Instituto Tecnológico Autónomo de México
Depto. Académico de Sistemas Digitales
Robótica

Proyecto Final
Diseño e implementación de componentes para la conducción del robot AutoNOMOs

Edgar A. Granados Osegueda- 129565
Mariana R. Hernández Rocha- 150845
Jorge Isaac Chang Ortega - 149961
Humberto Isaac Téllez Benítez - 151887

***Resumen*— Presentamos el desarrollo e implementación en el simulador Gazebo de un robot autónomo. A partir del modelo de Ackerman, se obtiene el modelo cinemático y un control para llegar a una pose deseada. La implementación de todos los algoritmos se realizó en ROS. Utilizando un sensor Lidar, se implementa un filtro de Kalman para determinar la velocidad de un obstáculo y seguirlo. Se presentan los resultados de esta prueba realizada en Gazebo. Por último, se presenta la implementación y resultados de aplicar un filtro de histogramas a imágenes de un carro autónomo para determinar su posición probabilística respecto a carriles en una carretera.**

I. INTRODUCCIÓN

En los últimos años ha habido un creciente interes en los carros autónomos, especialmente despues del *Darpa Challenge* [6]. Los carros autónomos tienen un gran potencial de disminuir accidentes de tránsito así como disminuir tiempos de traslado y realizar ahorros en combustible, estimados desde $2000 USD a $4000 por vehículo por año [2]. Uno de las principales limitaciones que se tienen actualmente son los costosos sensores utilizados así como el requerimiento de cómputo de los algoritmos utilizados [3].

Para realizar la implementación de un carro autónomo es necesario conocer su modelo cinemático así como localizarlo en un medio. Uno de los principales retos de la robótica autónoma es la integración de diversos sensores y el procesamiento de los datos. Necesario para localizar tanto al robot en el medio como a otros objetos respecto al robot. Debido a que el sensado y procesamiento de datos conlleva errores [5], algunos de los algoritmos más populares buscan incluir la incertidumbre representando la información mediante distribuciones de probabilidad. El filtro de histogramas así como el filtro de Kalman son utilizados con éste fin.

El robot *AutoNOMOS model car* [4] se creó como una forma de promover la investigación en carros autónomos, buscando soluciones de bajo costo. En éste proyecto se utiliza un modelo de simulación implementado en ROS y Gazebo para probar algoritmos de control y estimación de estados. Adicionalmente, se utilizan datos tomados por el robot para realizar una estimación de estados probabilística. En el marco teórico se profundizan los modelos y algoritmos utilizados así como conceptos relacionados a éstos. En la descripción de la solución se presentan detalles de la implementación de cada algoritmo mientras que en la sección *experimentos* se presentan algunos experimentos realizados con sus resultados. Por último, se presentan las conclusiones del trabajo.

II. MARCO TEÓRICO

El modelo cinemático que describe un carro, es conocido como Ackermann. Como se muestra en la Fig.1, se realiza una simplificación estableciendo dos ruedas situadas a la mitad de los ejes delatero y trasero.

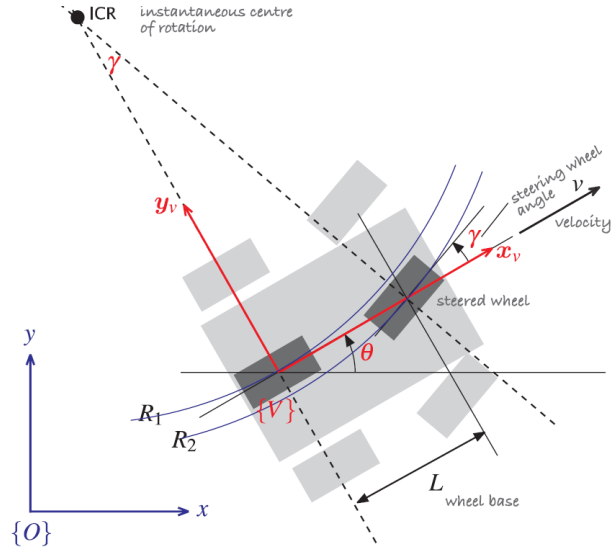

Fig. 1. Modelo de Ackerman. [1]

La pose del vehículo está representada por el marco de coordenadas V, con su eje x en la dirección de avance del vehículo y su origen en el centro de la parte trasera eje. La configuración del vehículo está representada por las coordenadas generalizadas $q = (x, y, \theta)$. El robot solamente puede tener velocidad en su eje x, siendo un modelo no-holónomo descrito por $\dot{y}\cos\theta - \dot{x}\sin\theta = 0$. La velocidad angular es $theta = v/R1$, obtenida a partir del radio de giro

$R1 = L/tan(lambda)$ donde L es la longitud del vehículo. Es importante resaltar que el ángulo de dirección lambda está limitado mecánicamente y su valor máximo lo dicta el valor mínimo de $R1$.

El método elegido para la determinación del estado en el que se encuentra el robot es el de un filtro de histogramas. Este método es una solución probabilística derivada del filtro de Bayes. La idea es discretizar un espacio de estados continuo, con el propósito de aplicar un filtro de Bayes a estos estados y así determinar el estado actual del robot. Se depende de la lectura de los sensores disponibles y la entrada de control del robot. En este caso, el procesamiento de imagen presentado anteriormente proporciona la información necesaria para estimar la posición actual del robot, pues obtiene las líneas que son visibles para el robot en un momento determinado. En la siguiente sección se explicará la manera en que esta información permite estimar el estado actual del robot. La entrada de control utilizada es el *steering* o viraje publicado en el tópico */manual_control/steering*. Esta señal permite hacer una predicción de los siguientes estados que tomará el robot, pues ya cuenta con una estimación de su estado actual y mediante el uso de probabilidad condicional puede predecir a qué estado es más probable que se dirija. Las fórmulas de probabilidad condicional utilizadas son detalladas en la sección de Descripción de la solución.

A partir del modelo cinemático, se puede obtener un control que lleve al robot a una pose (x, y, θ) deseada. En la siguiente ecuación se muestra al modelo que se llega aplicacando una transformación a coordenadas polares y realizando cambios de variables.

$$\begin{pmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{pmatrix} = \begin{pmatrix} -\cos\alpha & 0 \\ \frac{\sin\alpha}{\rho} & -1 \\ -\frac{\sin\alpha}{\rho} & 0 \end{pmatrix} \begin{pmatrix} \nu \\ \gamma \end{pmatrix} \text{if } \alpha \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right]$$
$$\nu = k_\rho \rho$$
$$\gamma = k_\alpha \alpha + k_\beta \beta$$

Dos de los sensores con los que el robot puede obtener datos del exterior son el *LIDAR* y una cámara. Mediante el *LIDAR* se obtienen 360 puntos que indican la distancia a objetos. Dado que es una medición discreta, es necesario procesar los datos para determinar objetos sólidos con cierto nivel de precisión. La posición de cada objeto se obtiene con respecto al robot marco de referencia V del robot. Si se requiere determinar la posición de un objeto en un marco de referencia global O es necesario realizar la transformación ${}^{O}\mathrm{P} = {}^{O}\mathrm{T}_{V}\,{}^{V}\mathrm{P}$, donde a T es la matriz de transformación homogénea.

Mediante el algoritmo 1 es posible obtener objetos a partir de las lecturas del *LIDAR*. El algoritmo itera sobre el arreglo de distancias y mediante una δ predefinida, determina cuáles puntos obtenidos pertenecen a distintos objetos. Es importante considerar el caso inicial del algoritmo. Otro caso especial es considerar la diferencia entre la primera y la última lectura ya que al tratarse de coordenadas polares, se podría tratar de un mismo objeto.

Data: Arreglo de distancias D
Result: Arreglo de objetos O
forall $d \in D$ **do**
  if $d \neq inf$ **then**
    if $d_{ant} - d > \delta$ **then**
      $O.add(o)$
      $o \leftarrow nuevoObjeto(d)$
    else
      $o \leftarrow actualizaObjeto(o, d)$
    end
  end
  $d_{ant} \leftarrow d$
end

Algorithm 1: Detección de Objetos

II-A. Procesamiento de imagen

Las tranformaciones necesarias para destacar las características de líneas y llegar a una I(x,y) deseada comienzan desde la detección de bordes en la imagen. Un borde es una frontera entre dos regiones con propiedades de nivel de gris o tono relativamente distintas, una manera de expresar los cambios es mediante el uso de derivadas. Un alto cambio de pendiente indica un cambio importante en la imagen.

El primer paso es convolucionar la imagen con el kernel de Sobel, un operador de diferenciación que trabaja en el dominio discreto, combinado con un suavizado gaussiano. Es una aproximación del gradiente de una función de intensidad de la imagen, se calcula por separado los cambios horizontales y verticales. Posteriormente, con el operador de Canny se adelgazan los bordes capturados, el objetivo es obtener bordes de 1 pixel de grosor y considerar únicamente pixeles cuya magnitud es máxima en bordes gruesos y descartar aquellos cuyas magnitudes no alcancen ese máximo. En otras palabras, la histéresis de umbral a la supresión no máxima permite eliminar máximos procedentes de ruido.

En 1962 Hough propuso una técnica para la detección de líneas rectas en imágenes digitales. La Transformada de Hough en general recibe como entrada una imagen binaria de los bordes que se detectaron previamente en la imagen original. Hough definió un espacio paramétrico utilizando como representación de la recta su fórmula en forma simplificada, b=-mx+y. El espacio paramétrico es una matriz bidimensional subdividida en celdas acumulador inicializadas en 0, un eje corresponde a m y el otro a b. Cada recta se representa en el espacio paramétrico en una ubicación de esta matriz y se utiliza para llevar el conteo de todas los puntos (xi , yi) que pertenecen a una misma recta. Los pasos para la detección de líneas rectas se pueden resumir como: Calcular el gradiente de una imagen, subdividir el plano en parámetro (m,b) o (ro,theta), examinar el contenido de las celdas del acumulador con altas concentraciones de pixeles y, finalmente, examinar la relación, principalmente de continuidad, entre el conjunto de los píxeles de las celdas elegidas, para ello se calcula la distancia entre los píxeles desconectados, se busca que un vacío entre un punto y su vecino más próximo no supere un umbral dado.

A estas alturas es posible realizar un modelo matemático

para un conjunto de datos experimentales utilizando el algoritmo de RANSAC. Un ajuste robusto, ya que reconoce la presencia de valores outliers en los datos. Para la extracción de una línea se inicia con un conjunto de puntos y se ajusta a una línea de dos puntos seleccionados al azar, de tal forma que no se repita el mismo par de puntos en las siguientes iteraciones. Para no hacer un nuevo cálculo de una misma línea supuesta, entonces se calculan los parámetros de la línea con los puntos escogidos. A continuación, todas las distancias de los puntos a la línea son calculadas. Teniendo en cuenta los valores de las distancias, se separan aquellas que se encuentran por encima del cincuenta porciento de la distancia máxima de los de la mediana, se calculan los datos outliers, para luego calcular el número de puntos pertenecientes a la línea.

III. DESCRIPCIÓN DE LA SOLUCIÓN

III-A. Estimación del estado propio

Para la estimación del estado propio se construyó una tabla de verdad con las posibles combinaciones en cuanto a la detección de líneas por parte de los tópicos *line/left*, *line/center*, *line/right* para las líneas izquierda, central y derecha respectivamente. A cada una de estas combinaciones se le asignó una combinación de 'hits' que indican si es probable que se encuentre en el carril correspondiente de acuerdo a las líneas detectadas, esta combinación de 'hits' es el filtro de histogramas. Por ejemplo, en la primera fila de la siguiente tabla se ilustra el caso en el que no se detecta ninguna línea (I para la línea izquierda, C para la línea central y D para la línea derecha). En este caso los estados que parecen probables son los que ubican al carro fuera de los carriles (AI para Afuera Izquierda y AD para Afuera Derecha) pues es en estos estados donde es más probable que no se encuentre ninguna línea en la imagen. La explicación de las demás combinaciones de 'hits' son similares y los estados restantes son: LI para (sobre la) Línea Izquierda, CI para Carril Izquierda, LC para (sobre la) Línea Central, CD para Carril Derecha y LD para (sobre la) Línea Derecha.

Estados			**Hits**						
I	**C**	**D**	**AI**	**LI**	**CI**	**LC**	**CD**	**LD**	**AD**
0	0	0	x	0	0	0	0	0	x
0	0	1	x	x	x	x	x	0	0
0	1	0	0	x	0	x	0	x	0
0	1	1	0	x	x	x	x	0	0
1	0	0	0	0	x	x	x	x	x
1	0	1	0	0	x	0	x	0	0
1	1	0	0	0	x	x	x	x	0
1	1	1	0	0	0	x	0	0	0

Teniendo esta nueva estimación, se utilizó el valor obtenido del tópico */manual_control/steering* para hacer la predicción del estado siguiente o al que hay más probabilidad de converger. Este valor fue separado en varios casos, puesto que en algunos casos el viraje es hacia la izquierda y en algunos es hacia la derecha e incluso cuando se sabe en qué dirección es, también existen distintas intensidades. A cada rango se le asignaron entonces varias funciones de probabilidad condicionada, las cuales se detallan a continuación. El símbolo u se refiere al steering, mientras que X_i se refiere a que el robot se encuentre en el estado i:

- 0 <= u < 45 P($x_{i0}|x_i$)=.4 P($x_{i-2}|x_i$)=.6
- 45 <= u < 85 P($x_{i-1}|x_i$)=.2 P($x_{i-2}|x_i$)=.8
- 85 <= u <= 95 P($x_i|x_i$)=.7 P($x_{i+1}|x_i$)=.15 P($x_{i-1}|x_i$)=.15
- 95 <= u < 135 P($x_{i+1}|x_i$)=.2 P($x_{i+2}|x_i$)=.8
- 135 <= u < 180 P($x_{i+1}|x_i$)=.4 P($x_{i+2}|x_i$)=.6

Mediante el algoritmo 1, se determina la distancia promedio al objeto con mayor número de lecturas. Primero, se descartan distancias que no corresponden a un obstáculo (objetos del propio carro o valores *infinitos*) para conservar las distancias de ángulos contiguos cuya diferencia sea menor a una δ. Entonces, se obtiene el promedio de la distancia y el ángulo medio del objeto. La pose del obstáculo, se obtiene a partir del marco de referencia del coche. Es necesario convertir los datos del obstáculo de coordenadas polares a coordenadas cartesianas utilizando:

$$x = d * \sin(\alpha)$$

$$y = d * \cos(\alpha)$$

Una vez estimada la posición del obstáculo respecto al robot, se determina la velocidad del obstáculo utilizando un filtro de kalman. El vector de medias está compuesto por la posición del obstáculo y su velocidad $(x, y, \dot{x}, \dot{y})$, siendo las velocidad las variables ocultas.

El filtro de Kalman se basa en dos simples acciones:

- Actualización por medición
- Predicción de movimiento

La implementación del filtro se muestra a continuación:

```
void kalman_filter()
{
        MatrixXf m_hat = a_t * m_ant;
        VectorXf m_hat = a_t * m_ant;
        MatrixXf cov_hat = a_t * cov_ant * a_t.transpose() + r_t;

        MatrixXf s_aux = c_t * cov_hat * c_t.transpose() ;
        MatrixXf k_t = cov_hat * c_t.transpose() * s_aux.inverse();

        MatrixXf aux_1 = z_t  - c_t * m_hat;
        MatrixXf aux_2 = k_t * aux_1;

        m_t = m_hat + k_t * aux_1 ;
        m_t = m_hat + k_t * ( z_t  - c_t * m_hat);

        s_t = (MatrixXf::Identity(4,4) - k_t * c_t) * cov_hat;
        cov_t = (MatrixXf::Identity(4,4) - k_t * c_t) * cov_hat;
}
```

La matriz que representa la transición de estados es *a_t* de dimesión 4x4, la matriz de medición es *c_t* de dimensión 2x4. Cabe mencionar que las matrices se operaron con Eigen.

Primero, se intenta predecir obteniendo la incertidumbre. Posteriormente, en el proceso de medición, obtenemos la innovación, después la proyección de la incertidumbre y el ruido en las mediciones, luego la ganancia de Kalman y finalmente volvemos a obtener la incertidumbre. El algoritmo pulirá los resultados progresivamente en cada iteración.

Utilizando un algoritmo de control PI se determina la señal de control del carro. El error se obtiene utilizando la diferencia entre la velocidad estimada del obstáculo y la velocidad conocida del carro. Un aspecto relevante de la solución es que la estimación obtenida mediante el filtro de kalman toma varias iteraciones para converger al inicio, por lo que la señal de control no se aplica hasta cierto tiempo t despues de iniciada la estimación.

Una manera de lograr que el coche aprenda a mantenerse dentro del carril, sería utilizando un modelo de aprendizaje por refuerzo, como *Q learning*.

Éste método parte de los procesos de decisión de Markov (*MDP*) que toman información de los estados y eligen una acción en base a la cuál se obtendrá una *recompensa* cuyo valor depende de la decisión tomada. El método de Q learning consiste en obtener la selección óptima de decisiones en base a un conjunto de MDPs para, así, maximizar las recompensas. Asignando un valor más alto a la recompensa que mantiene al robot dentro del carril que a la recompensa obtenida si se sale, aumentamos la probabilidad de que se mantenga en el camino.

IV. EXPERIMENTOS

IV-A. Filtro de Kalman

Para validar la implementación del filtro de Kalman, se colocó un segundo carro C_2 frente al carro C_1 que tiene implementado el filtro. El objetivo es que C_1 *siga* a C_2, es decir, $(\dot{x}_{C_1}, \dot{y}_{C_1}) \sim (\dot{x}_{C_2}, \dot{y}_{C_2})$. Para determinar la velocidad de C_1 se utiliza la estimación del filtro de Kalman como entrada de un algoritmo PI. Durante la prueba, la pose deseada de C_2 es dinámica: se tienen cambios en su orientación y dirección. En la Fig. 2 se muestra el error en el tiempo que se obtuvo en la prueba. Debido a que el filtro de Kalman necesita un tiempo para estabilizar las estimaciones iniciales, no se calcula la señal de control en las primeras iteraciones. Aunque se tienen varios *picos*, estos corresponden con cambios en la dirección de C_2, el error oscila alrededor del cero.

En la Fig. 3 se muestra la suma del error obtenida para poder aplicar el control PI. Se puede observar que se incrementa rápidamente para comenzar a oscilar alrededor de 0.7. Los cambios que se presentan corresponden a cambios en la dirección de C_2. Por último, en la Fig. 4 se muestra la señal de control aplicada a cada tiempo *t*, obtenida del algoritmo PI.

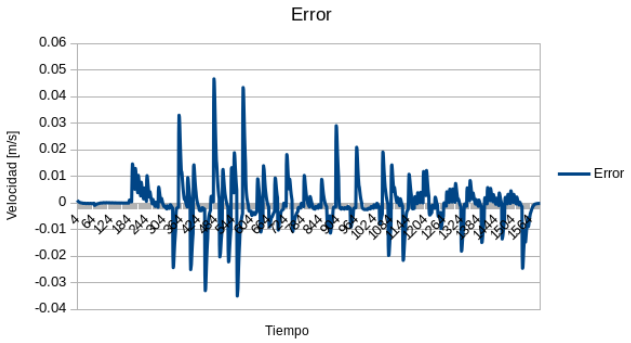

Fig. 2. Error en el tiempo

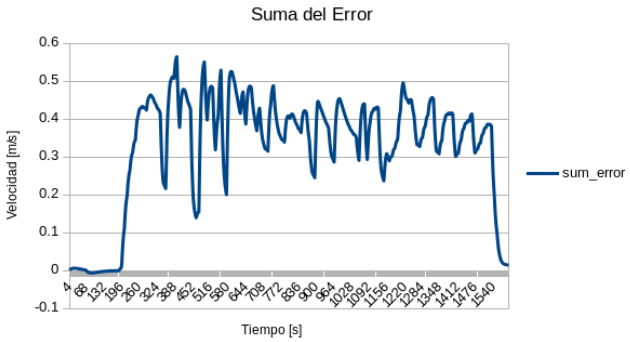

Fig. 3. Suma del Error en el tiempo

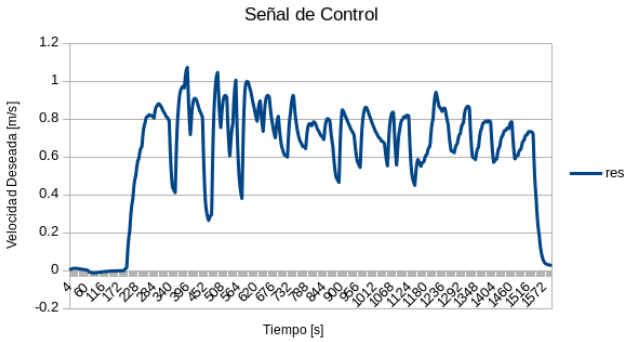

Fig. 4. Señal de Control Aplicada al Robot

V. CONCLUSIONES

El objetivo de éste proyecto fué integrar los conceptos vistos a lo largo del curso, ésta implementación incluye la estimación del estado del coche en el camino y la de un obstáculo con respecto al coche, que abarca desde los primeros temas de poses hasta robótica basada en probabilidad.

El control del auto, actualmente sólo sigue al obstáculo pero es un primer paso para implementar acciones más complejas como rebasar al obstáculo en moviemiento, que el coche se estacione, entre otras.

En la implementación del filtro de Kalman, la principal complicación fué definir las matrices de transición de estados y la función de medición. Además, también se presentaron problemas debido a que no sabíamos qué valores asignar a la matriz que representa el ruido en el sistema.

Los resultados obtenidos fueron satisfactorios, sin embargo un área de oportunidad para éste tipo de proyecto, es utilizar algún tipo de aprendizaje como, por ejemplo, por refuerzo para que el coche se mantenga dentro del carril de la carretera.Un aprendizaje así, haría más seguro el trayecto del coche en una situación en la que el camino sea continuo y no se necesite rebasar un obstáculo.

REFERENCIAS

[1] Peter Corke. *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*. Springer Tracts in Advanced Robotics. Springer Berlin Heidelberg, 2011. ISBN: 9783642201448. URL: `https://books.google.com.mx/books?id=6-kLBwAAQBAJ`.

[2] Daniel J Fagnant y Kara Kockelman. “Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations”. En: *Transportation Research Part A: Policy and Practice* 77 (2015), págs. 167-181.

[3] Michael Montemerlo y col. “Junior: The stanford entry in the urban challenge”. En: *Journal of field Robotics* 25.9 (2008), págs. 569-597.

[4] Raul Rojas y Zahra Boroujeni. *AutoModelCar*. `https://github.com/AutoModelcar`. 2017.

[5] Sebastian Thrun, Wolfram Burgard y Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

[6] Chris Urmson y col. “Autonomous driving in urban environments: Boss and the urban challenge”. En: *Journal of Field Robotics* 25.8 (2008), págs. 425-466.