CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS
AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL

UNIDAD ZACATENCO

DEPARTAMENTO DE CONTROL AUTOMÁTICO

"Modelado y control de un vehículo autónomo"

TESIS

Que presenta

OSCAR GONZÁLEZ MIRANDA

Para obtener el grado de

MAESTRO EN CIENCIAS

EN LA ESPECIALIDAD DE CONTROL AUTOMÁTICO

Director de la Tesis: Dr. Juan Manuel Ibarra Zannatha

Ciudad de México Diciembre, 2019

Resumen

El objetivo de esta tesis es obtener modelos aproximados de un vehículo autónomo a escala: el AutoNOMOS de la Universidad Libre de Berlín; así como desarrollar diversos esquemas de control, basados en visión, que permitan la conducción autónoma de dicho vehículo en una carretera de la cual se desconoce el mapa. Para ello deberá implementarse un sistema de visión que genere las mediciones del posicionamiento del vehículo en la carretera. El AutoNOMOS es un vehículo con dirección tipo Ackerman, para el cual en este trabajo se propone un modelo cinemático lineal y discreto, basado en el conocido modelo de la bicicleta. Las únicas mediciones disponibles para hacer el control son las que proporciona una cámara de video embarcada. Debido a las características de su óptica y al montaje sobre el vehículo, el campo de visión de dicha cámara permite ver el piso a partir de 17 centímetros frente al vehículo. Esto es, que no se dispone de mediciones del posicionamiento relativo actual entre el vehículo y la carretera; sino el que tendrá unos segundos más tarde. Así, el objetivo del sistema de visión es el de obtener la información de la forma de la carretera, a fin de poder controlar la posición del vehículo para que permanezca centrado en su carril y además, poder controlar la dirección del vehículo para que pueda tomar las curvas con precisión y a la mayor velocidad posible. Las leyes de control que se desarrollaron para guiar de manera autónoma al vehículo son: control basado en campos potenciales, control óptimo y un controlador tipo "*preview controller*". Tanto el sistema de visión como el de control se implementaron como nodos ROS y se ejecutan en el sistema de cómputo de abordo. El desempeño de los controladores se evaluó tanto en simulación como de manera experimental en el AutoNOMOS.

Abstract

The objective of this thesis is to obtain approximate models of an autonomous vehicle at scale: the AutoNOMOS of the Free University of Berlin; as well as developing various control schemes, based on vision, that allow the autonomous driving of this vehicle on a road whose map is unknown. For this, a vision system must be implemented that generates the measurements of the positioning of the vehicle on the road. The AutoNOMOS is a vehicle with Ackerman type steering, for which in this work a linear and discrete kinematic model is proposed, based on the well-known bicycle model. The measurements available to control are those provided by a video camera on board. Due to the characteristics of its optics and the mounting on the vehicle, the field of vision of this camera allows to see the floor from 17 centimeters in front of the vehicle. That is, there are no measurements of the current relative positioning between the vehicle and the road; but the one who will have a few seconds later. Thus, the objective of the vision system is to obtain the information of the shape of the road, in order to be able to control the position of the vehicle so that it remains centered in its lane and also, to be able to control the direction of the vehicle so that it can take curves accurately and at the highest possible speed. The control laws that were developed to guide the vehicle autonomously are: control based on potential fields, optimal control and a “ textit preview controller ” type controller. Both the vision and control systems were implemented as ROS nodes and are executed in the on-board computer system. The performance of the controllers was evaluated both in simulation and experimentally in the AutoNOMOS.

Agradecimientos

Primero quiero agradecer a mi familia pues todo su esfuerzo y dedicación me ha dado las oportunidades que ellos en su día no tuvieron. Quiero que sepan que este logro también es suyo y recordaré siempre lo que han hecho por mí. A mi mamá le agradezco su devoción y cuidado. Todo ese amor me motiva a ser mejor persona cada día y a hacer mi mejor esfuerzo. A mi papá le agradezco el esfuerzo y la confianza que da depositado en mí. Es un hombre a quien admiro y respeto; y me da orgullo poder decir que soy su hijo. Mi hermana Laura es una mujer fuerte, inteligente e independiente; cuyos logros, aún después de muchos años, me siguen sorprendiendo y admirando. Siempre me escucha y me aconseja; y me siento feliz al saber que me acompañará cuando lo necesité. A mi hermano Miguel le agradezco el empeño que pone en su trabajo y el cariño que siempre me demuestra.

También quiero agradecer al Dr. Juan Manuel Ibarra Zannatha y a sus alumnos, mis amigos: Santos Orozco Soto, Pablo Vera Bustamante y Alessio Daniel Hernandez Rojas. Ha sido un honor y un placer hacer investigación a su lado; y gracias a ellos he aprendido mucho. También quiero mencionar a mi Jurado de Tesis: el Dr. Alejandro Justo Malo Tamayo y al Dr. Jorge Antonio Torres Muñoz quienes con sus críticas ayudaron a mejorar y a darle calidad a este trabajo.

Finalmente quiero agradecer al Centro de Investigación y de Estudios Avanzadas del Instituto politécnico Nacional, por la oportunidad de hacer el posgrado y a la Comisión Nacional de Ciencia y Tecnología (CONACyT) por el apoyo económico brindado estos años. Todo esto me permitió continuar con mi formación como investigador.

Índice general

Capítulo 1

Introducción

Los vehículos autónomos han dejado de ser una fantasía de la ciencia ficción. Recientes avances tecnológicos han hecho posible el desarrollo de automóviles de conducción autónoma con cierto grado de robustez. Algunos de estos avances son: el desarrollo de sensores como: radars, lidars, cámaras RGBD, dispositivos GPS, etc; la creación de bases de datos de mapas y de reportes de tránsito; la optimización de los métodos de SLAM (por las siglas en inglés de *Simultaneous Localization an Mapping*), la mejora en los algoritmos de visión artificial, la creación de controladores de dirección, de velocidad, de señalización de un automóvil; entre otros [1, 2]. En la actualidad grandes empresas como Google, Uber, Tesla, Mercedes Benz, General Motors, Nissan, entre otras, han empezado una feroz carrera para obtener primero un vehículo autónomo cómodo y seguro [1]. Aunque ya existen algunos de estos vehículos en el mercado, aún existen muchos problemas técnicos que se deben resolver, para que estos automóviles sean seguros bajo cualquier condición. Algunos de estos problemas son: Clima extremo, caminos no pavimentados, interrupción en los sistemas *wireless*, etc.

Se prevee que esta nueva tecnología cuente con los siguientes beneficios [2]:

- **Reducir el estrés e incrementar la productividad:** Los pasajeros podrán descansar, jugar o trabajar mientras viajan.

- **Movilidad para las personas que no pueden conducir:** Los vehículos autónomos podrán dotar de movilidad a personas con capacidades diferentes, adolescentes y otros que; por alguna razón, no sean capaces de conducir. Esto beneficia directamente a los pasajeros ya que mejora sus oportunidades de obtener educación o un empleo y se incrementa su productividad económica.

- **Reducir los costos de viaje:** Se espera que el costo del transporte público se reduzca; al facilitar la compartición del vehículo, disminuir los viajes totales del mismo y disminuir la necesidad de un auto propio.

- **Incremento en la seguridad:** Se podrían reducir los riesgos de choque y por lo tanto, el costo de un seguro.

- **Incrementar la capacidad de las carreteras y reducir su costo:** El tránsito de vehículos será más eficiente y se reducirá la congestión y el costo de las carreteras.

- **Reducir el costo de estacionamiento:** Se reducirá la demanda y por lo tanto el costo de los espacios de estacionamiento.

- **Reducción del consumo de energía y contaminación:** Se prevee que los vehículos autónomos optimizarán su consumo de energía y reducirán la emisión de contaminantes.

Por el contrario, los vehículos autónomos podrían tener asociados los siguientes problemas:

- **Incrementar el costo de los vehículos autónomos:** Este tipo de autos requerirán un equipamiento especial, servicios y mantenimiento; lo que elevará su costo a corto y largo plazo.

- **Vulnerabilidad en la seguridad y la privacidad de los datos de los pasajeros:** Este tipo de tecnología podría hackearse para abusar de los datos de los pasajeros. Datos como la localización y el trayecto de un usuario ponen en riesgo su privacidad.

- **Riesgos adicionales:** La nula intervención humana convierte al vehículo en un lugar de riesgo donde se susciten actividades delictivas.

- **Reducción del empleo:** En una ciudad con vehículos autónomos es muy probable que desaparezcan oficios como el de chofer.

- **Incremento en el costo de infraestructuras:** Se podrían requerir carreteras, especialmente diseñadas para estos autos, que necesiten un mantenimiento específico.

De acuerdo con la SAE (*Society of Automobile Engineers*) se han definido 6 niveles de asistencia al conductor para vehículos autónomos [3, 4]:

Nivel 0: El vehículo es conducido por un humano y no hay alguna tecnología de asistencia.

Nivel 1: Existe un sistema avanzado de asistencia al conductor (ADAS por sus siglas en inglés *Advanced Driver Assistance System*) que, bajo ciertas circunstancias, puede auxiliar al conductor humano con el control de la dirección del vehículo o con la aceleración-frenado del mismo; pero no ambos a la vez.

Nivel 2: El ADAS del vehículo puede controlar simultáneamente, tanto la dirección del vehículo como la aceleración-frenado del mismo. Requiere que el conductor humano esté siempre monitoreando la conducción y desempeñando el resto de las tareas que el ADAS no pueda llevar a cabo.

Nivel 3: El ADAS puede desempeñar todas las tareas de conducción bajo ciertas circunstancias; sin embargo, requiere que el conductor humano esté listo para retomar el control cuando se le solicite o cuando el ADAS no pueda conducir el automóvil.

Nivel 4: Bajo ciertas circunstancias el ADAS del vehículo es capaz de conducir y monitorear el ambiente que lo rodea. Dentro de este régimen no es necesario que un humano ponga atención al camino.

Nivel 5: El vehículo es completamente autónomo en cualquier escenario. Sus ocupantes son solamente pasajeros que no son necesarios en la tarea de conducción.

En nuestro país ya se encuentran en el mercado automóviles con diferentes grados de autonomía, algunos de los cuales alcanzan hasta el nivel 3 de la definición de la SAE.

Por otro lado, importantes empresas del Sector Automotriz como Continental y Bosch así como algunas de las principales armadoras realizan diferentes proyectos de desarrollo tecnológico y aún de investigación en sus filiales mexicanas. Al mismo tiempo, Instituciones Educativas como el Instituto Politécnico Nacional y otras del Sector de la Educación Superior contemplan planes de estudios enfocados hacia el sector automotriz considerando las nuevas tecnologías de automatización. Por otro lado, el Dr. Raúl Rojas, eminente profesor mexicano de la Universidad Libre de Berlín (ULB) y pionero en el área de Vehículos Autónomos a nivel tanto de investigación como industrial ha venido colaborando intensamente con el IPN para que en México se desarrolle tan importante área de trabajo. Esa así que ha venido donando, con el concurso de importantes empresas alemanas del sector automotriz y de la embajada alemana en México, modelos a escala para que una treintena de grupos de trabajo de diversas Instituciones de Educación Superior e Investigación inicien trabajos sobre las temáticas de interés para los vehículos autónomos. Además, ha catalizado la donación al IPN de una docena de vehículos autónomos Smart-Mercedes Benz completamente equipados e instrumentados, a fin de reforzar la infraestructura disponible en la Academia para la investigación y desarrollo tecnológico en el área de los vehículos autónomos. Es en este contexto que nuestro Laboratorio recibió en donación uno de los vehículos a escala de la ULB, lo cual motiva la realización del presente proyecto de tesis, cuyos objetivos se presentan a continuación.

1.1. Objetivos y organización de la tesis

Desde 2016 se han realizando en México competiciones con los vehículos autónomos de la Universidad Libre de Berlín; primero organizadas por el Instituto Politécnico Nacional y más tarde por la Federación Mexicana de Robótica; como parte del Torneo Mexicano de Robótica que se realiza anualmente.

Han competido muchas de las instituciones que cuentan con dicho vehículo (una treintena hasta la fecha incluyendo nuestro Laboratorio). Así, el objetivo de la tesis es controlar la dirección del vehículo con base en información visual sobre el posicionamiento relativo del vehículo dentro de su carril, a fin de circular de manera autónoma en la carretera de las competiciones del TMR con precisión y a la más alta velocidad posible. La idea es poder ganar la edición 2020 de dicha competición. En la figura 1.1 se muestra un diagrama de la carretera utilizada en estas competiciones; carretera formada por dos carriles delimitados por dos líneas blancas continuas, cuya mediana es una línea blanca discontinua. Este primer capítulo se complementa a continuación con una breve historia de los vehículos autónomos y con una sección dedicada a la presentación del estado del arte sobre el control de esta clase de vehículos. En el segundo capítulo se hace una descripción detallada del vehículo AutoNOMOS de la Universidad Libre de Berlín, el cual es utilizado en la parte experimental de este trabajo. El Capítulo 3 está dedicado a la presentación del sistema de visión desarrollado para lograr la autonomía del AutoNOMOS, mientras que el Capítulo 4 se destina a la presentación tanto de los modelos obtenidos como a la presentación de los diferentes esquemas de control desarrollados e implementados. Por su parte, el Capítulo 5 está dedicado a la presentación de la fase experimental del proyecto, tanto a la realizada con base en simulaciones bajo ROS como a la que se hizo directamente con el vehículo AutoNOMOS; en este capítulo se muestran, además, los resultados experimentales obtenidos en ambos casos con los diferentes controladores implementados. Finalmente, el Capítulo 6 muestra las conclusiones del trabajo desarrollado, algunos comentarios finales y propone el trabajo que podría desarrollarse en un futuro como continuación de este proyecto.

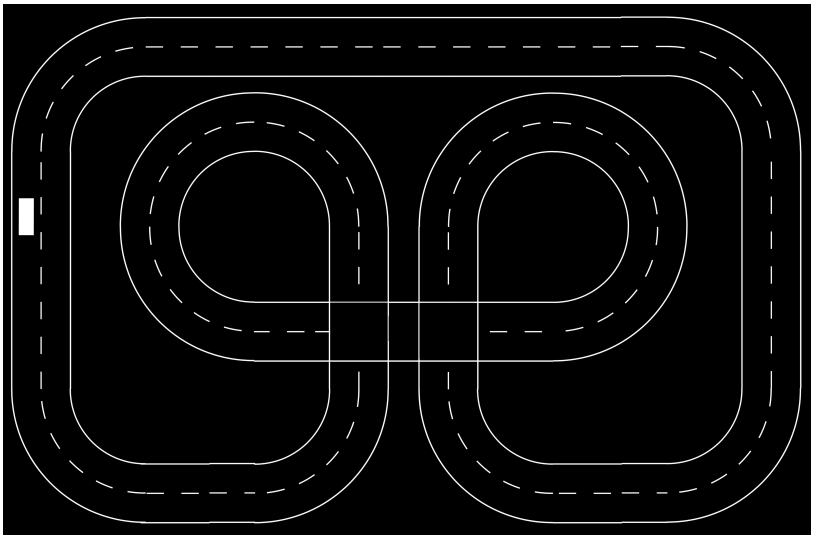

Figura 1.1: Pista utilizada en la competencia del Torneo Mexicano de Robótica [5].

1.2. Breve historia de los vehículos autónomos

Los primeros esfuerzos por desarrollar un vehículo autónomo se llevaron a cabo por Ernst Dickmanns y su equipo de la Bundeswehr University Munich [6,7]. En 1987 su vehículo "VaMoRs" (una Van equipada con una computadora y sistemas de visión) pudo conducirse por una carretera sin obtáculos por más de 20 km a una velocidad de 96 $\frac{km}{h}$. Esto dio comienzo al proyecto Europeo EUREKA PROMETHEUS (*PrograMme for a European Traffic of Highest Efficiency and Unprecedented Safety*); el cual fue un plan de desarrollo e investigación en el campo de los vehículos sin conductores.

Con un presupuesto de €749,000,000 y la participación de varias universidades y empresas Europeas, el proyecto culminó con la creación de los vehículos robóticos "VaMP" y "VITA-2"; los cuales recorrieron más de 1000 km sobre una carretera de París, con tránsito normal, a una velocidad por encima de los 130 $\frac{km}{h}$ [8].

Figura 1.2: Vehículo autónomo "VaMoRs" desarrollado por Ernst Dickmanns y su equipo [6].

En 2003 la DARPA (por las siglas en inglés de *Defense Advanced Research Projects Administration*) de Estados Unidos anunció el concurso *Grand Challenge* [6, 9]. La meta de este concurso era desarrollar un vehículo autónomo capaz de recorrer 142 millas por un camino en el desierto y evitar obstáculos. Se ofreció un millón de dólares al ganador. El concurso se llevó a cabo el 13 de Marzo de 2004 en el desierto de Mojave, California y aunque docenas de grupos de investigación participaron, ningún equipo pudo completar el recorrido; todos los vehículos chocaron después de recorrer algunas millas. El concurso se repitió el 8 de octubre de 2005, a la pista se le agregaron más curvas y obstáculos; y en esta vez cinco de los veintitrés participantes lograron completar el recorrido sin sufrir daños [1, 9]. El vehículo ganador se llamaba "Stanley" y fue desarrollado por la universidad de Stanford; este se muestra en la figura 1.3.

Figura 1.3: (a) El vehículo autónomo "Stanley" completó el recorrido en 6 h 53 min y 58 s. (b) El vehículo fue premiado por el director de DARPA, el Dr. Tony Tether [9].

Para el año 2007 DARPA organizó el *Urban Challenge* en el que los vehículos autónomos debían superar varias pruebas, ahora en un ambiente urbano simulado. De los ochenta y nueve equipos candidatos, treinta y cinco fueron invitados al evento; de ellos once se seleccionaron para competir en el evento final y sólo seis terminaron la prueba [6, 10]. El vehículo ganador se llamaba "Boss" y fue desarrollado un equipo compuesto de varios investigadores de varias instituciones y empresas como: La Universidad de Carnegie Mellon, General Motors, Caterpillar, Continental e Intel. La figura 1.4 muestra al vehículo "Boss".

Figura 1.4: Chevi Tahoe conocida como "Boss". Ganó el concurso *Urban Challenge* al conducirse de forma segura a 48 $\frac{km}{h}$ por un camino con tránsito [10].

Entre los participantes de esta competición se encontraba el equipo de Berlin; conformado por estudiantes e investigadores de la *Freie Universität Berlin*, la *Rice University*, y la *Fraunhofer Society*. El equipo estuvo liderado por el Prof. Javier Rojo y el Prof. Raúl Rojas; y juntos desarrollaron el vehículo "The Spirit of Berlin" [11]. Desde entonces, el equipo Autonomos de Raúl Rojas se ha dedicado a desarrollar automóviles autónomos; uno de los cuales ha sido el vehículo "AutoNOMOS". Este vehículo es un Volkswagen Passat Variant 3c modificado y equipado con siete escáneres láser, nueve cámaras de vídeo, siete radares y una unidad GPS de gran precisión.

En 2011 este vehículo se puso a prueba y recorrió con éxito más de 300 km entre Berlín y Leipzig, Alemania. Posteriormente, en octubre de 2015, el vehículo recorrió 2400 km en 25 horas; desde la ciudad fronteriza de Nogales, a través del desierto de Sonora, recorriendo la costa oeste de México hasta Guadalajara y luego a su destino final en el Instituto Politécnico Nacional en la Ciudad de México. Aunque el "AutoNOMOS" era capaz de alcanzar una velocidad de 140 km/h; solo tenía permitido viajar a 55 km/h entro de las ciudades y a 120 km en carretera [12, 13].

Figura 1.5: El vehículo "The Spirit of Berlin" es un Dodge Grand Caravan 2000 modificado; desarrollado por el equipo de Berlín para participar en el concurso *Urban Challenge* [11].

Figura 1.6: Vehículo "AutoNOMOS" el cual se condujo con éxito desde Nogales hasta la Ciudad de México. [11].

Recientemente y con la finalidad de iniciar trabajos de investigación en el área de los automóviles autónomos en México, el Dr. Raúl Rojas donó, a través de la Embajada Alemana en México y del Instituto Politécnico Nacional, una treintena de vehículos autónomos a escala desarrollados por la *Freie Universität Berlin*. Estos se repartieron a varios grupos de investigación, tanto del IPN como de otras instituciones, tanto públicas y como privadas. Dada la gran importancia del tema, la Federación Mexicana de Robótica organiza cada año un torneo con estos vehículos sobre un circuito típico [5].

1.3. Estado del arte

El problema de la conducción autónoma se suele separar en otros más específicos: Planeación de Trayectoria se refiere al problema de trazar una trayectoria adecuada para el vehículo, que conecte al punto de origen con el destino, que evite obstáculos y que sea óptima respecto a algún criterio (menor tiempo posible, recorrido más corto, consumo de energía mínimo, etc.). Seguimiento de trayectoria se refiere a las técnicas de control aplicadas para mantener al vehículo conduciéndose por una trayectoria de referencia dada. Este problema engloba las técnicas de control de dirección del vehículo (*steering*) y de velocidad de crucero. SLAM (por las siglas en inglés de *Simultaneous Localization an Mapping*) es una técnica con la cual un robot móvil puede mapear el medio que lo rodea y auto localizarse en dicho mapa al mismo tiempo. Es una técnica requerida si el vehículo no cuenta con un mapa de la zona donde esta. Los problemas de Visión Artificial son los relacionados con el sensado del medio que rodea al auto para que este reconozca el camino, señalizaciones, obstáculos, etc. La conducción autónoma también requiere la realización de maniobras concretas como la evasión de obstáculos, rebase de otros vehículos, estacionamiento, entre otros.

Se han propuesto muchas técnicas para resolver cada uno de los problemas mencionados pero en esta tesis se abordará el problema del seguimiento de trayectorias basado en información visual. En la literatura se suelen referir a este problema con las palabras clave: *Path Traking*, *Lane keeping*, *Longitudinal and lateral control of a car-like robot*, *Steering control of a autonomous vehicle*, *Control of an AGV (Autonomous Ground Vehicle)*, etc. Los controladores utilizados para resolver este problema pueden diseñarse con base en un modelo matemático y optimizando algún criterio; o alternativamente, también existen los "métodos inteligentes", en los cuales se utilizan controladores difusos o redes neuronales para conducir; sin lanecesidad de plantear un modelo matemático.

1.3.1. Controladores de la dirección basados en modelos matemáticos

Como se mencionó anteriormente, seguimiento de trayectoria se refiere al problema de mantener al vehículo siguiendo una trayectoria global preestablecida. Esto implica que el vehículo sigua una consigna de posición y orientación; actuando sobre la velocidad y el ángulo de dirección del mismo [14, 15]. La gran mayoría de los autores usan el modelo de la bicicleta para representar un vehículo tipo ackerman y dependiento del tipo modelo que usen se pueden clasificar en: geométricos, cinemáticos y dinámicos [15].

Modelo de la bicicleta de un automóvil

El mecanismo de dirección de Ackerman es el arquetipo que sigue la mayoría de los vehículos terrestres ya que este permite el cambio de dirección de un modo suave durante la conducción [16].

Este tipo de vehículos tiene un espacio de configuración $q = (x, y, \theta) \in \mathcal{C}$ con $\mathcal{C} \subset SE(2)$; esto debido a que tiene tres grados de libertad (dos coordenadas de posición y un ángulo de orientación). Ya que sólo cuenta con dos actuadores (tracción de las ruedas y ángulo de dirección) se dice que es subactuado. Mas adelante se demostrará que también es no-holonómico; esto debido a que no puede cambiar de orientación libremente; sin cambiar de posición.

Por su parte, el modelo de la bicicleta es una simplificación del mecanismo de dirección de Ackerman. La figura 1.7 muestra cómo se sustituye un vehículo de cuatro ruedas (en gris) por otro vehículo con una rueda trasera fija y una rueda frontal móvil (en negro) [16]. Los ejes de estas ruedas (lineas punteadas) se intersectan en un punto conocido como centro de rotación instantáneo. El radio de giro de la rueda trasera y delantera es R_1 y R_2 respectivamente. El eje x_V del marco referencial de navegación $\{V\}$, muestra la dirección de avance del auto, que se mueve a una velocidad v. γ denota el ángulo de dirección de la llanta frontal, medida desde el eje longitudinal del vehículo; mientras que θ (ángulo de guiñada) representa la orientación del vehículo, respecto al referencial global $\{O\}$. L es la distancia de separación entre los ejes de las ruedas traseras y las delanteras.

Así pues, la relación entre el ángulo de dirección y el radio de giro R_1, está dada por:

$$R_1 = \frac{L}{\tan(\gamma)} \tag{1.1}$$

Obsérvese que $R_2 > R_1$; esto significa que la rueda delantera sigue una trayectoria más larga y por lo tanto gira más rápido que la llanta trasera.

Por otro lado, el modelo no está definido cuando $\gamma = \frac{\pi}{2}$ ya que esto implicaría que las ruedas delanteras son ortogonales a las traseras y esto haría imposible moverse; más aún, el ángulo de dirección γ está limitado mecánicamente y su valor máximo implica el radio de giro R_1 más pequeño que se puede alcanzar.

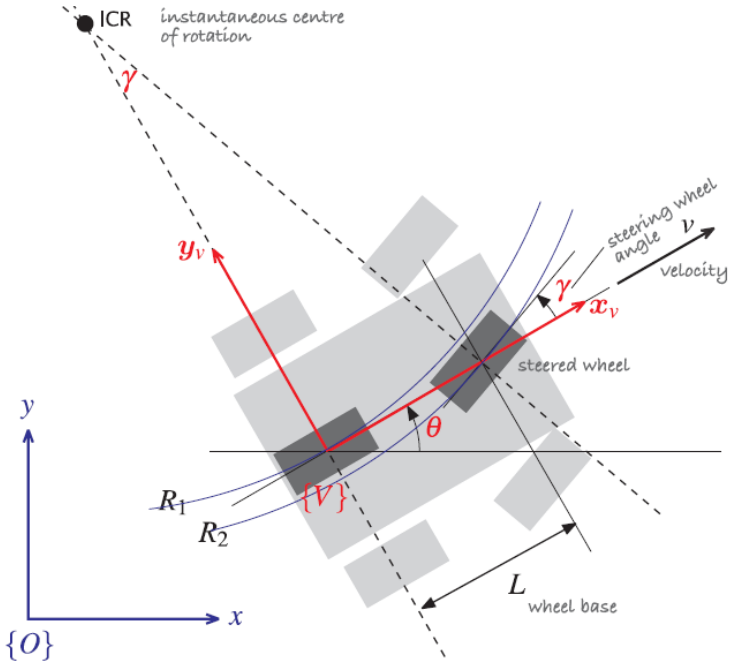

Figura 1.7: Modelo de la bicicleta de un automóvil. Se pueden sustituir las cuatro ruedas del vehículo (en gris) por dos ruedas virtuales (en negro) cuyos ejes (líneas punteadas) se intersectan en un punto conocido como centro de rotación instantaneo [16].

Modelos geométricos

El modelo de búsqueda geométrica pura reportado en [15, 17], se basa en calcular la curvatura del arco que conecta el eje trasero con un punto meta de la trayectoria de referencia. La figura 1.8 muestra cómo el auto se mueve al punto (x, y) con un radio de giro r. l es la distancia entre el auto y el punto meta.

Entonces:

$$\begin{aligned} x + d = r \quad &\Longrightarrow \quad d = r - x \\ d^2 + y^2 = r^2 \quad &\Longrightarrow \quad (r - x)^2 + y^2 = r^2 \\ &\Longrightarrow \quad r = \frac{x^2 + y^2}{2x} \\ &\Longrightarrow \quad r = \frac{l^2}{2x} \end{aligned}$$

Sustituyendo la ecuación 1.1 en la expresión anterior y resolviendo para γ obtenemos:

$$\gamma = \arctan\left(\frac{2L}{l^2}x\right) \tag{1.2}$$

Por lo tanto, la ley de control de la ecuación 1.2, queda completamente determinada midiendo el error de posición lateral x y usando la distancia l como una ganancia que se tiene que sintonizar. En [17] se reporta que se usó este controlador para conducir el "NavLab" a una velocidad de 8 $\frac{m}{s}$ y se obtuvo un buen desempeño.

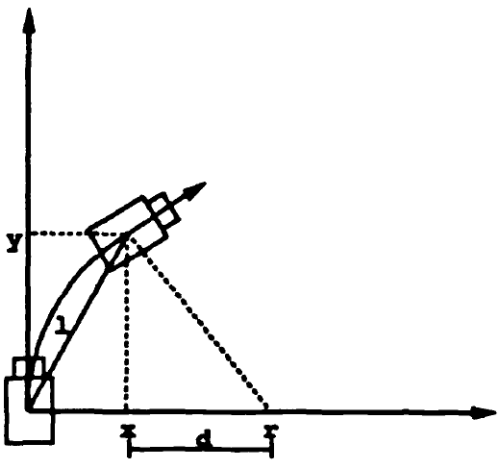

Figura 1.8: Modelo de búsqueda geométrica pura para controlar la dirección [17].

Otro método geométrico muy recurrente es el usado por el vehículo ganador del DARPA *Grand Challenge* del 2005; por lo que es conocido como el método de Stanley [9, 15, 18]. En la figura 1.9 se define el error lateral x como la distancia entre el centro del eje frontal del auto y el punto más cercano a la trayectoria de referencia; mientras que ψ representa el error de orientación, el cual es el ángulo formado entre el vector de avance y la recta tangente a la curva de referencia. Este método propone como ley de control de la orientación:

$$\delta(t) = \psi(t) + \arctan\left(\frac{kx(t)}{u(t)}\right) \tag{1.3}$$

Donde $u(t)$ es la velocidad del auto y k es un parámetro que se sintoniza.

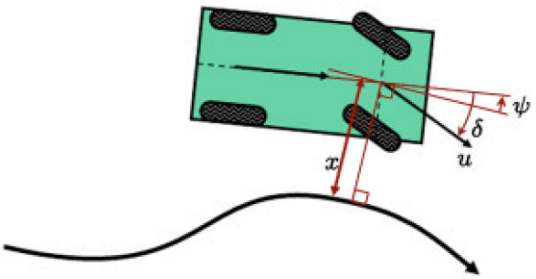

Figura 1.9: Modelo de Stanley para controlar la dirección [9].

En [9] se reporta que el controlador se ha mantenido estable conduciendo al vehículo Stanley en pavimento, barro profundo, charcos y en caminos con curvas muy pronunciadas que probocan derrape. Esto coincide con [15] en donde se concluye que el desempeño del controlador mejora conforme aumenta el parámetro k, hasta que alcanza un límite superior y se pierde la estabilidad.

Modelos cinemáticos

Del modelo de la bicicleta de la figura 1.7 podemos observar que, desde el referencial de navegación $\{V\}$, la velocidad del vehículo está dada por:

$$^{V}\dot{x} = v, \qquad ^{V}\dot{y} = 0 \tag{1.4}$$

La igualdad a cero de la componente y se debe a que el mecanismo de dirección no puede hacer que el vehículo se deslice hacia los lados. Por otro lado, la orientación del vehículo va a estar definida por:

$$\dot{\theta} = \frac{v}{R_1} \tag{1.5}$$

Si sustituimos en 1.5, la ecuación 1.1, y la combinamos con la posición del vehículo desde el referencial del mundo $(v\cos(\theta), v\sin(\theta))$; obtenemos el modelo cinemático [15, 16, 19]:

$$\dot{x} = v\cos(\theta) \tag{1.6}$$

$$\dot{y} = v\sin(\theta) \tag{1.7}$$

$$\dot{\theta} = \frac{v}{L}\tan(\gamma) \tag{1.8}$$

Este modelo describe las componentes de la velocidad del vehículo pero no las fuerzas y pares que rigen el movimiento del mismo. La variable $\dot{\theta}$ se conoce como, tasa de giro o tasa de guiñada y se puede medir con un giroscopio. De lo anterior se puede deducir que:

$$\dot{y}\cos(\theta) - \dot{x}\sin(\theta) = 0 \tag{1.9}$$

La ecuación 1.9 es una restricción que hace al vehículo no holonómico; es decir, el número total de grados de libertad controlables es menor que el número de grados de libertad (x, y, θ). También podemos observar que si $v = 0$ entonces $\dot{\theta} = 0$; por lo tanto, no es posible cambiar la orientación del vehículo sin moverse.

La principal suposición hecha en este modelo es que el vector velocidad $\overrightarrow{v}$ apunta en la misma dirección que la orientación θ del vehículo [19]. Esta suposición es razonable cuando la fuerza lateral de las llantas no influye en la dinámica del sistema.

Para implementar la ley de control a este modelo primero se hace el cambio de variables a las coordenadas polares:

$$\rho = \sqrt{\Delta_x^2 + \Delta_y^2} \tag{1.10}$$

$$\alpha = \arctan\left(\frac{\Delta_y}{\Delta_x}\right) - \theta \tag{1.11}$$

$$\beta = -\theta - \alpha \tag{1.12}$$

Tal y como se muestra en la figura 1.10, dada una pose de referencia G, Δ_x y Δ_y denotan la magnitud de las componentes del vector $\overrightarrow{\rho}$; el cual apunta a la posición deseada del auto. Con esta transformación se obtiene el sistema:

$$\dot{\rho} = -v\cos(\alpha) \tag{1.13}$$

$$\dot{\alpha} = v\frac{\sin(\alpha)}{\rho} - \gamma \tag{1.14}$$

$$\dot{\beta} = -v\frac{\sin(\alpha)}{\rho} \tag{1.15}$$

Así pues, la ley de control para la velocidad y el ángulo de dirección propuesta es:

$$v = k_\rho \rho \tag{1.16}$$

$$\gamma = k_\alpha \alpha + k_\beta \beta \tag{1.17}$$

Mientras que los términos $k_\rho\rho$ y $k_\alpha\alpha$ conducen al vehículo hacia la posición de referencia; el término $k_\beta\beta$ orienta el vehículo el ángulo deseado.

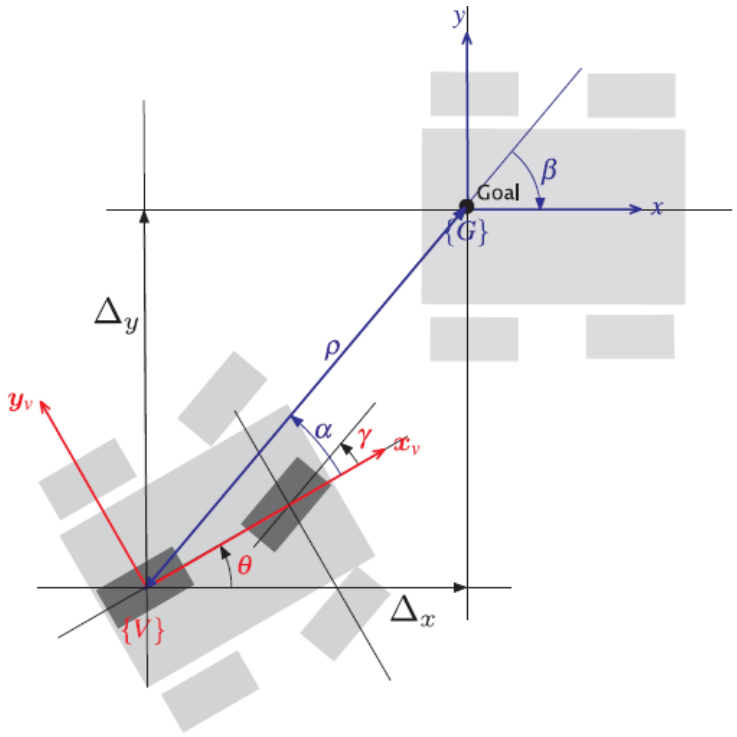

Figura 1.10: Modelo de cinemático para controlar la dirección [16].

Este controlador funciona cuando la posición de consigna queda al frente del vehículo, es decir $\alpha \in (-\frac{\pi}{2}, \frac{\pi}{2}]$; en otro caso, se usa como ley de control:

$$v = -k_\rho \rho \tag{1.18}$$

$$\gamma = -(k_\alpha \alpha + k_\beta \beta) \tag{1.19}$$

Este controlador lleva al auto a un único punto de equilibrio y es estable [16]. La gráfica de la figura 1.11 muestra la trayectoria del auto hasta la pose $(5, 5, \frac{\pi}{2})$; para diferentes condiciones iniciales.

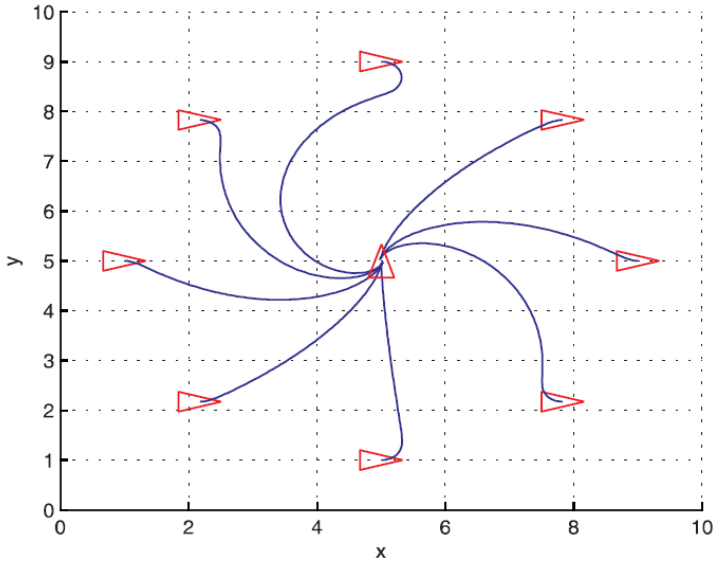

Figura 1.11: Simulación de la trayectoria de un auto que parte desde diferentes poses iniciales a la pose $(5, 5, \frac{\pi}{2})$ [16].

Alternativamente, en [20], se usa un modelo cinemático no lineal, con variables de estado observables desde el referencial de navegación. En la figura 1.12 se definen las variables de estado:

$$d_e = (y - y_d + L_h \sin(\theta)) \cos(\theta_d) - (x - x_d + L_h \cos(\theta)) \sin(\theta_d) \tag{1.20}$$

$$\theta_e = \theta - \theta_d \tag{1.21}$$

En donde L_h es la distancia entre el referencial de navegación del vehículo y el punto de observación; el cual es el punto desde donde un sensor, como una cámara, mide la distancia d_e a la que está el auto de la trayectoria de referencia. θ_d es el ángulo de referencia definido entre la recta tangente a un punto de la trayectoria de referencia y el eje x. θ_e es el error de orientación del vehículo.

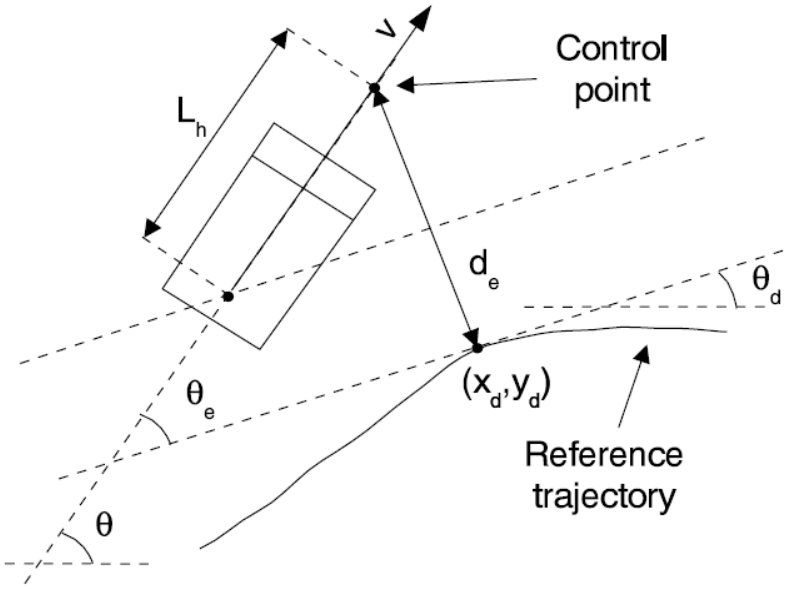

Figura 1.12: Modelo de cinemático no lineal para controlar la dirección [20].

La distancia L_h es una función de la velocidad; conforme aumenta v, el factor de amortiguamiento cambia y hace inestable al sistema en lazo cerrado. Para evitar esto se incrementa el valor de L_h hasta alcanzar un máximo. Empiricamente la relación matemática está dada por [20]:

$$L_h = \begin{cases} L_{min} & \text{si } v < v_{min} \\ vt & \text{si } v_{min} < v < v_{max} \\ L_{max} & \text{si } v > v_{min} \end{cases}$$

Al parámetro t se le conoce como *look ahead time* y se determina experimentalmente junto con v_{min} y v_{max}.

Considerando a x_d, y_d y θ_d como constantes calculadas por el sistema de visión en cada iteración, derivando las ecuaciones 1.20 y 1.21; y sustituyendo en el resultado, las ecuaciones 1.6, 1.7 y 1.8; obtenemos el modelo no lineal:

$$\dot{d}_e = v\sin(\theta_e) + \frac{vL_h}{L}\cos(\theta_e)\tan(\phi) \tag{1.22}$$

$$\dot{\theta}_e = \frac{v\tan(\phi)}{L} \tag{1.23}$$

El controlador propuesto en [20] usa el enfoque Algebro Diferencial de los Sistemas No Lineales para establecer la ley de control:

$$\phi = \arctan\left(\frac{-L\sin(\theta_e)\cos^3(\theta_e)(K_d\tan(\theta_e) + K_p d_e)}{\sin(\theta_e) + L_h\cos^4(\theta_e)(K_d\tan(\theta_e) + K_p d_e)}\right) \tag{1.24}$$

Con este controlador se garantiza que d_e y θ_e tienden a cero en estado estacionario; siempre y cuando $v > 0$ y $\theta_e \in (-\pi/2, \pi/2)$. Este controlador se implementó en el automóvil "Babieca"; el cual esta equipado con una cámara RGB para medir los errores d_e y θ_e, un tacómetro para medir la velocidad, una computadora Pentium y actuadores para manipular el acelerador y el ángulo de dirección del mismo. Dicho automóvil se muestra en 1.13.

Figura 1.13: Vehículo "Babieca" usado en [20].

De este modo se pudo conducir al vehículo "Babieca" a una velocidad máxima de 50 $\frac{Km}{h}$, con un error de posición de $\pm 25\ cm$ y un error de orientación de 1 deg en estado estacionario [20]. También se comparó su desempeño con el de un conductor humano. Conduciendo el auto a 20 $\frac{Km}{h}$ se observó que el conductor experto alcanza el estado estacionario más rápido; pero el error de posición es menor usando el controlador automático.

Modelos dinámicos

El modelo dinámico de la bicicleta toma en cuenta las fuerzas que aparecen en los neumáticos cuando el vehículo se mueve a grandes velocidades y es de los más utilizados para diseñar controladores para automóviles reales [14, 19, 21]. Este modelo se muestra en la figura 1.14 y los parámetros que se ilustran son: CG que es el centro de masa del automóvil y es el lugar donde se pone el referencial de navegación, cuyo eje x apunta a la dirección de avance y el eje y apunta al centro de giro instantáneo; l_r y l_f son las distancias que hay entre el centro de masa y los ejes trasero y delantero respectivamente; δ es el ángulo de dirección; $\overrightarrow{V}$ es el vector velocidad y β es el ángulo de deslizamiento que se forma entre $\overrightarrow{V}$ y el eje longitudinal del auto. Nótese que las fuerzas $\overrightarrow{F_f}$ y $\overrightarrow{F_r}$, que aparecen en los neumáticos, son paralelas a los ejes de las llantas. Las componentes en y de estas fuerzas inducen una fuerza lateral en el automóvil representada por la ecuación:

$$ma_y = F_{yf} + F_{yr} \tag{1.25}$$

Desde algún referencial fijo, podemos definir un ángulo de orientación ψ (ángulo de guiñada) para el auto; entonces podemos reescribir la aceleración centrípeta como:

$$a_y = \ddot{y} + V_x\dot{\psi} \tag{1.26}$$

Sustituyendo 1.35 en 1.34 se obtiene:

$$m(\ddot{y} + V_x\dot{\psi}) = F_{yf} + F_{yr} \tag{1.27}$$

Por otro lado, los resultados experimentales demuestran que las fuerzas laterales sobre los neumáticos se pueden modelar como [19]:

$$F_{yf} = 2C_{\alpha f}(\delta - \theta_{Vf}) \quad \text{para la llanta frontal} \tag{1.28}$$

$$F_{yr} = 2C_{\alpha r}(-\theta_{Vr}) \quad \text{para la llanta trasera} \tag{1.29}$$

Donde a las constantes de proporcionalidad $C_{\alpha f}$ y $C_{\alpha r}$ se conocen como la rigidez de las curvas de cada neumático y a los ángulos θ_{Vf} y θ_{Vr} se les llama ángulos de deslizamiento. Estos ángulos calculan con las ecuaciones:

$$\tan(\theta_{Vf}) = \frac{V_y + l_f\dot{\psi}}{V_x} \qquad \tan(\theta_{Vr}) = \frac{V_y - l_r\dot{\psi}}{V_x}$$

que para ángulos pequeños se puede usar la aproximación $\tan(x) \approx x$ en conjunto con $V_y = \dot{y}$ para obtener:

$$\theta_{Vf} = \frac{\dot{y} + l_f\dot{\psi}}{V_x} \qquad \theta_{Vr} = \frac{\dot{y} - l_r\dot{\psi}}{V_x} \tag{1.30}$$

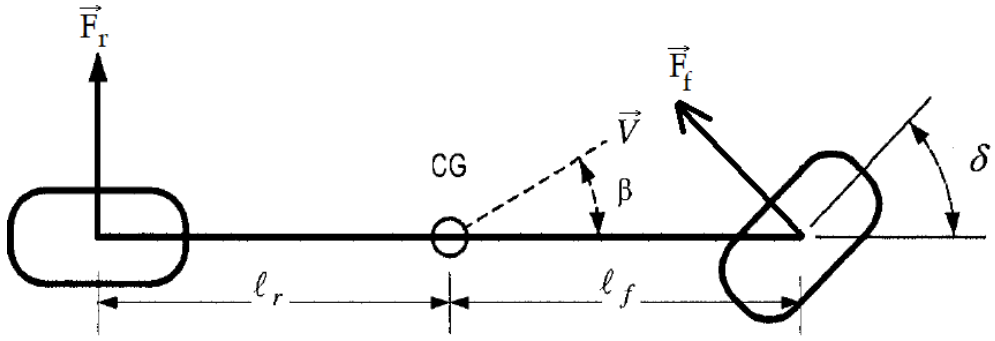

Figura 1.14: Parámetros usados en el modelo dinámico de la bicicleta [19].

Entonces, sustituyendo las ecuaciones 1.28, 1.29 y 1.30 en la ecuación 1.27 y despejando $\ddot{y}$ resulta:

$$\ddot{y} = -\frac{2(C_{\alpha f} + C_{\alpha r})}{mV_x}\dot{y} - V_x\dot{\psi} - \frac{2(C_{\alpha f}l_f - C_{\alpha r}l_r)}{mV_x}\dot{\psi} + \frac{2C_{\alpha f}}{m}\delta \tag{1.31}$$

En cambio, los pares alrededor del eje z del vehículo, son representados por:

$$I_z\ddot{\psi} = F_{yf}l_f - F_{yr}l_r \tag{1.32}$$

donde I_z es el momento de inercia del vehículo. Sustituyendo las ecuaciones 1.28, 1.29 y 1.30 en 1.32 y despejando $\ddot{\psi}$, resulta:

$$\ddot{\psi} = -\frac{2(C_{\alpha f}l_f - C_{\alpha r}l_r)}{I_zV_x}\dot{y} - \frac{2(C_{\alpha f}l_f^2 + C_{\alpha r}l_r^2)}{I_zV_x}\dot{\psi} + \frac{2l_fC_{\alpha f}}{I_z}\delta \tag{1.33}$$

Si ahora se define un ángulo de orientación deseado ψ_{des}, este se puede relacionar con la velocidad V_x usando la ecuación $\dot{\psi}_{des} = \frac{V_x}{R_{des}}$; siendo R_{des} el radio de giro deseado. Entonces la aceleración angular deseada es: $V_x\dot{\psi}_{des} = \frac{V_x^2}{R_{des}}$. Suponiendo que la velocidad V_x es constante, se pueden definir los errores:

$$\dot{e_1} = \dot{y} + V_x(\psi - \psi_{des}) \tag{1.34}$$

$$e_2 = \psi - \psi_{des} \tag{1.35}$$

Por lo que

$$\begin{aligned}\ddot{e}_1 &= \ddot{y} + V_x\dot{\psi} - \frac{V_x^2}{R_{des}} \\ &= \ddot{y} + V_x(\dot{\psi} - \dot{\psi}_{des}) \\ &= \ddot{y} + V_x\dot{e}_2\end{aligned}$$

La restricción de que la velocidad V_x sea constante se debe a que, al integrar esta última ecuación, se obtiene el estado: $\dot{e}_1 = \dot{y} + \int V_x\dot{e}_2dt$; el cual es no lineal.

Sustituyendo los errores 1.34 y 1.35 en las ecuaciones 1.31 y 1.33 resulta:

$$\ddot{e}_1 = -\frac{2(C_{\alpha f}+C_{\alpha r})}{mV_x}\dot{e}_1 + \frac{2(C_{\alpha f}+C_{\alpha r})}{m}e_2 - \frac{2(C_{\alpha f}l_f - C_{\alpha r}l_r)}{mV_x}\dot{e}_2 + \frac{2C_{\alpha f}}{m}\delta - \Big(\frac{2(C_{\alpha f}l_f - C_{\alpha r}l_r)}{mV_x} + V_x\Big)\dot{\psi}_{des} \quad (1.36)$$

$$\ddot{e}_2 = -\frac{2(C_{\alpha f}l_f - C_{\alpha r}l_r)}{I_z V_x}\dot{e}_1 + \frac{2(C_{\alpha f}l_f - C_{\alpha r}l_r)}{I_z}e_2 - \frac{2(C_{\alpha f}l_f^2 + C_{\alpha r}l_r^2)}{I_z V_x}\dot{e}_2 + \frac{2l_f C_{\alpha f}}{I_z}\delta - \frac{2(C_{\alpha f}l_f^2 + C_{\alpha r}l_r^2)}{I_z V_x}\dot{\psi}_{des} \quad (1.37)$$

Estas ecuaciones se pueden reescribir en términos de: el ángulo de deslizamiento β, la tasa de cambio del ángulo de guiñada $r = \dot{\psi}$ y la variable de control que será el ángulo de dirección δ. Entonces, para ángulos pequeños de β:

$$\begin{aligned}\beta &\approx \sin(\beta)\\ &= \frac{\dot{y}}{V_x}\\ &= \frac{1}{V_x}(\dot{e}_1 - V_x e_2)\\ &= \frac{1}{V_x}\dot{e}_1 - e_2\end{aligned}$$

Definiendo a la curvatura del camino como $\rho_{des} \triangleq \frac{1}{R_{des}}$ y recordando que $\dot{\psi} = \frac{V_x}{R_{des}}$, se puede demostrar que las ecuaciones 1.36 y 1.37 se reescriben como:

$$\dot{\beta} = -\frac{2(C_{\alpha f}+C_{\alpha r})}{mV_x}\beta - \Big(\frac{2(C_{\alpha f}l_f - C_{\alpha r}l_r)}{mV_x^2} + 1\Big)r + \frac{2C_{\alpha f}}{mV_x}\delta \quad (1.38)$$

$$\dot{r} = -\frac{2(C_{\alpha f}l_f - C_{\alpha r}l_r)}{I_z}\beta - \frac{2(C_{\alpha f}l_f^2 + C_{\alpha r}l_r^2)}{I_z V_x}r + \frac{2C_{\alpha f}l_f}{I_z}\delta + V_x\dot{\rho}_{des} \quad (1.39)$$

En la figura 1.15 se representan algunas de estas cantidades. Adicionalmente, a la distancia entre la trayectoria seguida y el sensor que está sobre el eje longitudinal del

vehículo, se le llama error de posición d y su dinámica está dada por:

$$\dot{d} = V_x\beta + l_s r + V_x\psi \tag{1.40}$$

Siendo l_s es la distancia entre el centro de masa y el sensor de posición.

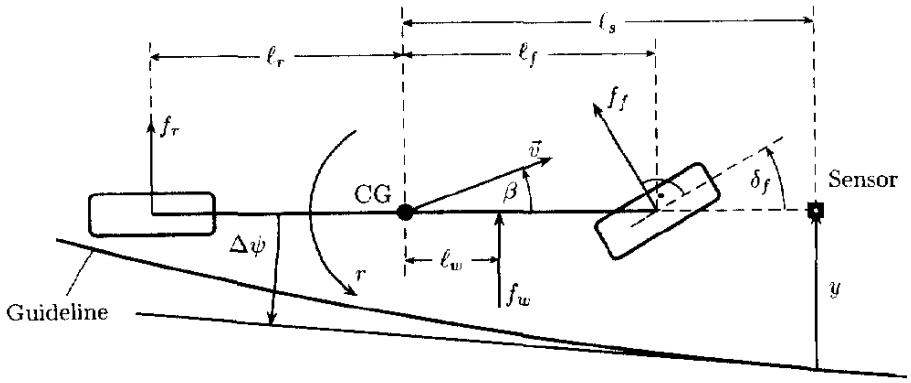

Figura 1.15: Modelo dinámico de la bicicleta [14].

Finalmente, el modelo dinámico de un automóvil queda definido por los estados $x = (\beta \quad r \quad \psi \quad d \quad \delta_f)^T$ [14]:

$$\dot{x} = \begin{pmatrix} a_{11} & a_{12} & 0 & 0 & a_{15} \\ a_{21} & a_{22} & 0 & 0 & a_{25} \\ 0 & 1 & 0 & 0 & 0 \\ V_x & l_s & V_x & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} x + \begin{pmatrix} 0 & 0 \\ V_x & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \dot{\rho}_{ref} \\ u_f \end{pmatrix} \tag{1.41}$$

$$y = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \end{pmatrix} x \tag{1.42}$$

con $y = (r \quad d)^T$ como las salidas del sistema y los parámetros:

$$
\begin{aligned}
a_{11} &= -\frac{2(C_{\alpha f} + C_{\alpha r})}{mV_x} \\
a_{12} &= -\Big(\frac{2(C_{\alpha f} l_f - C_{\alpha r} l_r)}{mV_x^2} + 1\Big) \\
a_{15} &= \frac{2C_{\alpha f}}{mV_x} \\
a_{21} &= -\frac{2(C_{\alpha f} l_f - C_{\alpha r} l_r)}{I_z} \\
a_{22} &= -\frac{2(C_{\alpha f} l_f^2 + C_{\alpha r} l_r^2)}{I_z V_x} \\
a_{25} &= \frac{2C_{\alpha f} l_f}{I_z}
\end{aligned}
$$

La salida $(r \quad y)^T$ esta compuestas por cantidades que se pueden medir con un sensor inercial y una cámara; respectivamente. De acuerdo con [21], todos los estados de este modelo son controlables con el ángulo de dirección δ; excepto cuando la velocidad V_x es cero. Por lo que la implementación del sistema requiere una velocidad mínima de conducción.

La gran mayoría de las publicaciones relacionadas con el control de la dirección de un automóbil se basan en este modelo matemático para proponer sus controladores. Por ejemplo, la ley de control empleada en las simulaciones de [14] para conducir un autobús O305 se expresa como:

$$
\delta_f = \omega_c^3 \frac{k_{DD} s^2 + k_D s + k_P + k_I/s}{(s^2 + 2D\omega_c s + \omega_c^2)(s + \omega_c)} - k_r r \tag{1.43}
$$

Con este controlador PIDD se obtuvo una velocidad máxima de $20\frac{m}{s}$ y con una tasa de ángulo de guiñada máxima de $23\frac{deg}{s}$.

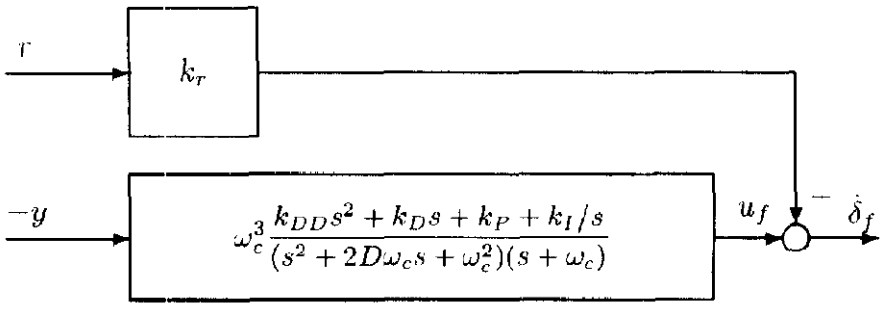

Figura 1.16: Controlador dinámico de dirección [14].

Se han utilizado muchos otros enfoques para proponer controladores de dirección. Por ejemplo en [22–30] se utiliza la teoría del *Optimal preview controller*. En este enfoque se desea minimizar un criterio de desempeño, tal y como se hace con un regulador cuadrático lineal; sin embargo, tal criterio también pondera las señales de referencia futuras que se conocen [31]. En este caso las referencias futuras son la curvatura del camino en diferentes puntos observados con el sistema de visión. Al final se obtiene una ley de control con dos componentes un término conocido como "*State Feedback*", que es igual al que se obtendría calculando las ganancias de un controlador P resolviendo la ecuación de Ricatti; y un término conocido como "*Preview Compensations*" el cual pondera las referencias futuras que se conocen.

En las referencias mencionadas se considera que el vehículo viaja a una velocidad constante, no se toman en cuenta las incertidumbres del modelo dinámico ni las variaciones de sus parámetros y solo se exponen resultados obtenidos con simuladores. En cambio en [32] se propone un modelo variante en el tiempo del vehículo y se utiliza un método de optimización robusto para proponer un controlador en la forma $LQ - H_\infty$. De este modo en [32] se extiende el esquema del *Optimal preview controller* para incluir variaciones en los parámetros e incertidumbres en el diseño del controlador; sin embargo sólo exponen resultados en simuladores y la formulación implica resolver un problema de LMI's (*Linear Matrix Inequalities*), lo cual puede ser un problema por el poder de cómputo que se requiere si se desea implementar en linea.

Otra formulación muy recurrente es la del *Predictive control theory* [33–41]. En esta metodología, el modelo matemático de la planta juega un papel medular ya que debe capturar lo mejor posible la dinámica del sistema; pero al mismo tiempo debe ser lo suficientemente simple para poderse implementar en linea. Usando este modelo, en el instante t se calcula la salida de la planta $y(t)$ para momentos futuros $t+k$. A estas salidas se les conoce como horizonte de predicción $\{y_i\}_{i=t+1}^{t+k}$. Después se define una trayectoria de referencia $\{r_i\}_{i=t+1}^{t+k}$, sobre el horizonte de predicción, hasta llegar a un punto conocido como punto de operación $w(t)$. Finalmente se calcula la ley de control, minimizando una función de costo específica que depende de los errores de control predichos; es decir, de la diferencia entre la trayectoria de referencia y el horizonte de predicción [42]. Esta metodología tiene las ventajas de ser lo suficientemente flexible para poderse combinar con otros enfoques y se pueden manejar problemas de control no lineales, variantes en el tiempo o con restricciones; sin embargo requiere el conocimiento pleno del modelo para hacer las predicciones y para sistemas variantes en el tiempo, el poder de cómputo se incrementa.

De las publicaciones mencionadas se destaca la de [37] ya que, además de proponer un controlador predictivo basado en un modelo no lineal; añade restricciones a los estados y las entradas del sistema para mantener al vehículo en su carril y evitar obstáculos; y desarrolla un algoritmo (basado en la linealización iterativa de la dinámica del vehículo) para reducir la complejidad computacional de su método. También presenta resultados experimentales favorables al conducir un automóvil Jaguar Type-S a 80 km/h, evitar obstáculos estáticos y rebasar obstáculos moviéndose a 36 km/h; en un camino con nieve. La figura 1.17 muestra la posición del auto en cinco instantes diferentes.

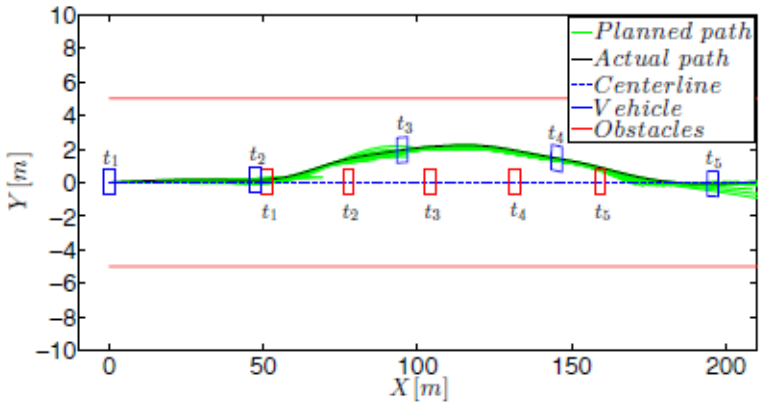

Figura 1.17: Vehículo Jaguar Type-S usado en [37]. Se muestra su posición (en azul) en cinco instantes diferentes mientras rebasa un obstáculo móvil (en rojo) que viaja a 36 km/h.

Recientemente, se han publicado trabajos con otros enfoques para resolver el problema del control lateral de un automóvil. Por ejemplo en [43], se diseña un controlador robusto usando la Teoría H_∞ el cual hace al sistema **??** y **??** asintóticamente estable y mantiene un buen desempeño en presencia de perturbaciones externas e incertidumbre en los parámetros. Para calcular las dieciséis ganancias que requiere su controlador; utiliza un algoritmo genético y la formulación del problema con LMI's. Todos los resultados que presenta están basados en CarSim-Simulink de MATLAB y muestran un buen desempeño conduciendo el auto de 18 a 30 m/s; y cuyos parámetros conocidos como rigidez en las curvas, tienen una incertidumbre del 20 %. Otro trabajo interesante es [44]; en el que se usa el esquema de Rechazo Activo de Perturbaciones para proponer un controlador que es: exponencialmente estable, robusto ante perturbaciones externas e incertidumbres en el modelo y de fácil implementación. Con este esquema se hicieron pruebas experimentales con el vehículo a escala 1:5 mostrado en la figura 1.18. Se concluyó que, comparado con un controlador PID, la ley de control de [44] mantiene al vehículo en su carril aún si existen variaciones en la velocidad y la curvatura del camino.

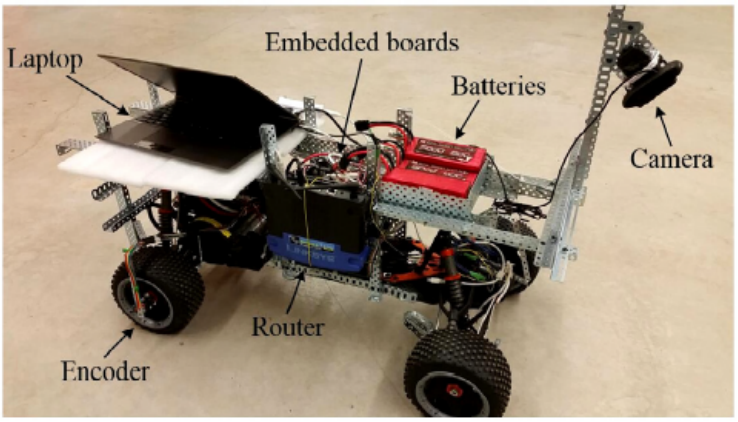

Figura 1.18: Vehículo a escala 1:5 con el que se hicieron pruebas experimentales en [44]. El auto se condujo a velocidades de 2 y 2.5 m/s.

1.3.2. Controladores inteligentes de la dirección

Además de los controladores basados en un modelo matemático también existen los métodos de control "inteligentes", los cuales no requieren un modelo matemático de la planta para controlar la salida del sistema. A continuación se describen los más utilizados que son los Controladores Difusos y las Redes Neuronales.

Controladores difusos

Existen numerosas publicaciones que utilizan los métodos del control difuso para resolver el problema de seguimiento de trayectorias de un automóvil tipo Ackerman [45–61]. En este marco teórico se busca capturar la experiencia del operador humano (en este caso un conductor), en forma de reglas lógicas que relacionan a las entradas y salidas del sistema con las acciones de control; para después calcular la señal de salida a través de un promedio ponderado.

El primer trabajo publicado en el que se usa controlador difuso para cambiar la dirección de un auto es [45].

En este reporte se utiliza el vehículo a escala 1 : 10 mostrado en la figura 1.19a, el cual esta equipado con un sensor ultrasónico para detectar obstáculos. Para conducirlo por un pasillo sin que choque con las paredes, primero se definieron cuatro variables de entrada para el sistema difuso que son: x_1, x_2, x_3, x_4; mostradas en la figura 1.19b.

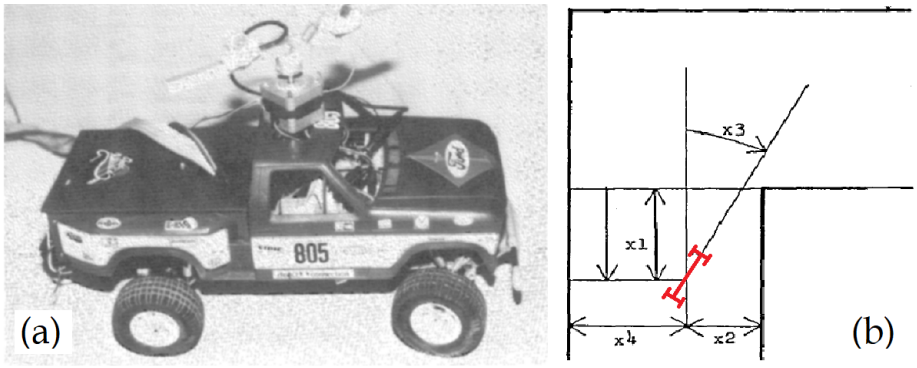

Figura 1.19: (a)Vehículo a escala utilizado en [45]. (b) Definición de las variables de entrada del sistema difuso.

El siguiente paso en el equema del control difuso es el de definir una función de pertenencia para cada señal, con sus respectivas particiones difusas. De este modo, para cada valor de la señal, este tiene asociados dos o más variables descriptivas, cuyos pesos van de 0 a 1 (variable difusa). Un ejemplo de función de pertenencia se muestra en la figura 1.21 [45] donde cada valor de x_1 tiene asociadas a las variables descriptivas: pequeño, mediano y grande; cada una con cierto peso A_1^i con $i = 1, 2, 3$.

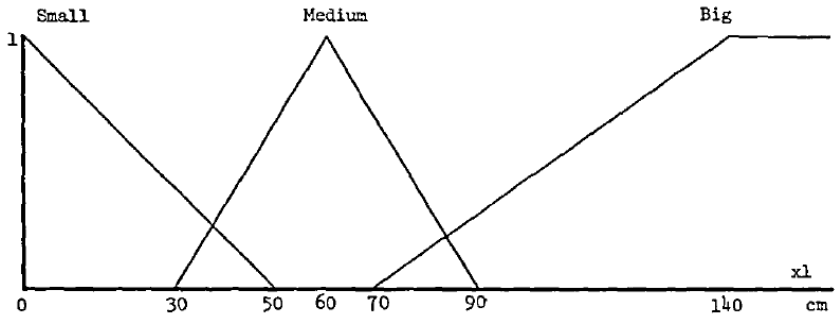

Figura 1.20: Ejemplo de una función de pertenencia y las particiones difusas que se usan para representar, con variables descriptivas, a la variable x_1 [45].

Posteriormente se definen las m reglas de control. Si se define a la señal de salida de la i-ésima regla como y^i, cada regla toma la forma:

$$R^i : \text{Si } x_1 \text{ es } A_1^i, \text{ Si } x_2 \text{ es } A_2^i, \cdots, \text{ Si } x_n \text{ es } A_n^i \Rightarrow y^i = p_0^i + p_1^i x_1 + \cdots + p_n^i x_n$$

Donde A_j^i son las variables difusas y p_j^i son coeficientes. Después, dadas las variables de entrada $x_1^0, \cdots, x_4^0$, se calcula el valor de verdad w^i de la i-ésima regla como una función de las variables difusas A_j^i. Con estos valores se calcula la señal de control y^0 a través del promedio ponderado:

$$y^0 = \frac{\sum_{i=1}^{m} w^i y^i}{\sum_{i=1}^{m} w^i}$$

Siguiendo un esquema similar, muchos otros autores han publicado metodologías de control difusas capaces de conducir automóviles reales en ambientes urbanos. Por ejemplo en [50] se utilizaron como variables de entrada del sistema difuso a: El error de orientación (diferencia entre la orientación del auto deseada y el ángulo de guiñada), el error de posición (distancia entre la posición deseada y la posición actual del auto), la distancia entre el auto y la curva más cercana y la velocidad del auto. Con estas variables se definieron 6 reglas difusas y se pudo conducir una furgoneta eléctrica del programa AUTOPÍA a velocidades constantes de entre 8 y 24 km/h por una carretera.

También llevaron a cabo otras maniobras como la conducción en rotondas y la marcha hacia atrás. Sin embargo, ni en este trabajo ni en los resultados experimentales de [47, 48, 51, 52] se utiliza información visual y la pose del vehículo se mide con dispositivos GPS. Por otro lado, de las publicaciones que combinan el control difuso con visión artificial [56–61], las que presentan resultados experimentales, lo hacen bajo condiciones muy específicas que no se asemejan a un ambiente urbano real [58–61].

Redes neuronales

Una red neuronal artificial (RNA) es un aproximador universal de funciones que consta de unidades de procesamiento llamadas neuronas. Estas adquieren y procesan la información de manera no lineal, están interconectadas y organizadas en capas; y operan en forma paralela. Una RNA es capaz de aprender asociaciones dado un conjunto de variables de entrada y su respectivo conjunto de variables de salida deseadas [62]. Debido a su robustez, adaptabilidad y fácil implementación; han sido una metodología atractiva para resolver el problema del seguimiento de trayectorias para un vehículo autónomo [63–71]. Por ejemplo, en el trabajo reportado en [63] se describe una red neuronal que utiliza como señales de entrada a la distancia entre el auto y una posición deseada; y el ángulo de orientación con el que se desea llegar allí. De la red se obtenía como salida el ángulo de orientación. En este artículo se comparan las arquitecturas de *Backpropagation* y *Functional-link* para las redes neuronales; encontrando que con la segunda se obtuvieron mejores resultados en simulación.

En trabajos más recientes se han usado redes neuronales convolucionales, las cuales con una variación de las RNA, y cuyas entradas son matrices. Su principal campo de aplicación es la Visión Artificial ya que, a partir de imágenes, son capaces de clasificar objetos, segmentar colores, reconocer patrones, entre otras cosas [72]. Este tipo de redes toma la imagen de entrada y utilizando operaciones de convolución, adquieren información que luego pasan a las demás capas.

En trabajos recientes se han utilizado redes neuronales convolucionales para obtener un ángulo de dirección a partir las imagenes adquiridas por una cámara [66,68,69]. Con este enfoque en [66] se reporta que se pudo conducir de forma autónoma al vehículo "Dave 2" de la empresa NVIDIA, a través de distintos ambientes como son: a través de la ciudad, en carretera, en caminos húmedos o con nieve e incluso en senderos sin señalizaciones.

Figura 1.21: Automóvil "Dave 2" de NVIDIA.

Capítulo 2

El vehículo AutoNOMOS

El vehículo AutoNOMOS es un automóvil tipo Ackerman a escala 1:10 desarrollado por la *Freie Universität Berlin* [73]. En este trabajo se utilizó una versión modificada del AutoNOMOS v2.0. Mide 40 cm de largo y 20 cm de ancho. Utiliza un motor sin escobillas modelo DB28M01 de 24 V para generar traccion en las 4 ruedas, un servomotor HS-645MG de 5 V para controlar el ángulo de dirección del vehículo y una batería Li-Po de 14.8 V de 4 celdas. Los sensores a bordo del vehículo son: Una cámara RGBD Intel RealSense SR300, una cámara tipo *fish-eye* ELP 1080p, un sensor de proximidad RPLIDAR A2 360 y un sensor inercial de 6 ejes o IMU (*Inertial Measurement Unit*) MPU6050. El vehículo cuenta con una computadora Odroid XU4 con el sistema operativo Ubuntu 16.04 y la distribución ROS Indigo instalada; así que, para hacer el procesamiento de los datos obtenidos por los sensores, estas mediciones se publicaron como tópicos de ROS a través de nodos programados en python y c++. Los datos generados por la IMU son publicados como un tópico de ROS, usando una placa Arduino NANO. Las figuras 2.2 y 2.3 muestran algunos elementos mencionados [73]. Para hacer la interconexión de los elementos se utilizaron un par de Hubs USB y una antena para WIFI para conectarse con otras computadoras a través de un módem; tal y como se muestra en la figura 2.1. Se utilizó un regulador de voltaje tipo *BOOST*, conectado a la batería, para generar los 5V de alimentación que necesitan los elementos lógicos a bordo.

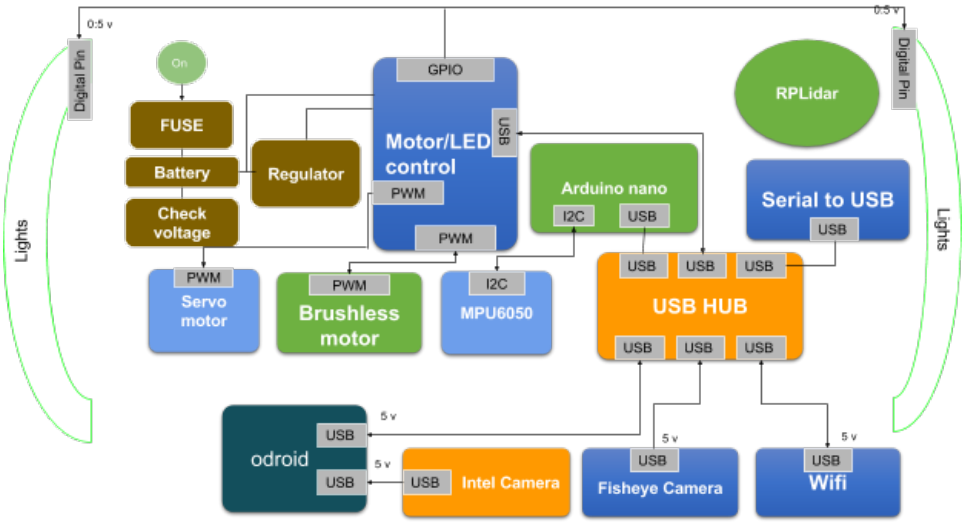

Figura 2.1: Interconexión de los elementos del AutoNOMOS [73].

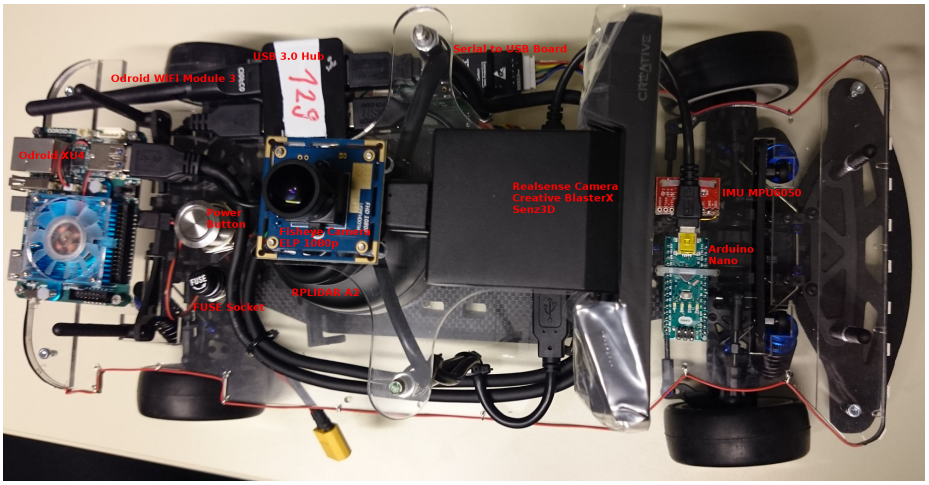

Figura 2.2: Vista desde arriba del vehículo AutoNOMOS [73].

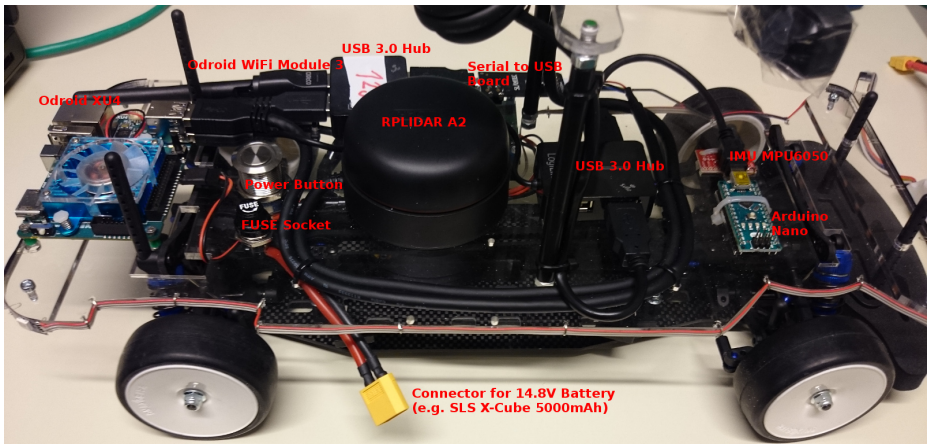

Figura 2.3: Algunos de los elementos del vehículo AutoNOMOS [73].

Para controlar la velocidad del motor sin escobillas y el servomotor, el AutoNOMOS contaba con una tarjeta desarrollada por la *Freie Universität Berlin*; sin embargo, esta se descompuso y se tuvo que diseñar un circuito conectado a un Arduino UNO R3. A continuación se presenta el diseño de este controlador.

2.1. Diseño de un controlador para los motores del AutoNOMOS

La figura 2.4 muestra un diagrama simplificado de un motor sin escobillas o motor BLDC (*Brush Less Direct Current*). Como se puede observar en este ejemplo un motor BLDC consta de un imán permanente en el rotor y al menos tres embobinados conectados en forma de estrella en el estator; así que, para transformar la energía eléctrica en energía mecánica, la corriente que atraviesa el estator induce un campo magnético el cual interactúa con el campo del iman permanente e induce un par en el rotor [74].

En este ejemplo el motor BLDC es trifásicos y la corriente debe atravesar, en cierto orden, cada una de sus fases denominadas como: A, B y C. La figura 2.4 ilustra esto con las flechas que aparecen a los lados del diagrama de estrella. En total hay 6 conmutaciones diferentes (un ciclo eléctrico) y para este modelo de motor, cada conmutación gira el rotor 60°; por lo tanto, para completar una vuelta se necesitan llevar a cabo dos ciclos eléctricos.

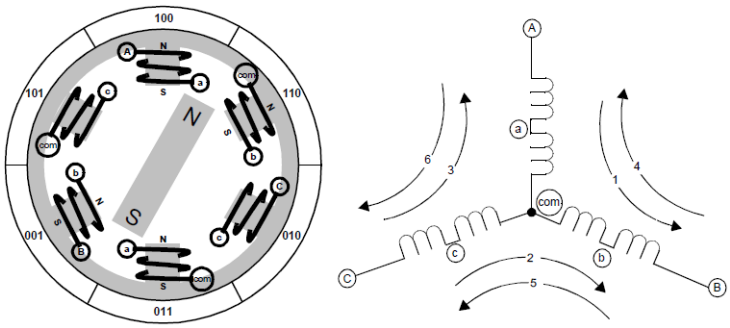

Figura 2.4: Diagrama simplificado de los componentes de un motor sin escobillas o BLDC (*Brush Less Direct Current motor*) [74].

El circuito usado para suministrar el voltaje a cada fase es el mostrado en la figura 2.5. Dependiendo de la conmutación que se requiera se pueden activar y desactivar cada uno de los tres medios puentes H mostrados en la figura; usando seis señales TTL para intercambiar los estados de los transistores $Q0, Q1, \cdots, Q5$. Así pues, se necesitan seis bits como señal de control.

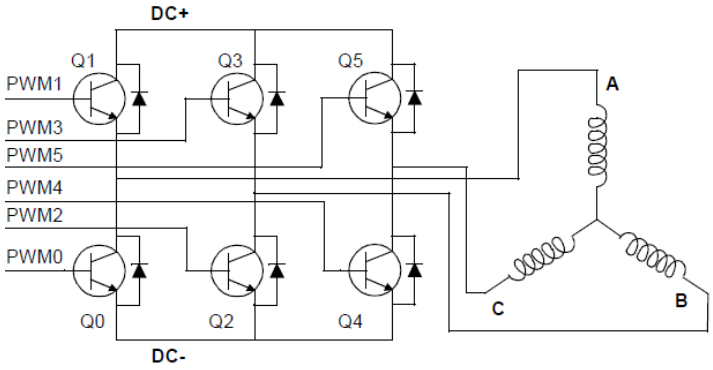

Figura 2.5: Circuito de 3 medios puentes H usados para suministrar un voltaje $DC+$ al motor BLDC. Este se puede controlar usando una señal de 6 bits (*Brush Less Direct Current motor*) [75].

Por otro lado, el máximo par se consigue cuando el imán permanente del rotor está alineado 90° con respecto al campo magnético del estator; así que, conocer la posición del rotor es fundamental para controlar adecuadamente este tipo de motores. El motor utilizado cuenta con tres sensores de efecto Hall, como los mostrados en la figura 2.7. De dichos sensores se obtiene una señal de voltaje que es máxima cuando el imán del rotor está alineado con el sensor. Amplificando y digitalizando estas señales es como obtenemos una señal de realimentación de 3 bits.

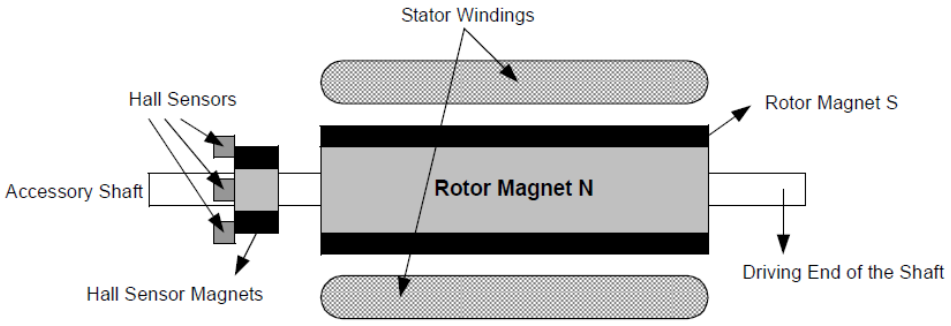

Figura 2.6: Diagrama simplificado de los componentes de un motor sin escobillas o BLDC (*Brush Less Direct Current motor*) [75].

Distinguiendo los sensores como: A, B y C; podemos relacionar las señales producidas por los sensores con el voltaje aplicado a cada fase, para construir la tabla 2.1.

Fase	Sensor A	Sensor B	Sensor C	Q0	Q1	Q2	Q3	Q4	Q5
1	1	0	1	0	0	0	1	1	0
2	1	0	0	1	0	0	1	0	0
3	1	1	0	1	0	0	0	0	1
4	0	1	0	0	0	1	0	0	1
5	0	1	1	0	1	1	0	0	0
6	0	0	1	0	1	0	0	1	0

Cuadro 2.1: Orden de activación de las fases para mover el motor BLDC en el sentido horario [74, 75].

Allí se muestran los tres bits de la señal de realimentación y los seis bits de la señal de control, utilizados para mover el motor BLDC en el sentido horario [74, 75]. Para mover el motor en el sentido antihorario también existe una tabla análoga.

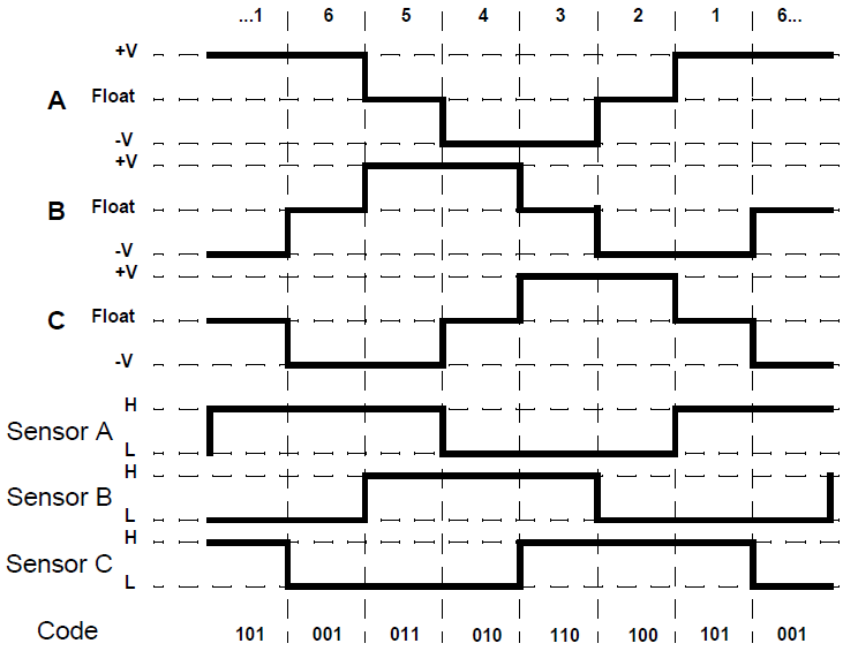

Figura 2.7: Señales porducidas por los sensores de efecto Hall y señales para activar cada fase del motor BLDC [74].

La tabla 2.1 define una función lógica que puede resolverse usando un mapa de Karnaugh [76]. Usando esta solución se diseñó un circuito con compuertas lógicas, para mover el motor BLDC DB28M01. La velocidad y el sentido del giro fueron controlados usando un Arduino UNO. El programa desarrollado en el Arduino UNO funciona como nodo suscriptor de ROS; el cual recibe las consignas de velocidad y sentido de giro, del motor BLDC y las lleva a cabo con el circuito diseñado. Adicionalmente también hay una función para recibir la consigna del ángulo de dirección del auto y mover el servomotor HS-645MG. Los detalles de este circuito, los circuitos impresos y los programas utilizados se detallan en el Apéndice A.

Capítulo 3

Visión

El vehículo AutoNOMOS tiene dos cámaras. La primera cámara es de tipo "*fish-eye*" ELP 1080p con una lente de gran angular y está dirigida hacia arriba; mientras que la segunda es una cámara RGBD Intel RealSense SR300 montada sobre el vehículo a 16.5 cm sobre el suelo; con su eje óptico paralelo al suelo y mirando hacia el frente. La primera cámara se utiliza para obtener la pose del auto (x, y, θ) con respecto a algún marcador ubicado en el techo (por ejemplo 4 cuadrados de diferente color); de este modo funciona como una especie de GPS visual. La segunda se utiliza para estimar la posición lateral del vehículo con respecto a su carril y para estimar la curvatura de la carretera. Estas mediciones geométricas serán utilizadas para controlar la dirección del vehículo mediante una señal st.

Debido a la altura a la que se encuentra la segunda cámara, el campo de visión de la misma solo captura el camino que se encuentra a una distancia mayor o igual a $L_h = 50\ cm$, sobre el eje Y del referencial de navegación $\{N\}$; tal y como se ilustra en la figura 3.1. Así pues, la posición lateral y la curvatura del camino se medirán usando los pares de puntos (x_1, y_1) y (x_2, y_2); siendo $y_1 = L_h$, $y_2 = L_h + l$ mientras que x_1 y x_2 son la distancia entre el eje Y y la línea de la derecha del carril.

Las distancias x_1 y x_2 se pueden medir usando las imágenes capturadas por la cámara a bordo; sin embargo, estas presentan una deformación de perspectiva la cual dificulta la obtención de las características del camino. Es por esta razón que en el análisis y procesamiento de datos del sistema de visión artificial se requiere aplicar una transformación de homografía para convertir las imágenes obtenidas en imágenes sin perspectiva, las cuales muestran el camino como si se viera desde arriba.

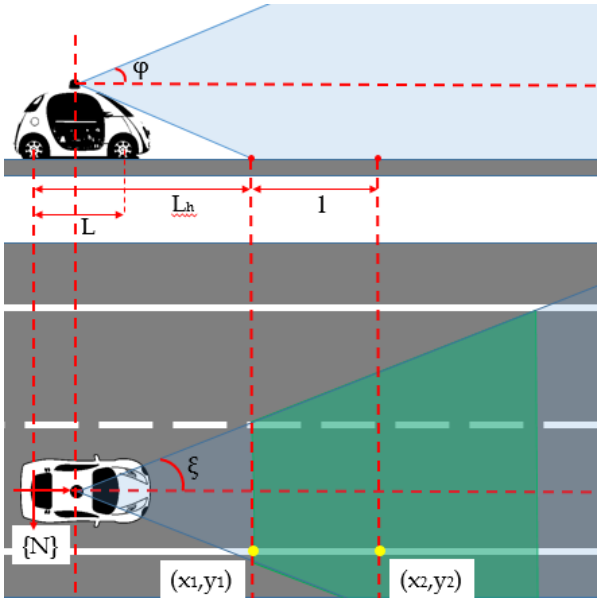

Figura 3.1: Campo de visión de la cámara a bordo del auto. Se supone la cámara tiene un ángulo de apertura vertical ϕ y un ángulo de apertura horizontal ξ.

3.1. Modelo de proyección de la cámara

La figura 3.2 muestra todos los referenciales presentes en el modelo de proyección de la cámara. Este modelo permite obtener la posición 2D en la imagen de un punto P en la escena. $\{C\}$ es el marco de referencia asociado a la cámara cuyo eje ${}^{C}Z$ (eje óptico) apunta a la escena. El plano imagen es el lugar geométrico donde se forma la imagen de la escena, es normal al eje óptico y se encuentra a una distancia f del origen de $\{C\}$.

Tiene asociado el marco de referencia $\{I\}$ cuyo origen es el punto donde se intersectan el plano imagen y el eje óptico (punto principal de la imagen). Un punto ${}^{C}P$ en la escena con coordenadas (X, Y, Z) en $\{C\}$; se proyectará en la imagen con coordenadas ${}^{I}P = (x, y)$ en $\{I\}$. La relación entre ambos sistemas coordenados está dada por [16]:

$$x = f\frac{X}{Z}, \quad y = f\frac{Y}{Z}$$

o bien, haciendo $\lambda = Z$, podemos reescribir en coordenadas homogéneas:

$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.1}$$

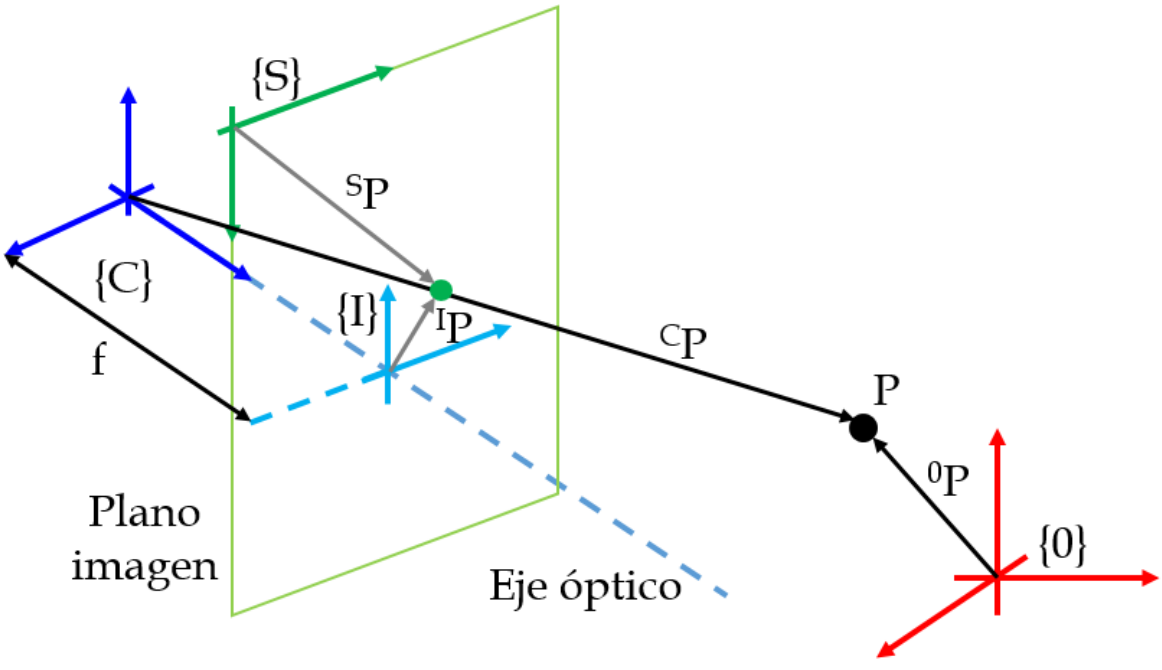

Figura 3.2: Modelo de proyección de la cámara.

Por otro lado, las cámaras digitales usan chips sensibles a la luz, los cuales están compuestos por una matriz de $W \times H$ elementos conocidos como fotositos, los cuales tienen asociado un ancho ρ_W y una altura ρ_H. Este sensor se ubica sobre el plano imagen y se le asocia el maro de referencia $\{S\}$ cuyo origen se encuentra en la esquina superior izquierda del sensor. De este modo las coordenadas del punto imagen ${}^S P = (u, v)$ en el referencial del sensor son:

$$u = \frac{x}{\rho_W} + u_0, \quad v = \frac{y}{\rho_H} + v_0 \tag{3.2}$$

en donde (u_0, v_0) son las coordenadas en pixeles del punto principal de la imagen. Concatenando la ecuación 3.2 con la ecuación 3.1 se obtiene:

$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} 1/\rho_w & 0 & u_0 \\ 0 & 1/\rho_H & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.3}$$

$$= \begin{pmatrix} f/\rho_w & 0 & u_0 & 0 \\ 0 & f/\rho_H & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.4}$$

Así pues, dada la posición (X, Y, Z) de un punto 3D en la escena, con respecto a $\{C\}$, esta ecuación permite obtener su proyección en el sensor, ubicado en el plano imagen, como una posición (u, v) en pixeles. Sin embargo, las coordenadas de P no suelen medirse desde $\{C\}$ sino que generalmente se expresan desde algún referencial del mundo $\{0\}$; siendo ${}^0 P = ({}^0 X, {}^0 Y, {}^0 Z)$.

Entonces, la transformación homogénea que define a la pose de la cámara con respecto a $\{0\}$ y que relaciona al punto ${}^0P$ con el punto ${}^CP$; está dada por:

$$ {}^CP = \underbrace{\begin{pmatrix} R & \overrightarrow{t} \\ \overrightarrow{0}^T & 1 \end{pmatrix}}_{T} {}^0P \tag{3.5} $$

donde R es la orientación de la cámara con respecto a $\{0\}$ y $\overrightarrow{t}$ su vector de posición posición. A la transformación T se le conoce como matriz de parámetros extrínsecos de la cámara.

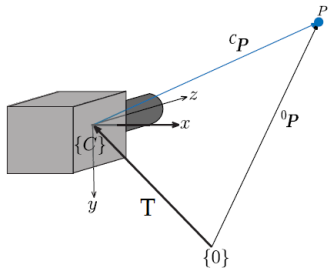

Figura 3.3: Relación entre los sistemas coordenados $\{C\}$ y $\{0\}$ [16].

Por lo tanto, al sustituir la ecuación 3.5 en 3.4 obtenemos:

$$ \lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \underbrace{\begin{pmatrix} f/\rho_w & 0 & u_0 & 0 \\ 0 & f/\rho_H & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{(K|\overrightarrow{0})} \begin{pmatrix} R & \overrightarrow{t} \\ \overrightarrow{0}^T & 1 \end{pmatrix} \begin{pmatrix} {}^0X \\ {}^0Y \\ {}^0Z \\ 1 \end{pmatrix} $$

$$ = K(I_{3\times 3}|\overrightarrow{0}) \begin{pmatrix} R & \overrightarrow{t} \\ \overrightarrow{0}^T & 1 \end{pmatrix} \begin{pmatrix} {}^0X \\ {}^0Y \\ {}^0Z \\ 1 \end{pmatrix} $$

$$\Rightarrow \lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = K(R|\overrightarrow{t}) \begin{pmatrix} {}^0X \\ {}^0Y \\ {}^0Z \\ 1 \end{pmatrix} \tag{3.6}$$

A la matriz K se le conoce como matriz de parámetros intrínsecos o de calibración, mientras que $K(R|\overrightarrow{t})$ es el modelo de proyección en perspectiva de la cámara. En este caso, los puntos de interés de este proyecto se encuentran en un plano: el suelo. Así pues, ubicando el referencial $\{0\}$ de modo que los puntos del suelo tengan coordenadas $({}^0X, {}^0Y, 0)$, el modelo de la ecuación 3.6 se puede simplificar reescribiendo $R = (\overrightarrow{r_1}|\overrightarrow{r_2}|\overrightarrow{r_3})$ como $R = (\overrightarrow{r_1}|\overrightarrow{r_2})$ y con esto obtener la ecuación:

$$\begin{aligned} \lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} &= \underbrace{K(\overrightarrow{r_1}|\overrightarrow{r_2}|\overrightarrow{t})}_{H^{-1}} \begin{pmatrix} {}^0X \\ {}^0Y \\ 1 \end{pmatrix} \\ &= H^{-1} \begin{pmatrix} {}^0X \\ {}^0Y \\ 1 \end{pmatrix} \end{aligned} \tag{3.7}$$

De este modo, el modelo de proyección en perspectiva se simplifica, quedando una matriz de homografía; mapenado puntos 2D en el suelo (con coordenadas en *cm* por ejemplo) a puntos 2D en el sensor del plano imagen (con coordenadas en pixeles). Sin embargo, para conocer el error de posición lateral del vehículo dentro de su carril (en *cm*); así como la curvatura de la carretera (en grados); es necesario utilizar la transformación inversa a 3.7 que es:

$$\begin{pmatrix} {}^0X \\ {}^0Y \\ 1 \end{pmatrix} = \lambda H \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \tag{3.8}$$

3.2. Proceso de identificación de la matriz de homografía

Para calcular los 9 elementos que definen a H^{-1} en 3.7 tenemos que considerar un conjunto de n puntos en la imagen (no co-lineales) $\{(u_1, v_1, 1)^T, \cdots, (u_1, v_1, 1)^T\}$ y conocer su correspondiente posición $\{({}^0X_1, {}^0Y_1, 1)^T, \cdots, ({}^0X_n, {}^0Y_n, 1)^T\}$ con respecto a algún referencial $\{0\}$; de este modo los podemos relacionarlos con la ecuación [77]:

$$\begin{pmatrix} \lambda_1 u_1 & \cdots & \lambda_n u_n \\ \lambda_1 v_1 & \cdots & \lambda_n v_n \\ \lambda_n & \cdots & \lambda_n \end{pmatrix} = H^{-1} \begin{pmatrix} {}^0X_1 & \cdots & {}^0X_n \\ {}^0Y_1 & \cdots & {}^0Y_n \\ 1 & \cdots & 1 \end{pmatrix}$$

La primera columna de la matriz de la derecha nos definirá tres ecuaciones que son:

$$\begin{aligned} \lambda_1 u_1 &= x_1 h_{11} + y_1 h_{12} + h_{13} \\ \lambda_1 v_1 &= x_1 h_{21} + y_1 h_{22} + h_{23} \\ \lambda_1 &= x_1 h_{31} + y_1 h_{32} + h_{33} \\ \Rightarrow u_1 &= \frac{x_1 h_{11} + y_1 h_{12} + h_{13}}{x_1 h_{31} + y_1 h_{32} + h_{33}} \\ v_1 &= \frac{x_1 h_{21} + y_1 h_{22} + h_{23}}{x_1 h_{31} + y_1 h_{32} + h_{33}} \end{aligned}$$

De modo análogo podemos escribir hasta $3 \times n$ ecuaciones y usar el algoritmos de mínimos cuadrados para determinar todos los elementos de H^{-1}; usando como función de costo a:

$$J = \sum_{i=1}^{n} \left(u_i - \frac{x_i h_{11} + y_i h_{12} + h_{13}}{x_i h_{31} + y_i h_{32} + h_{33}} \right)^2 + \left(v_i - \frac{x_1 h_{21} + y_1 h_{22} + h_{23}}{x_i h_{31} + y_i h_{32} + h_{33}} \right)^2$$

siendo h_{ij} los elementos de H^{-1}. Con esto es posible calcular H de 3.8.

Para calcular la matriz de homografía H se utilizó un programa escrito en Python 2.7 y la biblioteca de Opencv. El programa se encuentra en el apéndice B y con este se obtuvieron las matrices de homografía tanto del simulador como de la cámara real.

$$H_{SIM} = \begin{pmatrix} -7.90e-02 & -4.19e-01 & 1.28e+02 \\ 1.00e-02 & -1.46 & 4.00e+02 \\ 2.60e-05 & -4.15e-03 & 1.00 \end{pmatrix} \tag{3.9}$$

$$H_{REAL} = \begin{pmatrix} -1.03e-01 & -5.60e-01 & 1.40e+02 \\ 2.05e-02 & -1.88 & 4.05e+02 \\ 4.24e-05 & -5.50e-03 & 1.00 \end{pmatrix} \tag{3.10}$$

Una vez obtenida la matriz H, se utiliza para obtener las coordenadas $({}^0X, {}^0Y)$ en cm, de los puntos en el suelo, correspondientes a sus puntos en la imagen (u, v). Entonces, aplicando H a las imágenes obtenidas por la cámara del AutoNOMOS se puede reconstruir la escena de la carretera, como se muestra en la figura 3.4. La reconstrucción de la escena permite hacer mediciones en cm y en rad; lo cual no es posible hacer en la imagen debido a la distorsión que introduce la proyección en perspectiva de la cámara.

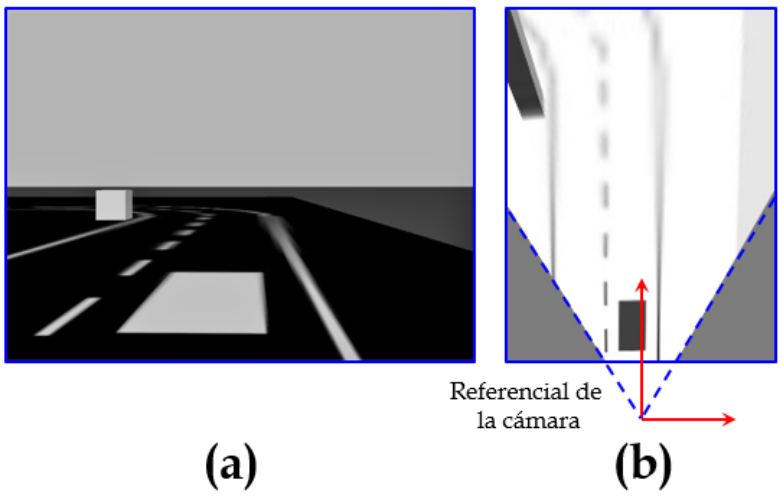

Figura 3.4: (a)Imagen obtenida por la cámara a bordo del AutoNOMOS en un ambiente simulado. (b)Reconstrucción de la escena en donde se encuentra la carretera.

Capítulo 4

Modelado y control

4.1. Modelo Cinemático del AutoNOMOS

La figura 4.1 ilustra algunas de las variables utilizadas para plantear un modelo cinemático del vehículo AutoNOMOS basado en el modelo de la bicicleta. A diferencia de los modelos planteados en la introducción, éste modelo resulta lineal, intuitivo, de fácil implementación y utiliza variables que pueden medirse desde el referencial de navegación $\{N\}$; el cual, se encuentra en la llanta trasera del modelo de la bicicleta. Como se puede observar en la figura, L_h es la distancia entre el origen de $\{N\}$ y la parte de la carretera más cercana al auto, que se puede observar con la cámara; L es la distancia entre el eje trasero y el eje delantero de las llantas; st representa el ángulo de dirección y es el ángulo que se forma entre la rueda delantera y el eje de avance ${}^{N}Y$ del vehículo y $\overrightarrow{v}$ es el vector velocidad del vehículo, el cual se supone paralelo a la dirección de avance y tiene magnitud v.

El parámetro l y las variables θ, x_1 y x_2 se miden en la imagen reconstruida donde: x_1 es la distancia que hay entre el eje ${}^{N}Y$ y la línea de la derecha que delimita el carril donde se conduce el auto. Como ${}^{N}Y$ coincide con el centro de la imagen, x_1 se mide en la parte de la carretera, más cercana al auto, que se puede observar con la cámara; es decir, a una distancia L_h.

x_2 es lo mismo que x_1 pero a una distancia l más adelante y el ángulo $\theta = \arctan((x_1 - x_2)/l)$ es el que se forma entre la tangente de la carretera y la dirección de avance.

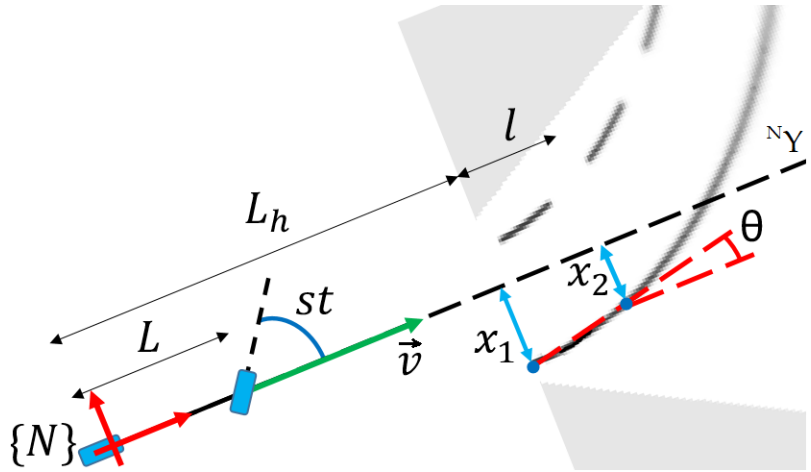

Figura 4.1: Variables utilizadas para modelar matemáticamente al vehículo AutoNOMOS.

Siendo x_{ref} una distancia constante e igual a la mitad del ancho del carril por donde se conduce el auto, se define al error de posición como $e_x \triangleq x_1 - x_{ref}$; entonces $\dot{e}_x = \dot{x}_1$. Nótese que $\dot{x}_1 = L_h\dot{\psi} + v\theta$; con ψ como el ángulo de guiñada del vehículo medido desde algún referencial inercial $\{W\}$, por lo que:

$$\dot{e}_x = L_h\dot{\psi} + v\theta \tag{4.1}$$

Por otro lado, desde el referencial $\{W\}$, la tangente a cada punto sobre la línea de la derecha del carril tiene un ángulo de inclinación θ_{ref} que es constante; tal y como se muestra en la figura 4.4. Si se define al error de orientación como $\theta \triangleq \psi - \theta_{ref}$ entonces:

$$\dot{\theta} = \dot{\psi} \tag{4.2}$$

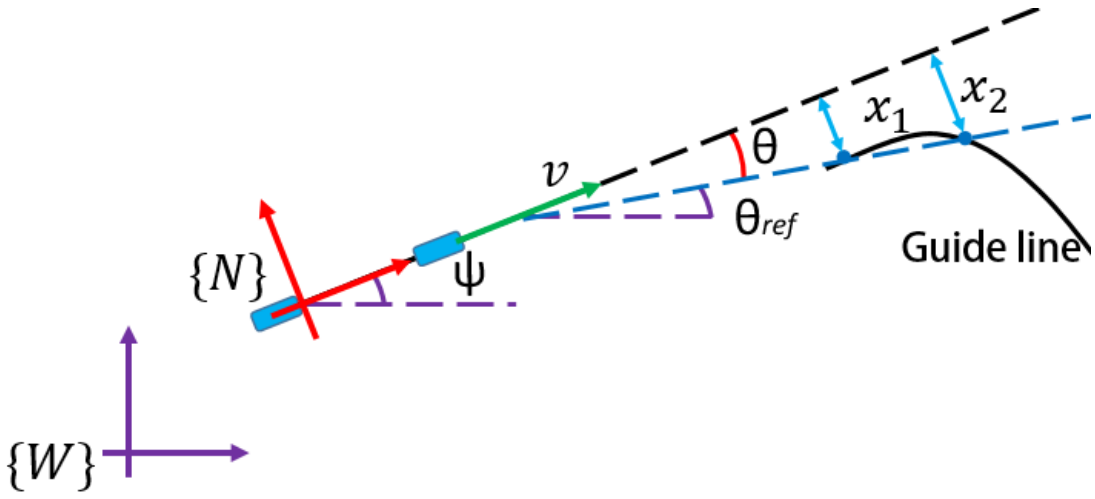

Figura 4.2: Medición del error de orientación.

Recordando que $\dot{\psi} = \frac{v}{L}\tan(st)$ [16] y definiendo la señal de control como:

$$u \triangleq \tan(st) \tag{4.3}$$

se puede modelar al sistema como:

$$\dot{e}_x = v\theta + \frac{vL_h}{L}u \tag{4.4}$$

$$\dot{\theta} = \frac{v}{L}u \tag{4.5}$$

o bien, reescrito en el espacio de estados resulta el modelo lineal:

$$\dot{x} = \underbrace{\begin{pmatrix} 0 & v \\ 0 & 0 \end{pmatrix}}_{A} x + \underbrace{\begin{pmatrix} \frac{vL_h}{L} \\ \frac{v}{L} \end{pmatrix}}_{B} u \tag{4.6}$$

$$y = \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}}_{C} x \tag{4.7}$$

con $x = (e_x \quad \theta)^T$. Este sistema y la definición 4.3 permitirán el diseño de controladores lineales que después se pueden convertir a señales de st usando la ecuación: $st = \arctan(u)$.

Nótese que este sistema tiene como matriz de controlabilidad a:

$$\mathcal{C} = \begin{pmatrix} \frac{vL_h}{L} & \frac{v^2}{L} \\ \frac{v}{L} & 0 \end{pmatrix}$$

Por lo tanto el sistema es controlable siempre que $v > 0$. Esto tiene significado físico ya que el vehículo es no-holonómico y no puede cambiar de orientación sin cambiar de posición. Por otro lado, de 4.3 se deduce que $st = \arctan(u)$, entonces $st \in (-\pi/2, \pi/2)$. Esto se debe a que ángulos de orientación tales que $|st| \geq \pi/2$ implican movimientos que un automóvil no puede realizar.

Aunque se nombran "error de posición" y "error de orientación" a las variables de estado, estas no representan los errores del automóvil en el instante t, sino que se pueden interpretar como los errores que tendrá en un instante futuro; si mantiene su angulo de dirección st. Esta conclusión resulta de calcular las cantidades e_x y θ en la imagen sin perspectiva; la cual está a una distancia L_h enfrente del vehículo.

Alternativamente, este sistema puede discretizarse utilizando una señal de control discreta u_k con un periodo de muestreo h; entonces de la ecuación 4.6 se obtiene [78]:

$$\dot{x} = Ax + Bu_k \quad \text{Mult. todo por la izquierda por } e^{-tA}$$
$$e^{-tA}\dot{x} - e^{-tA}Ax = e^{-tA}Bu_k \quad \text{Integrando de } t_k \text{ a } t_{k+1}$$

$$\int_{t_k}^{t_{k+1}} \frac{d}{d\tau}(e^{-tA}x(\tau))d\tau = \int_{t_k}^{t_{k+1}} e^{-\tau A}Bu_k d\tau$$

$$e^{-t_{k+1}A}x_{k+1} - e^{-t_kA}x_k = \int_{t_k}^{t_{k+1}} e^{-\tau A}Bd\tau u_k$$

$$e^{-t_{k+1}A}x_{k+1} = e^{-t_kA}x_k + \int_{t_k}^{t_{k+1}} e^{-\tau A}Bd\tau u_k$$

$$x_{k+1} = e^{(t_{k+1}-t_k)A}x_k + \int_{t_k}^{t_{k+1}} e^{(t_{k+1}-\tau)A}Bd\tau u_k \quad \text{haciendo: } s = t_{k+1} - \tau$$

$$x_{k+1} = e^{(t_{k+1}-t_k)A}x_k - \int_{t_{k+1}-t_k}^{0} e^{sA}Bds u_k \quad \text{como: } t_{k+1} - t_k = h$$

$$\Rightarrow x_{k+1} = \underbrace{e^{hA}}_{A_d}x_k + \underbrace{\int_0^h e^{sA}Bds}_{B_d} u_k$$

Por lo tanto, las ecuaciones 4.6 y 4.7 pueden reescribirse como el sistema discreto:

$$x_{k+1} = \underbrace{\begin{pmatrix} 1 & hv \\ 0 & 1 \end{pmatrix}}_{A_d} x_k + \underbrace{\begin{pmatrix} h\frac{vL_h}{L} + h^2\frac{v^2}{2L} \\ h\frac{v}{L} \end{pmatrix}}_{B_d} u_k \tag{4.8}$$

$$y_k = \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}}_{C_d} x_k \tag{4.9}$$

siendo $x_k = (e_{x_k} \quad \theta_k)^T$.

4.2. Controladores del ángulo de dirección

En total se plantearon 5 controladores diferentes para conducir el AutoNOMOS por una pista dibujada con líneas blancas sobre un fondo negro, sin obstáculos.

4.2.1. Controlador 1

El primer controlador está dado por la ecuación:

$$st = -K_x e_x \tag{4.10}$$

donde $K_x > 0$ es una ganancia proporcional que se usa para ponderar el error de posición y compensarlo mediante un ángulo de dirección st.

Se puede probar la estabilidad de este controlador usando como función candidata de Lyapunov:

$$\begin{aligned} V &= \frac{1}{2}(e_x^2 + \theta^2) \\ \Rightarrow \dot{V} &= \frac{1}{2}(2e_x\dot{e}_x + 2\theta\dot{\theta}) \\ &= e_x\Big(v\theta + \frac{vL_h}{L}u\Big) + \theta\Big(\frac{v}{L}u\Big) \\ &= ve_x\theta + \Big(u\frac{v}{L}\Big)(e_xL_h + \theta) \\ \text{como: } u &\triangleq \tan(st) \\ \Rightarrow \dot{V} &= ve_x\theta + \Big(\tan(-K_xe_x)\frac{v}{L}\Big)(e_xL_h + \theta) \end{aligned}$$

como se debe cumplir que $\dot{V} < 0$, entonces:

$$\begin{aligned} ve_x\theta + \Big(\tan(-K_xe_x)\frac{v}{L}\Big)(e_xL_h + \theta) &< 0 \\ \tan(-K_xe_x) &< \frac{-Le_x\theta}{e_xL_h + \theta} \\ -K_x &< \frac{1}{e_x}\arctan\Big(\frac{-Le_x\theta}{e_xL_h + \theta}\Big) \\ K_x &> \frac{1}{e_x}\arctan\Big(\frac{Le_x\theta}{e_xL_h + \theta}\Big) \\ K_x &> \frac{1}{e_x}\arctan\Big(\frac{L}{\frac{L_h}{\theta} + \frac{1}{e_x}}\Big) \end{aligned}$$

Debido a la forma del camino, el estado θ esta acotado entre dos valores: uno negativo (cuando hay una curva hacia la izquierda) y otro positivo (cuando hay una curva hacia la derecha); por lo que el rango de valores en el que varía no supone algún problema para la estabilidad. Por otro lado, de esta última expresión se observa una indeterminación cuando $e_x = 0$; así que, si se calcula el límite usando L'Hopital resulta:

$$\begin{aligned}\lim_{e_x\to 0}\frac{1}{e_x}\arctan\left(\frac{L}{\frac{L_h}{\theta}+\frac{1}{e_x}}\right) &= \lim_{e_x\to 0}\frac{\frac{-L(-\frac{1}{e_x^2})}{(\frac{L_h}{\theta}+\frac{1}{e_x})^2}}{1+\left(\frac{L}{\frac{L_h}{\theta}+\frac{1}{e_x}}\right)^2}\\ &= \lim_{e_x\to 0}\frac{L}{L^2e_x^2+(1+\frac{L_he_x}{\theta})^2}\\ &= L\end{aligned}$$

Ya que el límite existe, se concluye que sí existe un valor de $K_x > 0$ para el cual el sistema en lazo cerrado es globalmente asintóticamente uniformemente estable.

4.2.2. Controlador 2

Similar al controlador anterior, como segunda ley de control se propone:

$$st = -K_x e_x - K_\theta \theta \tag{4.11}$$

siendo $K_x, K_\theta > 0$. Del mismo modo, para probar la estabilidad se usa como función candidata de Lyapunov a:

$$\begin{aligned}V &= \frac{1}{2}(e_x^2+\theta^2)\\ \Rightarrow \dot{V} &= \frac{1}{2}(2e_x\dot{e}_x + 2\theta\dot{\theta})\end{aligned}$$

$$\begin{aligned}\dot{V} &= e_x\Big(v\theta + \frac{vL_h}{L}u\Big) + \theta\Big(\frac{v}{L}u\Big)\\ &= ve_x\theta + \Big(u\frac{v}{L}\Big)(e_xL_h + \theta)\end{aligned}$$

$$\text{como: } u = \tan(-K_xe_x - K_\theta\theta)$$

$$\Rightarrow \dot{V} = ve_x\theta + \Big(\tan(-K_xe_x - K_\theta\theta)\frac{v}{L}\Big)(e_xL_h + \theta)$$

Dado que, para que el sistema sea estable debe ocurrir que $\dot{V} < 0$

$$\Rightarrow \tan(-K_xe_x - K_\theta\theta) < \frac{-ve_x\theta}{\frac{v}{L}\Big(e_xL_h + \theta\Big)}$$

$$-K_xe_x - K_\theta\theta < \arctan\left(\frac{-ve_x\theta}{\frac{v}{L}\Big(e_xL_h + \theta\Big)}\right)$$

$$K_xe_x + K_\theta\theta > \arctan\left(\frac{L}{\frac{L_h}{\theta} + \frac{1}{e_x}}\right)$$

$$K_x + K_\theta\frac{\theta}{e_x} > \frac{1}{e_x}\arctan\left(\frac{L}{\frac{L_h}{\theta} + \frac{1}{e_x}}\right)$$

Así pues, este sistema en lazo cerrado será estable siempre que e_x y θ sean del mismo signo y para valores suficientemente grandes de K_x; sin embargo no se puede garantizar la estabilidad global.

4.2.3. Controlador basado en campos potenciales

La tercera ley de control propuesta está dada por:

$$st = -K_xe_x - K_\theta\theta_{PF} \tag{4.12}$$

La cual, a diferencia del controlador 4.11, utiliza el enfoque del Control Basado en Campos Potenciales para calcular el ángulo de orientación θ_{PF}; modelando al vehículo y algunos puntos sobre el suelo, como si fueran cargas eléctricas [79–81].

La figura 4.3 muestra la posición de las cargas virtuales con las que se generará un potencial de tipo electrostático, que sea repulsivo en las orillas del carril y atractivo en el centro. De este modo se pretende obtener una fuerza virtual cuya orientación se usa para mantener al auto en el centro de su carril.

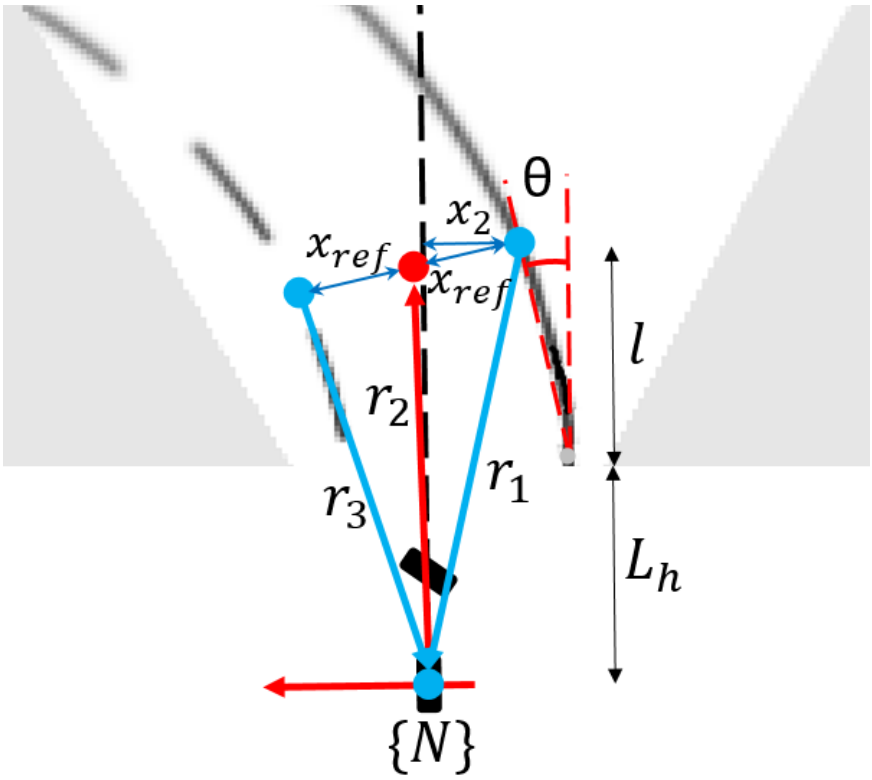

Figura 4.3: Para que el potencial resultante sea atractivo en el centro del carril y repulsivo en las orillas, se ponen cargas virtuales del mismo signo sobre $\{N\}$ y en las orillas del carril (puntos azules); y una de signo contrario en el centro del carril (punto rojo).

Siendo θ el error de orientación y x_2 como se definió anteriormente; de la figura 4.3 se puede deducir que:

$$\begin{aligned}
\overrightarrow{r}_1 &= x_2\hat{\imath} + (L_h + l)\hat{\jmath} \\
\overrightarrow{r}_2 &= (x_2 - x_{ref}\cos(\theta))\hat{\imath} + (L_h + l - x_{ref}|\operatorname{sen}(\theta)|)\hat{\jmath} \\
\overrightarrow{r}_3 &= (2x_{ref}\cos(\theta) - x_2)\hat{\imath} + (L_h + l - 2x_{ref}|\operatorname{sen}(\theta)|)\hat{\jmath}
\end{aligned}$$

Por lo tanto la fuerza resultante sobre la carga puesta en el referencial de navegación $\{N\}$ es:

$$\begin{aligned}
\overrightarrow{F} &= K_+\frac{\overrightarrow{r}_1}{|\overrightarrow{r}_1|^2} - K_-\frac{\overrightarrow{r}_2}{|\overrightarrow{r}_2|^2} + K_+\frac{\overrightarrow{r}_3}{|\overrightarrow{r}_3|^2} \\
&= (x_{ref}\cos(\theta)(2K_+ + K_-) - K_-X_2)\hat{\imath} \\
&+ ((L_h + l)(2K_+ - K_-) + x_{ref}|\operatorname{sen}(\theta)|(K_- - 2K_+))\hat{\jmath}
\end{aligned}$$

Donde: K_+ y K_- son ganancias ajustables propuestas para ponderar el peso de las fuerzas producidas por las cargas virtuales; $\hat{\jmath}$ es un vector unitario que apunta en la dirección de avance del auto e $\hat{\imath}$ es un vector unitario transversal al movimiento del mismo. De las componentes de esta fuerza se deduce que su orientación está dada por:

$$\begin{aligned}
\theta_{PF} &= \arctan\left(\frac{x_{ref}\cos(\theta)(K_- + 2K_+) - K_-x_2}{(2K_+ - K_-)(L_h + l - x_{ref}|\sin(\theta)|)}\right) \\
\therefore \theta_{PF} &= \arctan\left(\frac{x_2\left(\frac{K_-}{K_- - 2K_+}\right) - x_{ref}\cos(\theta)\left(\frac{K_- + 2K_+}{K_- - 2K_+}\right)}{L_h + l - x_{ref}|\sin(\theta)|}\right)
\end{aligned}$$

Para que la orientación resultante se mantenga entre $\pm\pi/2$ se debe cumplir que $K_- \leqslant 2K_+$.

4.2.4. Controlador óptimo

En el cuarto esquema de control propuesto se diseñó un controlador óptimo para el sistema discreto presentado en las ecuaciones 4.8 y 4.9 [78, 82, 83]; el cual minimiza el criterio de desempeño:

$$J = \sum_{k=1}^{\infty} x_k^T Q x_k + u_k^T R u_k$$

siendo $Q \geq 0$ y $R > 0$ matrices dadas. Usando programación dinámica y el principio de optimalidad, de este sistema se puede obtener la ecuación de Ricatti discreta:

$$P = A_d^T P A_d + Q - A_d^T P B_d (R + B_d^T P B_d)^{-1} B_d^T P A_d$$

cuya solución se usa para calcular la matriz de ganancias:

$$K = (R + B_d^T P B_d)^{-1} B_d^T P A_d$$

y que a su vez, se usa para calcular la señal de control óptima $u_k = -K x_k$. De la definición 4.3 se deduce que la ley de control resultante toma la forma:

$$st_k = -\arctan(K x_k) \tag{4.13}$$

4.2.5. *"Preview controller"*

El quinto esquema de control propuesto esta basado en la teoría del *Preview controller* [22–30]. Al igual que en el controlador óptimo, en este esquema se utiliza el modelo discreto de la planta y se propone una ley de control capaz de minimizar el criterio de desempeño [31]:

$$J = \sum_{k=1}^{\infty} (x_k^{ref} - x_k)^T Q (x_k^{ref} - x_k) + u^T R u \tag{4.14}$$

para $Q \geq 0$ y $R > 0$ matrices dadas, en este criterio de desempeño también se ponderan las referencias futuras de las que se disponen; esto con el objetivo de mejorar

el desempeño al seguir la trayectoria dada. Como se discutió en la introducción, este esquema es particularmente importante para vehículos autónomos, ya que si se dispone de un sistema de visión se puede obtener información del camino que se tiene adelante y por lo tanto de las referencias futuras. Siguiendo el mismo procedimiento que el del controlador óptimo, se puede obtener que:

$$u_k = -Kx_k - (f_1^T, f_2^T, \cdots, f_N^T) \begin{pmatrix} x_{k+1}^{ref} \\ x_{k+2}^{ref} \\ \vdots \\ x_{k+N}^{ref} \end{pmatrix}$$

donde $(x_{k+1}^{ref}, x_{k+2}^{ref}, \cdots, x_{k+N}^{ref})^T$ son las N referencias de las que se disponen y las constantes de proporción están dadas por:

$$K = (R + B_d^T P B_d)^{-1} B_d^T P A_d$$
$$f_i = (R + B_d^T P B_d)^{-1} B_d^T ((A_d - B_d K)^T)^{i-1} C_d^T Q$$

Al primer término de u_k se le conoce como "*State Feedback*" y es igual al calculado con el controlador óptimo; al segundo término se le conoce como "*Preview Compensations*" y es el término que pondera las referencias futuras para mejorar el desempeño del seguimiento de trayectorias. Así pues, la ley de control propuesta con este esquema es:

$$st_k = -\arctan \left(Kx_k + (f_1^T, f_2^T, \cdots, f_N^T) \begin{pmatrix} x_{k+1}^{ref} \\ x_{k+2}^{ref} \\ \vdots \\ x_{k+N}^{ref} \end{pmatrix} \right) \tag{4.15}$$

4.2.6. Implementación

Debido a las características del servomotor, cada ley de control st presentada en este capítulo se convierte de radianes a grados, se multiplica por -1 y se le suma un *offset* de 90°; para así obtener el ángulo de dirección deseado. Cabe resaltar que, tanto en el modelado como en la ley de control, no se toma en cuenta la dinámica presente entre el servomotor y el mecanismo de Ackerman con el que se cambia la dirección del auto y se asume que el auto gira con el ángulo de orientación deseado. En trabajos futuros se hará un estudio más detallado de este problema adicional.

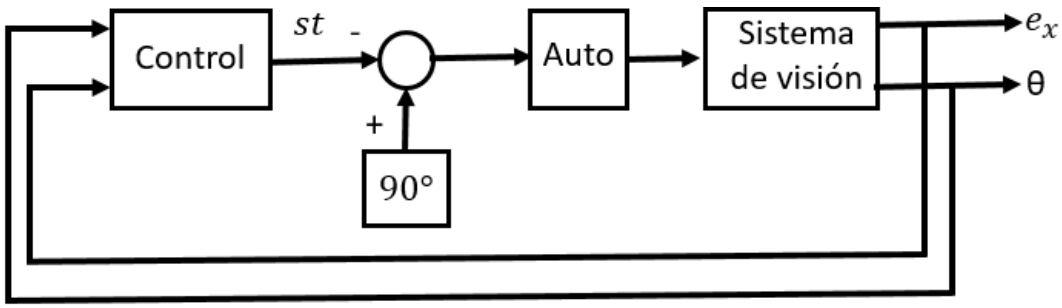

Figura 4.4: Diagrama de bloques del sistema en lazo cerrado.

Capítulo 5

Desarrollo experimental y resultados

Los controladores propuestos en el capítulo anterior fueron implementados tanto en simulador como en el vehículo real. El simulador utilizado es el desarrollado por la *Freie Universität Berlin* para el vehículo AutoNOMOS [73]. Este simulador se desarrolló con Gazebo y al igual que el vehículo real funciona con el software de ROS; pero de la versión Kinetic, por lo que para ambos sistemas los controladores propuestos se implementaron como nodos de ROS. La interfaz gráfica del simulador se muestra en la figura 5.1.

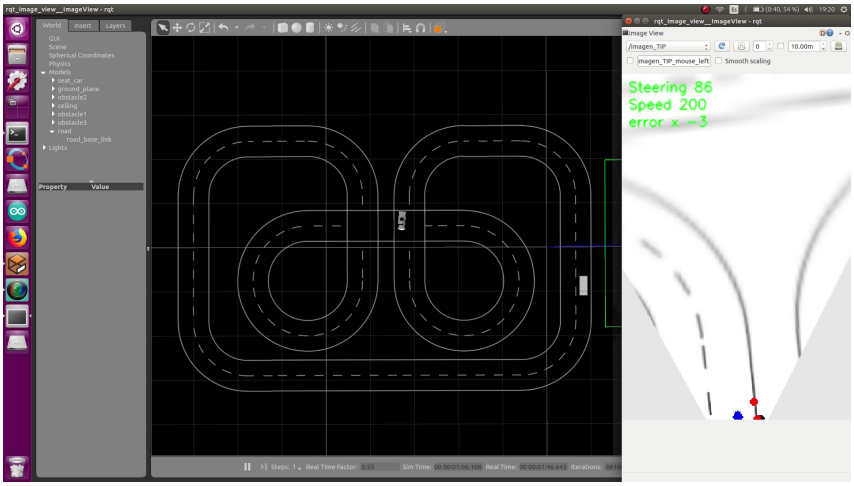

Figura 5.1: Interfaz gráfica del simulador del AutoNOMOS.

La figura 5.2 muestra el diagrama de conexiones entre los tópicos y nodos en ROS en el simulador. Los tópicos generados por el simulador son:

- `\camera_rgb\image_raw`
- `\manual_control\speed`
- `\manual_control\steering`
- `\IMU`

En el tópico `\camera_rgb\image_raw` se publican las imágenes obtenidas por la cámara del simulador; mientras que en `\IMU` se publica la información del sensor inercial del auto que se estaba simulando. Por otro lado, los tópicos `\manual_control\speed` y `\manual_control\steering` son utilizados por el simulador para obtener la velocidad de avance y el ángulo de orientación del auto; y con esos datos simular el movimiento.

Para automatizar la conducción, visualizar la imagen sin perspectiva y adquirir los datos para medir el desempeño; se programaron tres nodos. El nodo de control se usó para adquirir y procesar la imagen del camino; y usar la información obtenida para generar un ángulo de orientación st. Además publica una velocidad v constante y el ángulo st en sus respectivos tópicos. Este nodo también publica en `\vis_points`, la posición en la imagen sin perspectiva donde se encuentra la línea derecha del carril. Por otro lado, en el tópico `\data_send` publican las variables: t, e_x, th, st y x_1; antes definidas. Se usó un nodo de control diferente para cada controlador establecido en el capítulo anterior y se puede encontrar el código de cada uno en el apéndice C. Este nodo se ejecutó en la computadora a bordo del auto.

El nodo de visualización se ejecutó en una laptop conectada al AutoNOMOS via wifi por medio de ROS y se utilizó para visualizar la imagen sin perspectiva del camino. Se hizo de este modo ya que la visualización de los datos no es una tarea esencial para la conducción autónoma; por lo que se prefirió no utilizar la computadora Odroid XU4 para esta tarea. Dicho nodo es un suscriptor de los tópicos `\camera_rgb\image_raw`, `\manual_control\speed`, `\manual_control\steering` y `\vis_points`; y usa la información adquirida para obtener la imagen sin perspectiva, mostrar el ángulo st y la velocidad v; y para resaltar los puntos utilizados para medir x_1 y x_2 definidos en el capítulo anterior. La figura 5.3 muestra la imagen generada por este nodo. Los colores de la misma se invirtieron para favorecer la interpretación de la misma. El código de este nodo se encuentra en el apéndice C.

Finalmente, el nodo de adquisición de datos se usó para obtener y guardar la información generada por el simulador para así poder evaluar el desempeño de cada controlador utilizado. Este nodo es un suscriptor de los tópicos `\data_send` e `\IMU`. El código de este nodo también se encuentra en el apéndice C.

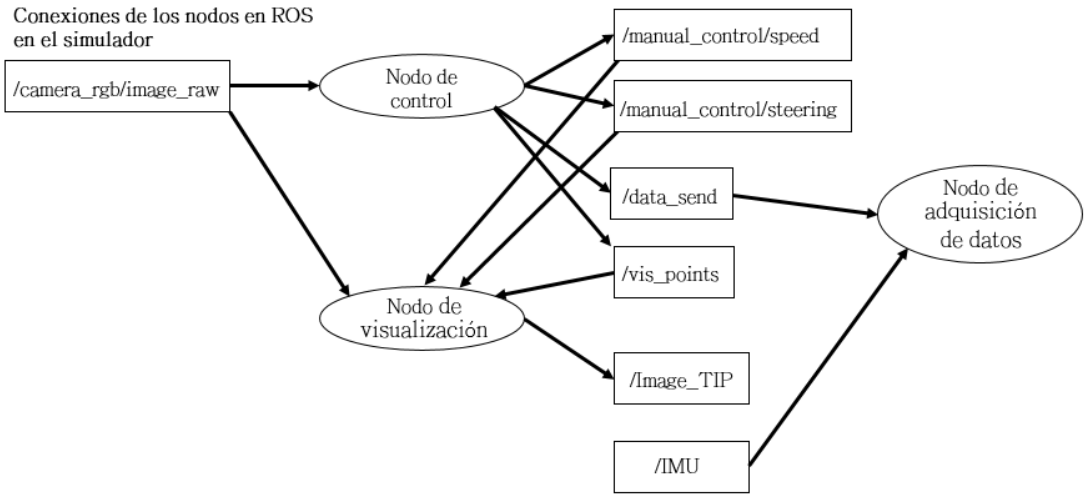

Figura 5.2: Diagrama de conexión entre los nodos de ROS del simulador.

Figura 5.3: Imagen sin perspectiva y visualización de otros datos generada por el nodo de visualización.

Respecto a los nodos utilizados por el vehículo real estos son casi los mismos que los utilizados en el simulador salvo por ínfimas diferencias en los nombres de los tópicos y los tipos de mensaje utilizados; por esta razón se omite su código en el apéndice C. La figura 5.4 muestra el diagrama de conexión entre los mismos. Como se puede observar, adicionalmente hay tres nodos más que en el caso del simulador. El nodo de la cámara es parte del software desarrollado por la *Freie Universität Berlin* para el vehículo AutoNOMOS y publica en `\camera_rgb\image_raw` la imagen adquirida por la cámara. El nodo en la imagen nombrado Arduino UNO control de motores, es el expuesto en el apéndice A y es el que se diseñó para controlar el servomotor y el motor sin escobillas. Finalmente el nodo Arduino NANO IMU es el nodo implementado en el Arduino NANO y fue diseñado para publicar en `\IMU` la información del sensor inercial a bordo de automóvil. Su código también se encuentra en el apéndice C.

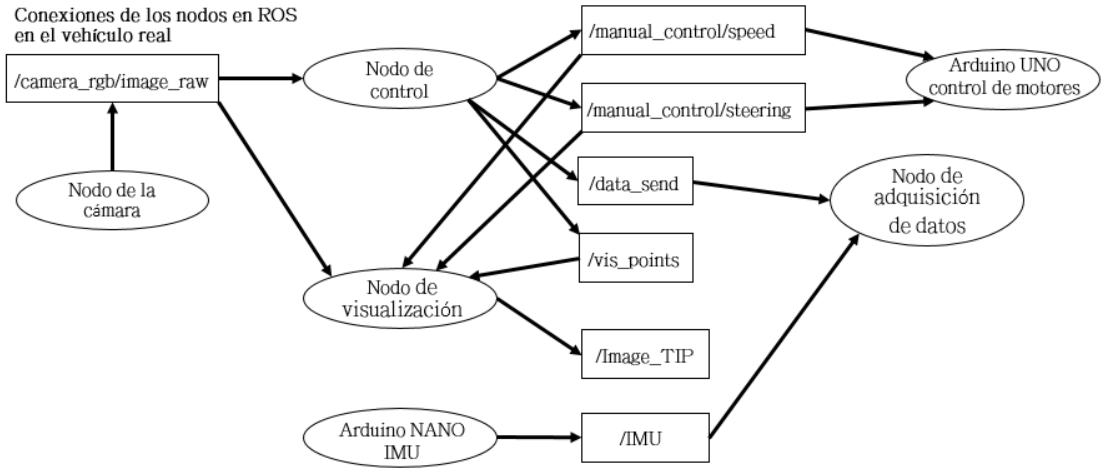

Figura 5.4: Diagrama de conexión entre los nodos de ROS del vehículo real.

5.1. Resultados en simulador

En la primera prueba que se hizo en el simulador el auto se condujo a una velocidad constante de $v = 0.588\ m/s$ por una pista como la mostrada en la figura 5.5. El ancho de cada carril es de 40 cm por lo que el ancho del camino es de 80 cm.

Las curvas que se observan tienen un radio exterior de 1.8 m y un radio interior de 1.0 m. A los parámetros definidos en el modelo matemático del capítulo anterior se les asignaron los valores: $L_h = 0.5\ m$, $L = 0.26\ m$ y $l = 0.12\ m$.

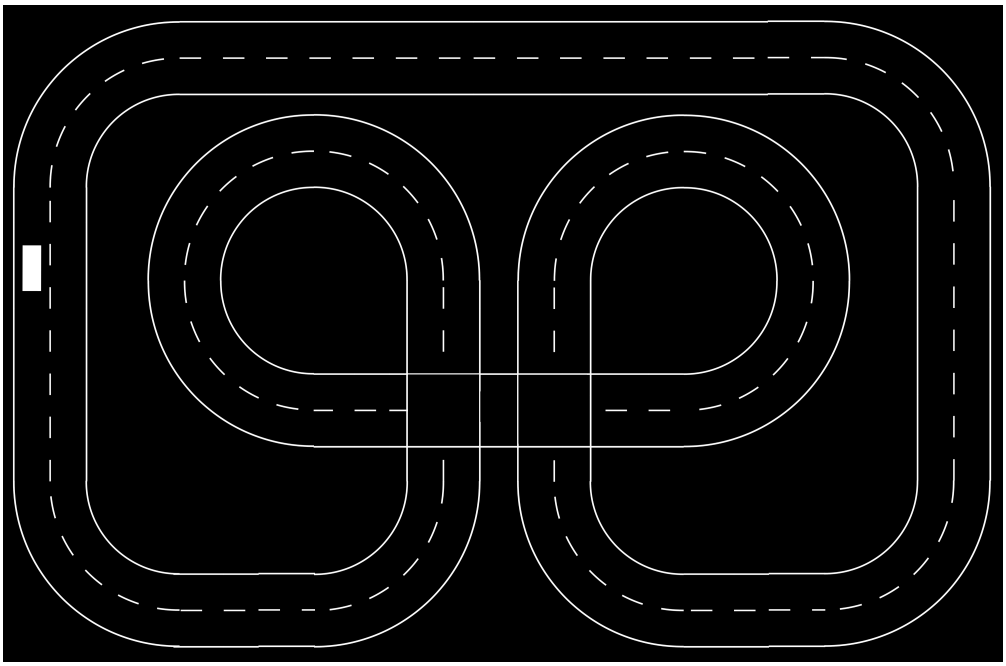

Figura 5.5: Pista donde se llevaron a cabo las pruebas en el simulador.

Utilizando el controlador 4.9 del capítulo anterior, con una ganancia $K_x = 3.0$, se obtuvieron las señales mostradas en la figura 5.6. La señal azul corresponde al error de posición e_x, la señal en rojo es el error de orientación θ y la señal amarilla es la señal de control $90° - st$. Como se puede observar existe una relación proporcional entre las tres señales así que una estrategia de control lineal es posible. Utilizando este controlador e_x se mantuvo entre 6 y -15 cm mientras que θ estuvo entre 3.81° y 45°; sin embargo en las tres señales se observan sobretiros. Aunque el automóvil pudo completar con éxito todo el recorrido y se mantuvo dentro de su carril se observó que en las curvas el auto roza la línea central del camino.

Por otro lado, la gráfica 5.7 muestra las mismas señales con los mismos colores, usando el controlador 4.10 con las ganancias $K_x = 3.0$ y $K_\theta = 0.25$. Como se puede observar el desempeño de este controlador es muy similar al anterior ya que mantiene acotados los errores entre los mismos valores y se observan los mismos patrones. Aunque con esta ley de control también se pudo recorrer todo el camino sin salirse del carril, también se observó que el auto rozaba la línea central del camino al tomar las curvas.

Con el controlador basado en Campos Potenciales de la ecuación 4.11 se obtuvieron las señales mostradas en la figura 5.8. Las ganancias asignadas a este controlador son: $K_+ = 0.5$, $K_- = 2.0$, $K_x = 2.0$ y $K_\theta = 0.1$. A diferencia de los resultados anteriores se obtuvieron menos puntos atípicos en las tres gráficas, la señal de e_x aumentó de amplitud, la señal θ se encuentra en una región diferente y tiene un adelanto de 400 ms con respecto a la señal del error de posición.

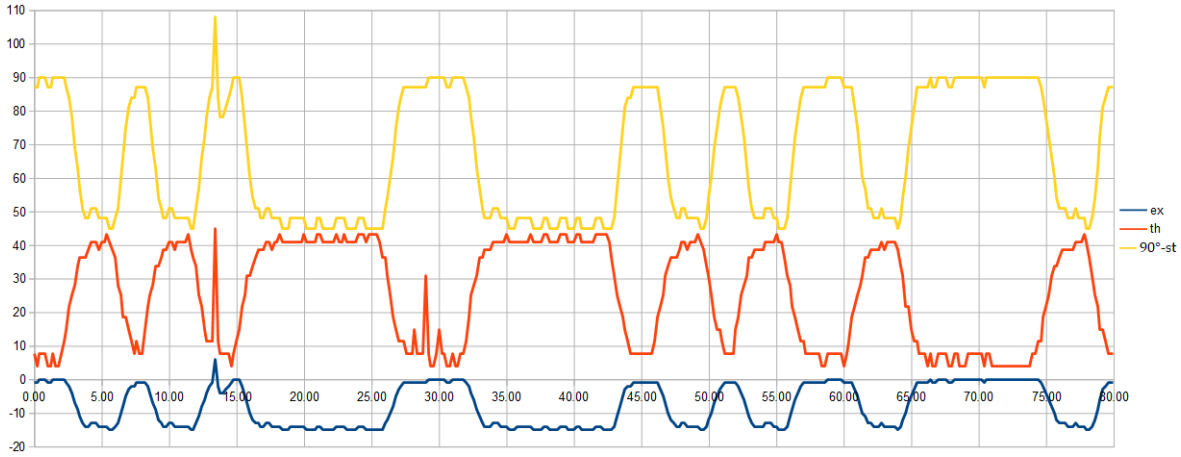

Figura 5.6: Señales del error de posición e_x (en azul), el error de orientación θ (en rojo) y la señal de control $90° - st$ (en amarillo). Estas se obtuvieron usando el controlador 4.9, conduciendo a una velocidad de 0.588 m/s.

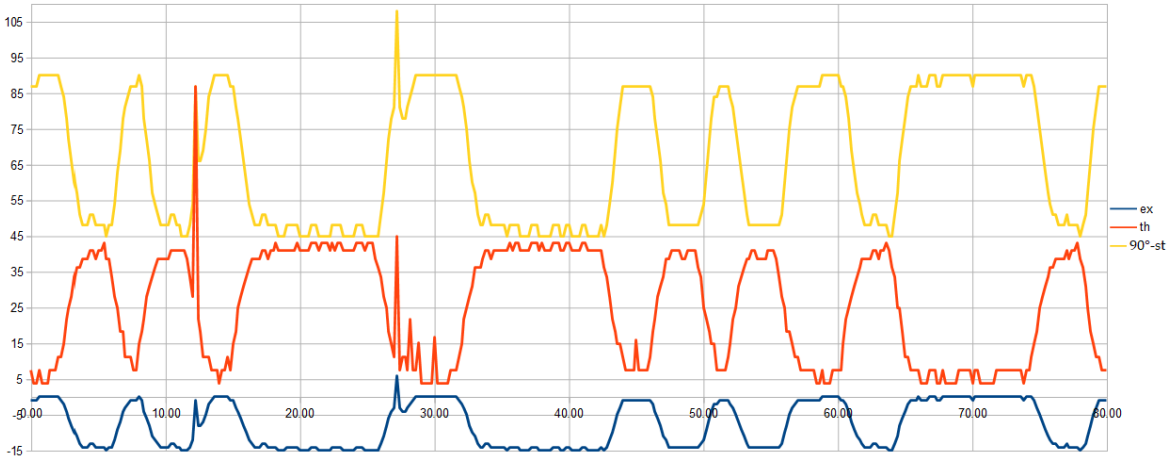

Figura 5.7: Señales del error de posición e_x (en azul), el error de orientación θ (en rojo) y la señal de control $90° - st$ (en amarillo). Estas se obtuvieron usando el controlador 4.10, conduciendo a una velocidad de 0.588 m/s.

Además de completar una vuelta por todo el camino, el vehículo también se mantuvo en el centro del carril con ínfimas desviaciones al tomar las curvas. La razón por la que se existe un aumento en el error de posición e_x, a pesar del buen desempeño observado, se debe a que la cámara observa el camino 50 cm adelante del referencial de navegación $\{N\}$ (el centro del eje de las llantas traseras). Así pues, a las señales e_x y θ se les puede interpretar como los errores de posición y orientación futuros; de mantenerse conduciendo el vehículo con el mismo ángulo de orientación st.

Usando el controlador óptimo se obtuvieron señales más suaves y con pocos puntos atípicos; tal y como se observa en la figura 5.9. Así como con el controlador basado en campos potenciales, también se observó un aumento en la amplitud de la señal del error de posición; sin embargo con este esquema se pudo conducir el vehículo por todo el circuito sin alejarse del centro del carril. Para resolver la ecuación de Ricatti y calcular el vector de ganancias se utilizó el programa mostrado al final del apéndice C. Usando este programa con $R = 5.85$ y $Q = 0.015I_{2\times2}$, se obtuvieron las ganancias $K_x 0.046259$ y $K_\theta = 0.140362$.

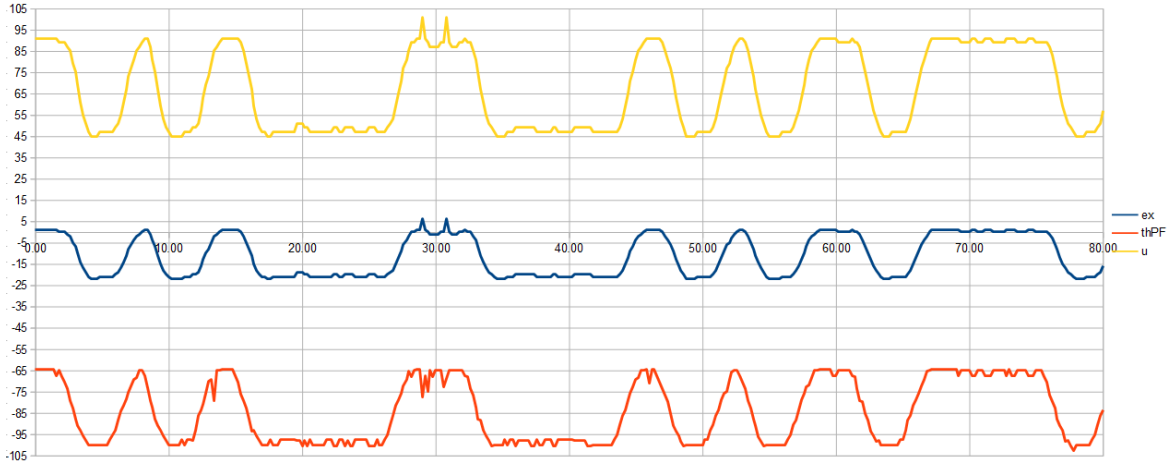

Figura 5.8: Señales del error de posición e_x (en azul), el error de orientación θ (en rojo) y la señal de control $90° - st$ (en amarillo). Estas se obtuvieron usando el controlador 4.11, conduciendo a una velocidad de 0.588 m/s.

Para implementar el "*preview controller*" se utilizaron como referencias futuras los ángulos θ_1 y θ_2; medidos del mismo modo que θ pero a una altura de 30 y 45 *cm* respectivamente, sobre la base de la imagen sin perspectiva. De este modo el término conocido como "*preview compensations*" de la ecuación 4.13 toma la forma: $f_{1x}x_{ref} + f_{1\theta}\theta_1 + f_{2x}x_{ref} + f_{2\theta}\theta_2$; siendo $f_{1x} = 0.000541$, $f_{1\theta} = 0.000968$, $f_{2x} = 0.000614$ y $f_{2\theta} = 0.000895$ las ganancias usadas para ponderar las referencias futuras. Estas ganancias se calcularon usando el mismo programa del apéndice C, utilizado por el controlador óptimo para resolver la ecuación de Ricatti.

El desempeño del "*preview controller*" es prácticamente el mismo que el del controlador óptimo; tal y como se observa en las gráficas de la figura 5.10.

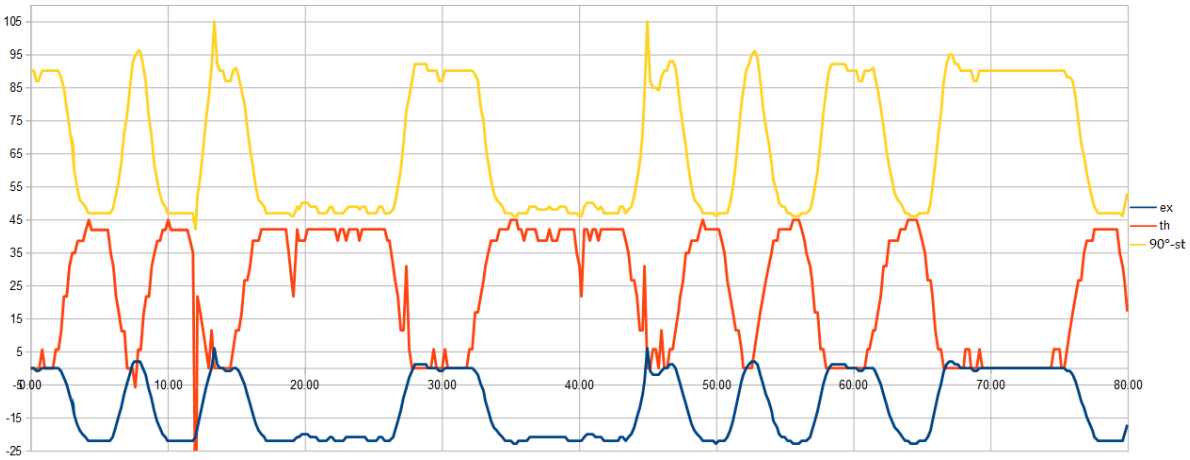

Figura 5.9: Señales del error de posición e_x (en azul), el error de orientación θ (en rojo) y la señal de control $90° - st$ (en amarillo). Estas se obtuvieron usando el controlador 4.12, conduciendo a una velocidad de 0.588 m/s.

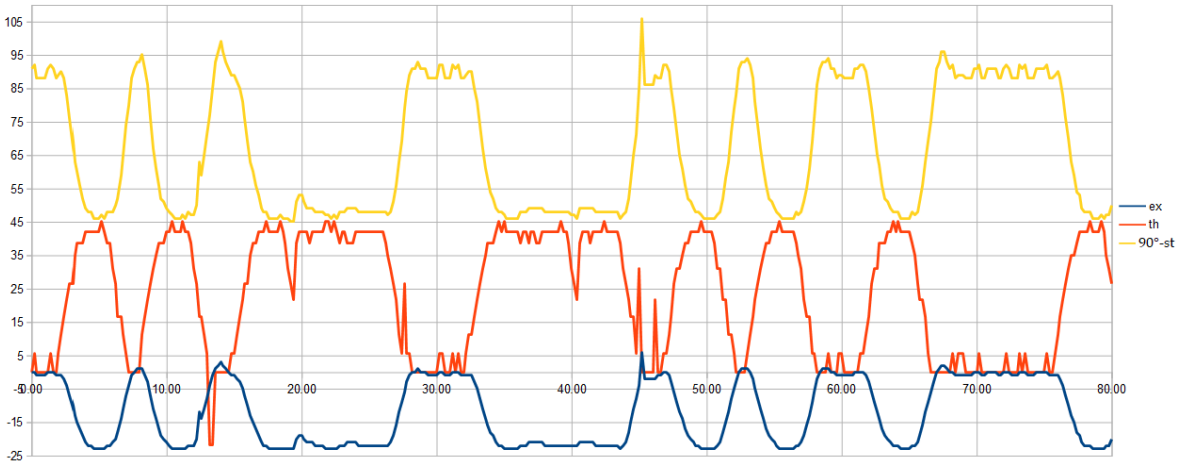

Figura 5.10: Señales del error de posición e_x (en azul), el error de orientación θ (en rojo) y la señal de control $90° - st$ (en amarillo). Estas se obtuvieron usando el controlador 4.13, conduciendo a una velocidad de 0.588 m/s.

En la segunda prueba realizada en el simulador, el auto se condujo a una velocidad constante de 0.970 m/s y se usaron los controladores 4.11, 4.12 y 4.13. Sus respectivas gráficas están en las figuras 5.11, 5.12 y 5.13.

0.97 m/s es la velocidad máxima a la que se pudo conducir el auto en el simulador y aunque se observaron pequeños sobretiros durante el cambio de dirección, con los tres controladores se pudo mantener al vehículo dentro de su carril en todo momento.

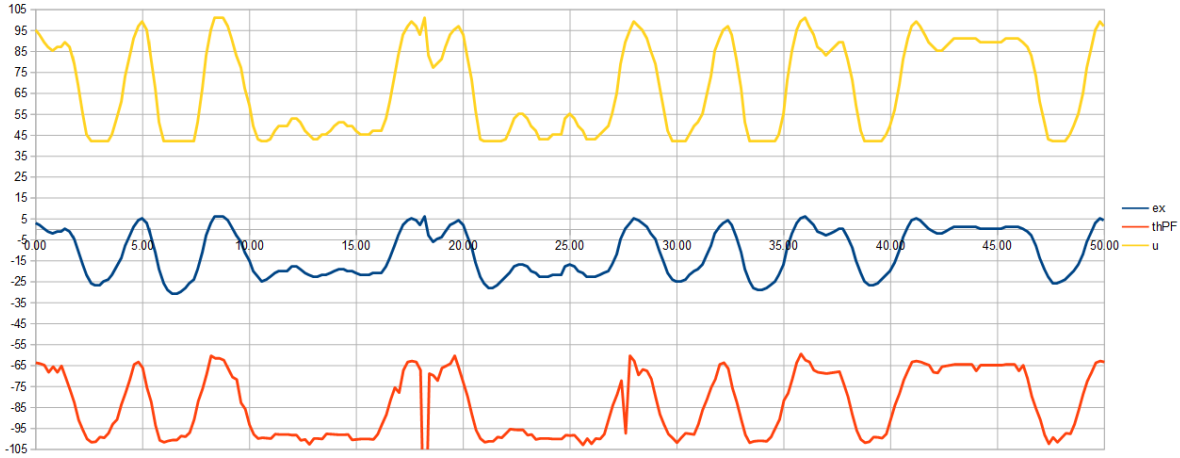

Figura 5.11: Señales del error de posición e_x (en azul), el error de orientación θ (en rojo) y la señal de control $90° - st$ (en amarillo). Estas se obtuvieron usando el controlador 4.11, conduciendo a una velocidad de 0.970 m/s.

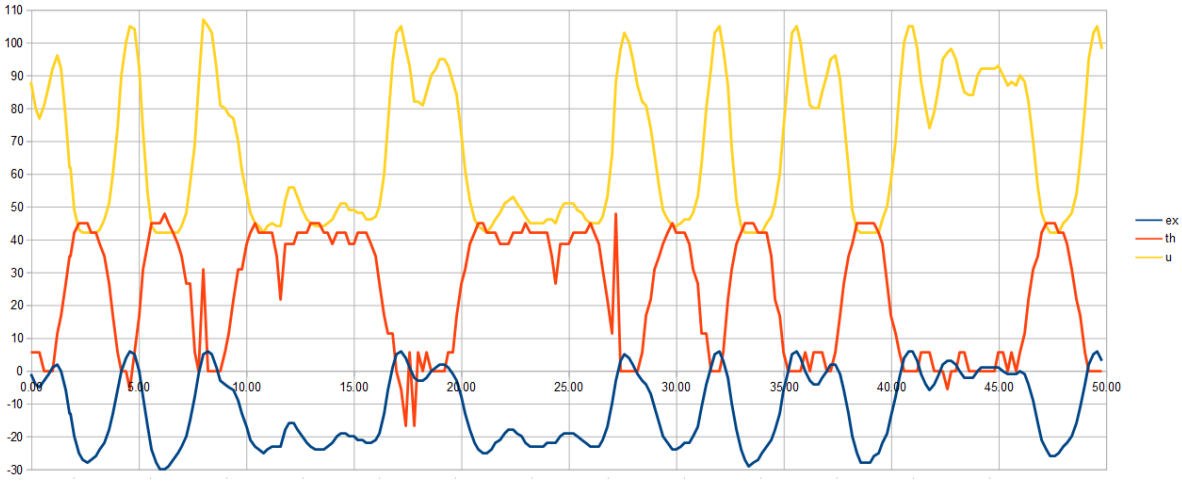

Figura 5.12: Señales del error de posición e_x (en azul), el error de orientación θ (en rojo) y la señal de control $90° - st$ (en amarillo). Estas se obtuvieron usando el controlador 4.12, conduciendo a una velocidad de 0.970 m/s.

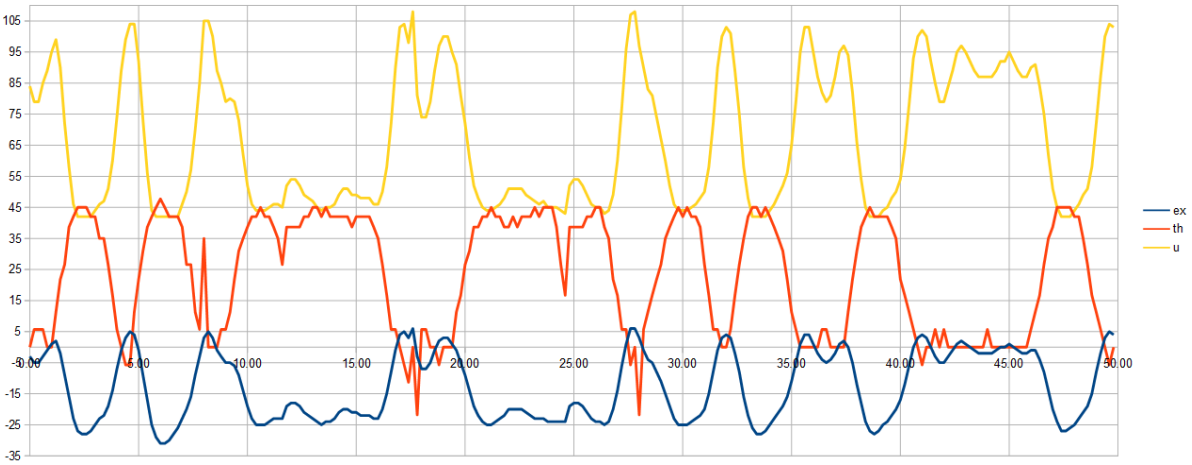

Figura 5.13: Señales del error de posición e_x (en azul), el error de orientación θ (en rojo) y la señal de control $90° - st$ (en amarillo). Estas se obtuvieron usando el controlador 4.12, conduciendo a una velocidad de 0.970 m/s.

5.2. Resultados en el vehículo real

La figura 5.14 muestra las dimensiones de la pista por la que se condujo el vehículo real para hacer las pruebas experimentales. La velocidad de avance del auto se mantuvo entre 0.6 y 0.75 m/s; y los parámetros Lh, L y l son los mismos que en el simulador.

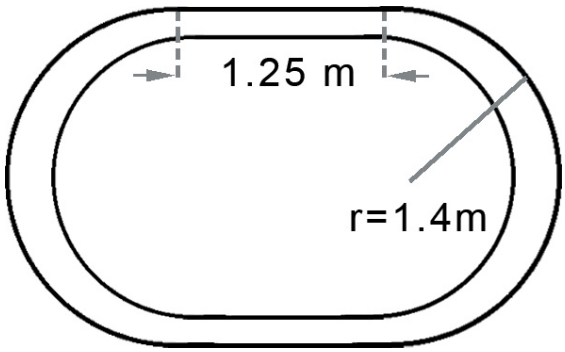

Figura 5.14: Pista por donde se condujo el vehículo AutoNOMOS.

Las señales en la figura 5.15 muestran a los errores de posición, de orientación y la señal de control $90° - st$; obtenidas conduciendo el auto usando la ley de control 4.9 con una ganancia $K_x = 0.05$. Para minimizar el efecto del ruido y facilitar su interpretación, estas señales se filtraron con un filtro pasabajos con una frecuencia de corte de 2 Hz. Con este controlador el automóvil recorrió la pista dos veces en 33 s; pero se observaron muchos sobretiros durante el cambio de dirección.

Del mismo modo, en la figura 5.16 se muestran los resultados obtenidos con el controlador 4.10 al recorrer la pista 3 veces, teniendo como ganancias $K_x = 0.05$ y $K_\theta = 0.0015$. Tal y como se puede deducir de la gráfica de la señal de control $90° - st$, en este caso los sobretiros eran de mayor amplitud. Con este controlador se condujo el auto hasta completar tres vueltas a la pista. Tanto con la ley de control 4.9 como con la 4.10, el error de posición se mantuvo entre 1 y -29 cm; y el error de orientación entre $-54°$ y $21°$.

Por otro lado, las gráficas mostradas en la figura 5.17 muestran los resultados obtenidos al utilizar el controlador basado en campos potenciales 4.11 al recorrer 3 veces la pista. Las ganancias asignadas a este controlador son: $K_- = 3.5$, $K_+ = 0.25$, $K_x = 0.025$ y $K_\theta = 0.12$. Con este esquema se observó un mejor desempeño que los anteriores al presentar un cambio de dirección más suave, obtener señales con menos ruido y mantenerse más cerca del centro del carril; sin embargo presenta un incremento en la amplitud de la señal del error de orientación, oscilando entre $-10°$ y $73°$ (eliminando los puntos atípicos).

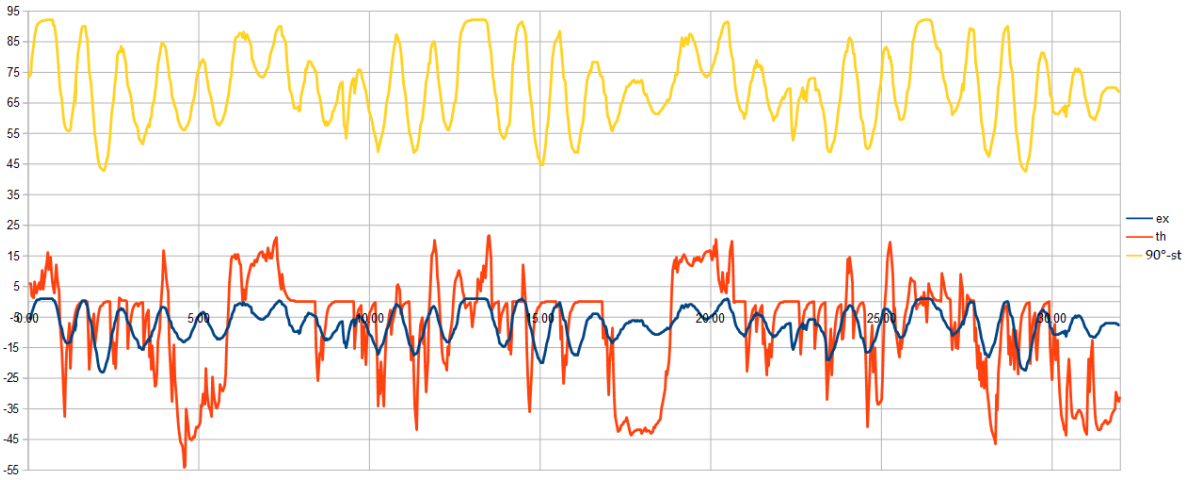

Figura 5.15: Señales del error de posición e_x (en azul), el error de orientación θ (en rojo) y la señal de control $90° - st$ (en amarillo). Estas señales se obtuvieron usando el controlador 4.9 y se pasaron por un filtro pasabajos con una frecuencia de corte de 2 Hz.

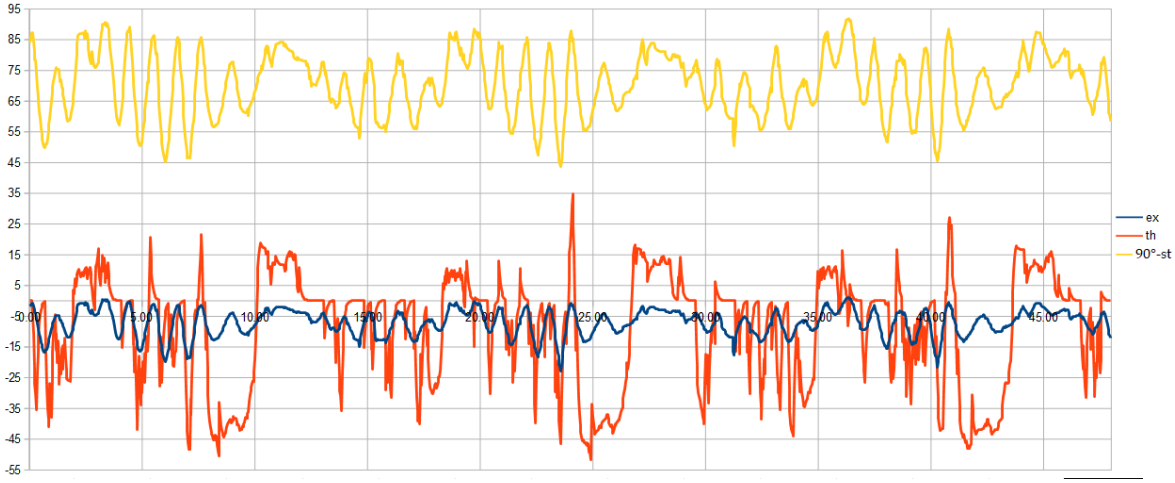

Figura 5.16: Señales del error de posición e_x (en azul), el error de orientación θ (en rojo) y la señal de control $90° - st$ (en amarillo). Estas señales se obtuvieron usando el controlador 4.10 y se pasaron por un filtro pasabajos con una frecuencia de corte de 2 Hz.

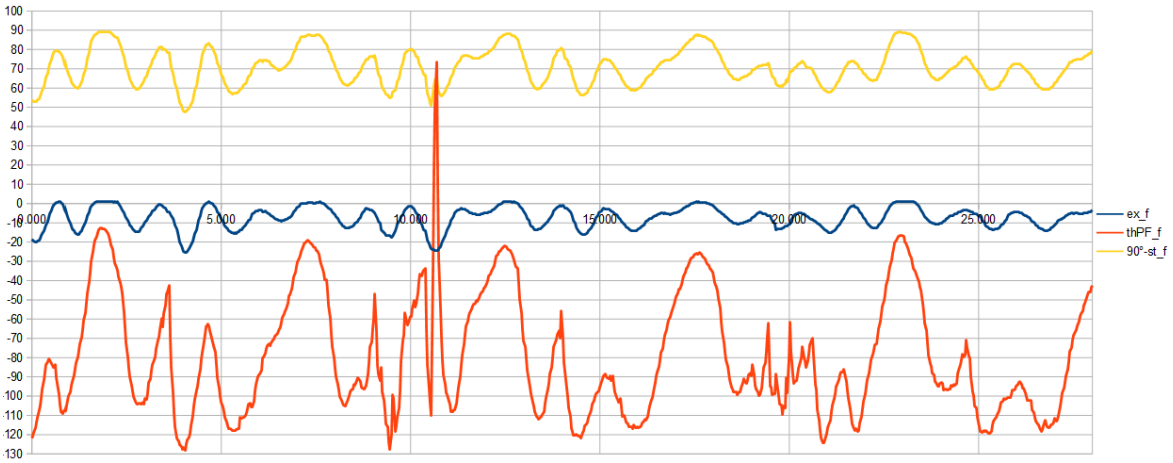

Figura 5.17: Señales del error de posición e_x (en azul), el error de orientación θ (en rojo) y la señal de control $90° - st$ (en amarillo). Estas señales se obtuvieron usando el controlador 4.11 y se pasaron por un filtro pasabajos con una frecuencia de corte de 2 Hz.

Respecto al controlador óptimo, los resultados se muestran en la figura 5.18. De igual modo que con el controlador basado en campos potenciales, con esta ley de control se obtuvieron cambios de dirección más suaves, señales con menos ruido, el auto pudo completar 3 vueltas a la pista sin alejarse mucho del centro del carril y la señal del error de orientación se mantuvo entre $-27°$ y $48°$.

Ajustando $R = 12$ y $Q = 0.015I_{2\times2}$, las ganancias calculadas para este controlador son $K_x = 0.035114$ y $K_\theta = 0.123157$; y se ajustaron utilizando el mismo programa que se usó en el caso el simulador.

Finalmente, el controlador con el mejor desempeño en los experimentos fue el "*preview controller*" cuyas señales de salida se muestran en la figura 5.19. Más claramente que en los experimentos anteriores, en la señal del error de orientación se pueden observar seis cambios en la tendencia de la señal. Estos representan las dos curvas que tiene el camino al ser recorrido tres veces. También se observa una disminución en la amplitud de las señales de error.

Las ganancias asignadas para este controlador son $K_x = 0.035114$, $K_\theta = 0.123157$, $f_{1x} = 6.139 \times 10^{-5}$, $f_{1\theta} = 1.198 \times 10^{-4}$, $f_{2x} = 6.357 \times 10^{-5}$ y $f_{2\theta} = 1.181 \times 10^{-4}$.

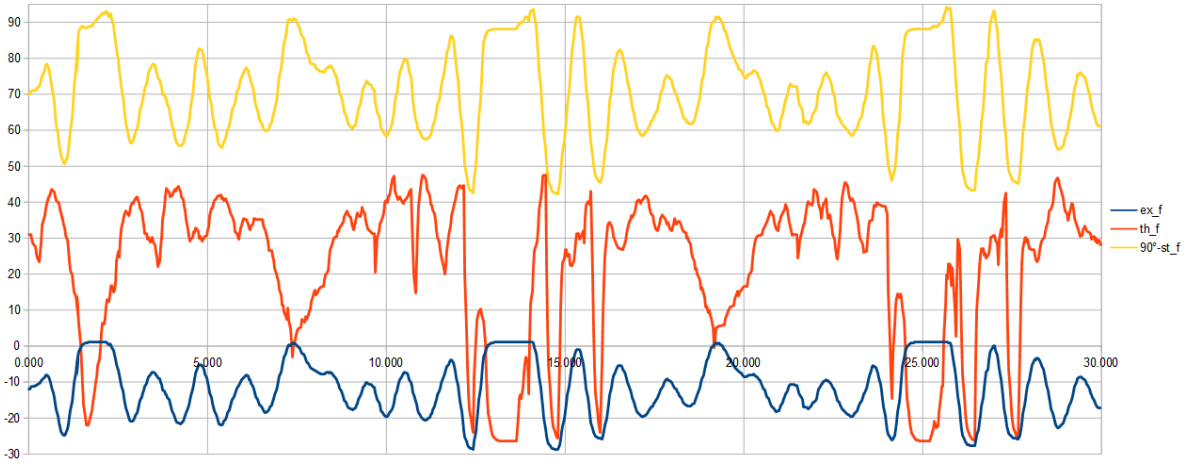

Figura 5.18: Señales del error de posición e_x (en azul), el error de orientación θ (en rojo) y la señal de control $90° - st$ (en amarillo). Estas señales se obtuvieron usando el controlador 4.12 y se pasaron por un filtro pasabajos con una frecuencia de corte de 2 Hz.

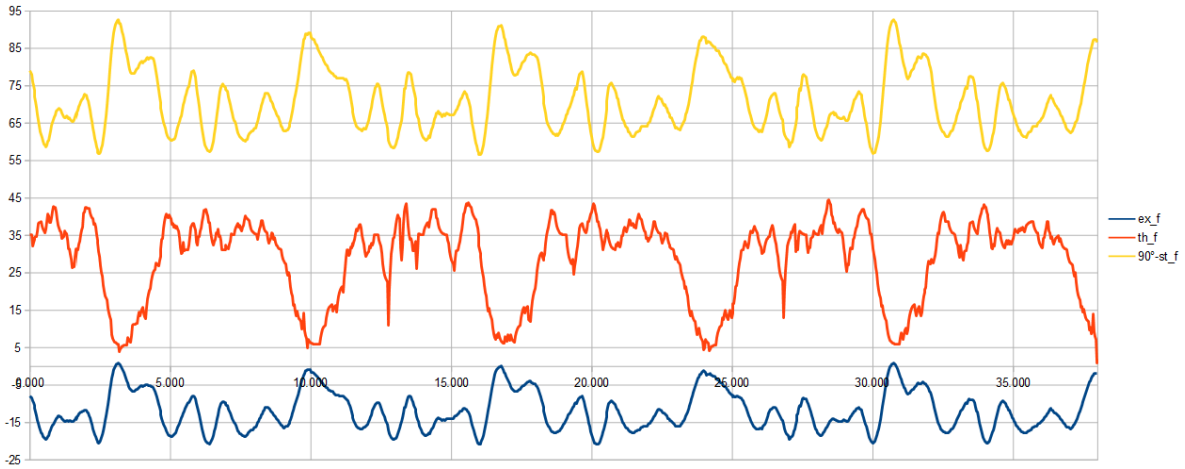

Figura 5.19: Señales del error de posición e_x (en azul), el error de orientación θ (en rojo) y la señal de control $90° - st$ (en amarillo). Estas señales se obtuvieron usando el controlador 4.13 y se pasaron por un filtro pasabajos con una frecuencia de corte de 2 Hz.

Particularmente, podemos observar en la figura 5.20 la comparación entre la señal del ángulo de dirección $90° - st$ y la velocidad angular del ángulo de guiñada gzf_N medida con el sensor inercial a bordo del auto. Dichas señales están normalizadas y la señal gzf_N fue filtrada con un filtro pasa altos con una frecuencia de corte de 0.5 Hz. Las señales se obtuvieron con el controlador 4.13. De aquí podemos observar un pequeño desface entre las señales debido al tiempo de respuesta entre la consigna del ángulo de dirección deseado y la velocidad angular obtenida. El modelo matemático planteado en el capítulo anterior no modela la dinámica de este subsistema pero se espera compensar ésto haciendo una identificación de parámetros y utilizando la velocidad angular del ángulo de guiñada como una señal de salida en trabajos futuros.

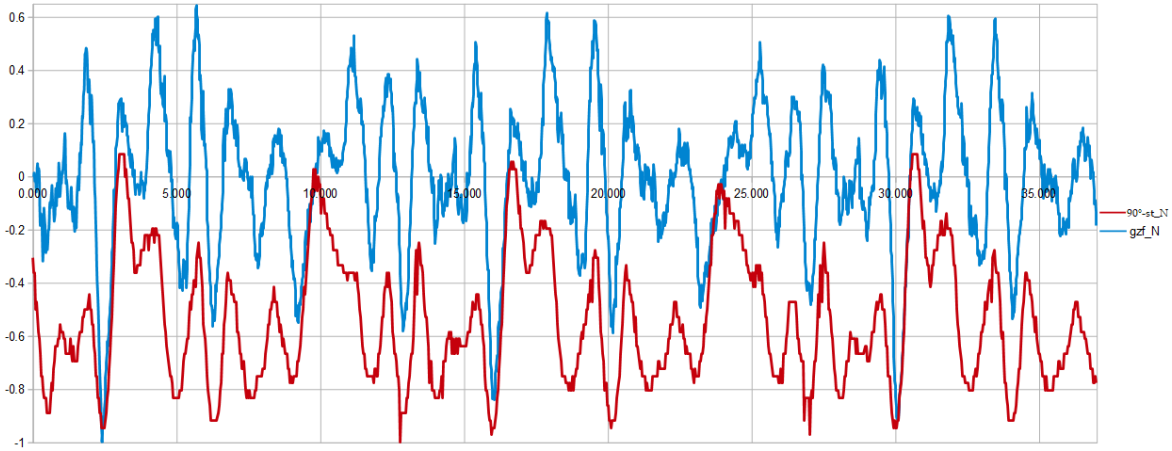

Figura 5.20: Comparación entre la señal de control $90° - st$ (en rojo) y la señal de la velocidad angular del ángulo de guiñada gzf_N (en azul). Estas señales están normalizadas y la señal gzf_N se filtró con un filtro pasa altos con una frecuencia de corte de 0.5 Hz. Las señales se obtuvieron con el controlador 4.13.

Capítulo 6

Conclusiones y trabajo futuro

En el marco del proyecto de tesis aquí reportado se realizaron los siguientes trabajos tanto tecnológicos como de investigación: *i)* acondicionamiento de la plataforma AutoNOMOS, *ii)* desarrollo de modelos matemáticos de dicha plataforma, *iii)* desarrollo de diversos esquemas de control de la dirección del vehículo AutoNOMOS, y *iv)* desarrollo del sistema de percepción visual de dicho vehículo.

i) El acondicionamiento de la plataforma consistió en sustituir la tarjeta controladora de los motores del AutoNOMOS, que estaba descompuesta, por un circuito de diseño propio; para poder controlar el servomotor y el motor sin escobillas, utilizando el software de ROS. Para ello fue necesario estudiar las técnicas de control utilizadas para conmutar las fases de un motor sin escobillas y así darle tracción a las ruedas.

ii) En la bibliografía reportada en el estado del arte los modelos matemáticos del automóvil que se deducen resultan no lineales o los estados no se pueden medir desde un referencial de navegación o son lineales pero tienen asociados parámetros que son difíciles de medir experimentalmente [14-44].

En este trabajo se propone el modelo matemático lineal de las ecuaciones 4.5 y 4.6 (hasta donde se conoce no ha sido reportado en algún otro trabajo); lo que nos permite aplicar técnicas de control clásicas como el regulador cuadrático lineal o el "*preview controller*" cuya estabilidad exponencial está demostrada [84].

Otras ventajas de este modelo es que sus estados se pueden medir utilizando la cámara a bordo, es simple y es fácil de implementar usando el modelo discreto de las ecuaciones 4.7 y 4.8. Sin embargo, las limitaciones que este modelo presenta son: que no toma en cuenta la dinámica que hay entre el servomotor y el mecanismo que hace cambiar al auto de dirección; y que solo se puede considerar al modelo como cinemático ya que no toma en cuenta las fuerzas y los pares que aparecen en los neumáticos y por lo tanto supone que el ángulo de desviación entre la velocidad de avance y el eje longitudinal del auto es cero. Así pues habría que replantear las ecuaciones propuestas antes de poderlo implementar en un vehículo normal.

iii) Respecto a las técnicas de control utilizadas, tanto en el simulador como en los experimentos con el vehículo real, las leyes de control 4.9 y 4.10 son las que tuvieron el menor desempeño al conducir el AutoNOMOS ya que: se observaron sobretiros durante los cambios de dirección, no existe un criterio para sintonizar las ganancias de cada uno y la estabilidad de 4.10 esta condicionada por el valor de los estados, tal y como se discutió en el capítulo 4. El mejor desempeño al conducir el auto se obtuvo utilizado el controlador basado en campos potenciales, el controlador óptimo y el "*preview controller*". Más aún, en el simulador con estos tres controladores no solo se pudo mantener el vehículo en el centro del carril a 0.588 m/s sino que también se pudo conducir más rápido hasta alcanzar 0.970 m/s.

Con el controlador basado en campos potenciales se observó que la señal del error de orientación $\theta(t)$ tenía un adelanto de 400 ms con respecto a la señal del error de posición e_x; esto supondría una ventaja ya que implica una disminución en el tiempo de retardo que hay entre la entrada y la salida del sistema; sin embargo este desfase solo se pudo observar en el simulador. En los experimentos se puede resaltar el hecho de que los cambios en la dirección del auto presentan menos sobretiros, lo que resulta en señales de salida más suaves. Entre las desventajas que presenta esta ley de contrrol está que no se puede garantizar la estabilidad global del sistema y se requiere el ajuste de 4 parámetros: K_+ y K_- que son las ganancias asociadas a los potenciales atractivos y repulsivos; y K_x y K_θ que son las ganancias del controlador.

Por otro lado, tanto el controlador óptimo como el "*preview controller*" sólo requieren el ajuste de los parámetros R y Q con los cuales, en conjunto con las matrices A, B y C del sistema; se pueden calcular las ganancias de las ecuaciones 4.12 y 4.13. Dichos valores garantizan la estabilidad exponencial del sistema en lazo cerrado y minimizan sus respectivos criterios de desempeño definidos en el capítulo 4. Aunque en el simulador estos dos esquemas tienen el mismo desempeño, en los experimentos se observó que el "*preview controller*" generaba señales de salida más suaves, con menos ruido y se distinguía mejor la tendencia en la señal de θ.

iv) Gracias al sistema de visión desarrollado es posible hacer mediciones del camino, utilizando las imágenes de la cámara a bordo, y sin preocuparse por las distorsiones de la perspectiva. Todo esto es gracias a la transformación de homografía la cual ha resultado ser una herramienta muy útil que nos permite reconstruir la escena, compensando las distorsiones que la imagen original pueda tener.

Como trabajo futuro se propone un estudio detallado de la dinámica que hay entre el servomotor y el mecanismo con el que se cambia el ángulo de dirección del auto; para que se pueda utilizar la velocidad angular del ángulo de guiñada como señal de salida. Así la conducción del automóvil no nada más dependerá del sistema de visión y el algoritmo de control será más robusto.

Otro de los experimentos pendientes, es el de usar otras estrategias de control más robustas como por ejemplo: el control con rechazo activo de perturbaciones, la teoría de H_∞ e incluso utilizar redes neuronales y controladores difusos para resolver el problema de la conducción autónoma.

Finalmente, para avanzar en la investigación en este tema, se podría incursionar en los otros problemas relacionados con la conducción autónoma como son: la planificación de trayectorias, el reconocimiento de señalizaciones de tránsito, realización de maniobras como la evasión de obstáculos, el rebase de otros vehículos y el estacionamiento autónomo; entre otras.

Apéndice A

Circuitos diseñados para controlar los motores del AutoNOMOS

Para amplificar y digitalizar la señal de salida de los sensores de efecto Hall del motor BLDC DB28M01, se utilizó el circuito mostrado en la figura A.1. Como se puede observar, en todos se usó un amplificador no inversor con una ganancia de 3.2. El capacitor se utilizó como filtro pasa bajos. Las señales A, B y C se conectaron al circuito mostrado en A.2. Este segundo circuito se diseñó basándose en la tabla 1.1 de la introducción.

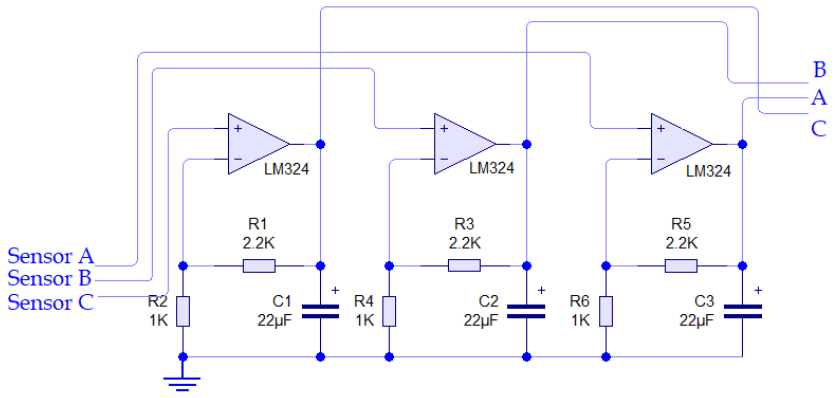

Figura A.1: Circuito diseñado para amplificar y digitalizar la señal de los sensores de efecto Hall del motor BLDC DB28M01. El amplificador operacional usado fue el LM324.

Ya que la tabla de verdad que hace girar al motor en el sentido antihorario es igual a la tabla 1.1, excepto porque en lugar de los bits A, B y C se usan los bits ¬A, ¬B y ¬C [74]; para hacer el cambio de giro con un solo bit, se utiliza una señal TTL denominada Dir, que sale del Pin D8 del Arduino UNO y llega a la compuerta XOR 7486 mostrada en la figura A.2. Por otro lado, para controlar la velocidad del motor BLDC, en vez de una señal binaria se le envía a los transistores $Q0, Q1, \cdots, Q5$ una señal de pulso modulado denominada PWM. Esta señal sale del pin D6 del Arduino UNO y llega a la compuerta AND 7411; de este modo, el ciclo de trabajo de la señal define la velocidad a la que gira el motor, independientemente de su sentido de giro o la fase en la que se encuentre.

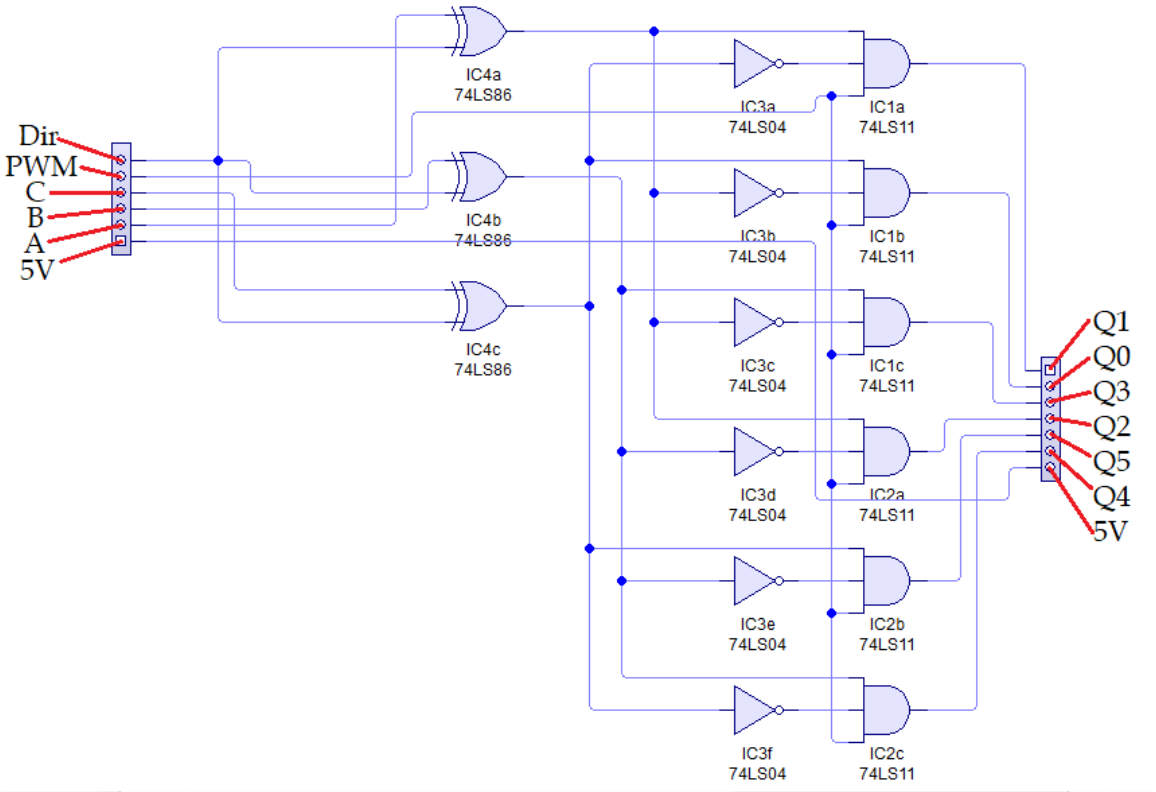

Figura A.2: Circuito diseñado para conmutar las fases del motor BLDC; utilizando las señales digitalizadas de los sensores de efecto Hall como señal de realimentación. Los 5 V se usaron para alimentar las compuertas lógicas. Se omiten las conexiones a tierra.

Finalmente, para suministrar el voltaje a cada fase, se utilizaron seis optoacopladores 4N35, seis transistores TIP122 y seis diodos 1N4001; conectados tal y como se muestran en la figura A.3.

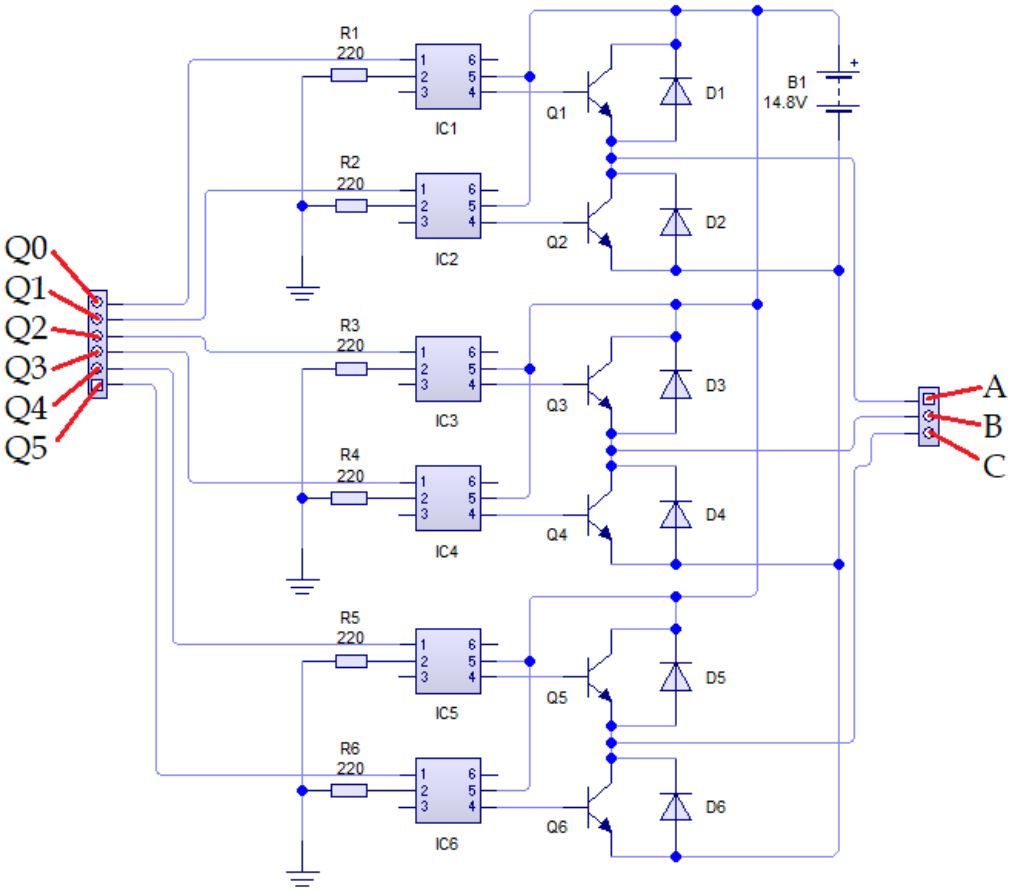

Figura A.3: Etapa de potencia que suministra el voltaje a las fases del motor BLDC.

El siguiente código se utilizó para programar el Arduino UNO como un suscriptor de ROS y para controlar el movimiento de los motores del auto:

```
#include <ros.h>
#include<Servo.h>
#include <std_msgs/Int16.h>
#include <std_msgs/Int32.h>

//V. globales
int OnOff=0;
```

```
int pwmPIN=6;
int dirPIN=8;
int servoPIN=10;
long vel=0;
int pwm=0;
Servo servo;

//*************************************
//*************************************
void st_cb(const std_msgs::Int16& STdata){
  //Recibe un valor de 0 o 1
  OnOff = STdata.data;
}
//*************************************
//*************************************
void servo_cb(const std_msgs::Int16& Sdata){
  //Recibe un angulo que va de 0-180
  servo.write(Sdata.data);
}
//*************************************
//*************************************
void brushless_cb( const std_msgs::Int32& Vdata){
        //Lee el topico /manual_control/speed
    vel = Vdata.data;
    pwm = map(abs(vel),0,1000,0,255);
}
//*************************************
//*************************************
// Instancias de ROS
ros::NodeHandle motor_control;
std_msgs::Int32 Vdata;
std_msgs::Int16 Sdata;
//Se suscribe al topico:/manual_control/steering
ros::Subscriber<std_msgs::Int16>
STsub("/manual_control/stop_start", st_cb);
//Se suscribe al topico:/manual_control/steering
ros::Subscriber<std_msgs::Int16>
Ssub("/manual_control/steering", servo_cb);
//Se suscribe al topico:/manual_control/speed
ros::Subscriber<std_msgs::Int32>
Vsub("/manual_control/speed", brushless_cb);
//*************************************
//*************************************
void move() {
//Emergency Stop_Start
  if (OnOff==1){
      if (vel>0){ //Adelante
        digitalWrite(dirPIN,LOW);
        analogWrite(pwmPIN,pwm);
      }
      if (vel<0){ //Atras
        digitalWrite(dirPIN,HIGH);
```

```
        analogWrite(pwmPIN,pwm);
      }
      if (vel==0){ //Detenido
        digitalWrite(dirPIN,LOW);
        analogWrite(pwmPIN,0);
      }}
  else{
    vel=0;
    pwm=0;
    digitalWrite(dirPIN,LOW);
    analogWrite(pwmPIN,0);
    servo.write(90);
    delay(5000);
  }
}
//************************************
//************************************
void setup() {
//PWM frequency for D5 & D6
// for PWM frequency of 62500.00 Hz
//  TCCR0B = TCCR0B & B11111000 | B00000001;
// for PWM frequency of 7812.50 Hz
//  TCCR0B = TCCR0B & B11111000 | B00000010;
  pinMode(5, OUTPUT); //No se usa
  // Pin del pwm () y pines del control de direccion ()
  pinMode(pwmPIN, OUTPUT);
  pinMode(dirPIN, OUTPUT);

  pinMode(servoPIN, OUTPUT);
  // attaches the servo on pin 6 to the servo object
  servo.attach(servoPIN);
  servo.write(90);

  motor_control.initNode();
  motor_control.subscribe(STsub);
  motor_control.subscribe(Ssub);
  motor_control.subscribe(Vsub);
}
//************************************
//************************************
void loop(){
  //Funcion que crea un bucle
  motor_control.spinOnce();
  move();
  //Delay de 1ms
  delay(1);
}
```

El circuito impreso mostrado en la figura A.4 está diseñado a partir del diagrama de la figura A.1 y se utiliza como un *shield* de Arduino UNO. Tiene pines para conectar los sensores de efecto Hall del motor BLDC, pines para conectar el servomotor y borneras para conectar la tierra y los 5V de alimentación. Por otro lado, el circuito impreso de la figura A.5 corresponde al diagrama de la figura A.2. Los pines de entrada son para la tierra, los 5V de alimentación y las señales digitalizadas de los sensores de efecto Hall. Los pines de salida son una tierra y los seis bits con los que se activan las fases del motor BLDC. Por último, el circuito impreso de la figura A.6 muestra la etapa de potencia cuyo diagrama se muestra en la figura A.3. Los pines de salida son las conexiones al motor BLDC y la bornera se conecta a los 14.8V que suministra la batería.

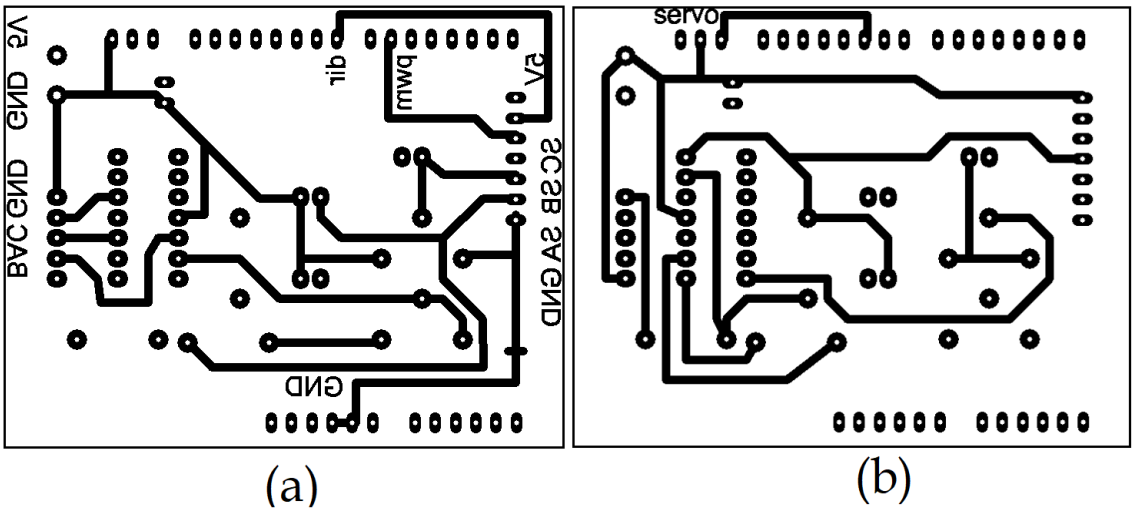

Figura A.4: Cara superior (a) e inferior (b) del circuito impreso diseñado a partir del diagrama de la figura A.1.

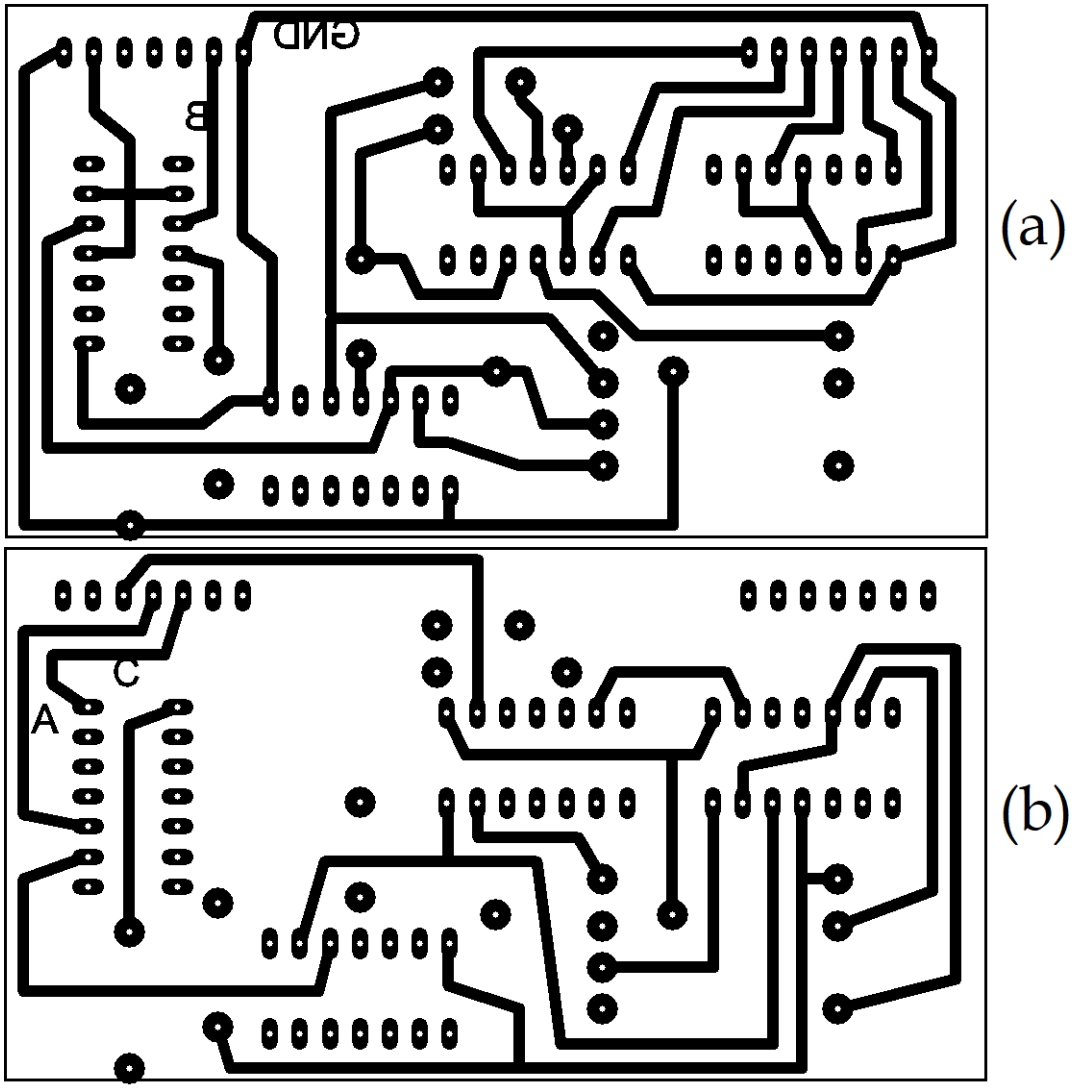

Figura A.5: Cara superior (a) e inferior (b) del circuito impreso diseñado a partir del diagrama de la figura A.2.

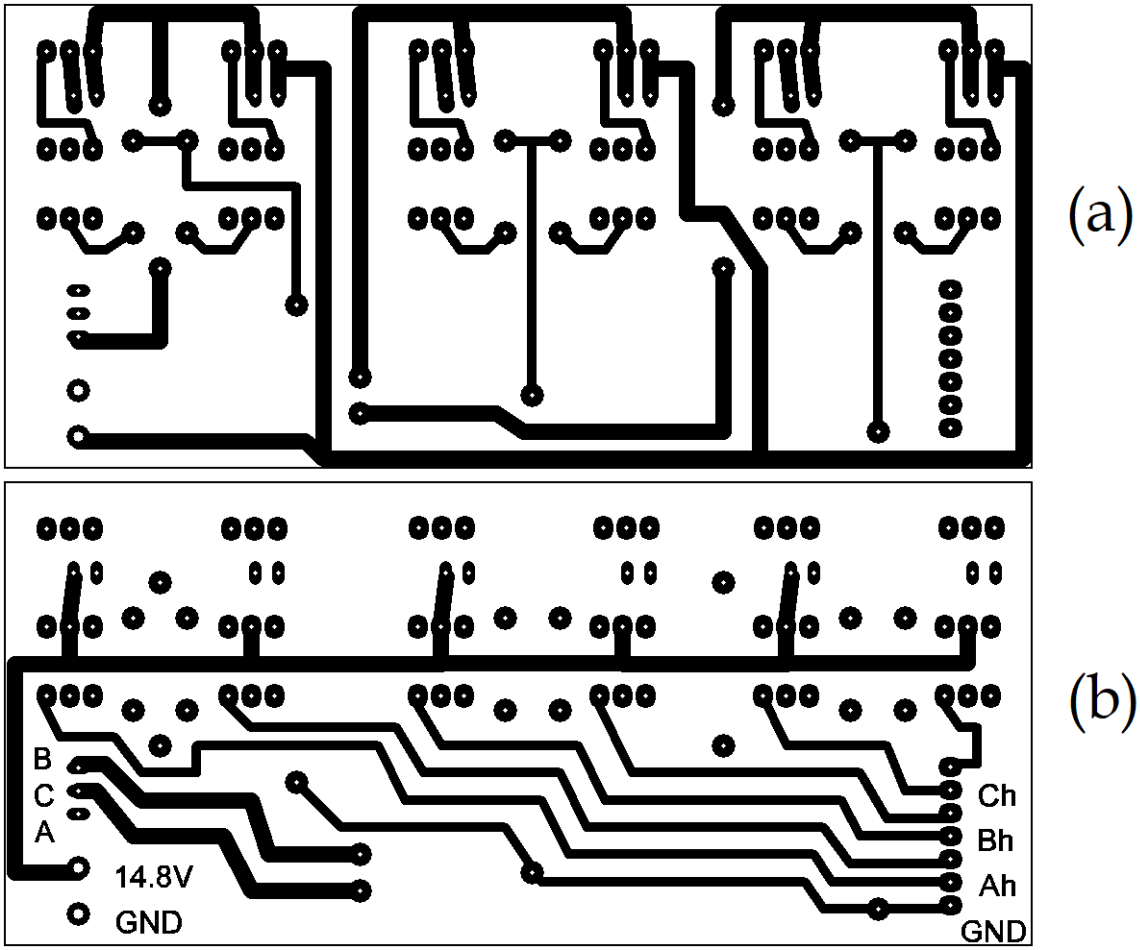

Figura A.6: Cara superior (a) e inferior (b) del circuito impreso diseñado a partir del diagrama de la figura A.3.

Apéndice B

Calculo de la matriz de homografía con Python

Para calcular la matriz de homografía que nos permita mapear los pixeles de una imagen, a puntos en el suelo sobre una imagen si perspectiva; se utilizó el siguiente programa escrito en Python 2.7, el cual utiliza la biblioteca de OpenCV. Este programa requiere una imagen de origen, cuatro puntos no co-lineales de la misma y su correspondiente posición en el suelo; medida desde algún referencial $\{T\}$. Con estos datos el programa calcula la matriz de homografía y le aplica la transformación a la imagen de origen para obtener una imagen de salida sin perspectiva; cuyo ancho mide 200 cm y de largo mide 300 cm. Tanto en el simulador como con la cámara real, se utilizó como imagen de origen una captura donde se muestra un rectángulo de dimensiones conocidas sobre el suelo. Se midió la posición de las esquinas en pixeles y se escribió su correspondiente posición en centímetros. La posición del referencial $\{T\}$ se fue variando hasta obtener una imagen de salida centrada.

```
import numpy as np
import cv2
import glob
import matplotlib.image as mpimg
from matplotlib import pyplot as plt
from numpy.linalg import inv
```

```
# Origen (Imagen del suelo)
img1 = cv2.imread('Camara_AutoNOMOS_REAL.png',1)
#*************************************
#*************************************
# Se definen 4 puntos en la imagen de origen.
# Las unidades son px.
p1_1 = [248,274]
p1_2 = [40,450]
p1_3 = [591,468]
p1_4 = [436,277]
P1 = np.concatenate(([[p1_1],[p1_2],[p1_3],[p1_4]]),axis=0)

# Se seleccionan 4 puntos en la imagen de destino, se escriben en cm
# y se usa como referencial al punto T.
T = [79,210]
p2_1 = np.add([0,0],T)
p2_2 = np.add([0,88.2],T)
p2_3 = np.add([39.5,88.2],T)
p2_4 = np.add([40.6,0],T)
P2 = np.array([p2_1,p2_2,p2_3,p2_4])
# Matriz de homografia
M, mask = cv2.findHomography(P1, P2, cv2.RANSAC,5.0)
print "Matriz de Homografia"
print M
#Obtenemos la imagen con correccion de pespectiva
imagenH = cv2.warpPerspective(img1, M, (200,300),
borderMode=cv2.BORDER_CONSTANT, borderValue=(125, 125, 125))
plt.imshow(imagenH, 'gray'), plt.show()
cv2.imwrite('Camara_AutoNOMOS_TMR_SIM_TIP.jpg',imagenH)
```

La figura B.1 muestra a la imagen de origen y la imagen sin perspectiva obtenidas con este programa. Las matrices de homografía tanto del simulador como de la cámara real son:

$$H_{SIM} = \begin{pmatrix} -7.90e-02 & -4.19e-01 & 1.28e+02 \\ 1.00e-02 & -1.46 & 4.00e+02 \\ 2.60e-05 & -4.15e-03 & 1.00 \end{pmatrix} \tag{B.1}$$

$$H_{REAL} = \begin{pmatrix} -1.03e-01 & -5.60e-01 & 1.40e+02 \\ 2.05e-02 & -1.88 & 4.05e+02 \\ 4.24e-05 & -5.50e-03 & 1.00 \end{pmatrix} \tag{B.2}$$

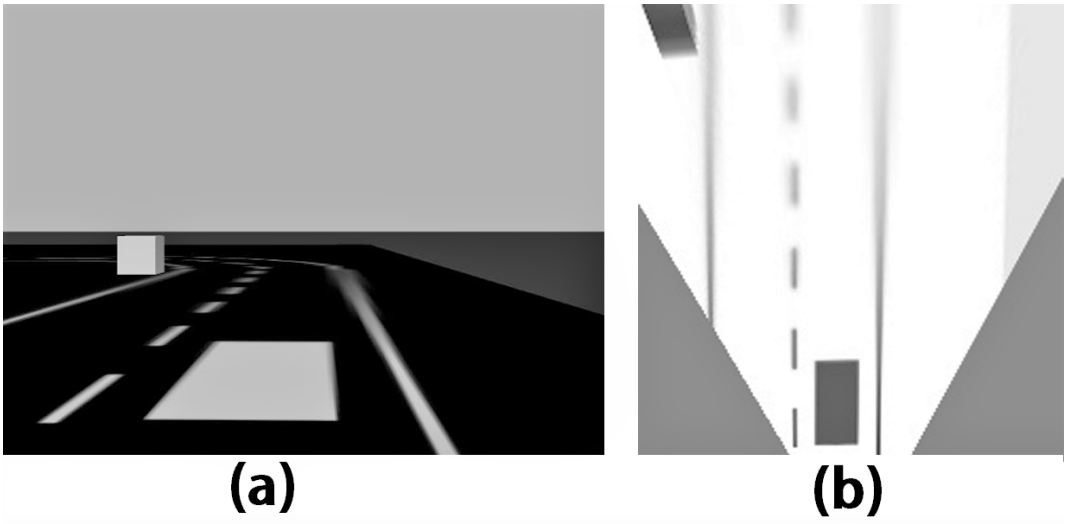

Figura B.1: (a)Imagen obtenida por la cámara a bordo del AutoNOMOS en un ambiente simulado. (b) Imagen sin perspectiva de la misma escena; los colores están invertidos para facilitar su visualización y las distancias en esta imagen se miden en cm.

Apéndice C

Código de los nodos programados en ROS

Con el siguiente programa se implementaron los controladores 4.9 y 4.10 en el simulador del vehículo AutoNOMOS, como nodos de ROS. Un código muy similar se usó para implementarlo en el vehículo real.

```
#! /usr/bin/env python
import rospy
import cv2
import numpy as np
from sensor_msgs.msg import Image
from cv_bridge import CvBridge, CvBridgeError
from std_msgs.msg import Int16, Int16MultiArray, Float32,
Float32MultiArray

# V. globales de la vision
x_ref = 120 #120
x1 = 0
x2 = 0
y1 = 0
y2 = 0
th = 0
x1_1 = 120
x2_1 = 120
th_1 = 0
#*****************************************
#*****************************************
#       ANALISIS Y PROCESAMIENTO DE IMAGENES
def callback_image(data1):
```

```
global x1
global x2
global y1
global y2
global th
global x1_1
global x2_1
global th_1
#_Se convierte la imagen de Image_msg a Image_cv2
bridge = CvBridge()
# Imagen cv2
imagen0 = bridge.imgmsg_to_cv2(data1, "bgr8")
#_Procesamiento de la imagen
# TIP
imagenT = tip(imagen0)
# Escala de grises
imagenG = cv2.cvtColor(imagenT, cv2.COLOR_BGR2GRAY)
# Binarizacion
_,imagenZ = cv2.threshold(imagenG,50,255,cv2.THRESH_BINARY)
#_Reconocimiento de la direccion
x0=np.linspace(130,70,60,endpoint=True,dtype=int)
y0=[299,284,269,254,239,224,209,194]
j=int(0)
K=True
#_Busqueda de puntos en la imagen TIP
# Busca x1
while K==True or j==7:
        y1=y0[j]
        for i in x0:
                if imagenZ[y1][i]>=10:
                        x1=i
                        break
        if x1==0:
                j=j+1
                print "NP___!!!"
        else:
                break
# Busca x2
xv=np.linspace(x1-1,x1+1,3,endpoint=True,dtype=int)
j = 1
while j <= 15:
        for i in xv:
                if imagenZ[y1-j][i]>=10:
                        x2 = i
                        y2 = y1-j
                        break
        j = j+1
        xv=np.linspace(x2-1,x2+1,3,endpoint=True,dtype=int)
# Filtro pasa bajos
if abs(x1-x1_1)>14:
        x1=x1_1
        x1_1 = x1
```

```
                print "x1___!!!"
        else:
                x1_1 = x1
        if abs(x2-x2_1)>14:
                x2=x2_1
                print "x2___!!!"
        else:
                x2_1 = x2

        #_Datos para visualizacion
        vis_p = Int16MultiArray()
        vis_p.data = [x1, y1, x2, y2,0,0,0,0]
        Vispub.publish(vis_p)
        #_Calculo de th
        th = np.arctan2(x1-x2,y1-y2)#En radianes
        controladorSO()
#****************************************
#****************************************
#                TIP
def tip(imagenT):
        M=np.array([[-7.89695268e-02,-4.18895405e-01,1.28024400e+02],
        [1.00393984e-02,-1.45821320,4.00364200e+02],
        [2.60135943e-05,-4.15395022e-03,1]])
        #Obtenemos la imagen con correccion de pespectiva
        imagenH = cv2.warpPerspective(imagenT, M, (200,300),
        borderMode=cv2.BORDER_CONSTANT, borderValue=(25, 25, 25))
        return imagenH
#****************************************
#****************************************
#       CONTROL DEL STEERING Y LA VELOCIDAD SIN OBSTACULO
def controladorSO():
        global th_1
        #_Ley de Control
        d = 90
        kx = 3.0 #3.0
        ex = x1-x_ref
        kth = 0.25        #0.25
        u = d+kx*ex+kth*th

        #_Saturacion del Steering
        if u < 42:
                u = 42
        if u > 140:
                u = 140

        print "theta_",th
        print "error_x_",ex
        print "Steering_",u
        print "**********************"
        #________Datos para graficar
        q = Float32MultiArray()
        q.data = [ex, th*(180/np.pi), u, 0, 0, x1]
```

```
        Datapub.publish(q)
        Vpub.publish(200) #250-330
        Spub.publish(u)
#*****************************************
#*****************************************
#                       PRINCIPAL
if __name__ == '__main__':
        print "Nodo inicializado: SIM_model_01.py"
        # Inicializa el nodo
        rospy.init_node('SIM_model_01',anonymous=True)
        Vpub = rospy.Publisher('/manual_control/speed',Int16,
        queue_size=5)
        Spub = rospy.Publisher('/steering',Int16,queue_size=5)
        Vispub = rospy.Publisher('/vis_points',Int16MultiArray,
        queue_size=5)
        Datapub = rospy.Publisher('/data_send',Float32MultiArray,
        queue_size=5)
        rospy.Subscriber("/camera/rgb/image_raw",Image,callback_image)
        rospy.spin()
```

Con el siguiente código se implementó en el simulador el controlador basado en campos potenciales 4.11 como un nodo de ROS. Un código muy similar se usó con el vehículo real.

```
#! /usr/bin/env python
import rospy
import cv2
import numpy as np
from sensor_msgs.msg import Image
from cv_bridge import CvBridge, CvBridgeError
from std_msgs.msg import Int16, Int16MultiArray, Float32,
Float32MultiArray

# V. globales de la vision
x_ref = 120 #120
x1 = 0
x2 = 0
y1 = 0
y2 = 0
th = 0.0
thPF = 0.0
x1_1 = 120
x2_1 = 120
th_1 = 0
#*****************************************
#*****************************************
# ANALISIS Y PROCESAMIENTO DE IMAGENES
def callback_image(data1):
```

```
        global x1
        global x2
        global y1
        global y2
        global th
        global thPF
        global x1_1
        global x2_1
        global th_1
        #-Se convierte la imagen de Image_msg a Image_cv2
        bridge = CvBridge()
        # Imagen cv2
        imagen0 = bridge.imgmsg_to_cv2(data1, "bgr8")
        #-Procesamiento de la imagen
        # TIP
        imagenT = tip(imagen0)
        # Escala de grises
        imagenG = cv2.cvtColor(imagenT, cv2.COLOR_BGR2GRAY)
        # Binarizacion
        _,imagenZ = cv2.threshold(imagenG,50,255,cv2.THRESH_BINARY)
        #-Reconocimiento de la direccion
        x0=np.linspace(130,70,60,endpoint=True,dtype=int)
        y0=[299,284,269,254,239,224,209,194]
        j=int(0)
        K=True
        #-Busqueda de puntos en la imagen TIP
        # Busca x1
        while K==True or j==7:
                y1=y0[j]
                for i in x0:
                        if imagenZ[y1][i]>=10:
                                x1=i
                                break
                if x1==0:
                        j=j+1
                        print "NP___!!!"
                else:
                        break
        # Busca x2
        xv=np.linspace(x1-1,x1+1,3,endpoint=True,dtype=int)
        j = 1
        while j <= 15:
                for i in xv:
                        if imagenZ[y1-j][i]>=10:
                                x2 = i
                                y2 = y1-j
                                break
                j = j+1
                xv=np.linspace(x2-1,x2+1,3,
                endpoint=True,dtype=int)

        if abs(x1-x1_1)>14:
```

```
                x1=x1_1
                x1_1 = x1
                print "x1___!!!"
        else:
                x1_1 = x1

        if abs(x2-x2_1)>14:
                x2=x2_1
                print "x2___!!!"
        else:
                x2_1 = x2

        #_Datos para visualizacion
        vis_p = Int16MultiArray()
        vis_p.data = [x1, y1, x2, y2,0,0,0,0]
        Vispub.publish(vis_p)
        #_Calculo de th y thPF
        Kpp = 0.5 #0.5 200-330rpm
        Kmm = 2.0       #2.0 200-330rpm
        Lh = 50.0
        l = 15.0
        th = np.arctan2(x1-x2,y1-y2)#En radianes
        num_thPF = x2*(Kmm/(Kmm-2*Kpp))-x_ref*np.cos(th)*
        ((Kmm+2*Kpp)/(Kmm-2*Kpp))
        den_thPF = Lh+l-x_ref*abs(np.sin(th))
        thPF = np.arctan2(num_thPF,den_thPF) #En radianes
        controladorSO()
#****************************************
#****************************************
#                                  TIP
def tip(imagenT):
        M=np.array([[-7.89695268e-02,-4.18895405e-01,1.28024400e+02],
        [1.00393984e-02,-1.45821320,4.00364200e+02],
        [2.60135943e-05,-4.15395022e-03,1]])
        #Obtenemos la imagen con correccion de pespectiva
        imagenH = cv2.warpPerspective(imagenT, M, (200,300),
        borderMode=cv2.BORDER_CONSTANT, borderValue=(25, 25, 25))
        return imagenH
#****************************************
#****************************************
#       CONTROL DEL STEERING Y LA VELOCIDAD SIN OBSTACULO
def controladorSO():
        global th_1
        #_Ley de Control
        d = 90
        kx = 2.0 #2.0 200-330rpm
        ex = x1-x_ref
        kth = 0.1       #0.1 200-330rpm
        u = d+kx*ex+kth*thPF

        #_Saturacion del Steering
        if u < 42:
```

```
                u = 42
        if u > 140:
                u = 140

        print "theta ",thPF
        print "error_x ",ex
        print "Steering ",u
        print "**********************"
        #--------Datos para graficar
        q = Float32MultiArray()
        q.data = [ex, thPF*(180/np.pi), u, 0, 0, x1]
        Datapub.publish(q)
        Vpub.publish(330) #250-330
        Spub.publish(u)
#****************************************
#****************************************
#                        PRINCIPAL
if __name__ == '__main__':
        print "Nodo inicializado: SIM_model_01.py"
        # Inicializa el nodo
        rospy.init_node('SIM_model_01',anonymous=True)
        # Topico del speed (Publicador)
        Vpub = rospy.Publisher('/manual_control/speed',Int16,
        queue_size=5)
        # Topico del steering (Publicador)
        Spub = rospy.Publisher('/steering',Int16,queue_size=5)
        Vispub = rospy.Publisher('/vis_points',Int16MultiArray,
        queue_size=5)
        Datapub = rospy.Publisher('/data_send',Float32MultiArray,
        queue_size=5)
        # Topico del analisis de la imagen (Suscriptor)
        rospy.Subscriber("/camera/rgb/image_raw",Image,callback_image)
        rospy.spin()
```

Del mismo modo, con el siguiente código se implementaron los controladores óptimo 4.12 y "*preview controller*" 4.13. Un código muy similar se usó para conducir el vehículo real.

```
#! /usr/bin/env python
import rospy
import cv2
import numpy as np
from sensor_msgs.msg import Image
from cv_bridge import CvBridge, CvBridgeError
from std_msgs.msg import Int16, Int16MultiArray, Float32,
Float32MultiArray

# V. globales de la vision
```

```
FT = True
x_ref = 120 #120
x1 = 0
x2 = 0
y1 = 0
y2 = 0
th = 0.0
th1 = 0.0
th2 = 0.0
x1_h = 120
x2_h = 120
x2_h1 = 120
x2_h2 = 120
x2_v = np.zeros((1,59),dtype=int)
y2_v = np.zeros((1,59),dtype=int)
u = 90

#*****************************************
#*****************************************
#                    TIP
def tip(imagenT):
        M=np.array([[-7.89695268e-02,-4.18895405e-01,1.28024400e+02],
        [1.00393984e-02,-1.45821320,4.00364200e+02],
        [2.60135943e-05,-4.15395022e-03,1]])
        #Obtenemos la imagen con correccion de pespectiva
        imagenH = cv2.warpPerspective(imagenT, M, (200,300),
        borderMode=cv2.BORDER_CONSTANT, borderValue=(25, 25, 25))
        return imagenH
#*****************************************
#*****************************************
#       ANALISIS Y PROCESAMIENTO DE IMAGENES
def callback_image(data1):
        global x1
        global x2
        global y1
        global y2
        global th
        global th1
        global th2
        global x1_h
        global x2_h
        global x2_h1
        global x2_h2
        global x2_v
        global y2_v
        global FT
        #-Se convierte la imagen de Image_msg a Image_cv2
        bridge = CvBridge()
        # Imagen cv2
        imagen0 = bridge.imgmsg_to_cv2(data1, "bgr8")
        #-Procesamiento de la imagen
        # TIP
```

```
imagenT = tip(imagen0)
# Cambia la matriz de R3 a R1
imagenG = cv2.cvtColor(imagenT, cv2.COLOR_BGR2GRAY)
# Binarizacion
_,imagenZ = cv2.threshold(imagenG,50,255,cv2.THRESH_BINARY)
#_Reconocimiento de la direccion
x0=np.linspace(199,0,200,endpoint=True,dtype=int)
y0=[299,269,239,209,179,149,119,89]
j=int(0)
K=True
#_Busqueda de puntos en la imagen TIP
# Busca x1
while K==True or j==7:
        y1=y0[j]
        for i in x0:
                if imagenZ[y1][i]>=127:
                        x1=i
                        K = False
                        break
        if x1==0:
                j=j+1
                print "x1 not found in y=",y1
        else:
                break
# Busca x2
xv=np.linspace(x1+2,x1-2,5,endpoint=True,dtype=int)
j = 1

while j <= 59:
        for i in xv:
                if imagenZ[y1-j][i]>=127:
                        x2_v[0][j-1] = i
                        y2_v[0][j-1] = y1-j
                        break
        xv=np.linspace(x2_v[0][j-1]+2,x2_v[0][j-1]-2,5,
        endpoint=True,dtype=int)
        j = j+1

l = 15
x2 = x2_v[0][l-1]
y2 = y2_v[0][l-1]

# Criterio de robustez (Filtro pasas bajos)
if FT == True:
        FT = False
        x1_h = x1
        x2_h = x2
        x2_h1 = x2_v[0][2*l-1]
        x2_h2 = x2_v[0][3*l-1]
else:
        G = 10 #10 cm
        if abs(x1-x1_h)>G:
```

```
                        x1 = x1_h
                        print "x1___!!!"
                else:
                        x1_h = x1
                if abs(x2-x2_h)>G:
                        x2 = x2_h
                        print "x2___!!!"
                else:
                        x2_h = x2
                if abs(x2_v[0][2*l-1]-x2_h1)>G+5:
                        x2_v[0][2*l-1] = x2_h1
                        print "x2_1___!!!"
                else:
                        x2_h1 = x2_v[0][2*l-1]
                if abs(x2_v[0][3*l-1]-x2_h2)>G+7:
                        x2_v[0][3*l-1] = x2_h2
                        print "x2_2___!!!"
                else:
                        x2_h2 = x2_v[0][3*l-1]

        # Datos para visualizacion
        vis_p = Int16MultiArray()
        vis_p.data = [x1, y1, x2, y2,x2_v[0][2*l-1],y2_v[0][2*l-1],
        x2_v[0][3*l-1],y2_v[0][3*l-1]]
        Vispub.publish(vis_p)

        #__Calculo de th
        th = np.arctan2(x2_v[0][l-6]-x2_v[0][l+4],
        y2_v[0][l-6]-y2_v[0][l+4]) #En radianes
        th1 = np.arctan2(x2_v[0][2*l-6]-x2_v[0][2*l+4],
        y2_v[0][2*l-6]-y2_v[0][2*l+4]) #En radianes
        th2 = np.arctan2(x2_v[0][3*l-6]-x2_v[0][3*l+4],
        y2_v[0][3*l-6]-y2_v[0][3*l+4]) #En radianes
        controladorSO()
#****************************************
#****************************************
#       CONTROL DEL STEERING Y LA VELOCIDAD SIN OBSTACULO
def controladorSO():
        global u
        #_Ley de Control
        d = 90
        kx = 0.04749206 #0.04625885
        ex = x1-x_ref
        kth = 0.14213116 #0.14036202
        #_Referencias futuras
        kx1 = 0.0 #0.00054085
        kx2 = 0.0 #0.00061402
        kth1 = 0.0 #0.00096788
        kth2 = 0.0 #0.00089512
        F = kth1*th1+kth2*th2+(x_ref-100)*(kx1+kx2)
        u = d+np.arctan(kx*ex+kth*th+F)*(180/np.pi) #En grados
```

```
        #_Saturacion del Steering
        if u < 42:
                u = 42
        if u > 140:
                u = 140
        print "F_",F
        print "theta_",th*(180/np.pi) #En grados
        print "error_x_",ex
        print "Steering_",u
        print "**********************"

        # Datos para graficar
        q = Float32MultiArray()
        q.data = [ex, th*(180/np.pi), u, th1, th2, x1]
        Datapub.publish(q)
        Vpub.publish(330) #250-330
        Spub.publish(u)

#****************************************
#****************************************
#                                PRINCIPAL
if __name__ == '__main__':
        print "Nodo_inicializado:_SIM_model_01.py"
        rospy.init_node('SIM_model_01',anonymous=True)
        Vpub = rospy.Publisher('/manual_control/speed',Int16,
        queue_size=5)
        Spub = rospy.Publisher('/steering',Int16,queue_size=5)
        Vispub = rospy.Publisher('/vis_points',Int16MultiArray,
        queue_size=5)
        Datapub = rospy.Publisher('/data_send',Float32MultiArray,
        queue_size=5)
        rospy.Subscriber("/camera/rgb/image_raw",Image,callback_image)
        rospy.spin()
```

El siguiente código es el del nodo de visualización usado para mostrar en pantalla la imagen sin perspectiva y las consignas de control del auto; mientras se condice en el simulador.

```
#! /usr/bin/env python
import rospy
import cv2
import numpy as np
from std_msgs.msg import Int16, Int16MultiArray
from sensor_msgs.msg import Image
from cv_bridge import CvBridge, CvBridgeError

u = 0 # Steering
V = 0 # Speed
```

```
# V. globales de la vision
x_ref = 120
x1 = 0
x2 = 0
y1 = 0
y2 = 0
x2_1 = 0
y2_1 = 0
x2_2 = 0
y2_2 = 0
#************************************
#************************************
#           TIP
def tip(imagenT):
        M=np.array([[-7.89695268e-02,-4.18895405e-01,1.28024400e+02],
        [1.00393984e-02,-1.45821320,4.00364200e+02],
        [2.60135943e-05,-4.15395022e-03,1]]) # TMR SIM Oscar,
        warpP(200,300)
        #Obtenemos la imagen con correccion de pespectiva
        imagenH = cv2.warpPerspective(imagenT, M, (200,300),
        borderMode=cv2.BORDER_CONSTANT, borderValue=(25, 25, 25))
        return imagenH
#************************************
#************************************
#                IMAGE_DRAW
def ImDraw(imagenD):
        # Se invierten los colores para visualizar mejor
        imagenD = cv2.bitwise_not(imagenD)
        imagenR =cv2.circle(imagenD, (x_ref,y1), 2, (0,0,0), 2)
        imagenR =cv2.circle(imagenR, (x1,y1), 2, (0,0,255), 2)
        imagenR = cv2.circle(imagenR, (x2,y2), 2, (0,0,255), 2)
        imagenR = cv2.drawMarker(imagenR, (100,296),(255,0,0),
        cv2.MARKER_TRIANGLE_UP,5,cv2.LINE_4)
        imagenR = cv2.line(imagenR,(x2_1-3,y2_1),(x2_1+3,y2_1),
        (110,200,0),1)
        imagenR = cv2.line(imagenR,(x2_2-3,y2_2),(x2_2+3,y2_2),
        (110,200,0),1)
        imagenR = cv2.line(imagenR,(x2_1,y2_1-3),(x2_1,y2_1+3),
        (110,200,0),1)
        imagenR = cv2.line(imagenR,(x2_2,y2_2-3),(x2_2,y2_2+3),
        (110,200,0),1)
        font = cv2.FONT_HERSHEY_SIMPLEX
        imagenR = cv2.putText(imagenR, 'Steering '+str(int(u)),
        (5, 15), font, 0.4, (0,255,0), 1, cv2.LINE_AA)
        imagenR = cv2.putText(imagenR, 'Speed '+str(int(V)), (5, 30),
        font, 0.4, (0,255,0), 1, cv2.LINE_AA)
        imagenR = cv2.putText(imagenR, 'error x '+str(x1-x_ref), (5, 45),
        font, 0.4, (0,255,0), 1, cv2.LINE_AA)
        return imagenR
        # El coche tiene un ancho de 20 cm
        # La posicion de x_ref=120
        # El auto esta en la coordenada (100,300) del suelo (imagenR).
```

```
        # El camino tiene un ancho de 80 cm. Cada carril de 40 cm.
#************************************
#************************************
def callback_S(data0):
        global u
        u = data0.data
#************************************
#************************************
def callback_V(data1):
        global V
        V = data1.data
#************************************
#************************************
def callback_Visp(data2):
        global x1
        global x2
        global y1
        global y2
        global x2_1
        global y2_1
        global x2_2
        global y2_2
        x1 = data2.data[0]
        y1 = data2.data[1]
        x2 = data2.data[2]
        y2 = data2.data[3]
        x2_1 = data2.data[4]
        y2_1 = data2.data[5]
        x2_2 = data2.data[6]
        y2_2 = data2.data[7]
#************************************
#************************************
#       ANALISIS,PROCESAMIENTO DE IMAGENES Y CONTROL DE LA DIRECCION
def callback_image(data3):
        # Se convierte la imagen de Image_msg a Image_cv2
        bridge = CvBridge()
        # Imagen cv2
        imagen0 = bridge.imgmsg_to_cv2(data3, "bgr8")
        # Procesamiento de la imagen
        # TIP
        imagenT = tip(imagen0)
        # Cambia la matriz de R3 a R1 (necesario).
        imagenG = cv2.cvtColor(imagenT, cv2.COLOR_BGR2GRAY)
        # Binarizacion
        _,imagenZ = cv2.threshold(imagenG,127,255,cv2.THRESH_BINARY)
        imagenF = ImDraw(imagenT)
        # Transforma imagen de cv2 a image_msgs
        imagenF = bridge.cv2_to_imgmsg(imagenF,"bgr8")
        Impub.publish(imagenF)
#************************************
#************************************
#                       PRINCIPAL
```

```
if __name__ == '__main__':
        print "Nodo inicializado: VisCam"
        rospy.init_node('VisCam',anonymous=True)
        rospy.Subscriber("steering",Int16,callback_S)
        rospy.Subscriber("/manual_control/speed",Int16,callback_V)
        rospy.Subscriber("vis_points",Int16MultiArray,callback_Visp)
        rospy.Subscriber("/camera/rgb/image_raw",Image,callback_image)
        Impub = rospy.Publisher('imagen_TIP',Image,queue_size=10)
        rospy.spin()
```

El siguiente código corresponde al nodo de adquisición de datos del simulador.

```
#!/usr/bin/env python
import rospy
from std_msgs.msg import Float32, Float32MultiArray
#************************************
#************************************
def callback_Visp(data0):
        t = rospy.get_time()
        ex = data0.data[0]
        th = data0.data[1]
        u = data0.data[2]
        th1 = data0.data[3]
        th2 = data0.data[4]
        x1 = data0.data[5]
        f = open('data_comrob.txt','a+')
        f.write("%5.2f  %i  %5.2f  %i  %5.2f  %5.2f  %i \n" %
        (t, ex, th, u, th1, th2, x1))
        f.close()
#************************************
#************************************
if __name__ == '__main__':
        print "************************************"
        print "  ....Guardando datos..."
        print "************************************"
        rospy.init_node('data_write', anonymous=True)
        rospy.Subscriber("data_send", Float32MultiArray , callback_Visp)
        rospy.spin()
```

El siguiente código se implementó en el Arduino NANO para publicar la información del sensor inercial en un mensaje tipo IMU.

```
/*****MEDICIONES DE LA IMU*****
 *  Conexiones del MPU6050 (Arduno UNO-Nano):
 *  SCL-----A5(Arduno UNO-Nano)//SCL(Arduino MEGA)
 *  SDA-----A4(Arduno UNO-Nano)//SDA(Arduino MEGA)
 *  VCC-----3.3V
```

```
 *  GND———GND
 */
//*****************************************************
//*****************************************************
#include<Wire.h>
#include <ros.h>                          // Libreria de ROS
#include <geometry_msgs/Pose.h>           // Mensaje tipo float
#if defined(ARDUINO) && ARDUINO >= 100
  #include <Arduino.h>
#else
  #include <WProgram.h>
#endif
#include "I2Cdev.h"


//Parametros utilizados:
const int MPU_addr=0x68;  // I2C address of the MPU-6050
float AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ,V;
/*
//  Variables de integracion
//------------------Filtro
float AcFX = 0.0;
float AcFX0 = 0.0;
float AcFY = 0.0;
float AcFY0 = 0.0;
float AcFZ = 0.0;
float AcFZ0 = 0.0;
*/
//-------------------Integrador
float VelX = 0.0;
float VelY = 0.0;
const float h = 0.01;//10ms sample
//*****************************************************
//*****************************************************
// Instancias de ROS
ros::NodeHandle Arduino_IMU;
geometry_msgs::Pose pose;
ros::Publisher pub("imu",&pose);
//*****************************************************
//*****************************************************
void setup(){
  Wire.begin();
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x6B);  // PWR_MGMT_1 register
  Wire.write(0);     // set to zero (wakes up the MPU-6050)
  Wire.endTransmission(true);
  //-----------------------------------OffSets
  // MPU6050_RA_XA_OFFS_H  0x06
  I2Cdev::writeWord(MPU_addr, 0x06, -2141);
  // MPU6050_RA_YA_OFFS_H   0x08
  I2Cdev::writeWord(MPU_addr, 0x08, 1234);
  // MPU6050_RA_ZA_OFFS_H   0x0A
```

```
  I2Cdev::writeWord(MPU_addr, 0x0A, 1561);
  I2Cdev::writeBits(MPU_addr, 0x00, 6, 6, -3);
  I2Cdev::writeBits(MPU_addr, 0x01, 6, 6, -111);
  I2Cdev::writeBits(MPU_addr, 0x02, 6, 6, 115);
  //-------------------------------------Configuracion de ROS
  Arduino_IMU.initNode();
  Arduino_IMU.advertise(pub);
  delay(1000);
}
//*****************************************************
//*****************************************************
void loop(){
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x3B);  // starting with register 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false);
  // request a total of 14 registers
  Wire.requestFrom(MPU_addr,14,true);
  // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
  AcX=Wire.read()<<8|Wire.read();
  AcX=AcX*(9.81/16384.0);  // (m/s**2)
  // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
  AcY=Wire.read()<<8|Wire.read();
  AcY=AcY*(9.81/16384.0);  // (m/s**2)
  // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
  AcZ=Wire.read()<<8|Wire.read();
  AcZ=AcZ*(9.81/16384.0);  // (m/s**2)
  // 0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)
  Tmp=Wire.read()<<8|Wire.read();
  // 0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
  GyX=Wire.read()<<8|Wire.read();
  GyX=GyX*(250.0/32768.0); // (grad/s)
  // 0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
  GyY=Wire.read()<<8|Wire.read();
  GyY=GyY*(250.0/32768.0); // (grad/s)
  // 0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)
  GyZ=Wire.read()<<8|Wire.read();
  GyZ=GyZ*(250.0/32768.0); // (grad/s)
  /*
  //-------------------------------------Filtro pasa bajos
  float pi = 3.1415926535;
  float f = 0.75;//0.25;  //Frecuencia de corte en (Hz)
  float h_ros = h+0.0033; //Periodo medido en ros (s)
  float k = 1.0;          //Ganancia del filtro
  AcFX = (1/(1+2*pi*f*h_ros))*(AcFX0+k*(2*pi*f*h_ros)*(AcX+0.07));
  AcFY = (1/(1+2*pi*f*h_ros))*(AcFY0+k*(2*pi*f*h_ros)*(AcY-0.10));
  AcFZ = (1/(1+2*pi*f*h_ros))*(AcFZ0+k*(2*pi*f*h_ros)*(AcZ-0.01));
  */
  //-------------------------------------Integrador
  float h_ros = h+0.0033; //Periodo medido en ros (s)
  VelX = h_ros*(AcX);
  VelY = h_ros*(AcY);
  V = sqrt(pow(VelX,2)+pow(VelY,2));
```

```
  pose.position.x = AcX;          //Publica la aceleracion en X en m/s2
  pose.position.y = AcY;          //Publica la aceleracion en Y en m/s2
  pose.position.z = AcZ;          //Publica la aceleracion en Z en m/s2
  pose.orientation.x = GyX;       //Publica la direccion en X
  pose.orientation.y = GyY;       //Publica la direccion en Y
  pose.orientation.z = GyZ;       //Publica la direccion en Z
  pose.orientation.w = V;         //Publica velocidad en W
  pub.publish(&pose);
  Arduino_IMU.spinOnce();         //Funcion que crea un bucle
  delay(h*1000);
  //Actualizacion de CI
  /*
  AcFX0 = AcFX;
  AcFY0 = AcFY;
  AcFZ0 = AcFZ;
  */
  clearFIFO();
}
//*****************************************************
//*****************************************************
void clearFIFO(){
  Wire.begin();
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x74);  // FIFO_R_W register
  Wire.write(0x00);      // Clear FIFO
  Wire.endTransmission(true);
}
```

Finalmente, para ajustar las ganancias del controlador óptimo 4.12 y del "*preview controler*" 4.13 se utilizó el siguiente programa el cual resuelve la ecuación de Ricatti y calcula el vector de ganancias; tanto para el término "*state feedback*" como para el término "*preview compensations*".

```
import sys
import numpy as np
import scipy.linalg

# Frec. de muestreo de la senal de control (Hz)
f = 29.7#16.6/SEG #29.7/BIN
# Periodo de muestreo de la senal de control (s)
h = 1/f
# Velocidad de avance del auto (rpm)
#vrpm = 200
# Velocidad de avance del auto (m/s)
v = 0.624 #vrpm*(0.588/200)
# Separacion de las llantas (m)
```

```
L = 0.26
# Distancia entre las llantas traseras y la base de la homografia (m)
Lh = 0.5
# Matrices del modelo cinematico usado
A = np.matrix([[1, v*h],[0, 1]])
B = (v*h/L)*np.matrix([[Lh+v*h/2],[1]])
C = np.matrix([[1, 0],[0, 1]])
# Matrices de ponderacion propuestas
Q = 0.015*np.matrix([[1, 0],[0, 1]])
R = 7.5
print "-------------------------------------------------------"
print "v " ,v
print "R " ,R
print "Q " ,Q
#print "-------------------------------------------------------"
P = np.matrix(scipy.linalg.solve_discrete_are(A, B, Q, R))
#print "P ",P
print "-------------------------------------------------------"
K = np.matrix(scipy.linalg.inv(B.T*P*B+R)*(B.T*P*A))
print "K " ,K
print "-------------------------------------------------------"
for i in range(1,5):
        fi = np.matrix(scipy.linalg.inv(B.T*P*B+R)*B.T
        *(((A-B*K).T)**(i-1))*C.T*Q)
        print i
        print fi
```

Bibliografía

[1] Peter Davidson and Anabelle Spinoulas. Autonomous vehicles: what could this mean for the future of transport. In *Australian Institute of Traffic Planning and Management (AITPM) National Conference, Brisbane, Queensland*, 2015.

[2] Todd Litman. Autonomous vehicle implementation predictions: Implications for transport planning. Technical report, 2015.

[3] SAE On-Road Automated Vehicle Standards Committee et al. Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems. *SAE International*, 2014.

[4] U.S. Department of Transportation Releases Policy on Automated Vehicle Development. Automated vehicles for safety. `https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety#issue-road-self-driving`, 2019. [Online; accessed 19-May-2019].

[5] Federación Mexicana de Robótica A.C. Federación mexicana de robótica. `https://femexrobotica.org/`, 2019. [Online; accessed 03-September-2019].

[6] Marc Weber. Where to? a history of autonomous vehicles. `https://www.computerhistory.org/atchm/where-to-a-history-of-autonomous-vehicles/`, 2014. [Online; accessed 19-May-2019].

[7] Ernst D Dickmanns. The development of machine vision for road vehicles in the last decade. In *Intelligent Vehicle Symposium, 2002. IEEE*, volume 1, pages 268–281. IEEE, 2002.

[8] EUREKA. Programme for a european traffic system with highest efficiency and unprecedented safety. `https://www.eurekanetwork.org/project/id/45`, 1995. [Online; accessed 19-May-2019].

[9] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, et al. Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, 23(9):661–692, 2006.

[10] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, MN Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

[11] Autonomos Labs. The road to the darpa urban challenge 2007. `https://autonomos-labs.com/projects/past-projects/darpa-urban-challenge/`, 2015. [Online; accessed 03-September-2019].

[12] sinembargo. Vehículo robot recorre las carreteras de méxico sin conductor. `https://www.sinembargo.mx/01-11-2015/1537299`, 2015. [Online; accessed 03-September-2019].

[13] University of Nevada. Autonomous car finishes record-setting trip in mexico. `https://www.unr.edu/nevada-today/news/2015/raul-rojas-autonomous-drive-in-mexico`, 2015. [Online; accessed 03-September-2019].

[14] Juergen Ackermann, Jürgen Guldner, Wolfgang Sienel, Reinhold Steinhauser, and Vadim I Utkin. Linear and nonlinear controller design for robust automatic steering. *IEEE Transactions on Control Systems Technology*, 3(1):132–143, 1995.

[15] Jarrod M Snider et al. Automatic steering methods for autonomous automobile path tracking. *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RITR-09-08*, 2009.

[16] Peter Corke. *Robotics, vision and control: fundamental algorithms In MATLAB® second, completely revised*, volume 118. Springer, 2017.

[17] Omead Amidi and Chuck E Thorpe. Integrated mobile robot control. In *Mobile Robots V*, volume 1388, pages 504–524. International Society for Optics and Photonics, 1991.

[18] Ruben Darío Hernandez Beleño, Giovani Bernardes Vítor, Janito Vaqueiro Ferreira, and Pablo Siqueira Meirelles. Planeación y seguimiento de trayectorias de un vehículo terrestre con base en el control de dirección en un ambiente real. *Scientia et technica*, 19(4):407–412, 2014.

[19] Rajesh Rajamani. *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[20] Miguel Angel Sotelo. Lateral control strategy for autonomous steering of ackerman-like vehicles. *Robotics and Autonomous Systems*, 45(3-4):223–233, 2003.

[21] Juergen Ackermann and Wolfgang Sienel. Robust control for automatic steering. In *1990 American Control Conference*, pages 795–800. IEEE, 1990.

[22] Charles C MacAdam. Application of an optimal preview control for simulation of closed-loop automobile driving. *IEEE Transactions on systems, man, and cybernetics*, 11(6):393–399, 1981.

[23] AY Lee. A preview steering autopilot control algorithm for four-wheel-steering passenger vehicles. *Journal of dynamic systems, measurement, and control*, 114(3):401–408, 1992.

[24] Huei Peng and Masayoshi Tomizuka. Preview control for vehicle lateral guidance in highway automation. In *1991 American Control Conference*, pages 3090–3095. IEEE, 1991.

[25] RS Sharp and V Valtetsiotis. Optimal preview car steering control. *Vehicle System Dynamics*, 35(SUPPL. 1):101–117, 2001.

[26] DJ Cole, AJ Pick, and AMC Odhams. Predictive and linear quadratic methods for potential application to modelling driver steering control. *Vehicle System Dynamics*, 44(3):259–284, 2006.

[27] RS Sharp. Application of optimal preview control to speed-tracking of road vehicles. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 221(12):1571–1578, 2007.

[28] Juyong Kang, Rami Y Hindiyeh, Seung-Wuk Moon, J Christian Gerdes, and Kyongsu Yi. Design and testing of a controller for autonomous vehicle path tracking using gps/ins sensors. *IFAC Proceedings Volumes*, 41(2):2093–2098, 2008.

[29] M Thommyppillai, S Evangelou, and RS Sharp. Car driving at the limit by adaptive linear optimal preview control. *Vehicle system dynamics*, 47(12):1535–1550, 2009.

[30] Dongwook Kim, Juyoung Kang, and Kyoungsu Yi. Control strategy for high-speed autonomous driving in structured road. In *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 186–191. IEEE, 2011.

[31] Shuuji Kajita, Hirohisa Hirukawa, Kensuke Harada, and Kazuhito Yokoi. *Introduction to humanoid robotics*, volume 101. Springer, 2014.

[32] Ali Boyali, Vijay John, Zheming Lyu, Rathour Swarn, and Seichi Mita. Self-scheduling robust preview controllers for path tracking and autonomous vehicles. In *2017 11th Asian Control Conference (ASCC)*, pages 1829–1834. IEEE, 2017.

[33] Tamás Keviczky, Paolo Falcone, Francesco Borrelli, Jahan Asgari, and Davor Hrovat. Predictive control approach to autonomous vehicle steering. In *2006 American control conference*, pages 6–pp. IEEE, 2006.

[34] Paolo Falcone, Francesco Borrelli, Jahan Asgari, H Eric Tseng, and Davor Hrovat. A model predictive control approach for combined braking and steering in autonomous vehicles. In *2007 Mediterranean Conference on Control & Automation*, pages 1–6. IEEE, 2007.

[35] Paolo Falcone, Francesco Borrelli, Jahan Asgari, Hongtei Eric Tseng, and Davor Hrovat. Predictive active steering control for autonomous vehicle systems. *IEEE Transactions on control systems technology*, 15(3):566–580, 2007.

[36] Guilherme V Raffo, Guilherme K Gomes, Julio E Normey-Rico, Christian R Kelber, and Leandro B Becker. A predictive controller for autonomous vehicle path tracking. *IEEE transactions on intelligent transportation systems*, 10(1):92–102, 2009.

[37] Ashwin Carvalho, Yiqi Gao, Andrew Gray, H Eric Tseng, and Francesco Borrelli. Predictive control of an autonomous ground vehicle using an iterative linearization approach. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pages 2335–2340. IEEE, 2013.

[38] Ashwin Carvalho, Yiqi Gao, Stéphanie Lefevre, and Francesco Borrelli. Stochastic predictive control of autonomous vehicles in uncertain environments. In *12th International Symposium on Advanced Vehicle Control*, pages 712–719, 2014.

[39] Jie Ji, Amir Khajepour, Wael William Melek, and Yanjun Huang. Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints. *IEEE Transactions on Vehicular Technology*, 66(2):952–964, 2016.

[40] Ziya Ercan, Metin Gokasan, and Francesco Borrelli. An adaptive and predictive controller design for lateral control of an autonomous vehicle. In *2017 IEEE international conference on vehicular electronics and safety (ICVES)*, pages 13–18. IEEE, 2017.

[41] Ugo Rosolia, Stijn De Bruyne, and Andrew G Alleyne. Autonomous vehicle control: A nonconvex approach for obstacle avoidance. *IEEE Transactions on Control Systems Technology*, 25(2):469–484, 2016.

[42] Diego Fernando Sendoya. ¿ qué es el control predictivo y hacia dónde se proyecta? *Publicaciones e Investigación*, 7:53–59, 2015.

[43] Chuan Hu, Hui Jing, Rongrong Wang, Fengjun Yan, and Mohammed Chadli. Robust h∞ output-feedback control for path following of autonomous ground vehicles. *Mechanical Systems and Signal Processing*, 70:414–427, 2016.

[44] Zhengrong Chu, Yuming Sun, Christine Wu, and Nariman Sepehri. Active disturbance rejection control applied to automated steering for lane keeping in autonomous vehicles. *Control Engineering Practice*, 74:13–21, 2018.

[45] M Sugeno and K Murakami. Fuzzy parking control of model car. In *The 23rd IEEE Conference on Decision and Control*, pages 902–903. IEEE, 1984.

[46] Kazuo Tanaka and Manabu Sano. Trajectory stabilization of a model car via fuzzy control. *Fuzzy Sets and Systems*, 70(2-3):155–170, 1995.

[47] Th Fraichard and Ph Garnier. Fuzzy control to drive car-like vehicles. *Robotics and autonomous systems*, 34(1):1–22, 2001.

[48] KRS Kodagoda, W Sardha Wijesoma, and Eam Khwang Teoh. Fuzzy speed and steering control of an agv. *IEEE Transactions on control systems technology*, 10(1):112–120, 2002.

[49] Ahmed El Hajjaji and Said Bentalba. Fuzzy path tracking control for automatic steering of vehicles. *Robotics and Autonomous Systems*, 43(4):203–213, 2003.

[50] Joshué Pérez Rastelli, Teresa De Pedro, and Matilde Santos. Controladores borrosos para la dirección de vehículos autónomos en maniobras dentro de entornos urbanos. 2012.

[51] Xinyu Wang, Mengyin Fu, Hongbin Ma, and Yi Yang. Lateral control of autonomous vehicles based on fuzzy logic. *Control Engineering Practice*, 34:1–17, 2015.

[52] Uma Anand. Fuzzy logic vision and control of autonomous vehicles. *IPASJ Int. J. Comput. Sci*, 4(1):1–7, 2016.

[53] Hong Sun, Changzhu Zhang, Guangyong An, Qijun Chen, and Chengju Liu. Fuzzy-model-based h∞ dynamic output feedback control with feedforward for autonomous vehicle path tracking. In *2017 International Conference on Fuzzy Theory and Its Applications (iFUZZY)*, pages 1–6. IEEE, 2017.

[54] Anh-Tu Nguyen, Chouki Sentouh, and Jean-Christophe Popieul. Fuzzy steering control for autonomous vehicles under actuator saturation: Design and experiments. *Journal of the Franklin Institute*, 355(18):9374–9395, 2018.

[55] Changzhu Zhang, Jinfei Hu, Jianbin Qiu, Weilin Yang, Hong Sun, and Qijun Chen. A novel fuzzy observer-based steering control approach for path tracking in autonomous vehicles. *IEEE Transactions on Fuzzy Systems*, 27(2):278–290, 2018.

[56] Bo Xiong and QU Shiru. Intelligent vehicle's path tracking based on fuzzy control. *Journal of transportation systems engineering and information technology*, 10(2):70–75, 2010.

[57] Sheng-Zhi Du and Chun-Ling Tu. An autonomous vehicle navigation system based on hough transform and fuzzy logic. In *Electronics, Communications and Networks V*, pages 89–95. Springer, 2016.

[58] Hiroshi Kamada, Satoshi Naoi, and Toshiyuki Gotoh. A compact navigation system using image processing and fuzzy control. In *IEEE Proceedings on Southeastcon*, pages 337–342. IEEE, 1990.

[59] Juan Manuel Ramírez, Pilar Gómez-Gil, and Filiberto López Larios. A robot-vision system for autonomous vehicle navigation with fuzzy-logic control using lab-view. In *Electronics, Robotics and Automotive Mechanics Conference (CERMA 2007)*, pages 295–302. IEEE, 2007.

[60] Miguel A Olivares-Mendez, Ignacio Mellado, Pascual Campoy, Ivan Mondragon, and Carol Martinez. A visual agv-urban car using fuzzy control. In *The 5th International Conference on Automation, Robotics and Applications*, pages 145–150. IEEE, 2011.

[61] Jing Yang and Nanning Zheng. An expert fuzzy controller for vehicle lateral control. In *IECON 2007-33rd Annual Conference of the IEEE Industrial Electronics Society*, pages 880–885. IEEE, 2007.

[62] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[63] N Kehtarnavaz and W Sohn. Steering control of autonomous vehicles by neural networks. In *1991 American Control Conference*, pages 3096–3101. IEEE, 1991.

[64] RMH Cheng, JW Xiao, and S LeQuoc. Neuromorphic controller for agv steering. In *Proceedings 1992 IEEE International Conference on Robotics and Automation*, pages 2057–2062. IEEE, 1992.

[65] Dongbing Gu and Huosheng Hu. Neural predictive control for a car-like mobile robot. *Robotics and Autonomous Systems*, 39(2):73–86, 2002.

[66] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[67] Gowtham Garimella, Joseph Funke, Chuang Wang, and Marin Kobilarov. Neural network modeling for steering control of an autonomous vehicle. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2609–2615. IEEE, 2017.

[68] Mariusz Bojarski, Philip Yeres, Anna Choromanska, Krzysztof Choromanski, Bernhard Firner, Lawrence Jackel, and Urs Muller. Explaining how a deep neural network trained with end-to-end learning steers a car. *arXiv preprint arXiv:1704.07911*, 2017.

[69] Zhilu Chen and Xinming Huang. End-to-end learning for lane keeping of self-driving cars. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1856–1860. IEEE, 2017.

[70] Gaining Han, Weiping Fu, Wen Wang, and Zongsheng Wu. The lateral tracking control for the intelligent vehicle based on adaptive pid neural network. *Sensors*, 17(6):1244, 2017.

[71] Xuewu Ji, Xiangkun He, Chen Lv, Yahui Liu, and Jian Wu. Adaptive-neural-network-based robust lateral motion control for autonomous vehicle at driving limits. *Control Engineering Practice*, 76:41–53, 2018.

[72] Wikipedia. Convolutional neural network. `https://en.wikipedia.org/wiki/Convolutional_neural_network`, 2019. [Online; accessed 19-September-2019].

[73] Freie Universität Berlin. Autonomos model. `https://github.com/AutoModelCar/AutoModelCarWiki/wiki`, 2016. [Online; accessed 15-April-2019].

[74] Ward Brown. Brushless dc motor control made easy. *Microchip Technology Inc*, 1, 2002.

[75] Padmaraja Yedamale. Brushless dc (bldc) motor fundamentals. *Microchip Technology Inc*, 20:3–15, 2003.

[76] Wikipedia. Mapa de karnaugh. `https://es.wikipedia.org/wiki/Mapa_de_Karnaugh`, 2019. [Online; accessed 01-September-2019].

[77] Opencv dev team. Camera calibration and 3d reconstruction. `https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html?highlight=findhomography`, 2014. [Online; accessed 02-October-2019].

[78] Karl J Åström and Björn Wittenmark. *Computer-controlled systems: theory and design*. Courier Corporation, 2013.

[79] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous robot vehicles*, pages 396–404. Springer, 1986.

[80] Yoram Koren and Johann Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, pages 1398–1404. IEEE, 1991.

[81] Frank L. Lewis, Emanuel Stingu, and Y TTXYVVVRFFRX. Potential fields in cooperative motion control and formations. 2013.

[82] AJ Shaiju and Ian R Petersen. Formulas for discrete time lqr, lqg, leqg and minimax lqg optimal control problems. *IFAC Proceedings Volumes*, 41(2):8773–8778, 2008.

[83] Donald E Kirk. *Optimal control theory: an introduction*. Courier Corporation, 2012.

[84] Vladimir G Boltyanski and Alexander S Poznyak. *The robust maximum principle: theory and applications*. Springer Science & Business Media, 2011.