Tarea 01 -Posicionamiento de un robot movil-

Robótica, Ingeniería Mecatrónica
Instituto de Ingeniería y Tecnología, UACJ
Catedrático: Dr. Edgar A. Martinez
José Félix Velazco Velazco
al140621@uacj.mx

23 de marzo de 2018

Abstract

En este reporte se verá el modelo de control para manejar un amigobot mediante el vector $\vec{u}$, que contiene v y ω. Con esto hecho se obtendrá la posición del robot mediante un sistema de visión que se localizó en el techo del laboratorio, esto lleva consigo el procesamiento de imágenes con el fin de obtener el mejor resultado posible. Además, también se obtendrá la posición odométrica que será provista por el amigobot directamente.

1. Modelo de generacion de trayectorias

Para la trayectoria a seguir del robot, se optó por tomar una función $y(x) = \sin(x)$. La trayectoria a seguir debe de tener una amplitud de -0.8 a 0.8, y que se complete una onda completa empezando de 0 a 2, por lo que la función $y(x)$ como se ve en la ecuación (1).

$$y(x) = 0{,}8 \cdot \sin\left(\frac{2\pi}{x^{max}} \cdot x\right) \tag{1}$$

Donde x^{max} será la distancia máxima de la onda, en este caso es igual a 2. Se obtuvieron 1000 valores de x, de 0 a 2 con incrementos de 0.002 y con estos valores se calcularon los valores de $y(x)$ y teniendo estos valores en una tabla, se obtuvieron $\dot{x}$ y $\dot{y}$, mediante derivadas numéricas, obteniendo las ecuaciones (2) y (3) para obtener nuevamente 1000 valores

$$\dot{x} = \frac{dx}{dt} = \frac{x_t - x_{(t-1)}}{\Delta t} \tag{2}$$

$$\dot{y} = \frac{dy}{dt} = \frac{y_t - y_{(t-1)}}{\Delta t} \tag{3}$$

Con estos valores, se obtuvieron los valores de $\dot{s}$ y θ con las ecuaciones (4) y (5).

$$\dot{s} = \sqrt{\dot{x}^2 + \dot{y}^2} \tag{4}$$

$$\theta = \arctan\left(\frac{\dot{y}}{\dot{x}}\right) \tag{5}$$

Al obtener los valores de $\dot{s}$ y θ, estos valores se meterán en una regresión de grado 7,

Se sabe que $y = A \cdot x$, al despejar x, obtendremos la solución inversa $x = A^{-1} \cdot y$. En la ecuación (6) se puede ver en su forma matricial, donde y_i se sustituirá por $\dot{s}_i$ y θ_i.

$$\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_7 \end{bmatrix} = \begin{bmatrix} N & \Sigma t & \cdots & \Sigma t^7 \\ \Sigma t & \Sigma t^2 & \cdots & \Sigma t^8 \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma t^7 & \Sigma t^8 & \cdots & \Sigma t^{14} \end{bmatrix}^{-1} \cdot \begin{bmatrix} \Sigma(y_i) \\ \Sigma(t_i y_i) \\ \vdots \\ \Sigma(t_i^7 y_i) \end{bmatrix} \tag{6}$$

Esta operación nos dará los polinomios de $v(t) = \dot{s}(t)$ y $\theta(t)$.

$$\begin{aligned} v(t) = {} & -4{,}19 \times 10^{-5} + 0{,}02t - 0{,}005t^2 \\ & +5{,}88 \times 10^{-4}t^3 - 3{,}17 \times 10^{-5}t^4 \\ & +8{,}35 \times 10^{-7}t^5 - 8{,}57 \times 10^{-9}t^6 \end{aligned} \tag{7}$$

$$\begin{aligned} \theta(t) = {} & -9{,}52 \times 10^{-5} + 0{,}06t - 0{,}015t^2 \\ & +0{,}0013t^3 - 5{,}89 \times 10^{-5}t^4 + 1{,}41 \times 10^{-6}t^5 \\ & -1{,}43345 \times 10^{-8}t^6 \end{aligned} \tag{8}$$

Se tiene la velocidad lineal $v(t)$ como se ve en la ecuación (7). Para la velocidad angular, solo derivamos el polinomio $\theta(t)$, que se ve en la ecuación (8), para obtener finalmente $\omega(t)$

$$\begin{aligned} \omega(t) = {} & 0{,}06 - 0{,}029t + 0{,}004t^2 - 2{,}4 \times 10^{-4}t^3 \\ & +7{,}076 \times 10^{-6}t^4 - 8{,}6 \times 10^{-8}t^5 \end{aligned} \tag{9}$$

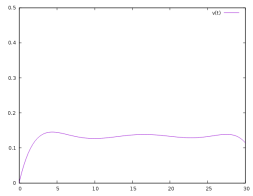


Figura 1: Gráfica de la velocidad lineal $v(t)$

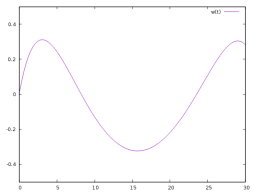


Figura 2: Gráfica de la velocidad angular $\omega(t)$

En las figuras 1 y 2 se pueden observar el comportamiento de estas velocidades.

2. Modelo odométrico

Para esta sección, se utilizará un modelo de 3 ruedas, donde una actua como *steer*, y las otras dos como *drive*, tal y como se muestra en la figura 3

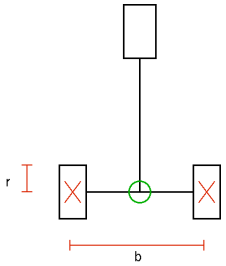


Figura 3: Esquema básico del amigobot

2.1. Odómetros

Tomando en cuenta que el amigobot sensa mediante encoders, entonces se obtiene el siguiente modelo

$$s = \frac{2\pi r}{R} * \eta \tag{10}$$

Con una derivada numérica sacamos las velocidades de cada llanta, así como se ve en la ecuación (11)

$$v_D = \frac{2\pi r}{R\Delta t}(\eta_{R2} - \eta_{R1})$$

$$v_L = \frac{2\pi r}{R\Delta t}(\eta_{L2} - \eta_{L1}) \tag{11}$$

La velocidad lineal del amigobot sería el promedio de las velocidades de las llantas. Y a partir de la ecuación (12), podemos inferir un modelo en donde se vean implicados las velocidades angulares de las llantas, como en la ecuación (13).

$$v_t = \frac{V_R + V_L}{2} \tag{12}$$

$$v_{(R,L)} = r\varphi_{(R,L)}$$

$$\therefore v_t = \frac{r\dot{\varphi}_R + r\dot{\varphi}_L}{2}$$

$$v_t(\dot{\varphi}_R, \dot{\varphi}_L) = \frac{r}{2}(\dot{\varphi}_R + \dot{\varphi}_L) \tag{13}$$

Ahora obtendremos el modelo de velocidad angular en función de las velocidades angulares de las llantas, asumiendo que

$$\hat{v}_t = v_R - v_L \tag{14}$$

Entonces, tendremos que

$$\omega_t = \frac{\hat{v}_t}{\frac{b}{2}} = \frac{2}{b}\hat{v}_t \tag{15}$$

$$\omega_t(\dot{\varphi}_R, \dot{\varphi}_L) = \frac{2r}{b}(\dot{\varphi}_R - \dot{\varphi}_L) \tag{16}$$

Y con la ecuación (13) y (16), será nuestra ley de control.

2.2. Cinemática

A partir de nuestra ley de control, vista en el tema pasado, podemos describirlo de manera matricial.

$$\begin{bmatrix} v_t \\ w_t \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{2r}{b} & \frac{2r}{b} \end{bmatrix} \cdot \begin{bmatrix} \dot{\varphi}_R \\ \dot{\varphi}_L \end{bmatrix} \tag{17}$$

Para obtener nuestro vector de control, así como se ve en la ecuación (17), tendremos que nuestra matriz de restricción cinemática lo multiplicaremos por nuestro vector de variables de control independiente, las cuales son las velocidades angulares de las llantas.

Esto se hace cuando se conocen las velocidades angulares de las llantas, pero se desconoce el valor las velocidades v_t y ω_t.

En caso de que si sepamos las velocidades v_t y ω_t, pero no las velocidades $\dot{\varphi}_R$ y $\dot{\varphi}_L$, entonces se

opta por la solución inversa, con lo que tendriamos:

$$\begin{bmatrix} \dot{\varphi}_R \\ \\ \dot{\varphi}_L \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \\ \frac{2r}{b} & \frac{2r}{b} \end{bmatrix}^{-1} \cdot \begin{bmatrix} v_t \\ \\ w_t \end{bmatrix} \tag{18}$$

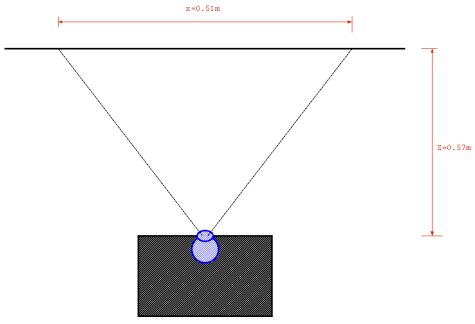
3. Modelo de sensado visual

3.1. Posicionamiento de la cámara

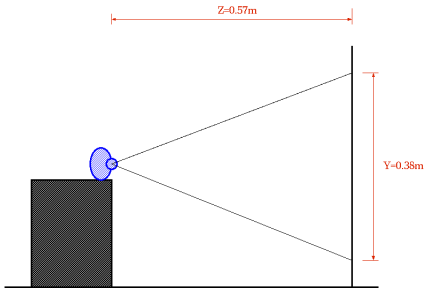
La labor de calibración de la cámara, es una parte fundamental al momento de querer utilizar un sistema de referencia global, esto es debido a que si no se calibra adecuadamente, de nada serviría lo demás de visión.

Se puso la cámara en un sitio lo más horizontal posible, y en una pared lo más ortogonal posible a este plano, se pusieron unas marcas del campo de vision.

Para calcular los angulos $\varphi_{Horizontal}$ y $\varphi_{Vertical}$, del campo de visión, se acomodó la cámara, así como se ve en la figura 4.



(a) Vista superior



(b) Vista lateral

Figura 4: Vistas del experimento para obtención del campo de visión.

$$\varphi_H = 2\arctan\left(\frac{x}{2z}\right)$$
$$\varphi_v = 2\arctan\left(\frac{y}{2z}\right) \tag{19}$$

Las ecuaciones (19), muestran la manera de calcular los angulos de campo de visión que posee la cámara, las cuales, sustituyendo los valores, tendríamos.

$$\varphi_H = 2\arctan\left(\frac{0{,}51}{2(0{,}57)}\right)$$
$$\varphi_V = 2\arctan\left(\frac{0{,}38}{2(0{,}57)}\right)$$

Obteniendo entonces los valores vistos en la ecuación (20)

$$\varphi_H = 48{,}21^\circ \ , \quad \varphi_V = 36{,}87^\circ \tag{20}$$

Después de esto, se localizó la cámara en el techo, con el fin de abarcar el mayor campo de visión posible. Al estar completamente calibrada, se obtuvo que la altura $z = 2{,}74m$ desde el lente de la cámara hasta la marca del amigobot, donde el amigobot estaba en el centro del campo de visión, tal y como se ve en la figura 5

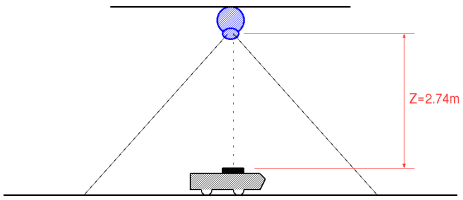


Figura 5: Localización de la cámara para el sistema de visión global

De las ecuaciones (19), podemos despejar para obtener los valores X y Y

$$X = 2z\tan\left(\tfrac{\varphi_H}{2}\right) = 2(2{,}74)\tan\left(\tfrac{48{,}21}{2}\right)$$
$$Y = 2z\tan\left(\tfrac{\varphi_V}{2}\right) = 2(2{,}74)\tan\left(\tfrac{36{,}87}{2}\right) \tag{21}$$

$$X = 2{,}45m \quad Y = 1{,}83m \tag{22}$$

Con los valores de X y Y podemos finalmente deducir cuanto sería equivalente en metros cada pixel. Tomando en cuenta que las imágenes que tomaremos son de 640x480 pixels, se puede deducir que:

$$X_{pixel} = \frac{1_{pixels}(2{,}45m)}{640_{pixels}} = 3{,}82mm$$
$$Y_{pixel} = \frac{1_{pixels}(1{,}83m)}{480_{pixels}} = 3{,}82mm \tag{23}$$

Con los valores obtenidos en las ecuaciones (23) podremos obtener la posición más adelante. Y con esto último se procede a tomar las fotos de la trayectoria del amigobot antes vista.

3.2. Mejoramiento y filtrado

Una vez tomada todas las imágenes, se procede a utilizar un tipo tipo de mejoramiento, cambiando el brillo y el contraste de estas. En la image 6 se puede observar la imagen original tomada por la cámara.

Figura 6: Imagen original

```
src.convertTo(dst,-1, 3,10);

for ( int i = 1; i < M; i = i + 2 )
{
        medianBlur ( dst, dst, i );
}
```

En el código anterior, se ve la función convertTo, que lo que hace es modificar el brillo y el contraste, en este caso particular se optó por darle un contraste de 3, y un brillo de 10.

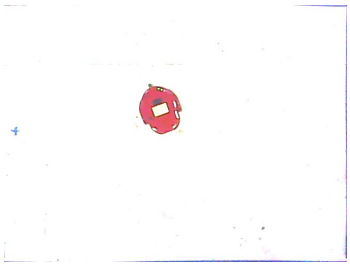


Figura 7: Imagen con mejoramiento y filtrado aplicado

La función medianBlur, lo que hace, es que aplica un filtro de mediana en la imagen, esto con el fin de suavizar más los colores, en este caso, se utilizó un kernel de 3x3.

3.3. Filtrado Morfológico

Para el filtrado morfológico, primero necesitamos convertir la imagen, en BGR, en una imagen binarizada, esto, se logra mediante la división de la imagen orignal en 3 canales distintos: B,G,R. Con esto, dejándolo todo para segmentar el color amarillo elimindando el canal B.
Obtenida la imagen binarizada, aplicariamos el siguiente código.

```
float morph_size = 4;

Mat kernel = getStructuringElement
(MORPH_RECT, Size(2*morph_size+1,
2*morph_size+1), Point(morph_size,
 morph_size) );


        erode(r, r, kernel);


morph_size = 4;

kernel = getStructuringElement
(MORPH_RECT, Size(2*morph_size+1,
2*morph_size+1), Point(morph_size,
morph_size) );

        dilate(r, r, kernel);
```

Aquí se debe de destacar más que nada las funciones *erode* y *dilate*, la cual, la primera lo que hace es ir degradando los trozos pequeños que quedan por ahí esparcidos en la imagen binarizada. Una vez degradada, se procede a volver a su tamaño original todos los demás elementos con la función *dilate*. En este caso se utilizo un kernel de 9x9. El resultado se puede apreciar en la figura 8

Figura 8: Imagen filtrada con operadores morfológicos

3.4. Segmentación por etiquetado de componentes conectados

```
Mat labels,stats, centroids;

int nLabels =
connectedComponentsWithStats
```

```
(r, labels, stats, centroids,8,
 CV_16U);

int z=0;
for(z=0; z<nLabels;z++)
{
        int area=0, temp=0;
        area=stats.at<int>(z,4);
        if(area>320 && area<500)
                break;
}
float x= centroids.at<double>(z,0);
float y= centroids.at<double>(z,1);
```

La función *connectedComponentsWithStats*() es la función que nos ayuda a poder aplicar el modelo de segmentación por etiquetado de componentes conectados, que a su vez, nos puede proporcionar datos como el área, o el centroide del objeto, por lo que con esto, se pudo obtener lo que se muestra en la figura 9

Figura 9: Marca segmentada (en negativo)

3.5. Modelo de visión 3D

Para el modelo de visión 3D, se tuvo que obtener el centroide de la marca durante todo el recorrido del amigobot, el cual se puede obtener mediante la función vista anteriormente *connectedComponentsWithStats*()

Los valores de x y y están dados en pixeles, por lo que se debe de pasar convertir a metros.

```
float x= centroids.at<double>(z,0);
float y= centroids.at<double>(z,1);

float x1=0.0, y1=0.0;
x1=((x-320)*3.82*pow(10,-3));
y1=((-y+240)*3.82*pow(10,-3));

B(i,0)=x;
B(i,1)=y;
```

x_1 y y_1 sería la distancia en metros, mientras que x y y esá dada en pixels, por lo que primero, movemos el origen que viene por default localizado en la ezquina superior izquierda, en (320,-240). Una vez hecho esto, multiplicamos el pixel dado por $3{,}82 \times 10^{-3}$ que es la equivalencia en metros de cada pixel, obteniendo así la posición del sistema de visión global, como se aprecia en la figura 10

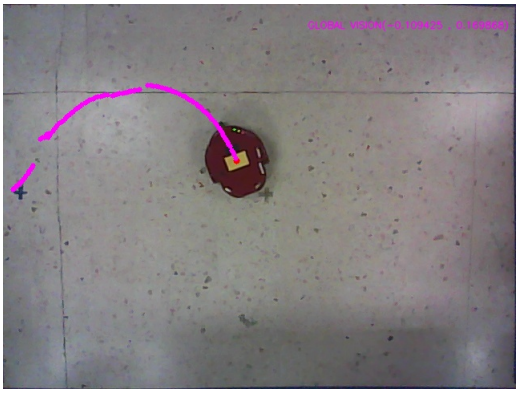


Figura 10: Sistema global de visión

4. Resultados

Las posiciones cartesianas se pueden observar en la figura 11

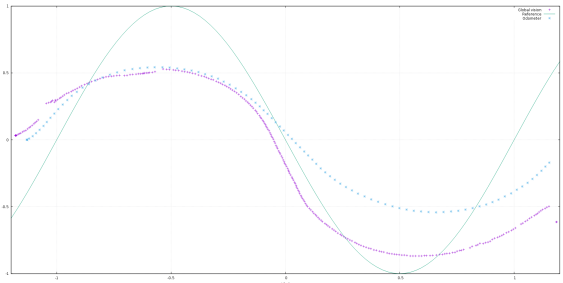


Figura 11: Gráfica de la posición cartesiana según el valor de referencia, la visión global, y la odometría

Las trayectorias también se pueden ver en la imagen 12, junto con la imagen del amigobot.

4.1. Error absoluto y relativo

En base de las ecuaciones (24) y (25), que se ven a continuación:

$$E_a = \left|\vec{P}_V - \vec{P}_R\right| \tag{24}$$

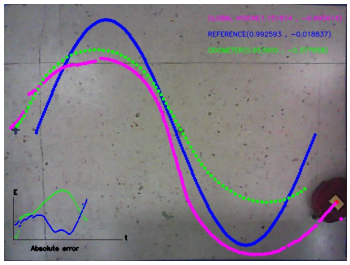


Figura 12: Recorrido del amigobot según: la odometría, la visión global, y la referencia

$$E_r = \left| \frac{\vec{P}_V - \vec{P}_R}{\vec{P}_V} \right| \tag{25}$$

Se puede sacar las gráficas de los errores absolutos de *Visión vs Odometria* y de *Visión vs Referencia*.

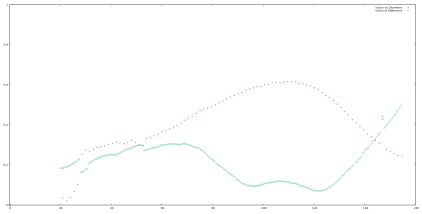


Figura 13: Gráfica del error absoluto de *Visión vs Odometria* y de *Visión vs Referencia*

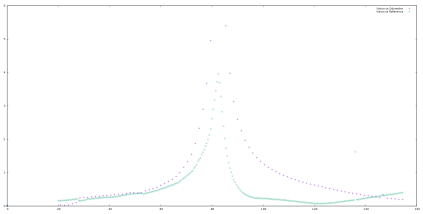


Figura 14: Gráfica del error relativo de *Visión vs Odometria* y de *Visión vs Referencia*

5. Conclusiones

Al final se logró cumplir con los requerimientos encomendados. Hubo una dificultad grande al tratar de conseguir el vector de control $\vec{u}$, pero se logró hallar el error en la programación que finalmente resolvió ese problema.

$$\omega(t) = 0{,}06 - 0{,}029t + 0{,}004t^2 - 2{,}4\times10^{-4}t^3 + 7{,}076\times10^{-6}t^4 - 8{,}6\times10^{-8}t^5 \tag{26}$$

$$y(x) = \pi x - \frac{\pi^3}{6}x^3 + \frac{\pi^5}{120}x^5 - \frac{\pi^7}{5040}x^7 \tag{27}$$

$$\omega(t) = -9{,}52\times10^{-5} + 0{,}06t - 0{,}015t^2 + 0{,}0013t^3 - 5{,}89\times10^{-5}t^4 + 1{,}41 \tag{28}$$

$$v(t) = -4{,}19\times10^{-5} + 0{,}02t - 0{,}005t^2 + 5{,}88\times10^{-4}t^3 - 3{,}17\times10^{-5}t^4 + \tag{29}$$