UNIVERSIDAD VERACRUZANA

CENTRO DE INVESTIGACIÓN EN INTELIGENCIA ARTIFICIAL

PREDICCIÓN ROBUSTA DEL TIEMPO AL CONTACTO MEDIANTE EL ANÁLISIS DEL TAMAÑO APARENTE DE LOS OBSTÁCULOS EN LA NAVEGACIÓN ROBÓTICA

TESIS

PARA OBTENER EL GRADO DE

DOCTOR EN INTELIGENCIA ARTIFICIAL

PRESENTA

M.I.A. ÁNGEL JUAN SÁNCHEZ GARCÍA

DIRECTOR

DR. HOMERO VLADIMIR RÍOS FIGUEROA

XALAPA, VER. 4 de Septiembre de 2018

Índice

Agradecimientos

A Dios, por brindarme todas las oportunidades para terminar este trabajo, aún sin merecerlas.

A mis padres, Sergio Sánchez y Gricelda García, quienes apoyaron de principio a fin mi carrera académica, poniendo todo a su alcance para que pudiera alcanzar este logro.

A mi hermano Andrés, quien no solo me apoyó con la parte gráfica y diagramas de este trabajo, sino que me acompañó, aprendiendo juntos de esta gran etapa que es el doctorado.

A mi novia Angie, pues siempre me alentó a terminar este trabajo, a pesar de las adversidades que surgieran, dándome su consejo sincero, con el afán de lograr esta meta.

Al Dr. Homero Ríos, quien confió en mí durante estos 4 años y algunos más atrás, en mis ideas y aportaciones para que este trabajo se culminara de la mejor manera.

A mis sinodales, los doctores Antonio Marín, Ericka Rechy, Aurora Montano y Guillermo Hoyos, por sus aportaciones para pulir este trabajo doctoral.

Al Dr. David Oliva, por su invitación al Doctoral School, cuya formación puedo decir que lo apliqué en mi trabajo, y que me sirvió para crecer personalmente. Sabe el Dr. David que para nosotros es parte de la familia.

Al Dr. Gustavo Quintana, por su invaluable apoyo en la parte experimental, y por ser un gran amigo durante mi estadía en Bélgica.

Al **Dr. Isaac Rudomín** del Barcelona Supercomputer Center, al **Dr. Johan Schoukens** de

Vrije Univertieit Brussel y **Hugues Garnier** de Université de Lorraine, por todos sus consejos y aportaciones a este trabajo.

Resumen

En la navegación de robots móviles es necesaria una medida para evaluar la proximidad de los obstáculos denominada tiempo al contacto. El tiempo al contacto (TTC) es un mecanismo bioinspirado que se utiliza principalmente en la navegación robótica para detectar posibles peligros con obstáculos en el entorno. A través de diversos trabajos, diferentes enfoques han sido utilizados para estimar el tiempo a la colisión, sin embargo, debido a la diversidad de configuraciones que se pueden encontrar en un ambiente, resulta difícil encontrar un método robusto a éstas diferencias. Mi propuesta está basada en detectar obstáculos mediante sensores pasivos (visión por computadora) en vez de sensores costosos y con un consumo de energía considerable. Este trabajo de investigación presenta una propuesta basada en algunos enfoques tanto de segmentación de obstáculos como de modelado del tamaño aparente en el tiempo del obstáculo. Además se presenta una propuesta para robustecer estimaciones y tomar decisión con base en el movimiento relativo del ambiente para esquivar el obstáculo. Los resultados muestran que la segmentación mediante contornos probabilísticos da resultados robustos en cuanto a cambio de iluminación ambiental, sin embargo tiene problemas con objetos cóncavos, mientras que la extracción de características permite segmentar obstáculos simúltáneos en la escena, con previo aprendizaje de características de estos obstáculos. También la generación de un modelo Autoregresico Exógeno ARX presenta un buen descriptor del cambio del tamaño de los obstáculos, eliminando observaciones atípicas en el ambiente.

Abstract

In the navigation of mobile robots, a measure is necessary for assessing the proximity of the obstacles called Time to Contact (TTC). TTC is a bioinspired mechanism that is used mainly in robotic navigation to detect possible dangers with obstacles in the environment. Through different works, some approaches have been used to estimate the time to collision, however, due to the diversity of configurations that can be found in an environment, it is difficult to find a robust method to these differences. My proposal is based on detecting obstacles through passive sensors (computer vision) instead of expensive sensors with high energy consumption. This research presents a proposal based on some approaches to both obstacle segmentation and modeling of the apparent size in the time of the obstacle. In addition, a proposal is presented to strengthen estimates and make decisions based on the relative movement of the environment to avoid the obstacle. Our results show the segmentation by probabilistic contours gives robust results in terms of environmental lighting change, however it has problems with concave or convex objects, while the extraction of features allows segmentation of simultaneous obstacles in the scene, with prior learning of characteristics of these obstacles. Also the generation of an autoregressive exogenous ARX model presents a satisfactory descriptor of the change of the size of the obstacles, eliminating atypical observations in the environment.

Introducción

El Centro de Investigación en Inteligencia Artificial (CIIA) de la Universidad Veracruzana es un Centro que promueve la investigación en el área de Inteliencia Artificial dividido en tres cuerpos académicos, cada uno con una Línea de Generación y Aplicación del Conocimiento (LGAC):

- El Cuerpo Académico Investigación y Aplicaciones de la Inteligencia Artificial cultiva la **LGAC1 Aprendizaje computacional**, que busca desarrollar investigación de punta en el área de Aprendizaje, Agentes y Tecnologías Web y aplicar este conocimiento para la resolución de problemas complejos. Lo cual permitirá generar un grupo de investigación reconocido por su sólida producción científica y su colaboración con otros grupos especializados nacionales y extranjeros.

- El Cuerpo Académico Investigación y Aplicaciones de la Robótica Inteligente cultiva la **LGAC2 Robótica Inteligente**, la cuál se enfoca en el desarrollo de sistemas robóticos capaces de resolver problemas complejos de manera robusta y eficiente. Este tipo de sistemas combinan los desarrollos de diversas temáticas de las Ciencias Computacionales, como son: Visión y Percepción, Sistemas Decisionales, Planificación y Seguimiento de Trayectorias, Navegación y Localización Autónoma e Interfaces Hombre-Maquina, entre muchos otros temas.

- El Cuerpo Académico Física Estadística de los Sistemas Complejos cultiva la **LGAC3 Minería de Datos de los Sistemas Complejos** donde se estudia y desarrolla métodos de análisis de sistemas complejos económicos, biológicos y de otros tipos (procesamiento de imágenes, autómatas celulares, redes neuronales) mediante metodologías basadas en técnicas y conceptos tales como simulación de agentes, información, entropía y entropía de Tsallis, minería de datos, etc.

El presente trabajo de investigación surge en el cuerpo académico “Investigación y Aplicaciones de la Robótica Inteligente” y la LGAC2.

Además, este trabajo fue complementado mediante ideas y aportaciones científicas de miembros de los siguientes departamentos:

- Facultad de Estadística e Informática, de la Universidad Veracruzana, Xalapa, México.
- ELEC Department, Vrije Universiteit Brussel, Bruselas, Bélgica.
- Centre de Recherche en Automatique de Nancy (CRAN), Université de Lorraine, CRAN, Vandoeuvre-les-Nancy, Francia.
- National Center for Scientific Research (CNRS), CRAN, Francia.

Hoy en día, muchas actividades automáticas requieren la asistencia de entidades autónomas, como robots industriales y de atención domiciliaria, automóviles que pueden conducirse solos y sistemas que ayudan en el aterrizaje de aeronaves. Las actividades anteriores se pueden dividir en subtareas que, en sí mismas, no son triviales y se han gestionado a lo largo del tiempo [1], por ejemplo, localización [2][3], comunicación [4][5], planeación de rutas [6][7][8][9] y evasión de obstáculos [10][11].

De todas estas tareas presentes en la robótica, éste trabajo de investigación se centra en la prevención de obstáculos, es decir, en el cálculo de una secuencia de control de movimiento donde no existan colisiones. Sin embargo, este problema se vuelve no trivial debido a la gran cantidad de diferentes condiciones y configuraciones ambientales que influyen en la percepción del medio ambiente y sus cambios a lo largo del tiempo.

Más específicamente, se aborda el problema conocido como **“Tiempo al Contacto"**, que nos

servirá para saber el tiempo en que el robot colisionará con un obstáculo identificado. Para identificar obstáculos, nuestro trabajo utiliza sensores pasivos como la visión por computadora, pues la vista es el sentido en el que más confían muchos seres vivos para navegar en su ambiente, además del bajo consumo de energía en contraste con otros sensores. La reflectancia de la luz en los objetos puede darnos mucha información sobre sus características, como la forma, el material, la temperatura, la iluminación y el dinamismo.

Sin embargo, esta información no siempre se puede obtener a partir de dispositivos electrónicos. De hecho, el sistema visual real es tan complejo, que puede distinguir características que no son perceptibles para los dispositivos en ciertas situaciones, como la oclusión de objetos, el cambio de iluminación, datos faltantes en los objetos entre otras, por lo que el interpretar de manera exitosa los datos que recibe el dispositivo, requiere de métodos complejos y sofisticados, además de gran poder de cómputo. Conforme la tecnología avanza, el desempeño y funcionalidad de la visión por computadora ha mejorado aplicaciones para dispositivos, seguridad, industria, entre otras.

Una de las tareas primarias de la visión por computadora es reconstruir, a partir de imágenes en dos dimensiones, las propiedades tridimimensionales de una escena, como la forma, movimiento y orden espacial de los objetos. En visión monocular, una meta importante es recuperar a partir de varias imágenes en el tiempo, el movimiento relativo entre un observador y el entorno. La estructura del entorno obtenida es utilizada para generar distancias relativas de puntos sobre una superficie en la escena del observador. En teoría al menos, las distancias absolutas se pueden determinar a partir de las imágenes si se conoce el movimiento [12].

En el campo de la robótica se han desarrollado diversas técnicas basadas en visión para la manipulación o posicionamiento inteligente de objetos, la movilidad en robots autónomos, mapeo de ambientes en tres dimensiones o localización de objetos. Especialmente para los robots móviles,

la información visual contribuye a la fiabilidad de movimiento en virtud de diversos entornos [13]. Generalmente, la visión funciona bien para entender los ambientes, y es usada para planear la trayectoria o cambio del movimiento de acuerdo a la posición de un objetivo visual [14][15].

Los robots como sistemas autónomos deberían ser capaces de controlar sus movimientos en el ambiente y adaptarlos a eventos inesperados. Esto es posible solo si el robot es capaz de "sensar" el ambiente. Por ejemplo, si tiene la capacidad de hacer movimientos exploratorios con una cámara monocular, puede ser capaz de obtener características del ambiente, como forma de los objetos. Otro ejemplo es cuando un robot móvil navega a través de un corredor con obstáculos, el robot lleva una cámara para recoger datos sobre el medio ambiente y evitar los posibles obstáculos en su camino. Mientras el robot está en movimiento, está tomando imágenes del entorno que le permite estimar el movimiento a partir de una imagen. Esta estimación es de gran importancia con el fin de detectar los objetos en frente del robot y evitarlos. Para navegar de manera segura, es necesario el uso de la visión, o más específicamente, de la visión activa, para estimar tanto los obstáculos que puedan aparecer en el camino del robot y el movimiento del robot mismo.

Con base en lo anterior, éste trabajo utilizará la visión por computadora para identificar obstáculos. Sin embargo, se apoyará de otras áreas como la probabilidad y estadística, así como la identificación de sistemas para obtener resultados más robustos que permitan obtener estimaciones más tolerantes a factores ambientales como el cambio de iluminación o movimiento del robot, mismos que serán abordados en detalle durante los capítulos subsecuentes.

Este trabajo está organizado como sigue: El capítulo I describe los fundamentos sobre los cuales recae y motivan a este trabajo de investigación, además de resumir nuestra contribución. El capítulo II muestra un resumen de los antecedentes, marco teórico y trabajos relacionados. Además describimos ventajas y desventajas de estos trabajos, así como la ubicación del nuestro en el estado del arte.

Por último resalta nuestra propuesta y la metodología para realizarla. El capítulo III describe en detalle 3 enfoques para localizar obstáculos en una escena (módulo de segmentación). El capítulo IV explica 3 enfoques para realizar el modelado del comportamiento del tamaño aparente del obstáculo segmentado a lo largo del tiempo. El capítulo V muestra el módulo para estimar e interpretar el Tiempo al contacto, así como el apartado de toma de decisión. El capítulo VI describe la configuración y justificación de los experimentos, así como los resultados obtenidos. Por último se describen las conclusiones extraídas de este trabajo.

1. Capítulo I: Fundamentos del trabajo de investigación

En este capítulo se presentan los fundamentos que dan pie a este trabajo de investigación, después de haber realizado el protocolo de investigación sobre el tema.

1.1. Problemática

Cuando se habla de la navegación autónoma, uno de los problemas que surgen es que los robots deben distinguir colisiones potenciales que puedan encontrar en su entorno. Por otro lado, al reconocer obstáculos mediante una cámara, la precisión y el tiempo para procesar datos por robots es muy importante, ya que de esto depende el éxito de muchas tareas que involucran el movimiento autónomo de los robots. Durante años, el problema de procesar secuencia de imágenes para movimiento ha sido estudiado en el área de navegación de robots móviles.

Dada una secuencia de imágenes adquiridas por un observador monocular siguiendo el movimiento de objetos rígidos sin restricciones, el problema de la estimación del *egomotion* es definido como “el proceso de calcular el movimiento de un vehículo en un espacio de tres dimensiones"[17]. En otras palabras, se puede entender como el problema de recuperar la traslación y/o rotación que comprende el movimiento relativo entre el observador y el ambiente. El problema de estimar egomotion usando una entrada visual es particularmente difícil, debido al hecho de que a partir de imágenes, solo es posible recuperar información relacionada con el movimiento 2D, mientras que el egomotion requiere información 3D. Además, dado que la dependencia del movimiento 2D sobre la estructura de la escena no es lineal, si existen pequeños errores en la estimación pueden tener un impacto significativo en la precisión de la recuperación del movimiento.

Por último, la confusión entre la traslación y rotación hace que el problema de estimar egomotion sea más difícil que el problema de estimar solamente rotación o traslación.

Además, diferentes factores como el cambio de iluminación o los diferentes relieves donde se desplaza el robot, hacen la estimación del tiempo a colisionar con un obstáculo pueda no ser tan confiable como se espera. Además, un problema desafiante en la detección de obstáculos es la enorme cantidad de diferentes condiciones ambientales que influyen en la percepción del medio ambiente y sus cambios a lo largo del tiempo.

1.2. Hipótesis

Ante los problemas antes mencionados, la hipótesis sobre la que sustenta este trabajo doctoral es la siguiente:

Es posible robustecer la estimación del tiempo al contacto utilizando sensores pasivos para pronosticar colisiones con obstáculos de manera precisa en la navegación de robots autónomos.

Derivado de la hipótesis enunciada anteriormente, aclaramos que nos referimos con la palabra *robustecer*, a la acción a tolerar cambios de iluminación, así como la detección de diferentes objetos y formas.

1.3. Justificación

Como se ha mencionado, el conocimiento de parámetros visuales tales como el TTC son esenciales para que un robot o ser vivo pueda desempeñarse apropiadamente en un entorno móvil y realizar tareas como la evasión de obstáculos, estacionarse de manera autónoma, tomar algún objeto, en-

tre otras. El problema de egomotion requiere de la estimación de los movimientos de traslación y rotación.

Además, en varios trabajos se han utilizado hardware especializado, en otros cámaras espaciales, sonares o sensores infrarrojos. Esto conlleva a la inversión en sensores costosos y de alto consumo de energía. Por lo anterior, se propone obtener resultados robustos para el problema del egomotion, especialmente del Tiempo al contacto, utilizando sensores pasivos (como una cámara) sin usar más sensores costosos, que permitan reducir costos de equipamiento y consumo de energía a robots móviles.

1.4. Objetivos

A continuación se listan los objetivos a alcanzar en éste trabajo doctoral de investigación.

1.4.1. Objetivo general

Estimar el TTC para la navegación robótica de manera *robusta*, (precisa, invariante a la iluminación, más tolerante al ruido e incertidumbre en los parámetros de entrada y funcional en diferentes ambientes) basada en sensores pasivos (no dependiente de sensores como el laser, o el kinect) como lo haría un ser vivo.

1.4.2. Objetivos particulares

1. Realizar una revisión sistemática de la literatura para conocer ventajas y desventajas de los métodos propuestos en trabajos relacionados.

2. Proponer tres mecanismos para segmentar e identificar obstáculos en una o varias imágenes.

3. Proponer tres enfoques para robustecer las estimaciones del cambio de tamaño de los obstáculos en el tiempo.

4. Comparar las estimaciones con el Ground truth.

5. Comparar los enfoques propuestos para concluir sobre los ambientes en los cuales utilizar cada enfoque.

6. Publicar al menos cinco artículos de congreso y uno de revista sobre los resultados obtenidos.

1.5. Contribución

Como se detallará en el siguiente capítulo, mi contribución está ubicada en el bien conocido problema de estimar el TTC. Dicha contribución se diferencia de todas aquellas que están basadas en sensores que obtienen directamente la distancia a los obstáculos para conocer la distancia a ellos. Al calcular tiempo a la colisión, nosotros utilizamos visión monocular, lo que reduce costos de energía y dinero. Este cálculo está basado en tres tareas principales.

1. **Segmentación.** En esta tarea se proponen dos enfoques principales para encontrar objetos y que serán descritos con detalle a lo largo de este trabajo.

   a) El primero se basa en generar contornos probabilísticos basados en el método Maximum A posteriori y cadenas ocultas de Markov para estimar el tamaño de los objetos.

   b) El segundo está basado en un aprendizaje de características de objetos que pueden ser extraídas y seguidas a lo largo del tiempo, con el fin de que el robot conozca a priori los obstáculos con los que puede colisionar.

2. **Modelado.** En esta tarea nos enfocamos en generar modelos que describan y pronostiquen el cambio del tamaño de los obstáculos en el tiempo. Los tres enfoques utilizados son: Filtro de Kalman, Series temporales e Identificación de sistemas.

3. **Robustecimiento.** En esta tarea se pretende estimar el TTC de manera robusta y sin oscilaciones, utilizando métodos estadísitcos y basados en un comportamiento de velocidad constante, para que el robot pueda tomar la decisión de esquivarlo con base en el movimiento de su ambiente.

1.6. Publicaciones y reconocimientos

A continuación se listan las publicaciones realizadas a partir de los resultados de este trabajo de investigación.

1.6.1. Artículos de revista JCR

1. Sánchez, A., Ríos, H., Garnier, H., Quintana, G., Rechy, E., Marín, A.: *Predicting collisions: time-to-contact forecasting based on probabilistic segmentation and system identification.* Advanced Robotics 2018. Taylor and Francis. Publicado on-line 8 de Abril de 2018. DOI: 10.1080/01691864.2018.1455604. Vol. 32, Issue 8, pp. 426-442. 2018.

1.6.2. Artículos de congreso indizados por SCOPUS

1. Sánchez, A., Ríos, H., Marín, A., Contreras, G.: Decision Making for Obstacle Avoidance in Autonomous Mobile Robots by Time to Contact and Optical Flow. 25th International Conference on Electronics, Communications and Computers CONIELECOMP 2015 (25-27 de

Febrero de 2015) y publicado en IEEE Catalog number: CPF15363-ART, ISBN: 978-1-4799-7436-8, pp. 130-134. DOI: 10.1109/CONIELECOMP.2015.70869392015.

2. Sánchez, A., Ríos, H., Hoyos, G., Marín, A.: Estimation of Time-to-Contact for Navigation of Autonomous Robots Using Parallel Processing. IEEE International Conference on Mechatronics, Electronics and Automotive Engineering ICMEAE 2016 Cuernavaca Morelos (22-25 de Noviembre de 2016) y publicado en IEEE Explore, pp. 26-31. DOI: 10.1109/ICMEAE.2016.014. 2016.

3. Sánchez, A., Ríos, H., Quintana, G., Montano, A., Marín, A.: Time-To-Contact Forecasting by Modeling the Apparent Size of Obstacles. IEEE International Conference on Mechatronics, Electronics and Automotive Engineering ICMEAE 2017 Cuernavaca Morelos (21-24 de Noviembre de 2017) y publicado en IEEE Explore, pp. 3 -7. DOI: 10.1109/ICMEAE.2017.12. 2017.

1.6.3. Artículos de congreso indizados por SCOPUS fuera del país

1. Sánchez, A., Ríos, H., Marín, A., Cortes, K., Contreras, G.: Estimation to Time-to-Contact from Tau-margin and Statistical Analysis Behavior. 23rd International Conference on Systems, Signals and Image Processing IWSSIP 2016 Bratislava, Eslovaquia (23-25 de Mayo de 2016) y publicado en IEEE IEEE Explore, ISSN electrónco: 2157-8702, pp. 1-6. DOI: 10.1109/IWSSIP.2016.7502702. 2016.

2. Sánchez, A., Ríos, H., Marín, A., Rechy, E., Oliva, D.: Finding Learned Obstacles to Avoid Collisions in Autonomous Robotic Navigation. IEEE Image and Vision Computing New Zealand IVCNZ 2017, Christchurch Nueva Zelanda (4-6 de Diciembre de 2017) y publicado en IEEE Explore, ISBN electrónico: 978-1-5386-4276-4. DOI: 10.1109/IVCNZ.2017.8402489. 2017.

1.6.4. Capítulos de Libro

1. Sánchez, A., Ríos, H., Quintana, G., Marín, A.: *Predicting Collisions in Mobile Robot Navigation by Kalman Filter*. Kalman Filters - Theory for Advanced Applications, ISBN 978-953-51-5618-5., Chapter 8, Publicado el 21 de Febrero de 2018. DOI: 10.5772/intechopen.71653. 2018.

1.6.5. Reconocimientos

1. **"Hoare Quicksort Award"**, en el International Conference on Image and Vision Computing New Zealand (IVCNZ 2017), celebrado del 4 al 6 de Diciembre de 2017 en Christchurch, Nueva Zelanda por haber atentido las sugerencias y observaciones de los revisores del conference antes que cualquier artículo, con el paper titulado "Finding Learned Obstacles to Avoid Collisions in Autonomus Robotic Navigation".

1.7. Alcances

Para este trabajo doctoral se establecen los siguientes alcances:

1. La estimación del Tiempo al contacto se basará en el cambio del tamaño de los objetos.
2. Los experimentos en ambientes reales no se harán en muros, solamente en objetos que puedan diferenciarse del ambiente.
3. Los experimentos se realizarán en ambientes cerrados (no al aire libre).
4. Se experimentará con velocidad constante o casi constante en el movimiento del robot.

5. El robot avanzará sobre diferentes tipos de superficie.

6. Los obstáculos permanecerán estáticos.

En este capítulo se presentaron los fundamentos del trabajo de investigación, donde se describió la problemática que se aborda, la hipótesis y los objetivos sobre los cuáles está basado esta investigación, mi contribución al estado del arte y las publicaciones que surieron a partir de los resultados de este trabajo, donde se destacan dos congresos fuera del país y un artículo publicado en un Journal indexado por JCR.

2. Capítulo II: Trabajos relacionados y aportación

En este capítulo se presenta el marco teórico en el que se definen y describen términos relevantes con los cuales se sustenta ésta investigación. Además se presenta la revisión del estado del arte en cuanto a la estimación del TTC, mediante sensores especializados y visión monocular. Así también se muestra dónde se ubica mi propuesta.

2.1. Antecedentes

A continuación son presentados algunos antecedentes que sirven como fundamento para el objetivo de este trabajo. Estos antecedentes han servido para el problema del egomotion y algunas ideas son rescatadas para el problema del TTC. Las bases teóricas se remontan algunos años atrás, que sutentan los trabajos más actuales.

Rieger et al. en [16] desarrollaron un método basado en paralelaje móvil (*motion parallax*). Este método está basado en la idea de que la inferencia de los parámetros tridimensionales de movimiento de la cámara y la disposición de una escena a partir de los flujos de imagen se vuelve particularmente simple computacionalmente si la escena contiene variaciones de profundidad. Bajo esta condición, el movimiento diferencial de la imagen produce una estimación simple de las líneas de campo traslacional en las ubicaciones de imagen correspondientes a las discontinuidades de profundidad en la escena. Esto, a su vez, facilita las soluciones de forma cerrada de los parámetros de movimiento de la cámara y la profundidad ambiental. Este algoritmo localiza el foco de expansión de las diferencias de los vectores locales de flujo óptico. Sin embargo, a este algoritmo le es dificil calcular movimiento cuando tiene que medir vectores de flujo óptico cerca de los límites de oclusión.

Dos años más tarde, Negahdaripour y Horn presentan una aportación en [12], donde trabajaron el problema de recuperar el movimiento a partir de un observador monocular relativo a una escena, explotando los gradientes espaciales y la tasa de tiempo de cambio de brillantez sobre una imagen completa imponiendo la restricción de que la superficie de un objeto debe estar enfrente de la cámara, es decir, la profundidad sea positiva. Sin embargo, el método tiene problemas cuando el movimiento de los objetos consiste en rotación y traslación.

Nelson y Aloimonos en [18] demostraron un sistema robótico que calculó la divergencia mediante técnicas espacio-temporales aplicadas a imágenes con superficies con mucha textura, por lo que es sabido que la divergencia puede ser usada para la evasión de obstáculos en navegación robótica.

En [19], Cipolla y et al. proponen un método para medir las invariantes diferenciales del campo de velocidad de la imagen calculando los valores promedio a partir de la integral de velocidades normales de la imagen alrededor de contornos de la imagen, es decir, lo equivalente a medir cambios temporales en el área de un contorno cerrado. Esto evita tener que recuperar el campo de velocidad de una imagen y tomar derivadas parciales. En su trabajo muestran como un observador activo, haciendo pequeños movimientos, puede usar las estimaciones de la divergencia y deformación del campo de velocidad de la imagen para calcular la orientación superficial de un objeto y el Tiempo al contacto. Entre sus experimentos, muestran una alternativa para la navegación robótica calculando la información de la orientación de superficies y tiempo al contacto a partir de divergencia y deformación de la imagen.

En [20] Tistarelli y et al. demostraron las ventajas de usar mapeo polar y log-polar para la navegación. Ellos demuestran que las ecuaciones de movimiento que relacionan al egomotion y/o al movimiento de los objetos en la escena al flujo óptico son considerablemente simplificadas si la velocidad es representada en un sistema de coordenadas polar o log-polar, en contra parte a una

representación cartesiana. Lo anterior surge del hecho de que muchos investigadores se preocupan por el cómo "procesar datos" y no por definir estrategias y dispositivos para la adquisición de información.

Tian y et al. en [21] presentan una comparación de algunos algoritmos para el cálculo de Egomotion a partir de velocidades. Ellos parten del siguiente problema: cuando una cámara se mueve con respecto a una escena rígida, la imagen cambia con el tiempo. El objetivo del cálculo egomotion es estimar el movimiento 3D a partir de una secuencia de imágenes. Las técnicas para calcular egomotion a partir de secuencias de imágenes pueden clasificarse como métodos de tiempo discreto o como métodos de tiempo instantáneo, dependiendo de si la entrada es desplazamiento de imagen o velocidad de imagen (aunque solo se centran en algoritmos de tiempo instantáneo). La velocidad de la imagen, debido al movimiento de una cámara con respecto a una escena rígida y bajo una proyección en perspectiva, está dada por la conocida ecuación (1).

$$u(\mathbf{x}) = \begin{bmatrix} 1 & 0 & -x_1 \\ 0 & 1 & -x_2 \end{bmatrix} \left(\frac{T}{Z(x)} + \Omega \times x \right) \tag{1}$$

donde $u(\mathbf{x})$ es la velocidad de la imagen en la posición $\mathbf{x} = (x_1, x_2, 1)^t$, T es la velocidad traslacional, Ω la velocidad rotacional, Z la profundidad y la distancia focal es tomada (sin pérdida de generalidad) como (1). El problema de Egomotion es estimar el movimiento 3D, T y Ω a partir de un conjunto de vectores de velocidad muestreados en alguna posición de imagen (tal vez todos).

En dicho trabajo se reportan algunos resultados diferentes de lo que hasta el momento se creía. Primero, se escribía en la literatura que el problema de egomotion era difícil porque la traslación y rotación produce velocidades similares, sin embargo, encontraron que por el contrario, los algoritmos que estudian son invariantes con respecto a los ejes de rotación. También que mientras que se pensaba

que la fijación ayuda a hacer el problema de Egomotion más fácil, ellos probaron que la fijación no ayuda cuando el ruido es independiente de las velocidades de la imagen. La fijación ayuda si el ruido es proporcional a la velocidad. Por último que era ampliamente creído que el incremento del campo visual lleva a un mejor desempeño, sin embargo no necesariamente es cierto. Los algoritmos evaluados en este trabajo se enuncian a continuación:

1. **Bruss y Horn** [22]: aplicaron una manipulación algebraica para quitar la profundidad de la ecuación (1) obteniendo una restricción bilineal sobre T y Ω para cada pixel de la imagen, expresada como sigue:

$$T^t\left(x \times u(x)\right) + \left(T \times x\right)^t\left(x \times \Omega\right) = 0 \tag{2}$$

   A partir de la restricción bilineal, la rotación como una función de traslación T, se estima mediante mínimos cuadrados. Sustituyendo la estimación de la rotación en la restricción bilineal da como resultado una restricción no lineal sobre la traslación T. Entonces estimaron la traslación minimizando esta restricción no lineal sobre todas las velocidades de la imagen sujetas a $|T| = 1$. Como pudiera ser intuitivo, la desventaja de este algoritmo es que requiere una minimización numérica.

2. **Jepson y Heeger** [23] [24]: Propusieron una serie de métodos subespaciales para estimar el movimiento. El más simple de ellos es el llamado método subespacial lineal (*Linear subspace method*). Dado una muestra de flujo óptico en N puntos discretos en la imagen $x_k, k = 1, ..., N$, se puede construir un conjunto de vectores τ_i como es mostrado en (3):

$$\tau_i = \sum_{k=1}^{N} c_{ik}\left[u(x)^k \times x^k\right] \tag{3}$$

   tal que los vectores τ_i son ortogonales a T, es decir, $\tau_i \cdot T = 0$. El detalle está en escoger

$c_i = [c_{i1}, ..., c_{iN}]^t$ que sean ortogonales a todos los polinomios cuadráticos de x_{1k} y x_{2k}. Para N muestras de velocidades de la imagen, hay $N - 6\tau_i$ vectores de restricción. La estimación de T es el eigenvector correspondiente al eigenvalor más pequeño de $\sum \tau_i \tau_i^t$. La ventaja del método subespacial lineal es que T se calcula directamente sin requerir alguna optimización numérica. Su desventaja es que el método no hace uso de toda la información disponible ($N-6$ restricciones lineales contra N restricciones bilineales).

3. **Tomassi y Shi** [25]: Desarrollaron un método que usa información de paralelaje móvil de otra manera. Su método estima la traslación T a partir de deformaciones en la imagen, definidas como el cambio $\dot{\alpha}$ en la distancia angular $\alpha = arcos\left(x^i \cdot x^j\right)$ entre pares de puntos x^i y x^j de la imagen, como los movimientos de la cámara. Dado que las deformaciones de la imagen son independientes a la rotación de la cámara, se puede derivar la restricción bilineal sobre T y los dos valores de profundidad $Z(x^i)$, $Z(x^j)$ como es mostrado en (4).

$$\dot{\alpha} = sin\alpha[Z(x^j), Z(x^i), 0][x^i, x^j, w^{ij}]^{-T}T \tag{4}$$

donde $w^{ij} = \frac{(x^i \times x^j)}{||x^i \times x^j||}$. Las restricciones combinadas bilineales para un conjunto de todos los posibles pares de puntos se minimizan y resuelven para T usando el método de la variable de proyección sobre una esfera unitaria $|T| = 1$. Esta minimización involucra resolver tres parámetros de traslación y N parámetros de profundidad donde N es el número de puntos. Cuando N es muy grande, es más costoso que el algoritmo de Bruss y Horn, que resuelve solo tres parámetros.

4. **Prazdny** [26]: Hasta ahora, los algoritmos descritos anteriormente, empiezan por estimar el movimiento traslacional T, pero una vez que se conoce, se puede sustituir en la ecuación (2) para encontrar el movimiento rotacional Ω. Prazdny propuso un algoritmo que estima primero

Ω . A partir de tres puntos en una imagen, obtiene la siguiente restricción a partir de la ecuación (1):

$$n_3 \cdot (n_1 \times n_2) = 0 \tag{5}$$

donde $n_i = (\Omega \times s_i + v_i) \times s_i$, donde s_i y v_i denotan las coordenadas y la velocidad de la imagen en una retina esférica respectivamente ($i = 1, 2, 3$). En la implementación original se combinaron localmente 3 restricciones para obtener tres ecuaciones polinomiales de tercer orden con 3 variables desconocidas para poderse resolver.

5. **Kanatani** [27]: También llamada la restricción epipolar, sirve como base para varios algoritmos lineales de tiempos discretos para Egomotion [28][29]. Teniendo X y X' como las posiciones de un punto sobre una superficie antes y después de la rotación de una cámara. La restricción del movimiento rígido relaciona las dos posiciones: $X' = RX + T$, donde R es la matriz de rotación, y T es el vector de traslación. La restricción epipolar afirma que los vectores (RX), T y X' descansan sobre el mismo plano.

En [30] Stein et al. presentan un método para calcular el Egomotion vehicular partiendo de la premisa de que los puntos de la escena no siempre son fiables en escenas confusas, por lo que utilizan métodos directos en los que los valores de dos imágenes se combinan en una función de probabilidad global. Combinado con el uso de matrices de distribución de probabilidad, esto permite la formulación de un método robusto que puede ignorar un gran número de valores atípicos como uno se encontraría en situaciones reales de tráfico vehicular, aún en situaciones con mucha luz, lluvia y mucho tráfico vehicular.

Badino en su trabajo publicado en 2014 [31], propone un enfoque robusto para el problema de

la estimación Egomotion usando una plataforma móvil estero, donde para cada cuadro calculan los puntos en 3D. Además ocupan el flujo óptico para encontrar correspondencias entre puntos en cuadros siguientes.

Ancona y Poggio en [32] usan un método basado en correlaciones 1-D para detectar movimiento y el movimiento visual solo se calcula donde las medidas del TTC se deberían tomar.

Song y colaboradores [33] presentan un filtro de Kalman integrado con flujo óptico para medir la velocidad de robots móviles, debido a que los métodos de flujo óptico diferenciales requieren un gran traslape para mayor precisión en la velocidad. Por lo que para reducir el problema de los desplazamientos grandes, usaron el Filtro de Kalman para predecir las transformaciones en la imagen. Reduciendo el área de búsqueda, el filtro de Kalman le permite al flujo óptico converger rápidamente y dar velocidades precisas.

Samija [34] Consideran el problema del análisis de flujo óptico apartir de una cámara omnidireccional, con el fin de segmentar objetos dinámicos de egomotion causados por flujo óptico. Puesto que el objetivo es utilizar la cámara en un robot móvil, es razonable suponer que el movimiento de la cámara se realiza en el plano horizontal. Proponen mapear geométricamente los vectores de flujo óptico sobre una esfera, centrada en el centro de una proyección panorámica de la cámara, ya que observaron que el espacio angular es invariante a los errores del espacio planar asumido.

Yamagushi et al. en [35] abordan el problema del cálculo del flujo óptico a partir de imágenes tomadas de una cámara en un vehículo. Ellos proponen estimar el flujo a lo largo de líneas epipolares del egomotion y presentan entonces un algoritmo de agrupamiento de abajo hacia arriba que produce más segmentaciones que respeten los límites de flujo.

Por último, en los trabajos presentados en [36][37][38] se ha estudiado el flujo óptico mediante el

método de Lucas y Kanade junto con el filtro del Kalman para modelar el movimiento relativo a un observador. Con esta propuesta se identifica y predice el movimiento de regiones segmentadas por flujo óptico.

2.2. Marco teórico

Una vez revisados y presentados de manera general algunos enfoques de la estimación Egomotion en la navegación robótica, éste trabajo se enfoca en el estudio de una tarea que consta de calcular el **"Tiempo al contacto"** o **"Tiempo a la colisión"(TTC)** para manejar la evasión de obstáculos. El TTC es "un método inspirado biológicamente para la detección de obstáculos y el control reactivo del movimiento"[39]. El TTC se estudió por primera vez y se definió en [40] como "la distancia a un obstáculo dividido por la velocidad relativa entre ellos". En otras palabras, TTC es el tiempo transcurrido antes de que un observador (el centro de proyección) haga contacto con la superficie que se está viendo si el movimiento relativo actual entre el observador (por ejemplo, la cámara de un robot) y la superficie continúa sin cambios, es decir, bajo una velocidad relativa constante.

En el caso de los animales, es usado para detectar peligro, por las aves para su aterrizaje, evadir obstáculos y para los depredadores cuando necesitan saber en qué momento atacar cuando sus presas están en movimiento. Por otra parte, a partir del tiempo al contacto, es posible calcular la distancia a un objeto, con lo cual ayuda al robot móvil a decidir entre girar o detenerse cuando el choque es inminente. Z es la distancia del centro de proyección (COP) al objeto. Las coordenadas x e y se miden a partir del punto principal. $(u, v) = (\dot{x}, \dot{y})$ es el campo de movimiento y $(U, V, W) = (\dot{X}, \dot{Y}, \dot{Z})$ es la velocidad de un punto sobre el objeto relativo al sensor. La velocidad $W = dZ/dt$ es negativa si el objeto se aproxima a la cámara. El TTC básicamente es la tasa de la distancia a la velocidad [41]:

$$TTC = -Z/\frac{dZ}{dt} = -1\frac{d}{dt}log_e(Z) \tag{6}$$

Mientras que la distancia y la velocidad no se pueden recuperar a partir de imágenes tomadas con una sola cámara sin requerir información a priori, como pudiera ser la distancia principal o el tamaño del objeto, la tasa de la distancia a la velocidad si se puede recuperar directamente, incluso sin calibrar el sensor [41].

Existen varias maneras de expresar el tiempo al contacto. Una de ellas es tomándolo como relación con el foco de expansón (FOE). Si consideramos solo el modelo traslacional, por diferenciación de las ecuaciones de la proyección de la perspectiva con respecto al tiempo, se obtiene una relación entre el campo de movimiento y el Tiempo al contacto como se muestra en las ecuaciones (7) y (8).

$$u = -\frac{W}{Z}(x - x_0) = \frac{(x - x_0)}{TTC} \tag{7}$$

$$v = -\frac{W}{Z}(y - y_0) = \frac{(y - y_0)}{TTC} \tag{8}$$

donde $(x - x_0) = f(U/W)$ y $(y - y_0) = f(V/W)$, que es el foco de expansión (FOE) que es el momento en el cual la información del movimiento del foco de expansión es cero.

Se puede utilizar también el cálculo del flujo óptico y relacionarlo con el TTC como se muestra en las ecuaciones (9)(10).

$$u_x = \frac{1}{TTC} \tag{9}$$

$$v_x = \frac{1}{TTC} \tag{10}$$

Estos métodos se explicarán en detalle en la siguiente sección.

Otros métodos para calcular el flujo óptico se basan en la razón de tamaño S de la imagen del objeto a la velocidad de cambio del tamaño como se muestra en la ecuación (12).

$$TTC = \frac{S}{\frac{S}{dt}} = \frac{1}{\frac{d}{dt} log_e S} \tag{11}$$

Sin embargo se necesita mucha precisión en la medida del tamaño de la imagen de los objetivos para obtener estimaciones precisas del tiempo al contacto.

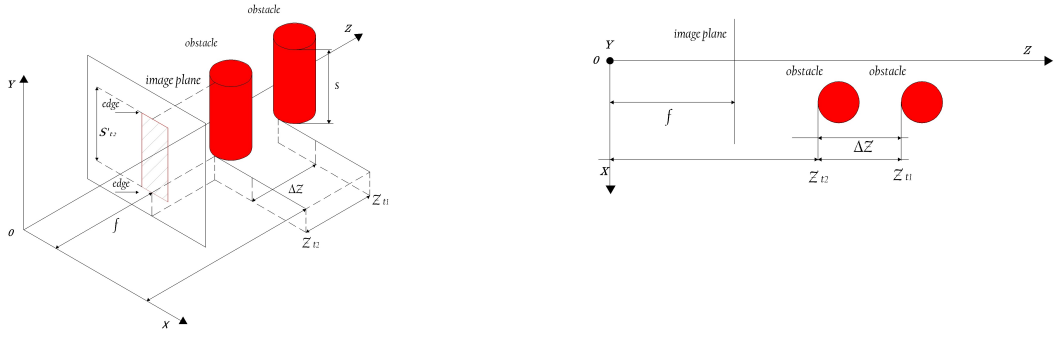
TTC generalmente se expresa en términos de la velocidad y la distancia del obstáculo considerado. La ecuación clásica para calcular TTC viene dada por (12)

$$TTC = -\frac{Z}{\frac{dZ}{dt}} \tag{12}$$

.

donde Z es la distancia entre el observador y el obstáculo, y $\frac{dZ}{dt}$ es la velocidad del robot con respecto al obstáculo.

Durante la navegación, muchos sistemas robóticos usan una sola cámara (visión monocular) para percibir el entorno [42] y esta cámara representa al observador en la estimación TTC. La Figura 1 muestra el modelo de la cámara y la percepción de los obstáculos de un robot móvil que usa visión monocular. Este modelo se presenta en vistas en perspectiva y desde arriba. Como se puede ver, hay seis parámetros principales involucrados en el modelo: i) t representa el tiempo; ii) Z_{t_i} definido como la distancia entre el centro de proyección y el obstáculo en el momento t_i; iii) ΔZ definido como la variación en distancia desde un cuadro en el tiempo t_1 y otro frame en el tiempo t_2; iv) f referido como la distancia focal obtenida previamente de la calibración de la cámara; v) S que representa la altura de un obstáculo; y vi) S'_{t_i} es el tamaño aparente del obstáculo proyectado en la imagen en el momento t_i.



(a) (b)

Figura 1: Modelo de la cámara acercándose a un obstáculo. (a) Vista isométrica del modelo. (b) Vista desde arriba del modelo.

2.3. Trabajos relacionados

Un punto clave en un sistema robótico es que sus algoritmos deberían ser adecuados para ser ejecutados bajo requisitos en tiempo real (por ejemplo, tiempo de respuesta bajo y predecible, bajos costos computacionales) [43]. En consecuencia, se han propuesto varios enfoques para estimar el TTC rescatados durante la revisión de la literatura. Por ejemplo, es común proporcionar robots móviles autónomos con sensores especializados [44] [45] , como los sonares [46][47][48] para una mayor precisión durante la navegación. La detección de profundidad utilizada en la literatura para la navegación robótica y el egomotion se ha llevado a cabo mediante la visión binocular (visión estereoscópica) [49][50], donde tienen dos cámaras generalmente calibradas que miran a un punto. Con la disparidad en el mapeo de las dos imágenes, de forma similar a cómo percibimos el mundo a través de nuestros dos ojos, es posible obtener la profundidad de los objetos. Otro estudio [51] ha estimado TTC utilizando sensores catadióptricos con buenos resultados en tiempo real. Aunque los sensores de rango ultrasónico tienen un gran campo de visión, pueden aparecer problemas de interferencias si se usa más de un sensor simultáneamente. Como resultado, “la frecuencia de detección de obstáculos está limitada por la cantidad de sensores en uso y el tiempo requerido para que un eco regrese de un obstáculo"[42].

Una deficiencia importante de estos enfoques es que el sistema robótico podría aumentar su consumo de energía debido a los sensores empleados. Además, estos sensores pueden ser costosos de adquirir. Por lo tanto, para minimizar el consumo y el costo de energía, decidimos usar solo una cámara, es decir, visión monocular.

Diversos estudios han empleado la visión monocular para estimar TTC. Por ejemplo, el TTC se ha calculado a partir de imágenes en el espacio y los dominios espectrales [52]. En este contexto, el TTC se ha estimado utilizando un cambio temporal en los espectros de potencia entre las imágenes

sucesivas [53]; porque a partir de las características de frecuencia espacial, el espectro de potencia no cambia con el desplazamiento traslacional de la imagen y solo depende del cambio de profundidad (contigüidad espacial del objetivo) [54]. A pesar de que los estudios sugieren que existe una equivalencia completa entre el espacio y los dominios espectrales desde puntos de vista teóricos y de información [55], el cambio del dominio de tiempo al dominio de frecuencia implica un procesamiento adicional.

Además, el TTC se puede estimar a través del cálculo del flujo óptico [56], en especial si se tiene el acceso al componente traslacional del campo de flujo [57]. Estos métodos se basan en la esimación del flujo óptico, ya sea por algún método denso como el de Horn y Schunk [56], o uno poco denso como el de Lucas y Kanade [58]. En investigaciones como [59], el flujo óptico se utiliza para determinar la dirección y la velocidad del robot para el conocer siguiente paso, y en [60] se presenta un proyecto destinado a explorar y evaluar el potencial de flujo óptico basado en técnicas que tiene orientación hacia las personas con discapacidad visual para evitar obstáculos utilizando retroalimentación auditiva y táctil.

Los enfoques basados en flujo óptico [61][62][63] toman dos suposiciones básicas: i) el brillo de la imagen de los objetos en movimiento permanece constante (13), donde $E(x, y, t)$ es el brillo de imagen en la posición (x, y) al tiempo t, y ii) la divergencia de flujo se mide examinando las derivadas espaciales parciales de componentes de velocidad de imagen en direcciones ortogonales en una ubicación de imagen dada. La divergencia de flujo del campo de flujo óptico en el punto (x, y) se define comúnmente como (14) donde (x, y) es un punto en la imagen, y u y v son los componentes de velocidad de imagen en las direcciones x e y, respectivamente.

$$\frac{d}{dt}E(x, y, t) = 0 \tag{13}$$

$$\{\nabla \cdot [u(x,y), v(x,y)]\}(x,y) = \frac{\partial u(x,y)}{\partial x} + \frac{\partial v(x,y)}{\partial y} \tag{14}$$

Con esto, el TTC (τ) en un punto a lo largo del eje óptico de la cámara es medido a partir del divergencia del flujo y es comunmente definida en [64] como (15):

$$\tau = -\frac{Z}{T_r} = -\frac{Z}{D(x_0, y_0)} \tag{15}$$

donde Z es la distancia al objeto en la dirección de partida y T_r es la velocidad en esa dirección. Nótese que para el caso típico de un robot acercándose a una superficie, se mide $Z > 0$ y $T_r < 0$, por lo tanto se decrece el valor de Z conforme el robot se acerca. En particular, la divergencia $D < 0$ es negativa para el campo de la divergencia del flujo y τ es definida como positiva. La relación mencionada entre D y τ asume que la dirección del origen está alineada con el eje óptico de la cámara, en el centro de la imagen (x_0, y_0).

La divergencia de flujo es constante a través del plano de la imagen si el plano de la superficie es perpendicular al eje óptico de la cámara (es decir, *fronto-parallel* con el plano de la imagen), y puede, por lo tanto, ser calculado en cualquier parte del área de la imagen de la superficie. Si la alineación precisa *fronto-parallel* no se mantiene con el plano de estacionado (docking), entonces se introduce la deformación de la imagen, haciendo que la divergencia se mida para variar a través de la superficie proyectada. [65].

En [66], los autores presentan un método cooperativo de una manera que permite que cada robot pueda tomar ventaja de los recursos particulares de los otros robots en el ambiente. Un robot con la cámara calcula los valores de flujo óptico a partir de las imágenes que recibe de la cámara, y a

partir de estos valores se calcula el TTC, es decir, valores que determinan el tiempo que el robot tiene antes de chocar con un obstáculo delante de la misma.

Además, el foco de expansión (Focus of Expansion - FOE) se puede calcular desde el campo de flujo mediante diferentes métodos [67][20]. Por lo tanto, se puede estimar TTC para diferentes objetivos, por ejemplo, estacionarse (docking) y aterrizaje (landing) [68][69]. Una vez que se calcula el campo de flujo, es posible detectar la proximidad relativa a los objetos de acuerdo con la longitud de los vectores de flujo óptico como en los enfoques que se muestran en [70][71].

Para ilustrar someramente el foco de expansión, a continuación se describe la representación del movimiento visual. Considerando el sistema de coordenadas $OXYZ$ al centro óptico de una cámara, tal que el eje OZ coincide con el eje óptico como se muestra en la Figura 2. Se supone que la cámara se mueve rígidamente con respecto a su entorno estático 3D con movimiento traslacional (U, V, W) y movimiento rotacional (α, β, γ). Bajo la proyección perspectiva, las ecuaciones relacionadas con la velocidad 2D (u, v) de un punto en la imagen $p(x, y)$ a la velocidad 2D de un punto 3D proyectado $P(X, YZ)$ son (16) y (17)[72]:

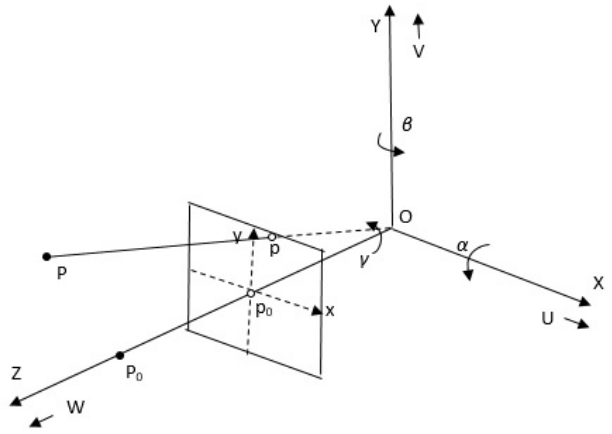


Figura 2: Sistema coordenado $OXYZ$ moviendose con el ojo, y las correspondientes coordenadas retinales *(x,y)*.

$$u = \frac{(-Uf + xW)}{Z} + \alpha\frac{xy}{f} - \beta\left(\frac{x^2}{f} + f\right) + \gamma y \tag{16}$$

$$v = \frac{(-Vf + yW)}{Z} + \alpha\left(\frac{y^2}{f} + f\right) - \beta\frac{xy}{f} + \gamma x \tag{17}$$

Las ecuaciones (16) y (17) describen un vector del campo de movimiento, que relaciona el movimiento 3D de puntos a su movimiento 2D proyectado sobre una imagen plana. De estas ecuaciones se puede concluir dos cosas.

1. El efecto de la traslación sobre el movimiento 2D observado es independependiente de la orientación, es decir, los componentes traslacionales y rotacionales del movimiento se pueden separar.

2. Los vectores definidos por los componentes traslacionales del campo de movimiento, recaen sobre las líneas que pasan por el punto $(x_0, y_0) \equiv (Uf/W, Vf/W)$, que es conocido como el foco de expansión (*Focus of expansion FOE*).

El FOE define la dirección del movimiento traslacional, y es de gran importancia para diversos problemas de análisis de movimiento. Finalmente el componete rotacional del movimiento es independiente de la estructura de la escena, dado que la profundidad Z influye solo en la componente traslacional. Si las cantidades W y Z son multiplicadas por el mismo factor de escala, el flujo definido por las ecuaciones (16) y (17) permanecen sin cambio. En otras palabras, hay una ambiguedad de escala que nos previene de diferenciar entre los objetos cercanos moviéndose lentamente y uno distante que se está moviendo rápido. Por lo tanto, la información relacionada a la componente

traslacional del egomotion que se puede recuperar a partir de las ecuaciones (16) y (17) es a lo más el foco de expansión.

Sin embargo, estos enfoques se basan en estimar el flujo óptico; por lo tanto, "son iterativos, necesitan trabajar en escalas múltiples, tienden a ser computacionalmente costosos y requieren un esfuerzo significativo para implementarse correctamente" [41].

En [41] se propone un método para resolver tres de los cuatro casos mostrados en la Figura 3.

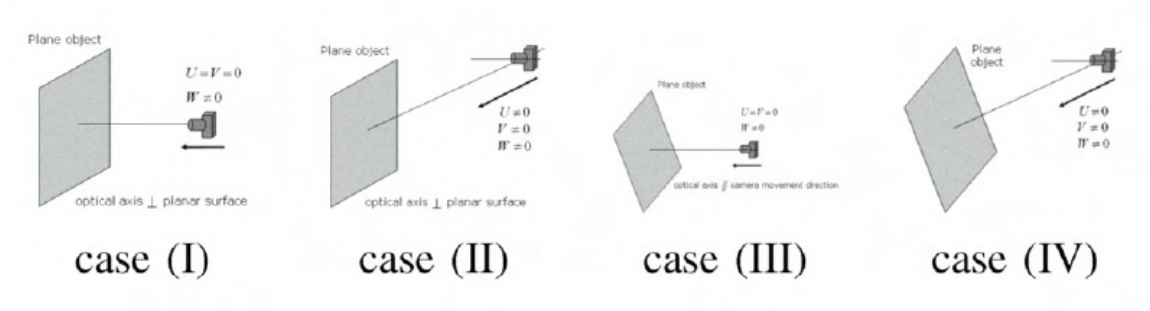


Figura 3: Cuatro casos del movimiento relativo

Basándose en la ecuación de la constancia de brillantez propuesta por Horn [56], la cual necesita solo derivadas de la imagen y no requiere detectar puntos de interés, se define $C = -W/Z$ (la inversa del tiempo al contacto) y $G = (xE_x + yE_y)$ como una manera corta de expresar el "*gradiente radial*", mientras que E_x, E_y y E_t son las derivadas parciales de brillantez x, y y t. También p y q que son las pendientes de la superficie plana en las direcciones X e Y.

Los resultados de los tres primeros casos de movimiento relativo se mencionan a continuación:

1. **caso 1:** Movimiento traslacional a lo largo del eje óptico para una superficie plana perpendi-

cular al eje óptico.

$$C = -\sum G\ E_t / \sum G^2 \tag{18}$$

2. **caso 2:** Movimiento traslacional en una dirección arbitraria relativo a una superficie plana que es perpendicular al eje óptico.

$$\begin{bmatrix} \sum E_x^2 & \sum E_x E_y & \sum G E_x \\ \sum E_x E_y & \sum E_y^2 & \sum G E_y \\ \sum G E_x & G E_y & \sum G^2 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = - \begin{bmatrix} \sum E_x E_t \\ \sum E_y E_t \\ \sum G E_t \end{bmatrix} \tag{19}$$

donde $A = f(U/Z)$, $B = f(V/Z)$, y el foco de expansión está dado por:

$$x_0 = -A/C \text{ y } y_0 = -B/C \tag{20}$$

3. **caso 3:** Movimiento traslacional a lo largo del eje óptico relativo a una superficie plana de orientación *arbitraria*.

$$\begin{bmatrix} \sum G^2 x2 & \sum G^2 xy & \sum G^2 x2 \\ \sum G^2 xy & \sum G^2 y^2 & \sum G^2 y \\ \sum G^2 x & \sum G^2 y & \sum G^2 \end{bmatrix} \begin{bmatrix} P \\ Q \\ C \end{bmatrix} = - \begin{bmatrix} \sum G x E_t \\ \sum G y E_t \\ \sum G E_t \end{bmatrix} \tag{21}$$

donde $P = (p/f)(W/Z_0)$, $Q = (q/f)(W/Z_0)$, esto es:

$$p = -f\frac{P}{C} \text{ y } q = -f\frac{Q}{C} \tag{22}$$

Estos métodos también suelen llamarse basados en "gradiente" dado que los elementos matriciales

en (21) son la función de gradiente de brillantez. Los casos 1, 2 y 3 son situaciones especiales del caso 4 y tienen solución. El caso más general (4) requiere de técnicas de optimización no lineales.

Otro método para estimar TTC es el 'Método directo' o IBD (Image Brightness Derivative), que trabaja directamente con las derivadas del brillo de la imagen y no requiere detección de características, seguimiento de características o estimación del flujo óptico [73]. Este método utiliza restricciones entre el gradiente de brillo (derivadas espaciales de brillo) y la derivada de brillo en el tiempo (13). Las derivadas espaciales y temporales del brillo se pueden usar para ubicar el punto de la imagen hacia el cual se está realizando el movimiento [74]. A pesar de que este método ha logrado buenos resultados para acercarse a las superficies, hay casos en los que la precisión se ve comprometida (por ejemplo, cuando el robot se acerca a muros no texturizados (planos) y el cambio de brillo es cero).

Además, el TTC se ha calculado utilizando cambios en el tamaño del obstáculo. Por ejemplo, los estudios en [42][75][76] han utilizado el hecho de que los animales y los insectos obtienen información de los objetos que se aproximan del tamaño aparente S' de los objetos y los cambios temporales en el tamaño $\frac{dS'}{dt}$. Esta información de aproximación se llama 'tau-margin' definida como (23). Tau-margin se deriva de (12) usando un tamaño característico del obstáculo en la imagen [77] y la aproximación de que el obstáculo es plano y paralelo al plano de la imagen.

$$\tau = -\frac{S'}{\frac{dS'}{dt}} \tag{23}$$

Sin embargo, (23) solo es adecuado cuando las deformaciones del contorno se pueden modelar mediante deformaciones afines. Un aspecto clave de este tipo de enfoque es que la precisión depende de la medición y una adecuada segmentación del obstáculo; por lo tanto, se deben aplicar métodos

más precisos para segmentar y seguir objetos. Por ejemplo, un método con resultados de segmentación fiables podría ser más preciso que los métodos basados en el flujo óptico o FOE, porque estos últimos a menudo son inexactos en las discontinuidades. Por este motivo, se han utilizado diferentes métodos para segmentar objetos y estimar su cambio de tamaño, utilizando, por ejemplo, contornos activos como ACAS (Active Contour Affine Scale) [39].

Otro enfoque que se ha utilizado para la evasión de obstáculos, es el cálculo del TTC, por lo que en [39] se presenta una comparación de tres métodos para medir el tiempo al contacto: ACAS, Scale Invariant Ridge Segments (SIRS) and Image Brightness Derivatives (IBD), donde experimentalmente y analíticamente los métodos ACAS fueron mejores.

Finalmente, otro enfoque para calcular TTC se basa en modelar el movimiento del robot. En este contexto, un estudio [43] ha calculado el TTC en el movimiento vehicular usando tres escenarios: i) una aceleración relativa constante, no nula entre el obstáculo y el vehículo anfitrión; ii) una aceleración cero y una velocidad relativa no nula; y iii) una distancia constante. En este estudio, los obstáculos se encuentran usando puntos de interés con un detector de esquina Harris [78]. La idea es prometedora; sin embargo, debido al uso de puntos de interés, se debe implementar un mecanismo para agrupar estos puntos en diferentes regiones que representan diferentes obstáculos.

Las características y desventajas de las tecnologías encontradas en la revisión de la literatura se resumen en el Cuadro 1 mientras que los métodos se encuentran en el Cuadro 2.

Con base en las características de los enfoques explicados anteriormente, la Figura 4 muestra dónde se ubica nuestro trabajo de investigación con base en la revisión de la literatura realizada, donde se observa que utilizamos visión monocular y para estimar el TTC nuestra propuesta se basa en el modelado del tamaño de obstáculos en el tiempo.

Cuadro 1: Resumen de características de tecnologías propuestas para estimar el TTC.

Enfoque	Idea principal	Inconvenientes
Visión estéreo	Calcula Z diréctamente	▪ Las cámaras deben ser calibradas previamente. ▪ Costoso en términos económicos y de consumo de energía.
Sonares	Calcula Z diréctamente	▪ Costoso en términos económicos y de consumo de energía.

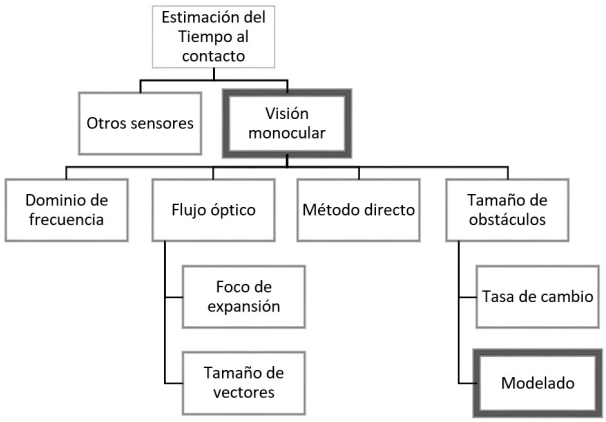


Figura 4: Clasificación de enfoques para calcular Tiempo al contacto y ubicación de nuestro trabajo en el estado del arte.

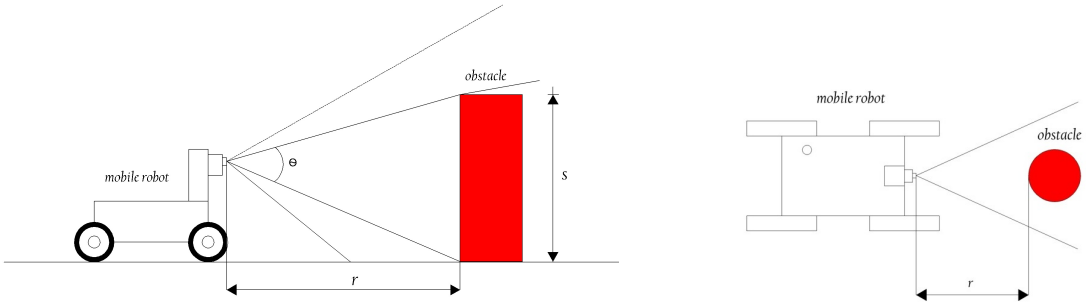
2.4. Cálculo del tamaño aparente del obstáculo

La Figura 5 presenta un modelo de robot móvil desde las vistas lateral (a) y superior (b). Como se puede observar, hay tres parámetros principales involucrados en el campo de visión de un obstáculo detectado: S, que representa el tamaño del objeto detectado; r, que representa la distancia entre

Cuadro 2: Resumen de características de métodos propuestos para estimar el TTC.

Enfoque	Idea principal	Inconvenientes
Power spectra	Los datos son procesados en dominio de frecuencia.	▪ Los datos deben ser transformados de dominio de tiempo a dominio de frecuencia y viceversa.
FOE	Depende de las ecuaciones del flujo óptico.	▪ Es costoso computacionalmente. ▪ Trabaja mejor sobre superficies con textura.
Flujo óptico	Depende de las ecuaciones del flujo óptico.	▪ Una vez que el movimiento es detectado, se deben implementar mecanismos precisos de segmentación.
Método directo o IBD	Depende de las derivadas de brillantez de la imagen.	▪ Trabaja mejor sobre superficies con textura. ▪ Es sesgado debido a la inclusión de fondo en el cálculo del TTC.
Tamaño de objeto	Calcula el cambio del tamaño del objeto en el tiempo.	▪ Se debe aplicar un buen método de segmentación. ▪ Puede tener problemas con objetos de forma irregular (cóncavos).

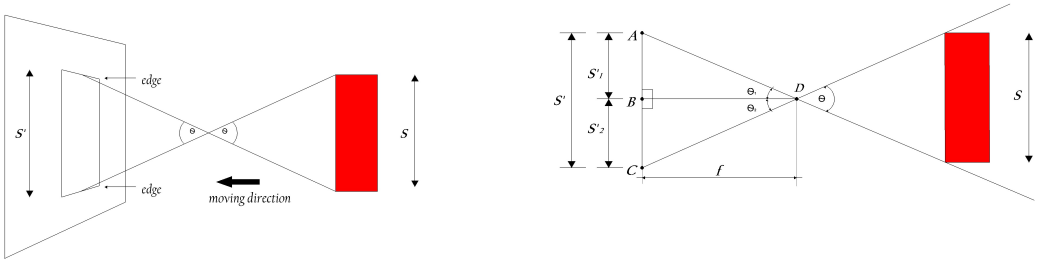
el robot móvil y el obstáculo detectado; y θ es la apertura del ángulo proporcional al tamaño del objeto.



(a) (b)

Figura 5: Modelo visual de un robot acercándose a un obstáculo. (a) Vista lateral del modelo del robot móvil. (b) Vista desde arriba del modelo del robot móvil.

La Figura 6 ilustra cómo se expande el tamaño aparente del objeto mientras se acerca a la cámara. Desde una perspectiva, el tamaño aparente S' (S proyectado en la imagen) crece proporcionalmente a medida que el robot se acerca al obstáculo (Figura 6(a)).



(a) (b)

Figura 6: Modelo de la cámara acercándose a un obstáculo. (a) Vista en perspectiva del modelo de la cámara. (b) Vista geométrica del modelo de la cámara.

Podemos encontrar varios casos. El primer caso se presenta en la Figura 6(b), donde $\theta_1 = \theta_2$. En este caso, para encontrar el valor de θ, el triángulo ACD (ver Figura 6(b)) se divide en dos triángulos rectángulos (ABD y BCD) y por ángulos opuestos por un vértice, tenemos (24).

$$\theta_1 = \theta_2 = \frac{\theta}{2} \tag{24}$$

Para obtener θ_1, basado en el triángulo rectángulo ABD y sus propiedades, tenemos la equivalencia (25). Luego, dando valores a esta equivalencia, obtenemos (26), donde f es la distancia focal. Finalmente, θ está dado por (27). En el caso de que $\theta_1 \neq \theta_2$ podemos dividir S' en S'_1 y S'_2, asociados con θ_1 y θ_2 entre sí, y θ estará dado por (28).

$$\tan(\angle ADB) = \frac{AB}{BD} \tag{25}$$

$$\tan(\theta_1) = \frac{\frac{S'}{2}}{f} \tag{26}$$

$$\theta = 2\left[\arctan\left(\frac{\frac{S'}{2}}{f}\right)\right] \tag{27}$$

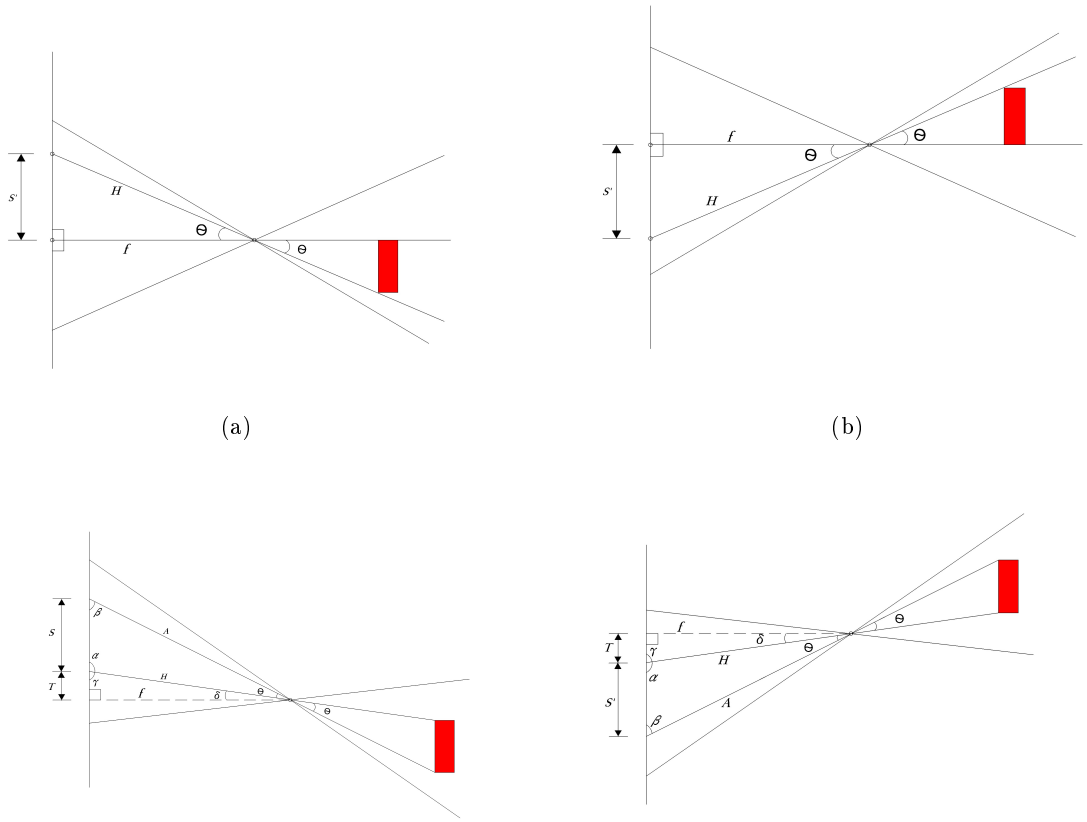


Figura 7: Diferentes casos del modelo de la cámara mirando a un obstáculo que se aproxima. (a) S' empieza del punto central de la imagen hacia arriba. (b) S' empieza del punto central de la imagen hacia abajo. (c) El centro de proyeción está abajo de la imagen proyectada (d) El centro de proyeción está arriba de la imagen proyectada

$$\theta = \theta_1 + \theta_2 = \arctan\left(\frac{S_1'}{f}\right) + \arctan\left(\frac{S_2'}{f}\right) \tag{28}$$

Sin embargo, el obstáculo puede no proyectarse uniformemente en el plano de la imagen, es decir, el obstáculo puede proyectarse debajo o encima del centro de la imagen ($\theta_1 = 0$ o $\theta_2 = 0$).

Ejemplificamos dos escenarios más para delinear cinco casos posibles y generalizar las formas de abordar un obstáculo. El primer escenario muestra cuando θ está por encima del centro de la imagen (Figura 7 (a)) y el segundo es cuando θ comienza debajo del centro de la imagen (Figura 7 (b)) .

Los otros escenarios muestran cuando el obstáculo está por debajo o por encima del eje principal y la región T es diferente de cero (Figura 7 (c) y 7 (d)). Cuando esto sucede, se forma un triángulo rectángulo fTH, donde $H = \sqrt{f^2 + T^2}$. Como T y f se pueden conocer, podemos obtener $\angle\gamma$ por (29). Entonces, $\angle\delta$ y $\angle\alpha$ se calculan usando (30) y (31), respectivamente.

$$\angle\gamma = \arctan\left(\frac{f}{T}\right) \tag{29}$$

$$\angle\delta = 90 - \angle\gamma \tag{30}$$

$$\angle\alpha = 180 - \angle\gamma \tag{31}$$

Podemos obtener A como se muestra en (32) usando la ley de los cosenos. Con esto, podemos obtener el ángulo β en (33) usando la ley de los senos. Finalmente, θ viene dado por (34).

$$A = \sqrt{S'^2 + H^2 - 2S'H\cos\angle\alpha} \tag{32}$$

$$\angle\beta = \arcsin\left(\frac{H\sin\angle\alpha}{A}\right) \tag{33}$$

$$\theta = 180 - (\angle\beta + \angle\alpha) \tag{34}$$

La ecucación (34) está en términos de $\angle\beta$ y $\angle\alpha$ para mayor simplicidad, aunque sustituyendo los cálculos, la ecuación final quedaría como se muestra en (35).

$$\theta = 180 - \left(\arcsin\left(\frac{H\sin\left(180 - \arctan\left(\frac{f}{T}\right)\right)}{\sqrt{S'^2 + H^2 - 2S'H\cos\left(180 - \arctan\left(\frac{f}{T}\right)\right)}}\right) + 180 - \arctan\left(\frac{f}{T}\right)\right) \tag{35}$$

A partir de estos dos escenarios comunes, se pueden resumir cinco posibles casos:

1. El centro de la imagen se encuentra entre la imagen proyectada del objeto, es decir, $S' = S_1 + S_2$. Esto incluye $S_1 \neq S_2$ y $\theta_1 \neq \theta_2$ (ver Figura 6 (b)).

2. S' comienza desde el punto central de la imagen y hacia arriba, es decir, $T = 0$ (ver Figura 7 (a)).

3. S' comienza desde el punto central de la imagen y hacia abajo, es decir, $T = 0$ (ver Figura 7(b)).

4. S' está completamente sobre el punto central de la imagen, es decir, $T \neq 0$ (ver Figura 7 (c)).

5. S' está completamente debajo del punto central de la imagen, es decir, $T \neq 0$ (ver Figura 7 (d)).

2.5. Propuesta de la estimación del TTC

Mi propuesta se basa principalmente en tres áreas de investigación: *Visión por computadora*, porque la estimación de TTC se basa en sensores pasivos; *Probabilidad y Estadística*, debido al uso de métodos probabilísticos para encontrar obstáculos y ajustar resultados; e *Identificación de sistemas*, porque se extraen métodos de esta área para conocer el comportamiento del entorno del robot y predecir posibles colisiones.

Teniendo en cuenta los cinco casos anteriores que el robot podría encontrar en el entorno, decidí estimar el TTC utilizando un enfoque de modelado debido a las siguientes tres razones:

1. Debido a los cambios en la intensidad de la luz, un enfoque de segmentación de color puede estimar el tamaño del objeto de forma incorrecta. Por ejemplo, supongamos que un robot se aproxima al obstáculo, entonces suponemos que el tamaño aparente del obstáculo en el tiempo $t + 1$ será mayor que el tamaño del objeto en el tiempo t. Sin embargo, si medimos el tamaño del objeto con errores, podría ocurrir que en el tiempo $t + 1$ la imagen de obstáculo parezca más pequeña que en el tiempo t. En consecuencia, el robot supondría que se está moviendo hacia atrás y, en el peor de los casos, colisionaría con el obstáculo.

2. Intento mitigar el problema asociado a superficies de terreno irregulares con surcos. Dada esta situación, el robot obtendría estimaciones inconsistentes sobre la segmentación de obstáculos y del TTC.

3. La estimación de TTC debería ser menos costosa desde el punto de vista computacional y en tiempo real. Además, cualquiera de los dos problemas anteriores podría causar oscilaciones (picos) en las estimaciones de TTC debido a problemas de percepción. Valdría la pena si el robot pudiera reconocer las estimaciones correctas y descartar las incorrectas. Proponemos un enfoque donde se modela el tipo de movimiento, de modo que podamos identificar estimaciones incorrectas (valores atípicos) para resolver los primeros dos problemas. Además, con base en el modelo estimado, predeciremos las estimaciones para los siguientes cuadros en lugar de segmentar objetos en cada cuadro; por lo tanto, el costo computacional se reduce.

En función de estos problemas, dividimos nuestra solución propuesta en tres módulos: módulo de segmentación, módulo de modelado y módulo de estimación TTC y toma de decisión (Figura 8). Primero, se obtiene la ubicación del obstáculo en cada cuadro usando uno de los diferentes enfoques planteados en este documento (color, contornos probabilísticos o aprendizaje de características de obstáculos). Luego, se obtienen y modelan las mediciones para generar modelos que permitan describir de manera robusta el comportamiento (Filtro de Kalman, series temporales o identificación de sistemas). Finalmente, la estimación TTC se calcula a través de las predicciones realizadas en la etapa anterior y se toma una decisión para evitar la colisión. Estos tres módulos se describen en detalle en los siguientes capítulos.

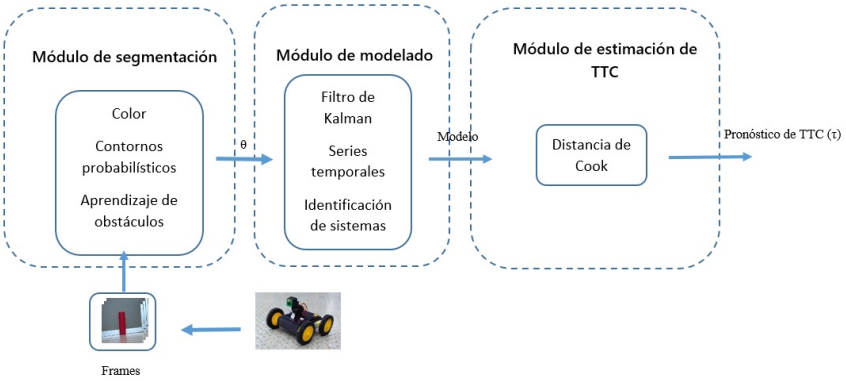


Figura 8: Metodología para la estimación del TTC mediante 3 módulos.

En este segundo capítulo se presentaron los antecedentes teóricos del problema del egomotion mostrando algunos trabajos que abordan este problema. También trabajos que están directamente relacionados con el TTC,y se describen los métodos comunes de calcular el TTC para ubicar la propuesta dentro del estado del arte de la literatura.

3. Capítulo III: Módulo de segmentación de obstáculos

Debido a que el TTC se puede estimar por la tasa de cambio del tamaño de la proyección S' (ec. 23), el robot debe medir el tamaño de los obstáculos proyectados. El proceso de segmentación divide una imagen en regiones que corresponden únicamente a los objetos que se deben evitar. Por esta razón, se presentan a continuación 3 métodos propuestos para la identificación de estos obstáculos y la obtención de su tamaño aparente: color, contornos probabilísticos y basado en aprendizaje de características.

3.1. Segmentación por color

Como se describió anteriormente, el valor de τ, está basado en el cambio en el tamaño aparente S' del objeto con el que el robot podría colisionar. Para calcular el tamaño del objeto, una primera propuesta es la segmentación por color, es decir, elegimos un color para encontrarlo en la imagen, y verificamos en cada píxel si este es del mismo color (o similar usando un umbral de tolerancia). Por ejemplo, si queremos saber si un píxel tiene un grado similar de color rojo por el cual segmentamos, el valor debe estar en el rango *[R(píxel) - umbral, R (píxel) + umbral]*, y así para los valores de azul y verde en un modelo tradicional RGB.

Esto es importante porque puede calcularse usando cómputo paralelo debido a que es la misma operación por cada pixel. En el cuadro 2, se muestra un análisis entre el número de comprobaciones que se deben realizar en diferentes tamaños de imágenes. Por ejemplo, en una imagen de 640 x 480 píxeles debemos evaluar 307.200 píxeles. Para cada píxel debemos comparar 3 colores (rojo, verde y azul), es decir, 921 600 comparaciones. Finalmente, cada color se debe comparar dos veces, es

decir, cuando se comprueba contra el límite inferior formado por el valor del color menos el umbral y cuando se compara con el límite superior formado por el valor del color sobre el umbral.

Cuadro 3: Comparación del número de revisiones por tamaño de imágenes.

Tamaño de la imagen	Total de pixeles	Revisiones con color	Revisiones con umbral
640 x 480	307 200	921 600	1 843 200
1920 x 1080	2 073 600	6 220 800	12 441 600

Es importante mencionar que este enfoque tiene un buen desempeño siempre y cuando se cumplan las siguientes dos características:

1. La textura del obstáculo es lisa.
2. El obstáculo solo tiene un color.
3. El color del obstáculo es diferente al resto del ambiente.

La Figura 9, muestra un ejemplo de segmentación por color de un obstáculo rojo en el ambiente, donde se aprecia una segmentación ruidosa debido a cambios de iluminación.

(a) (b)

Figura 9: Ejemplo de segmentación por color. (a) Ejemplo de una escena con un obstáculo rojo en el campo visual. (b) Ejemplo de una segmentación por color rojo.

3.2. Implementación paralela de segmentación por color

Como resultado de estas tareas más sofisticadas, deben de surgir acciones que permitan combinar la precisión y la rapidez en la resolución de dichas tareas. En los últimos años, se ha hablado mucho del cambio de la industria de la informática generalizada a la computación paralela. Con el paso del tiempo, este cambio ya no es exclusivo de las supercomputadoras o mainframes, sino que cada vez más se encuentra al alcance de todos los usuarios, pues actualmente ya contamos con computadoras portátiles y dispositivos móviles (celulares, tabletas, etc) que incluyen varios núcleos.

Como propuesta en este trabajo, se muestra el uso del cómputo paralelo para segmentar por color, utilizando las ventajas que la Unidad de Procesamiento Gráfico *(GPU)* que una computadora con tarjeta gráfica de este tipo nos ofrece. Esto debido a que, el buscar un color en una imagen es realizar al misma operación a cada elemento de la misma.

En comparación con el procesamiento de datos tradicional de la Unidad Central de Procesamiento (CPU), el realizar cálculos de propósito general en una Unidad de Procesamiento Gráfico (GPU) es un concepto novedoso. De hecho, el propio GPU es relativamente nuevo en comparación con el campo de la informática en general. Sin embargo, la idea de la computación en los procesadores gráficos no es tan nueva como se podría pensar [79].

"El modelo de computación sobre tarjetas gráficas consiste en usar conjuntamente una CPU (Central Processing Unit) y una GPU de manera que formen un modelo de computación heterogéneo" [80]. La parte secuencial de una aplicación se ejecutaría sobre la CPU (comúnmente denominada host) y la parte más costosa del cálculo se ejecutaría sobre la GPU (que se denomina device).

3.2.1. Compute Unified Device Architecture: CUDA

La tarjeta GeForce 8800 GTX de NVIDIA, fue la primera GPU construida con una arquitectura CUDA. Esta arquitectura incluyó varios componentes nuevos diseñados estrictamente para el cómputo GPU y dirigida para solventar muchas de las limitaciones que impedían que los procesadores gráficos anteriores fueran útiles para el cómputo de propósito general.

NVIDIA pretendía que esta nueva familia de procesadores de gráficos puedieran usarse para propósito general, por lo que las Unidades Aritmética-Lógica de cada chip, comenzaron a ser capaces de realizar aritmética con precisión de punto flotante y así tener instrucciones que pudieran usarse para cualquier propósito y no necesariamente para gráficos [79]. Además, las unidades de ejecución en la GPU permitieron la lectura arbitraria y escritura a la memoria, así como el acceso a una memoria caché de software conocida como memoria compartida. Todas estas características de la arquitectura CUDA se añadieron en la GPU para realizar tareas además de las tareas gráficas tradicionales.

Unos meses después, con el lanzamiento de la GeForce 8800 GTX, NVIDIA hizo publico un lenguaje para este compilador, *CUDA C* (por sus siglas en inglés Compute Unified Device Architecture) y con eso, *CUDA C* se convirtió en el primer lenguaje diseñado por una empresa para facilitar el uso de la GPU para propósito general.

3.2.2. Arquitectura de funcionamiento

Como sabemos, una imagen puede verse como una estructura tipo matriz. Sin embargo, la estimación del tiempo al contacto depende de la segmentación por color, por lo tanto, como se muestra en la Figura 11, puesto que cada pixel en una imagen en formato RGB está formado de tres valores:

rojo, verde y azul (ver Figura 10), es necesario convertir la estructura de imagen a una estructura arreglo.

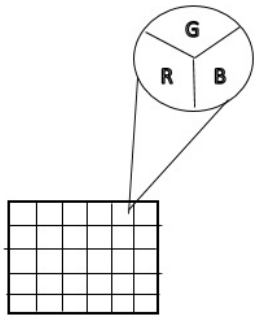


Figura 10: Valores en cada pixel de una imagen.

Para el propósito anterior, nos apoyamos del conjunto de funciones de la biblioteca OpenCV. OpenCv es una biblioteca de código abierto para visión por computadora. Esta biblioteca está escrita en el lenguaje C y C++. Fue diseñada para eficiencia computacional con enfoque en aplicaciones de tiempo real [81]. El diseño de la arquitectura, comunicación e interacción entre las partes de ésta, es mostrada en la Figura 11. Para esta comunicación entre el programa que usa las funciones de CUDA C y el programa que usa las funciones OpenCV, se construyó una biblioteca en C, para que el programa que hacía uso de la GPU mediane CUDA pudiera utilizar las funciones programadas en OpenCV en otro archivo con extensión $.h$.

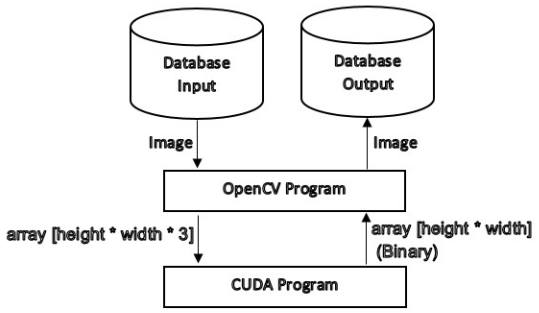


Figura 11: Arquitectura de comunicación entre programas.

3.2.3. Transformación imagen-Arreglo

Con funciones de OpenCV se carga cada imagen en memoria y son transformadas en un arreglo unidimensional que pueda ser procesado por CUDA. Este arreglo es de tamaño = *alto* x *ancho* x 3, debido a que necesitamos los 3 valores de cada pixel de la matriz. Para acomodar estos valores en un arreglo hubieron 2 propuestas. La primera, como se muestra en la Figura 12, se dividen por color, es decir, los primeros (*alto* x *ancho*) lugares son los valores en rojo para todos los pixeles de la matriz. De la posición (*alto* x *ancho*) hasta ((*alto* x *ancho* x 2)- 1) son los valores de color verde para todos los pixeles, y de la posición (*alto* x *ancho* x 2) hasta ((*alto* x *ancho* x 3)- 1) son los valores en azul para cada pixel. En cada espacio de color, los valores en el arreglo fueron llenados por filas.

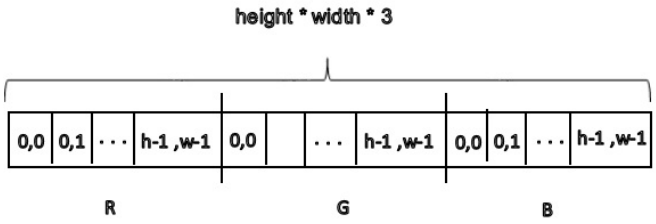


Figura 12: Propuesta 1 de la estructura unidimensional de una imagen.

En la segunda propuesta, mostrada en la Figura 13, el llenado es por pixel, es decir, los primeros 3 elementos corresponden a los valores en rojo, verde y azul respectivamente, del primer pixel (posición $(0, 0)$ en la imagen). Los siguientes 3 elementos en el arreglo son los valores rojo, verde y azul del segundo pixel (posición $(0, 1)$ en la imagen), y así hasta llegar a los últimos 3 elementos en el arreglo que pertenecen al último pixel de la imagen (posición $(altura - 1, ancho - 1)$ en la imagen). Como se puede apreciar, el vaciado de los pixeles en el arreglo también es por fila.

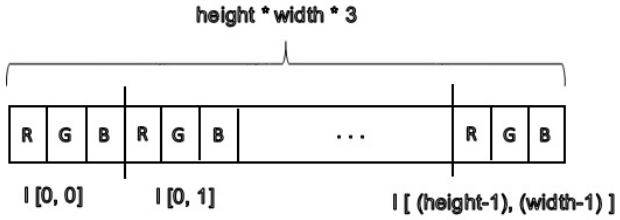


Figura 13: Propuesta 2 de la estructura unidimensional de una imagen.

3.2.4. Uso de bloques e hilos

Como se mencionó en el apartado de segmentación por color, el ir revisando pixel por pixel suele ser muy costoso entre más grandes sean las imágenes. Sin embargo, podemos notar que la misma pregunta por cada color es realizada a cada pixel, por lo tanto, si se tuvieran múltiples procesadores que ejecutara cada uno la misma operación a cada pixel, el tiempo de ejecución sería menor. CUDA propone una manera sencilla para trabajar de manera paralela con los *bloques* de la tarjeta de video (GPU), es decir, el número de bloques que tiene la GPU, es el número de procesos paralelos que puede ejecutar.

En tiempo de ejecución, CUDA permite separar cada *bloque* en *hilos* [79]. Esto es, si quisiéramos tener N procesos, podemos ejecutar N bloques, o $N/2$ bloques con 2 hilos cada bloque. Para saber el

número total de operaciones iguales a cada pixel, se utiliza la ecuación (36). Lo anterior nos ayudará a conocer cuantos procesos paralelos se necesitan.

$$Total_{operaciones} = alto * ancho \tag{36}$$

Cuando sabemos el número de procesos paralelos, podemos dividir este número en diferentes bloques e hilos. Para lo anterior, necesitamos saber cuantos hilos podemos ocupar por bloque, y se obtiene con una propiedad de la tarjeta de video llamada $maxThreadsPerBlock$ como se muestra en (37). Por último, tratando de utilizar todos los hilos por bloques, la cantidad de bloques necesarios se obtiene de (38).

$$n_{Hilos} = MaxHilosPBloque \tag{37}$$

$$n_{Bloques} = \lceil Total_{operaciones}/n_{Hilos} \rceil \tag{38}$$

3.2.5. Análisis de Complejidad

Un tema importante a comentar y analizar es la complejidad de la segmentación de manera secuencial y la paralela. Si tomamos en cuenta el proceso de segmentación, en la manera secuencial se debe ir recorriendo pixel por pixel, y no se puede evaluar uno hasta que no se haya evaluado el anterior. Por lo tanto, si n es el número de filas y m es el número de columnas de una imagen n x m, entonces el número de preguntas es = (n x m x 6) debido a que por cada pixel se evalúan 3 colores

y por cada color 2 umbrales. En el peor de los casos, cuando n y m son iguales, la complejidad es *cuadrática* $O(n^2)$.

Por otro lado, en cualquiera de las dos propuestas mencionadas sobre la transformación a un arreglo unidimencional, se hacen el mismo número de preguntas, sin embargo, puesto que se hacen de manera paralela, la complejidad es *constante* $O(1)$.

Otra característica a mencionar, es el número de procesos que aumentan o disminuyen según las dimensiones de la imagen. Si por ejemplo si se tiene una imagen de 640 x 480 y otra de 1920 x 1080 pixeles, pudiera pensarse que el tiempo aproximado en procesar la segunda imagen sería 3 veces mayor, puesto que aproximadamente triplica el número de filas y el número de columnas. Sin embargo, esto no es así, si no que sería aproximadamente 9 veces más tiempo que la primera, puesto que como vimos en la parte secuencial, la complejidad es cuadrática y depende del número de filas y columnas.

3.2.6. Obtención del tamaño aparente

Una vez segmentado el objeto mediante el color, se necesita saber el tamaño del objeto. Para ello basta con saber cual es el pixel segmentado más arriba y cual es el que está más abajo, y con la resta de ellos se obtiene S'. Debido a que se quiere saber cómo cambia el tamaño con respecto al tiempo, también podría medirse el ancho en vez del alto.

3.3. Segmentación por contornos probabilísticos

En los métodos de segmentación de color [76], es necesario conocer los colores de los obstáculos, que pueden tener pequeñas cantidades de colores diferentes. Además, estos métodos pueden generar errores con cambios imprevistos en la iluminación, por lo que los tonos de color del objeto deben recalcularse.

Un enfoque de segmentación diferente es inferir la posición de un contorno cerrado que delinea un obstáculo que debe evitarse. Estos modelos se conocen como serpientes o modelos de contorno activos [82]. El contorno activo “busca un límite cercano que sea consistente con las características de la imagen local"[83]. Una ventaja clave de este enfoque es que es menos vulnerable a los cambios en la intensidad de la luz debido al uso de gradientes.

Sin embargo, es necesario mencionar algunos inconvenientes, como la forma de los objetos, ya que se genera una cubierta convexa. Además, los puntos de control deben inicializarse cerca del obstáculo. Sin embargo, una vez que se obtiene el polígono del obstáculo en el primer cuadro, el método buscará el obstáculo en los píxeles cercanos en los siguientes cuadros.

Se puede ver en la Figura 14 (a), que el objeto segmentado por color presenta partes sombreadas debido a las condiciones de iluminación. Por el contrario, la Figura 14 (b) muestra el mismo objeto que la Figura 14 (a) con resultados de segmentación confiables debido al uso de un enfoque de segmentación basado en el contorno activo.

(a) (b)

Figura 14: Segmentación de obstáculos por dos diferentes enfoques: (a) Ejemplo de segmentación por color de un obstáculo. (b) Ejemplo de segmentación por contorno activo de un obstáculo.

La Figura 15 muestra el método para obtener el contorno probabilístico, donde se observa que a cada imagen le es aplicado un operador de detección de bordes. Estos bordes sirven para que iterativamente se actualicen los puntos de control del contorno mediante la inferencia con Maximum Aposteriori Probability (MAP) y así obtener como resultado el tamaño aparente del obstáculo segmentado. Los detalles de este procedimiento son descritos a continuación.

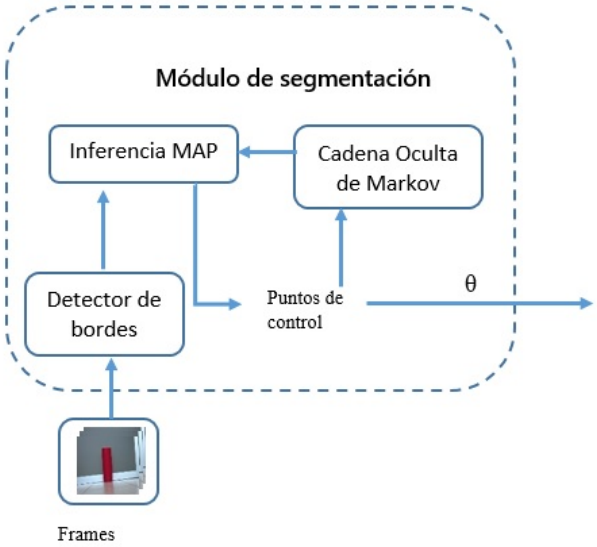


Figura 15: Metodología del proceso de segmentación mediante contornos probabilísticos.

3.3.1. Modelos ocultos de Markov

Un Modelos Oculto de Markov o HMM (por sus silas en inglés: Hidden Markov Model), es un modelo de visión común, que utiizamos en nuestra propuesta para realizar la tarea de segmentación. Un HMM consiste principalmente en: i) estados ocultos, que no se observan; ii) las probabilidades de transición de estados; y iii) observaciones, que son funciones probabilísticas de los estados [84]. En HMM, observamos una "secuencia de mediciones $\{x_n\}_{n=1}^{N}$, cada una de las cuales describe algo sobre el correspondiente estado discreto del mundo $\{w_n\}_{n=1}^{N}$ " [85].

Para nuestra tarea de segmentación, el objetivo es inferir las posiciones de un conjunto de puntos de control $\{w_n\}$ que forman una cubierta convexa que describe el objeto segmentado en función de los datos de imagen x. El propósito de utilizar un HMM (en particular, una estructura tipo cadena) se debe a que cuando cada punto de control en el contorno se ve como un estado, cada uno depende del anterior para hacer que los puntos de control estén lo más cerca posible. Las cadenas de Markov se

pueden representar mediante un modelo gráfico no dirigido, que corresponde a una descomposición factorizada dada por (39).

$$Pr(W) = \Phi_{n,n-1}(w_n, w_{n-1})...\Phi_{i,i-1}(w_i, w_{i-1})...\Phi_{2,1}(w_2, w_1) \tag{39}$$

donde $\Phi_{i,i-1}$ es un factor de probabilidad conjunta.

3.3.2. Inferencia con Maximum A posteriori Probability (MAP)

En el proceso de búsqueda del mejor contorno, una solución candidata especifica una posición para cada punto de control, a partir de un conjunto de posiciones posibles de k en una cuadrícula discreta. Podemos escribir una solución candidata como $(x_1, ...x_n)$ y hay k^n soluciones candidatas [83]. La probabilidad de tomar un valor particular $w_n = k$ es baja en regiones constantes; mientras que es alta en las posiciones en la imagen donde la intensidad cambia rápidamente (es decir, los bordes). Además, "los puntos vecinos están conectados y tienen una fuerza de atracción: es más probable que estén cerca el uno del otro" [85].

La inferencia se puede hacer usando un método de programación dinámica [86]. Para realizar esta tarea, utilizamos una inferencia de probabilidad máxima a posteriori MAP (por sus siglas en inglés: Maximum A posteriori Probability), es decir, buscamos estimar la secuencia más probable (máximo a posteriori) w de estados como el modo de la distribución posterior, dado un modelo ω y un conjunto de datos x.

En inferencia MAP, se tiene una cadena con N variables desconocidas $\{w_n\}_{n=1}^{N}$, con K valores posibles. La solución MAP está dada por (40) y es una expresión de la regla de Bayes vista como

un problema de minimización. Sustituyendo en la expresión por la probabilidad log, obtenemos (41) que tiene la forma general en (42) [85].

$$\hat{w}_{1...N} = \underset{w_{1...N}}{\operatorname{argmin}} \left[-\log \left[Pr\left(x_{1...N}, w_{1...N}\right) \right] \right] \tag{40}$$

$$\hat{w}_{1...N} = \underset{w_{1...N}}{\operatorname{argmin}} \left[-\sum_{n=1}^{N} \log \left[Pr\left(x_n | w_n\right) \right] - \sum_{n=2}^{N} \log \left[Pr\left(w_n | w_{n-1}\right) \right] \right] \tag{41}$$

$$\hat{w}_{1...N} = \underset{w_{1...N}}{\operatorname{argmin}} \left[\sum_{n=1}^{N} U_n(w_n) + \sum_{n=2}^{N} P_n(w_n, w_{n-1}) \right] \tag{42}$$

donde U_n es un término unario que depende solo de una variable w_n, y P_n es un término por pares que depende de dos variables w_n y w_{n-1} [85]. Para dar valores a estos términos, usamos una función de energía simple para formalizar el problema como se describe en [83]. El término unario indica qué tan buena es una posición particular para el punto de control i^{th} (los datos de imagen RGB x). U_n debe ser pequeño donde el gradiente de la imagen es alto, por lo que el contorno se ve atraído por los límites de intensidad. En consecuencia, para este caso utilizamos un detector de bordes, como un filtro Sobel [87], que devuelve valores pequeños cuando el punto está en un borde y valores altos en caso contrario (está en regiones planas o lisas). La probabilidad de este término viene dada por (43), donde la función $sobel(x, w)$ devuelve la magnitud del operador de bordes Sobel (también puede usarse Canny). El término por pares codifica un término, que generalmente tiene la forma $P_n(w_n, w_{n-1}) = ||w_n - w_{n-1}||^2$ fomentando que el contorno sea corto. Como se puede ver en (42), la restricción de forma del contorno generado solo busca minimizar las distancias entre los puntos de control y acercarse a los bordes.

$$Pr(x|W) \propto \prod_{n=1}^{N} exp[sobel(x, w_n)] \tag{43}$$

Para optimizar la función de costo en la ecuación (42), aplicamos el algoritmo de Viterbi (un algoritmo de optimización de programación dinámica) a la secuencia de estados $w_1, ...w_n$. También es equivalente a un caso especial de propagación de creencias de máximo producto. Para comenzar el proceso, dos de los ocho puntos de control se inicializan manualmente, y los otros 6 se colocan arbitrariamente y cerca del obstáculo en el primer cuadro. El algoritmo de Viterbi se repite usando un par diferente de puntos adyacentes elegidos para ser fijados en cada paso. A medida que continúa el procedimiento de inferencia, " el contorno se mueve a través de la imagen y finalmente se fija en el límite de un objeto " [85].

Para encontrar el obstáculo en el siguiente cuadro, el contorno se expande asumiendo que el robot se acercó al obstáculo (por lo tanto, el tamaño del objeto crecerá en el peor de los casos). Es decir, los puntos ubicados en la parte superior de la imagen tomarán posiciones superiores; mientras que los puntos ubicados en la parte inferior de la imagen tomarán posiciones aún más bajas (o hasta sus límites). El mismo caso aplica con aquellos que se encuentran más hacia las direcciones derecha e izquierda. Obviamente, esta expansión debería ser consistente con la velocidad de aproximación del robot. Aunque esta expansión puede ser un parámetro de usuario dado, también se puede estimar dependiendo de la velocidad a la que se acerca el robot, ya que la magnitud de expansión aumenta de acuerdo con la velocidad del robot. En otras palabras, cuanto más rápido se acerca el robot, mayor debe ser la expansión.

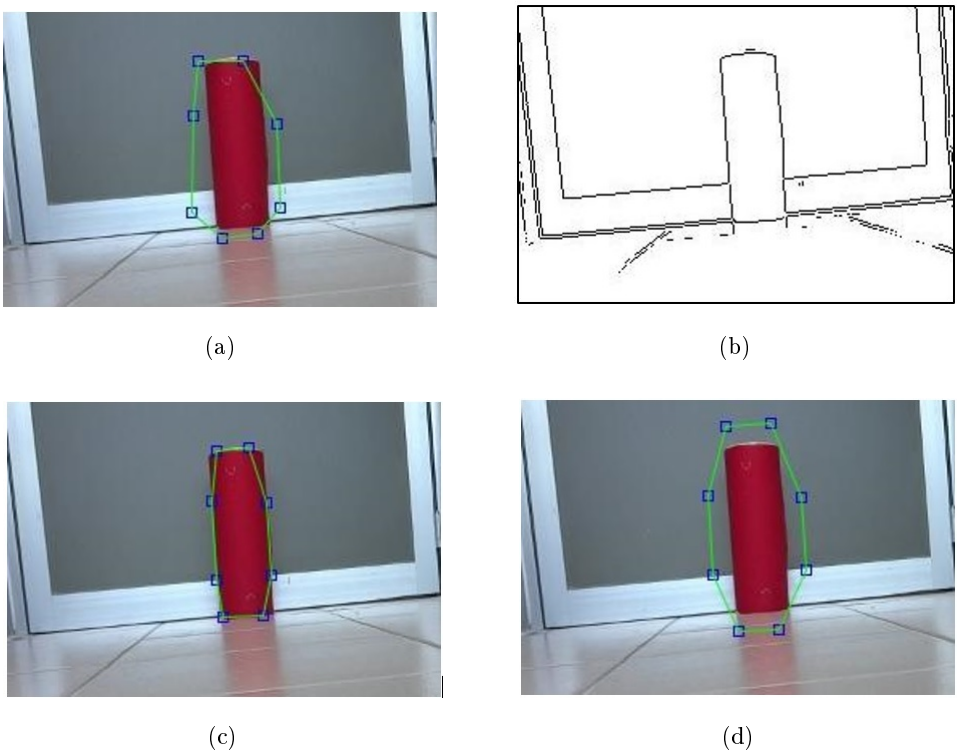


Figura 16: Propuesta del proceso de segmentación para identificar obstáculos. (a) Inicialización de puntos de control. (b) Resultado de aplicar el filtro Sobel. (c) Resultado del proceso de segmentación. (d) Expansión de puntos de control para encontrar el obstáculo en el siguiente frame.

La figura 16 muestra un ejemplo del proceso de segmentación. Como se puede observar, primero se realiza un proceso de inicialización para obtener puntos los de control (información a priori) (Figura 16 (a)). Los cuadrados azules representan las ventanas de 3 x 3, desde donde se inferirán los puntos de control. Luego, se aplica un filtro de Sobel a la imagen (Figura 16 (b)), que se utilizará como información para el término unario del modelo. Después de algunas iteraciones, la segmentación finaliza (Figura 16 (c)). Finalmente, los puntos de control se expanden (Figura 16 (d)). Estos puntos de control expandidos se usarán para segmentar el obstáculo en el siguiente cuandro.

Un problema que podría surgir cuando se construye el contorno, ocurre cuando comenzamos a

segmentar el obstáculo si está lejos. Se debe tener en cuenta que los cambios en la proyección son mínimos; en consecuencia, el efecto del ruido en la estimación del contorno puede conducir a una estimación de parámetros inapropiada.

Por esta razón, proponemos inicializar los puntos si el objeto a segmentar excede un umbral γ en el rango $(0, 1)$, donde este umbral indica el porcentaje del objeto en comparación con la imagen completa. Si el objeto tiene un tamaño lo suficientemente grande, el ruido en la segmentación se puede minimizar. Experimentalmente, sugerimos un $\gamma = .05$.

Otro tema importante para mencionar es que es posible perder el seguimiento. Podemos intuir que el robot ha perdido el objeto cuando la diferencia absoluta del tamaño aparente θ_{t_i} con respecto a $\theta_{t_{i-1}}$ supera el umbral ϵ , como se representa en (44). En tal caso, el valor de theta θ_{t_i} no debe incluirse en el modelo y debe indicarse que el obstáculo se perdió.

$$abs(\theta_{t_i} - \theta_{t_{i-1}}) > \epsilon \tag{44}$$

3.4. Segmentación por aprendizaje de características

Los enfoques anteriores funcionan relativamente bien cuando se tiene enfocado el obstáculo con el que puede colisionar el robot, pero el problema surge cuando tenemos múltiples obstáculos en la escena. Sin embargo, hay otro problema aún más serio que responde a la siguiente pregunta: ¿Qué es un obstáculo para un robot que usa una sola cámara?. Uno de los objetivos de la visión por computadora es poder describir los objetos que el agente percibe a través de un sensor pasivo como una cámara monocular.

Para que un robot sepa qué objeto está percibiendo, es necesario que conozca el objeto de antemano. Por esta razón, un enfoque más robusto que se propone utilizar en este trabajo, se centra en explorar esta posibilidad.

La Figura 17 muestra el método para segmentar los obstáculos a partir del aprendizaje de características, donde se observa que primero existe un proceso de extracción de características y aprendizaje, para después ubicarlas en cada imagen que toma el robot. Por último se realiza un proceso de filtrado para obtener resultados robustos y evitar datos erróneos o atípicos. Los detalles de este procedimiento son descritos a continuación.

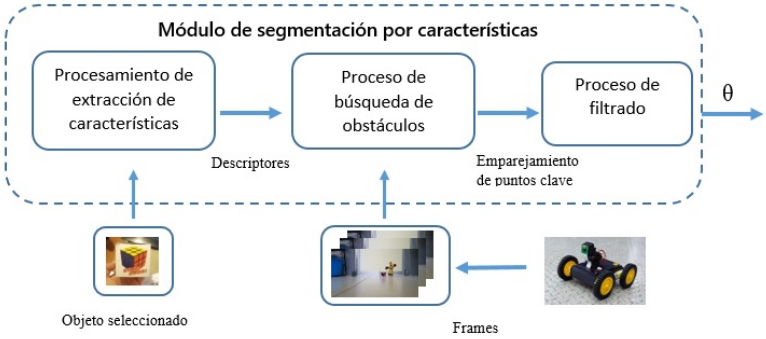


Figura 17: Metodología del proceso de segmentación mediante extracción de características.

3.4.1. Detección de puntos clave (Keypoints)

Para poder encontrar obstáculos en su ambiente, el robot extraerá características importantes de los obstáculos de entrenamiento, que se buscarán en la escena. Una **punto clave** es un área de interés en la imagen que es única y, por lo tanto, fácilmente reconocible. Esta característica se puede encontrar en áreas de la imagen con mucha textura. Incluso esta característica se puede describir, según su orientación relativa, lo que es útil para encontrarla independientemente de su rotación.

Las **esquinas** y las áreas con mucha textura son buenas características. Los **bordes** son buenas características porque tienden a dividir en dos regiones una imagen. Un **blob** es un área de una imagen que difiere de sus áreas alrededor y también es una buena característica. Muchos algoritmos de detección de características giran alrededor de identificación de esquinas, bordes y blobs, con algunos enfocándose en el concepto de "ridge", el cual puede conceptualizarse como el eje de simetría de un objeto elongado.

Por tal motivo, exploramos e investigamos los algoritmos que existen, implementarlos en Opencv y Python 2.7 (versión soportada por el robot móvil) para ver las ventajas y desventajas de cada uno. Los métodos presentados en este documento son:

1. Harris: detectar esquinas
2. Good Featrues to track: detectar esquinas
3. SIFT: detectar puntos de interés con propiedades
4. SURF: detectar blobs
5. FAST: detectar esquinas
6. BRIEF: detectar blobs
7. ORB: Significa Oriented FAST y Rotated BRIEF

3.4.2. Detección de esquinas

Para la detección de esquinas, hay varios algoritmos descritos en la literatura. Este es el caso del método Harris [78], en el cual, se construye una matriz 2x2 basada en derivadas parciales de

imágenes en escala de grises y se analizan los eigenvalores. Un punto que se denomine esquina, es aquel donde ambos eigenvalores tendrían valores grandes. Si solo un eigenvalor es grande entonces se trata de un borde. Por último, si ninguno es grande entonces el pixel o punto se encuentra en una región lisa.

El detector de equinas de Harris se desempeña bien en muchos casos, pero se pierde en algunos otros. Después del artículo original del Harris y Stephens, Shi y Tomasi describieron un mejor detector [88]. Ellos usaron una función de puntuación diferente para mejorar la calidad en general. Usando este método se pueden encontrar las N esquinas más "fuertes" en la imagen . Es muy útil cuando no queremos usar todas las esquinas para extraer información a partir de la imagen.

Los métodos anteriores son buenos para detectar esquinas independientemente de la rotación que tengan; sin embargo, cuando queremos que coincidan las características de los obstáculos con las de la escena, estas características no siempre tendrán la misma rotación, puesto que podrían también estar a diferentes escalas, lo que hace que tenga una calidad de esquina superior o inferior. Cuando hablamos sobre el contenido de una imagen, queremos una firma que sea invariante a cuestiones como escala, rotación e iluminación.

3.4.3. Descriptores SIFT

SIFT (Scale Invariant Feature Transform) es uno de los algoritmos más populares en visión por computadora [89]. Este algoritmo se usa para extraer keypoints y construir descriptores de las características correspondientes. Para identificar keypoints potenciales, SIFT construye una pirámide que reduce el tamaño de una imagen y toma la diferencia de Gaussianas. Esto significa que aplicamos un filtro Gaussiano en cada nivel y se toma la diferencia para construir los niveles sucesivos de la

pirámide. Para ver si el punto actual es un keypoint, mira a los vecinos, así como los píxeles en la misma ubicación en los niveles vecinos de la pirámide. Si éste es un máximo, entonces el punto actual es escogido como punto clave. Con esto se asegura que mantenemos keypoints invariantes a la escala.

Para la invarianza a la rotación, una vez que se identifican los keypoints, a cada uno se le asigna una orientación. Tomamos la vecindad alrededor de cada keypoint y calculamos la magnitud del gradiente y la dirección. Si tenemos esta información, seremos capaces de hacer match de este keypoint con el mismo punto en otra imagen incluso si está rotado. Si ya tenemos la orientación, se normalizan esos keypoints antes de hacer comparaciones.

Ya que tenemos lo anterior, necesitamos cuantificarlos. Para hacer esto tomamos una vecindad de 16x16 alrededor de cada keypoint y la dividimos en 16 bloques de 4x4. Para cada bloque, se calcula el histograma en orientación con 8 contenedores. Entonces tenemos un vector de longitud 8 asociado con cada bloque, lo que significa que el vecino es representado por un vector de tamaño 128(8x16). Este es el descriptor final del keypoint que será usado. Si extraemos N keypoints de una imagen, tendremos N descriptores de longitud 128 cada uno. Este arreglo de N descriptores caracteriza la imagen. Hacemos énfasis en que SIFT no detecta keypoints, sino que describe la región que los rodea por medio de un vector de características.

3.4.4. SURF

SURF (Speeded Up Robust Features) es un algoritmo de detección de características publicado en 2006 por Herbert Bay [90], el cual es varias veces más rápido que SIFT, y parcialmente inspirado en él. SIFT es computacionalmente intensivo. Esto significa que es lento y se tienen pro-

blemas al implementarlo en aplicaciones de tiempo real. SIFT usa diferencias de Gaussianas para construir la pirámide y este proceso es lento. Para superar esto, SURF usa un “box filter"simple para aproximar la Gaussiana. El descriptor de SURF se basa en propiedades similares a SIFT, con una complejidad aún más reducida. El primer paso consiste en fijar una orientación reproducible basada en la información de una región circular alrededor del punto de interés. Luego, se construye una región cuadrada alineada con la orientación seleccionada, y se extrae el descriptor de SURF de ella. El problema de SURF es que no es libre para aplicaciones comerciales.

3.4.5. FAST

Incluso, aunque SURF es más rápido que SIFT, no es lo suficiente para sistemas en tiempo real, especialmente cuando hay limitaciones de recursos. Necesitamos algo que sea realmente rápido y que no sea costoso computacionalmente. Por lo tanto, Rosten y Drummond aportan con **FAST (Features From Accelerated Segment Test)**, y como su nombre lo indica realmente es rápido [91].

En lugar de realizar todos los cálculos costosos, este algoritmo sugiere una prueba de alta velocidad para rápidamente determinar si el punto actual es un punto clave potencial. FAST implementa una prueba de alta velocidad, la cual intenta dar un salto rápidamente a la prueba de 16 pixeles. Este algoritmo dibuja un círculo alrededor del pixel incluyendo 16 pixeles. Luego marca cada pixel más brillante o más obscuro que un umbral particular comparado con el centro del círculo. Una esquina es definida por la identificación de un número de pixeles contiguos marcados como más brillantes o más obscuros. Debemos recalcar que FAST es solo para detecciones de Keypoints. Una vez que estos son detectados, necesitaremos usar SIFT o SURF para calcular los descriptores.

3.4.6. BRIEF

Hasta ahora sabemos que FAST es muy rápido para detectar esquinas y usamos SIFT o SURF para calcular descriptores. Ahora es necesaria una manera más rápida para calcular descriptores. **BRIEF (Binary Robust Independent Elementary Features)** por otro lado, no es un detector de características, sino un descriptor [92]. Al no poder detectar esquinas por sí mismo, se utiliza en conjunto con un detector de keypoints. Lo bueno de BRIEF es que es compacto y rápido.

BRIEF es uno de los descriptores más rápidos disponibles. La teoría atrás de BRIEF es algo complicada, pero basta con decir que BRIEF adapta una serie de optimizaciones que lo hacen una muy buena opción para “feature matching".

3.4.7. ORB

Ahora llegamos a la mejor combinación de todas las combinaciones que hemos discutido. Si SIFT es nuevo y SURF lo es aún más, ORB es aún más reciente que ambos. **Oriented Fast and Rotated Brief (ORB)** fue publicado en 2011 como una manera alternativa para SIFT y SURF. Es un algoritmo que surge en los laboratorios de OpenCV. Es rápido, robusto y de código abierto [93]. Ambos SIFT y SURF son algoritmos patentados y no se pueden usar para propósitos comerciales. Esto es por lo que ORB es bueno entre otras cosas.

ORB mezcla técnicas usadas en la detección de keypoints de FAST y descriptores de BRIEF, por esta razón fue necesaria la descripción de los métodos anteriores, anque fuera de manera general. En el artículo, los autores buscan obtener los siguientes resultados:

- La adición de un componente rápido y preciso a FAST.

- El cálculo eficiente de características orientadas BRIEF.
- Análisis de varianza y correlación de características orientadas BRIEF.
- Un método de aprendizaje para quitar correlación de características BRIEF bajo invarianza de rotación, mejorando el desempeño en aplicación del vecino más cercano.

ORB trata de optimizar operaciones, incluyendo el paso importante de utilizar BRIEF en una forma de sensibilidad a la rotación de manera que se mejore la coincidencia (“match") incluso en situaciones en las que una imagen de entrenamiento tiene una rotación muy diferente a la imagen de consulta.

3.4.8. Aprendizaje (encontrando obstáculos)

Con las características de los métodos mencionados anteriormente, se decidió realizar una combinación de características entre los obstáculos de entrenamiento y la escena, para ubicar áreas de la imágen, pasa saber la región donde pueden ubicarse los obstáculos en el campo visual.

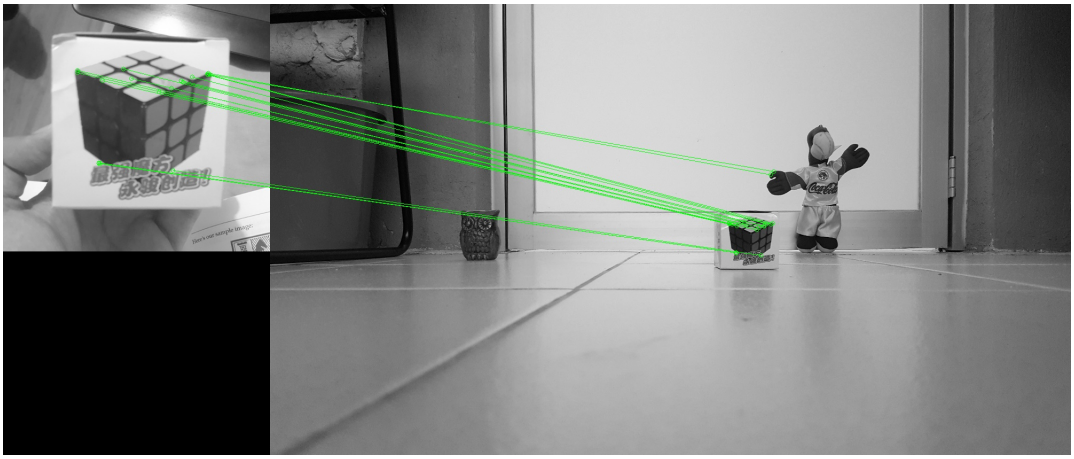
La figura 18 muestra un ejemplo de un obstáculo para encontrar en el ambiente y los puntos clave o keypoints encontrados mediante el algoritmo FAST que se buscará en la imagen de la escena para segmentar el obstáculo.

Una vez que los puntos clave del obstáculo se han encontrado en la escena, puede haber algunos puntos clave no coincidentes. Estos puntos clave erróneos los llamaremos **atípicos u outliers**. Para descartar los valores atípicos, agrupamos los puntos clave en regiones midiendo la distancia entre ellos mediante (45). Si alguno de ellos excede una distancia α al resto del grupo, entonces ese punto se considera atípico y se descarta. La figura 19 muestra un ejemplo del resultado de este proceso de

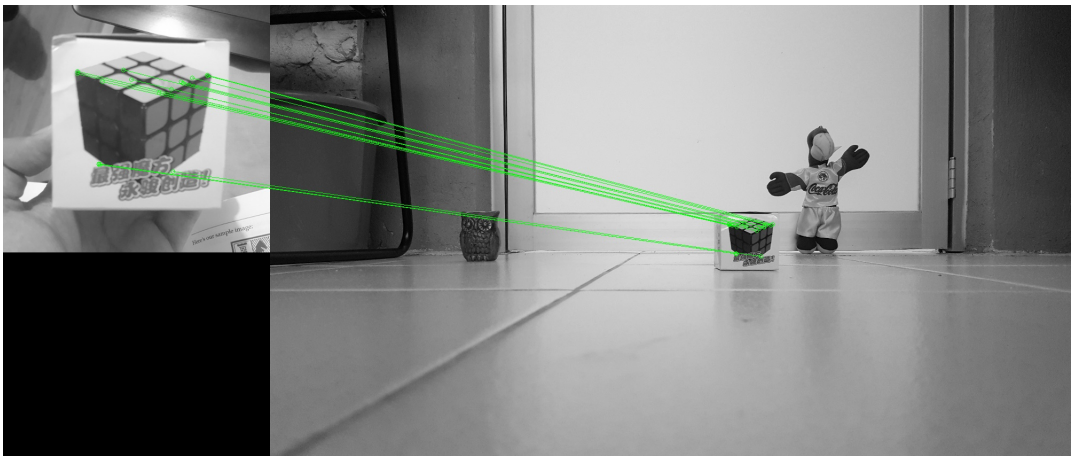
Figura 18: Ejemplo de puntos clave (keypoints) a partir de un obstáculo.

filtrado. La figura 19(a) muestra dos emparejamientos atípicos y la Figura 19(b) muestra el resultado de descartar esas dos coincidencias.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{45}$$



(a)



(b)

Figura 19: Ejemplo del descarte de outliers en el proceso de emparejamiento (matching). (a) Coincidencia con datos atípicos (outliers). (b) Coincidencia sin datos atípicos (outliers) después del proceso de filtrado.

Finalmente, cuando el robot tiene una región de puntos clave que describen un obstáculo, es necesario conocer la altura del blob, ya que la altura será la característica de la región que se seguirá a lo largo de los frames para saber cómo cambia con el tiempo. Para calcular la altura, simplemente restamos la posición de la fila en la que está el punto clave más alto, a la posición de la fila del punto

clave que está más bajo. La Figura 20 muestra en un recuadro rojo, la región segmentada donde se encontró el objeto que aprendió el robot.

Figura 20: Resultado de la segmentación de obstáculos mediante aprendizaje de puntos clave.

Como nota importante, no estamos interesados en segmentar el objeto completo, sino simplemente la región de puntos que puede seguirse a lo largo del tiempo. En el ejemplo de la Figura 20, la parte blanca de la caja que se toma como un obstáculo, no nos interesa porque no es una buena región para seguir.

En este tercer capíulo se presentaron las propuestas para el módulo de segmentación: por color, por contornos probabilísticos y por segmentación por características. Como conclusión, el método para segmentar dependerá de las características del ambiente y de los obstáculos, pues sin los obstáculos no tienen textura que puedan ser ubicados en el mundo real, el método de segmentación por contornos probabilísticos puede ser utilizado para seguir a los obstáculos a lo largo del tiempo y es invariante al cambio de iluminación. Por otro lado, si los obstáculos tienen características fácilmente reconocibles, estas pueden ser aprendidas para ser buscadas en el ambiente.

4. Capítulo IV: Módulo de modelado de tamaño aparente de obstáculos

Los seres humanos estamos constantemente construyendo modelos para obtener una mejor comprensión del mundo; por lo tanto, aprendemos a controlar nuestras acciones al predecir sus efectos. Estas predicciones se basan en un modelo innato ajustado a la realidad, que utiliza la experiencia pasada.

El enfoque de modelar en nuestro problema, es una de mis principales contribuciones, pues busco robustecer las estimaciones mediante datos "coherentes" del cambio del tamaño aparente. Cabe mencionar se busca modelar el cambio del tamaño aparente y no del TTC en sí. Lo anterior es debido a que TTC está en función de $S'(t)$ y no de $\frac{d}{dt}S'(t)$. Por esta razón, en este capítulo se presentan 3 enfoques utilizados para el modelado del tamaño aparente de los obstáculos: Filtro de Kalman, series de tiempo e Identificación de sistemas.

4.1. Filtro de Kalman

Supongamos que tenemos un robot móvil acercándose hacia un obstáculo y tenemos que generar un modelo del tamaño del obstáculo, es decir, cuando se acerca al obstáculo, la cámara detecta que el obstáculo está creciendo. Este es un escenario para el filtro de Kalman, porque el método es parte de los modelos temporales y de seguimiento, que aplicados a nuestro problema, significa el seguimiento del tamaño aparente de cualquier obstáculo en el tiempo. La característica principal de los modelos temporales es que relacionan el estado del sistema con el tiempo $t-1$ y t como lo muestra (46).

$$w_t = \mu_p + Fw_{t-1} + \epsilon \tag{46}$$

En (46), w_t es un vector n-dimensional de los componentes de estado, F es una matriz de transferencia o de transición de dimensión nxn, que relaciona la media del estado en el tiempo t con el estado en el tiempo $t-1$ y ϵ es una variable aleatoria (generalmente llamada ruido del proceso) asociada con eventos aleatorios o fuerzas que afectan directamente el estado real del sistema, y que se distribuye normalmente y determina qué tan estrechamente relacionados están los estados en los momentos t y $t-1$ [94].

También en (46), podemos ver que se trata de un modelo recursivo, porque suponemos que cada estado depende solo de su predecesor. Suponemos que es condicionalmente independiente de los estados $w_1, ..., w_{t-2}$ dado su predecesor inmediato w_{t-1} y solo modela la relación condicional $Pr(w_t|w_{t-1})$ [85].

A continuación damos una breve descripción del caso específico del filtro de Kalman. No es nuestra intención explicar en detalle el filtro de Kalman, solo queremos resaltar algunas características y explicar la aplicación a nuestro problema.

4.1.1. Descripción

El filtro de Kalman es un conjunto de ecuaciones matemáticas, descritas por primera vez en [95], donde se presenta una solución recursiva al problema de filtrado lineal de datos discretos. Este método ha sido ampliamente investigado y aplicado en varios campos porque nos proporciona un mecanismo computacional (recursivo) eficiente para estimar el estado de un proceso. El filtro es

poderoso porque implica estimaciones de estados pasados, presentes e incluso futuros. El filtro de Kalman involucra estos elementos con el uso del conocimiento del sistema y dispositivo de medición, la descripción estadística de los ruidos del sistema y cualquier información disponible sobre las condiciones iniciales de las variables de interés [96].

Para comenzar, debemos recordar que la idea básica del filtro de Kalman es, que bajo un conjunto de suposiciones, será posible, dado un historial de mediciones de un sistema, construir un modelo para el estado del sistema que maximice la probabilidad a posteriori de mediciones previas. Podemos maximizar una probabilidad a posteriori sin tener una larga historia de mediciones anteriores. En cambio, "podemos actualizar iterativamente nuestro modelo de estado de un sistema y mantener solo ese modelo para la próxima iteración"[94]. Estas iteraciones están formadas principalmente por procesos de predicción, medición y actualización del estado.

Antes de explicar el proceso utilizado, debemos recordar que el filtro de Kalman se basa en 3 suposiciones:

1. La evolución del espacio de estados es lineal.
2. Los errores o ruido sujetos a las mediciones son "blancos".
3. Este ruido también es gaussiano.

En otras palabras, la primera suposición significa que el estado del sistema en el tiempo t puede modelarse como una matriz multiplicada por el estado en el tiempo $t-1$. Eso es bueno porque los sistemas lineales son más fáciles de manipular y prácticos que los no lineales. Las suposiciones adicionales de que el ruido es blanco y gaussiano significa que el ruido no está correlacionado en el

tiempo y que su amplitud puede ser modelada con precisión utilizando una media y una covarianza (es decir, el ruido se describe por completo en su primer y segundo momento)

4.1.2. Filtrado del tamaño aparente en el tiempo

A continuación se brindará una breve explicación de cómo se usa el filtro de Kalman para predecir nuevas medidas de tamaño aparente del obstáculo a evitar. Se hace hincapié en que no modelamos el TTC como tal, sino que modelamos el comportamiento del tamaño aparente del obstáculo proyectado en la imagen porque el TTC depende de este crecimiento.

Entonces, en algún momento, determinamos el tamaño aparente θ a ser θ_1. Sin embargo, debido a las imprecisiones inherentes al dispositivo de medición (como los cambios en la intensidad de la luz o el piso no liso mencionado anteriormente), el resultado de las mediciones es algo incierto. Entonces decidimos que la precisión es tal que la desviación estándar involucrada es σ_1 (solo una variable). Por lo tanto, podemos establecer la probabilidad condicional, el valor en el tiempo t_1, condicionado al valor observado de la medición θ_1, es decir, tenemos la probabilidad de que θ tenga un valor, basado en la medida que tomamos. En este momento, nuestra mejor estimación de $\hat{\theta}_1 = \theta_1$ y la varianza $\hat{\sigma}^2_{\theta_1} = \sigma^2_{\theta_1}$.

Después, el robot toma otra medida basada en la segmentación (cualquier de los planteados anteriormente) en el momento t_2, por lo tanto, se obtiene θ_2 con una varianza $\sigma^2_{\theta_2}$ (la cual se asume que es menor que la primera). Para combinar estas medidas y obtener una nueva con su propia variación (distribución Gaussiana), se utilizan las ecuaciones (47) y (48), donde se puede ver que el nuevo valor es solo una combinación ponderada de las dos medias medidas y la ponderación está determinada por las incertidumbres relativas de las dos mediciones (media condicional). El peso

en estas ecuaciones se puede ver como: si σ_{θ_1} es mayor que σ_{θ_2} (es decir, mayor variabilidad), σ_{θ_2} tendría más peso porque tiene menos variabilidad. También la incertidumbre en la estimación de la nueva θ se ha reducido al combinar las dos piezas de información [96].

$$\mu = \theta_{12} = \left(\frac{\sigma^2_{\theta_2}}{\sigma^2_{\theta_1} + \sigma^2_{\theta_2}}\right)\theta_1 + \left(\frac{\sigma^2_{\theta_1}}{\sigma^2_{\theta_1} + \sigma^2_{\theta_2}}\right)\theta_2 \tag{47}$$

$$\sigma^2_{\theta_{12}} = \frac{\sigma^2_{\theta_1}\sigma^2_{\theta_2}}{\sigma^2_{\theta_1} + \sigma^2_{\theta_2}} \tag{48}$$

Ahora que sabemos cómo obtener una próxima medida, podemos continuar con este proceso N veces (N mediciones). Esto se debe a que se puede combinar los dos primeros, luego el tercero con la combinación de los dos primeros, el cuarto con la combinación de los tres primeros y así sucesivamente [94]. Esto es lo que sucede cuando estamos haciendo un seguimiento el tiempo, obtenemos una medida seguida de otra y seguida de otra.

Usualmente, la ecuación (47) se reescribe como (49) y la ecuación (48) como (50) porque con estas nuevas formulaciones, se puede separar la información anterior de la nueva. La nueva información $(\theta_2 - \theta_1)$ se llama *innovación*.

$$\hat{\theta}_2 = \hat{\theta}_1 + \frac{\hat{\sigma}^2_{\theta_1}}{\hat{\sigma}^2_{\theta_1} + \sigma^2_{\theta_2}}(\theta_2 - \hat{\theta}_1) \tag{49}$$

$$\hat{\sigma}^2_{\theta_2} = \left(1 - \frac{\hat{\sigma}^2_{\theta_1}}{\hat{\sigma}^2_{\theta_1} + \sigma^2_{\theta_2}}\right)\hat{\sigma}^2_{\theta_1} \tag{50}$$

Finalmente, la ecuación (51) muestra el factor de actualización iterativo óptimo, que se conoce como la *ganancia de actualización* K o *ganancia de Kalman*, y así, obtenemos la forma recursiva descrita de forma general en (52) y (53). Para una explicación más detallada, sugerimos al lector revisar [96].

$$K = \frac{\hat{\sigma}^2_{\theta_1}}{\hat{\sigma}^2_{\theta_1} + \sigma^2_{\theta_2}} \tag{51}$$

$$\hat{\theta}_t = \hat{\theta}_{t-1} + K(\theta_t - \hat{\theta}_{t-1}) \tag{52}$$

$$\hat{\sigma}^2_{\theta_t} = (1 - K)\hat{\sigma}^2_{\theta_{t-1}} \tag{53}$$

4.2. Series de tiempo

El pronóstico es el resultado del análisis de fenómenos o eventos que suceden a través del tiempo; es decir, se evalúan las características del fenómeno a través del tiempo y con base en dichas características se puede tener un conocimiento anticipado de un suceso futuro. Para esto, definimos una serie de tiempo como un conjunto de observaciones cronológicas de un fenómeno que tiene ocurrencia en un periodo de tiempo definido.

Por lo tanto, el pronóstico de series de tiempo o series temporales es el uso de un modelo para predecir valores futuros basados en valores observados previamente. ARIMA por sus siglas en inglés *Autoregressive Integrated and Moving Average* es un modelo probabilístico que supone que los

errores tienen una distribución normal con media cero y varianza σ^2. Esta suposición se llama *ruido blanco*. Además, esto supone que no hay autocorrelación en los errores. La expresión utilizada para representar esta suposición se muestra en (54).

$$\varepsilon_t \sim N(0, \sigma^2) \tag{54}$$

Si bien los métodos de suavizado exponencial no hacen suposiciones sobre las correlaciones entre valores sucesivos de la serie temporal, en algunos casos podríamos hacer un mejor modelo predictivo teniendo en cuenta las correlaciones en los datos.

4.2.1. Autocorrelación

Para introducir el uso de un correlograma, primero es necesario describir el fundamento que existe detrás de su construcción; para ello, se debe definir la Autocorrelación.

Algunas propiedades importantes de una serie de tiempo están dadas por un conjunto de cantidades a las cuales llamaremos *Coeficientes de autocorrelación*, las cuales miden la correlación entre dos puntos, observados en diferentes instantes del tiempo. Estos coeficientes de autocorrelación a menudo nos proporcionan una idea sobre el modelo probabilístico que se pretende utilizar. Para describir la construcción de la autocorrelación, decimos que sean N pares de observaciones de dos variables x e y; entonces, el coeficiente de correlación ordinario esta dado por (55).

$$r = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{[\sum_{i=1}^{N}(x_i - \bar{x})^2 \sum_{i=1}^{N}(y_i - \bar{y})^2]}} \tag{55}$$

Una idea similar es la que se ocupa para evaluar la correlación que existe entre dos observaciones sucesivas de la misma variable; esto es, dadas N observaciones, $x_1, x_2, ..., x_N$ de una serie de tiempo discreta, es posible formar $N-1$ parejas de observaciones de la forma (x_1, x_2), (x_2, x_3),..., (x_{N-1}, x_N). De acuerdo a estas parejas de observaciones, es posible pensar que el primer elemento de estas parejas es una variable y el segundo elemento también es una variable; entonces, la correlación entre x_t y x_{t+1} esta dada por (56), donde $\bar{x}_{(1)}$ es la media de las primeras $N-1$ observaciones y está dado por (57), mientras que $\bar{x}_{(2)}$ es la media de las últimas $N-1$ observaciones y está dado por (58).

$$r_1 = \frac{\sum_{t=1}^{N-1}(x_t - \bar{x}_{(1)})(x_{t+1} - \bar{x}_{(2)})}{\sqrt{[\sum_{t=1}^{N-1}(x_t - \bar{x}_{(1)})^2 \sum_{t=1}^{N-1}(x_t - \bar{x}_{(2)})^2]}} \tag{56}$$

$$\bar{x}_{(1)} = \sum_{t=1}^{N-1} \frac{x_t}{N-1} \tag{57}$$

$$\bar{x}_{(2)} = \sum_{t=1}^{N-1} \frac{x_{t+1}}{N-1} \tag{58}$$

Al coeficiente que resulta de la (56) se le llama **Coeficiente de autocorrelación**. Simplificando el cálculo de este coeficiente de autocorrelación, podemos observar que $\bar{x}_{(1)} \simeq \bar{x}_{(2)}$, por lo que podemos simplificar como (59), donde $\bar{x}$ es la media aritmética de todos los datos dada por (60).

$$r_1 = \frac{N \sum_{t=1}^{N-1}(x_t - \bar{x})(x_{t+1} - \bar{x})}{(N-1)\sum_{t=1}^{N}(x_t - \bar{x})^2} \tag{59}$$

$$\bar{x} = \sum_{t=1}^{N} \frac{x_t}{N} \tag{60}$$

Además, cuando N es suficientemente grande, la fracción $N/(N-1)$ se aproxima a 1, por lo que podemos simplificar la autocorrelación como (61).

$$r_1 = \frac{\sum_{t=1}^{N-1}(x_t - \bar{x})(x_{t+1} - \bar{x})}{(N-1)\sum_{t=1}^{N}(x_t - \bar{x})^2} \tag{61}$$

Adicionalmente, esta forma se puede generalizar para cualquier distancia k entre dos observaciones, resultando en (62), donde a k se le conoce como retraso o lag.

$$r_1 = \frac{\sum_{t=1}^{N-k}(x_t - \bar{x})(x_{t+k} - \bar{x})}{(N-1)\sum_{t=1}^{N}(x_t - \bar{x})^2} \tag{62}$$

A continuación se detallan las características y bases del modelo utilizado para el pronóstico en serie de tiempo.

4.2.2. Modelo Autoregresivo: AR

El modelo *autorregresivo (AR)* [97][98] especifica que la variable de salida depende linealmente de sus propios valores previos. Su configuración se basa en el uso de datos observados en el pasado para desarrollar los coeficientes del modelo AR. Una vez que se establece el modelo, las ocurrencias futuras pueden predecirse por las ocurrencias actuales. El modelo autorregresivo de orden uno, simbolizado por $AR(1)$, parte de la ecuación (63), donde los errores ε_t satisfacen una secuencia de ruido blanco.

$$X_t = c + \phi X_{t-1} + \varepsilon_t \tag{63}$$

Analizando las características de este proceso se pueden obtener algunos supuestos necesarios para que la realización de este sea congruente. Reescribiendo este modelo de la forma (64) y obteniendo su valor esperado obtenemos la derivación (65-70).

$$X_t - \phi X_{t-1} = c + \varepsilon_t \tag{64}$$

$$E[X_t - \phi X_{t-1}] = E[c + \varepsilon_t] \tag{65}$$

$$E[X_t] - \phi E[X_{t-1}] = E[c] + E[\varepsilon_t] \tag{66}$$

$$\mu - \phi\mu = c \tag{67}$$

$$(1 - \phi)\mu = c \tag{68}$$

$$\mu = \frac{c}{(1 - \phi)} \tag{69}$$

Esto nos indica que cuando $t \to \infty$ el valor al que converge la variable X_t es μ, sin embargo, como se puede observar en la ecuación (69), si la variable X_t es no estacionaria implicaría que el valor de $|\phi| = 1$ y esto indicaría que $\mu \to \infty$. Por tal motivo, al realizar modelos bajo este enfoque, es de estricta importancia que la variable X_t sea estacionaria en la media.

Ahora analizando las covarianzas de la ecuación (64), se tiene (70) y (71).

$$\gamma_0 = \frac{\sigma^2}{(1-\phi)^2} \tag{70}$$

$$\gamma_i = \left[\frac{\phi^i}{(1-\phi)^2}\right] * \sigma^2 \tag{71}$$

Similarmente a lo que sucede con la media de X_t, en las ecuaciones (70) y (71) se puede observar que cuando la variable X_t no es estacionaria en la varianza, el proceso se vuelve divergente en la varianza. Por ello, es importante que la variable X_t es estacionaria en media y en varianza. El cálculo de la función de autocorrelación resulta de la manera (72).

$$\rho_i = \frac{\gamma_i}{\gamma_0} = \phi^i \tag{72}$$

Una vez comprobado el hecho de la estacionariedad de la variable X_t, el proceso a seguir es simple, en los procesos autorregresivos lo que se busca es obtener el valor de ϕ que hace que los cuadrados medios del error sea el más pequeño y como se puede observar en la ecuación (63), lo que se realiza es una regresión simple de la variable consigo misma. Esto implica que si la variable X_t es

estacionaria, entonces, se puede seguir un proceso de regresión lineal simple, siguiendo los supuestos correspondientes de ésta.

Por último, el modelo autorregresivo de orden p ($AR(p)$), se expresa por (73).

$$X_t = c + \phi_1 X_{t-1} + \phi_2 X_{t-2} + ... + \phi_p X_{t-p} + \varepsilon_t \tag{73}$$

En este caso el supuesto de estacionariedad cambia ligeramente, ya que, la media límite de la variable X_t esta dada por (74). Esto nos indica que la estacionariedad de la serie ahora depende de que $\sum_i^p < 1$, importante para el ajuste de los modelos autorregresivos de orden p.

$$\mu = \frac{c}{1 - \phi_1 - \phi_2 - ... - \phi_p} = \frac{c}{(1 - \sum_{i=1}^{p} \phi_i)} \tag{74}$$

Si continuamos observando los detalles de este modelo tenemos que las covarianzas están dadas por (75).

$$\gamma_i = \begin{cases} \phi_1 \gamma_{i-1} + \phi_2 \gamma_{i-2} + ... + \phi_p \gamma_{i-p}, \text{ Para i = 1,2,...} \\ \phi_1 \gamma_1 + \phi_2 \gamma_2 + ... + \phi_p \gamma_p + \sigma^2, \text{ Para i = 0} \end{cases} \tag{75}$$

Ahora, si dividimos las covarianzas entre γ_0, se obtiene la ecuación de Yule-Walker para la autocorrelación (76), lo cual nos indica la importancia de la estacionariedad de la serie, ya que, si ésta no se cumple, el valor de la autocorrelación podría resultar en valores no posibles para la medición de las autocorrelaciones.

$$\rho_i = \phi_1\rho_{i-1} + \phi_2\rho_{i-2} + ... + \phi_p\rho_{i-p} \tag{76}$$

para i = 1,2,...

4.2.3. Modelo de Medias Móviles: MA

El modelo de *Medias Móviles MA (Moving Average)* es uno de los indicadores técnicos ampliamente conocidos que se usa para predecir los datos futuros en el análisis de series de tiempo [99]. MA es un promedio común de los n datos anteriores en series de tiempo. Cada punto en los datos de series de tiempo tiene la misma ponderación.

Sea ε_t una secuencia de ruido blanco, entonces, el modelo de medias móviles de orden uno ($MA(1)$) parte de la ecuación (77), donde μ y θ son constantes. Se dice que la serie es un proceso de medias móviles de orden uno, si el valor de X_t se construye a partir de la adición de una suma ponderada al promedio general μ; esto es, se sabe que si el proceso es estacionario y por lo tanto el valor esperado al que converge X_t es μ, entonces, la predicción de X_t estaría dada por la adición de un término de error a este valor esperado. El orden del proceso esta condicionado al número de errores que se involucran en el proceso aparte del error del proceso en el estado t.

$$X_t = \mu + \varepsilon_t + \theta\varepsilon_{t-1} \tag{77}$$

El valor esperado del proceso de medias móviles de orden uno, bajo el supuesto de estacionariedad de la serie es derivado de(78).

$$
\begin{aligned}
E[X_t] &= E[\mu + \varepsilon_t + \theta\varepsilon_{t-1}] \\
&= E[\mu] + E[\varepsilon_t] + E[\theta\varepsilon_{t-1}] \\
&= \mu + 0 + \theta * 0 = \mu
\end{aligned} \tag{78}
$$

Con esto se comprueba una vez más que la estacionariedad de la serie es de suma importancia, ya que, si en este caso la serie fuera NO estacionaria $\mu \to \infty$.

Continuando el análisis teórico de este proceso, la varianza está dada por (79) y la autocovarianza por (80).

$$
\begin{aligned}
\gamma_0 = E[X_t - \mu]^2 &= E[\varepsilon_t + \theta\varepsilon_{t-1}]^2 \\
&= E[\varepsilon_t^2 + 2\theta\varepsilon_t\varepsilon_{t-1} + \theta\varepsilon_{t-1}^2] \\
&= E[\varepsilon_t^2] + 2\theta E[\varepsilon_t\varepsilon_{t-1}] + \theta^2 E[\varepsilon_{t-1}^2] \\
&= \sigma^2 + 0 + \theta^2\sigma 2 = (1 + \theta^2)\sigma^2
\end{aligned} \tag{79}
$$

$$
\begin{aligned}
\gamma_1 = E[(X_t - \mu)(X_{t-1} - \mu)] &= E[(\varepsilon_t + \theta\varepsilon_{t-1})(\varepsilon_{t-1} + \theta\varepsilon_{t-2})] \\
&= E[\varepsilon_t\varepsilon_{t-1} + \theta\varepsilon_{t-1}\varepsilon_{t-1} + \theta\varepsilon_t\varepsilon_{t-2} + \theta^2\varepsilon_{t-1}\varepsilon_{t-2}] \\
&= E[\varepsilon_t\varepsilon_{t-1}] + \theta E[\varepsilon_{t-1}^2] + \theta E[\varepsilon_t\varepsilon_{t-2}] + \theta^2 E[\varepsilon_{t-1}\varepsilon_{t-2}] \\
&= 0 + \theta\sigma^2 + 0 + 0 = \theta\sigma^2
\end{aligned} \tag{80}
$$

Además, es fácil observar que para autocovarianzas de orden mayor, éstas serán cero, haciendo que este sistema sea estacionario en covarianzas. Más aún, si por definición sabemos que las autocorrelaciones están dadas por (81), entonces, podemos saber que para el caso de un proceso de medias móviles de orden uno, las autocorrelaciones de orden mayor a 1 serán todas cero.

$$\rho_i = \frac{\gamma_i}{\gamma_0} \tag{81}$$

Finalmente, se puede ver que el valor de la autocorrelación de orden uno, está dado por (82).

$$\rho_1 = \frac{\theta}{1+\theta^2} \tag{82}$$

Similarmente a los procesos de medias móviles de orden uno, los modelos de orden q está dado por (83).

$$X_t = \mu + \theta_1\varepsilon_{t-1} + \theta_2\varepsilon_{t-2} + ... + \theta_q\varepsilon_{t-q} + \varepsilon_t \tag{83}$$

Analizando las propiedades de esta extensión del modelo de medias móviles de orden uno, se tiene que el valor esperado para éste modelo es (84) y la función de autocovarianzas está dada por (85).

$$E[X_t] = E[\mu] + E[\varepsilon_t] + E[\theta_1\varepsilon_{t-1}] + ... + E[\theta_q\varepsilon_{t-q}] = \mu \tag{84}$$

$$\gamma_i = \begin{cases} (1 + \theta_1^2 + \theta_2^2 + ... + \theta_q^2)\sigma^2, \text{ Para i = 0} \\ (\theta_i + \theta_{i+1}\theta_1 + ...)\sigma^2 \text{ Para i = 1, 2, ..., q} \\ 0, \text{ Para i >q} \end{cases} \tag{85}$$

4.2.4. Modelo Autoregresivo de Médias Móviles: ARMA

El proceso autorregresivo y medias móviles (por sus siglas en ingles: AutoRegressive Moving Average *ARMA (p, q)*) , es una combinación de un proceso autoregresivo de orden p y un proceso de medias móviles de orden q. Esta combinación se puede escribir como se muestra en la ecuación (86), donde ε_t son los errores que satisfacen una secuencia de ruido blanco, el valor esperado X_t está dado por (87) y las autocovarianzas para valores $i > q$ por (88).

$$\begin{aligned} X_t = c + \phi_1 X_{t-1} + \phi_2 X_{t-2} + ... + \phi_p X_{t-p} \\ +\varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + ... + \theta_q \varepsilon_{t-q} \end{aligned} \tag{86}$$

$$\mu = \frac{c}{(1 - \phi_1 - \phi_2 - ... - \phi_p)} \tag{87}$$

$$\gamma_i = \phi_1 \gamma_{i-1} + \phi_2 \gamma_{i-2} + ... + \phi_p \gamma_{i-p} \tag{88}$$

En los modelos combinados *ARMA (p, q)*, tenemos que seleccionar los valores de p y q que generen un mejor modelo. Para esto, se generan y prueban varios modelos y para seleccionar el

mejor ajuste al modelo se utiliza el criterio de información *Akaike (AIC)* (por sus siglas en inglés Akaike information criterion), el cual, es una medida de la calidad relativa de un modelo estadístico para un conjunto dado de datos. Sin embargo, aún cuando nos podemos ayudar de los criterios de AIC y los cuadrados medios del error, es importante saber cómo determinar los p y q iniciales con los que probaremos varios modelos y con esto contrastaremos los modelos ajustados y seleccionaremos el que tenga menores valores de AIC y σ^2.

Para realizar esta selección inicial es necesario conocer dos cosas:

1. La función de autocorrelación.
2. La función de autocorrelación parcial, la cual se deriva de la función de autocorrelación, solo que en este caso, las autocorrelaciones se ponderan tratando de eliminar la influencia de las relaciones que pudieran existir entre los errores (La teoría de esta función no se aborda y se deja la explicación como lectura adicional al lector).

4.2.5. Modelo Autoregresivo e Integrado de Médias Móviles: ARIMA

Para que se pueda generar un modelo se deben cumplir los siguientes supuestos:

1. La serie debe ser estacionaria en la media.
2. La serie debe ser estacionaria en la varianza.

El primer supuesto indica que si es estacionaria en la media, no se debe apreciar en la serie una tendencia positiva o negativa. Con respecto al segundo supuesto, debe evitarse que conforme pasa

el tiempo, la variación se vuelva infinita, es decir, como un cono que abre conforme pasa el tiempo. Cuando se tiene una serie de tiempo con tendencia y varianza infinita, éstas deben eliminarse.

Los modelos autorregresivos integrados y de medias móviles (por sus siglas en inglés: autoregressive integrated moving average), simbolizados como *ARIMA(p, d, q)*, siguen un proceso similar que los procesos *ARMA(p, q)*, solo que ahora dentro del modelo inicial se toma encuenta que la serie es no estacionaria y que es posible que se necesiten diferenciaciones de algún orden. Por ello la simbología p denota el número de parámetros autoregresivos del modelo, d denota el orden de las diferenciaciones que se requiere para resolver problemas de no estacionariedad y q denota el número de parámetros de medias móviles que se requieren.

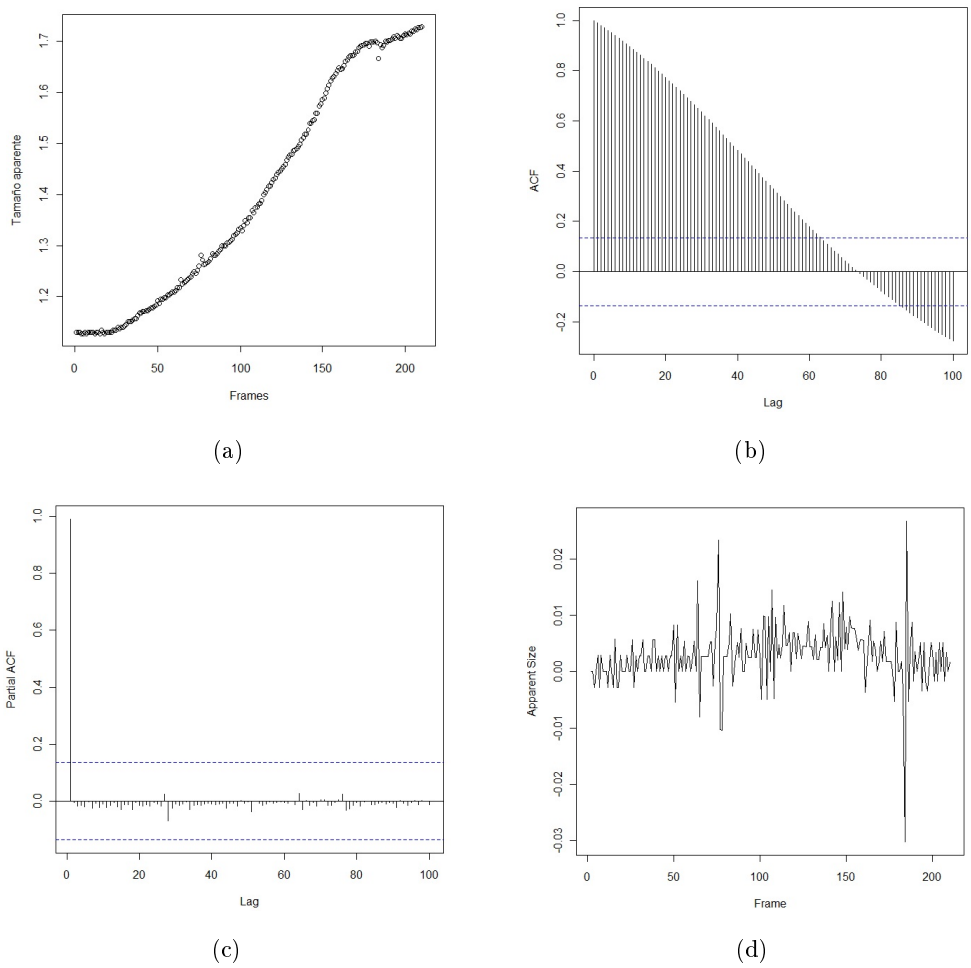


Figura 21: Ejemplo de obtención de parámetros en un proceso ARIMA. (a) Ejemplo de periodograma con tendencia. (b) Correlograma del periodograma. (c) Correlograma parcial del periodograma. (d) Serie diferenciada despues del proceso de diferenciación.

En la Figura 21 (a) se muestra un ejemplo se una serie (periodograma) que describe el cambio del tamaño aparente de un obstáculo sobre el tiempo mediante una segmentación por color, donde se aprecia que existe una tendencia, es decir, existe un crecimiento o decremento continuo en los valores de la series de tiempo durante un periodo de tiempo prolongado. Esto es lógico puesto que entre más cerca esté el robot del obstáculo, este último se verá más grande.

Una forma de gran utilidad en la interpretación de los coeficientes de autocorrelación descritos en la sección 4.2.1 es el uso del correlograma. El correlograma se construye graficando en el eje de las x los k retrasos o lags establecidos y en el eje de las y el coeficiente de autocorrelación para cada uno de los retrasos r_k. La Figura 21 (b) muestra el correlograma del periodograma, donde podemos ver que existe una correlación fuerte con datos anteriores. Por otro lado, se dice que si se aprecian varias autocorrelaciones significativas y consecutivas, es decir, las autocorrelaciones crecen o decrecen muy lento (como es el caso, pues decrecen muy lento), entonces la serie tiene una tendencia (como ya observamos en el periodograma y se analizó que es normal en nuestros experimentos). Este correlograma está relacionado con la parte AR de un modelo $ARMA$, donde la cantidad de autocorrelaciones sería el total de parámetros p para el modelo.

Por otra parte, la Figura 21 (c) muestra el correlograma parcial del periodograma, donde se observa que solo existe un valor significativo. Este grafico está relacionado con el componente MA del modelo $ARMA$, donde nos indicaría que el modelo solo requiere de un parámetro de medias móviles q.

Para eliminar la tendencia y/o periodicidad, pueden aplicarse algunos métodos de suavización como la *suavización exponencial de Holt* (cuando existen tendencia), la *suavización exponencial de Winter* (cuando hay periodicidad) o la *suavización de índices estacionales* (cuando hay ciclos). Sin embargo, otro enfoque es realizar *diferenciaciones* a la serie.

Este enfoque consiste en suponer que la tendencia evoluciona lentamente en el tiempo, de manera que en el instante t la tendencia debe estar próxima a la tendencia en el instante $t-1$. De esta forma, si restamos a cada valor de la serie el valor anterior, la serie resultante estará aproximadamente libre de tendencia. Por lo tanto consiste en pasar de la serie original x_t a la serie y_t mediante (89).

$$y_t = x_t - x_{t-1} \tag{89}$$

De este modo, la serie diferenciada resulta ser estacionaria. De este modo el modelo *ARIMA (p, d, q)*, sigue un proceso similar al de *ARMA (p, q)*, solo que ahora en el modelo inicial se toma en cuenta que la serie no es estacionaria y es posible que se necesiten diferenciaciones de algún orden.

Como conclusión, p denota el número de parámetros autorregresivos del modelo, d denota el orden de las diferenciaciones requeridas para resolver problemas de no estacionariedad y q denota el número de parámetros requeridos promedios móviles. Este tipo de modelo es crucial, ya que nos ayuda a vincular la *tendencia* y la *periodicidad* en la serie temporal al modelo, si se presentan estos elementos. En la Figura 21 (d), se aprecia visualmente que despues de una diferenciación ($d = 1$) que ya no hay tendencia (es estacionaria con respecto a la media) y no aunque hay variabilidad, esta no se hace infinita.

4.2.6. Medición del error del pronóstico

El error de pronóstico es una medida de suma importancia que nos indica qué tan preciso es un modelo. Para ilustrar esta idea, supongamos que X_t es la observación de la variable X al tiempo t y si definimos $\hat{X}_t$ como la estimación de la variable X al tiempo t mediante algún modelo de pronóstico. Entonces, el error de predicción o pronóstico está dado por (90).

$$e_t = X_t - \hat{X}_t \tag{90}$$

La medida que más comunmente se utiliza para determinar el error de pronóstico, es el error cuadrático medio (MSE por sus siglas en inglés), el cual se define como (91).

$$MSE = \sum_{t=1}^{N} \frac{(X_t - \hat{X}_t)^2}{N} \tag{91}$$

Algunas otras medidas usadas para la estimación del error de pronóstico sugeridas en la literatura son: la Desviación Absoluta Media (92), Porcentaje del Error Medio Absoluto (93) y Porcentaje Medio de error (94).

$$DAM = \sum_{t=1}^{N} \frac{|X_t - \hat{X}_t|}{N} \tag{92}$$

$$PEMA = \sum_{t=1}^{N} \frac{|X_t - \hat{X}_t|/X_t}{N} \tag{93}$$

$$PME = \sum_{t=1}^{N} \frac{(X_t - \hat{X}_t)/X_t}{N} \tag{94}$$

Sin embargo, para evaluar los modelos se propondrá otra medida de error la mostrada en (105) en la siguiente sección, con el fin de penalizar errores más grandes que pequeños.

4.3. Identificación de sistemas

La identificación de sistemas "tiene como objetivo construir modelos matemáticos de sistemas dinámicos a partir de datos medidos" [100] (ver, por ejemplo, [100][101][102][103]). Por lo tanto, podemos definir la identificación de sistemas, como los estudios de técnicas que persiguen la obtención de modelos matemáticos de sistemas dinámicos a partir de mediciones realizadas en el proceso: entradas o variables de control, salidas o variables controladas y perturbaciones.

El enfoque de la identificación se puede realizar en función de la estructura del modelo, y del comportamiento físico o no del mismo. Entonces se pueden distinguir tres:

- **Caja negra (Black-box):** los parámetros del modelo no tienen una interpretación física. En la realidad un modelo basado en leyes fundamentales es muy complicado o se desconoce.
- **Caja gris (Gray-box):** algunas partes del sistema son modeladas basándose en principios fundamentales, y otras como una caja negra. Algunos de los parámetros del modelo pueden tener una interpretación física; a este tipo de modelos también se les conoce como "Tailor-made", estimando sólo los parámetros no conocidos.
- **Caja blanca (White-box):** la estructura del modelo se obtiene a partir de leyes fundamentales. Los parámetros tienen una interpretación física.

4.3.1. Tipos de modelos

Existen varias formas de catalogar los modelos matemáticos [109] deterministas o estocásticos, dinámicos o estáticos, de parámetros distribuidos o concentrados, lineales o no lineales, y de tiempo continuo o tiempo discreto; i.e.:

1. **Modelos mentales.** Sin formalismo matemático.

2. **Modelos no paramétricos.** Se caracterizan mediante gráficos, diagramas o representaciones que describen las propiedades dinámicas mediante un número no finito de parámetros, i.e., respuesta al impulso, al escalón, o en frecuencia.

3. **Modelos paramétricos o matemáticos.** Describen las relaciones entre las variables del sistema mediante expresiones matemáticas; i.e., ecuaciones diferenciales en sistemas continuos y ecuaciones de diferencias en sistemas discretos.

En función del tipo de sistema y de la representación matemática utilizada, los sistemas pueden clasificarse en:

- **Determinísticos o estocásticos:** se dice que un modelo es determinístico, cuando expresa la relación entre entradas y salidas mediante una ecuación exacta; se estudia la relación entre la entrada y la salida con una parte no conocida. Por contra, un modelo es estocástico si posee un cierto grado de incertidumbre. Quedan definidos mediante conceptos probabilísticos o estadísticos.

- **Dinámicos o estáticos:** un sistema es estático cuando la salida depende únicamente de la entrada en ese instante de tiempo. La función que relaciona las entradas con las salidas, es independiente del tiempo. En un sistema dinámico las salidas varían con el tiempo, el valor actual de la salida en función del tiempo transcurrido desde la aplicación de la entrada. Tienen como objetivo conocer el comportamiento dinámico de un proceso.

- **Continuos o discretos:** los sistemas continuos trabajan con señales continuas, se formalizan mediante ecuaciones diferenciales. Los sistemas discretos trabajan con señales muestreadas, se describen por medio de ecuaciones en diferencias.

- **De parámetros concentrados:** no se considera la variación en función del espacio.
- **Lineales o no lineales:** un sistema lineal se define por una función matemática lineal; i.e., $y'(t) + ay(t) = u(t)$; siendo $y'(t) + ay^2(t) + by^3(t) = u(t)$ un sistema no lineal.

4.3.2. Proceso de identificación de sistemas

El proceso para la identificación de sistemas puede resumirse en los siguientes pasos:

1. **Planificación de experimentos.** Cuando se planean los experimentos hay varios aspectos que se deben tomar en cuenta, puesto que el proceso de experimentar puede ser costoso. Principalmente se debe tomar en cuenta que los experimentos deben poner de manifiesto toda la dinámica del sistema.

   Otro aspecto importante a tener en cuenta son las señales de entrada, es decir, la dificultad de la manipulación, la amplitud, el rango de frecuencias o si tiene exitación permanente. Aunado a esto, se debe establecer el periodo de muestreo, el tiempo de identificación (cantidad de datos necesarios) y si se usarán técnicas on-line u off-line.

2. **Realización de experimentos y reistro de datos.** Una vez realizados los experimentos, es posible analizar si los datos que se obtuvieron fueron los adecuados. En este procesamiento, se pueden realizar algunas tareas como la eliminación de componente continua, algún filtrado de perturbaciones de alta frecuencia, eliminación de perturbaciones de baja frecuencia, eliminación de datos erróneos, escalado de variables y/o ddentificación de los retardos.

3. **Selección del tipo de modelos.** Una vez recogido los datos, es necesario seleccionar el o los modelos mediante los cuales se aproximará el sistema. Por ejemplo, si es de caja blanca, negra

o gris, o también depende de la parametrización (paramétricos o no paramétricos). También se debe definir el tipo de modelos que se utilizarán (dominio temporal o frecuencial, determinista o estocástico, dinámico o estático, lineal o no lineal, continuo o discreto). Sin embargo, se debe recordar que los modelos son representaciones simplificadas de la realidad, el modelo debe ser apropiado para el fin que se persigue y que la complejidad debe ser la suficiente para recoger los aspectos esenciales del sistema.

4. **Elección de un criterio de bondad del modelo.** En esta fase se expresa lo bueno que es el modelo obtenido, es decir, se buscan el conjunto de parámetros que mejor se ajusta a los datos. Comúnmente se busca minimizar algún criterio que difiera entre los datos predichos y los experimentales. Algunos métodos de estimación se hacen fuera de línea (algoritmo no recursivo) y otros en línea (algoritmo recursivo).

   La diferencia de estos enfoques radica en que para la identificación fuera de línea, primero se obtienen todos los datos y después se ajusta el modelo sobre todos los datos, suelen tener buena convergencia y en general son fiables. Sin embargo, los resultados pueden ser pobres si la dinámica cambia a lo largo del experimento. Por otro lado, en la identificación en línea la estimación del modelo se hace en tiempo real, conforme se van tomando los datos. Sin embargo es un proceso más complejo y se necesita de supervisión. Aunque es apropiado si la dinámica del proceso cambia con el tiempo.

5. **Estimación de parámetros del modelo.** Para esta etapa simplemente se describen algunas técnicas aunque no se describirán a detalle. Las técnicas de estimación depende del conjunto de parámetros (paramétrica y no paramétrica).

   - *Técnicas de identificación no paramétrica.* Se realiza un análisis de respuesta transitoria: impulso, escalón, entre otras. Entre las técnicas frecuenciales podemos encontrar Fourier o

análisis espectral. Son buenas técnicas para procesos con estructura no conocida, aunque su análisis suele ser más complicado.

- *Técnicas de identificación paramétrica.* Se realiza la estimación de parámetros (continuos o discretos) y la predicción del error. Los algoritmos más comunes para predecir el error se encuentra el de mínimos cuadrados y estimador de máxima verosimilitud. Estas técnicas requieren que se conozca la estructura del modelo. Suelen dar mejores resultados si existe una relación lineal entre el error y los parámetros.

6. **Validación del modelo.** En esta última etapa se validan los modelos según reproduzca el modelo datos no usados, por lo que se suelen tener datos de estimación y datos de validación.

4.3.3. Descripción de la propuesta

En trabajos recientes, la identificación de sistemas se ha utilizado para mejorar áreas en la robótica, como el diseño de controladores integrados para estabilizar el diseño personalizado de auto-equilibrio de dos ruedas de robots [104], aplicación de modelos para el control de navegación autónoma en robots agrícolas [105], la simulación de sistemas de posicionamiento interior (IPS) en robots industriales como montacargas [106], entre otras.

En este trabajo, se utiliza la identificación de sistemas para modelar el cambio del tamaño aparente de los obstáculos, robusteciendo las estimaciones del TTC. Para realizar la identificación del sistema, suponemos que es un sistema lineal estático variable en el tiempo, porque a medida que disminuye la distancia entre el observador y el objeto, el crecimiento del tamaño aparente aumenta más rápido, incluso si la velocidad es constante. Esto es lo mismo con los objetos percibidos por la vista humana. Cuando miramos un objeto que está muy lejos y nos acercamos a él, el tamaño del

objeto percibido cambia muy poco. Sin embargo, si el objeto está muy cerca de nosotros, su tamaño que percibimos crece más rápido cada vez que nos acercamos a él. Este modelo será continuo, con una identificación paramétrica.

Para obtener información sobre la medida del tamaño aparente del obstáculo con respecto a la posición del robot, identificamos un modelo apropiado para el ángulo de apertura θ. Este ángulo se determina a partir del proceso de segmentación utilizando datos medidos.

4.3.4. Modelo cinemático

Consideremos primero la posición $s(t)$ del robot en el instante de tiempo t moviéndose con aceleración constante a, velocidad inicial v_0 y posición inicial s_0. De la ley cinemática, obtenemos (95).

$$s(t) = \frac{1}{2}at^2 + v_0 t + s_0 \tag{95}$$

A partir de la medición de la posición del robot $s(t)$ y el tiempo t, sería fácil estimar los tres parámetros reescribiendo (95) como un modelo de regresión lineal (96)

$$s(t) = \phi^T(t) \cdot \nu + e(t) \tag{96}$$

donde $\phi(t)$ y ν están dados por (97) y (98) respectivamente.

$$\phi(t) = \begin{bmatrix} \frac{1}{2}t^2 & t & 1 \end{bmatrix} \tag{97}$$

$$\nu = [a \quad v_0 \quad s_0]^T \tag{98}$$

Este modelo de la posición del robot no es lineal en t; sin embargo, es lineal en los parámetros desconocidos. Estos parámetros se pueden estimar mediante la versión off-line o recursiva del algoritmo de mínimos cuadrados. "Observese que el modelo cinemático, aunque depende explícitamente del tiempo t, conduce a una relación lineal estática" [101].

Una relación lineal estática aproximada entre el tiempo t y θ puede entonces asumirse como (99)

$$\theta(t) = \alpha_2 t^2 + \alpha_1 t + \alpha_0 \tag{99}$$

Este modelo puede ser reescrito como un Modelo de Regresión Lineal (100), donde $e(t)$ es un término de error, y $\varphi(t)$ y α están dadas por (101) y (102) respectivamente.

$$\theta(t) = \varphi^T(t)\alpha + e(t) \tag{100}$$

$$\varphi(t) = \begin{bmatrix} t^2 & t & 1 \end{bmatrix} \tag{101}$$

$$\alpha = [\alpha_2 \quad \alpha_1 \quad \alpha_0]^T \tag{102}$$

Desde N mediciones del ángulo de apertura $\theta(t_k)$ y tiempo t_k para $k = 1, \ldots, N$, los parámetros podrían estimarse mediante mínimos cuadrados simples (off-line).

$$\hat{\alpha} = \left[\sum_{k=1}^{N} \varphi(t_k)\varphi^T(t_k)\right]^{-1} \left[\sum_{k=1}^{N} \varphi(t_k)\theta(t_k)\right]. \tag{103}$$

Ahora, si los parámetros varían lentamente, se puede utilizar la siguiente versión recursiva (on-line) (modelo ARX recursivo) con exponencial, ignorando la estimación de mínimos cuadrados (104)[102][107][108], donde $\lambda(t_k)$ es una posible variable, ignorando el factor el cual hace posible seguir los parámetros que varían en el tiempo.

$$\begin{cases} \hat{\alpha}(t_k) = & \hat{\alpha}(t_{k-1}) + L(t_k)\left[\theta(t_k) - \varphi^T(t_k)\hat{\alpha}(t_{k-1})\right], \\ L(t_k) = & \dfrac{P(t_{k-1})\varphi(t_k)}{\lambda(t_k) + \varphi^T(t_k)P(t_{k-1})\varphi(t_k)}, \\ P(t_k) = & \dfrac{1}{\lambda(t_k)}\left[P(t_{k-1}) - \dfrac{P(t_{k-1})\varphi(t_k)\varphi^T(t_k)P(t_{k-1})}{\lambda(t_k) + \varphi^T(t_k)P(t_{k-1})\varphi(t_k)}\right]. \end{cases} \tag{104}$$

4.3.5. Medición del error del modelo

Se han propuesto diferentes criterios para cuantificar la cercanía de los valores predichos y las medidas. Una opción popular es la medida de ajuste del modelo (FIT) definida como (105) [109].

$$\text{FIT} = 100\left(1 - \frac{\left\|\theta(t_k) - \hat{\theta}(t_k)\right\|_2}{\left\|\theta(t_k) - E\{\theta(t_k)\}\right\|_2}\right) \tag{105}$$

Este criterio se elige porque contiene una suma del término de error (diferencia entre la variable simulada y la variable observada en cada paso de tiempo) normalizado por una medida de la variabilidad en las observaciones. Este criterio enfatiza errores más grandes, mientras que los errores más pequeños pueden ser ignorados. Sin embargo, queremos penalizar errores más grandes. En consecuencia, usamos (105), de modo que el rango de FIT se encuentre entre $-\infty$ y 100 %. Un valor de FIT inferior a cero indica que el valor medio de la serie temporal observada tiene un predictor más preciso que el modelo; mientras que un valor de 100 % indica un ajuste perfecto.

En este capítulo se presentaron tres propuestas para modelar el tamaño aparente de los obstáculos segmentados: Filtro de Kalman, un modelo ARIMA mediante enfoque se series de tiempo y un modelo ARX recursivo y no recursivo. El enfoque ARX no recursivo extrajo el comportamiento sin tantas alteraciones, debido a los otros dos enfoques se ajustaron demasiado a la señal, involucrando datos atípicos cuando estos existan.

5. Capítulo V: Módulo de estimación del TTC

5.1. Cálculo del TTC

Una vez que se han obtenido los parámetros del modelo y se ha predicho el tamaño aparente θ para los frames posteriores (por cualquiera de los tres métodos presentados), entonces es posible estimar el TTC.

Aunque la ecuación para obtener el TTC es muy sencilla, decidimos describir el origen de ésta y cómo empata con la obtención del tamaño aparente θ que nosotros utilzamos. Para ello, es necesario ejemplificar algunos conceptos de geometría. Primero, un **arco**, es un segmento de la longitud de la circunferencia. Es necesario recordar que la longitud de la circunferencia es 2
pi veces su radio r, es decir, el ángúlo perígono (que mide 360°) es $2\pi r$. Un **radián** es la medida de un ángulo central que subtiende un arco de longitud igual al radio de la circunferencia[110]. Por lo tnato, la proporcionalidad que existe entre la longitud del arco y el ángulo central asociado nos permite establecer una equivalencia para ir de una unidad angular (grados, minutos y segundos) a otra, que se denomina unidad circular por su relación con la longitud de la circunferencia (radianes).

Como se muestra en la Figura 22, el arco $S = \overset{\frown}{\mathrm{PQ}}$ es la parte de la longitud de la circunferencia que corresponde al ángulo central $\theta = \measuredangle POQ$. Por otro lado, tenemos que la longitud de una circunferencia L está dada por (106).

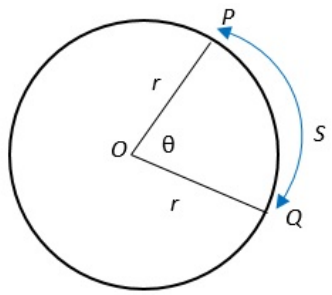


Figura 22: Segmento de una circunferencia.

$$L = 2\pi r \tag{106}$$

donde r es el radio de la circunferencia. Esta longitud corresponde al diámetro, cuyo ángulo sería $360°$. Si quisiéramos saber cuanto vale el segmento S de la circunferencia, dado un ángulo θ, por regla de tres obtendríamos (107)

$$S = \frac{2\pi r(\theta)}{360} \tag{107}$$

donde θ es un ángulo dado. Cuando se trabaja con radianes en vez de ángulos, existe una propiedad entre el concepto de **ángulo central** y segmento de una circunferencia. Un ángulo central es un ángulo cuyo vértice está en el centro de un círculo. En la Figura 22, el centro es el punto O, la longitud del radio es r, y $\theta = \measuredangle POQ$ es el ángulo central. La medida en radianes del ángulo central, se define como la razón de la longitud del arco y la longitud del radio. Entonces la medida en radianes de θ está dada por (108):

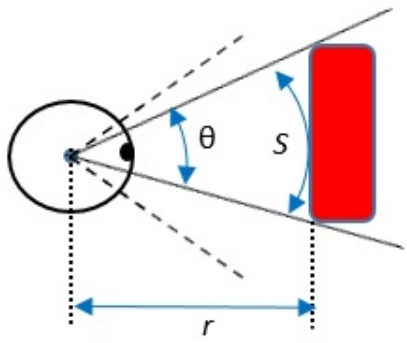


Figura 23: Percepción visual de un objeto.

$$\theta = \frac{S}{r} \tag{108}$$

donde la longitud del arco S, y el radio r, deben tener las mismas unidades.

Retomando la propuesta de Kaneta [75], en la cual afirma que los animales obtienen el tau-margin del tamaño aparente de los objetos y los cambios temporales en el tamaño, la Figura 23 muestra el mecanismo de la percepción visual de un objeto, donde el arco S es tangente con el objeto.

En la Figura 23, S indica el arco subtendido del ángulo visual y es tangente con el objeto que se está enfocando (y puede verse como la altura del objeto proyectada en la imagen como se vió en el capítulo 2), y r denota la distancia entre el objeto y el punto de observación. Tomando el tamaño aparente del objeto como el ángulo central de una circunferencia, y apartir de la ecuación (108), el cambio del tamaño aparente del objeto θ ($\frac{d\theta}{dt}$) está expresado mediante la fórmula de la derivada de una división en (109).

$$\frac{d\theta}{dt} = \frac{r\frac{dS}{dt} - S\frac{dr}{dt}}{r^2} = \frac{r\dot{S} - S\dot{r}}{r^2} \tag{109}$$

Al ser S (tamaño del objeto) un valor constante, su derivada se hace cero, por lo tanto el término $(r\dot{S})$ se hace cero y el cambio del tamaño aparente está dado por (110).

$$\frac{d\theta}{dt} = -\frac{S}{r^2}\frac{dr}{dt} \tag{110}$$

Aquí, τ indica el *tau - margin*, y está dado por la ecuación (111) y esto es igual a calcular el cambio de la distancia r por unidad de tiempo. Esta ecuación muestra que τ puede ser determinado directamente a partir de la θ y su cambio temporal, sin requerir información sobre la distancia o la velocidad relativa del objeto.

$$\tau = -\frac{\theta}{\frac{d\theta}{dt}} \approx -\frac{r}{\frac{dr}{dt}} \tag{111}$$

Por lo tanto, simplemente usamos (112) basado en la ecuación clásica descrita de (23). A partir de esta ecuación, estimamos τ, que describe la tasa de cambio del tamaño aparente del objeto a lo largo del tiempo. Con este proceso, los costos para encontrar el obstáculo en los siguientes frames se reducen. Además, las estimaciones robustas sin muchos valores atípicos se obtienen a través de los resultados modelados.

$$\tau = -\frac{\theta}{\frac{d\theta}{dt}} \tag{112}$$

Para clarificar el comportamiento de lo que se está obteniendo como medida al calcular τ, necesitamos ver el cambio en el tamaño aparente como se muestra en (23). Partimos de que la derivada de una función cualquiera $f(x)$ en el punto x_0 está dada y denotada por (113).

$$f'(x_0) = \lim_{\Delta \to 0} \frac{f(x_0 + \Delta) - f(x_0)}{\Delta} \tag{113}$$

Para valores pequeños de Δ, podemos aproximar la derivada de f en x_0 de la siguiente como se muestra en (114).

$$f'(x_0) \approx \frac{f(x_0 + \Delta) - f(x_0)}{\Delta}, \Delta \neq 0 \tag{114}$$

Reemplazando en esta ecuación los valores para calcular la variación de θ con respecto al tiempo que tenemos (115), y por lo tanto, si tomamos la variación como 1 frame, τ puede calcularse a partir de la ecuación (116).

$$\dot{\theta} = \frac{d\theta}{dt} \approx \frac{\theta_t - \theta_{t-1}}{\Delta = 1} \tag{115}$$

$$\tau_t = -\frac{\theta_t}{\dot{\theta}_t} \tag{116}$$

Las unidades de τ dependen del intervalo de tiempo con el que se tomó la imagen. Es decir, si se analiza cada cuadro, las unidades de τ serán los segundos en que fueron tomados cada frame, suponiendo que el intervalo es constante.

Para construir las ecuaciones anteriores, es necesario resaltar algunas consideraciones. En primer lugar, el tiempo de contacto no se define si no hay un cambio en el tamaño aparente del objeto. Esto se puede observar en (115) cuando el tamaño aparente en el tiempo t es igual al tiempo $t-1$, es decir, $\theta_t = \theta_{t-1}$ entonces $\dot{\theta} = 0$. Por lo tanto, en la ecuación (116), el denominador se vuelve cero, y el tiempo de contacto no puede ser estimado.

Visto de otra manera, la derivada de una función constante es cero. Si el tamaño aparente del objeto no cambia, entonces la derivada del ángulo subtendido θ y también la tasa de cambio del tamaño del objeto es cero. En este caso, el tiempo de contacto se volvería infinito, o mejor dicho, indefinido.

La interpretación es que si el objeto permanece sin cambio de tamaño, ya que no se acerca o aleja, no podemos decir cuánto queda para la colisión, porque aparentemente se mantiene a una distancia constante. En este caso, proponemos que si no hay un cambio aparente, el número de cuadros para colisionar seguirá siendo el mismo que el anterior donde se puede calcular el valor de τ. Con esto, nos aseguramos de que el próximo período, el tiempo de contacto se puede estimar.

Cabe mencionar que en la ecuación (114) que es llevada a (115), al verla como una aproximación, se introduce un término de error. Para estimar el error asociado a esta aproximación, nos ayudamos del polinomio de Taylor de primer grado cuya estructura es (117). Esta serie proporciona un medio para predecir el valor de una función en un punto en términos de la función y sus derivadas en otro punto. El teorema de Taylor establece que cualquier función suave puede aproximarse mediante un polinomio.

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(\xi)}{2}(x - x_0)^2 \tag{117}$$

Si se toma $x = x_0 + \Delta$, entonces $\Delta = x - x_0$ y se reemmplaza en el polinomio de la ecuación (117), teniendo entonces (118).

$$f(x_0 + \Delta) = f(x_0) + f'(x_0)\Delta + \frac{f''(\xi)}{2}\Delta^2,\ x_0 < \xi < x_0 + \Delta \tag{118}$$

Despejando $f'(x_0)$, obtenemos (119).

$$f'(x_0) = \frac{f(x_0 + \Delta) - f(x_0)}{\Delta} - O\Delta \tag{119}$$

donde $O = \frac{f''(\xi)}{2}$, denominado el error de truncamiento. Este error es aquel que resulta al usar una aproximación en lugar de un procedimiento matemático exacto y es debido a la omisión de términos en una serie que tiene un número finito de términos. En general, se considera que el error de truncamiento disminuye agregando términos a la serie de Taylor. Nótese que si Δ es suficientemente pequeño, entonces el término de primer orden y otros términos de orden inferior causan un porcentaje muy alto de error. Entonces podemos hacer una buena aproximación de derivadas utilizando el primer término, siempre y cuando se usen incrementos pequeños.

5.2. Robustecimiento

Como se mencionó anteriormente, nuestro objetivo es estimar que el TTC tan robusto como sea posible. Nosotros esperamos que el comportamiento de las estimaciones sea lineal si el robot se mueve con velocidad constante. También podemos observar en los resultados que hay valores atípicos u outliers (datos no siguen el mismo comportamiento), debido a diversos factores mencionados como

el cambio en la intensidad de la luz del entorno o mediante el movimiento del robot móvil. Estos cambios causan que $tau - margin$ sea positivo cuando el tamaño aparente del objeto en el momento t sea mayor que en el tiempo $t-1$ y negativo cuando el tamaño aparente del objeto es menor en el tiempo t que en el momento $t-1$, y por lo tanto, el tiempo de contacto no es confiable.

5.2.1. Propuesta

Para corregir estos valores atípicos y tener menos procesamiento, propongo que el robot genere un modelo lineal para predecir el tiempo de contacto sin volver a calcular el tamaño aparente del obstáculo. También el robot tiene la capacidad de identificar cuándo una observación podría ser influenciada por un agente externo en el entorno, como cambiar la iluminación, por lo que esta observación debe descartarse. Entonces, debemos identificar qué observaciones afectan el modelo y comportarnos como valores atípicos.

A menudo, encontramos modelos en los que el comportamiento de una variable y puede explicarse a través de una variable x, que se puede representar como $y = f(x)$. Si consideramos que la relación f es lineal, entonces tenemos un modelo de regresión lineal simple dado por (120).

$$Y = \beta_0 + \beta_1 X + u \tag{120}$$

La expresión (120) muestra una relación lineal, donde tomamos cada valor de $tau - margin$ (τ) como una observación del modelo lineal. Además, el valor de τ se toma como la variable de respuesta y y el intervalo de tiempo (número de frame) como la variable de predicción x. Por lo tanto, nuestra

propuesta tiende a ser robusta porque no solo depende de datos anteriores, sino que también depende de los datos que se comportan de la misma manera a lo largo del tiempo.

5.2.2. Determinando datos atípicos

El robot debe determinar qué observaciones influyen en un comportamiento lineal, es decir, en el modelo lineal. Entonces, la influencia ayuda al robot a decidir qué cálculo τ puede ser un valor atípico. La influencia de una observación se puede pensar en términos de cuánto diferirían los puntajes pronosticados para otras observaciones si no se incluyera la observación en cuestión. La influencia de una observación es una función de dos factores:

1. Cuánto difiere el valor de la observación en la variable de predicción de la media de la variable de predicción, y
2. La diferencia entre el valor pronosticado para la observación y su valor real.

El primer factor se llama el **apalancamiento (leverage)** de la observación. El último factor se llama la **distancia de la observación**. Entonces, el robot descartará una observación si éste es un punto leverage y si tiene una distancia grande.

5.2.3. Apalancamiento o leverage

El apalancamiento en un modelo estadístico tiene como objetivo identificar aquellas observaciones que están muy lejos de los valores predictivos promedio correspondientes. El apalancamiento de una observación h_i se basa en cuánto difiere el valor de las observaciones en la variable de predicción

de la media de la variable de predicción. Una mayor influencia de una observación indica que la observación no influye en el modelo. Esto es, una observación con un valor igual a la media en la variable de predicción no tiene influencia en la pendiente de la línea de regresión. Por otro lado, una observación que es extrema en la variable predictor tiene el potencial de afectar la pendiente en gran medida [111]. El apalancamiento está estrechamente relacionado con la distancia de Mahalanobis (121) y se calcula como se muestra en (122).

$$d_M(\overrightarrow{x}, \overrightarrow{\mu}) = \sqrt{(\overrightarrow{x} - \overrightarrow{\mu})' \sum^{-1} (\overrightarrow{x} - \overrightarrow{\mu})} \tag{121}$$

$$h_i = \frac{1}{n} + \frac{1}{n-1} \left(d_M^2 \left(x_i, \overline{x} \right) \right) \tag{122}$$

Un valor h_i por debajo de 0.2, indica que el potencial es bajo, los valores entre 0.2 y 0.5 pueden ser potencialmente atípicos, mientras que los puntos cuyos valores son mayores que 0.5 deben analizarse.

5.2.4. Distancia de Cook

La distancia de Cook es una buena medida de la influencia de una observación, porque mide cómo cambia el vector de estimadores $\widehat{\boldsymbol{\beta}}$ cuando se elimina cada observación. Es proporcional a la suma de las diferencias cuadradas entre las predicciones realizadas con todas las observaciones en el análisis y las predicciones realizadas que omiten la observación en cuestión. Si las predicciones son las mismas con y sin la observación en cuestión, entonces la observación no tiene influencia en el modelo de regresión. Si las predicciones difieren mucho cuando la observación no está incluida en el

análisis, entonces la observación es influyente. Como la matriz de covarianza de $\boldsymbol{\beta}$ puede estimarse con S_R^2, dada por (123), donde $\boldsymbol{X}$ es el vector de las medidas del modelo.

La distancia de Cook del punto i-ésimo D_i, viene dada por (124), donde $\widehat{\boldsymbol{\beta}}$ es el vector de coeficientes, $\widehat{\boldsymbol{\beta}}(\boldsymbol{i})$ es el vector de coeficientes calculado después de eliminar la i-ésima observación, X es la matriz de diseño. Una regla empírica común es que una observación con un valor de distancia de Cook sobre 1.0 tiene demasiada influencia.

$$S_R^2 = (X'X)^{-1} \tag{123}$$

$$D_i = \frac{[\widehat{\boldsymbol{\beta}} - \widehat{\boldsymbol{\beta}}(\boldsymbol{i})]'X'X[\widehat{\boldsymbol{\beta}} - \widehat{\boldsymbol{\beta}}(\boldsymbol{i})]}{(k+1)S_R^2} \tag{124}$$

Por lo tanto, el robot descartará la observación con un apalancamiento mayor a 0.4 y una distancia de Cook mayor que 1.0 para que estos valores atípicos no influyan en la estimación del TTC.

En este capítulo se presentó la manera en que se calcula el TTC, así como la explicación del significado cuando no existe variación en el tamaño aparente. También se presentó una propuesta para robustecer las estimaciones mediante la eliminación de datos atípicos encontrados a través de la distancia de Cook. Sin embargo, experimentaciones muestran que no es viable debido que cada vez que se eliminan outliers, aparecen otros, y esto implica que en un proceso iterativo se podrían quedar con pocas estimaciones que no representen el modelo real.

6. Capítulo VI: Experimentos y discusión de resultados

En este capítulo se describen en detalle el diseño de experimentos y resaltarán algunos resultados importantes que nos llevarán a extraer conclusiones de este trabajo de investigación.

6.1. Diseño de experimentos

Finalmente, un punto clave para evaluar la precisión de un método son los experimentos que se realizaron. Hay dos tipos de experimentos utilizados en la revisión de la literatura: con imágenes sintéticas y con imágenes reales. Los experimentos con imágenes en ambientes reales son realizados en ambientes de interiores y al aire libre. Con respecto al tipo de obstáculos con los cuales se puede tener colisiones pueden ser obstáculos segmentados y paredes o muros con alguna textura. Por último, otra característica que diferencia a los experimentos es la velocidad con la cual un robot avanza o retrocede: velocidad constante, aceleración constante o aceleración variable. Los experimentos realizados en este trabajo se destacan en líneas gruesas en la Figura 24.

Figura 24: Clasificación de las configuraciones de los experimentos.

Para nuestros experimentos, decidimos hacerlo en un ambiente real, en el cual se escogió un ambiente interior con el fin de controlar la luz ambiental y tener mayor control de los parámetros para la segmentación por color. También se escogió calcular el TTC para obstáculos detectados debido a que considero mayor complejidad en la tarea que solamente obtener el TTC en paredes con textura. Por último se escogió la velocidad constante del robot para asegurarnos de conocer el Ground truth.

Existen dos grandes tareas involucradas en esta propuesta para el TTC: segmentación y modelado. La Figura 25 muestra la experimentación del problema de segmentación. Para abordar este

problema, se utilizaron los tres enfoques mencionados anteriormente: segmentación por color, por contornos probabilísticos y por aprendizaje de características. Para los dos primeros enfoques se probaron en escenarios donde el obstáculo es un cuerpo rígido con un color contrastante con el ambiente. En el caso del cilindro rojo es un color opaco, mientras que la esfera verde tiene un color brillante que refleja la luz. Para el enfoque de aprendizaje de características se realizaron en un ambiente con tres obstáculos: El primero con mucha textura (búho), el segundo con textura y puntos clave bien definidos (cada de cubo de rubik) y el tercero con textura pero es un cuerpo no rígido (águila). Por último para el enfoque de color fue probada la segmentación de manera secuencial y paralela.

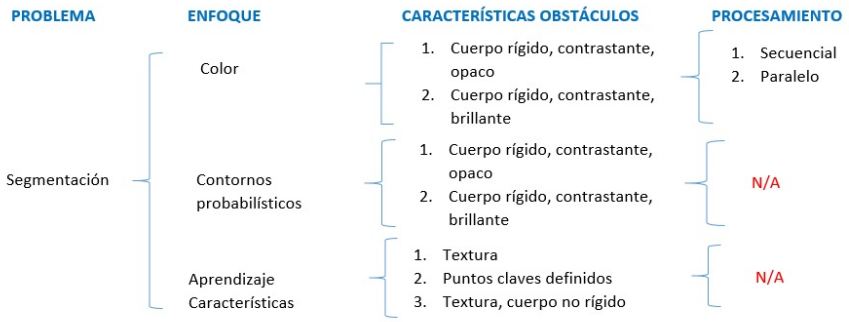


Figura 25: Características de experimentos para el problema de segmentación.

La Figura 26 muestra la experimentación del problema de modelado. Para abordar este problema, se utilizaron los tres enfoques mencionados anteriormente: Filtro de Kalman, un modelo ARIMA como modelo de serie temporal y un modelo ARX recursivo y no recursivo. Estos tres enfoques se probaron en cada uno de los 5 obstáculos descritos en el párrafo anterior. Sin embargo, para los dos primeros obstáculos, la segmentación probada fue mediante contornos probabilísticos porque demostró mayor precisión que por color, mientras que para los 3 restantes obstáculos se utilizó una segmentación por aprendizaje de características ya que estos obstáculos contienen textura que puede ser identificada en los cuadros.

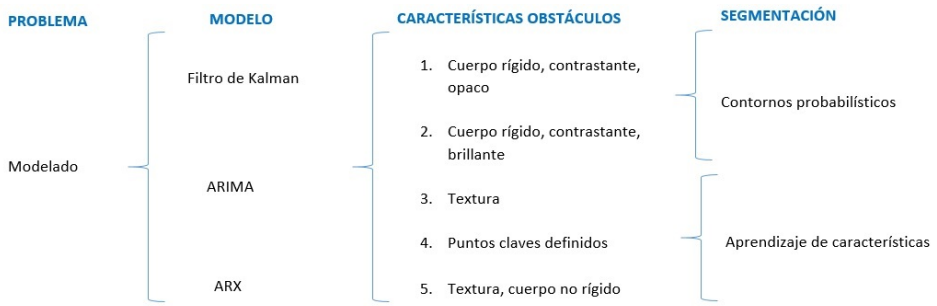


Figura 26: Características de experimentos para el problema de modelado.

Para todos nuestros experimentos, se utilizó el robot BRLX01 mostrado en la Figura 27. Este robot se creó a partir de una tarjeta raspBerry pi, y una cámara robotizada de 5 mpx y dos grados de libertad, 4 ejes de tracción con llantas antideslizantes. Para los experimentos descritos en este documento, los videos se tomaron con una frecuencia de 30 fotogramas por segundo. El tamaño de cada imagen era de 1920 x 1080 píxeles.

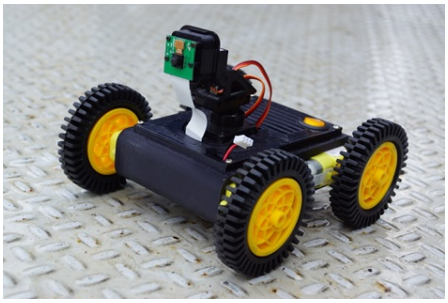


Figura 27: Robot BRLX01

Los primeros escenarios controlados constan de una configuración del robot aproximándose de frente a dos obstáculos centrados y contrastantes con el ambiente. El primero es un cilindro de color rojo y el otro es una esfera de color verde.

Para nuestros experimentos de aprendizaje de obstáculos, se tomaron para aprender tres obs-

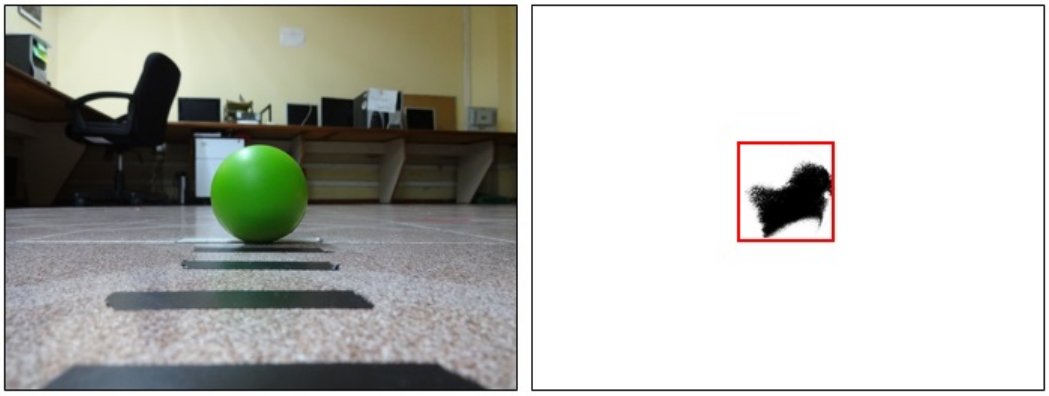
táculos con diferentes características, formas y tamaños. El primer obstáculo fue una caja de un cubo de rubik, el segundo es una estatua de un búho y el tercero es un juguete con forma de águila vestida. Para el diseño de nuestros experimentos, primero identificamos cada objeto por separado y, como último experimento, se le pidió al robot que identificara los tres obstáculos en la escena. En este diseño de experimento, el robot está de frente a uno de ellos, pero los otros dos están a cada extremo de la imagen, por lo tanto, aunque se acerque a uno de ellos, también se calcula para todos los de la escena.

En todos los experimentos se usó una cámara como sensor para ubicar los objetos en el entorno del robot. Para nuestros experimentos, se usó $a = 0$ (ver (95)), para controlar mejor y especificar el Ground truth.

6.2. Resultados del módulo de segmentación

Para la segmentación por color Los colores utilizados para segmentar los obstáculos fueron para rojo: [R = 210, G = 50 y B = 50] y para verde [R = 20 G = 220 y B = 20] con un umbral = 50 (obtenido experimentalmente). La Figura 28 muestra ejemplos de segmentación de obstáculos por color con los parámetros antes mencionados. En el caso de la esfera verde, podemos notar que el color no es uniforme en toda la esfera. En la parte de arriba se tiene un color blanco porque la luz es reflejada en la esfera. De igual manera la parte de abajo no lo toma en cuenta porque se trata de una sobra que es proyectada porque la luz está justamente arriba de la esfera.

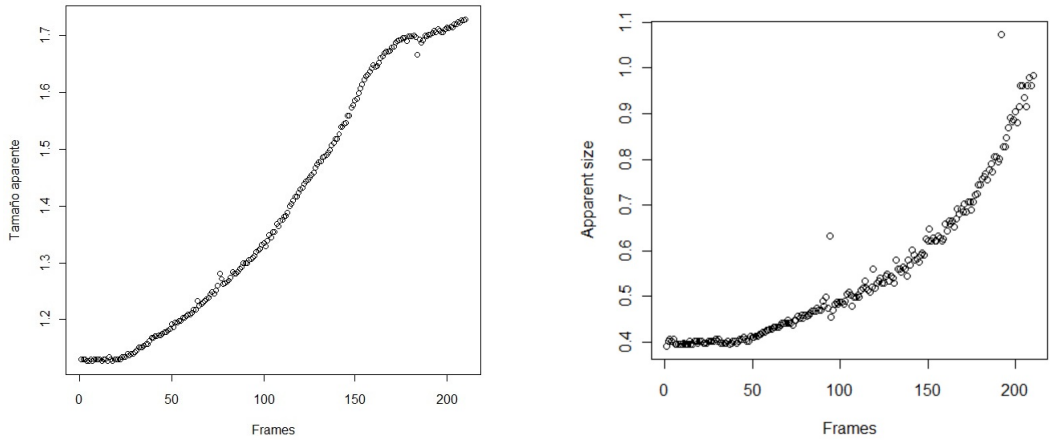
(a)



(b)

Figura 28: Ejemplo de segmentación por color. (a) Muestra la segmentación de un obstáculo por color rojo de en un ambiente real. (b) Muestra la segmentación de un obstáculo por color verde en un ambiente real.

La Figura 29 muestra cómo cambia el tamaño aparente de los obstáculos mencionados a través del tiempo. De estas gráficas podemos notar que existen outliers, y puede deberse a los factores antes mencionados (cambio en intensidad de luz o relieve del suelo).

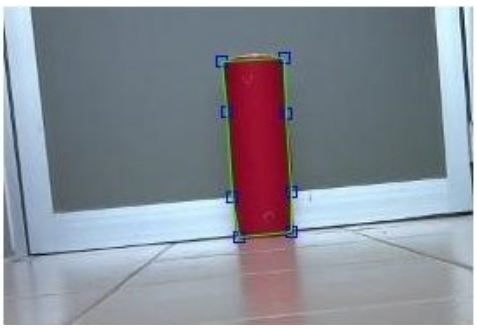


(a) (b)

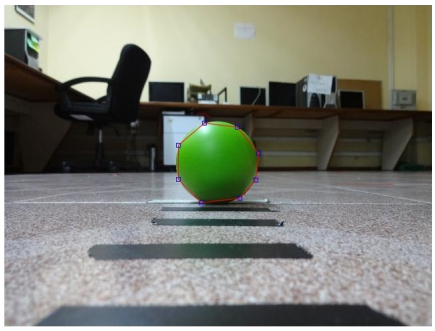
Figura 29: Tamaño aparente de los obstáculos segmentados por color en ambientes reales. (a)Caso I: muestra el cambio de tamaño aparente en el tiempo el obstáculo segmentado por color rojo. (b)Caso II: muestra el cambio de tamaño aparente en el tiempo el obstáculo segmentado por color verde.

Para evaluar nuestra propuesta de segmentación basada en contornos probabilísticos e inferencia MAP tomamos los mismos 210 fotogramas de un robot que se mueve hacia dos objetos (un cilindro y una esfera) a velocidad constante en escenarios reales, los cuales se reportaron en [76].

Las Figuras 30 (a) y 30 (b) muestran ejemplos de segmentación en imágenes reales de un cilindro y una esfera utilizando nuestro método de contorno probabilístico. Los contornos con visualizaciones spline de estos ejemplos de segmentación se presentan en las figuras 30 (c) y 30 (d).

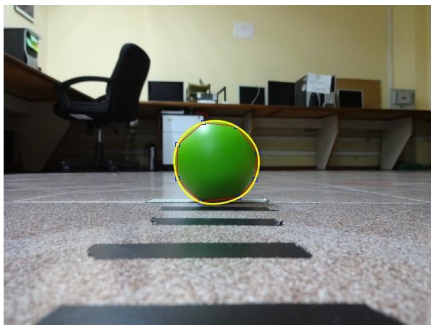


(a)



(b)

(c)

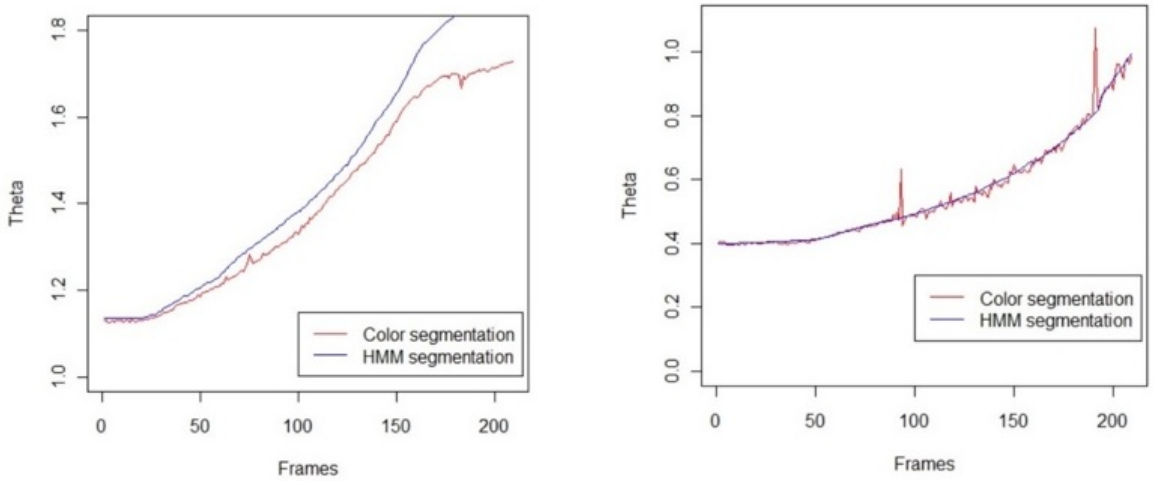


(d)

Figura 30: Ejemplo de segmentación por contornos probabilísticos. (a)Segmentación por contorno probabilístico de un cilindro rojo. (b)Segmentación por contorno probabilístico de una esfera verde. (c)Ejemplo de la segmentación del cilindro rojo con visualización spline. (d)Ejemplo de la segmentación de la esfera verde con visualización spline.

Ahora comparamos nuestra propuesta de segmentación por contornos probabilísticos y el enfoque de segmentación de color clásico. Los resultados de segmentación de estas imágenes que emplean ambos enfoques se presentan en las figuras 31 (a) y 31 (b). A partir de estas gráficas, se puede observar que la propuesta basada en HMM (líneas azules) obtuvo segmentaciones más suaves (es decir, los tamaños aparentes (θ) de las secuencias de imágenes) que el enfoque basado en el color (líneas rojas). Del mismo modo, los contornos probabilísticos obtuvieron menos valores atípicos que el enfoque basado en el color. Específicamente, se puede ver que el tamaño aparente del cilindro

rojo (Figura 31 (a)) aumenta significativamente en ambos enfoques en comparación con el tamaño aparente de la esfera verde (Figura 31 (b)) . Este rápido aumento está asociado con la secuencia de imágenes, que fue diseñada para saber cómo se comporta la estimación cuando hay una oclusión del obstáculo. En otras palabras, en esta secuencia, en un momento determinado, la cámara ya no ve la parte superior del cilindro y ahora se basa únicamente en la parte inferior.



(a) (b)

Figura 31: Ejemplo de segmentación de imágenes reales usando contornos probabilísticos y enfoque basado en color. (a)Caso I: Comparación de enfoques de segmentación para el obstáculo cilíndrico. (b)Caso II: Comparación de enfoques de segmentación para el obstáculo esférico.

Utilizamos la prueba no paramétrica de dos muestras Kolmogorov-Smirnov (prueba K-S) para determinar si el enfoque de segmentación basado en el contorno probabilístico y el enfoque de segmentación de color tienen las mismas distribuciones de respuesta (hipótesis nula) o no (hipótesis alternativa). Decidimos elegir la prueba K-S en lugar de la prueba de Wilcoxon-Mann-Whitney, porque la prueba K-S es "una prueba ómnibus con buena potencia contra alternativas generales en las que ambas poblaciones pueden diferir en forma y ubicación" [112]. Se puede ver en el Cuadro 3 que hubo una diferencia significativa entre los enfoques en la imagen del cilindro rojo (p = 7.87e-06).

Sin embargo, los datos de ambos métodos provienen estadísticamente de una misma distribución en la imagen de esfera verde (p = 0.1591), a pesar de que el método de segmentación de color presentó más oscilaciones que la segmentación por Cadenas ocultas de Markov (ver Figura 31 (b)).

Cuadro 4: Resultados de la prueba Kolmogorov-Smirnov

Descripción	P-valor	Resultado
Cilindro rojo	7.87e-06	Signiticativo
Esfera verde	0.1591	No significativo

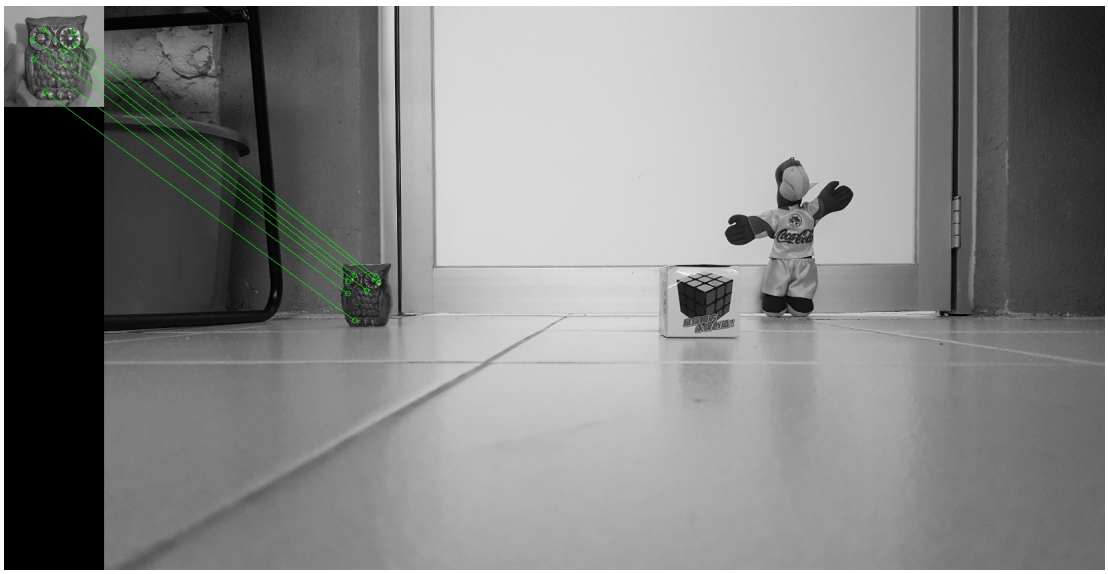
Para evaluar el aprendizaje de obstáculos, la Figura 32, muestra la extracción de características ofline, de los obstáculos que no están en el ambiente.

(a) (b) (c)

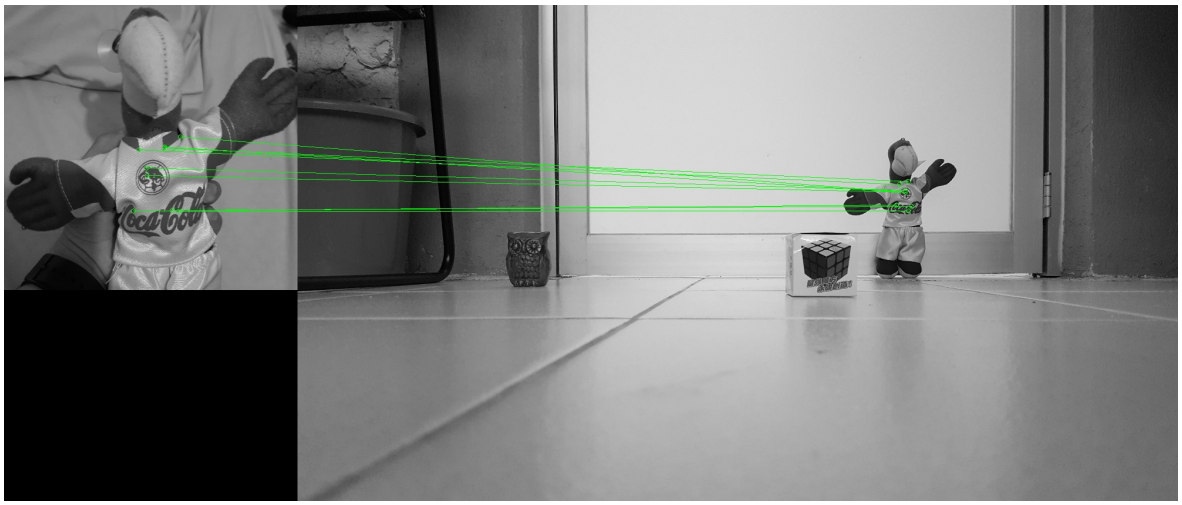
Figura 32: Aprendizaje de características de obstáculos. (a)Caso I: Caja de un cubo de rubik, con esquinas en la separación de cada cuadro. (b)Caso II: Pieza de cerámica de un búho, con mucha textura. (c)Caso II: Figura de aguila con textura en la vestimenta.

La Figura 33 muestra los resultados del proceso de coincidencia después del proceso de filtrado, donde se puede observar que para el segundo y el tercer obstáculo no tenemos valores atípicos. La figura 34 muestra los resultados de la altura de la región característica de cada obstáculo, es decir, el

área del obstáculo con mejores características para rastrear o seguir. Con estos resultados estamos indicando que podemos encontrar obstáculos aprendiendo características importantes de los objetos.



(a)



(b)

Figura 33: Ejemplo del proceso de emparejamiento o coincidencia (match) de la pieza del buho y pieza de águila vestida. (a) Búho encontrado en la imagen. (b) Águila encontrada en la imagen

(a)

(b)

Figura 34: Área de la escena segmentada en un recuadro a partir de la coincidencia de puntos. (a) Búho encontrado en la imagen. (b) Águila encontrada en la imagen

La Figura 35 muestra cómo cambia en el tiempo los objetos segmentados por aprendizaje de características. Se puede notar que existen variaciones debido a diversos factores, como la iluminación o el relieve del piso. Además, puede existir variación ya que en cada frame, no siempre se encuentran los mismos puntos (por oclusión o por no ser tan significativos).

También la Figura 35 muestra que aunque el objeto águila es más grande que el de cubo de rubik, en el primero solo se está segmentando una parte porque solo segmenta mediante las características

que se encontraron. También se muestra que la secuencia del búho es más corta, debido a que el robot se encuentra centrado en la escena (de frente al cubo de rubik), en algún momento el obstáculo del búho se sale del campo visual, (puesto que el robot pasa por un lado) y solo termina por ver los otros dos.

Este ejemplo es muy ilustrativo para indicar que se debe tener cuidado si se segmenta la altura o la anchura de objetos que no están enfocados directamente. Aunque pareciera que podría colisionar, realmente no se encuentra en la trayectoria del robot, sin embargo, este logra identificar que existen obstáculos en la escena.

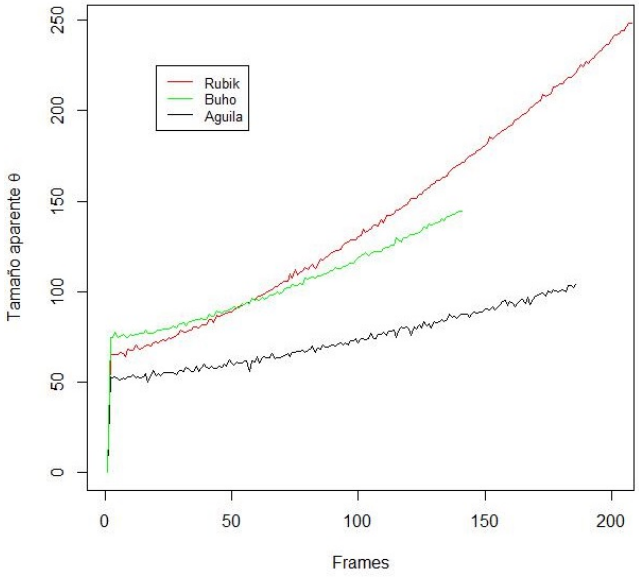
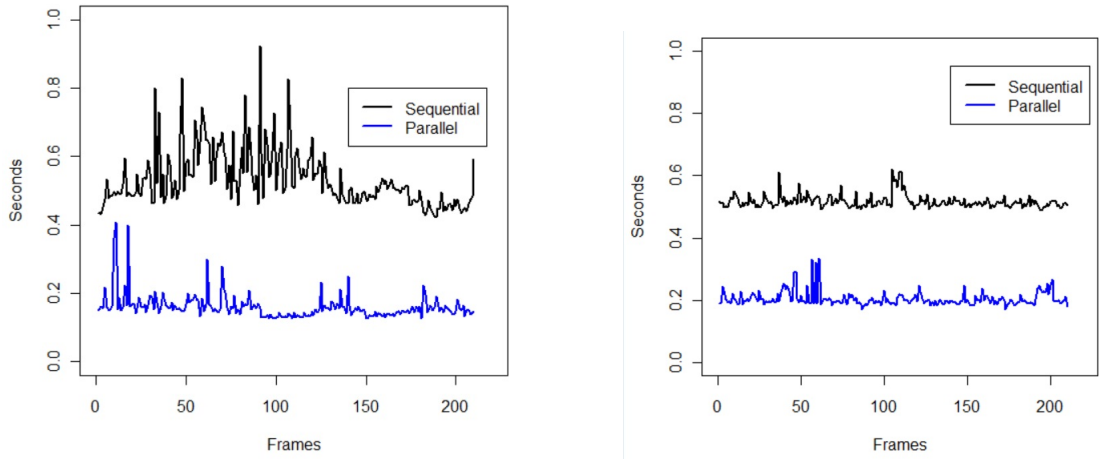


Figura 35: Cambio del tamaño aparente de los obstáculos segmentados por aprendizaje de características

6.3. Resultados del módulo del proceso paralelo de segmentación

Para los experimentos de esta sección, se tomaron las secuencias de imágenes para la segmentación por color y contornos probabilísticos. Los experimentos fueron controlados con el mínimo de procesos corriendo en una computadora con procesador Intel Core i7-3537U a 2.00 GHz, con 8 GB de memoria RAM. La tarjeta utilizada fue una GeForce FT 735 M con 65536 bloques con un máximo de 1024 hilos por bloque.

Los experimentos consistieron en analizar un conjunto de 210 frames (equivalente a 7 segundos) de manera secuencial y de manera paralela con las dos propuestas descritas en el apartado del procesamiento paraelo mediante la arquitectura CUDA y con los obstáculos definidos en la Figura 25. Los resultados muestran que no hay diferencia entre usar alguna de estas dos propuestas. En la Figura 36 se muestran los tiempos para cada cuadro en las dos diferentes maneras de procesar la segmentación (secuencial y paralela). Como se puede apreciar, ningún tiempo máximo para la segmentación paralela iguala al menor tiempo de la segmentación secuencial.



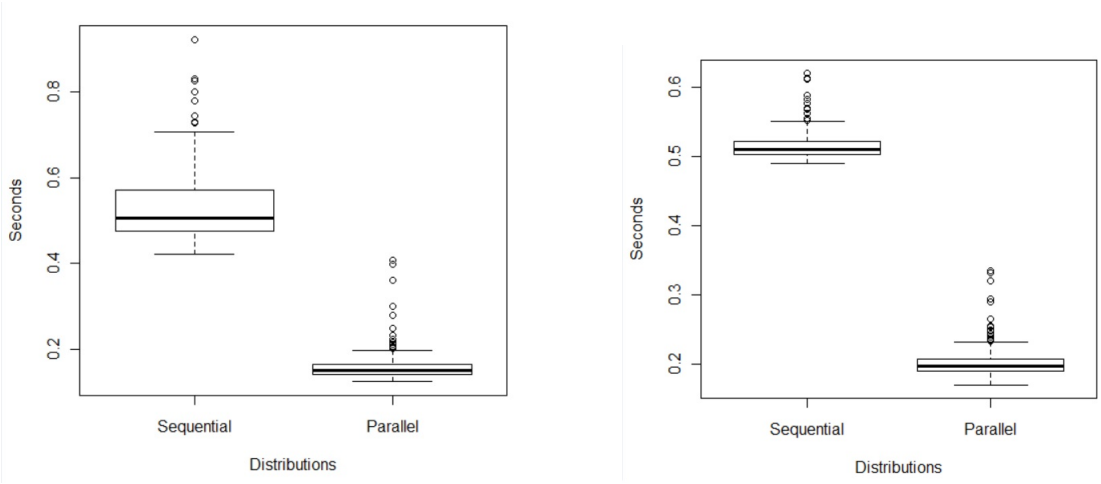
(a) (b)

Figura 36: Series de tiempos tomados para segmentar en cada Frame por manera de procesamiento. (a) Segmentación del cilindro rojo. (b) Segmentación de la esfera verde.

En el cuadro 5 se muestra el análisis de tiempo de manera secuencial y de manera paralela. Como se puede apreciar en el cuadro 5, en promedio existe una ganancia del 70 %. Sin embargo es necesario verificar si ese promedio es realmente la representatividad de los valores obtenidos. Para ello, la Figura 37 compara la variabilidad entre los tiempos obtenidos en cada una de las pruebas (secuencial y paralela por caa obstáculo segmentado por color). De la Figura 37, podemos notar en el tamaño de la caja de los gráficos, que la los tiempos del procesamiento secuencial tienen mayor variabilidad, por lo que creemos se debe a la cantidad de procesos que atiende el procesador de manera intermitente, mientras que en el procesamiento paralelo hay menor variabilidad pues el procesamiento depende de la GPU y no del CPU, aunque podemos notar que según el gráfico, hay datos atípicos.

Cuadro 5: Comparación de tiempos en segundos.

Obstáculo	Procesamiento	Min.	Mediana	Promedio	Max.	D. S.
Cilindro	Secuencial	0.4220	0.5060	0.5325	0.9220	0.0549
Cilindro	Parallela	0.1250	0.1515	0.1596	0.4080	0.0277
Esfera	Secuencial	0.4900	0.5100	0.5220	0.62	0.0214
Esfera	Parallela	0.1700	0.1980	0.2051	0.3250	0.0211



(a) (b)

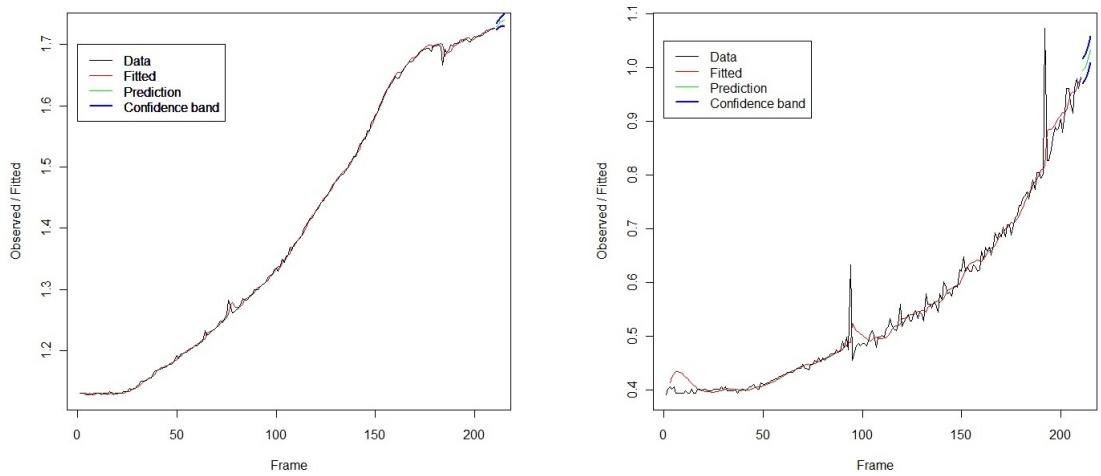
Figura 37: Variabilidad en tiempos de procesamiento. (a) Variabilidad de tiempo de procesamiento de segmentación del cilindro rojo. (a) Variabilidad de tiempo de procesamiento de segmentación de la esfera verde.

Para saber si estas dos distribuciones siguen un comportamiento similar, se utilizó la prueba no paramétrica de $Kolmogorov - Smirnov$ para dos muestras, la cual es una prueba donde la hipótesis nula es que las dos distribuciones son iguales contra la alterna que son diferentes. Dado que el $p - valor$ de la prueba fue menor a α (0.00002), a un nivel de significancia del 95 %, rechazamos la hipótesis nula y comprobamos que las dos distribuciones son diferentes.

6.4. Resultados del módulo de Modelado

Para poder comparar las propuestas de modelado, se utilizaron los datos recabados a partir los métodos de segmentación presentados anteriormente. Primero, la Figura 38 muestra lo resultados del modelado y predicción del tamaño aparente mediante un enfoque ARIMA de los escenarios del obstáculo del cilindro y la esfera. En dicha Figura se muestra en color negro el periodograma de los

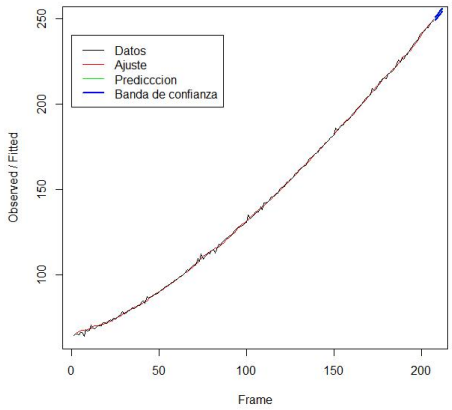
datos en el tiempo. El color rojo muestra el modelo ajustado. Debido a que la idea es recolectar un tamaño suficiente para poder pronosticar, se tomaron los primeros 200 frames para generar un modelo del cambio del tamaño aparente y a partir de ese instante poder pronosticar. Entonces el color verde muestra el pronóstico a 5 frames y el color azul muestran las bandas de confianza. Cabe mencionar que conforme pronosticábamos más datos, las bandas se hacían muy grandes, sin embargo el modelo se ajusta bien a los datos.



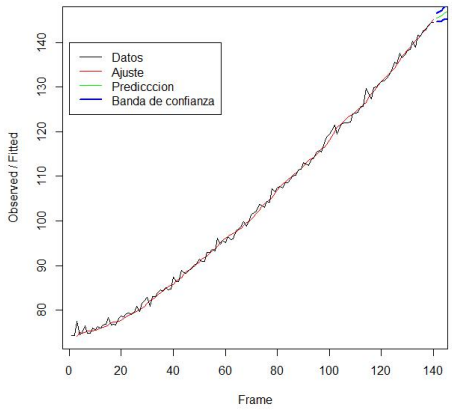
(a) (b)

Figura 38: Ajuste del modelo ARIMA y pronóstico. (a)Caso I: Pronóstico del tamaño aparente para la percepción del cilindro rojo. (b)Caso II: Pronóstico del tamaño aparente para la percepción de la esfera verde.

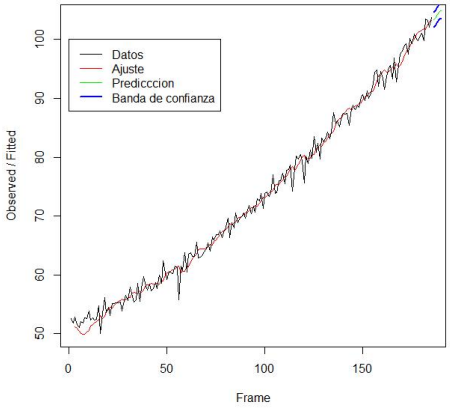
La Figura 39 muestra de igual manera las series temporales, los ajustes y los pronósticos para los 3 objetos que aparecen de manera simultánea. Se aprecia en el inciso (a) las bandas de confianza están prácticamente alineadas con el pronóstico y esto es porque de las tres series es la que tiene menor variabilidad. En los incisos (b) y (c) se pronostica con menor confiabilidad y se grafican un lag de 5 elementos pronosticados. Sin embargo los modelos se ajustan bien a los datos observados.



(a)



(b)

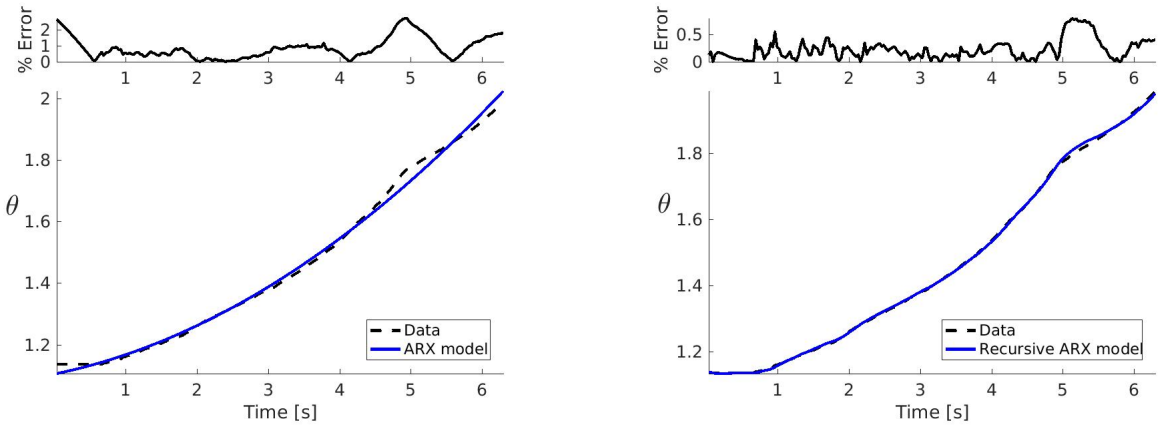


(c)

Figura 39: Ajuste del modelo ARIMA y pronóstico de los tres obstáculos simultáneos. (a)Pronóstico del tamaño aparente para la percepción de la caja del cubo de rubik. (b)Pronóstico del tamaño aparente para la percepción de la figura de búho. (c)Pronóstico del tamaño aparente para la percepción de la figura de águila.

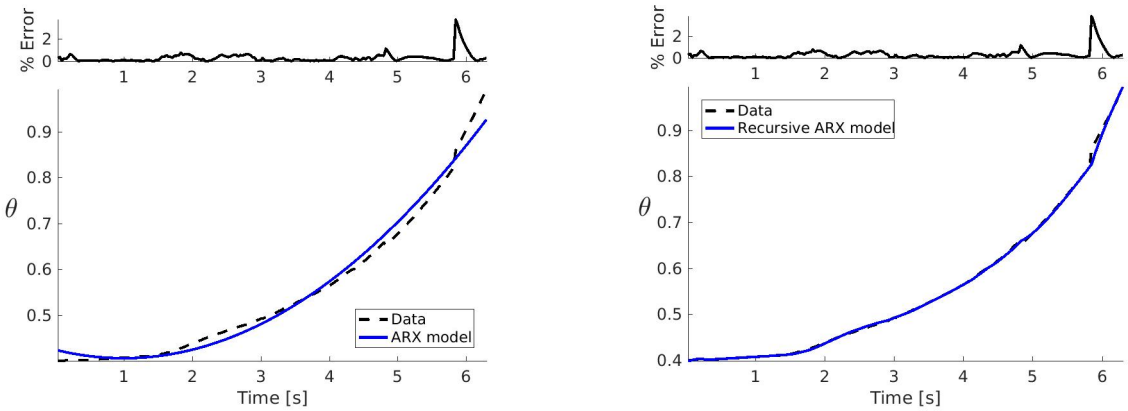
Las Figuras 40 y 41 muestran los resultados de utilizar el enfoque de identificación de sistemas a las secuencias anteriores. En ambas Figuras se comparan la versión recursiva y no recursiva de ARX. En la línea punteada negra se representa el comportamiento de los datos originales, mientras

que la línea continua azul representa el ajuste del modelo. En cada Figura se coloca en la parte de arriba el error obtenido con respecto al ajuste, donde podemos notar que en el caso del cilindro rojo el modelo ARX recursivo se ajusta mejor porque se obtiene información en cada frame y el modelo es actualizado. En el caso de la esfera verde, el recursivo o no recursivo prácticamente arrojan el mismo ajuste.



(a) (b)

Figura 40: Comparación entre los datos y los modelos ARX y ARX recursivo en la secuencia del cilindro rojo. (a)Modelo usando el enfoque ARX del cilindro rojo. (b)Modelo usando el enfoque ARX recursivo del cilindro rojo.



(a) (b)

Figura 41: Comparación entre los datos y los modelos ARX y ARX recursivo en la secuencia de la esfera verde. (a)Modelo usando el enfoque ARX de la esfera verde. (b)Modelo usando el enfoque ARX recursivo de la esfera verde.

Los cuadros 6 y 7 presentan los parámetros de los modelos ARX y ARX recursivo para los obstáculos cilíndricos y esféricos. Se puede ver en estos cuadros que ambos métodos de modelado producen prácticamente los mismos valores de parámetros. Es importante tener en cuenta que los parámetros cambian cuando se estiman a través del método Recursive ARX. Esto se debe a que los datos se están procesando en secuencia. Comienza desde una aproximación inicial, luego los datos se procesan y causan cambios en los valores de los parámetros.

Cuadro 6: Parámetros de los modelos obtenidos mediante ARX y ARX recursivo para el caso del cilindro rojo.

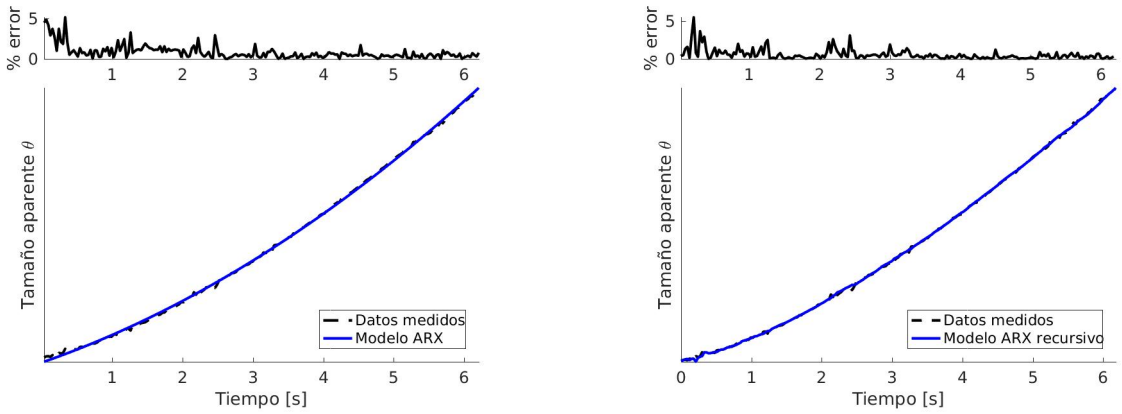
Modelo	α_2	α_1	α_0
Modelo ARX	1.407e-05	0.001421	1.104
Modelo ARX recursivo	1.4069e-05	0.0014	1.1042

Para el escenario de los tres objetos simultáneos, las Figuras 42, 43 y 44 muestran el ajuste a las secuencias del tamaño aparente en el tiempo (en segundos) de estas tres figuras: la caja del cubo

Cuadro 7: Parámetros de los modelos obtenidos mediante ARX y ARX recursivo para el caso de la esfera verde.

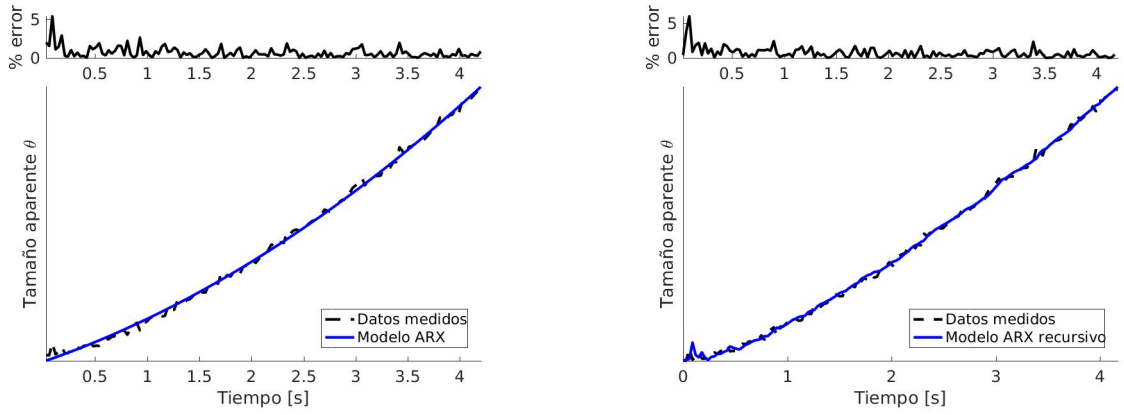
Modelo	α_2	α_1	α_0
Modelo ARX	1.664e-05	-0.001103	0.424
Modelo ARX recursivo	1.6636e-05	-0.0011	0.4240

de rubik, el búho y el águila vestida respectivamente. También se grafican los errores donde puede observarse que en el obstáculo del águila existe mayor error, y eso se debe a que la ropa tiene ruido, pues con los pliegues de ésta genera diversas configuraciones en vez de un cuerpo rígido.



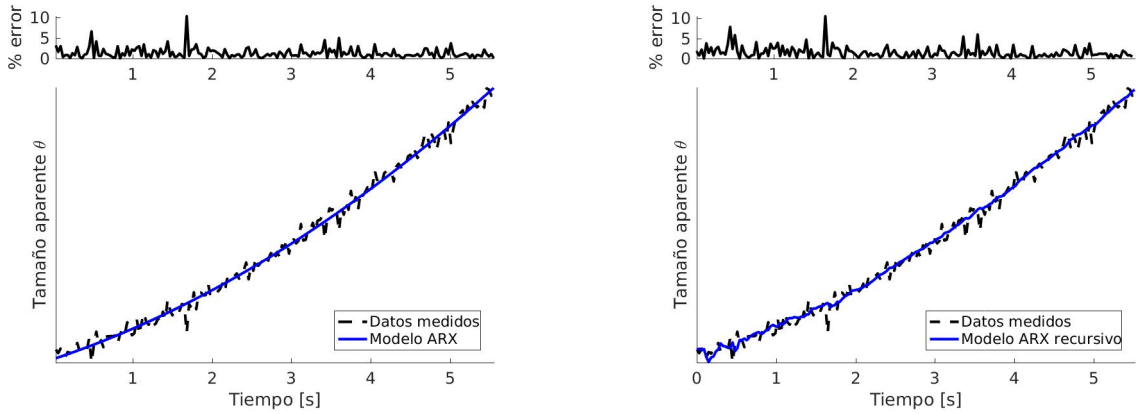
(a) (b)

Figura 42: Comparación entre los datos y los modelos ARX y ARX recursivo en la secuencia de la caja del cubo de rubik en la escena de obstáculos simultáneos. (a)Modelo usando el enfoque ARX de la caja del cubo de rubik. (b)Modelo usando el enfoque ARX recursivo de la caja del cubo de rubik.



(a) (b)

Figura 43: Comparación entre los datos y los modelos ARX y ARX recursivo en la secuencia de la figura del búho en la escena de obstáculos simultáneos. (a)Modelo usando el enfoque ARX de la figura del búho. (b)Modelo usando el enfoque ARX recursivo de la figura del búho.



(a) (b)

Figura 44: Comparación entre los datos y los modelos ARX y ARX recursivo en la secuencia de la figura de águila vestida en la escena de obstáculos simultáneos. (a)Modelo usando el enfoque ARX del águila vestida. (b)Modelo usando el enfoque ARX recursivo del águila vestida.

También los Cuadros 8, 9 y 10 muestran los parámetros de cada objeto en sus versiones ARX y

ARX recursivo, donde se observa que los parámetros en todos los obstáculos, en ambas versiones de ARX, son prácticamente los mismos, sin embargo existe menos error en los recursivos.

Cuadro 8: Parámetros de los modelos obtenidos mediante ArX y ARX recursivo para el caso de la caja del cubo de rubik

Modelo	α_2	α_1	α_0
Modelo ARX	0.00198	0.5088	61.52
Modelo ARX recursivo	0.001867	0.5104	64.67

Cuadro 9: Parámetros de los modelos obtenidos mediante ArX y ARX recursivo para el caso de la figura del búho

Modelo	α_2	α_1	α_0
Modelo ARX	0.001685	0.2906	72.79
Modelo ARX recursivo	0.001163	0.3878	68.67

Cuadro 10: Parámetros de los modelos obtenidos mediante ArX y ARX recursivo para el caso de la figura del águila

Modelo	α_2	α_1	α_0
Modelo ARX	0.0007138	0.1551	50.92
Modelo ARX recursivo	0.0007216	0.1431	52.56

Además de la comparación visual en cada escenario, el ajuste del modelo se comparó con los datos usando (105). El cuadro 11 muestra el porcentaje de ajuste de cada modelo a los datos correspondientes. Se puede ver que el modelo ARX recursivo proporciona un mejor ajuste que el modelo ARX no recursivo en la mayoría de los casos, excepto cuando existe demasiada variabilidad como es el caso de la figura del águila.

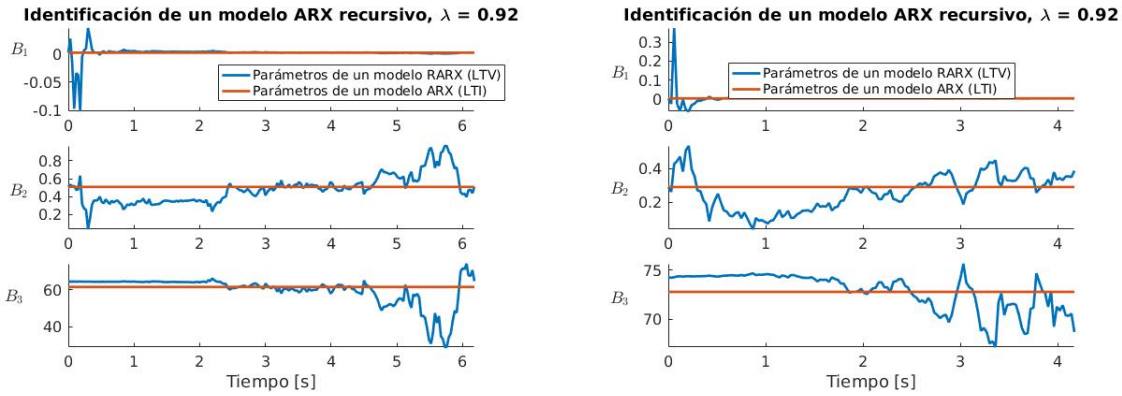
También mostramos en la Figura 45 cómo varían los parámetros de cada uno de los modelos de los obstáculos en el segundo escenario (obstáculos múltiples). Esto es importante, pues los parámetros de un modelo ARX son constantes en el tiempo debido a que se obtienen offline al final de recolectar los

Cuadro 11: Comparación del porcentaje de ajuste de los modelos de los datos usando identificación de sistemas.

Escenario	Modelo ARX	Modelo ARX recursivo
Cilindro rojo	93.62 %	98.23 %
Esfera verde	88.46 %	97.28 %
Caja cubo rubik	97.9 %	98.3136 %
Figura búho	95.75 %	95.5563 %
Figura águila	91.97 %	90.8766 %

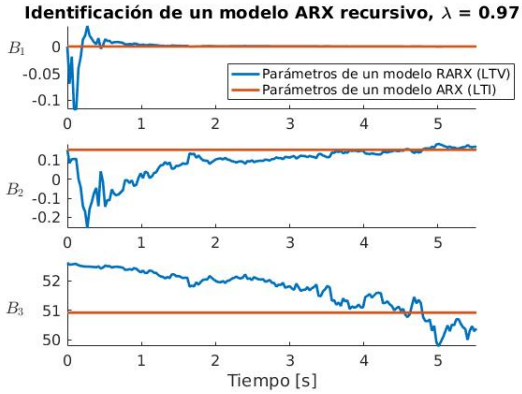
datos. Por lo tanto se ven como una linea horizontal indicando que no existen cambios en el tiempo. Las líneas azules muestran las actualizaciones de los parámetros en un modelo ARX recursivo, es decir, los valores que van tomando dependen de los datos que obtenemos en el tiempo y del factor de olvido (este factor entre 0 y 1 da un peso exponencialmente menor a las muestras de errores pasados o anteriores) $\lambda = 0.92$). En otras palabras, cada vez que se obtiene una nueva medición, se reconstruye el modelo y por lo tanto cambian los parámetros (similarmente al filtro de Kalman).

Se puede notar que los parámetros que menos variaron son los de la Figura 45 (b) (ver escalas) y esto se debe a que las características encontradas en este obstáculo son menos variables que las de los otros dos, que aunque estos dos tienen mucha textura, las característias del cubo de rubik son muy diferentes entre ellas y hace que no existan tanto ruido.



(a)

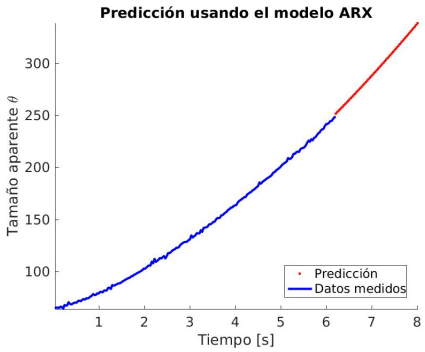
(b)



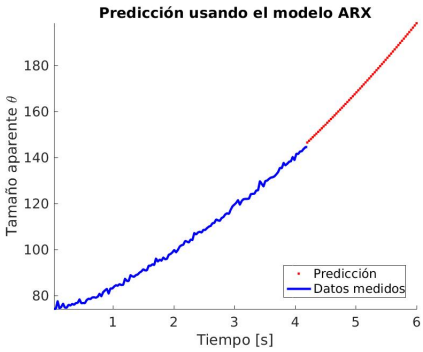
(c)

Figura 45: Comparación en el tiempo de los parámetros de los modelos ARX y ARX recursivo de los tres obstáculos simultáneos. (a)Cambio de parámetros para la caja del cubo de rubik. (b)Cambio de parámetros para la figura de búho. (c)Cambio de parámetros para la figura de águila.

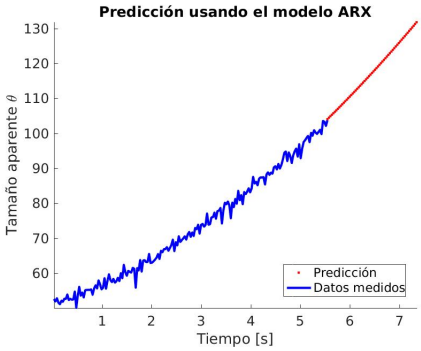
En la Figura 46 se aprecian los pronósticos del cambio del tamaño aparente en el tiempo de los 3 obstáculos simultáneos mediante el modelado ARX no recursivo, donde se aprecia que el pronóstico en rojo es muy suave, sin variaciones y sin outliers, lo que nos provee estimaciones más confiables de TTC.



(a)



(b)

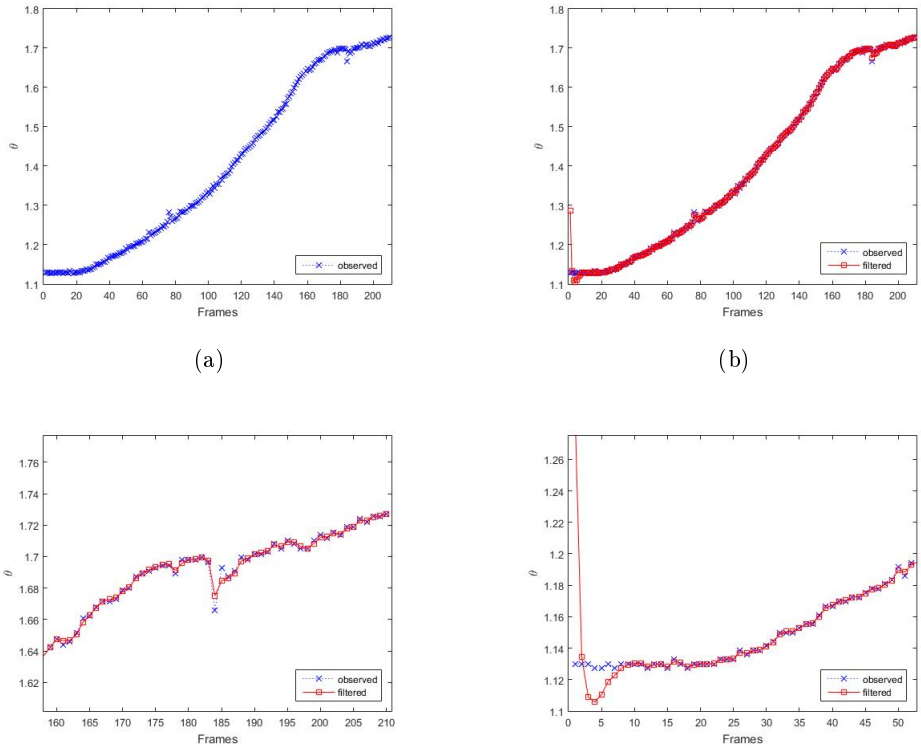


(c)

Figura 46: Pronósticos del tamaño aparente de los tres obstáculos simultáneos mediante el modelo ARX no recursivo. (a)Pronóstico para la caja del cubo de rubik. (b)Pronóstico para la figura de búho. (c)Pronóstico para la figura de águila.

Utilizando el enfoque del filtro de Kalman, la Figura 47 muestra el filtrado de las observaciones del escenario del cilindro rojo, mediante este enfoque. Figura 47 (a) y 47 (b) muestran las observaciones en azul y los datos filtrados en rojo respectivamente. En la Figura 47 (c) mostramos un ejemplo de las observaciones y fltrado donde existe mucha variabilidad, con el fin de notar más de cerca cómo se comporta el método cuando existe variación en las observaciones, con lo que notamos que a pesar de esto, el modelo se recupera rápidamente. En la Figura 47 (d) muestra un acercamiento al inicio del filtrado. Esto último es mostrado con el fin de ver cuántos frames tarda en encontrar un modelo

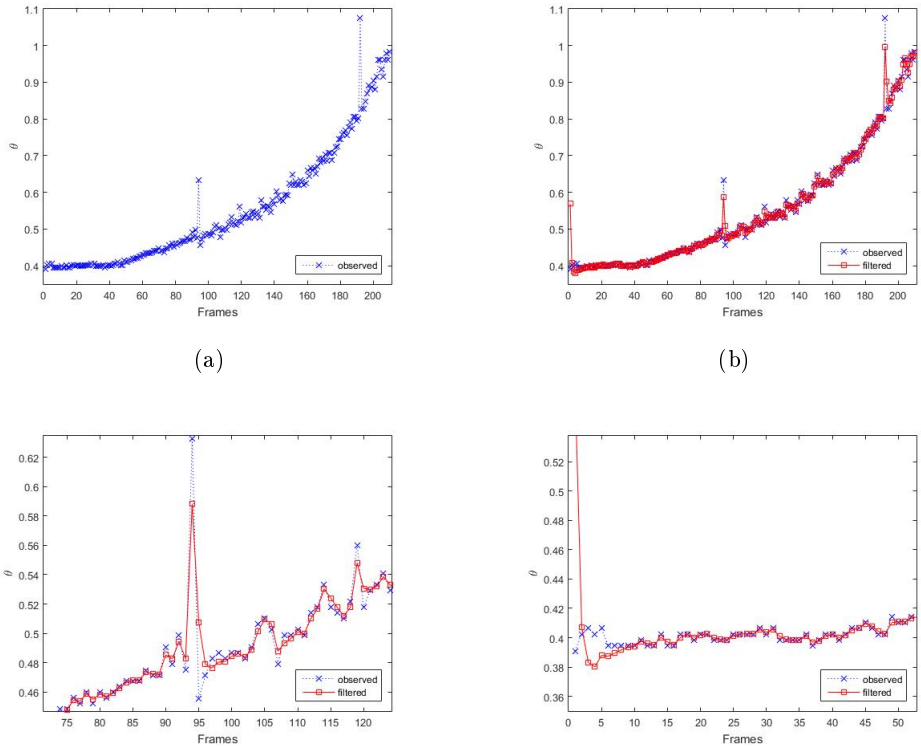
ajustado adecuado, y se puede observar que se tardó alrededor de 10 frames. Por último la Figura 48 resalta los mismos aspectos que la Figura 47, solo que esta vez con el escenario de la esfera verde.



(a) (b)

(c) (d)

Figura 47: Filtrado de las observaciones del escenario del cilindro rojo, mediante el Filtro de Kalman. (a)Observaciones obtenidas. (b)Modelo filtrado. (c)Ejemplo de las observaciones con mucha variabilidad. (d)Inicio del filtrado.

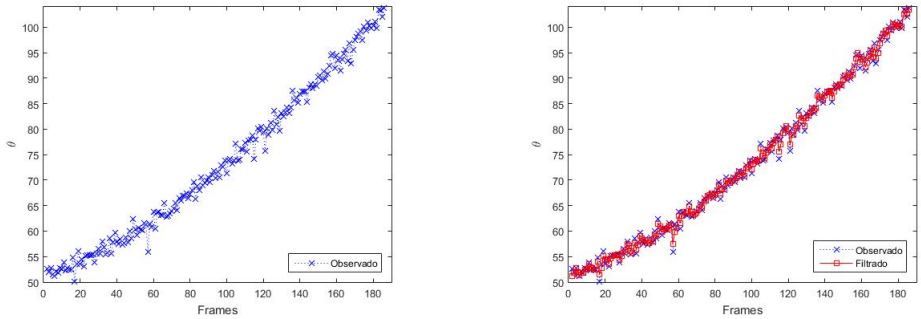


(a) (b)

(c) (d)

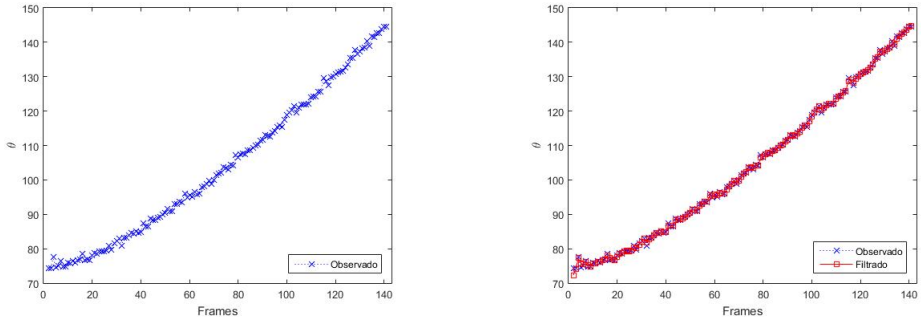
Figura 48: Filtrado de las observaciones del escenario de la esfera verde, mediante el Filtro de Kalman. (a)Observaciones obtenidas. (b)Modelo filtrado. (c)Ejemplo de las observaciones con mucha variabilidad. (d)Inicio del filtrado.

Las Figuras 49, 50 y 51 muestran las secuencias para los obstáculos de la configuración de los 3 obstáculos simultáneos mencionados anteriormente. En dichas Figuras se observa un buen ajuste a los datos originales y un filtrado que suaviza dichas secuencias.



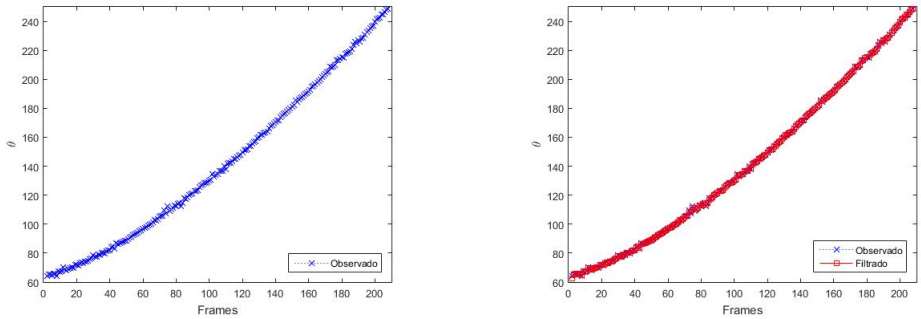
(a) (b)

Figura 49: Filtrado de las observaciones del escenario de 3 obstáculos simultáneos en la escena, específicamente la figura del águila. (a)Observaciones obtenidas. (b)Modelo filtrado.



(a) (b)

Figura 50: Filtrado de las observaciones del escenario de 3 obstáculos simultáneos en la escena, específicamente la figura del buho. (a)Observaciones obtenidas. (b)Modelo filtrado.



(a) (b)

Figura 51: Filtrado de las observaciones del escenario de 3 obstáculos simultáneos en la escena, específicamente la figura de la caja del cubo de rubik. (a)Observaciones obtenidas. (b)Modelo filtrado.

Cuadro 12: Conclusiones de los enfoques presentados para modelar el tamaño aparente de los obstáculos

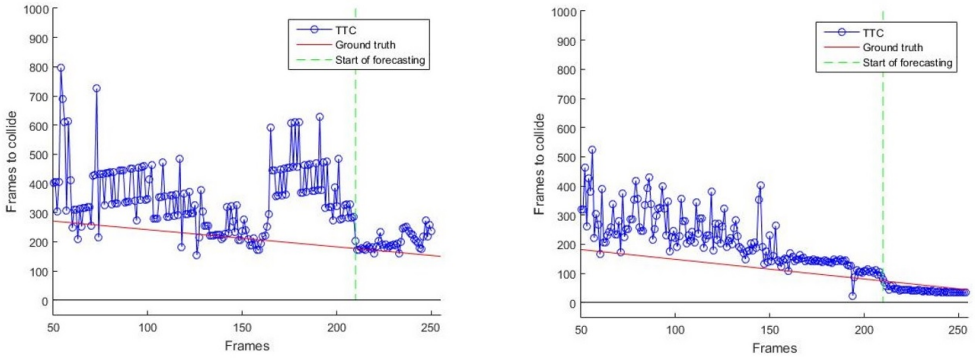
Método	Características	Inconvenientes
Series de tiempo	▪ Colecta un conjunto de datos y pronostica.	▪ Estadísticamente es confiable alrededor de 5 datos hacia adelante.
Identificación de sistemas	▪ Genera un modelo a partir de un conjunto de datos y suaviza. ▪ En su versión recursiva puede ir incorporando datos.	▪ En su versión no recursiva, no toma en cuenta si el ambiente cambió. ▪ Puede ser lento al incorporar información.
Filtro de Kalman	▪ En cada momento toma un valor, genera un modelo, predice y recalcula con base en el error.	▪ Tarda en generar un modelo "preciso" o converger. ▪ Puede ser lento al incorporar información.

Para concluir las evaluaciones de estos enfoques para modelar, el Cuadro 12 muestra de manera resumida las características de cada enfoque, así mismo como los inconvenientes de cada uno.

6.5. Evaluación del Tiempo al Contacto

A continuación las Figuras 52 y 53 muestran los resultados de la estimación del TTC en los ambientes reales. En ambas Figuras, el color azul indica la estimación del TTC online, la línea roja indica el Ground truth obtenido empíricamente mediante mediciones manuales y la línea verde indica el instante de tiempo a partir del cual se realiza el pronóstico.

En la Figura 52 se muestran las estimaciones del TTC obtenidos a partir de la segmentación por contornos probabilísticos (debido a que como se mencionó anteriormente, se obtuvieron mejores resultados que con la segmentación por color) y el modelado ARX no recursivo, mostrando que posterior al lapso de tiempo para pronóstico, la variabilidad disminuye, acercándose al ground truth.

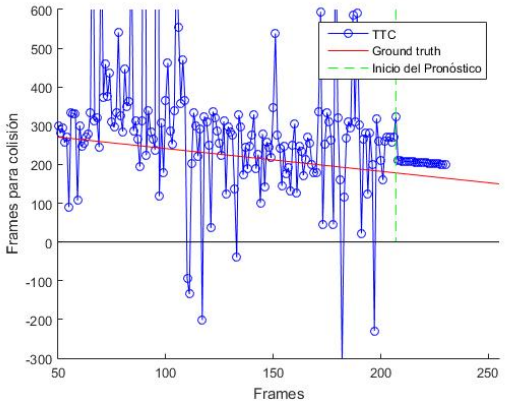


(a) (b)

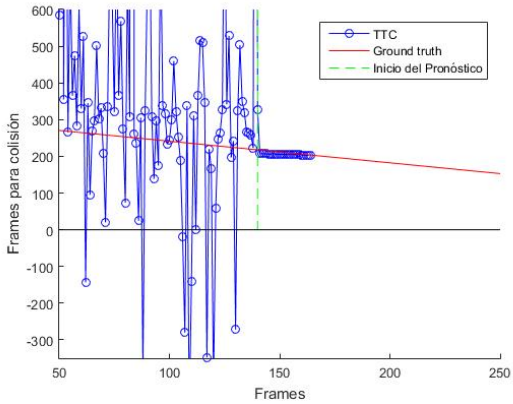
Figura 52: Estimación del TTC. (a) TTC para la percepción del cilindro rojo. (b) TTC para la percepción de la esfera verde.

En la Figura 53 se muestran las estimaciones del TTC obtenidos a partir de la segmentación

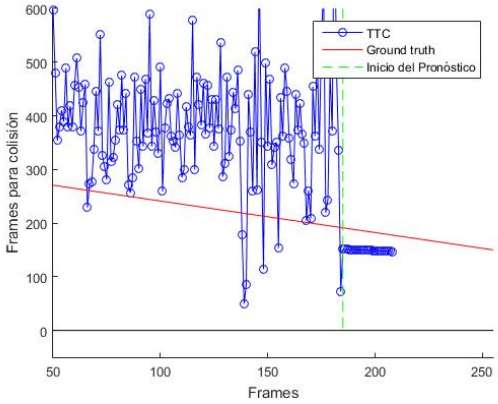
por aprendizaje de características en el escenario de obstáculos simultáneos y el modelado ARX no recursivo. Aunque para este caso podrían estimarse muchas predicciones, para visualización solo se obtuvo en todos los casos el pronóstico de 25 frames hacia adelante. En esta Figura se aprecia que los pronósticos se vuelven más suaves, pues por la predicción es suave y constante. En el caso de la Figura del búho este se ajusta al ground truth, mientras que en los otros casos, se sobre estima o se subestima. El peor caso sería la sobrestimación, pues puede inferir que aún falta por colisionar cuando el robot ya chocó con el obstáculo.



(a)



(b)



(c)

Figura 53: Estimación del TTC para la escena de obstáculos simultáneos. (a)TTC para la percepción de la caja del cubo de rubik. (b)TTC para la percepción de la figura de búho. (c)TTC para la percepción de la figura de águila.

6.6. Evaluación del módulo de robustecimiento

Partiendo de que el robot se mueve a velocidad constante, evaluamos el comportamiento de los valores de $tau - margin$ obtenidos como una aproximación lineal, para saber qué tan precisa es esta estimación. Como se planteó en el capítulo V, la idea es aproximar los datos a un modelo lineal para

robustecer las estimaciones. En esta sección se mostrará los resultados de uno de los casos, aunque el detalle de los demás se muestran en el apéndice A.

La Figura 54 muestra el TTC del caso de la esfera, donde en verde aparece cómo se ajustaría un Modelo de Regresión Lineal simple, en el que el TTC es la variable dependiente y el número de frame en el tiempo es la variable independiente. Como se puede apreciar, el modelo que se obtiene sigue el comportamiento de los datos, sin embargo como los datos de TTC sobreestiman el TTC del ground truth, también lo hace su modelo lineal. Lo mismo pasaría si las estimaciones de TTC subestiman en el ground truth.

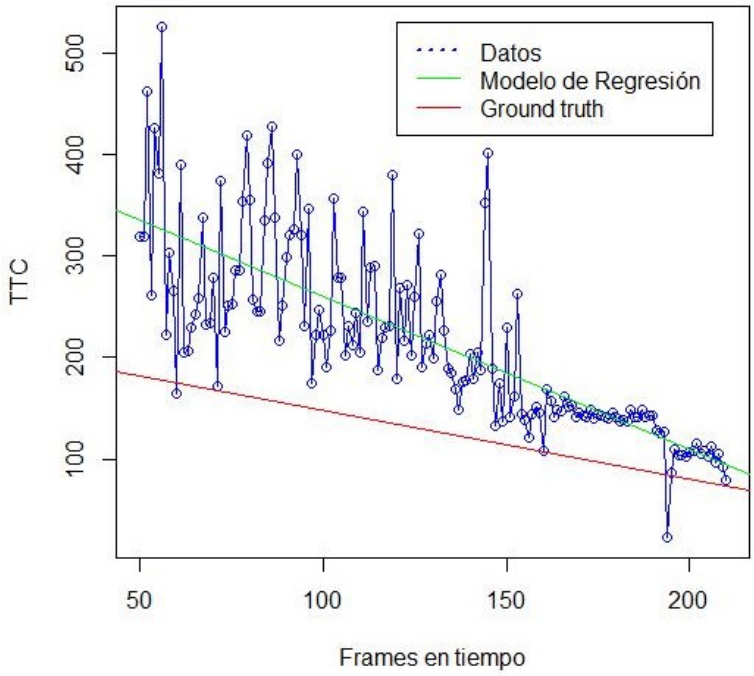


Figura 54: Modelo de regresión lineal del caso de la esfera verde.

Consideraremos cuatro supuestos para los modelos lineales y diferentes formas de diagnosticarlos.

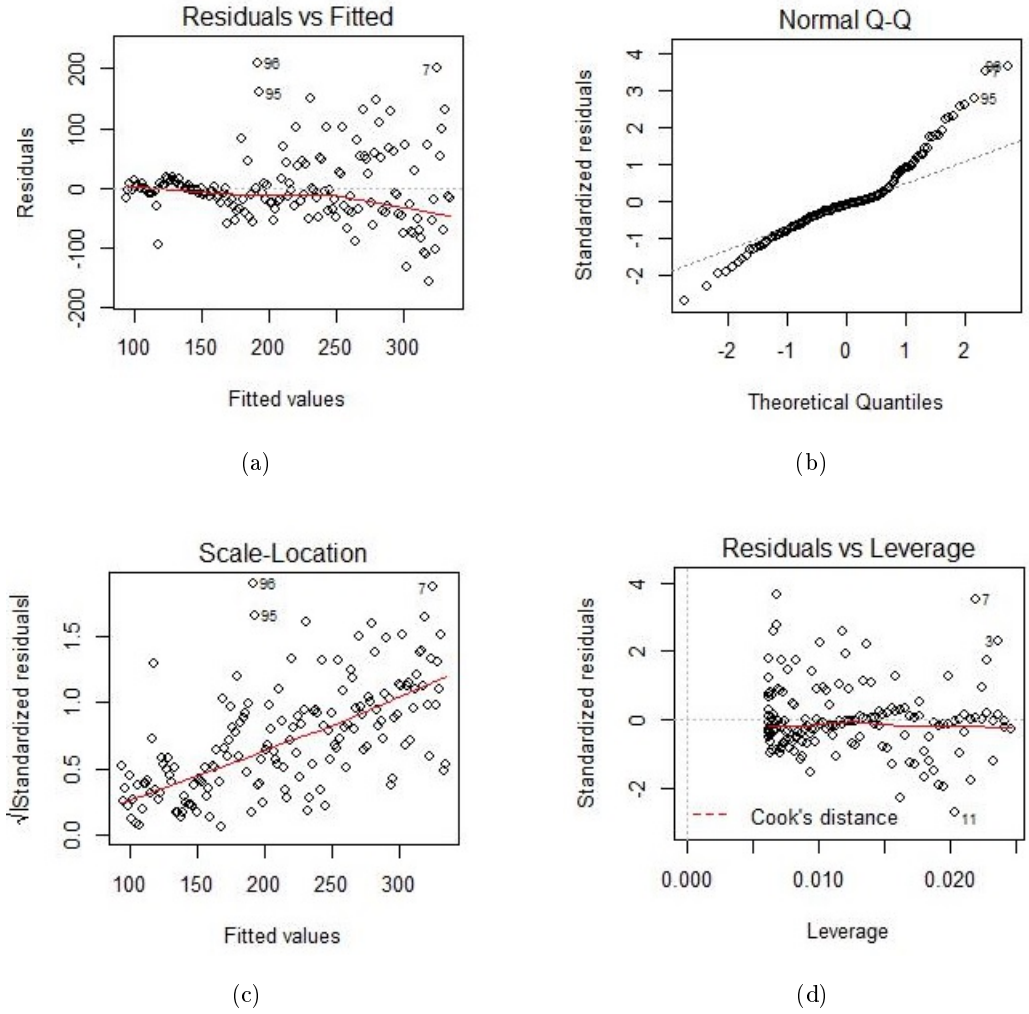
- **Normalidad.** Para valores fijos de la variable independiente, la dependiente se distribuye de manera normal. Consiguientemente, los residuos se distribuyen de manera normal y por lo tanto tienes una media con una respectiva variabilidad.

- **Linealidad.** La variable dependiente y la independiente están asociadas de manera lineal vs. asociadas a través de una curva, parábola, etc.

- **Independencia.** Los valor de y son independientes unos de otros, es decir, no existe autocorrelación. También se eliminan las posibles formas de dependencia entre x e y que no sean la relación lineal entre las variables.

- **Homocedasticidad.** La varianza de la variable dependiente es homogénea a lo largo de los valores de la variable independiente.

Diagnosticar si estamos violando alguno de estos supuestos requiere un conjunto de pruebas de diagnóstico. La Figura 55 muestra los resultados del análisis del Modelo Lineal que se obtuvo.



(a) (b)

(c) (d)

Figura 55: Ejemplo de los resultado para saber si existe violación de supuestos. (a)Gráfico de comparación de valores residuales contra ajustados. (b)Gráfico Normal Q-Q. (c)Gráfico de valores residuales estandarizados. (d)Gráfico de distancia de Cook y Leverage.

Para el supuesto de **Normalidad**, esperamos que los residuos se distribuyan de manera normal. La Figura 55 (b) Normal-QQ compara a los residuos estandarizados con una distribución normal teórica. Si se cumple el supuesto de normalidad de los residuos, los puntos deberían alinearse sobre

la recta que corta el gráfico en 45 grados. Como vemos, no es el caso: especialmente para los valores más altos hay una desviación importante de los residuos.

Para el caso del supuesto de **independencia**, las probabilidades de las variables de x e y deben ser independientes y las probabilidades de cada valor de y independientes de las otras.

Para el supuesto de **linealidad**, la variable dependiente está linealmente relacionada con la independiente. La relación podría ser de otro modo: curva, parabólica, etc. Una forma de diagnosticar este problema es graficar los datos y graficar la recta de ajuste lineal y la curva de ajuste de algún modelo de regresión local. En la Figura 55 (c) Residuals vs. Fitted podemos ver que la curva de ajuste en rojo no es recta y horizontal, lo que señalaría que valores observados y residuos se distribuyen entre sí aleatoriamente. Posiblemente una función que incluya una curva sea mejor que nuestra aproximación lineal.

Con respecto al supuesto de **homocedasticidad**, indica que la varianza de la variable dependiente no varía a lo largo de la variable independiente, es decir, en el modelo de regresión simple la varianza es constante de los errores. Podríamos llamarlo “varianza constante", pero usualmente es referido comohomocedasticidad. la Figura 55 (c) nos ayuda para este diagnóstico, donde se puede observar que la dispersión de los residuos sí varía. Sin embargo, esto podría deberse a que al principio existe mucha incertidumbre, y conforme se acerca al obstáculo existe más certidumbre de la estimación del TTC.

Por último, dada la propuesta planteada en el capítulo anterior, suponemos que hay valores que influyen mucho en el modelo y que al descartarlos podría generar una estimación más robusta. Es una propuesta realizada ya que una de las fuentes de violaciones de los supuestos en el modelado lineal muy común es la presencia de casos atípicos. Un caso atípico es aquel que, dados ciertos de

valores de x, tiene valores de y muy diferentes a los demás y por lo tanto producen un residuo muy alto. Estos casos atípicos pueden tener gran influencia sobre el modelo, ya que el criterio de mínimos cuadrados buscará minimizar el error y cambiará la pendiente para dar cuenta de estos casos. La Figura 55 (d) muestra el apalancamiento (leverage) y la distancia de Cook descritos en el capítulo anterior. Este gráfico muestra que, no existen datos muy influyentes en el modelo de regresión lineal.

De hecho se realizó el experimento de bajar el umbral de la distancia de Cook a 0.02 e ir quitando en cada iteración datos atípicos con ese umbral, hasta contar con un conjunto de datos que no tuvieran datos con influencia superior a ese umbral. Aunque es un número muy bajo (recordar que se esperarían valores superiores a 0.5), quisimos ver si cambiaba el modelo de manera significativa. La Figura 56 muestra el resultado de este procesamiento, quedándose solo con 60, de los 210 datos al principio, notándose que la recta de regresión es casi la misma y por lo tanto no tiene sentido realizar este procesamiento. Incluso el TTC está sobreestimado, por lo cual se chocaría puesto que el TTC indicará que falta aún tiempo para la colisión.

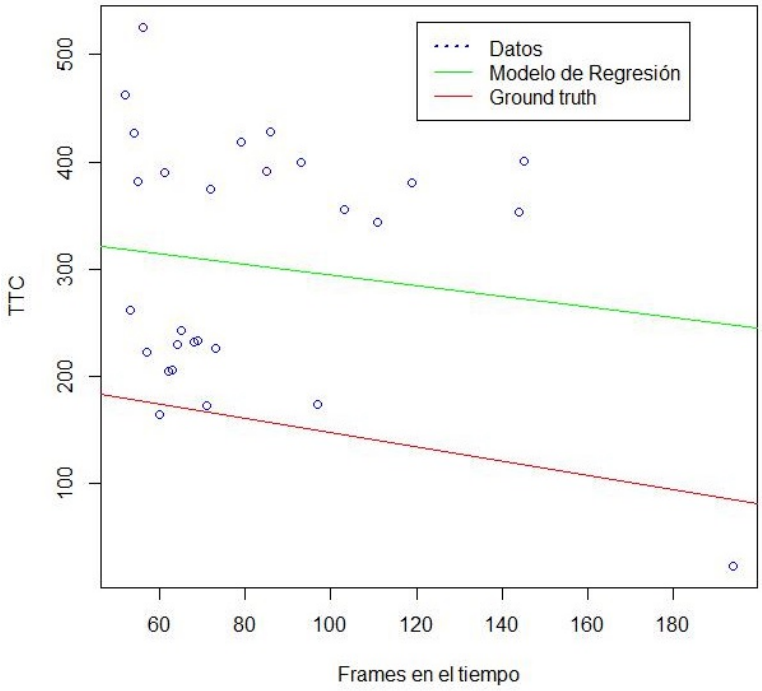


Figura 56: Modelo de regresión lineal del caso de la esfera verde quitando datos influyentes con un umbral = 0.02.

Cuadro 13: Comparación de los atributos de los Modelos Lineales generados

Modelo Lineal	Pendiente	Intercepto
Ground truth	-0.6730769	260
TTC	-1.50195	409.726
TTC sin datos influyentes	-0.495	343.616

El Cuadro 13 muestra los valores de la pendiente y el intercepto del Ground truth, el modelo lineal de los datos, y del nuevo modelo lineal quitándo datos atípicos con un umbral de la distancia de Cook muy bajo, observándose que aunque la pendiente se acerca al Ground truth, el intercepto es sobreestimado generando potenciales colisiones. Por lo tanto se muestra que no tiene caso explorar esta propuesta. Además los datos más influyentes son los que se acercaban al TTC o los más próximos a colisionar.

Por lo tanto, en este capítulo se presentó el diseño y los resultados de la experimentación de esta investigación. Se realizaron experimentos sobre el módulo de sementación de los 3 enfoques (color, contornos probabilísticos y aprendizaje de características), así como de los enfoques de modelado (Filtro de Kalman, modelo ARIMA, y modelo ARX recursivo y no recursivo). También se presentaron pronósticos de estimaciones de TTC mediante los modelos ARX no recursivos, los cuales muestran ser más estables que si se estimara en cada cuadro mediante la fórmula convencional.

Conclusiones

En este trabajo, se presenta una propuesta para robustecer y pronosticar la estimación del Tiempo de contacto (TTC) en la navegación robótica, es decir, calcular cuántos cuadros quedan para colisionar con un objeto cuando el robot se mueve a una velocidad constante basados en tau-margin (τ). Se presenta una descripción de la literatura, destacando las ventajas y desventajas de los diferentes enfoques utilizados.

Con los resultados obtenidos, confirmamos la hipótesis planteada en el Capítulo I, afirmando que es posible robustecer la estimación del TTC utilizando sensores pasivos para pronosticar colisiones con obstáculos de manera precisa en la navegación de robots autónomos.

Una de las principales aportaciones, es que en nuestra propuesta, se utiliza un sensor pasivo como una cámara, a través de visión por computadora para la navegación robótica sin requerir hardware especializado y así reducir costos de energía y monetarios, a diferencia de otros enfoques mencionados como estimación mediante sonares, láseres, visión estéreo, entre otros.

Conclusiones de segmentación

Primero se describieron las bases matemáticas para calcular el tamaño aparente de los obstáculos proyectados en una imagen sin usar algún crecimiento de regiones. Con el fin de identificar los obstáculos, nosotros probamos tres enfoques para segmentar los obstáculos: segmentación por color, contornos probabilísticos basados en modelos ocultos de Markov e inferencia MAP, y basados en extracción de características de objetos previamente conocidos.

También la implementación y evaluación de la segmentación de manera secuencial y paralela se

describió en este trabajo. El análisis muestra que el rendimiento cuando usamos la segmentación con procesamiento paralelo mediante el uso la GPU se reduce en un 66 %, por lo tanto, funciona en tiempo real incluso con imágenes grandes, que nos da una mayor precisión. Además, para comparar las distribuciones de los tiempos, se concluye que la segmentación secuencial varía con el tiempo debido a los diversos procesos que se ejecutan dentro de la computadora de prueba. Estas diferencias fueron estadísticamente probadas,y entonces concluimos que el procesamiento paralelo usando GPU mejora sustancialmente el rendimiento del procesamiento y se aplica a tareas que requieren procesamiento en tiempo real como estimación del TTC. Estos resultados van acompañados de una descripción del problema en los tamaños de imagen y el análisis de complejidad de secuenciales y paralelos, para conocer las ventajas de segmentación en paralelo.

El segundo método para identificar obstáculos utiliza contornos activos basados en modelos ocultos de Markov e inferencia MAP, donde se demuestra estadísticamente que es más consistente ya que este método es invariante al cambio de iluminación como lo hace la segmentación por color; por consiguiente, los errores se reducen cuando el robot detecta un obstáculo.

También fue presentado un método para detectar posibles obstáculos con características aprendidas y al mismo tiempo. Es importante enfatizar que en nuestro propuesta, el objeto no se identifica, sino se buscan las características que nos permiten seguir al objeto. En nuestro método, a diferencia de los métodos basados en la detección de obstáculos por color y descritos en revisión de trabajos relacionados, eliminamos los problemas que surgen al calibrar el color específico del objeto, y que esta calibración debería cambiar de acuerdo con la luz ambiental que existe. También con este método es posible detectar múltiples obstáculos en la escena. A diferencia de los métodos basados en flujo óptico, con este método, el robot podrá saber qué partes de la escena es un obstáculo (información a priori) y que no lo es.

Conclusiones de modelado

La principal contribución consiste en predecir el crecimiento del obstáculo detectado para estimar directamente el tiempo que llevaría al robot colisionar con un obstáculo en lugar de calcularlo en cada cuadro. Por lo tanto, el tamaño aparente del obstáculo se puede predecir a través de un proceso de modelado recursivo en línea (en tiempo real), es decir, sin buscar la imagen completa en cada cuadro. Para ello, probamos tres métodos: series temporales, Filtro de Kalman y modelado basado en indentificación de sistemas, siendo identificación de sistemas en su versión no recursiva, la opción que nos parece más viable debido a que, de acuerdo a la comparación de tres enfoques, nos permite suavizar las estimaciones, suavisando las observaciones y no requiere ir incorporando nueva información en cada frame. Además en esta versión el modelo offline proporciona un valor constante de los parámetros del modelo. Por ejemplo, si hay cambios en la iluminación, la predicción del modelo se puede ajustar de manera subóptima al sistema real utilizando el modelo ARX. Por el contrario, los parámetros del modelo Recursive ARX se actualizan en tiempo real (es decir, cada vez que se procesa una nueva imagen (frame)).

Algunas características del filtro de Kalman también se destacan y describimos cómo hace la estimación de nuestro problema. Este enfoque también es probado en casos reales donde se observa el proceso de modelado y la proximidad a las medidas tomadas, reduciendo ruido de las medidas. Este enfoque está ganando fuerza porque es más fácil predecir (dadas algunas medidas tomadas anteriormente) que buscar obstáculos en cada cuadro. Sin embargo, el hecho de aproximarse demasiado a las observaciones, podría llevarnos a un sobreajuste, lo que implicaría llevar al modelo todos los datos atípicos.

Cuando es generado un modelo de cómo se comporta el cambio de tamaño aparente de un obstáculo detectado, descartamos medidas que no tienen un comportamiento igual al resto, y que

pueden ser causadas por factores del entorno o algún problema en el sensor, con lo cual estamos fortaleciendo nuestras estimaciones.

Conclusiones de robustecimiento

En esta propuesta por robustecer el TTC derivado de problemas que pudieran ocurrir, notamos en las secuencias que adquirimos, existen pocos datos que sean influyentes y se toma como un modelo de regresión lineal (bajo el supuesto de la velocidad constante), por lo que realizar un procesamiento de eliminar outliers parece irrelevante.

Se generó también una propuesta para robustecer las estimaciones, ya que, bajo la suposición de que el robot se mueve constantemente, el tamaño aparente del objeto en el momento t, será mayor que el tiempo $t - 1$. Para dar mayor solidez a nuestras estimaciones, se propone, en las mismas circunstancias, que el robot genera una aproximación lineal mediante la eliminación de los valores que aparentemente no comparten un comportamiento lineal como el resto de las estimaciones. Esto con el fin de considerar factores externos como cambios en la iluminación o problemas cuando el robot está en movimiento, no afecta la estimación del tiempo de contacto. Además, el método es robusto porque no solo depende de datos anteriores, sino que depende de los datos que se comportan en de la misma manera con el tiempo. Finalmente, los modelos y los datos obtenidos se compararon y se analizaron estadísticamente de forma visual y cuantitativa, donde la robustez del método se observó mediante la eliminación de valores atípicos para crear modelos más confiables. Finalmente, es importante mencionar que los modelos podrían ser más confiables con una mayor cantidad de frames. Sin embargo, como se puede ver en este documento, los resultados con datos limitados encajan bien con el comportamiento buscado.

Trabajo futuro

Como trabajo futuro se puede hablar de tres vertientes principales. El primero de ellos está en estimar el Ground truth obtenida mediante algún sensor especializado, como láser para medir distancia, posición o desplazamiento sin contacto ni rozamiento, sensores de distancia por ultrasonidos o sensores de distancia inductivos que miden en rangos bajos sin contacto, sobre superficies metálicas, entre otros.

Otra vertiente para continuar con este trabajo se encuentra ubicada en el robustecimiento. En este trabajo propuse la eliminación de datos atípicos mediante aproximaciones a un modelo lineal, sin embargo cada vez que se encontraban datos atípicos, éstos eran eliminados y se volvían a identificar datos atípicos. Como se mostró en este trabajo, el inconveniente de este enfoque es que al final podríamos quedarnos con pocos datos que no representaran de manera adecuada el comportamiento del modelo. Por eso, como trabajo futuro también se propone experimentar utilizando Random sample consensus (RANSAC). RANSAC es un método iterativo para calcular los parámetros de un modelo matemático de un conjunto de datos observados que contiene valores atípicos. La desventaja es que es un algoritmo no determinista en el sentido de que produce un resultado razonable sólo con una cierta probabilidad, pero experimentar con un determinado número de iteraciones podría aproximarse más al comportamiento de la mayoría de los datos sin necesidad de descartar datos que potencialmente parecieran atípicos.

La última opción que se deja para trabajo futuro es la aplicación de la estimación robusta del TTC a algún problema en particular. No solo se utiliza el TTC en tareas de evasión de obstáculos, en particular, se ha pensado en tareas como aterrizaje ("landing") y estacionado ("docking").

Referencias

[1] Abdulla A, Liu H, Stoll N, Thurow N. A New Robust Method for Mobile Robot Multi-floor Navigation in Distributed Life Science Laboratories. Journal of Control Science and Engineering.2016:1–7. 2016.

[2] Yoon WS, Park SB, Kim JS. Kalman filter sensor fusion for Mecanum wheeled automated guided vehicle localization. Journal of Physical Agents. 7:3–11. 2013.

[3] Lopez J, Watkins C, Perez D, Diaz M. Evaluating different landmark positioning systems within the RIDE architecture. Journal of Sensors. 7:3–11. 2013.

[4] Mi Z, Yang Y, Yang JY. Restoring connectivity of mobile robotic sensor networks while avoiding obstacles. IEEE Sensors Journal. 15:4640–4650. 2015.

[5] Birk A, Liu H,Schwertfeger S, Pathak K. A networking framework for teleoperation in safety, security, and rescue robotics. IEEE Wireless Communications. 16:6–13. 2009.

[6] Chetty RMK, Singaperumal M, Nagarajan T. Distributed formation planning and navigation framework for wheeled mobile robots. Journal of Applied Sciences. 11:1501–1509. 2011.

[7] Ohya A, Kosaka A, Kak A. Vision-based navigation by a mobile robot with obstacle avoidance using single-camera vision and ultrasonic sensing. IEEE Transactions on Robotics and Automation. 1998;14:969–978. 1998.

[8] Voss C, Moll M, Kavraki E. A heuristic approach to finding diverse short paths. Proceedings of 2015 International Conference on Robotics and Automation (ICRA). May 26–30; Seattle, WA, USA. 2015.

[9] Palmieri L, Rudenko A, Arras K. A Fast Random Walk Approach to Find Diverse Paths for Robot Navigation. IEEE Robotics and Automation Letters. 2017;2:269–276. 2017.

[10] Mohanty PK, Parhi DR. A new hybrid optimization algorithm for multiple mobile robots navigation based on the CS-ANFIS approach. Memetic Computing. 7:255–273. 2015.

[11] Korayem MH, Zehfroosh A, Tourajizadeh H, Manteghi S. Optimal motion planning of non-linear dynamic systems in the presence of obstacles and moving boundaries using SDRE: application on cable-suspended robot. Nonlinear Dynamics. 76:1423–1441. 2014.

[12] Negahdaripour, S. *A direct Method for locating the Focus of Expansion*, Tecnical Report. A. I. Memo No 939. 1987.

[13] Oda, N., Yoneda, J. *Visual Feedback Control Based on Optical Flow Vector Field for Biped Walking Robot*, Mechatronics (ICM), 2013 IEEE International Conference on. 2013.

[14] Asano, Y., Kawamura, A. *Decoupled rotational motion control for visual walking stabilization*, IEEE Int. Workshop on Advance Motion Control (AMC2008), pp.68-73, 2008.

[15] Kim, J., Park, I. Lee, J., Oh, J. *Experiments of vision guided walking of humanoid robot*, KHR-2?, IEEE-RAS Int. Conf. on Humanoid Robots, pp.135-140, 2005.

[16] Rieger, J. H., Lawton, D.T. *Processing differential image motion*, J. Opt. Soc. Ame. A, 2(2):354-359. 1985.

[17] Angelopoulou, M., Bouganis, C. *Vision-Based Egomotion Estimation on FPGA for Unmanned Aerial Vehicle Navigation*, IEEE TTransactions on Circuits and Systems for Video Techology, Vol. 24:6, 2014.

[18] Nelson, R., Aloimonos *Using flow field divergence for obstacle avoidance: towards qualitative vision*, Proc. 2nd int. conf. on Computer visión, pp. 188 - 196. 1988.

[19] Cipolla, R., Blake, A. *Surface Orientation and Time to Contact from Divergence and Deformation*, Computer Vision - ECCV '92. Second European Conference on Computer Vision. 1992.

[20] Tistarelli M, Grosso E, Sandini G. Dynamic Stereo in Visual Navigation. Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Jun 3–6; Lahaina, Maui, Hawaii. 1991

[21] Tian, T., Tomasi, C., Heeger, D. *Comparison of Approaches to Egomotion Computation*, Proceedings CVPR 96. IEEE Computer Society on Computer Vision and Pattern Recognition. 1996.

[22] Bruss, A., Horn, B. K. *Passive navigation.* Computer Graphics and Image Processing, 21:3?20, 1983.

[23] Jepson, A. D., Heeger, D. J. *A fast subspace algorithm for recovering rigid motion*, In Proceedings of IEEE Workshop on Visual Motion,pages 124?131, Princeton, NJ, 1991.

[24] Jepson, A. D., Heeger, D. J. *Linear subspace methods for recovering translation direction*, In L Harris and M Jenkin, editors, Spatial Vision in Humans and Robots, pages 39?62. CambridgeUniversity Press, New York, 1993.

[25] Tomasi, C., Shi, J. *Direction of heading from image deformations*, In Proceedings of IEEE Computer Vision and Pattern Recognition, pp. 422?427, New York, 1993.

[26] Prazdny, K. *Egomotion and relative depth map from optical flow*, Biological Cybernetics, 36:87?102, 1980.

[27] Kanatani, K. *3-d interpretation of optical flow by renormalization*, International Journal of Computer Vision, 11(3):267?282, 1993.

[28] Faugueras, O. D., Lustman, F., Toscani, G. *Motion and structure from motion from point and line matches*, Proceedings of the First International Conference on Computer Vision, pp. 25-34, 1987.

[29] Longuet-Higgins, H. C., *A Computer Algorithm for reconstructing a scene from two projections*, Nature, 293(10): 133-135, 1981.

[30] Stein, G., Mano, O., Shashua, A. *A robust Method for Computing Vehicle Ego-Motion*, Intelligent Vehicles Symposium. Proceedings of IEEE. 2000.

[31] Badino, H. A robust Approach for Ego-Motion Estimation using a Mobile Stereo Platform, First International Workshop, IWCM 2004, Günzburg, Germany, 2004.

[32] Ancona, N., Poggio, T. *Optical flow from 1d correlation: Application to a simple time-to-crash detector*, Proceedings of the International Conference on Computer Vision, 2005.

[33] Song, X., Seneviratne, L., Althoefer, K. *A Kalman Filter-Integrated Optical Flow Method for velocity Sensing of Mobile Robots*, IEEE-ASME Transactions on mechatronics, Vol. 16, No. 3, 2011.

[34] Samija, H., Markovic, I., Petrovic, I. *Optical Flow Field Segmentation in an Omnidirectional Camera Image Based on Known Camera Motion*, In proceeding of: MIPRO, 2011 Proceedings of the 34th International Convention. 2011.

[35] Yamagushi, K. McAllester, D., Urtasun, R. *Robust monocular Epipolar Flow Estimation*, Computer Vision and Pattern Recognition (CVPR). 2013.

[36] Sánchez, A., Ríos, H. *Segmentación de Objetos en movimiento por flujo óptico y color sin información a priori de la escena*, Research in Computing Science Avances en Inteligencia Artificial, ISSN: 1870-4069, Vol 62, pp. 151-160. 2013

[37] Sánchez, A., Ríos H., Marín, A. *Description of Motion of Segmented Regions*, IEEE Mexican International Conference on Computer Science (ENC). 2013.

[38] Sánchez, A., Ríos, H., Marín, A., Acosta, H. *Tracking and Prediction Motion of Segmented Regions Using the Kalman Filter*, International Conference on Electronics, Communications and Computers CONIELECOMP. 2014.

[39] Alenya G, Negre A, Crowly JL. A Comparison of Three Methods for Measure of Time to Contact. Proceedings of International Conference on Intelligent Robots and Systems, Oct 10–15; Saint Louis, USA, 2009

[40] Lee DN. A theory of visual control of braking based on information about time-to-collision. Perception. 1976;5:437–459. 1976.

[41] Horn BKP, Fang Y, Masaki I. Hierarchical framework for direct gradient-based time-to-contact estimation. Proceedings of IEEE Intelligent Vehicle Symposium. Jun 3–5; Xi'an, China. 2009.

[42] Alenya G, Negre A, Crowly JL. Time To Contact for Obstacle Avoidance. Proceedings of the 4th European Conference on Mobile Robots (ECMR) 09; Sep 23–25; Dubrovnik, Croatia. 2009.

[43] Muller D, Pauli J, Nunn C, et al. Time To Contact Estimation Using Interest Points. Proceedings of the 12th International IEEE Conference on Intelligent Transportation Systems,Oct 3–7; St. Louis, Mo. 2009.

[44] Dev, A., Krose, B., and Groen, F., *Navigation of a mobile robot on the temporal development of the optic flow*, in Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems, pp. 558 - 563, 1997.

[45] Carelli, R., Soria, C., and Nasisi, O. *Stable agv corridor navigation with fused vision-based control signals*,in Proceedings of the 28th Annual Conference of IEEE Industry Electronics Society, pp. 2433 - 2438, 2002.

[46] Shim BK, Yang JS, Kim EG, et al. A travelling control of mobile robot based on sonar sensors.

Proceedings of the 15th International Conference on Control, Automation and Systems (ICCAS), Oct 13–16; Busan, Korea. 2015

[47] Bamorovat MH, Oskoei MA, Fakharian A. Mobile robot navigation using sonar vision algorithm applied to omnidirectional vision. Proceedings of the AI and Robotics (IRANOPEN 2015),Apr 12; Qazvin, Iran. 2015.

[48] Vascak J, Hvizdos J. Vehicle navigation by fuzzy cognitive maps using sonar and RFID technologies. Proceedings of the 14th International Symposium on Applied Machine Intelligence and Informatics (SAMI), Jan 21–23; Her?any, Slovakia. 2016

[49] Gehrig SK, Eberli F, Meyer T. A Real-Time Low-Power Stereo Vision Engine Using Semi-Global Matching. Proceedings of the 7th International Conference on Computer Vision Systems: Computer Vision Systems, Oct 13–15; Liège, Belgium. 2009.

[50] Muffert M, Milbich T, Pfeiffer D, et al. May I enter the roundabout? A time-to-contact computation based on stereo-vision. Proceedings of the IEEE Intelligent Vehicles Symposium (IV), Jun 3–7; Alcala de Henares, Spain. 2012

[51] Benamar FZ, Fkihi SE, Demonceaux C, et al. Gradient-Based time to contact on paracatadioptric camera. Proceedings of the International Conference on Image Processing, ICIP'2013. Sep 15–18; Melbourne, Australia. 2013

[52] Yu W, Sommer G, Daniilidis K.Multiple motion analysis: in spatial or in spectral domain?. Computer Vision and Image Understanding. Vol. 90:129–152. 2003

[53] Izumi S, Yamaguchi T. Time-to-Contact Estimation on the Scaled-Matching of Power Spectra. Proceedings of the SICE, 2007 Annual Conference. Sep 17–20; Kagawa, Japan. 2007

[54] Honda R. Algorithm of rotation estimation using Fourier transform [master's thesis]. [Kumamoto]: Kumamoto University, Japan; 2001.

[55] Schoukens J, Pintelon R, Rolain Y. Time domain identification, frequency domain identification. Equivalencies! Differences?. Proceedings of the 2004 American Control Conference. Jun 30–Jul 2; Boston, USA. 2004.

[56] Horn BKP, Schunck, B. Determining optical flow. Journal of Control Science and Engineering.17:185–203. 1981.

[57] O'Donovan P. Optical Flow: Techniques and applications [B.Sc. thesis]. [Saskatchewan]: The University of Saskatchewan; 2005.

[58] Lucas,B. D., Kanade, T. *An iterative image registration technique with an application to stereo vision*, Proceedings of the 1981 DARPA Imaging Understanding Workshop, pp. 121-130, 1981.

[59] Illeperuma, G. and Sonnadara, U. *An Autonomous Robot Navigation System Based on Optical Flow*, Proceedings of the 2010 IEEE, International Conference on Robotics and Biomimetics, 2010.

[60] Liyanage, D. and Perera, M. *Optical Flow based Obstacle Avoidance for the Visually Impaired*, 6th International Conference on Industrial and Information Systems, ICIIS, 2011.

[61] Meyer F. Time-to-collision from first-order models of the motion field. IEEE Transactions on Robotics and Automation. Vol. 10:792–798. 1994.

[62] Meyer F, Bouthemy P. Estimation of time-to-collision maps from first order motion models and normal flows. Proceedings of the 11th IAPR International Conference on Pattern Recognition; Aug 30–Sep 3; The Hague, The Netherlands. 1992.

[63] Naigong Y, Yuling Z, Xu L, et al. Optical Flow Based Mobile Robot Obstacle Avoidance Method in Unstructured Environment. Journal of Beijing University of Technology. 43:65–69. 2017.

[64] Coombs, C., Herman, M., Hong, T., and Nashman, M.*Real-time obstacle avoidance using central flow divergence and peripheral flow*, in Computer Vision, 1995. Proceedings., Fifth International Conference on, pp. 276 - 283, IEEE, 1995.

[65] Subbarao, M. *Bounds on time-to-collision and rotational component from first-order derivatives of image flow* Comput. Vis., Graph. Image Process, Vol. 50, pp. 329?341. 1990.

[66] Nebot, P., Cervera, E. *Cooperative Navigation Using the Optical Flow and Time-to-Contact Techniques*, 2008 10th Intl. Conf. on Control, Automation, Robotics and Vision, Hanoi, Vietnam, 2008.

[67] Hatsopoulos NG, Warren WH. Visual navigation with a neural network. Neural Networks. 4:303–317. 1991.

[68] McCarthy C, Barnes N, Mahony R. A Robust Docking Strategy for a Mobile Robot Using Flow Field Divergence. IEEE Transactions on Robotics. Vol. 24:832–842. 2008.

[69] McCarthy C, Barnes N. A Unified Strategy for Landing and Docking Using Spherical Flow Divergence. IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 34:1024–1031. 2012

[70] Liau YS, Zhang Q, Li Y, et al. Non-Metric Navigation for Mobile Robot Using Optical Flow. Proceedings of 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. Oct 7–12; Vilamoura, Algarve, Portugal. 2012

[71] Sanchez A, Rios H, Marin A, et al. Decision making for obstacle avoidance in autonomous mobile robots by time to contact and optical flow. Proceedings of 2015 International Conference

on Electronics, Communications and Computers (CONIELECOMP) 2015. Feb 25–27; Cholula, Mexico. 2015

[72] Longuet-Higgins, H. C., Prazdny, K.*The Interpretation of a Moving Retinal Image*, In Proc. of the Royal Society B, pp. 385?397.1980.

[73] Horn BKP, Fang Y, Masaki I. Time to Contact Relative to a Planar Surface. Proceedings of the 2007 Intelligent Vehicles Symposium. Jun 13–15; Istanbul, Turkey. 2007.

[74] McQuirk IS, Horn BKP, Lee HS, et al. Estimating the Focus of Expansion in Analog VLSI. International Journal of Computer Vision. Vol. 28:261–277. 1998.

[75] Kaneta Y, Hagisaka Y, Ito K. Determination of time to contact and application to timing control of mobile robot. Proceedings of the 2010 International Conference on Robotics and Biomimetics (ROBIO). Dec 14–18; Tianjin, China. 2010.

[76] Sanchez A, Rios H, Marin A, et al. Estimation of Time-To-Contact from Tau-margin and Statistical Analysis of Behavior. Proceedings of the 2016 International conference on Systems, Signals and Image Processing IWSSIP,May 23–25; Bratislava, Slovakia. 2016.

[77] Negre A, Braillon C, Crowley JL, et al. Real-time time-to-collision from variation of intrinsic scale. Proceedings of the 10th International Symposium on Experimental Robotics. Jul 6–12; Rio de Janeiro, Brazil. 2006.

[78] Harris C, Stephens M. A combined corner and edge detector. Proceedings of Fourth Alvey Vision Conference. Aug 31–Sep 2; Manchester, UK. 1988.

[79] Sandres, J. and Kandrot, E. *Cuda By Example An Introduction to General Purpose GPU Programming* , NVIDIA Corporation, 2011.

[80] Represa, C., Camara, J., Sanchez, P. *Introducción a la programación en CUDA*, Departamento de Ingeniería Electromecánica, Universidad de Burgos. Vol. 3.1, 2016.

[81] Bradski G. and A. Kaebler, *Learning OpenCV. Computer vision with the OpenCV library*, Oreilly, First Edition, 2008.

[82] Kass M, Witkin A, Terzopoulos D. Snakes: Active contour models. International Journal of Computer Vision. Vol. 1, pp. 321–331. 1988.

[83] Felzenszwalb P, Zabih R. Dynamic Programming and Graph Algorithms in Computer Vision. IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 33:721–740. 2011.

[84] Narasimba M, Susheela V. Pattern Recognition An Algorithmic Approach. Springer; 2011.

[85] Prince SJD. Computer vision: models, learning and inference. Cambridge University Press; 2012.

[86] Amini A, Weymouth T, Jain R. Using dynamic programming for solving variational problems in vision. IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol 12:855–867. 1990

[87] Duda RO, Hart PE. Pattern Classification and Scene Analysis. John Wiley & Sons Inc; 1973.

[88] Shi, J. and Tomasi, C. *Good Features to Track*, IEEE Conference on Computer Vision and Pattern Recognition, 1994.

[89] Lowe, D. *Distinctive Image Features from Scale-Invariant Keypoints*, International Journal of Computer Vision, Vol. 60, No. 3, pp. 91–110, 2004.

[90] Bay, H., Tuytelaars, T. and Van Gool, L. *SURF: Speeded Up Robust Features*, Computer Vision and Image Understanding (CVIU) 110 (3), pp. 346–359, 2008.

[91] Rosten, E. *FAST Corner Detection*, Engineering Department, Machine Intelligence Laboratory, University of Cambridge, 2006.

[92] Calonder, M., Lepetit, V., Strecha, C. and Fua, P., *BRIEF: Binary Robust Independent Elementary Features*, European Conference on Computer Vision, pp. 778–792, 2010.

[93] Rublee, E., Rabaud, V., Konolige, K. and Bradski, G., *ORB: an ef?cient alternative to SIFT or SURF*, IEEE International Conference on Computer Vision (ICCV), 2011.

[94] Bradski G. and Kaehler A. Learning OpenCV. 1st ed. USA: O'Reilly Media, Inc; 555 p. 2008.

[95] Kalman R. E. A New Approach to Linear Filtering and Prediction Problems. Transactions of the ASME-Journal of Basic Engineering. Vol. 82(1):35-45. DOI: 10.1115/1.3662552. 1960.

[96] Maybeck P. Introduction. In: Academic Press, editor. Stochastic models, estimation and control. 1st ed. London. p. 1-16. 1979.

[97] S. Haykin, *Topics in Applied Physics, Nonlinear Method of Spectral Analysis*, Vol. 34 Springer-Verlag, 1983.

[98] S. M. Kay and S. L. Marple, *Spectrum Analysis- a modern perspective*, Proc. IEEE, vol. 69, pp. 1380 - 1419, 1981.

[99] S. Hansun, *A New Approach of Moving Average Method in Time Series Analysis*, Microwave Symposium Digest, New Media Studies (CoNMedia), 2013.

[100] Pintelon R, Schoukens J. System identification: A Frequency Domain Approach. IEEE Press; 2012.

[101] Keesman K. System Identification: An Introduction (Advanced Textbooks in Control and Signal Processing). Springer; 2011.

[102] Ljung L. System identification. Theory for the user. Prentice Hall; 1999.

[103] Garnier H, Wang L. Identification of continuous-time models from sampled data. Springer-Verlag; 2008.

[104] Engin M. Embedded LQR Controller Design for Self-Balancing Robot. 7th Mediterranean Conference on Embedded Computing(MECO), 11-14 june, Budva, Montenegro; 2018.

[105] Jiago J, Chen J, Quiao J, Wang W, Wang C, Gu L. Single Neuron PID Control of Agricultural Robot Steering System Based on Online Identification. IEEE Fourth International Conference on Big Data Computing Service and Applications; 2018.

[106] Prakash A, Zuhayr M. Indoor Positioning System Simulation for a Robot using Radio Frequency Identification. 13th IEEE Conference on Industrial Electronics and Applications(ICIEA); 2018.

[107] Young P. Recursive Estimation and Time-series Analysis. An Introduction for the Student and Practitioner. Springer-Verlag; 2011.

[108] Markovsky I. Low-Rank Approximation, Algorithms, Implementation, Applications. Springer; 2014.

[109] Ljung L. System identification toolbox for use with Matlab, 9th edition, The Mathworks, Inc, Natick, MA, 2014.

[110] Salazar, P., Sánchez, S. *Matemáticas 2*, Compañía Editorial Nueva Imagen, ISBN: 978-607-8387-63-2, 2017.

[111] Everitt, B. S. *The Cambridge Dictionary of Statistics*, 2nd edition. Cambridge University Press, Cambridge, UK. ISBN 0 521 81099, 2002.

[112] StatXact, Statistical Software for Exact Nonparametric Inferece, user manual, Cytel Statistical Software, 2004.