001

El resultado del entrenamiento de la red se plasma en la gráfica de barras de la Figura 7.3. Esta gráfica muestra el número de neuronas asociadas a cada objeto entrenado. Se observa que para algunos objetos los histogramas se separaron en más neuronas. Por ejemplo, para los objetos 2 y 14 se distribuyeron en 19 neuronas, para los objetos 4 y 19 en 18 neuronas y para los objetos 3, 13 16 y 18 se distribuyeron en 17 neuronas. Los histogramas de los demás objetos fueron distribuidos en menos de 16 neuronas.

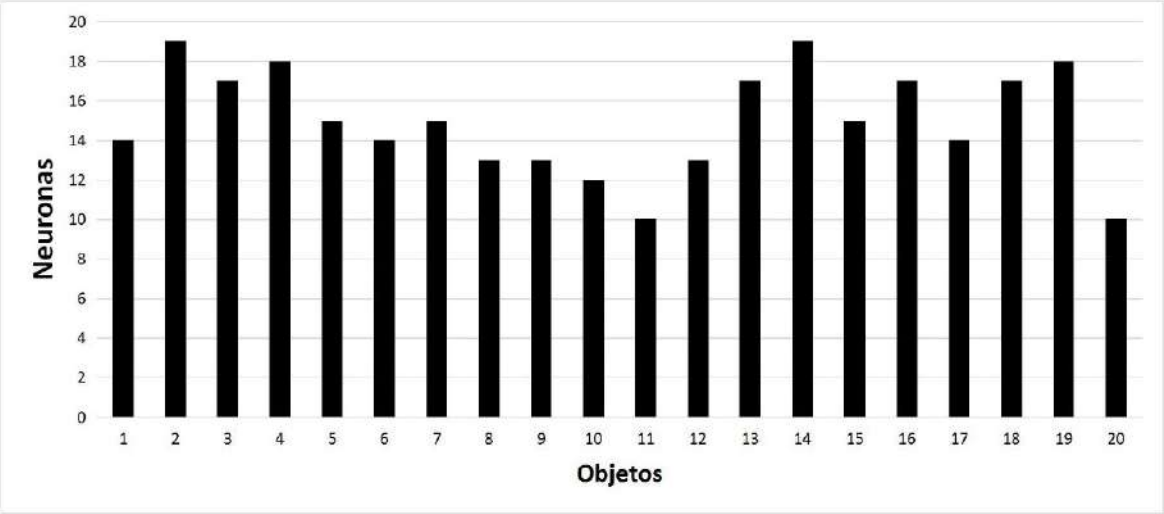


Figura 7.3: Gráfica de barras de objetos contra neuronas del entrenamiento de los 20 objetos.

Para la evaluación del módulo, se tomaron 10 imágenes más de cada uno de los objetos y se reconocieron con la red neuronal entrenada. Los resultados se plasma en la matriz de confusión de la Tabla 7.2. En esta matriz los 20 objetos se evalúan contra los 20 mismos, la diagonal entre éstos muestra los aciertos del reconocimiento.

Tabla 7.2: Matriz de confusión de la identificación de los 20 objetos.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
1	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	3	5	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
3	1	1	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	2	1	0	4	0	1	0	0	0	0	1	0	0	0	0	0	1	0	0	0
5	0	0	0	0	9	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	3	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	1	0	0	0	2	0	6	0	0	0	0	1	0	0	0	0	0	0	0
9	1	0	5	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0
10	0	1	0	0	1	0	0	1	0	5	1	0	0	0	1	0	0	0	0	0
11	0	0	0	1	0	0	0	1	0	0	8	0	0	0	0	0	0	0	0	0
12	0	0	3	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	2	0	0	6	0	1	0	0	0	1	0
14	1	1	0	0	1	0	0	0	0	0	0	1	0	4	1	0	0	1	0	0
15	0	2	0	0	0	2	0	0	0	0	0	1	0	0	4	1	0	0	0	0
16	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	7	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	8	0	0	0
18	1	1	1	1	0	1	0	0	0	0	0	0	0	0	1	0	0	4	0	0
19	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	4	0
20	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	8

El porcentaje total de la clasificación es de 64 %, el cual es aceptable. Sin embargo, se considera que se obtuvo un porcentaje bajo debido a algunos de los objetos entrenados. Por ejemplo, el cuarto objeto es el perfume, este objeto no tiene mucha textura y seguramente los histogramas extraídos no pudieron describirlo correctamente por lo que se confundió con más de un objeto. El noveno objeto correspondiente al medicamento QG5 se confundió con el objeto 3 siendo éste el corrector, un objeto similar ya que ambos tienen muchas letras en sus etiquetas. Los objetos 14, 15 y 18 son el gel, el medicamento y el spray, estos objetos tienen textura sin embargo solo se pueden observar desde ciertas vistas. De igual forma que con el perfume, los histogramas de ciertas vistas no lograron describir al objeto haciendo que éstos se confundiera con más de un objeto. El osito de peluche, el cual es el objeto 19, se confundió con el objeto 6, siendo este la tortuga. Los objetos son muy diferentes pero los histogramas del osito se tienden a parecer al de la tortuga.

Se consideran varios factores para justificar la evaluación obtenida. El robot humanoide NAO tiene una cámara de baja calidad a la cual se le pueden hacer algunos ajustes para mejorarla. Sin embargo, la definición de sus capturas no permite percibir toda la textura de los objetos. Además, las condiciones del ambiente como iluminación, perturban las tomas realizadas por el robot. Para mejorar el porcentaje obtenido con el robot NAO se considera que es necesario tener una base de datos con objetos con suficiente textura y un tamaño más grande. Estas consideraciones permitirán que las imágenes obtenidas por el robot tengan más información ya que la iluminación del ambiente es un problema que no se puede controlar en todo momento.

7.2. Construcción de Mapa Bidimensional

La construcción del mapa bidimensional se llevó a cabo en una habitación de 4×3 metros, de la cual el robot relizó la construcción del mapa bidimensional en un espacio de 3×3 metros. La habitación tiene diversos elementos, entre ellos posters con diferente información. En la Figura 7.4 se muestran las cuatro paredes de la habitación donde se pueden apreciar los elementos utilizados para el aprendizaje de ésta. Las paredes se han enumerado de la uno a la cuatro en contra de las manecillas del reloj para mostrar los resultados de manera sencilla.

Figura 7.4: Paredes que conforman la habitación de la cual se construye el mapa bidimensional numeradas en contra de las manecillas del reloj de la 1 a la 4.

Los objetos se han colocado de manera aleatoria en el centro de una habitación en una plataforma de 30 cm, ver Figura 7.5. En esta plataforma se encuentran los 20 objetos de la base de datos distribuidos en las orillas con el fin de que el robot pueda apreciarlos mejor y que queden dentro del área de trabajo de sus manipuladores al momento de tomarlo.

Figura 7.5: Los 20 objetos de la base de datos se colocaron por el centro de la habitación en una plataforma de 30 cm.

Se realizaron tres recorridos en la habitación para poder construir un mapa más preciso. El robot inició su recorrido desde la esquina derecha de la imagen 7.5 quedando está como coordenada global (0,0) de la habitación. Los recorridos los realizó en circuito cerrado cuadrangular de 3×3 m, girando su cabeza hacia la pared para poder capturar las imágenes mientras avanzaba y guardando su posición relativa, ver Figura 7.6. El primer y tercer recorrido los realizó en contra de las manecillas del reloj, el segundo en sentido contrario.

El número de capturas que realizó en cada recorrido por pared se indican en la Tabla 7.3. En esta tabla se indica el número de circuito realizado, las imágenes totales tomadas y las

Figura 7.6: El robot realiza su recorrido girando su cabeza hacia la pared más cercana para poder capturar imágenes de la 1ra. a la 4ta. pared, (1) captura de la pared 1: posición inicial (0,0), (2) captura pared 1: posición (2,0), (3) captura pared 2, posición (3,0), (4) captura pared 2, posición (3,2), (5) captura pared 4, posición (0,3), (6) captura pared 4, posición (0,0).

imágenes capturadas correspondientes a cada pared. Se almacenaron un total de 68 imágenes y poses utilizada para la construcción del mapa bidimensional.

Tabla 7.3: Parámetros principales de los experimentos: Construcción del mapa bidimensional.

Circuito	**Imágenes**	**Pared1**	**Pared2**	**Pared3**	**Pared4**
1	25	7	7	5	6
2	26	5	8	5	8
3	17	5	5	4	3

Las imágenes capturadas y las poses son la entrada a la variante del *Módulo de Reconocimiento de Objetos* utilizado en este trabajo para el aprendizaje de la habitación. El uso del módulo requiere especificar ciertos parámetros para el entrenamiento de la red neuronal que conformará el mapa de la habitación. Los parámetros se muestran en la Tabla 7.4, estos incluyen número de experimento por mapa bidimensional, número total de imágenes a entrenar, número máximo de neuronas y número de épocas. Se enviarán las 68 imágenes que conforman la base de datos de los recorridos realizado por el robot y se entrenará con 400 y 500 neuronas para construir dos mapas bidimensionales.

Tabla 7.4: Parámetros del módulo de reconocimiento de objetos para la construcción del mapa bidimensional.

Experimento	**Entrenamiento**	**Neuronas**	**Épocas**
1	68	400	100
2	68	500	200

Después del entrenamiento, los mapas bidimensionales quedaron con 27 y 36 poses del primer y segundo experimento respectivamente. En la Figura 7.7, se presentan los mapas bidimensionales obtenidos, los puntos azules en el gráfico representan las poses donde el robot capturó las imágenes. El (1) representa el mapa ideal que muestra las poses reales donde se

tomaron las capturas, el (2) corresponde al mapa construido del experimento 1 y el (3) representa el mapa construido en el experimento 2. Estos mapas representan la distribución de las neuronas con la pose asociada. Se observa que al incrementar las neuronas de 400 a 500 la distribución mejora un poco, sin embargo, el mapa construido se redujo de 3×3 metros a $1{,}5 \times 1{,}5$ metros. Se deduce que la obtención de un mapa más pequeño fue debido a que hay muchos falsos positivos y las poses se asociaron entre ellas a tal punto de reducirse.

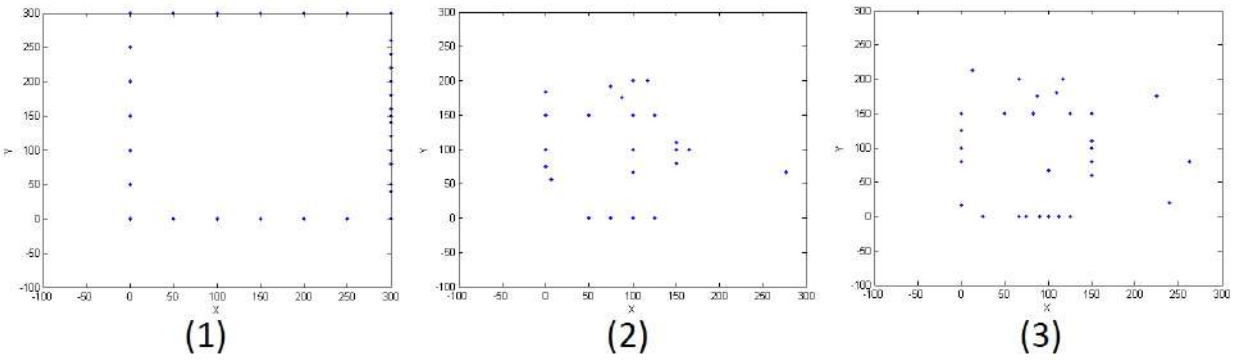


Figura 7.7: Mapa bidimensional construidos, distribución de las neuronas por pose en la habitación.

Es importante mencionar que las poses almacenadas por el robot fueron homogeneizadas en la coordenada que permanecía constante al realizar el recorrido para mostrar una distribución limpia. Como se ha mencionado en secciones anteriores de esta tesis, el mapa bidimensional apoyará al robot a identificar las paredes para no ir hacia ellas mientras realiza una búsqueda de objeto, así como a regresar al punto del cual partió considerado la coordenada global (0,0) para simular que entrega el objeto requerido al usuario.

7.3. Búsqueda de Objetos

Para la ejecución de la búsqueda de objetos se propuso buscar de 10 objetos entrenados de la base de datos. Los objetos se colocaron por el centro de la habitación, en una plataforma de 30 cm de altura, distribuidos en las orillas para que el robot pudiera apreciarlos mejor, ver Figura 7.8.

Considerando que el robot ya ha aprendido los objetos, se inicia la búsqueda de objetos. Se han propuesto la búsqueda de 10 de los objetos: (1) corrector líquido, (2) caja de cereal, (3) tortuga, (4) mcdonald, (5) plumón azul, (6) celular, (7) plumón rojo, (8) libreta, (9) dado, (10) regalo. Se busca que el robot logre identificar a cada uno de los objetos entre cuatro exploraciones alrededor de la mesa, ver Figura 7.9. En la primer exploración todos los objetos se encuentran en la mesa, en la segunda se dejan 10 objetos sobre la mesa, en la tercera 5 y en la cuarta 2.

El robot NAO capturó 10 imágenes por exploración, éstas son de diferentes posiciones alrededor de la mesa que contiene a los objetos. La idea es que de las exploraciones el robot

Figura 7.8: Plataforma de 30 cm de altura con los 20 objetos de la base de datos.

Figura 7.9: Capturas de los objetos sobre la mesa realizadas por el robot NAO durante las exploraciones, (1) 20 objetos, (2) 10 objetos, (3) 5 objetos, (4) 2 objetos.

identifique el objeto buscado. En la Tabla 7.5 se señalan los objetos, el número de exploración en la que se está buscando y los resultados que comprenden tasa de verdaderos positivos (TP), tasa de falsos positivos (FP) y porcentaje total obtenido. La tasa TP es el conteo total de veces acertadas donde el objeto aparece, la tasa FP es el conteo total de veces que confundió al objeto y el porcentaje es el resultado obtenido considerando los anteriores.

Algunas imágenes de los cuatro recorridos se muestran en la Figura 7.10, presentando resultados para cada objeto. En (1) se observa que la etiqueta del corrector a sido detectada, el siguiente paso es centrar la imagen lo que permitirá detectar mejor el objeto. En (2), (3) y (5) se busca al regalo, al dado y al celular respectivamente, si observa que debido al valor pequeño de tasa positiva el robot procederá a moverse para observar mejor los objetos. En la imagen (4) y (7) el robot ha detectado al objeto y ya que éste se encuentra ligeramente centrado le es posible tomarlo. En la (6) el objeto plumón azul a sido confundido completamente con uno de los posters de la habitación. En las imágenes (8), (9) y (10) los objetos se han detectado con una tasa de positivos alta y ya que están centrados el robot puede proceder a tomarlos.

Los resultados muestran un desempeño pobre con un porcentaje de reconocimiento del 54.74 %. Se considera que el porcentaje ha salido bajo ya que las imágenes del robot son difusas y de baja calidad. Otra razón es que este módulo depende del entrenamiento de los objetos, si se obtuvo bajo porcentaje de clasificación NBV tenderá a confundir los objetos. Sin embargo, los resultados demuestran que el método NBV cuenta con buenas bases para ejecutar una búsqueda de objeto y se espera que el porcentaje pueda mejorar si se utiliza una mejor cámara o procesamiento de imagen antes de evaluar.

Tabla 7.5: Tasa de positivos por objeto identificados en las exploraciones.

Exploración	Objeto	Tasa (TP)	Tasa (FP)	Porcentaje (%)
1	Corrector	101	100	50.24
1	Dado	6	9	40
1	Regalo	26	22	54.16
1	Mcdonald	60	46	56.6
2	Celular	52	51	50.48
2	Plumón Azul	17	18	48.57
2	Plumón Rojo	24	38	38.7
2	Libreta	172	86	66.67
3	Tortuga	90	45	67
4	Cereal	81	27	75

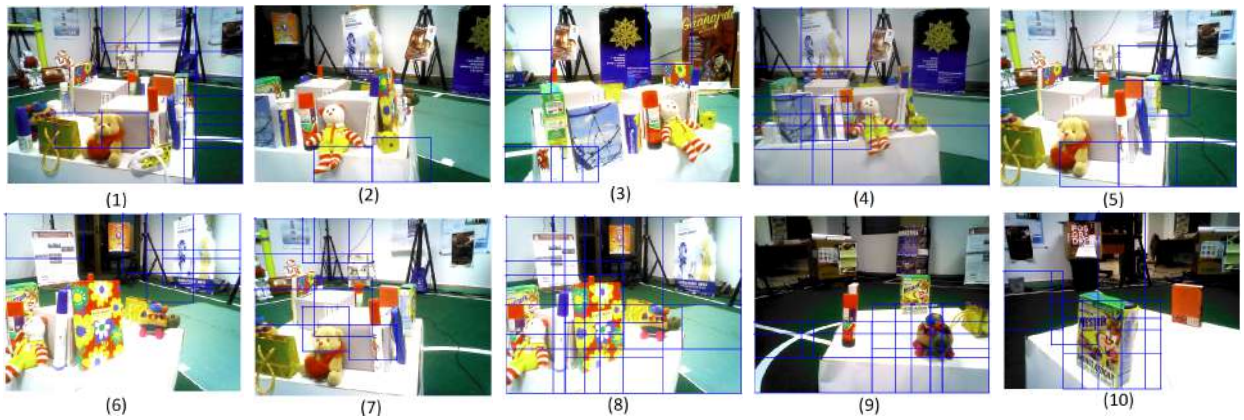


Figura 7.10: Capturas de los objetos sobre la mesa realizadas por el robot NAO virtual.

7.4. Manipulación de Objetos

El *Módulo de Manipulación de Objetos* se ejecuta una vez que el robot encontró el objeto y se acercó a él. Entonces, para llevar a cabo la validación de este módulo, el robot se ha colocado frente a la plataforma de 300 mm de altura que se encuentra en el centro de la habitación. Por practicidad, los objetos se colocan frente al robot en posición aleatoria dentro de su área de trabajo. Se eligieron los 10 objetos de la base da datos que no tuvieran colores claros para poder realizar una detección apropiada. Los objetos a manipular son (1) Bolsa, (2) Regalo, (3) Tortuga, (4) Cereal, (5) Mcdonald, (6) Osito, (7) Libro, (8) Resistol, (9) Dado, (10) Zapato. En la Figura **??** se observa al robot enfrente a los primeros cuatro objetos.

Como se ha mencionado en la descripción del módulo de manipulación de objetos, la entrada a éste son dos imágenes capturadas por el robot. Estas imágenes se capturan con la cámara superior e inferior del robot NAO, después de girar su cabeza hacia abajo para observar la superficie de la mesa, ver Figura 7.11.

Figura 7.11: Robot enfrente de cada uno de los objetos a manipular, (1) Bolsa, (2) Regalo, (3) Tortuga, (4) Cereal.

En la Figura 7.12 se observan las dos imágenes capturadas de los 10 objetos a manipular. Ya que las cámaras del robot NAO tiene una desviación entre ellas, algunos objetos pequeños no se logran apreciar con ambas cámaras por ejemplo el dado y el regalo. En la mayoría de las imágenes capturadas con la cámara superior, la detección falla detectando el suelo de la habitación como objeto. Sin embargo, el módulo permite realizar la manipulación mientras el objeto aparezca en una de las dos imágenes capturadas por el robot considerando primero la detección en la imagen inferior.

Figura 7.12: Capturas del robot NAO de las dos imágenes de los 10 objetos y la detección de estos, (1) Bolsa, (2) Regalo, (3) Tortuga, (4) Cereal, (5) Mcdonald, (6) Osito, (7) Libro, (8) Resistol, (9) Dado, (10) Zapato.

El resultado del módulo es el cálculo de la cinemática inversa de ambos manipuladores del robot humanoide NAO. Los ángulos calculados se plasman en la Tabla 7.6 para su visualización, se identifican los 10 objetos por su nombre y 5 ángulos correspondientes a las articulaciones del robot NAO.

Tabla 7.6: Ángulos en grados calculados para las posiciones de los 10 objetos.

Objeto	R/LShoulderPitch	RShoulderRoll	LShoulderRoll	RElbowRoll	LElbowRoll
Bolsa	2.67	3.063	16.8	67.75	-85.29
Regalo	5.16	3.309	13.833	66.83	-82.42
Tortuga	0.977	2.124	35.01	71.53	-88.5
Cereal	2.564	2.278	40.93	70.88	-88.5
Mcdonald	7.42	0.956	14.44	77.11	-88.5
Osito	1.58	3.25	30.98	67.02	-88.5
Libro	0.103	3.35	11.96	66.65	-80.84
Resistol	37.12	9.74	64.82	88.5	-88.5
Dado	3.88	1.19	40.76	75.88	-88.5
Zapato	3.315	5.89	17.98	58.122	-79.4

El módulo depende del proceso de segmentación de objetos para detectar el tamaño correcto del objeto. Además, es necesario mejorar la detección para que no confunda en objeto de la superficie de la mesa con otro que no esté en ella. A pesar de las restricciones y observaciones se considera que el módulo de manipulación de objetos obtuvo buen desempeño.

7.5. Ubicación en el Mapa Bidimensional

La tarea de ubicación en el mapa bidimensional tiene varios objetivos, uno de ellos es regresar al punto del cual partió para simular la entrega del objeto requerido por el usuario. Otro uso es ubicar las paredes de la habitación para evitarlas cuando éste realice una búsqueda de objeto.

Tabla 7.7: Parámetros principales de los experimentos para la ubicación en el mapa.

No.	(x,y) m
1	(0,0)
2	(0,1)
3	(0,2)
4	(0,3)
5	(1,0)
6	(2,0)
7	(3,0)
8	(3,3)
9	(1.5,2)
10	(2,1.5)

Para esta sección de experimentos se considera el mapa construido de la habitación de la sección de *Construcción del Mapa Bidimensional* de este capítulo. Entonces, con el mapa construido el robot es capaz de ubicarse en la habitación con una o dos imágenes de las paredes más cercanas. Se realizaron 10 experimentos, en la Tabla 7.7 se encuentran los parámetros que incluyen el número de experimento y la posición real que se pretende calcular (x,y) en metros.

Figura 7.13: Ejemplos de capturas realizadas por el robot NAO.

Para la evaluación de la precisión de la ubicación en el mapa bidimensional, el robot capturó 2 imágenes desde las 10 posiciones a evaluar de las pared más cercana en ese punto. Ejemplos de las capturas realizadas por el robot se observan en la Figura 7.13, las dos primeras imágenes pertenecen a la posición (0,0), mientras que las otras dos pertenecen a la posición (3,3) de la habitación.

Tabla 7.8: Resultado de las evaluaciones de poses para cada experimento.

	1	**2**
1	(0,16.66)	(25,0)
2	(150,150)	(150,60)
3	(75,0)	(125,0)
4	(100,0)	(0,80)
5	(0,80)	(25,0)
6	(0,100)	(25,0)
7	(25,0)	(66.67,0)
8	(125,0)	(0,100)
9	(106.5,175)	(125,100)
10	(150,80)	(125,100)

Los resultados se muestran en la Tabla 7.8, a las 10 ubicaciones se les asocia dos poses correspondientes al resultado de la evaluación de las dos imágenes tomadas desde esa posición. Debido a que en la sección de *Construcción del Mapa Bidimensional* el mapa obtenido no fue preciso, las poses obtenidas no son muy cercanas a las reales. La precisión más alta la obtuvo la pose número (1) con $\pm(25, 16{,}66)$ cercano a la pose real esperada. La siguiente mejor precisión se obtuvo en la pose número (9) con $\pm(34{,}25, 62{,}5)$. Las poses más bajas fueron (2) y (8) con precisiones de $\pm(150, 50)$ y $\pm(175, 200)$.

El mapa construido no fue preciso, sin embargo, la evaluación arroja resultados favorables considerando el mapa entrenado. En la Figura 7.14, se muestra la representación gráfica con las 10 ubicaciones determinadas con el módulo en el mapa bidimensional entrenado con anterioridad. Se observa que las poses son muy cercanas a las entrenadas menos las poses 7, 2 y 8 que se confundieron completamente.
Si se desea tener más precisión en la construcción del mapa bidimensional, es necesario tomar más capturas mientras se realiza el aprendizaje de la habitación y considerar un número elevado de neuronas para entrenar a la red neuronal. Además, la calidad de las imágenes permitirá

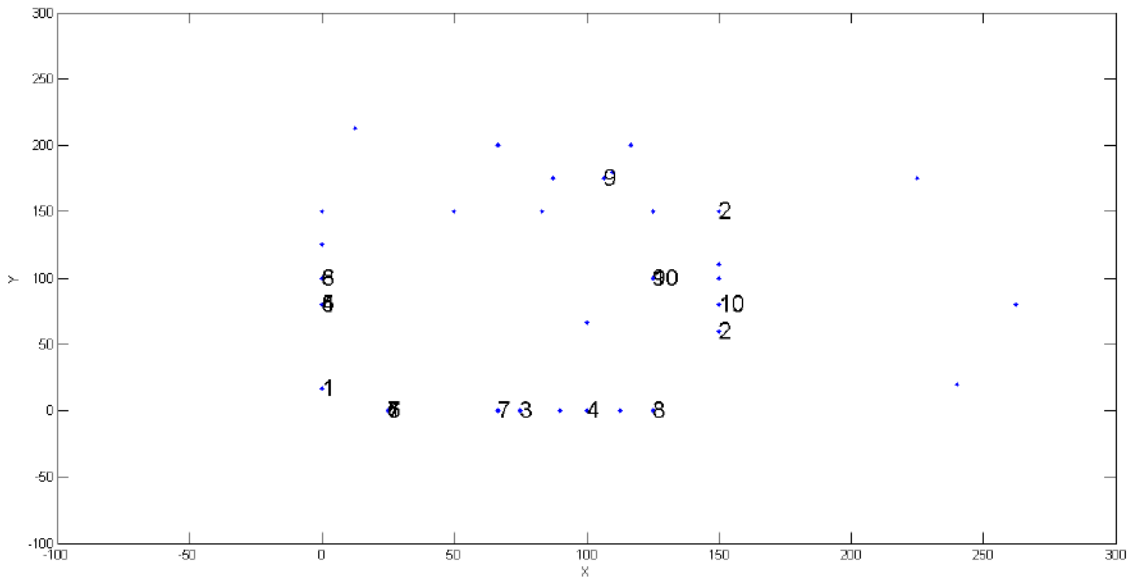


Figura 7.14: Gráfica que muestra las 10 ubicaciones determinadas con el módulo en el mapa bidimensional entrenado con anterioridad.

capturar más textura y puntos clave para asociarlos entre sí a la pose almacenada.

7.6. Conclusión

En este capítulo se realizó la implementación del sistema modular para el reconocimiento de objetos en un contexto de robótica de servicio. La implementación se realizó con un robot humanoide NAO como plataforma robótica. La implementación se realizó en un ambiente semiestructurado diseñado en una habitación de 4×3. En está habitación se colocaron diversos objetos y posters con información para utilizarlos como objetos clave en la construcción del mapa bidimensional de la habitación. En el centro de la habitación se colocó una plataforma de 30 cm de altura para colocar los objetos a buscar por el robot. El robot NAO construyó una base de datos con 20 objetos reales con 20 imágenes cada uno para el aprendizaje y su posterior búsqueda.

El *Módulo de Reconocimiento de Objetos* tuvo un desempeño de clasificación de 64 %. El cual es bajo en comparación con los experimentos realizados en el capítulo 3. Los resultados de la implementación se vieron afectados por la resolución de la cámara del robot NAO. La definición de sus capturas no permite percibir toda la textura de los objetos y se pierde información. Si se quisiera mejorar el porcentaje de clasificación es necesario considerar una cámara con mejor resolución y/o objetos con más texturas.

La construcción del mapa bidimensional por parte del *Módulo de Navegación Espacial* y también se vio afectada por la resolución de las imágenes capturadas por el robot. Sin embargo, se logró construir un mapa bidimensional con distribución aceptable pero con tamaño reducido.

Se considera que es necesario realizar más capturas durante el recorrido para poder mejorar la distribución del mapa.

La búsqueda de objetos por parte del *Módulo de Navegación Espacial* obtuvo un resultado bajo de igual manera. Ya que la búsqueda depende del aprendizaje de los objetos con el *Módulo de Reconocimiento de Objetos*, los resultados se consideran aceptables pues se consiguió un porcentaje similar al obtenido en el experimento anterior de clasificación.

En la implementación del *Módulo de Manipulación de Objetos* los resultados fueron buenos ya que los objetos fueron tomados sin mayor problema por el robot. Además, se conoce la restricción del color en los objetos lo cual permite seleccionar que objetos pueden ser detectados y manipulados correctamente.

La ubicación en el mapa bidimensional depende del mapa construido y como el mapa no se construyó de manera precisa la ubicación fallará. Sin embargo, si no se considera el mapa ideal o real, la ubicación realizada por el robot en el mapa entrenado tiene una precisión considerablemente buena.

La implementación del sistema modular en una plataforma robótica depende mucho de las características de ésta. Como se observo en este capítulo, la implementación en el robot humanoide NAO tuvo resultados bajos pues las características de la cámara de este robot ocasionaron la captura de imágenes de baja calidad. Es necesario considerar una calidad de imagen buena la cual permita obtener imágenes en las cuales se pueda percibir los detalles de los objetos. A pesar del problema con la cámara, se considera que el desempeño del sistema modular fue bueno pues los resultados fueron constantes. La evaluación de los módulos que dependen de otros como el *Módulo de Navegación Espacial* que depende del *Módulo de Reconocimiento de Objetos* obtuvieron porcentajes de clasificación similares.

Capítulo 8

Conclusiones

Este trabajo de tesis consistió en el desarrollo de un sistema modular para el reconocimiento de objetos en un contexto de robótica de servicio. El sistema modular puede ser utilizado por cualquier plataforma robótica móvil que tenga mínimo un brazo manipulador, una cámara con al menos un grado de libertad en ángulo *pitch* y con altura superior a la base del manipulador. El sistema necesitará mínimos ajustes para ser utilizado, principalmente en el módulo de manipulación de objetos. Se trabaja en un entorno semiestructurado considerando que, un robot de servicio en el hogar, oficina u hospital, primero debe conocer el lugar y los elementos no dinámicos con los que va a interactuar. El sistema se conforma de los siguientes módulos:

1. Reconocimiento de objetos
2. Manipulación de Objetos
3. Navegación Espacial
4. Interacción Hombre-Robot

Los módulos del sistema permiten: la creación de un mapa del lugar donde se buscarán los objetos, el aprendizaje de los objetos, la búsqueda de objetos indicado por el usuario, la manipulación y la entrega del objeto. Todas estás tareas se realizan, en su mayoría, en base al módulo de reconocimiento de objetos el cual es considerado como la tarea principal del sistema modular.

El módulo de reconocimiento de objetos utiliza el extractor de características tipo parche A-KAZE y la red neuronal GCS. El módulo de manipulación de objetos permite detectar un objeto, estimar su posición 3D y planificar los movimientos de manipuladores para levantarlo sin conocimiento previo del tamaño o forma del objeto. Este módulo se ha enfocado al robot humanoide NAO ya que la manipulación es particular para la plataforma robótica a utilizar. El módulo de navegación espacial comprende la localización espacial mediante la construcción de un mapa bidimensional con odometría y elementos visuales, además, la búsqueda de objetos mediante el enfoque NBV para navegar por una habitación analizando imágenes y tomando decisiones de lo observado en ella. El módulo de interacción hombre-robot permite que un usuario le indique al sistema cuando empezar un nuevo aprendizaje de objetos, apoyar con la etiqueta de los objetos entrenados, empezar nueva navegación, construcción de mapa y comenzar una

búsqueda de objeto. Además, es el intermediario entre los cuatro módulo y la plataforma robótica. Cada módulo ha sido desarrollado individualmente, además, se han expuesto experimentos y resultados en sus respectivos capítulos. De igual manera, se ha realizado una implementación del sistema modular completo en un ambiente semiestructurado, haciendo uso de la plataforma robótica NAO y utilizando una base de datos propia de 20 objetos. Los resultados, ventajas y desventajas del sistema se plasman de manera individual para cada módulo a continuación:

Módulo de Reconocimiento de Objetos

El módulo se validó usando una base de datos propia, en simulación y mediante la implementación en un robot real. En la validación con base de datos propia se entrenó al sistema con diferente número de objetos e imágenes por objeto. El porcentaje de clasificación correcta obtenido es arriba de 80 %. Mientras que la validación del módulo en simulación se realizó utilizando la plataforma humanoide NAO. Los resultados de clasificación bajan un poco a 75 % ya que los objetos simulados pierden calidad y textura. En los experimentos con el robot real la resolución de la imagen es muy importante en el aprendizaje de los objetos ya que se obtuvieron porcentajes diferentes y más bajos que los experimentos anteriores.

Por mencionar algunas restricciones del módulo, la principal es que los objetos a entrenar deben tener suficiente textura para poder ser diferenciados unos de otros. Como se mencionó anteriormente, la resolución de la cámara también es un factor importante que influye en el entrenamiento de los objetos. Otra restricción es que en el módulo de reconocimiento de objetos la clase nula no puede ser considerada ya que si el robot trata de reconocer un objeto que no ha aprendido, éste lo asociará o a un objeto ya existente o a una neurona libre que no haya sido utilizada en el entrenamiento. Si lo segundo llega a pasar entonces el robot habrá aprendido un objeto nuevo.

Módulo de Manipulación de Objetos

De este módulo es importante recalcar las restricciones que presenta, por ejemplo, el cálculo de la posición en el espacio del objeto, la cinemática inversa y la manipulación depende en gran medida de la plataforma robótica a utilizar. Además, es necesario conocer la altura de la superficie donde se encuentran los objetos para poder realizar el cálculo de una de las coordenadas del objeto en el espacio. Otra restricción es que los objetos deben estar dentro del área de trabajo de los manipuladores del robot y deben tener un tamaño y peso tales que el robot sea capaz de levantarlos y manipularlos. La segmentación de objetos está restringida a detectar objetos de color ya que se recomiendo que la plataforma o mesa donde se encuentren los objetos debe ser de un tono más claro que éstos. Por último, si el robot no puede alcanzar la posición calculada simplemente no tomará el objeto.
Considerando estas restricciones, los resultados obtenidos en los experimentos son aceptables pues el objetivo de levantar los objetos se ha cumplido. Los experimentos realizados con el módulo demuestran que éste presenta un buen desempeño ya que tanto en el experimento en simulación como en la ejecución con el robot real, la mayoría de los objetos se han tomado sin problemas.

Módulo de Navegación Espacial

La validación de este módulo depende en gran medida del aprendizaje realizado por el módulo de reconocimiento de objetos y la calidad de las imágenes capturadas con el robot. En los experimentos realizados con el robot real y el virtual, se tuvo el problema de baja calidad en las imágenes que ocasionó perdida de información. Sin embargo, los resultados en simulación son aceptables ya que se logró la construcción de un mapa bidimensional de la habitación y se realizaron experimento de ubicación con una precisión de hasta $\pm(0{,}06, 0{,}1)$. De igual manera se realizaron búsqueda de objetos con un porcentaje de 68.19 %. Los resultados no mejoraron en la implementación en un ambiente real y con una plataforma robótica real ya que la calidad de las imágenes no era buena, sin embargo los resultados se mantuvieron constantes.

Módulo de Interacción Hombre-Robot

Este módulo se utilizó en la implementación con el robot real en un ambiente semiestructurado llevado a cabo en el capítulo 7. Es poco lo que se tiene que decir de éste módulo pues la tarea de sincronización entre información de un módulo y otro lo realiza de manera estática. La comunicación entre módulos es poca, cada módulo escribe su información en archivos diferentes, cuando otro módulo lo requiere simplemente lo abre, lo lee y lo vuelve a cerrar. El módulo sincroniza las tareas de tal manera que al usuario le sea cómodo, sencillo y confiable. Como ya se mencionó, la implementación del sistema modular en una plataforma robótica depende mucho de las características de ésta. La implementación en el robot humanoide NAO realizada en esta tesis tuvo resultados bajos pues las características de la cámara de este robot ocasionaron la captura de imágenes de baja calidad. Es necesario considerar una calidad buena la cual permita obtener imágenes en las cuales se pueda percibir un poco más los detalles de los objetos. Se considera que el desempeño del sistema modular fue bueno pues los experimentos individuales tuvieron porcentajes altos de buena ejecución y los resultados de la implementación en el robot real no fueron muy bajos. Bastará con realizar la implementación del sistema modular en otras plataformas para corroborar su desempeño.

8.0.1. Trabajo Futuro

Como trabajo futuro para el sistema modular se propone:

- ✓ Implementar el sistema modular en diferentes plataformas móviles robóticas.
- ✓ Realizar un algoritmo genérico para la manipulación de objetos que dependa muy poco de la plataforma robótica a la cual será implementado.
- ✓ Utilizar SLAM visual para la construcción del mapa bidimensional de la habitación.
- ✓ Optimizar el enfoque NBV implementándolo con información de vídeo en tiempo real.
- ✓ Realizar una interfaz hombre-robot más dinámica que permita utilizar voz y descripción de los objetos a buscar.
- ✓ Mejorar la sincronización de módulos mediante un sistema en línea que esté en constante comunicación con todos los módulos.

Apéndice A

Robot Humanoide NAO

A.1. Plataforma de Desarrollo

El robot *NAO*, Figura A.1 (a), será la plataforma robótica en la cual se planea implementar el sistema desarrollado. Este, es un robot autónomo, programable, de mediana estatura, [18]. Además, es considerado uno de los más sofisticados y completos que se encuentra en el mercado, se adquiere principalmente por personas interesadas en la investigación, desarrollo de la robótica y áreas afines, así como escuelas y centros de investigación. Desde el 2006, año en que fue lanzado al mercado, se han creado cinco versiones que muestran algunas mejoras, pero entre estas prevalece el mismo concepto.

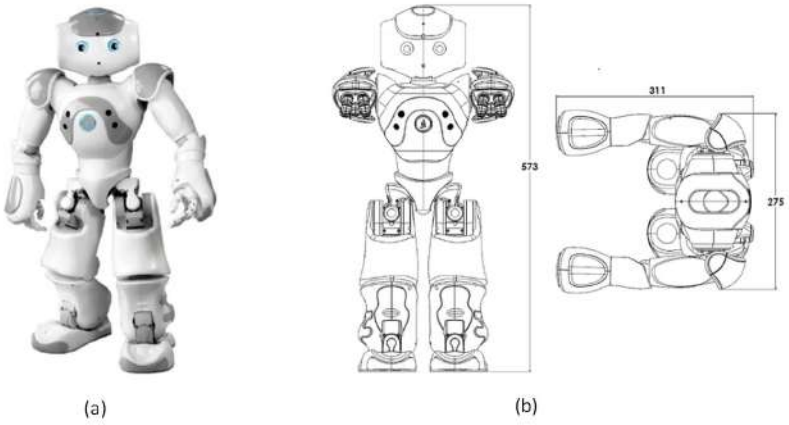


Figura A.1: (a) Robot NAO, (b) Dimensiones del robot humanoide NAO en mm [18].

En la Tabla A.1 se muestran algunas características sobre hardware y software de la cuarta versión de este robot. En la Figura A.1 (b), se presenta un esquemático del robot donde se señalan las dimensiones en milímetros tanto de alto como de ancho y largo de los brazos. El robot humanoide *NAO* cuenta con: software embebido y software de escritorio. El software embebido se denomina NAOqi. Este es el software principal que se ejecuta en el robot y lo controla para darle autonomía. NAOqi está contenida en el sistema operativo del robot el cual se denomina OpenNAO. Este sistema operativo es una distribución embebida de GNU/Linux basada en Gentoo. En este se encuentra un gran número de librerías y programas necesarios para que NAOqi pueda darle vida al humanoide [18]. Es posible utilizar copias del software

Tabla A.1: Características del robot humanoide NAO v4.

CARACTERÍSTICAS	
ALTURA	573 mm
PESO	4.3 kg
AUTONOMIA	90 min (60 min activo)
CPU	ATOM Z530 1.6 GHz
SISTEMA OPERATIVO (OS)	Linux Gentoo
OS COMPATIBLES	Windows, Mac OS, Linux
LENGUAJE DE PROGRAMACIÓN	C++, C, Python, Urbi
CONECTIVIDAD	Ethernet, WI-FI
MOTOR	Brush DC Coreless
GRADOS DE LIBERTAD	Robocup Edition 21 (DoF), NAO Academics Edition 25 (DoF)
SISTEMA MULTIMEDIA	4 Micrófonos, 2 Cámaras KVGA
OTROS	Sensores Inerciales, 2 girómetros, acelerómetro de tres dimensiones, 2 receptores de ultrasonido, 2 altavoces, IR, Sonar, sensor de posición MRE, Leds, etc.

NAOqi en una computadora para permitir el uso robots virtuales.
La programación de aplicaciones, para el robot *NAO*, puede ser desarrollada en diferentes lenguajes. El SDK disponible soporta Python, C++, .NET, Java, Matlab, Urbi, entre otros. Así mismo, es posible crear código para controlar remotamente al robot, crear nuevos módulos y cargarlos en el robot o/y crear código para enriquecer las librerías de Choregraphe. Este último es un entorno de programación gráfico desarrollado por Aldebaran Robotics.

El robot humanoide cuenta con una unidad de medición inercial (IMU). Esta IMU está constituida por 2 giroscópios y 3 acelerómetros montados en la parte inferior del torso del robot, como se observa en la Figura A.2 y la Tabla A.2. La medición de la velocidad de rotación del giroscópio es tanto en el eje X como en el Y. Mientras que el acelerómetro mide la aceleración linear a lo largo de los tres ejes.

Tabla A.2: Posición del dispositivo (IMU) con relación al marco de referencia del Torso del robot NAO.

DISPOSITIVO	**X (m)**	**Y (m)**	**Z (m)**
Acelerómetro	-0.008	0.00606	0.027
Giroscópio	-0.008	0.006	0.029

Para determinar la orientación del torso, Aldebaran llevó a cabo la implementación de un algoritmo que utiliza el acelerómetro (caso estático) y giroscopio (dinámico) [18]. La orientación del torso permite una estimación de la velocidad y el ángulo (yaw, pitch, roll) tanto del torso como de la cámara.

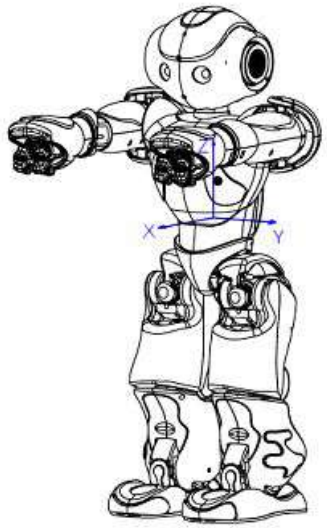


Figura A.2: Posición de IMU en el robot NAO.

De esta manera la IMU arroja valores enteros equivalentes a 1 gravedad (g) el cual se representa como el valor entero 56. Los datos que arroje el dispositivo deben ser convertidos a un valor en metros sobre segundos cuadrados para poder trabajar con ellos. Así se tiene entonces la equivalencia de:

$$1g = 56 = 9{,}81m/s^2 \tag{1}$$

A.1.1. Webots para NAO

Se utilizará el software *Webots para NAO* para la simulación del sistema y su respectiva validación. Este software es un simulador donde se puede hacer pruebas con las aplicaciones de *NAO* y los algoritmos en un mundo virtual. En la Figura A.3 se muestra el entorno del software. Se trata de un mundo virtual que permite lanzar una simulación de movimientos de *NAO* tomando en cuenta leyes físicas. Ofrece un lugar seguro para probar el funcionamiento de los comportamientos antes de reproducirlos en un robot real. El simulador es una versión específica de Webots 7, dedicado exclusivamente a la utilización de una simulación de *NAO*. Se ofrecen simulaciones predefinidas de *NAO* con sus controladores listos para usar [103].

A.1.2. Cinemática directa e inversa de manipuladores del robot NAO

Habiendo determinado la posición 3D del objeto en el espacio, es necesario utilizar un algoritmo que planifique trayectorias en el espacio de ángulos de las articulaciones del robot NAO para tomarlo. El diseño del algoritmo debe considerar las restricciones de movimiento de las articulaciones del robot. En esta sección se presenta la cinemática directa e inversa de los manipuladores del robot humanoide NAO.

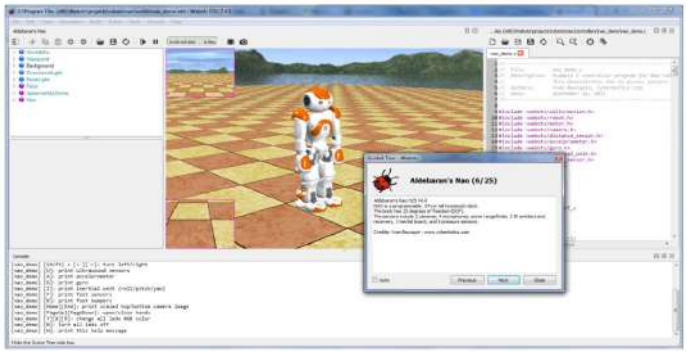
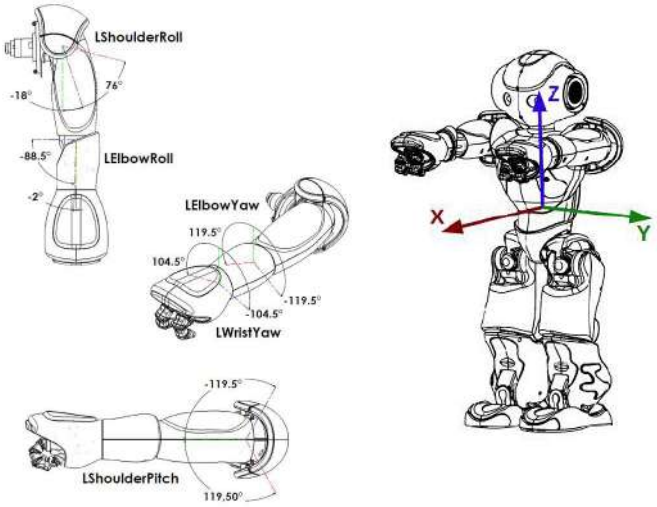


Figura A.3: Entorno *Webots para NAO*



(a) Articulaciones del brazo (b) Posición cero

Figura A.4: Especificaciones del robot NAO: denominación de articulaciones y posición cero, (Imagen extraída de [18].

El robot humanoide NAO cuenta con dos brazos de 4 grados de libertad (GDL) cada uno. Cada una de las articulaciones de los brazos se encuentran delimitadas a moverse cierto ángulo, como se observa en la Figura A.4 (a). Estos ángulos definidos como negativo o positivo según el brazo y partiendo de la posición cero que muestra la Figura (b).

Un análisis cinemático de los manipuladores del robot humanoide NAO fue desarrollado en [132] y [133]. Este análisis muestra de manera individual ambas cadenas cinemáticas de los brazos. Además, parte de los parámetros Denavit-Hartenberg (D-H) y los tamaños de los eslabones proporcionados por Aldebaran Robotics. Los parámetros para D-H, el brazo izquierdo se muestran en la Tabla A.3 y del brazo derecho en la Tabla A.4. Se utilizan medidas métricas y la transformación de la base a los brazos se considera constante con respecto al marco de referencia del torso, el cual se asume constante cuando el robot se encuentra de pie. Los parámetros utilizados para el análisis cinemático de los brazos son los siguientes:

- ✓ NeckOffsetZ = 126.50

Tabla A.3: Parámetros D-H para el brazo izquierdo del robot humanoide NAO.

Articulación	**a**	α	**d**	θ
Base	A(0, ShoulderOffsetY, ShoulderOffsetZ)			
LShoulderPitch	0	$-\frac{\pi}{2}$	0	θ_1
LShoulderRoll	0	$\frac{\pi}{2}$	0	$\theta_2 + \frac{\pi}{2}$
LElbowYaw	ElbowOffsetY	$\frac{\pi}{2}$	UpperArmLength	θ_3
LElbowRoll	0	$-\frac{\pi}{2}$	0	θ_4
Rotación	$R_z(-\frac{\pi}{2})$			
Efector final	A(HandOffsetX + LowerArmLength, 0, 0)			

Tabla A.4: Parámetros D-H para el brazo derecho del robot humanoide NAO.

Articulación	**a**	α	**d**	θ
Base	A(0, -ShoulderOffsetY, ShoulderOffsetZ)			
LShoulderPitch	0	$-\frac{\pi}{2}$	0	θ_1
LShoulderRoll	0	$\frac{\pi}{2}$	0	$\theta_2 + \frac{\pi}{2}$
LElbowYaw	-ElbowOffsetY	$\frac{\pi}{2}$	UpperArmLength	θ_3
LElbowRoll	0	$-\frac{\pi}{2}$	0	θ_4
Rotación	$R_z(-\frac{\pi}{2})$			
Efector final	A(HandOffsetX + LowerArmLength, 0, 0)			

- ✓ ShoulderOffsetY = 98.00
- ✓ ElbowOffsetY = 15.00
- ✓ UpperArmLength = 105.00
- ✓ LowerArmLength = 55.95
- ✓ ShoulderOffsetZ = 100.00
- ✓ HandOffsetX = 57.75
- ✓ HandOffsetZ = 12.31

La matriz de transformación final queda representada de la siguiente manera (2):

$$T_{Base}^{End} = A_{Base}^{0} T_0^1 T_1^2 T_2^3 T_3^4 R_z(-\frac{\pi}{2}) A_4^{End} \tag{2}$$

La cinemática inversa parte de la matriz de transformación final para cada cadena. Esta matriz se utiliza para despejar simbólicamente cada uno de las variables angulares de cada articulación. La metodología utilizada para la solución de la cinemática inversa incluye los siguientes pasos:

1. Construir la matriz de transformación numérica para el punto a alcanzar.

2. Construir la matriz de transformación simbólica a través de la cadena cinemática.
3. Formar un sistema no lineal igualando las dos matrices.
4. Manipular ambos lados del sistema para reducir las variables.
5. Encontrar valores para algunas articulaciones a través de geometría y trigonometría.
6. Encontrar valores para el resto de las articulaciones a través del sistema no lineal.
7. Validar todas las soluciones candidatas a través de cinemática directa.

El brazo derecho y el izquierdo son simétricos, sin embargo, la solución entre ambas difiere en la distancia a lo largo del eje y y en las articulaciones RShoulderRoll y RElbowRoll. La diferencia entre las distancias a lo largo del eje y es el ElbowOffsetY que cambia a negativo en $l_1 = -ElbowOffsetY$.

$$T' = (A^0_{Base})^{-1} T (A^{End}_4)^{-1} \tag{3}$$

$$T'' = (T')^{-1} (R_z(\frac{\pi}{2}))^{-1} \tag{4}$$

Para el brazo derecho se debe agregar una matriz extra de rotación después de la traslación final ya que el eje z se invierte:

$$T' = (A^0_{Base})^{-1} T (A^{End}_4)^{-1} (R_z(\frac{\pi}{2}))^{-1} \tag{5}$$

$$T'' = (T')^{-1} \tag{6}$$

Las ecuaciones para cada articulación quedan de la siguiente manera:

$$\theta_3 = \begin{cases} \arcsin(\frac{T''_{(3,4)}}{l_1}) \\ \pi - \arcsin(\frac{T''_{(3,4)}}{l_1}) \end{cases} \tag{7}$$

$$\theta_4 = \pm \arccos(\frac{l_3 T''_{(2,4)} - l_2 T''_{(1,4)} \cos\theta_3}{l_2^2 + l_1^2 \cos^2\theta_3}) \tag{8}$$

$$T''' = T' (T^3_2)^{-1} (T^4_3)^{-1} \tag{9}$$

$$\theta_2 = \arctan(\frac{T'''_{(2,1)}}{T'''_{(2,2)}}) - \frac{\pi}{2} \tag{10}$$

$$\theta_1 = \arctan(\frac{T'''_{(1,3)}}{T'''_{(3,3)}}) \tag{11}$$

Apéndice B

Adaptative Resonance Theory

Las RNA *Adaptative Resonance Theory* (ART) se basan en las redes de Grossberg [128], la innovación clave es el uso de "patrones prototipo". ART [127] se refiere a RNA auto-organizadas para el reconocimiento de patrones en tiempo real que se mantienen estables a diferentes patrones de entrada. Las redes ART codifican nuevos patrones de entrada, gracias al cambio de pesos o el cambio en las "huellas" de la memoria a largo plazo, de un filtro adaptativo retroalimentado.

Cuentan con un tipo de aprendizaje competitivo modificado, diseñado para mejorar el problema de estabilización de aprendizaje. La estabilidad de aprendizaje hace referencia a cómo es que un sistema puede mantener un aprendizaje estable a nuevos patrones de entrada ya que en la fase de adaptación se llega a modificar lo ya aprendido. Este problema se considera un "dilema estabilidad/plasticidad". Cada que un patrón de entrada se presente en la red, se compara con el vector prototipo al cual coincida más. Si la coincidencia entre el prototipo y el vector de entrada no es adecuada, un nuevo prototipo se selecciona. De esta manera prototipos aprendidos previamente no serán afectados por el nuevo aprendizaje.

La arquitectura básica del modelo ART se presenta en la Figura B.1. Esta arquitectura consiste en tres secciones: patrones prototipo de la capa 2 a capa 1, orientación del subsistema y la ganancia de control. Se describe como un conjunto de comparación y otro de reconocimiento compuesto de neuronas, un parámetro de vigilancia y un módulo de reinicio. Cuando un patrón de entrada se presenta a la red, éste es normalizado por la matriz de pesos. El conjunto de comparación toma una entrada y lo transfiere al conjunto de reconocimiento donde se identifica que fila de la matriz de pesos es más cercana al vector de entrada. La fila se mueve hacia el vector de entrada y cuando el aprendizaje se termina cada fila de la matriz de pesos se convierte en un patrón prototipo que representa un conjunto de vectores de entrada. El parámetro de vigilancia influye en el valor que produce la entrada, uno alto indica que debe almacenarse en memoria permanente, un valor bajo en memoria general.

El aprendizaje se realiza mediante un conjunto de conexiones retroalimentadas desde la capa 2 a la capa 1. Cuando un nodo en la capa 2 es activado, este reproduce un patrón prototipo en la capa 2. La capa 1 ejecuta una comparación entre el patrón prototipo reproducida y el patrón de entrada. Pero cuando el patrón de entrada y el prototipo no coinciden, la orientación del subsistema reinicia la capa 2. Este reinicio deshabilita la neurona llamada ganadora

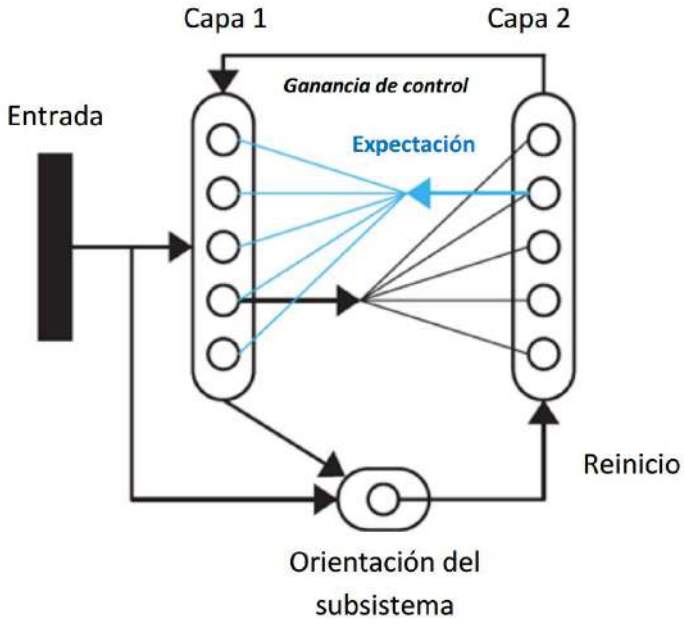


Figura B.1: Arquitectura básica del modelo ART [127]

perteneciente al patrón prototipo almacenado el cual es removido. Una nueva competición es ejecutada en la capa 2 para determinar una nueva neurona ganadora y proyectar un patrón prototipo.

El modelo básico de red ART o ART1 caracterizado como un sistema de ecuaciones diferenciales, predice el orden de búsqueda como función del historial de búsqueda de la red, y la estructura de categorías auto- organizadas asintótica por secuencias arbitrarias de patrones de entrada binarios. Probando la propiedad de auto estabilización y mostrando que los pesos adaptativos del sistema oscilan al menos una vez, sin embargo, no queda atrapado en los estados de memoria falsas o mínimos locales. Esta arquitectura es utilizada sólo para patrones de entrada binarios. Una variante de este modelo es ART2 que permite clasificar entradas binarias y análogas. La estructura básica es muy similar a ART1, solo que la capa 1 es substituida por varias subcapas ya que las entradas analógicas pueden encontrarse muy cercanas. Las subcapas ejecutan una combinación entre normalización y supresión de ruido, además de la comparación del vector de entrada y el vector prototipo que se necesitan para la orientación del sistema.

En la Figura B.2 se muestra la arquitectura típica de ART 2. En esta arquitectura, la capa 1 se divide en varios niveles de procesamiento y sistemas de ganancia de control. La retroalimentación de patrones de entrada y alimentación de señales se reciben en diferentes posiciones. Los ciclos positivos de retroalimentación permiten resaltar características sobresalientes y reducir el ruido. En la capa 2 se presentan las ecuaciones LTM (memoria a largo plazo), las cuales son más sencillas que en la arquitectura ART 1.

En la capa 1 se encuentra el potencial o la actividad STM (memoria a corto plazo). Para

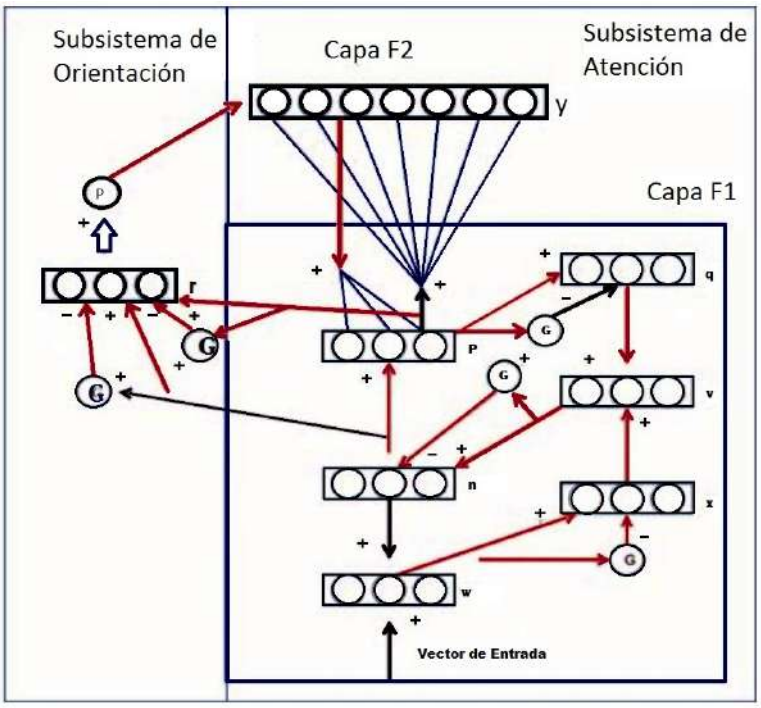


Figura B.2: Arquitectura típica de ART 2 [130]

cada nivel de esta capa el valor V_i obedece la siguiente función de membrana:

$$\varepsilon \frac{d}{dt} V_i = -AV_i + (1 - BV_i)J_i^+ - (C + DV_i)J_i^- \tag{1}$$

$(i = 1...M)$. El término J_i^+ es la entrada total excitadora del nodo i–ésimo y J_i^- es la entrada inhibidora. En ausencia de estas entradas, V_i decae a 0. El parámetro ε representa la proporción entre el tiempo de relajación de la actividad STM y LTM. Con la tasa O(1) LTM, entonces:

$$0 < \varepsilon << 1 \tag{2}$$

Además, en esta capa si $B \equiv 0$ y $C \equiv 0$, en su forma singular $\varepsilon \to 0$ se reduce a:

$$V_i = \frac{J_i^+}{A + DJ_i^-} \tag{3}$$

Entonces las ecuaciones de los diferentes niveles quedan de la siguiente manera:

$$w_i = I_i + au_i \tag{4}$$

$$x_i = \frac{w_i}{e + \|w\|} \tag{5}$$

$$v_i = f(X_i) + bf(q_i) \tag{6}$$

$$u_i = \frac{v_i}{e + \|v\|} \tag{7}$$

$$p_i = u_i + \sum_j g(y_j) z_{ij} \tag{8}$$

$$q_i = \frac{p_i}{e + \|p\|} \tag{9}$$

Donde $\|v\|$ denota la norma L_2 del vector V donde y_i es la actividad STM del j–ésimo nodo de la capa 2. La función de la señal no lineal f es de la forma:

$$f(x) = \begin{cases} \frac{2\theta x^2}{x^2+\theta^2} & \text{si } 0 \leq x \leq \theta \\ x & \text{si } x \geq \theta \end{cases} \tag{10}$$

El cual es diferenciable continuamente,

$$f(x) = \begin{cases} 0 & \text{si } 0 \leq x \leq \theta \\ x & \text{si } x \geq \theta \end{cases} \tag{11}$$

Linealmente en partes. Las variables x_i y q_i están siempre entre 0 y 1, así como los valores de la función $f(x_i)$ y $f(q_i)$. Alternativamente, la función de señal $f(x)$ puede ser elegida para saturarse en valores altos de x.

Ecuaciones de la Capa 2 STM

La representación de categorías de esta capa es similar a la de ART 1. Las propiedades clave son la mejora del contraste del patrón de entrada filtrado $C_1 \rightarrow C_2$, el reinicio, o la duración de la inhibición, de los nodos activos en la capa 2 cada que un patrón erróneo de la capa 1 sea lo suficientemente grande para activar la orientación del subsistema. El realce de contraste se lleva a cabo por competición en la capa 2. Se realiza una elección cuando un nodo reciba la entrada total más grande, donde T_j sea la entrada sumada y filtrada $C_1 \rightarrow C_2$ del nodo j-ésimo:

$$T_j = \sum_j p_j z_{ij} \tag{12}$$

$j = M + 1...N$. Entonces la capa 2 hará una elección del j-ésimo nodo máximo activo, mientras todos los demás nodos serán inhibidores cuando,

$$T_j = max(T_j : j = M + 1...N) \tag{13}$$

El reinicio de la capa 2 se llevará a cabo en diferentes formas, una de ellas es el uso de una red de campo dipolar cerrada. Cuando una entrada excitadora no especifica alcanza una entrada al campo dipolar, los nodos son inhibidos o reiniciados en proporción a los niveles de actividad STM. Esta inhibición crece hasta que la retroalimentación desde la capa 1 termina. Los elementos de la entrada al campo dipolares caracterizan por la siguiente dinámica,

$$g(y_j) = \begin{cases} d & \text{si } T_j = \text{máx}(T_j : \textit{el } j-\textit{ésimo nodo de la capa 2 no ha sido reiniciado}) \\ 0 & \textit{otro caso} \end{cases} \tag{14}$$

En tal caso para p de la capa 1 quedará como,

$$p_i = \begin{cases} u_i & \textit{si la capa esta inactiva} \\ u_i + dz_{ji} & \textit{si el } j-\textit{ésimo nodo de la capa 2 est inactivo} \end{cases} \tag{15}$$

Ecuaciones de la Capa 2 LTM (memoria a largo plazo)

La alimentación LTM hacia delante y hacia atrás está dada por,

$$\begin{array}{l} hacia\ adelante(C_2 \rightarrow C1) : \frac{d}{dt} z_{ji} = g(y_j)[p_i - z_{ji}] \\ hacia\ atrás(C_1 \rightarrow C2) : \frac{d}{dt} z_{ij} = g(y_j)[p_i - z_{ij}] \end{array} \tag{16}$$

Si la capa "realiza una elección", implica que si el j–ésimo nodo está activado, entonces,

$$\begin{array}{l} \frac{d}{dt} Z_{ji} = d[p_i - z_{ji}] = d(1-d)[\frac{u_i}{1-d} - z_{ji}] \\ \frac{d}{dt} Z_{ij} = d[p_i - z_{ij}] = d(1-d)[\frac{u_i}{1-d} - z_{ij}] \end{array} \tag{17}$$

con $0 < d < 1$. $\forall\ j\ \neq\ J,\ \frac{dz_{ji}}{d_t} = 0\ y\ \frac{dz_{ij}}{d_t} = 0$.

Ecuaciones de reinicio: subsistema de orientación

El cálculo de la coincidencia entre patrones análogos no necesita información de patrones. El grado de coincidencia entre el patrón STM de la capa 1 y un patrón LTM activo se determina por el vector $r = (r1...rm)$,

$$r_i = \frac{u_i + cp_i}{e + \|u\| + \|cp\|} \tag{18}$$

El subsistema de orientación se reinicia cada que un patrón de entrada se activa y,

$$\frac{\rho}{e + \|r\|} > 1 \tag{19}$$

Cuando el parámetro de vigilancia se encuentra entre 0 y 1.

El vector r da lugar a todas las propiedades que se requieren satisfacer en el intercambio. Cuando el j–ésimo nodo de la capa 2 se activa implica,

$$\|r\| = \frac{(1 + 2\|cp\| \cos(u,p) + \|cp\|^2)^{1/2}}{1 + \|cp\|} \tag{20}$$

donde el $\cos(u,p)$ denota el coseno del ángulo entre el vector u y el vector p. Asimismo, el vector p es igual a la suma $u + d_{Z_J}$, donde $Z_J \equiv (Z_{J1}...Z_{JM})$ denota el vector hacia delante de la proyección LTM desde el nodo j–esimo. Ya que $\|u\| = 1$, la geometría del vector suma $p = u + d_{Z_J}$, implica que:

$$\|p\| = \cos(u,p) = 1 + \|d_{Z_J}\| \cos(u, z_j) \tag{21}$$

Además,

$$\|p\| = (1 + 2\|d_{Z_J}\| \cos(u, Z_J) + \|d_{Z_J}\|^2)^{1/2} \tag{22}$$

Esto implica:

$$\|r\| = \frac{((1+c)^2 + 2(1+c)\|cd_{Z_J}\| \cos(u, Z_J) + \|cd_{Z_J}\|^2)^{1/2}}{1 + [c^2 + 2c\|cd_{Z_J}\| \cos(u, Z_J) + \|cd_{Z_J}\|^2]^{1/2}} \tag{23}$$

Los valores iniciales para los vectores de pesos ascendentes deben satisfacer,

$$z_{ji}(0) = 0 \tag{24}$$

Para $i = 1...M$ y $j = M + 1...N$, esta condición asegura que un reinicio no ocurra cuando el nodo activo erróneo se active. Los valores están dados por,

$$z_{ji}(0) \leq \frac{1}{(1 - d\sqrt{M})} \tag{25}$$

donde M es el número de unidades de cada subcapa de la capa 1.

Apéndice C

Comparativa entre las redes neuronales: GCS y ART2

La clasificación y reconocimiento de objetos es un tema que puede ser abordado por diversas herramientas de visión computacional. Las redes neuronales artificiales son una de estas herramientas que tienen cierta popularidad en esta rama de la ciencia. Según [56], las redes neuronales son un vehículo para el desarrollo adaptativo de coeficientes para funciones de decisión mediante la sucesión de las presentaciones de los conjuntos de entrenamiento de patrones. Lo anterior se considera un proceso de entrenamiento de cierto conjunto de características para la obtención de funciones de decisiones. Este proceso es llamado comúnmente *learning* o *training*, donde las características a utilizar serán las obtenidas por el método de detección de características tipo puntos en espacio de escala.

Son muchas las variantes de redes neuronales artificiales en la actualidad y se considera que no existe una manera precisa de elegir el tipo de red neuronal a utilizar para cierto problema en particular. Es por esto que se pretende realizar una comparativa entre las redes neuronales artificiales descritas en las secciones anteriores para evaluar su desempeño en clasificación de objetos sobre una tarea en particular. La comparativa parte del hecho de considerar un modelo de red neuronal artificial que pueda ser aplicada en tiempo real en ambientes complejos, que permanezca consistente, sea rápida y precisa.

Para la comparativa se consideran los tres tipos de redes neuronales descritos en la sección anterior, cuya principal similitud entre ellas, y tal vez la única, es el hecho de contar con aprendizaje no supervisado o automático. Estas redes son los modelos: Growing Cell Structure GCS y Adaptative Resonance Theory ART. Se considerarán diferentes parámetros de evaluación: pruebas cruzadas de los datos a evaluar, matriz de confusión de las diferentes redes y su respectivo análisis de la curva de característica operativa del receptor.

En la siguiente sección se describen las diferentes pruebas realizadas con sus respectivos resultados. Se realizaron dos experimentos:

Prueba 1: Atributos Binarios y nombres de animales,

Prueba 2: Base de datos COIL-100.

Las redes neuronales: Mapas Auto-organizados de Kohonen, GCS y ART 2, han sido utilizadas para la clasificación de los datos. Los mapas auto-organizados sólo se han utilizados para la primera prueba por su naturaleza binaria. Los parámetros para las pruebas con las diferentes redes se describen en la Tabla C.1-C.3, estos incluyen número de neuronas, número de iteraciones, coeficiente de aprendizaje, entre otros parámetros considerados para la red a la que corresponde. En la Tabla C.1 se describen los parámetros considerados para las redes neuronales de Kohonen en 1D y 2D. En la Tabla C.2 se presentan los parámetros utilizados para la implementación de la red neuronal GCS. En la Tabla C.3 se incluyeron los datos utilizados para la implementación de la red neuronal ART 2. En cada prueba, para cada red neuronal se han utilizado los mismos parámetros.

Se utilizó MATLAB 2015 para la ejecución de los algoritmos de las diferentes redes neuronales. La implementación tanto del entrenamiento como de la evaluación de los experimentos fue llevada a cabo en una computadora con procesador Intel(R) Core(TM) i7-4770 3.40GHz, con memoria RAM de 12 GB y sistema operativo Windows 8 de 64 bits.

Tabla C.1: Parámetros considerados para la red neuronal: Mapas Auto-organizados de Kohonen.

Red Neuronal	**Número de Neuronas**	**Número de Iteraciones**	**Radio de Vecindad**	**Coeficiente de Aprendizaje** α
KOHONEN 1D	10	10000	2.5	0.01
KOHONEN 2D	10X10	10000	2.5	1.5

Tabla C.2: Parámetros considerados para las redes neuronales: GCS.

Neuronas Iniciales	**Iteraciones**	**Nodos máximos**	**Iteraciones antes de inserción** λ	**Adaptación (neurona ganadora)** ε_b	**Adaptación (vecindario neurona ganadora)** ε_n	**Decremento** α	**Umbral de Eliminación** θ
3	1000	16	longitud del vector de entrada	0.05	0.005	0.999	0.001

Tabla C.3: Parámetros considerados para las redes neuronales: ART 2.

a	***b***	***c***	***d***	θ	***M***	ρ	**Capacidad**	**Dimensión de aprendizaje**	**Velocidad de aprendizaje**	**Relación f2-f1** ε
10	10	0.1	0.9	0.2	25	0.9	100	5	0.6	0.95

Prueba 1: Atributos Binarios y nombres de animales

Esta prueba consiste en el uso de un ejemplo ilustrativo con información de alta dimensionalidad propuesto por Ritter y Kohonen en 1989 [76]. Se trata de la descripción de algunas características de 16 animales en una lista binaria, (Figura C.2). Se conforma de 13 propiedades y el nombre del animal, $(1...n)$, el cual da un total de 16 vectores con dimensión 29. Lo que se busca con esta implementación es que las redes neuronales logren distinguir lo mejor posible los 16 animales, con el objetivo de obtener 16 clases una para cada animal.

Animal	Dove	Hen	Duck	Goose	Owl	Hawk	Eagle	Fox	Dog	Wolf	Cat	Tiger	Lion	Horse	Zebra	Cow
Is																
Small	1	1	1	1	1	1	0	0	0	0	1	0	0	0	0	0
Medium	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
Big	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
Has																
Two legs	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
Four legs	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
Hair	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
Hooves	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
Mane	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0
Feathers	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
Likes to																
Hunt	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	0
Run	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0
Fly	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
Swim	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

If an attribute applies for an animal the corresponding table entry is 1, otherwise 0.

Figura C.1: Atributos binarios y nombres de animales [76].

Los resultados de la clasificación de las propiedades binarias de animales se presentan en la Tabla C.4; donde se muestran en forma vertical los nombres de los animales y en horizontal la red neuronal utilizada para la clasificación. Los números indican a que clase pertenece cada animal. Se utilizaron *Mapas Auto-organizados de Kohonen* 1D y 2D, GCS y ART 2.

Los resultados de clasificación realizado con Kohonen 1D muestran una clara separación de los datos en aves, agrupados en la clase 10, y mamíferos, agrupados en la clase 6. Mientras que la clasificación realizada con Kohonen 2D realiza más de dos clases:

1. En la neurona (1,6) se agrupan las aves que comparten las siguientes características: pequeñas, con dos patas, plumas y vuelan.

2. En la neurona (1,3) se clasificó el ave que nada.

3. En la neurona (2,8) se clasificó el ave que no tiene ninguna habilidad en particular.

4. En la neurona (7,8) se clasificó el ave mediana que caza.

5. En la neurona (1,1) se agruparon los mamíferos grandes.

6. En la neurona (1,2) se agruparon el resto de los mamíferos.

La clasificación realizada con GCS obtuvo los siguientes grupos:

Tabla C.4: Clasificación de las propiedades binarias de animales.

ANIMALES	*KOHONEN 1D*	*KOHONEN 2D*	*GCS*	*ART 2*
Dove	10	(1,6)	10	1
Hen	10	(1,3)	4	1
Duck	10	(2,8)	3	2
Goose	10	(1,6)	8	2
Owl	10	(1,6)	15	3
Hawk	10	(1,6)	1	3
Eagle	10	(7,8)	1	4
Fox	6	(1,2)	2	5
Dog	6	(1,2)	16	6
Wolf	6	(1,2)	2	5
Cat	6	(1,2)	5	7
Tiger	6	(1,1)	14	8
Lion	6	(1,1)	9	8
Horse	6	(1,1)	7	9
Zebra	6	(1,1)	7	9
Cow	6	(1,1)	6	10

1. Aves con la habilidad de cazar y volar.
2. Mamíferos medianos con la habilidad de cazar.
3. Aves con la habilidad de nadar.
4. Aves sin ninguna habilidad en particular.
5. Mamíferos pequeños con la habilidad de cazar.
6. Mamíferos grandes sin ninguna habilidad en particular.
7. Mamíferos grandes, con melena, pezuñas y la habilidad de correr.
8. Aves con la habilidad de nadar y volar.
9. Mamíferos con melena, la habilidad de cazar y correr.
10. Aves con la habilidad de volar.
11. Mamíferos grandes con la habilidad de cazar y correr.
12. Aves con la habilidad de volar y cazar.
13. Mamíferos medianos con la habilidad de correr.

La clasificación con ART 2 obtuvo las siguientes clasificaciones:

1. Aves pequeñas

2. Aves con la habilidad de nadar.
3. Aves pequeñas con la habilidad de cazar.
4. Aves medianas con la habilidad de cazar.
5. Mamíferos medianos con la habilidad de cazar.
6. Mamíferos medianos con la habilidad de correr.
7. Mamíferos pequeños con la habilidad de cazar.
8. Mamíferos grandes que cazan y corren.
9. Mamíferos grandes con pesuñas, melena y la habilidad de correr.
10. Mamíferos grandes con pesuñas, sin ninguna habilidad en particular.

La clasificación de las propiedades binarias de animales ya se había realizado en trabajos anteriores con Kohonen y GCS en los cuales se obtuvieron resultados similares a los de la implementación realizada en este trabajo. Sin embargo, para este experimento se le especificó a la red GCS un total de 16 nodos o neuronas máximo con el fin de que a cada animal le correspondiera una clase. No obstante, se obtuvieron 13 clases diferentes con GCS lo cual indica que ciertos animales tienen características muy similares difícil de separar. A pesar de lo anterior, se siguen obteniendo mejores resultados que con la implementación de la red neuronal Kohonen 2D.

La clasificación con ART 2, por su parte, creó 10 clases diferentes, agrupando a los animales con características más similares entre sí como se le iban presentando a la red. Comparado con los resultados de la implementación con la red neuronal GCS, se puede apreciar que se consigue una mejor clasificación de los animales al crear más clases diferentes.

Prueba 2: Base de datos COIL-100

Para esta prueba, se seleccionan 30 objetos de la base de datos COIL-100 (Columbia University Image Library), se usan de 10 a 30 imágenes de cada objeto para entrenamiento, el resto de las imágenes se utilizan para su evaluación, lo anterior mediante las redes neuronales GCS y ART 2. Se propone el uso de atributos de color y niveles de grises en forma de histogramas como características para integrar el vector de entrada a las redes. Para la evaluación de los algoritmos se desprecia el tiempo de segmentación de las imágenes. Los objetos utilizados se muestran en la Figura , estos son los primeros 30 objetos de la base de datos COIL-100. En la Tabla VII se muestran los parámetros utilizados en los experimentos correspondientes a la cantidad de objetos y características utilizadas para la clasificación. Los resultados de cada experimento, tanto con GCS como con ART 2, se muestran en las siguientes secciones: resultados de clasificación y matrices de confusión.

El entrenamiento muestra las diferentes clases obtenidas con las dos redes (GCS y ART 2) para cada uno de los experimentos. La clasificación de los objetos se presenta en la Tabla C.6. Los números verticales indican el número del objeto clasificado. En todos los experimentos se utilizaron 30 imágenes de entrenamiento, esas imágenes se agruparon en clases las cuales le corresponden al objeto entrenado. Se esperaba que se obtuvieran 30 clases, sin embargo, estas

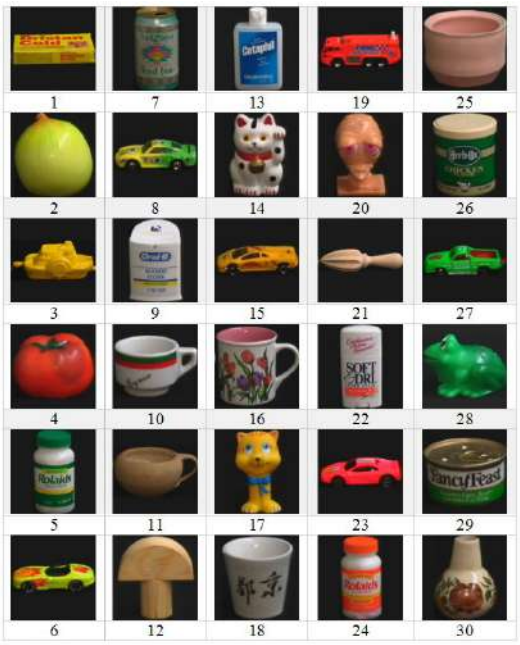


Figura C.2: Objetos de la base de datos COIL-100 utilizados para la clasificación.

no fueron suficientes para agrupar la información proveniente de todas las imágenes ingresadas a la red. Por esta razón se observa que más de un objeto se clusteriza en más de una clase a pesar de que las imágenes pertenecían al mismo objeto. Se hizo un conteo de las clases repetidas que corresponden a otros objetos a las cuales se le indica como falsos positivos pues clasifican la información de un objeto en otro objeto diferente.

La evaluación de la clasificación de cada uno de los experimentos se presenta en matrices de confusión: experimento 1 en C.3, experimento 2 en C.4, experimento 3 en C.5 y experimento 4 en C.6. En estas matrices de confusión, los números verticales indican el objeto que se está clasificando, en horizontal se encuentran los objetos a los que reconoció, se incluye la casilla n/a la cual indica que no clasificó en ninguna de las clases de los objetos y una casilla de porcentaje de reconocimiento para ese objeto. Al final de la tabla se presenta el porcentaje total del reconocimiento para ese experimento.

Según los resultados obtenidos, el experimento con mejor desempeño de clasificación fue el experimento 3 con un porcentaje de 77.34 %. Sin embargo, se obtuvo un porcentaje de 45.33 % de falsos positivos el cual fue el valor de clasificaciones erróneas mayor comparado a los otros experimentos. En este experimento se utilizó la red neuronal GCS y los atributos de niveles de grises. Además, se utilizaron para el entrenamiento 10 imágenes del objeto de diferentes vistas. Se puede esperar que un aumento de las imágenes de entrenamiento muestre un mejor resultado.

No obstante, al comparar el desempeño dependiendo de las características consideradas para la clasificación; entre los experimentos que utilizaron los histogramas de color, el que mejor desempeño realizó fue el experimento 2 con un porcentaje de 61.136 %. De este experimento se esperaba un desempeño pobre, ya que se eligieron para el entrenamiento las 30 primeras imágenes del objeto, a comparación del experimento 1 que utilizó 10 imágenes del objeto que incluía diferentes vistas.

Tabla C.5: Parámetros utilizados en los experimentos de clasificación.

Número de Experimento	**Objetos de Prueba**	**Imágenes de Entrenamiento**	**Imágenes de Evaluación**	**Vector de Entrada: Características**	**Vector de Entrada: Tamaño**	**Red Neuronal**
1	30	10	61	Histograma de color	768	GCS
2	30	30	41	Histograma de color	768	ART 2
3	30	10	61	Histograma de niveles de grises	256	GCS
4	30	10	61	Histograma de niveles de grises	256	ART 2

Si se considera el porcentaje de clasificación y el porcentaje recíproco de falsos positivos, las calificaciones finales para cada experimento quedarían: experimento 1 (45.16/)=55.08 %, experimento 2 (61.136/)=76.128 %, experimento 3 (77.34/)=66.005 %, experimento 4 (60.71/)=69.69 %. Lo cual califica al experimento 2 como mejor resultado. En este experimento se utilizaron histogramas de color, 30 imágenes de entrenamiento y 41 de evaluación, con la red neuronal ART 2.

Tabla C.6: Resultados del entrenamiento presentado en clases y correspondencia al objeto entrenado.

Conteo de neuronas correspondientes a los objetos				
	Experimento 1	**Experimento 2**	**Experimento 3**	**Experimento 4**
1	2	1	1	4
2	3	4	1	2
3	2	1	1	3
4	2	1	1	1
5	1	2	2	2
6	3	1	3	3
7	1	1	1	1
8	3	9	8	6
9	4	14	5	8
10	2	12	5	7
11	2	7	5	4
12	10	13	8	7
13	4	10	8	7
14	1	8	3	7
15	2	3	5	5
16	1	8	4	7
17	1	3	3	5
18	1	2	1	1
19	5	1	6	4
20	1	2	1	4
21	3	10	6	8
22	2	4	2	4
23	2	1	4	4
24	1	2	1	3
25	1	1	1	1
26	4	4	6	3
27	2	5	8	8
28	6	13	4	7
29	4	5	5	7
30	2	6	1	3
Falsos positivos	35 %	8.88 %	45.33 %	21.33 %

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	n/a	%
1	5											55																			1	**8.91**
2		30																	30												1	**49.18**
3			60									1																				**98.31**
4				61																												**100**
5					54				6			1																				**88.53**
6						4						17																			39	**6.56**
7							61																				1					**100**
8								0											7										3		51	**0**
9									6			34	1															19			1	**9.84**
10										5		1																11			44	**8.2**
11							2		1		32	8																	17		15	**52.46**
12									5			20											5		1				2	3	23	**32.8**
13													33										3								25	**54.1**
14												55		0																	6	**0**
15	2											14			0																45	**0**
16							4									57																**93.44**
17																	0														61	**0**
18																		61														**100**
19																			37								5				15	**60.66**
20																				61												**100**
21									50			2									1		4								4	**1.64**
22												1										50	6								3	**81.97**
23						4						48											0				6				3	**0**
24																								61								**100**
25																									59						2	**96.72**
26																										0					32	**0**
27								57																			0				4	**0**
28												21																19			21	**31.15**
29							6																						49		5	**80.33**
30									33																					0	28	**0**
																															Porcentaje total del clasificador	**45.16**

Figura C.3: Matriz de confusión para el experimento 1.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	n/a	%
1	0					39																									2	**0**
2		31																													10	**75.61**
3			41																													**100**
4				41																												**100**
5					15	26																										**36.59**
6						40																					1					**97.56**
7							38																									**92.68**
8								27																							13	**65.85**
9									2																						39	**4.88**
10										8																					33	**19.51**
11											16																			12	13	**39.02**
12												2																			39	**4.88**
13													10																		31	**24.39**
14														2																	39	**4.88**
15															39																2	**95.12**
16																31															10	**75.6**
17																	41															**100**
18																		41														**100**
19																			21												20	**51.21**
20																				39											2	**95.12**
21																					7	1									33	**17.07**
22																						20									21	**48.78**
23																							41									**100**
24																								41								**100**
25																									38						3	**92.68**
26																										40					1	**97.56**
27																											38				3	**92.68**
28																												1			40	**2.439**
29																													27		14	**65.85**
30																														14	27	**34.15**
																															Porcentaje total del clasificador	**61.136**

Figura C.4: Matriz de confusión para el experimento 2.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	n/a	%
1	48							5		8																						78.69
2		61																														100
3			60					1																								98.4
4				61																												100
5					61																											100
6	1					45		7	4	2			1													1						74
7							61																									100
8	4							17	1	4	2	1	3		5	3							8								14	28
9	1				1				47	1		1									1										6	77.05
10					10			2		48						1																78.69
11				1			7			1	41	4				3															3	67.2
12		8		3						10		18				5			1		4		1			1					7	29.51
13		2				7				10			34													1					5	55.7
14		26				1								34																		55.74
15					1							2			35	3															18	57.38
16																41							8								11	67.2
17						5									3		53															87
18																		56					5									91.8
19																			37				5								1	60.66
20								3											1	57												93.44
21												2									52										6	85.25
22																						61										100
23							1	2		3						1							54									88.52
24																								61								100
25																									61							100
26																										45					14	73.8
27				1	2																						33	2			7	54.1
28											5	2				2			5									23			11	37.7
29				4							20					2													49			80.33
30				4							8																			61		100
																															Porcentaje total del clasificador	77.34

Figura C.5: Matriz de confusión para el experimento 3.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	n/a	%
1	30																1														30	49.18
2		49																													12	80.32
3			46																								1				14	75.41
4				61																												100
5					15								1																		11	80.32
6						54									1																6	88.52
7							61																									100
8								25					1						1		2						1			1	30	40.98
9									19			1																			41	31.15
10										15																					46	24.6
11											39					1				1							1				18	63.93
12									1			22									1										37	36.06
13													25																		36	41
14														18	3		2														38	29.5
15								4							36																21	59
16																4				20			1							3	33	6.56
17																	25		9												35	41
18																		59		2												96.72
19																9			40												12	65.58
20																				60											1	98.36
21																					26										35	42.62
22																					17	14									30	22.95
23																			3				48								10	78.69
24																								61								100
25																									61							100
26																										61						100
27																			3								21				37	34.43
28																												38			23	62.3
29																				4									30		27	49.18
30																														14	47	22.95
																															Porcentaje total del clasificador	60.71

Figura C.6: Matriz de confusión para el experimento 4.

Bibliografía

[1] J. Collado, "Mitología universal: breve historia de la mitología", J Collado, 1933, pp. 470

[2] A. Ollero, "Robótica: manipuladores y robots móviles", Alfa omega Grupo Editor, México, 2007

[3] Oxford English Dictionary, "Robotics", disponible en línea, http: //www.oed.com, último acceso octubre 2015

[4] A. Barrientos, L. Penín, C. Balaguer, R. Aracil, "Fundamentos de robótica", McGraw-Hill Interamericana de España, 2007

[5] Y. dos Santos, A. C. de Pina, A. C. de Pina, "Reserch on bipedal robots applied to society", DINCON, 9th Brazilian Conference on Dynamics Control and their Applications, 2010

[6] J.J Craig, "Robótica", Tercera Edición, Pearson Prentice Hall, México, 2006

[7] V. I. Rybak, "Enciclopedia de cibernética", Kiev, vol. 2, 1973

[8] International Federation of Robotics, "Executive summary: world robotics 2014", p 11-24

[9] WowWee, Robot MiP, disponible en línea, http://www.wowwee.com/mip último acceso octubre 2015

[10] Adele, Feeling Robots, PARO, disponible en línea, http://www.adelerobots.com/es/nuka/ último acceso octubre 2015

[11] iRobot, Vaccum Cleaning, "Introducing roomba", disponible en línea, http://www.irobot.com/For-the-Home/Vacuum-Cleaning/Roomba.aspx, último acceso octubre 2015

[12] Care-O-bot 4, "The new modular service robot generation", disponible en línea, http://www.care-o-bot-4.de/, último acceso octubre 2015

[13] da Vinci Surgery, "Changing the experience of surgery", disponible en línea, http://www.davincisurgery.com/, último acceso octubre 2015

[14] Boston Dynamics, "BigDog: the most advanced rough-terrain robot on earth", disponible en línea, http://www.bostondynamics.com/robot_ bigdog.html, último acceso octubre 2015

[15] SONY Corporation,"QRIO SDR-4XII", disponible en línea, https://www.sony.net/SonyInfo/CorporateInfo/History/sonyhistory-j.html, último acceso octubre 2015

[16] FUJITSU Automation Limited, "Miniature humanoid robot hoap-3", disponible en línea, http://www.automatic.fujitsu.com, último acceso octubre 2015

[17] ROBOTIS, "Educational robot kit: bioloid", disponible en línea, http://www.robotis.com, último acceso octubre 2015

[18] Aldebaran Robotics, "Nao documentation", disponible en línea, https://www.aldebaran.com/en/humanoid-robot/nao-robot, último acceso octubre 2015

[19] G. Bugmann, "The what and when of service robotics", Industrial Robot, 2005, pp.437

[20] V. Kumar, G. Bekey, Y. Zheng, "Industrial, personal, and service robots", International Assessment of Research and Development in Robotics, WTEC, 2006, pp. 50-62

[21] M. Hirose, K. Ogawa, "Honda humanoid robots development", Philosophical transactions of the royal society A, 2007, pp. 11-19

[22] H. Hirukawa, F. Kanehiro, K. Kaneko, S. Kajita, "Humanoid robotics platforms developed in HRP", Robotics and Autonomus Systems 48, Elsevier, 2004, pp. 165-175

[23] Competencia Internacional RoboCup, disponible en línea, http://www.robocup.org/about-robocup/, último acceso octubre 2015

[24] Competencia Internacional de Robótica de Servicio, RoboCup@Home, disponible en línea, http://www.robocupathome.org/, último acceso octubre 2015

[25] DARPA Robotics Challenge, disponible en línea, http://www.theroboticschallenge.org, último acceso octubre 2015

[26] United Nations Population Division, "World population prospects, the 2002 revision", 2003

[27] CONAPO, A. Ojeda, "Aspectos generales de los resultados de las proyecciones de población", Disponible en línea, http://www.conapo.gob.mx/es/CONAPO/, último acceso octubre 2015

[28] D. P. Valencia, L. E. González, "Diseño e implementación de un prototipo de robot asistente para personas con discapacidad motriz y adultos mayores, basado de inteligencia artificial", Tesis de Licenciatura, Universidad Politécnica Salesiana, Ecuador, 2014

[29] A. Grasp, "An introduction to wavelets", Computational Science Engineering IEEE, vol. 2 no. 2, 1995, pp. 50-61

[30] P. N. Belhumeur, J. P. Hespanha, D. J. Kriegman, "Eigenfaces vs. fisherfaces: recognition using class specific linear projection", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no. 7, 1997

[31] Z. Qiu-bo1, Z. Jie, T. Chun-ya, "Design and implement of entertainment and competition humanoid robot", IIETA, Review of Computer Engineering Studies vol. 2, no. 1, 2015

[32] I. Elizaga, "Localización e interacción con objetos mediante visión artificial con el robot nao", Tesis de Licenciatura, Universidad Carlos III de Madrid, 2012

[33] Robot Operating System (ROS), disponible en lineal, http://www.ros.org/about-ros/, último acceso octubre 2015

[34] S. Morante, "Interfaz y librería para visión artificial, navegación y seguimiento en robótica", Tesis de Licenciatura, Universidad Carlos III de Madrid, 2012

[35] H. Bay, A. Ess, T. Tuytelaars, L.Van Gool, "SURF: Speeded up robust features", Computer Vision and Image Understanding (CVIU), vol. 110, no. 3, 2008, pp. 346-359

[36] Y, Cheng, "Mean shift, mode seeking, and clustering", Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol.17, no.8, 1995, pp. 790-799

[37] A. Burton, J. Radford, "Thinking in perspective: critical essays in the study of thought processes", Methuen, London, 1978

[38] D. Lorencik, J. Ondo, P. Sincak, H. Wagatsuma, "Cloud-based object recognition for robots", Springer-Verlag Berlin Heidelberg, adfa, 2011

[39] D. G. Lowe, "Distinctive image features from scale-invariant keypoints", International Journal of Computer Vision, 2004

[40] A. Bodnárová, "The mf-artmap neural network", Latest Trends in Applied Informatics and Computing, 2012, pp. 264-269

[41] M. Erazo, D. Matamoros, J. Minchala, D. Paillacho, "Diseño e implementación de un robot móvil teleoperado en base al reconocimiento de forma y movimiento de objetos", Tesis de Licenciatura, Escuela Superior Politécnica del Litoral, Ecuador, 2007

[42] R. A. Schulz, "Visión activa en un robot humanoide antropomorfo", Tesis de Ingeniería, Universidad de Chile, Santiago de Chile, 2010

[43] A. Blanco, "Optimización de la adquisición de modelos de oponentes en la robocup- categoría de simulación 2d", Tesis de Licenciatura, Universidad Carlos III de Madrid, 2012

[44] J. Aristondo, "Algoritmo de reconocimiento de forma y color para una plataforma robótica", Tesis de Maestría, Universidad del País Vasco, 2010, www.ccia-kzaa.ehu.es

[45] A. V. López, "Detección de trayectorias y reconocimiento de objetos regulares para el control por visión artificial de un robot móvil", Tesis de Licenciatura, Instituto Politécnico Nacional, México, 2007

[46] G. Welch, G. Bishop, "An introduction to the kalman filters", ACM, Inc., 2001

[47] A. Peña, "Módulo de visión artificial del robot humanoide hoap-3. Aplicación al seguimiento de objetos móviles", Tesis de Licenciatura, Universidad Carlos III de Madrid, España, 2010

[48] R. Pérez, "Una introducción al cómputo neuronal artificial", El Cid Editor, Primer edición, 2012

[49] W. L. Monar, "Aplicación de las redes neuronales al reconocimiento de objetos en robots manipuladores", Tesis de Maestría, Escuela Politécnica Nacional, Ecuador, 2014

[50] D. E. Rumelhart, G. E. Hinton, R. J. Williams, "Learning representations by back-propagation errors", Nature Publishing Group, 1986, pp. 533-536

[51] M. Peña, R. Osorio, "Un proceso de aprendizaje para reconocimiento de objetos en línea en tareas robotizadas", Sistemas, Cibernética e Informática vol. 1, no. 2, 2013

[52] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, D. B. Rosen, "Fuzzy artmap: a neural network architecture for incremental supervised learning of analog multidimensional maps", IEEE Transactions on Neural Networks, vol. 3, no. 5, 1992, pp. 698-713

[53] E. Sobrado, J. C. Tafur, "Sistema de visión artificial para el reconocimiento y manipulación de objetos utilizando un brazo robot", Tesis de Maestría, Pontificia Universidad Católica del Perú, 2003

[54] J. de Vries, "Object Recongition: A shaped base approach using artificial neural networks", Tesis de Maestría, University of Utrecht, Países Bajos, 2006

[55] G. Battiston, "Collaborative action planning for humanoid robots exchanging a small object", Tesis de Maestría, Kungliga Tekniska Högskolan, Suecia, 2014

[56] R. C. Gonzalez, R. E. Woods, "Digital image processing", Prentice Hall, Second Edition, 2002, pp. 25-30

[57] V. Ramanathan, A. Pinz, "Active object categorization on a humanoid robot", VISAPP-International Conference on Computer Vision Theory and Applications, 2011

[58] C. F. Tsai, "Bag-of-words representation in image annotation: a review", International Scholarly Research Network, ISRN Artificial Intelligence, 2012

[59] E. Krause, M. Zillich, T. Williams, M. Scheutz, "Learning to recognize novel objects in one shot through human-robot interactions in natural language dialogues", Association for the Advancement of Artificial, 2014, Intelligence (www.aaai.org)

[60] K. Litomisky, "Consumer rgb-d cameras and their applications", Computer Science Alumni, University of California, 2012 http://alumni.cs.ucr.edu/

[61] S. Metzler, "Visual robot detection and free space recognition for standar platform league", Tesis de Licenciatura, University of Bonn, 2011

[62] I. Schwarz, M. Hofmann, O. Urbann, S. Tasse, "A robust and calibration-free vision system for humanoid soccer robots", Symposium RoboCup 42, 2015

[63] A. Bolotnikova, "Melioration of color calibration, goal detection and self-localization systems of nao humanoid robots", Tesis de Licenciatura, University of Tartu, 2015

[64] A. Doucet, A. M. Johansen, "A tutorial on particle filtering and smoothing: fifteen years later", Oxford Handbook of Nonlinear Filtering, 2011

[65] M. Llofriu1, G. Tejera, A. Barrera, A. Weitzenfeld, "A humanoid robotic platform to evaluate spatial cognition models", Proceedings of 8th Workshop on Humanoid Soccer Robots, 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids), Atlanta, GA, 2013

[66] J.J. Leonardand, H.F. Durrant, "Simultaneous map building and localization for an autonomous mobile robot", Intelligent Robots and Systems '91, Íntelligence for Mechanical Systems, Proceedings IROS '91, IEEE/RSJ International Workshop on, vol.3, 1991, pp. 1442-1447

[67] M. Pereyra, E. Destefanis, "Reconocimiento de objetos para control de calidad mediante la descomposición en figuras geométricas", 2º Congreso Nacional de Ingeniería Informática/ Sistemas de Información, Universidad Tecnológica Nacional, Argentina, 2014

[68] L. Gelsi, C. A. Di Caro, H. A. Fernández, "Reconocimiento rápido de objetos", Anales AFA, vol. 22 pp. 95-97, 2011

[69] A. Escudero, "Reconocimiento de objetos", Tesis de Licenciatura, Universitat de Barcelona, España, 2009

[70] B. Albanesi, N. Funes, F. Chichizola, L. Lanzarini, "Reconocimiento de objetos en video utilizando sift paralelo", XVI Congreso Argentino de Ciencias de la Computación, 2010

[71] J. Sanz, "Reconocimiento de objetos por descriptores de forma", Tesis de Licenciatura, Universitat de Barcelona, España, 2008

[72] Standford Encyclopedia of Philosophy, "Fuzzy logic", disponible en línea, http://plato.stanford.edu/entries/logic-fuzzy/, último acceso octubre 2015

[73] R. E. Schapire, "Explaning adaboost", Princeton University, Dept. of Computer Science, USA, disponible en línea http://rob.schapire.net/papers/explaining-adaboost.pdf, último acceso octubre 2015

[74] J. D. Gómez, "Proceso de reconocimiento de objetos, asistido por computador, aplicando gases neuronales y técnicas de data mining", Revista Científica y Tecnológica de la Facultad de Ingeniería, Universidad Distrital Francisco José de Caldas, vol.12 no.1, 2007

[75] B. Fritzke, "A growing neural gas network learns topologies", Advances in Neural Information Processing Systems 7, 1995, pp. 625-632

[76] B. Fritzke, "Growing cell structures, a self-organizing network for unsupervised and supervised learning", Elsevier, Neural Networks, vol. 7, no. 9, 1994, pp. 1441-1460

[77] T. Martinez, K. Schulten,"A "neural gas" network learns topologies", Artificial Neural Networks, ELSEVIER Science Publishers B. V., 1991, pp.397-402

[78] P. F. Alcantarilla, A. Bartoli, A. J. Davison, "Kaze features", ECCV, Springer-Verlag Berlin Heidelberg, 2012, pp. 214-227

[79] M. Agrawal, K. Konolige, M. R. Blas, "Censure: center surround extremas for realtime features detection and matching", Computer Vision, ECCV, vol. 5305, 2008, pp. 102-115

[80] P. Perona, J. Malik, "Scale-space and edge detection using anistropic diffusion", IEEE Transactions on Pattern Análisis and Machine Inteligece, vol. 12, no. 7, 1990, pp.629-639

[81] P. F. Alcantarilla, J. Nuevo, A. Bartoli, "Fast explicit diffusion for accelerated features in nonlinear scale spaces", In British Machine Vision Conference (BMVC), 2013

[82] MIA, "Fast explicit diffusion (fed) and fast-jacobi (fj)", http://www.mia.uni-saarland.de/Research/SC_ FED.shtml, disponible en lineal, último acceso octubre 2015

[83] X. Yang, K. T. Cheng, "LDB: An ultra-fast feature for scalable augmented reality", In IEEE and ACM Intl. Sym. on Mixed and Augmented Reality (ISMAR), 2012

[84] S. Leutenegger, M. Chli, R. Y. Siegwart, "BRISK: Binary robust invariant scalable keypoints", In IEEE Intl. Conf. on Computer Vision (ICCV), 2011, pp. 2548-2555

[85] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, "Orb: an efficient alternative to siftt or surf", In IEEE Intl. Conf. on Computer Vision (ICCV), 2011, pp. 2564-2571

[86] S. Salamanca, A. Adán, C. Cerrada, M. Adán, P. Merchán, E. Pérez, "Reconocimiento de objetos de forma libre a partir de los datos de rango de una vista parcial usando cono curvaturas ponderadas", Revista Iberoamericana de Automática e Informática Industrial, vol. 4, no. 1, 2007

[87] A. Adán, M. Adán, "A flexible similarity measure for 3d shapes recognition", IEEE Transactions on Pattern Analysis and Machine Intelligence 26, 2004, pp. 1507-1520

[88] H. Chen, B. Bhanu, "3d free-form object recognition in range images using local surface patches", ELSEVIER, Pattern Recognition Letters 28, 2007, pp. 1252-1262

[89] S. Z. Li, A. J, "Encyclopedia of biometrics", Springer, 2009, pp. 949

[90] A. E. Johnson, M. Hebert, "Using spin images for efficient object recognition in cluttered 3d scenes", IEEE Transactions on Pattern Analysis and Machine Intelligence 21, 1999, pp. 433-449

[91] S. Correa, L. Shapiro, "A new signature-based method for efficient 3-d object recognition", In Proc. IEEE Conf. on Computer Vision and Pattern Recognition, vol. 1, 2001, pp. 769-776

[92] C.I. Connolly, "The determination of next best views", Proceedings: Conference on Robotics and Automation, IEEE, 1985, pp. 432

[93] C. Potthast, G.S. Sukhatme, "A probabilistic framework for next best view estimation in a cluttered environment", J. Vis. Commun, 2013, http://dx.doi.org/10.1016/j.jvcir.2013.07.006

[94] M. Krainin, B. Curless, D. Fox, "Autonomous Generation of Complete 3D Object Models Using Next Best View Manipulation Planning", ICRA, 2011

[95] S. Kriegel, T. Bodenmüller, M Suppa, G. Hirzinger, "A surface-based next-best-view approach for automated 3d model completion of unknown objects", ICRA, 2011

[96] E. Dunn, J. van den Berg, J. M. Frahm, "Developing visual sensing strategies through next best view planning", IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009

[97] J. I. Vasquez-Gomez, L. E. Sucar, R. Murrieta-Cid, E. Lopez-Damian, "Volumetric next-best-view planning for 3d object reconstruction with positionning error", International Journal of Advance Robotic Systems, 2014

[98] J. I. Vasquez-Gomez, L. E. Sucar, R. Murrieta-Cid, "View/state planning for three-dimensional object reconstruction under uncertainty", Auton Robot, Springer, 2015

[99] S. Grossberg, "Competitive learning: from interactive action to adaptative resonance", Cognitive Science 11, 1987, pp. 23-63

[100] Cuentos Cuánticos, disponible en línea, http://cuentos-cuanticos.com/2011/12/15/robotica-estimacion-de-posicion-por-odometria/, último acceso diciembre 2016

[101] Python, disponible en línea, https://www.python.org/, último acceso octubre 2015

[102] Open Source Computer Vision, disponible en línea, http://opencv.org/, último acceso octubre 2015

[103] Webots, disponibles en línea, http://doc.aldebaran.com/1-14/software/webots/webots_index.html, último acceso octubre 2015

[104] W. Martínez, "Desarrollo de los medios básicos para diseñar sistemas robóticos de rehabilitación motora fina por movimientos funcionales", Tesis de Maestría, Universidad Tecnológica de la Mixteca, México, 2013

[105] J. Ortiz, "Diseño y realización de maxcota, un robot inteligente para niños", Tesis de Licenciatura, Universidad Tecnológica de la Mixteca, México, 2004

[106] R. A. Gracía, "Desarrollo de un robot tipo ballbot para aplicaciones de control", Tesis de Licenciatura, Universidad Tecnológica de la Mixteca, México, 2014

[107] H. D. Alfaro, "Diseño de un robot controlado por computadora, orientado a la enseñanza pedagógica", Tesis de Licenciatura, Universidad Tecnológica de la Mixteca, México, 1997

[108] J. A. Chavez, "Diseño y construcción de un brazo robótico pedagógico jugador de gato, dotado de un sistema básico de visión artificial", Tesis de Licenciatura, Universidad Tecnológica de la Mixteca, México, 1999

[109] F. Lopez, "Diseño y construcción de una plataforma de robot móvil para ambientes externos con comunicación inalámbrica", Tesis de Licenciatura, Universidad Tecnológica de la Mixteca, México, 2011

[110] I. Arroyo, "Evaluación de dos técnicas de reconocimiento de patrones para su implementación en el simulador de pilotaje automático (pa-135, nm-79 chopper) de talleres stc metro de la ciudad de méxico", Tesis de Maestría, Universidad Tecnológica de la Mixteca, México, 2013

[111] I. G. López, "Obtención de la forma de una superficie asférica a partir de radios de curvatura locales utilizando algoritmos genéticos", Tesis de Licenciatura, Universidad Tecnológica de la Mixteca, México, 2000

[112] I. S. Jiménez, "Control de temperatura de un horno eléctrico mediante lógica difusa", Tesis de Licenciatura, Universidad Tecnológica de la Mixteca, México, 2012

[113] A. A. Casanova, "Control difuso del quadrotor ar. Drone 2.0 para el seguimiento autónomo de trayectorias", Tesis de Licenciatura, Universidad Tecnológica de la Mixteca, México, 2015

[114] M. L. Cruz, "Implementación y análisis comparativo de una red neuronal art2 y una Backpropagation, aplicadas al reconocimiento de patrones en imágenes digitales", Tesis de Licenciatura, Universidad Tecnológica de la Mixteca, México, 2001

[115] N. Peláez, "Aprendizaje no supervisado y el algoritmo wake-sleep en redes neuronales", Tesis de Licenciatura, Universidad Tecnológica de la Mixteca, México, 2012

[116] M. P. García. "Implementación de redes neuronales sobre lógica reconfigurable", Tesis de Maestría, Universidad Tecnológica de la Mixteca, México, 2015

[117] G.A. Baxes, "Digital image processing, principles and applications", J. Wiley, 1994

[118] Fu, Gonzalez, Lee, "Robotics: control, sensing, vision, and intelligence", McGraw-Hill, 1987

[119] J. R. Magnus, H. Neudecker, "Matrix differential calculus with aplications in statistics end econometrics", John Wiley & Sons, Tercera Edición, 2007, pp. 114

[120] H. Scharr, "Optimal operators in digital image processing", Tesis de Doctorado, Rupertus Carola University of Heidelberg, Germany, 2000

[121] Ramírez Q. Juan A., Chacón M., "Redes neuronales artificiales para el procesamiento de imágenes, una revisión de la última década", RIEE&C, Revista de Ingeniería Eléctrica, Electrónica y Computación, vol. 9, no. 1, 2011

[122] E. R. Davies, "Computer and machine visión, theory and algorithms practicalities", Elsevier Inc., 4th Edition, 2012

[123] Shruti B. Hiregoudar, Manjunath. K, K.S.Patil, "A survey research summary on neural networks", IJRET: International Journal of Research in Engineering and Technology eISSN, vol. 03, 2014, http://www.ijret.org 385

[124] S. Kajita, H. Hirukawa, K. Harada, K, Yokoi, "Introduction to humanoid robotics", Springer, 2014

[125] Real Academia de la Lengua Española, disponible en línea, http://dle.rae.es/, último acceso diciembre 2016

[126] Que es la odometria (2012), Revista ARQHYS.com, disponible en línea, http://www.arqhys.com/contenidos/quees-odometria.html, http://www.arqhys.com/contenidos/quees-odometria.html, último acceso diciembre 2016

[127] M. T. Hagan, H. B. Demuth, M. Hudosn, O. de Jesús, "Neural networks design", 2nd edition, eBook, Copyright by Martin T. Hagan and Howard B. Demuth, hagan.okstate.edu/nnd.html pp. 756-782

[128] G. A. Carpenter, S. Grossberg, "A massively parallel architecture for self-organizing neural pattern recognition machine", Computer Vision, Graphics, and Image Processing 37, 1987, pp. 54-115

[129] P. Corke, "Robotics, vision and control, fundamental algorithms in matlab", Springer, Primera Edición, (2011), pp. 335-450

[130] G. A. Carpenter, S. Grossberg, "ART 2: self-organization of stable category recognition codes for analog input patterns", Applied Optics, Vol. 26, No. 23, 1987, 4919-4930

[131] SoftBank Robotics, disponible en línea, https://www.ald.softbankrobotics.com/en/cool-robots/nao/find-out-more-about-nao, último acceso febrero 2017

[132] N. Kofinas, "Forward and Inverse Kinematics for the NAO Humanoid Robot", Tesis, Technical University of Crete, July 2012

[133] D. Gouaillier, V. Hugel, P. Blazevic, B. Maisonnier, "Mechatronic design of NAO humanoid", IROS 2009, pp. 769-774