ESTRATEGIA DE NAVEGACIÓN PARA LA PLATAFORMA ROBÓTICA NAO

USANDO LANDSMARKS MEDIANTE RECONOCIMIENTO DE FIGURAS GEOMÉTRICAS

NAVIGATION STRATEGY FOR THE NAO ROBOTIC PLATFORM

USING LANDSMARKS THROUGH RECOGNITION OF GEOMETRIC FIGURES

Cumaco P. Jose Nicolas, Mendoza R. Williams Raul
Montiel A. Holman

Resumen*:* El presente documento muestra el desarrollo de la estrategia de navegación para el manejo de la plataforma robótica NAO, mostrando el paso a paso y los resultados obtenidos. Se muestra el desarrollo del algoritmo para la detección de los landsmark, que son tres figuras geométricas básicas de color azul, así como el proceso de integración con la plataforma robótica NAO y el desarrollo de la toma de decisiones por parte del robot para ejecutar un movimiento de acuerdo a la figura detectada. Finalmente, se realiza un análisis de tiempos de las figuras detectadas y su procesamiento, y se muestran los resultados obtenidos.

Palabras clave: Detección, movimiento, algoritmo, landmark, procesamiento, filtrado.

Abstract: The current document shows a development of the strategy of navigation for the management of the robotic platform NAO, showing the step by step and the corresponding results. Shows the development of the algorithm to the detection of landsmarks, which are the geometrical figures of color blue, as the integration process with the robotic platform NAO and the development of the decision making to the robot to execute a movement according to the figure detected. Finally, realize analysis of the times of the detected figures and their processing, and shows the obtained results.

Key Words: Detection, movement, algorithm, landmark, processing.

1 Introducción

La detección de elementos en el entorno es un campo de trabajo dentro del procesamiento de imágenes que constantemente evoluciona, debido a las tecnologías que se desarrollan y los procesos que se implementan para esta detección. Una aplicación de esta detección es el uso de landsmark, que puede servir de orientación en un espacio determinado o para la clasificación de elementos de acuerdo a características definidas como pueden ser forma, tamaño o color.(Romero Acero, Marín Cano, & Jiménez Builes, 2015)

De esta manera, se han desarrollado diferentes métodos y técnicas de procesamiento que permite realizar el desarrollo de aplicaciones en campos como medicina, agricultura, electrónica y demás. Algunos de estas técnicas de procesamiento son los métodos de esqueletización, que permite obtener el esqueleto de determinadas figuras u objetos (Méndez, Villa, & Cisneros Barahona, 2017). Estos métodos consisten en tomar la cantidad mínima de pixeles de una figura, manteniendo la forma de la misma. De esta forma, el resultado será un esqueleto delgado, centrado y continuo, que permita simplificar el análisis de la imagen (Carranza Athó, 2006).

Existen varios métodos como lo son los diagramas de Voroni, que consiste en separar la figura en pequeñas secciones y determinar puntos que se acerquen hasta los bordes, generando líneas que atraviesan la figura. Otro método es el algoritmo de Zhang-Suen, debe buscar los pixeles o puntos vecinos y determinar cuales pertenecen al esqueleto o cuales pueden ser eliminados.

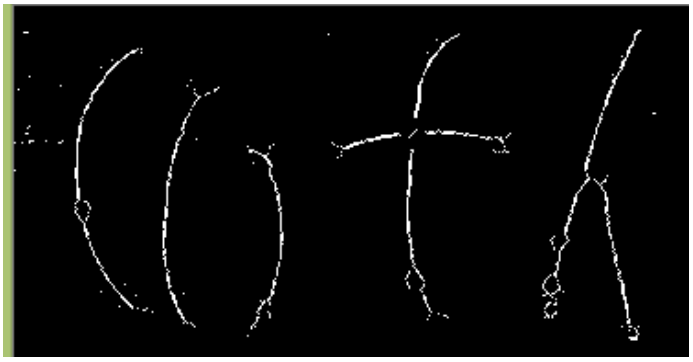


Figura 1. Ejemplo de algoritmo de esqueletización

Para el desarrollo de este método es necesario procesar previamente la imagen, convirtiendo la imagen a blanco y negro, ya que este método detecta las formas blancas sobre un fondo negro, debido a que los valores binarios que se generan en la imagen (1 si es blanco, 0 si es negro), simplifican la detección del elemento en cuestión. También se debe aplicar, en el procesamiento previo, filtros a la imagen, que nos permita separar figuras de un color específico, como pueden ser colores primarios o RGB, o aplicar filtros que reduzcan elementos no deseados en la imagen, como se observa en la figura 1.(Botero Valencia & Delgado Trejos, 2009)

Otro método utilizado para el procesamiento de imágenes es el algoritmo de Canny, utilizado principalmente en la detección de bordes de una imagen (Delgado-Gutiérrez, Herrera-Guillén, Medina-Barragán, & Corredor–Gómez, 2017). Este algoritmo consiste en, inicialmente, suavizar la imagen a través de un filtro gaussiano. Este filtro es un proceso matemático, donde se utiliza un promedio de los valores de los pixeles en un arreglo definido utilizando la desviación estándar, que permite suavizar la imagen, eliminando posibles ruidos o elementos no deseados en la imagen, que pueden ser detectados posteriormente por el algoritmo (María Calderón & González Gómez Gabriela Kramer Bustos Doricela Gutiérrez Cruz, 2007).

Con la imagen filtrada, se procede a obtener los bordes de la imagen, calculando el valor del gradiente de la imagen filtrada, generando un valor de histéresis de umbral y suprimiendo los

máximos locales generados en este proceso y si es necesario, completar los espacios vacíos que se puedan generar al realizar la supresión. Esto con el fin de completar el borde de la imagen y evitar elementos incompletos (Orozco, Ruiz Salguero, & Jasnoch, 2005).

Figura 2. Aplicación del algortimo de Canny

Como complemento, la aplicación de erosión y dilatación en la imagen antes de aplicar el algoritmo de Canny ayuda a reducir elementos parásitos o ruido en la imagen que puede llegar a ser detectado.(Jaramillo, Fernández, & Martínez De Salazar, 2000)

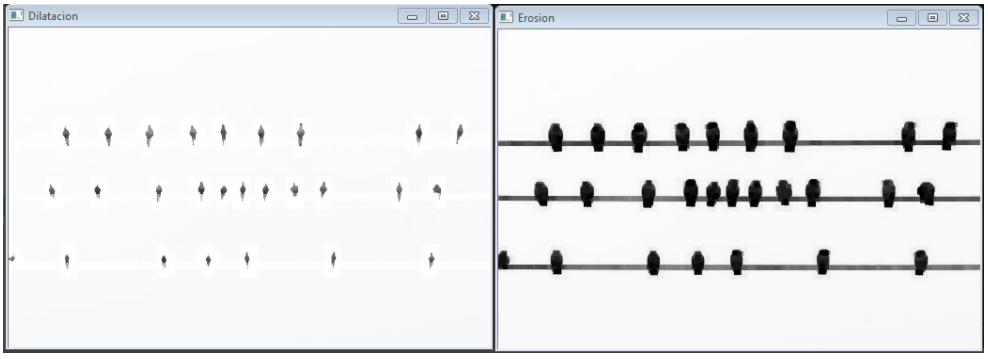


Figura 3. Erosión y dilatación de una imagen

De manera similar al método de esqueletización, es necesario procesar la imagen a blanco y negro, ya que, al trabajar con una matriz binaria, de ceros y unos, el proceso matemático se simplifica, y permite obtener con facilidad la imagen a procesar, por lo que, si se requiere

procesar elementos de un color especifico, es necesario aplicar una máscara o filtro de color que permita aislar los objetos a detectar y luego aplicar el algoritmo previamente indicado.

Para el tratamiento de figuras circulares, se puede aplicar la transformada de Hough, técnica que permite ubicar figuras en una imagen, sin importar si son rectas o círculos (Giraldo, González, & Camargo, 2011). Para ser aplicado, se debe pre procesar la imagen, convirtiendo la imagen en blanco y negro, para realizar la correcta detección de los bordes. También, es necesario aplicar filtros a la imagen, como puede ser filtros gaussianos, erosión y dilatación o algún otro filtro que reduzca los elementos parásitos o ruidos detectados en la imagen (Cuevas, Oliva, Osuna-enciso, & Wario, 2011).

Esta transformada consiste en tomar los puntos de la imagen como si fuera un plano cartesiano y transformarlo en un espacio de parámetros. Con este espacio definido, se puede detectar los bordes de dos maneras. La primera es aplicar rectas, para determinar los puntos que cruzan los bordes de la imagen a trabajar. Así, se puede obtener un acumulador que indica si la recta ha pasado por este punto, y con los acumuladores más altos se pueden determinar los bordes de la imagen. (Joo, 2006)

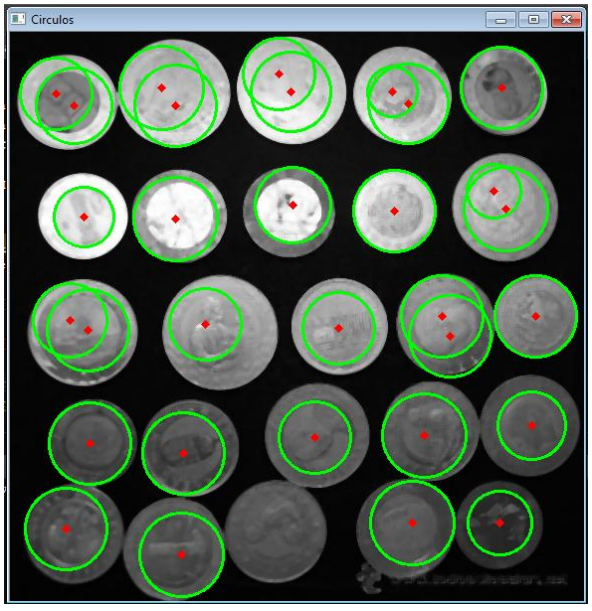


Figura 4. Detección de círculos con transformada de Hough

El inconveniente que presenta es al detectar una línea vertical, ya que al aplicar la ecuación de la recta, los valores se indeterminan. La segunda manera es utilizar coordenadas polares, que permite obtener la distancia de la línea con el origen y el ángulo de la misma, para hacer pasar curvas por los puntos, obtener los acumuladores y detectar los bordes de la imagen.(Jose Armando Ruiz Leyva, s. f.)

Este método, al implicar tantos cálculos y especialmente por los acumuladores, puede presentar errores en la detección de los elementos, al considerar ruidos o elementos parásitos como parte de los bordes a trabajar, además de generar gran carga computacional para realizar el análisis de la imagen.

Estas técnicas mencionadas anteriormente pueden ser ubicadas y utilizadas con la biblioteca OpenCV, un compilado de librerías gratuitas para el procesamiento y tratamiento de la imagen diseñado en el lenguaje de programación C++, pero puede ser usado en diferentes lenguajes de programación como lo es Python, que facilita el desarrollo y aplicación de las mismas. Esta biblioteca puede ser descargada de internet de manera gratuita y puede ser utilizada directamente en la consola de Python o puede ejecutarse a través de algún procesador de texto como puede ser Pycharm, software diseñado para la generación de códigos en Python, aunque puede ser usado para otros lenguajes. A través de este software, se puede instalar librerías adicionales que son requeridas para que OpenCV funcione, tales como numpy, librería desarrollada para el tratamiento y manejo de matrices y arreglos de datos, o matplotlib, librería utilizada para graficar arreglos de datos en Python.

2 Metodología a utilizar

Para el desarrollo de la estrategia de navegación, se utiliza la plataforma robótica NAO, que puede ser programada en el lenguaje de programación Python, con el cual se puede utilizar la

biblioteca OpenCV, y cuenta con dos cámaras que se puede configurar en su resolución, espacio de color a trabajar y características específicas como luz, contraste, y demás. La imagen se obtiene de la cámara de la parte superior del robot, y se configura a una resolución de 640x480 pixeles y el espacio de color se configura como RGB.

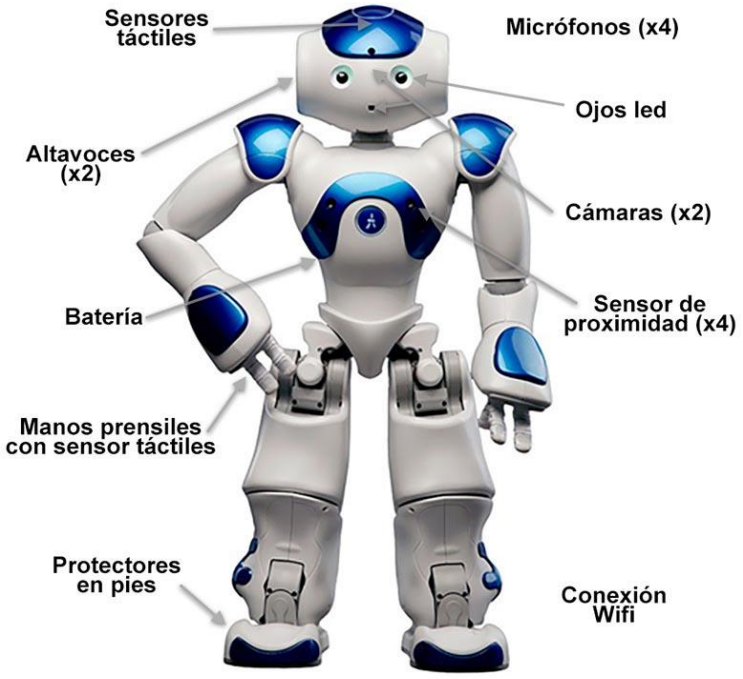


Figura 5. Robot NAO. Fuente: Alisys.net

A continuación, se muestra el diagrama de bloques que expone la metodología propuesta:

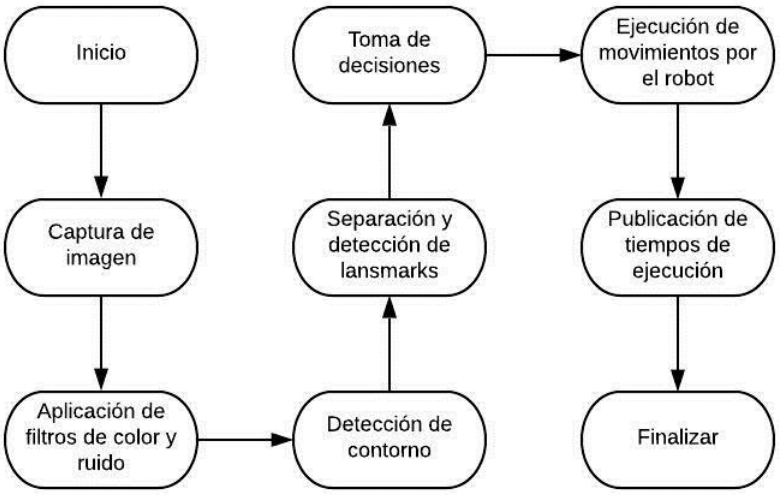


Figura 6. Diagrama de bloques

Los landsmark a detectar son tres figuras geométricas básicas, que son el círculo, el cuadrado y triangulo. Se utilizan estas figuras debido a que son formas básicas, regulares, que pueden encontrarse en elementos comunes y el procesamiento de la imagen se simplifica. Estas figuras son de color azul debido a que es una de las componentes de color básicas de la imagen al utilizar imágenes RGB. Debido a que el triángulo puede apuntar en 4 direcciones diferentes, la detección de los landsmarks es de seis figuras en total. Las figuras se observan en la figura 7.

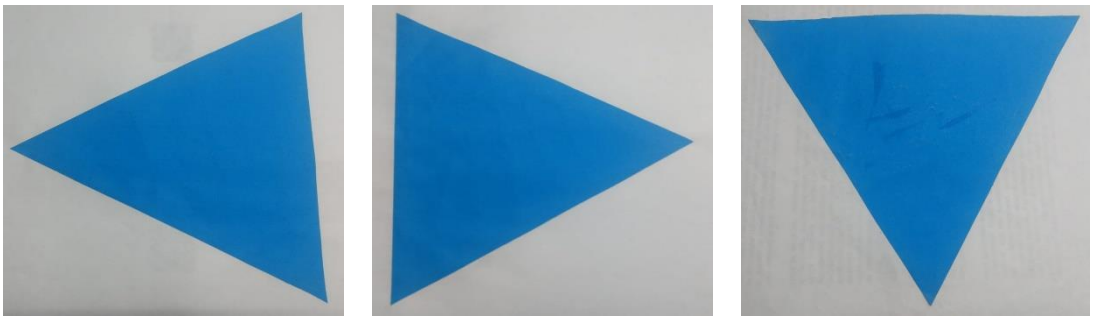


Figura 7. Figuras a detectar

Para el desarrollo de la estrategia de navegación, el algoritmo se desarrolla en el software Pycharm, ya que permite imprimir mensajes en la consola, lo que facilita la validación y depuración del mismo. Inicialmente se genera un algoritmo que detecte las figuras o landsmarks a utilizar. Para ello, se procede a capturar la imagen de la cámara superior del robot, como se observa en la figura 8. El robot, a través de su librería instalada previamente, envía a información de la imagen, es decir, se debe generar el proceso de creación de la imagen. Para realizar este proceso, inicialmente se comunica con el robot a través de la dirección IP que indica, se genera un objeto y se configura con los parámetros de resolución y espacio de color, luego se obtienen los datos de la cámara seleccionada y se guardan en un vector. Este vector posee la información del tamaño de la imagen, así como la información de cada pixel, por lo que, a través de la librería PIL, se genera la imagen definiendo el ancho y largo de la imagen y la información a contener.

Al almacenar la imagen, se convierte a escala de grises y a escala de color HSV, para facilitar el filtrado de color. Posteriormente, se utiliza un filtro de color para separar los elementos de color azul. Esto se genera a través de dos vectores de tres posiciones que definen las componentes rojo, verde y azul (RGB), de la imagen, lo que permite definir los límites superior e inferior de color que se necesitan. Luego, se utilizan filtros gaussianos, cierres y aperturas morfológicas y erosión y dilatación para mejorar la calidad de la imagen, para evitar y suprimir el ruido que se pueda presentar durante la separación de color.

Figura 8. Captura de imagen del robot

Con la imagen filtrada, se utiliza el algoritmo de Canny para detectar los bordes de las formas obtenidas al usar los filtros anteriores, de esta manera, se procede a detectar si los bordes obtenidos pertenecen a alguno de los landsmark. El proceso de detección de los círculos se realiza con la misma imagen filtrada por el color, pero se utiliza la transformada de Hough para la detección de los mismos, ya que la biblioteca OpenCV posee la librería para realizar este proceso directamente. Dentro de la detección de los círculos, la librería permite configurar parámetros como son el radio mínimo y máximo de los círculos a detectar, y la distancia mínima que debe tener un círculo de otro. Esto permite definir un valor en pixeles, en este caso 150, que evita la detección de círculos demasiado pequeños o alejados del robot, y solo detectar aquellos con un tamaño superior o cercano.

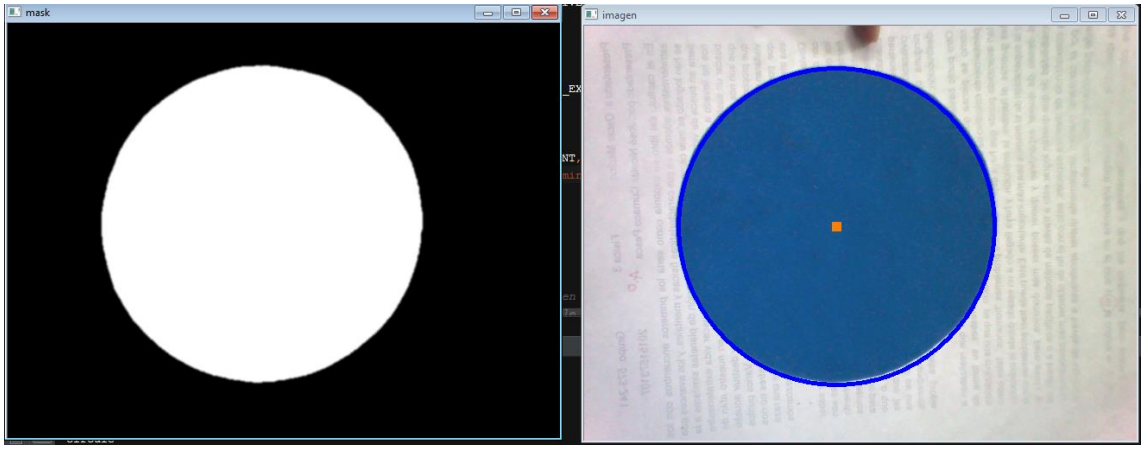


Figura 9. Detección del círculo

Con la detección de bordes realizada, se procede a validar si los contornos encontrados pertenecen a un triángulo o un cuadrado. Para ello, se utiliza una función definida en OpenCV llamada findcontours, que permite ubicar los puntos que pertenecen a un contorno, o ubicar las aristas de la figura detectada. Así, limitando la cantidad de puntos a detectar, se conoce cuál es la imagen detectada. Con tres puntos detectados, se valida que es un triángulo, con cuatro puntos, se valida que es un cuadrado.

Se debe tener en cuenta que los triángulos a detectar son 4, por lo que se realiza un análisis del triángulo detectado para determinar la dirección en la cual apunta. Para realizar este análisis, se procede a determinar cuál es la arista del triángulo más cercana a la parte superior de la imagen. De esta manera, se dibuja la recta que genera con la siguiente arista y se calcula el ángulo que posee con respecto al eje X. Conociendo el ángulo generado, se compara con el valor que debe poseer el ángulo y se determina la dirección en la que apunta. Si el ángulo generado no cumple con estas condiciones, entonces no es un landsmark valido. Los ángulos determinados se pueden observar en la figura 10.

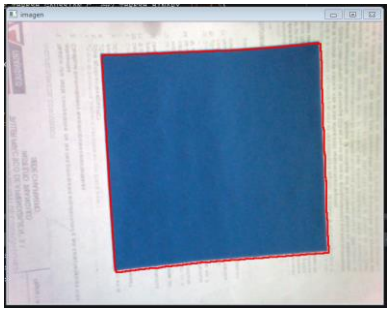
Ángulo	Dirección
120° - 140°	Arriba

50° - 60° y > 170°	Abajo
140° - 160°	Izquierda
75° - 95°	Derecha

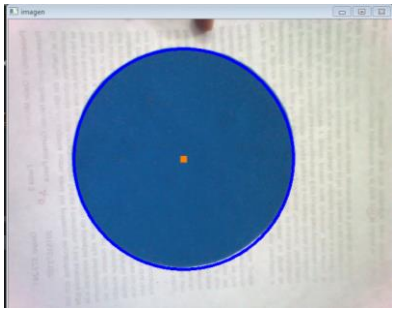
Figura 10. Ángulos válidos para detección de triangulo

Estos valores se obtienen de manera experimental de acuerdo a los landsmarks utilizados, observando para que ángulo se cumple la condición de cada uno. Pueden ser ajustados de acuerdo al triángulo a utilizar.

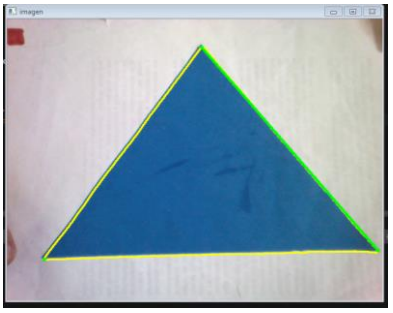
Realizado el procesamiento y detección de landsmarks, se procede a desarrollar el algoritmo para la ejecución de los movimientos del robot. A través del código de programación, se define las acciones que debe realizar el robot de acuerdo al landsmark detectado. Para ello, se utiliza condicionales que determina que acción realiza el robot con el landsmark detectado. Estas acciones se pueden observar en la figura 11.



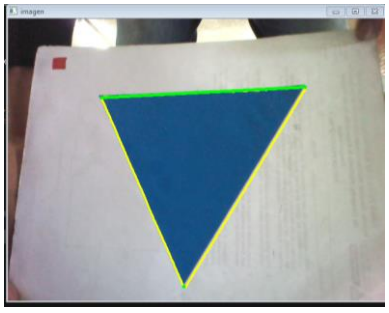
El robot se sienta



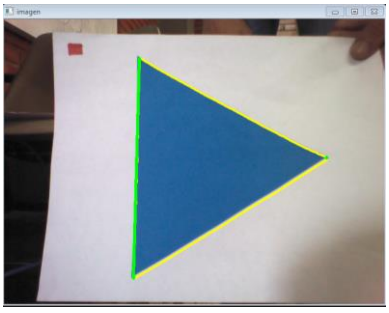
El robot se pone de pie



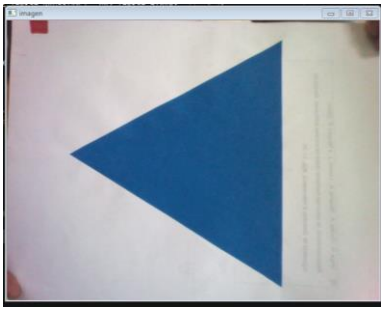
El robot avanza



El robot retrocede



El robot gira hacia la derecha 90°

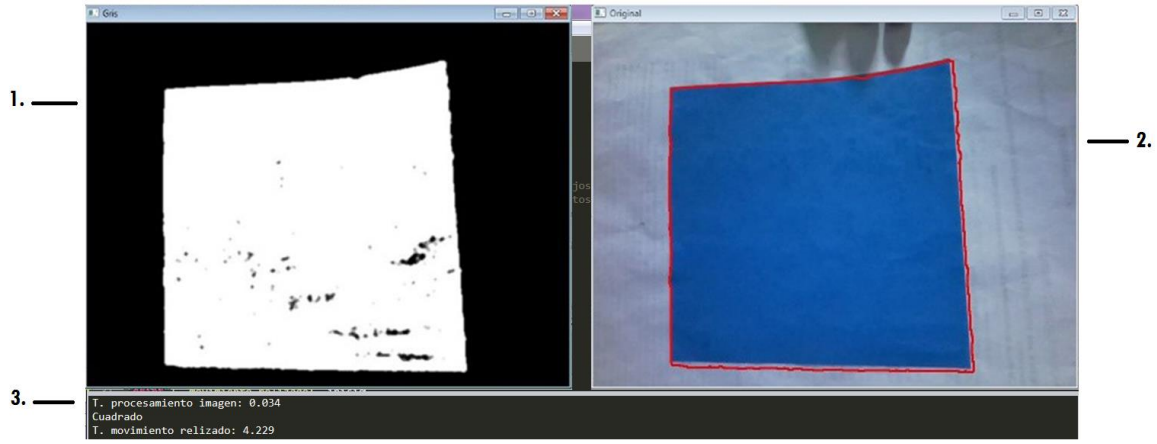


El robot gira hacia la izquierda 90°

Figura 11. Figuras detectadas

Con estas indicaciones, a través de la librería del robot, se ejecuta el movimiento de acuerdo al landsmark detectado. Mientras el robot está en movimiento no captura nuevas imágenes, luego de ejecutar el movimiento, vuelve y captura la imagen proveniente de la cámara. Dentro de la librería del robot es posible ejecutar movimientos personalizados, pero para el desarrollo de la navegación se utilizaron movimientos preestablecidos, donde se puede configurar parámetros como ángulo de giro, en el caso de la detección de los triángulos hacia los lados, o la distancia a recorrer o pasos a dar cuando el robot necesita avanzar o retroceder. Los movimientos de sentarse y ponerse de pie no son configurables.

Luego de la ejecución de los movimientos, se realiza la publicación de los tiempos de detección de los landsmark y el tiempo de ejecución de cada movimiento en la consola del software Pycharm. Para realizar el cálculo del tiempo de la detección de los landsmarks, se utiliza marcadores que toman el tiempo en dos puntos del algoritmo, cuando captura la imagen, y cuando termina de realizar el proceso de detección de cada landsmark. De esta forma, restando los dos valores obtenidos, se obtiene el valor en segundos del tiempo que tardo en realizar la detección. Estos marcadores permiten obtener el valor que demora en detectar un círculo, un cuadrado o alguno de los triángulos.



1.Recuadro mascara aplicada
2.Recuadro de imagen original con remarca de figura
3. Tiempos de procesamiento de imagen y movimiento realizado

Figura 12. Identificación de elementos

Para determinar el tiempo que demora en ejecutar cada movimiento, se utiliza el mismo proceso de los marcadores, pero estos se implementan después de que detecta el landsmark y cuando termina de ejecutar la acción, así, restando estos valores, se calcula cuanto tiempo tarda en ejecutar estas acciones. Para facilitar el proceso de depuración y validación, también se imprime en la consola la figura detectada.

3 Resultados obtenidos

Para verificar el funcionamiento del algoritmo planteado, se integra el algoritmo desarrollado en el software Pycharm, en el lenguaje de programación Python, en el robot NAO, y se realizan las siguientes pruebas.

3.1 Prueba de detección de landsmark: Se realizó la detección de cada landsmark 50 veces con el robot, verificando que la detección fuera correcta. En caso de que el robot detectara una figura diferente a la mostrada, se clasifica como erróneo. Los datos obtenidos de esta detección se visualizan en la figura 13.

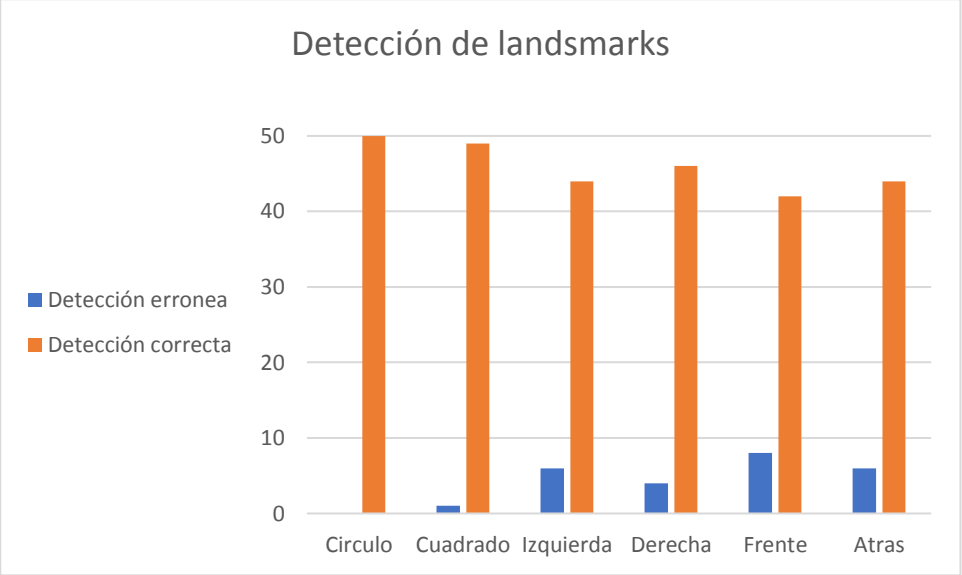


Figura 13. Detección de landsmarks

Realizando el cálculo porcentual de error, se determina los siguientes valores, de acuerdo a las pruebas realizadas. Los porcentajes de error se observan en la figura 14. El mayor error presentado es al momento de detectar el triángulo hacia arriba, debido a los ruidos que se filtraban en la imagen, e impedían la detección de la figura correcta. En algunos casos, durante las pruebas, los cambios de luz generaban que el color de la imagen cambiara, logrando que la cámara detectara nuevos elementos incorrectos o, por el contrario, no detectara correctamente la figura en la imagen, dañando la detección o detectando otra figura.

En el caso de los triángulos, la imagen se distorsionaba en las puntas, ya que no detectaba la totalidad del triángulo, por lo que, al pasar por el filtro de color, se convertía en una imagen diferente, generalmente siendo detectado como un cuadrado.

Figura	% Error
Circulo	0%
Cuadrado	2%

Triangulo a la izquierda	12%
Triangulo a la derecha	8%
Triangulo hacia arriba	16%
Triangulo hacia abajo	12%

Figura 14. Porcentaje de error en la detección

3.2 Prueba de toma de decisiones: Después de validar el funcionamiento del algoritmo de detección, se procede a validar la ejecución de los movimientos del robot del robot y los tiempos de detección de los landsmark, junto con los tiempos de ejecución de cada movimiento. Para esta prueba, se muestra al robot los landsmark y se verifica que el movimiento generado corresponda con la acción programada, como se indica en la figura 10, permitiendo la navegación en el ambiente al momento de detectar las figuras. Dentro de esta prueba, se evidencia los errores de detección antes mencionados.

En la figura 15 se evidencia el movimiento generado por el robot al ejecutar un giro a la derecha. Se puede validar el ángulo de giro del robot, que es de 90 grados, parámetro configurable a través del código. En la figura 16 se puede validar el movimiento del robot hacia la izquierda.

Figura 15. Giro a la derecha del robot

Figura 16. Giro a la izquierda del robot

Las condiciones para el movimiento del robot no presentan fallas, mientras la detección del landsmark sea correcta, por lo que el movimiento del robot es correcto, además de ejecutar los mensajes programados para cada movimiento.

Para la evaluación de los tiempos de detección y procesamiento de la imagen se realizaron 50 pruebas, donde se muestra al robot el landsmark y se captura el tiempo que tarda en procesar y detectar el mismo. Se toman los tiempos de procesamiento de la imagen, obteniendo los resultados de la figura 17. Estos tiempos fueron obtenidos con los marcadores descritos anteriormente, determinando tiempos de respuesta que oscilan entre los 31 y 37 mS, por lo que se realiza un promedio del tiempo de respuesta del robot para aplicar a futuros procesos. Este promedio de tiempos puede observarse en la figura 18.

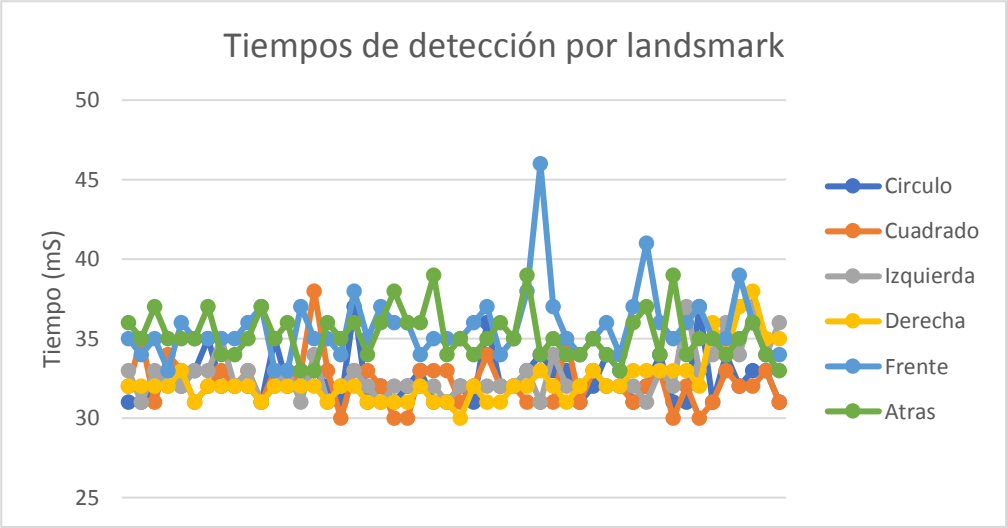


Figura 17. Tiempos de detección de landsmark

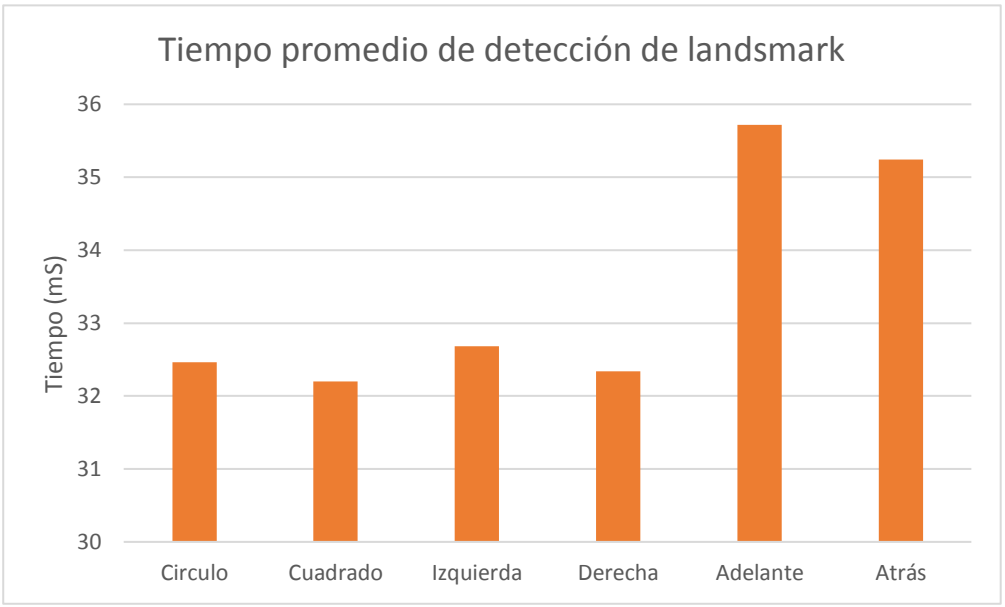


Figura 18. Tiempo promedio de detección de landsmark

El tiempo promedio más alto es cuando el robot detecta el landsmark del triángulo hacia arriba, cuyo tiempo es aproximadamente 36 mS, 10,53% más alto que el valor promedio más bajo registrado, perteneciente al cuadrado, de aproximadamente 32 mS, mientras que cuando el robot detecta el landsmark del triángulo hacia abajo, el tiempo aumenta aproximadamente 7,83% respecto al cuadrado, siendo un tiempo de 35,2 mS, lo cual no es un aumento significativo en la detección de la imagen, pero puede ser tenido a consideración en futuros

procesos. De acuerdo al landsmark detectado, se puede observar los tiempos de ejecución de las acciones del robot, que son medidos utilizando los marcadores implementados en el algoritmo. Por las acciones implementadas, el robot se mueve en un tiempo aproximado entre 4 y 7 segundos, siendo el tiempo más corto cuando el robot se sienta, dando un tiempo de 4,2 segundos, y el tiempo más largo cuando el robot gira hacia la izquierda, siendo un tiempo de 6,4 segundos. Estos tiempos promedios se puede observar en la figura 19.

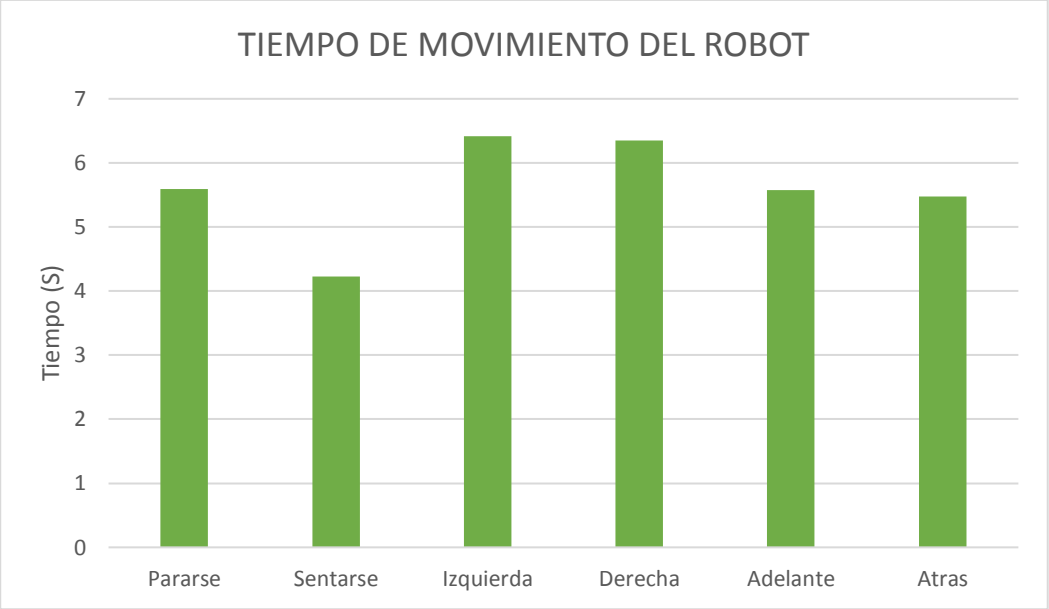


Figura 19. Tiempo promedio de movimiento del robot

4 Conclusiones

Utilizando el procesamiento de imágenes, aplicando técnicas como transformada de Hough, el algoritmo de Canny y filtros para prevenir el ruido y separar las imágenes de un color especifico, se puede lograr un proceso de detección de landsmark de manera sencilla, obteniendo resultados con porcentajes de error que no superan el 20%, y que es integrable con una plataforma robótica que permite el desarrollo de aplicaciones como movimiento y navegación de acuerdo a una figura detectada, atendiendo a movimiento básicos como moverse adelante, atrás y girar sobre su posición. Se puede observar como a partir de figuras

básicas como lo son el circulo, el triángulo, y el cuadrado se puede generar un desarrollo de movimientos específicos, que puede aplicarse a casos reales, como puede ser clasificación de objetos, resolución de laberintos de acuerdo a los landsmark detectado o detección de figuras o elementos irregulares.

También se puede dar pie a desarrollos de procesamiento más complejos, como puede ser el uso de redes neuronales para la detección de elementos o detección de emociones en experimentos de interacción humano-robot, así como desarrollo de sistemas de aprendizaje por parte del robot. Se valida las posibilidades de desarrollo que tiene el robot, ya que solo se utilizó la cámara y los sistemas de movimiento, mientras que los sensores que este posee pueden ser integrados y aplicados a desarrollos más complejos o aplicarse al mismo desarrollo pero de manera complementaria,

5 Bibliografía

Botero Valencia, J. S., & Delgado Trejos, E. (2009). Segmentación para la identificación y planeación de trayectorias de Robots en 2D sobre secuencias de video. *Stsiva 2009*.

Carranza Athó, F. (2006). Esqueletización Tópicos Especiales en Procesamiento Gráfico, 1-9.

Cuevas, E., Oliva, D., Osuna-enciso, V., & Wario, F. (2011). Detección de primitivas circulares usando un algoritmo inspirado en el electromagnetismo. *Ingeniería, Investigación y Tecnología*. https://doi.org/http://dx.doi.org/10.1016/S1405-7743(13)72231-5

Delgado-Gutiérrez, M. J., Herrera-Guillén, D. F., Medina-Barragán, L. M., & Corredor–Gómez, J. P. (2017). Implementación de un sistema de procesamiento de imágenes integrado con Raspberry PI 2B para reconocimiento y recolección de fresas maduras. *REVISTA POLITÉCNICA*.

Fernando, Ó. F., & Correa, H. L. (2009). Identificación biométrica utilizando imágenes infrarrojas de la red vascular de la cara dorsal de la mano Biometric identification using

infrared dorsum hand vein images. *Revista Ingeniería e Investigación*.

Giraldo, F. N., González, M. F., & Camargo, E. (2011). ALGORITMOS DE PROCESAMIENTO DE IMÁGENES SATELITALES CON TRANFORMADA HOUGH. *Revista Visión Electronica*.

Ioannou, A., Andreou, E., & Christofi, M. (2015). Pre-schoolers' Interest and Caring Behaviour Around a Humanoid Robot. *TechTrends*. https://doi.org/10.1007/s11528-015-0835-0

Jaramillo, M. A., Fernández, J. Á., & Martínez De Salazar, E. (2000). Implementación del Detector de Bordes de Canny sobre Redes Neuronales Celulares. *Proc. SAAEI* .

Joo, J. M. (2006). Hough-transform based algorithm for the automatic invariant recognition of rectangular chocolates . Detection of defective pieces . *Engineering*. https://doi.org/10.15381/idata.v9i2.6617

Jose Armando Ruiz Leyva. (s. f.). Reconocimiento de figuras geométricas mediante Hough.

María Calderón, D., & González Gómez Gabriela Kramer Bustos Doricela Gutiérrez Cruz, E. (2007). PROCESAMIENTO DE IMÁGENES APLICADOS A LA EDUCACIÓN. *Año Facultad de Ciencias Químicas e Ingeniería. UABC. Copyright*.

Méndez, P. M., Villa, H. M., & Cisneros Barahona, A. S. (2017). Nuevo algoritmo para la detección de bordes en imágenes para esteganografía. *Maskana*.

Orozco, M. S., Ruiz Salguero, O. E., & Jasnoch, U. (2005). Edge and corner identification for tracking the line of sight. *Ingeniería y Ciencia - ing.cienc.*

Romero Acero, Á., Marín Cano, A., & Jiménez Builes, J. A. (2015). Sistema de clasificación por visión artificial de mangos tipo Tommy. *Revista UIS Ingenierías*.

Sanchez, T. G., & Intelligence, A. (2009). Artificial Vision in the Nao Humanoid Robot. *Computer*.

Valentí Soler, M., Agüera-Ortiz, L., Olazarán Rodríguez, J., Mendoza Rebolledo, C., Pérez Muñoz, A., Rodríguez Pérez, I., … Martínez Martín, P. (2015). Social robots in advanced

dementia. *Frontiers in Aging Neuroscience*. https://doi.org/10.3389/fnagi.2015.00133