6

Índice general

Capítulo 2
Robots Móviles y Agentes Inteligentes

47

Capítulo 3
Robots Móviles y Agentes Inteligentes

1 Introducción y Generalidades

1.1 Introducción

Tipos de arquitecturas de los robots móviles.

1.2 Modelos Tradicionales

Características

Se tiene representación del medio ambiente

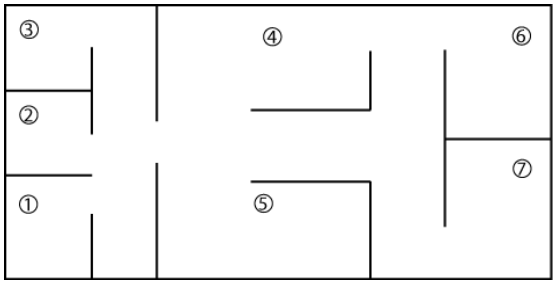
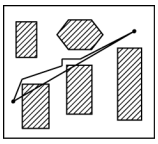


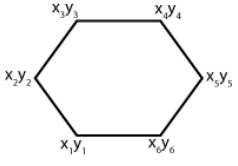
Figura 1.1: (Objeto mesa (cuarto 1 $x_1y_1, x_2y_2, \ldots, x_ny_n$))

Se planean las acciones y los movimientos del robot.



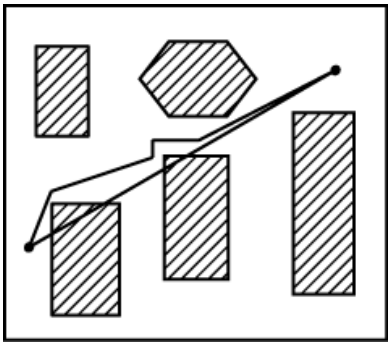
El robot debe llegar de 1 a 6. Dado que el robot conoce el mapa del lugar puede trazar un árbol y encontrar lo mejor ruta (rama de menor peso).

Sin embargo, desconoce los objetos que obstruyen su camino; por lo que no debe descartar las otras ramas.

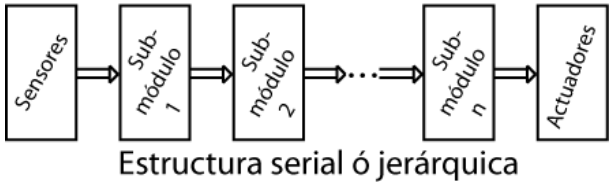


Si se conocen objetos obstáculo, se aproximan a polígonos y se almacenan sus vértices. Si uno intersecta el camino de línea

recta entre origen y destino, el robot se desplaza a la esquina más cercana y bordea el obstáculo.



Se tiene una organización serial, si un modulo falla... Se tiene un grupo de sensores para detectar el medio ambiente



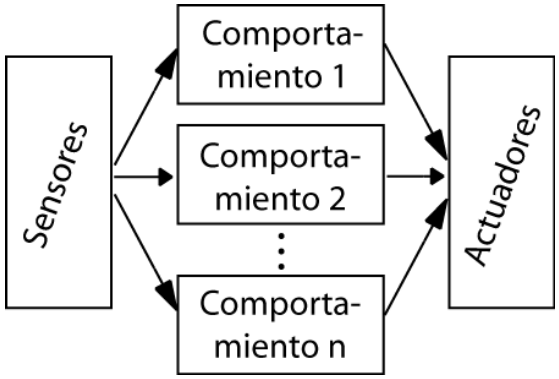
Estructura serial ó jerárquica

Este tipo de sistemas no es adecuado para entornos dinámicos y para robots que presentan errores en el movimiento y sensado.

1.3 Modelos Reactivos

Características:

- Basado en el comportamiento de los insectos
- No es necesaria una representación del medio ambiente
- No utiliza planeación de acciones ni de movimiento
- Es adecuado para entornos dinámicos y con errores en el sensado
- Esta basado en comportamientos funcionando en paralelo.

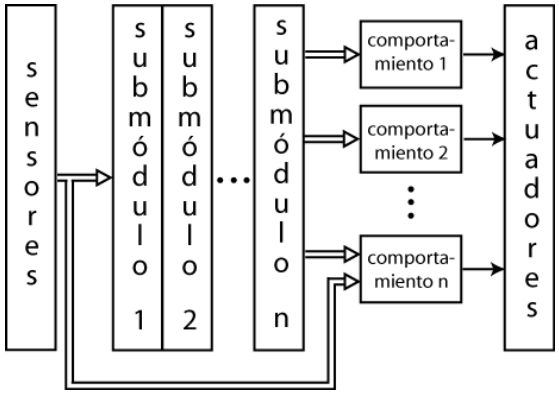


La salida de cada comportamiento debe ser instantánea a partir del momento que hay una entrada. Los comportamientos son independientes entre si. Por ejemplo si se tienen un grupo de robots en un campo con discos con la siguiente programación:

1. Moverse aleatoriamente hasta 2 o 3
2. Si se encuentra disco y no se porta disco, recoger el disco $\rightarrow$ 1
3. Si se encuentra disco y se porta disco soltar el disco $\rightarrow$ 1

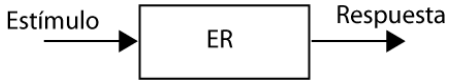
1.4 Modelos Híbridos

Se combinan las arquitecturas tradicionales y reactivas para suplir las deficiencias de cada una de ellos.

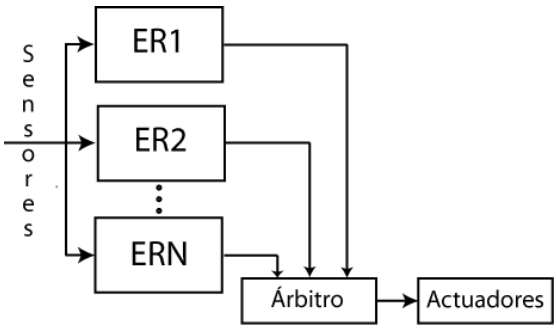


1.5 Comportamientos Reactivos

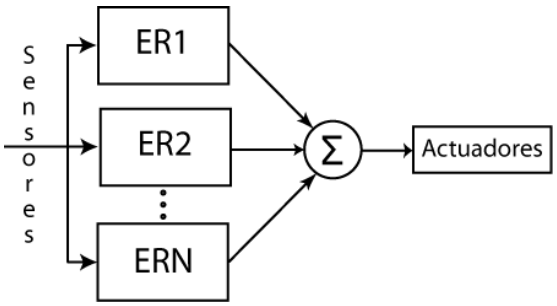
Se manejan mediante diagramas estimulo- respuesta o ER



O bien



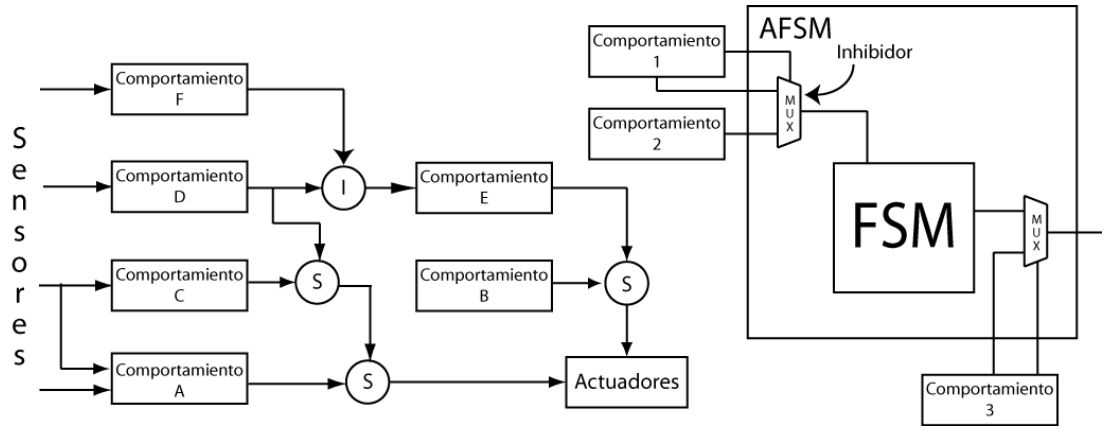
O bien



Inteligencia espontánea: los robots hacen algo que no sabrán que estaban haciendo. En un robot que tiene varios comportamientos coordinados por un agente, todos los comportamientos deben ofrecer una salida por ciclo de reloj, es decir, un comportamiento debe dar salida inmediata.

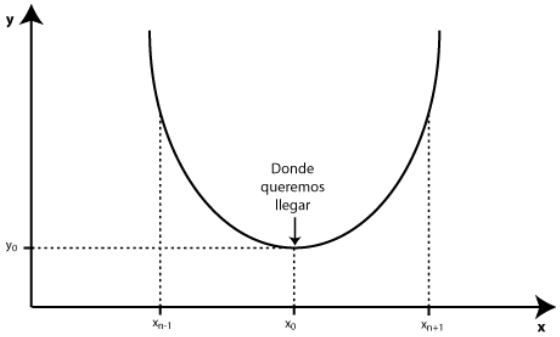
Ronald Brooks propone

Tenemos entonces:



1.6 Campos Potenciales

El destino se determina por un capo potencial de atracción y los obstáculos como campos potenciales de repulsión.



$$x_n = f(x_{n-1}) = x_{x-1}\,\delta\,\frac{dy}{dy}$$

Por ejemplo para una parábola $y = y_0 + (x - x_0)^2$

$$\frac{dy}{dx} = 2(x - x_0)$$

$$si\ \delta = \frac{1}{2} \Rightarrow x_{n-1} = -\frac{1}{2}(2(x_{n-1} - x_0)) = x_0,\ llegamos\ en\ un\ paso.$$

Esta técnica se conoce como: descendiendo por la pendiente más pronunciada o steppest descent.

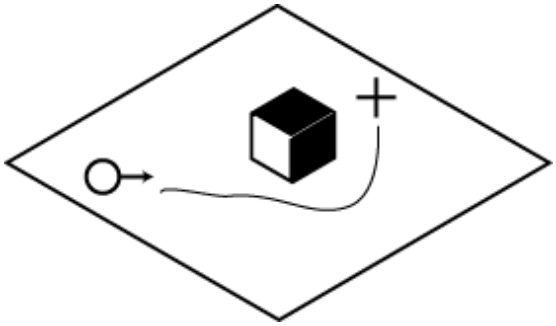
$$\overline{q}_n = [x_n, y_n] \qquad \textit{la posición del robot} \qquad \overline{q}_n = \overline{q}_{n-1} - \delta \overline{f}\left(\overline{q}_{n-1}\right)$$

donde $\overline{f}\left(\overline{q}_{n-1}\right)$ *es un vector fuerzas unitario en la dirección del gradiente* $\overline{f}(\overline{q}) = \dfrac{\overline{F(\overline{q})}}{\left|\overline{F}(\overline{q})\right|}$

donde $\overline{F}(\overline{q}) = \nabla U(\overline{q}) = \left(\dfrac{\partial u}{\partial x}\hat{i} + \dfrac{\partial u}{\partial x}\hat{j}\right)$

El gradiente del campo potencial *donde* $u(\overline{q}) = U_{atracción}(\overline{q}) + U_{repulsión}(\overline{q})$

Los campos potenciales atractivos y repulsivos y $\overline{F}(\overline{q}) = \overline{F}_{atr}(\overline{q}) + \overline{F}_{rep}(\overline{q})$ las fuerzas de atracción y repulsión



1.7 Campos Potenciales Atractivos

$$\overline{q} = (x, y) \textit{ Es la posición del robot}$$

$$\overline{q}_{dest} \textit{ Posición del punto al que queremos llegar}$$

$$\overline{F}_{atr}(\overline{q}) = \varepsilon_1(\overline{q} - \overline{q}_{dest}) \textit{ siempre que } |\overline{q} - \overline{q}_{dest}| \leq d_i \textit{ para } |\overline{q} - \overline{q}_{dest}| > d_i,$$
$$U_{atr}(\overline{q}) = \varepsilon_2 |\overline{q} - \overline{q}_{dest}| \textit{ o bien,}$$

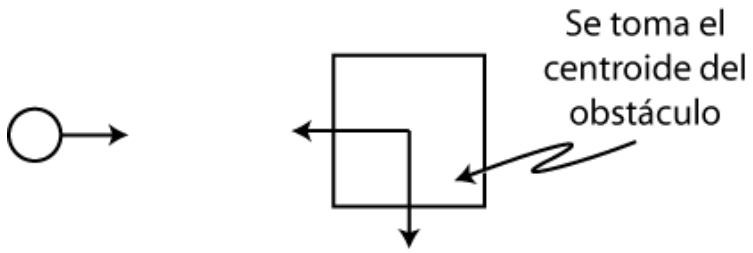
$$U_{atr}(x,y) = \varepsilon_2 \left((x-x_0)^2 + (y-y_0)^2\right)^{\frac{1}{2}}$$

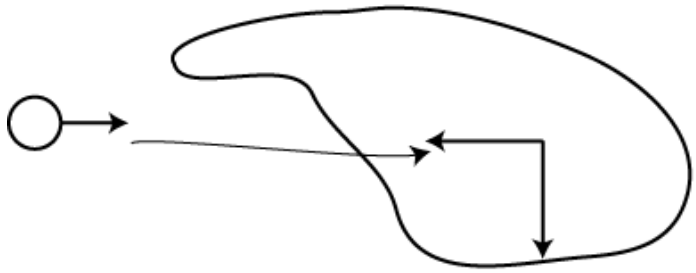
$$\frac{\partial U_{atr}(x,y)}{\partial x} = \frac{\varepsilon_2 2(x-x_0)}{2\sqrt{(x-x_d)^2 + (y-y_d)^2}}$$

$$\frac{\partial U_{atr}(x,y)}{\partial y} = \frac{\varepsilon_2 (y-y_0)}{\sqrt{(x-x_d)^2 + (y-y_d)^2}}$$

$$\Rightarrow \nabla U_{atr(x,y)} = \frac{\varepsilon_2 \left((x-x_0) + (y-y_d)\right)}{\sqrt{(x-x_d)^2 + (y-y_d)^2}} = \frac{\varepsilon_2 (\overline{q} - \overline{q}_d)}{|\overline{q} - \overline{q}_d|} = F_{atr}(\overline{q})$$

1.8 Campos Potenciales Repulsivos





Sin embargo para cuerpos muy grandes, esto no necesariamente se cumple puesto que aunque la distancia robot - centroide del obstáculo sea grande, una porción del obstáculo puede encontarse cerca del robot.

$$|\overline{q} - \overline{q}_{obs}| \leq d_0$$

$$U_{rep}(\overline{q}) = \frac{1}{2}\eta \left(\frac{1}{|\overline{q} - \overline{q}_{obs}|} - \frac{1}{d_0} \right)^2$$

$$U_{rep}(x,y) = \frac{1}{2}\eta \left(\frac{1}{\sqrt{(x - x_{obs})^2 + (y - y_{obs})^2}} - \frac{1}{d_0} \right)^2$$

$$\frac{\partial U_{rep}(x,y)}{\partial x} = \eta \left(\frac{1}{d_0} - \frac{1}{\sqrt{(x - x_{obs})^2 + (y - y_{obs})^2}} \right)^2 \left(\frac{x - x_{obs}}{((x - x_{obs})^2 + (y - y_{obs})^2)^{\frac{3}{2}}} \right)$$

$$\frac{\partial U_{rep}(x,y)}{\partial y} = \eta \left(\frac{1}{d_0} - \frac{1}{\sqrt{(x - x_{obs})^2 + (y - y_{obs})^2}} \right)^2 \left(\frac{y - y_{obs}}{((x - x_{obs})^2 + (y - y_{obs})^2)^{\frac{3}{2}}} \right)$$

$$\nabla U_{rep}(\overline{q}) = -\eta \left(\frac{1}{|\overline{q} - \overline{q}_{obs}|} - \frac{1}{d_0} \right) \left(\frac{1}{|\overline{q} - \overline{q}_{obs}|^2} \right) \left(\frac{\overline{q} - \overline{q}_{obs}}{|\overline{q} - \overline{q}_{obs}|} \right) = \overline{F}_{rep}(\overline{q})$$

Fuerza de repulsión del obstáculo k

$$\overline{F}_{rep}(\overline{q}) = 0 \;\; si \;\; |\overline{q} - \overline{q}_{obs}| > 0$$

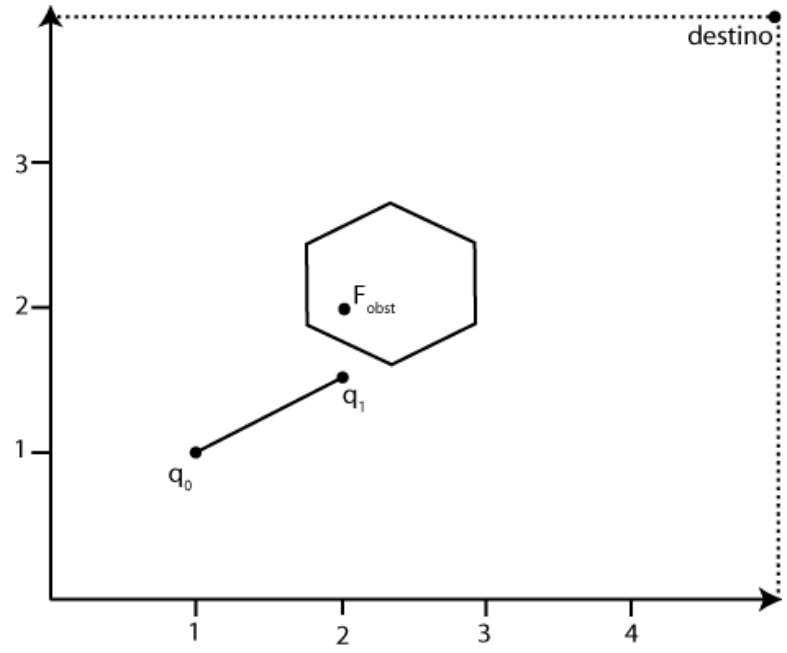
$$\overline{F}(\overline{q}) = \overline{F}_{atr}(\overline{q}) + \sum_{i=1}^{n} \overline{F}_{rep}(\overline{q})$$

Luego

$$f(\overline{q}) = \frac{\overline{F}(\overline{q})}{|\overline{F}(\overline{q})|}$$

$$\overline{q}_{i+1} = \overline{q}_i - \delta_i \overline{f}(q_i)$$

Ejemplo:



$$\overline{q}_0 = (1,1), \overline{q}_{obs} = (2,2), \overline{q}_{dest} = (5,4), d_0 = 5, \varepsilon_1 = 5, \eta = 2, \delta_0 = 1, \delta_1 = 10$$

$$\overline{F}_{atr}(\overline{q}_0) = \overline{F}_{atr}(1,1) = \varepsilon_1(\overline{q}_0 - \overline{q}_{dest}) = 1((1,1) - (5,4)) = (-4,-3)$$

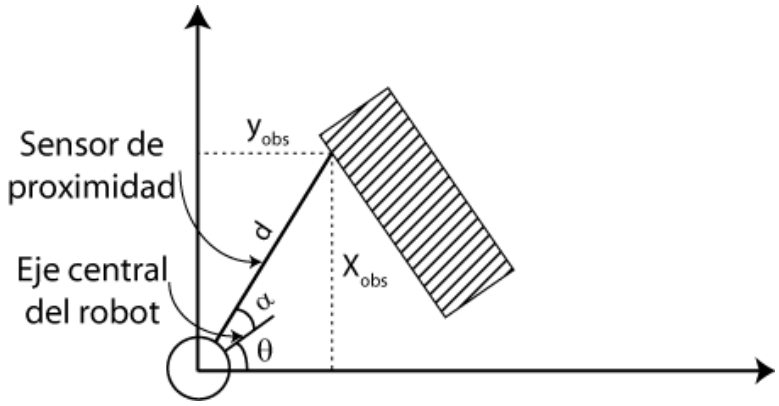
$$\overline{F}_{rep}(\overline{q}_0) = \left(-2\left(\frac{1}{\sqrt{2}} - \frac{1}{5}\right)\left(\frac{1}{2}\right)\left(\frac{(-1,-1)}{\sqrt{2}}\right)\right) = (0.3585, 0.3585)$$

$$\overline{F}(\overline{q}) = (-4,-3) + (0.3585, 0.3585) = (-3.64, -2.64)$$

$$f(\overline{q}) = \frac{(-3.64,-2.64)}{|4.4985|} = (-0.8091, -0.5868)$$
$$\Rightarrow q_1 = q_0 - \delta_0 \overline{f}(\overline{q}_0) = (1,1) + (-0.8091, -0.5868)$$
$$\Rightarrow q_1 = (1.8091, 1.5868)$$

Encontrar los siguientes tres puntos de posicionamiento del robot. Obtenga q_2, q_3 y q_4.

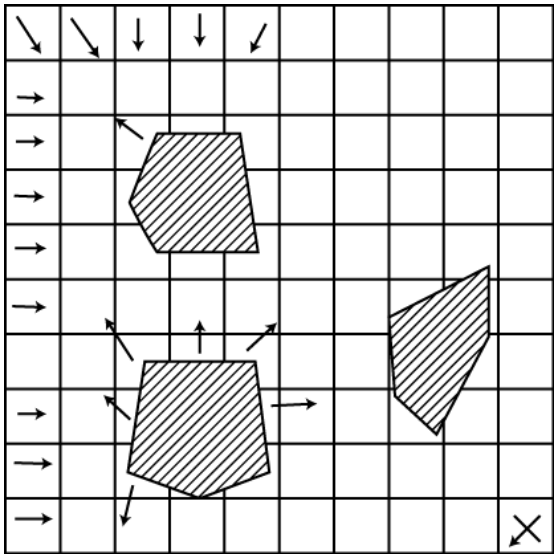
1.9 Campos Potenciales Usando Sensores de Proximidad



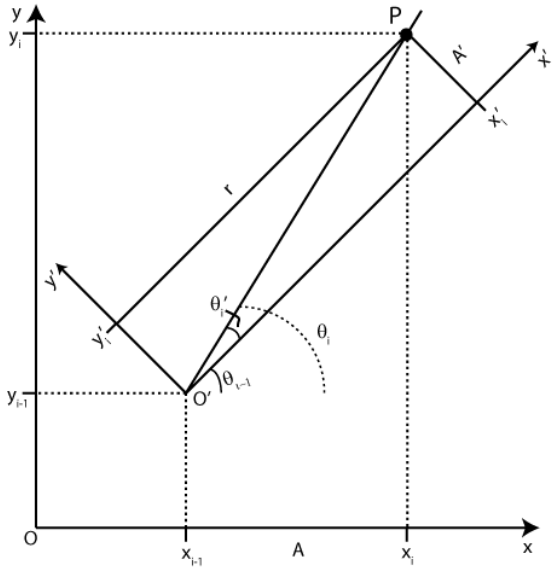
En el ángulo de determina hacia donde mira el robot respecto a su posición alfa es el ángulo del sensor respecto a la orientación del robot.

D es la distancia reportada por el sensor al obstáculo.

$$x_{obs} = d\cos(\theta + \alpha)$$
$$y_{obs} = d\,\mathrm{sen}(\theta + \alpha)$$
$$\overline{q} - \overline{q}_{obs} = (0,0) - (x_{obs}, y_{obs}) = (-x_{obs}, -y_{obs})$$



Cuando se conoce la posición de los objetos se puede almacenar en una matriz el campo de repulsión. Esto requiere de un procesador más potente.



Dados (x_{i-1}, y_{i-1}) para llegar al punto (x_i, y_i) se desea saber el ángulo que el robot debe girar para llegar a su destino, o bien el desplazamiento.

$$x_i = \overline{OA} = x_{i-1} + r\cos(\theta_{i-1} + \theta_i)$$

$$y_i = \overline{AP} = y_{i-1} + r\,\mathrm{sen}(\theta_{i-1} + \theta_i)$$

$$x_i = x_i' \cos(\theta_{i-1}) - y_i' \,\mathrm{sen}\,\theta_{i-1} + x_{i-1}$$

$$y_i = x_i' \,\mathrm{sen}(\theta_{i-1}) - y_i' \cos\theta_{i-1} + y_{i-1}$$

Esto para un sistema omnidireccional. En forma matricial:

$$\begin{bmatrix} x_i \\ y_i \\ \theta_i \end{bmatrix} = \begin{bmatrix} \cos\theta_{i-1} & \operatorname{sen}\theta_{i-1} & 0 \\ \operatorname{sen}\theta_{i-1} & \cos\theta_{i-1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_i' \\ y_i' \\ \theta_i' \end{bmatrix} + \begin{bmatrix} x_{i-1} \\ y_{i-1} \\ \theta_{i-1} \end{bmatrix}$$

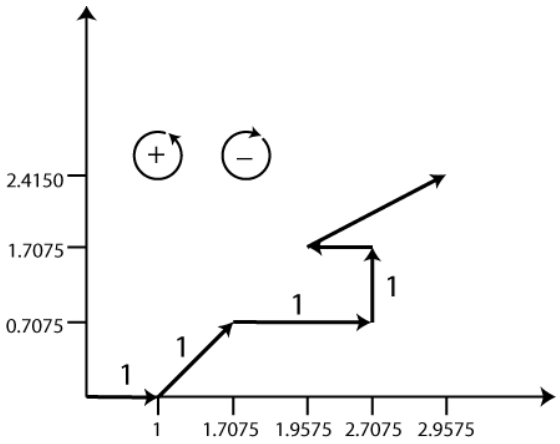
Despejando, robot omnidireccional

$$\begin{bmatrix} x_i' \\ y_i' \\ \theta_i' \end{bmatrix} = \begin{bmatrix} \cos\theta_{i-1} & \operatorname{sen}\theta_{i-1} & 0 \\ -\operatorname{sen}\theta_{i-1} & \cos\theta_{i-1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_i - x_{i-1} \\ y_i - y_{i-1} \\ \theta_i - \theta_{i-1} \end{bmatrix}$$

En el caso de un sistema robot no omnidireccional, las ecuaciones son las mismas, pero la matriz se despeja y queda como sigue:

$$\begin{bmatrix} x_i' \\ y_i' \\ \theta_i' \end{bmatrix} = \begin{bmatrix} \cos\theta_{i} & \operatorname{sen}\theta_{i} & 0 \\ -\operatorname{sen}\theta_{i} & \cos\theta_{i} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_i - x_{i-1} \\ y_i - y_{i-1} \\ \theta_i - \theta_{i-1} \end{bmatrix}$$

Ejemplo:



tiempo $i = 1$

$$x_0 = 0, y_0 = 0, \theta_0 = 0$$

$$x_1 = 1, y_1 = 0, \theta_1 = 0$$

$$\theta_i = \tan^{-1}\left(\frac{y_i - y_{i-1}}{x_i - x_{i-1}}\right)$$

Terminar ejercicio.

Caso 1. Robot no omnidireccional

$$x_1' = 1\cos 0 + 0\operatorname{sen} 0 = 1$$

$$y_1' = \text{sen}\,0 - 0\cos 0 = 0$$
$$\theta_1' = 0$$

$$i = 2$$
$$x_1 = 1, y_1 = 0, \theta_1 = 0$$
$$x_2 = 1.7075, y_2 = 0.7075, \theta_2 = 45^\circ$$

$$x_2' = 0.7075\cos(45^\circ) + 0.7075\,\text{sen}(45^\circ) = 1$$
$$y_2' = 0.7075\,\text{sen}(45^\circ) + 0.7075\cos(45^\circ) = 0$$
$$\theta_2' = 45^\circ$$

$$i = 3$$
$$x_2 = 1.7075, y_2 = 0.7075, \theta_2 = 45^\circ$$
$$x_3 = 2.7075, y_3 = 0.7075, \theta_3 = 0$$
$$x_3' = 1\cos 0 + 0\,\text{sen}\,0 = 1$$
$$y_3' = -1\,\text{sen}\,0 + 0\cos 0 = 0$$
$$\theta_3' = -45^\circ$$

Caso 2. Robot omnidireccional

tiempo $i = 1$

$$x_0 = 0, y_0 = 0, \theta_0 = 0$$
$$x_1 = 1, y_1 = 0, \theta_1 = 0$$
$$x_1' = 1\cos 0 + 0\,\text{sen}\,0 = 1$$
$$y_1' = -\,\text{sen}\,0 - 0\cos 0 = 0$$
$$\theta_1' = 0$$

$$i = 2$$
$$x_1 = 1, y_1 = 0, \theta_1 = 0$$
$$x_2 = 1.7075, y_2 = 0.7075, \theta_2 = 45^\circ$$
$$x_2' = 0.7075\cos 0 + 0.7075\,\text{sen}\,0 = 0.7075$$
$$y_2' = -0.7075\,\text{sen}\,0 + 0.7075\cos 0 = 0.7075$$
$$\theta_2' = -45^\circ$$

$$i = 3$$

$$x_2 = 1.7075, y_2 = 0.7075, \theta_2 = 45^\circ$$

$$x_3 = 2.7075, y_3 = 0.7075, \theta_3 = 0$$

$$x_3' = 1\cos(45^\circ) + 0\,\text{sen}(45^\circ) = 0.7075$$

$$y_3' = -1\,\text{sen}(45^\circ) + 0\cos(45^\circ) = -0.7075$$

$$\theta_3' = -45^\circ$$

1.10 Trayectorias

Posición inicial	Velocidades iniciales y finales
$f(0) = 0$	$f'(0) = 0$
$f(t_f) = x_i'$	$f'(t_f) = 0$

Posición: $f(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$

Mínimo de orden tres para controlar la aceleración.

Velocidad: $f'(t) = a_1 + 2a_2 t + 3a_3 t^2$

Aceleración: $f''(t) = 2a_2 + 6a_3 t$

$$f(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

$$f'(t) = a_1 + 2a_2 t + 3a_3 t^2$$

$$f''(t) = 2a_2 + 6a_3 t$$

Condiciones lineales para encontrar las constantes:

$$f(0) = 0 = a_0$$

$$x_i' = a_0 + a_1 t_f^2 + a_3 t_f^3$$

$$f'(0) = a_1$$

$$f'(t_i) = 0 = a_1 + 2a_2 t_f + 3a_3 t_f^2$$

$$\Rightarrow a_2 = \frac{3x_i}{t_f^2}$$

$$a_3 = \frac{-2x_i}{t_f^3}$$

4 ecuaciones con 4 incógnitas

$$f(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 = \frac{3x_i}{t_f^2} t^2 - \frac{2x_i}{t_f^3} t^3$$

$$\dot{f}(t) = a_1 + 2a_2 t + 3a_3 t^2 = \frac{6x_i}{t_f^2} t - \frac{6x_i}{t_f^3} t^2$$

$$\ddot{f}(t) = 2a_2 + 6a_3 t = \frac{6x_i}{t_f^2} t - \frac{12x_i}{t_f^3} t$$

Ejemplo:

$$t_f = 3seg, x_i = 1$$

$$f(t) = \frac{1}{3} t^2 - \frac{2}{27} t^3$$

$$\dot{f}(t) = \frac{2}{3} t - \frac{6}{27} t^2$$

$$\ddot{f} = \frac{2}{3} - \frac{12}{27} t$$

$$t_f = 1seg, x_i' = 2, y_i' = 5$$

t	$x'(t)$	$y'(t)$
0	0	0
0.1	0.056	0.14
0.2	0.208	0.52
0.5	1	2.5
0.8	1.792	4.48
1	2	5

$$f(t) = \frac{3(2)}{1^2} t^2 - \frac{2(2)}{1^2} t^3 \Big|_{t=0.1} = 0.056 \Leftarrow x$$

$$f(t) = \frac{3(5)}{1^2} t^2 - \frac{2(5)}{1^2} t^3 \Big|_{t=0.1} = 0.14 \Leftarrow y$$

1.11 Repaso Transformada de Laplace

$$L\{f(t)\} = \int_0^\infty f(t)\, e^{-st}\, dt = F(s)$$

$$L\{e^{-\alpha t}\} = \int_0^\infty e^{-\alpha t}\, e^{-st}\, dt = \int_0^\infty e^{-(s+\alpha)t}\, dt = \frac{e^{(s+\alpha)}}{s+\alpha} \Big|_0^\infty = \frac{1}{s+\alpha}$$

Función escalón

$$L\{u(t)\} = \int_0^\infty e^{-st}\,dt = -\frac{e^{st}}{s}\bigg|_0^\infty = \frac{1}{s}$$

$$L\{\cos(\omega t)\} = \frac{s}{s^2+\omega^2}$$

$$L\{tu(t)\} = L\{u_{-1}(t)\} = \frac{1}{s^2}$$

$$L\left\{\frac{df(t)}{dt}\right\} = sF(s) - \dot{F}(t)$$

$$L\left\{\int_0^\infty f(t)\,dt\right\} = \frac{1}{s}F(s)$$

1.12 Transformada Inversa

$$f(t) = L^{-1}\{F(s)\} = \int_0^\infty F(s)\,e^{st}\,ds$$

Sea la ecuación diferencial

$$\frac{d^n y(t)}{dt^n} + b_{n-1}\frac{d^{n-1}y(t)}{dt^{n-1}} + \ldots + b_1\frac{dy(t)}{dt} + b_0 y(t)$$

$$= a_m\frac{d^m x(t)}{dt^m} + a_{m-1}\frac{d^{m-1}x(t)}{dt^{m-1}} + \ldots + a_1\frac{dx(t)}{dt} + a_0 x(t)$$

Usando la transformada de *Laplace* para resolver la ecuación diferencial (condiciones iniciales nulas).

$$s^nY(s) + b_{n-1}s^{n-1}Y(s) + \ldots + b_1sY(s) + b_0Y(s) =$$
$$a_ms^mX(s) + a_{m-1}s^{m-1}X(s) + \ldots + a_1sX(s) + a_0X(s)$$

$$\Rightarrow (s^n + b_{n-1}s^{n-1} + \ldots + b_1s + b_0)Y(s) = (a_ms^m + a_{m-1}s^{m-1} + \ldots + a_1s + a_0)X(s)$$

$$\Rightarrow \frac{Y(s)}{X(s)} = \frac{a_ms^m + a_{m-1}s^{m-1} + \ldots + a_1s + a_0}{s^n + b_{n-1}s^{n-1} + \ldots + b_1s + b_0}$$

Se resuelve por fracciones parciales

1.13 Fracciones Parciales

$$F(s) = \frac{P(s)}{Q(s)} = \frac{P(s)}{s(s-s_1)(s-s_2)\cdots(s-s_n)}$$
$$= \frac{A_0}{s} + \frac{A_1}{s-s_1} + \frac{A_2}{s-s_2} + \ldots + \frac{A_n}{s-s_n}$$

$$\Rightarrow f(t) = A_0 + A_1 e^{s_1 t} + A_2 e^{s_2 t} + \ldots + A_n e^{s_n t}$$

$$(s-s_k)F(s) = \frac{(ss_k)P(s)}{Q(s)}$$

$$= \frac{(s-s_k)}{s}A_0 + \frac{(s-s_k)}{s-s_1}A_1 + \ldots + \frac{(s-S_k)}{s-s_k}A_k + \ldots + \frac{(s-s_k)}{s-s_n}A_n$$

$$= \frac{(s-s_k)}{s}A_0 + \frac{(s-s_k)}{s-s_1}A_1 + \ldots + A_k + \ldots + \frac{(s-s_k)}{s-s_n}A_n$$

Si $s = s_k$

$$A_k = \left[(s-s_k)\frac{P(s)}{Q(S)}\right]_{s=s_k}$$

Ejemplo

$$F(s) = \frac{s+2}{s(s+1)(s+2)}$$

$$A_0 = \left[\frac{s(s+2)}{s(s+1)(s+3)}\right]_{s=0} = \frac{2}{3}$$

$$A_1 = \left[\frac{(s+1)(s+2)}{s(s+1)(s+3)}\right]_{s=-1} = -\frac{1}{2}$$

$$A_2 = \left[\frac{(s+3)(s+2)}{s(s+1)(s+3)}\right]_{s=-3} = -\frac{1}{6}$$

$$F(s) = \frac{\frac{2}{3}}{s} - \frac{\frac{1}{2}}{s+1} - \frac{\frac{1}{6}}{s+3}$$

$$\Rightarrow f(t) = \frac{2}{3} - \frac{1}{3}e^{-t} - \frac{1}{6}e^{-3t}$$

1.14 Polos Rales Múltiples

$$F(s) = \frac{P(s)}{(s-s_1)^2(s-s_2)}$$

$$= \frac{A_{13}}{(s-s_1)^3} + \frac{A_{12}}{(s-s_1)^2} + \frac{A_{11}}{s-s_1} + \frac{A_2}{s-s_2}$$

Transformada inversa

$$f(t) = A_{13}\frac{t^2}{2}e^{s_1 t} + A_{12}\,t\,e^{s_1 t} + A_{11}\,e^{s_1 t} + A_2\,e^{s_2 t}$$

Ejemplo

$$F(s) = \frac{1}{(s+2)^3(s+3)} = \frac{A_{13}}{(s+2)^3} + \frac{A_{12}}{(s+2)^2} + \frac{A_{11}}{s+2} + \frac{A_2}{s+3}$$

$$A_{13}\left[(s+3)^3F(S)\right]_{s=-2} = \left[A_{13} + A_{12}(s+2)^2 + \frac{A_2(s+2)^3}{s+3}\right]_{s=-2}$$

Valuando con $A_{13} = 1$

$$A_{12} = \left[(s+2)^2F(S)\right]_{s=-2} = \left[\frac{A_{13}}{s+2} + A_{12} + A_{11}(s+2) = \frac{A_2(s+2)^2}{s+3}\right]_{s=-2}$$

Existe una indeterminación en el primer término. Sin embargo, se puede demostrar que

$$A_{12} = \left[\frac{d}{ds}\left((s+2)^3F(s)\right)\right]_{s=-2} = \frac{d}{ds}\left(\frac{1}{s+3}\right)$$

$$= \frac{d}{ds}\left(A_{13} + A_{12}(s+2) + A_{11}(s+2)^2 + \frac{A_2(s+2)^3}{s+3}\right)\Bigg|_{s=-2}$$

$$\Rightarrow \frac{d}{ds}\left(\frac{1}{s+3}\right) = A_{12}$$

$$\Rightarrow A_{12} = \frac{-1}{(s+3)^2}\Bigg]_{s=-2} = -1$$

Luego, para A_{11}

$$A_{11}\frac{1}{2}\frac{d^2}{ds^2}\left[(s+2)^2F(s)\right]_{s=-2} = \frac{1}{2}\frac{2}{(s+3)^3}\Bigg|_{s=-2} = 1$$

$$\therefore\ A_q(r-k) = \left\{ \frac{1}{k!}\frac{d^k}{ds^k}\left[(s-s_q)^r \frac{P(s)}{Q(s)}\right]\right\}_{s=s_q}$$

$$\Rightarrow F(s) = \frac{1}{(s+2)^3} - \frac{1}{(s+2)^2} + \frac{1}{s+2} - \frac{1}{s+3}$$

$$\therefore\ f(t) = \frac{1}{2}t^2e^{-2t} - te^{-2t} + e^{-2t} - e^{-3t}$$

1.15 Polos Complejos Conjugados

$$F(s) = \frac{P(s)}{(s^2+2\zeta\omega_n s+\omega_n^2)(s-s_3)}$$

$$= \frac{A_1}{s+\zeta\omega_n - j\omega_n\sqrt{1-\zeta^2}} + \frac{A_2}{s+\zeta\omega_n - j\omega_n\sqrt{1-\zeta^2}} + \frac{A_3}{s+s_3}$$

Como $As^2+Bs+C = s^2+2\zeta\omega_n s+\omega_n^2,\ s_{1,2} = \dfrac{-b\pm\sqrt{b^2-4ac}}{2a}$.

La transformada inversa de $F(s)$ es

$$f(t) = A_1e^{-\left(\zeta\omega_n+j\omega_n\sqrt{1-\zeta^2}\right)t} + A_2e^{-\left(\zeta\omega_n-j\omega_n\sqrt{1-\zeta^2}\right)t} + A_3e^{s_3t}$$

Eliminando los coeficientes complejos

$$f(t) = 2|A_1|e^{-\zeta\omega_n t}\,\text{sen}\left(\omega_n\sqrt{1-\zeta^2}t+\phi\right) + A_3e^{s_3t}$$

$$\phi = \text{Ángulo de } A_1 + 90^\circ$$

$$A_1 = [(s-s_1)F(s)]_{s=s_1} = a_{1Re} + a_{1Im}$$

$$|A_1| = \sqrt{a_{1Re}^2 + a_{1Im}^2}$$

$$deg(A_1) = \tan^{-1}\left(\frac{a_{1Re}}{a_{1Im}}\right)$$

Ejemplo

$$F(s) = \frac{10}{(s^2+6s+25)(s+2)} = \frac{A_1}{s+3-j4} + \frac{A_2}{s+3+j4} + \frac{A_3}{s+2}$$

$$A_1 = [(s+3-j4)F(s)]_{s=3+j4} = \left.\frac{10}{(s+3+j4)(s+2)}\right|_{s=3+j4}$$

$$= \frac{10}{(j8)(-1+j4)} = \frac{10}{-32-j8}$$

Multiplicando por el conjugado

$$\frac{10(-32+j8)}{(-32-j8)(-32+j8)} = \frac{-10}{8(17)}(4-j)$$

$$|A_1| = \frac{10}{8(17)}\sqrt{16+1} = 0.303$$

$$deg(A_1) = 194^\circ \Rightarrow \phi = -104^\circ$$

$$A_3 = [(s+2)F(s)]_{s=-2} = 0.59$$

$$\Rightarrow f(t) = 0.6e^{-3t}\,\text{sen}(4t-104^\circ) + 0.59e^{-2t}$$

1.16 Motores de Corriente Directa

$$v(t) = Ri(t) + L\frac{di(t)}{dt} + \varepsilon(t)$$

$$\varepsilon(t) = \text{Voltaje de fuerza electromotriz } = k_m\omega(t)$$

Por *Laplace*

$$V(s) = (R+Ls)I(s) + \varepsilon(s)$$

Pero

$$\tau_a(s) = k_m I(s)$$

$$\tau_l(s) = Js\omega(s) + f\omega(s)$$

$\tau_l \sqcup$ Torque de aplicación

$J \sqcap$ Momento de incercia

$f \sqsubset$ Coeficiente de fricción

$$\omega(s) = \frac{\tau_a}{Js+f}$$

$$\Rightarrow \omega(s) = \frac{k_m I(s)}{Js+f} = \frac{k_m \dfrac{V(s)-\varepsilon(s)}{R+Ls}}{Js+f}$$

$$\omega(s) = \frac{k_m V(s) - k_m^2 \omega(s)}{(Js+f)(R+Ls)}$$

$$\Rightarrow \omega(s) = \frac{k_m V(s)}{(Js+f)(R+Ls)+k_m^2}$$

$$\Rightarrow \frac{\omega(s)}{V(s)} = \frac{k_m}{(Js+f)(R+Ls)+k_m^2} = \frac{s\theta(s)}{V(s)}$$

$$\Rightarrow \frac{\theta(s)}{V(s)} = \frac{k_m}{s(k_m^2+(Js+f)(R+Ls))}$$

Se puede hacer la siguiente simplificación

$$\frac{\theta(s)}{V(s)} = \frac{k_0}{s(s+\alpha)}$$

$$F(V(t)) = \int_0^{t_1} V_p e^{-st}\, dt = -\frac{V_p e^{-st}}{s}\bigg|_0^{t_1} = V_p\left(\frac{1-e^{-st_1}}{s}\right)$$

$$\theta(s) = V(s)H(s) = \left(\frac{k_0}{s(s+\alpha)}\right)\left(\frac{V_p(1-e^{-st_1})}{s}\right)$$

$$\theta(s) = \frac{k_0 V_p(1-e^{-st_1})}{s^2(s+\alpha)}$$

Por fracciones parciales

$$\Rightarrow \frac{A_{12}}{s^2} + \frac{A_{11}}{s} + \frac{A_2}{s+\alpha}$$

$$\theta(t) = A_{12}te^{-t} + A_{11}e^{-t} + A_2 e^{-\alpha t}$$

Encontrar A_{11}, A_{12} y A_2

Implementar $A_{12}te^{-t} + A_{11}e^{-t} + A_2 e^{-\alpha t}$ requiere de amplificadores operacionales y condensadores, en configuración integrador y/o derivador; luego entonces es cero y cinviene hacerlo digital.

2^3	2^2	2	2^0	$\cdot$	2^{-1}	2^{-2}	2^{-3}	2^{-4}
0	0	1	1	$\cdot$	1	1	0	0

1.17 Transformada Z

$$Z[x[n]] = \sum_{n=-\infty}^{\infty} x[n]z^{-n} = X(z)$$

Representar 3.76 a 8 bits, 4 enteros y 4 decimales

Z es un número complejo limitado a un número de convergencia.

$$\Rightarrow Z\{x[n-1]\} = \sum_{-\infty}^{\infty} x[n-1]z^{-n}$$

$$k = n-1$$

$$n = -\infty \; k = -\infty$$

$$n = \infty \; k = \infty$$

$$n = k+1$$

$$Z\{x[n-1]\} = \sum_{-\infty}^{\infty} x[k]z^{-(k+1)} = z^{-1}\sum_{-\infty}^{\infty} x[k]z^{-k} = z^{-1}X(z)$$

De manera similar

$$Z\{x[n-1]\} = zX(z)$$

1.18 Ecuación en Diferencias

$$a_n y[n-N] + a_{n-1}y[n-(N-1)] + \ldots + a_0 y[n] =$$
$$b_m x[n-N] + b_{m-1}x[n-(N-1)] + \ldots + b_0 x[n]$$

Análogo a la ecuación

$$c_n \frac{d^n y(t)}{dt^n} + \ldots + c_0 y(t) =$$
$$d_m \frac{d^m x(t)}{dt^m} + \ldots + d_0 y(t)$$

Se requiere una función que transforme de *Laplace* a Z ($s \to z$) conservando los polos estables.

Esta función es la función Bilineal,

$$s = \frac{2}{T}\left(\frac{z-1}{z+1}\right)$$

$$H(z) = H(s)|_{s=\frac{2}{T}\left(\frac{z-1}{z+1}\right)}$$

$$\text{Si } H(s) = \frac{k_0}{s(s+\alpha)}$$

$$\Rightarrow H(z) = \left.\frac{k_0}{s(s+\alpha)}\right|_{s=\frac{2}{T}\left(\frac{z-1}{z+1}\right)}$$

$$H(z) = \frac{k_0}{\left(\frac{2(z-1)}{T(z+1)}\right)\left(\frac{2(z-1)}{T(z+1)}+\alpha\right)}$$

$$H(z) = \frac{k_0}{\frac{4(z-1)^2}{T^2(z+1)^2}+\frac{2(z-1)}{T(z+1)}+\alpha}$$

$$= \frac{k_0T^2(z^2+2z+1)}{z^2(2T\alpha+4)-8z+4-2T\alpha}$$

Multiplicando por $\frac{z^{-2}}{z^{-2}}$

$$= \frac{k_0T^2(1+2z^{-1}+z^{-2})}{(2T\alpha+4)-8z^{-1}+(4-2T\alpha)z^{-2}} = \frac{\theta(z)}{V(z)}$$

$$V(z)\left[k_0T^2(1+2z^{-1}+z^{-2})\right] = \theta(z)\left[(2T\alpha+4)-8z^{-1}+(4-2T\alpha)z^{-2}\right]$$

$$Z^{-1}\{V(z)\} = v(n)$$

Antitransformando

$$K_0T^2v[n]+2k_0T^2v[n-1]+k_0T^2v[n-2] = \theta[n](2T\alpha+4)-8\theta[n-1]+(4-2T\alpha)\theta[n-2]$$

1.19 Repaso Matemática Discreta

El impulso

$$x_k = \ldots + x_{-2}\delta_{k+2} + x_{-1}\delta_{k+1} + x_0\delta_k + x_1\delta_{k-1} + x_2\delta_{k-2} \Rightarrow x_k = \sum_{i=-\infty}^{\infty} x_i\delta_{k-i}$$

$x_k \rightarrow$ Función.

$x_i \rightarrow$ Constante.

En el mundo continuo

Pero en el discreto:

Se espera que

En general una espuesta lineal

Como

$$T\{\delta_{k-1}\} = h_k$$

En el continuo:

$$a_n\frac{d^n y(t)}{dt^n} + a_{n-1}\frac{d^{n-1}y(t)}{dt^{n-1}} + \ldots + a_0 y(t) =$$

$$b_0 x(t) + b_1\frac{dx(t)}{dt} + \ldots + b_m\frac{d^m x(t)}{dt^m}$$

$$\Rightarrow Y(s)(a_n s^n + a_{n-1}s^{n-1} + \ldots + a_0) =$$
$$X(s)(b_0 + b_1 s + \ldots + b_m s^m)$$

$$\Rightarrow Y(s) = \frac{X(s)(b_0 + b_1 s + \ldots + b_m s^m)}{a_n s^n + a_{n-1}s^{n-1} + \ldots + a_0}$$

$$= \frac{A_0}{s+s_0} + \frac{A_1}{s+s_1} + \ldots + \frac{A_n}{s+s_n}$$

$y(t) = \ldots$

En el discreto:

$$x_k = 2^k U_k$$

$$U_k = \begin{cases} 1 & \text{si} \quad k \geq 0 \\ \\ 0 & \text{si} \quad k < 0 \end{cases}$$

$$x(z) = \sum_{k=-\infty}^{\infty} x_k z^{-k} = \sum_{k=0}^{\infty} 2^k z^{-k} = \sum_{k=0}^{\infty} (2^{-1}z)^{-k}$$

Multimplicando ambos lados por $(2^{-1}z)^{-1}$

$$(2^{-1}z)^{-1}X(z) = \sum_{k=0}^{\infty} (2^{-1}z)^{-1}(2^{-1}z)^{-k} = \sum_{k=0}^{\infty} (2^{-1}z)^{-(k+1)}$$

si $i = k+1 \Rightarrow k = 0; i = 1, k = \infty; i = \infty$.

$$(2^{-1}z)^{-1}X(z) = \sum_{k=1}^{\infty} (2^{-1}z)^{-i}$$

Como i es variable muda, se puede

$$(2^{-1}z)^{-1}X(z) = \sum_{k=1}^{\infty} (2^{-1}z)^{-k}$$

Restando a la ecuación original

$$X(z) - (2^{-1}z)^{-1}X(z) = \sum_{k=0}^{\infty} (2^{-1}z)^{-k} - \sum_{k=1}^{\infty} (2^{-1}z)^{-k}$$

$$X(z) - (2^{-1}z)^{-1}X(z) = 1$$

Despejando

$$X(z) = \frac{1}{1-2z^{-1}} \leftrightarrow 2^k$$

Volviendo con el robot

$$\frac{\theta(s)}{V(s)} = \frac{k_0}{s(s+\alpha)} = H(s)$$

$$s = \frac{2}{T}\left(\frac{z-1}{z+1}\right)$$

$$H(z) = H(s)\Big|_{s=\frac{2}{T}\left(\frac{z-1}{z+1}\right)}$$

$$\frac{\theta(s)}{V(z)} = \frac{k_0 T^2(1+2z^{-1}+z^{-2})}{(sT\alpha+4)-8z^{-1}+(4-2T\alpha)z^{-2}}$$

$$\theta(z)\left((2T\alpha+4)-8z^{-1}+(4-2T\alpha)z^{-2}\right) = V(z)\left(k_0 T^2(1+2z^{-1}+z^{-2})\right)$$

Antitransformando

$$(2T\alpha+4)\theta_n - 8\theta_{n-1} + (4-2T\alpha)\theta_{n-2} = k_0 T^2(V_n + 2V_{n-1} + V_{n-2})$$

Despejando θ_n

$$\theta_n = \frac{k_0 T^2(V_n + V_{n-1} + V_{n-2}) + 8\theta_{n-1} - \theta_{n-2}(4-sT\alpha)}{2T\alpha+4}$$

1.20 Movimiento en las llantas

$J_w =$ Momento de inercia de la llanta.

$\ddot{\theta}_w =$ Aceleración rotacional de la llanta.

$m_w =$ Masa de la llanta.

$F_R =$ Fuerza de fricción $= B_w\dot{\theta}_w$.

$B_w =$ Constante de fricción.

$R_w =$ Radio de la llanta.

Torque de la llanta:

$\tau_w = J_w\ddot{\theta}_w + B_w\dot{\theta}_w$

$\tau_w = F_w R_w$

$F_w = \dfrac{\tau_w}{R_w} = \dfrac{1}{R_w}\left(J_w\ddot{\theta}_w + B_w\dot{\theta}_w\right)$

1.21 Movimiento Lineal de Robot

$\ddot{x}_R =$ Aceleración lineal del robot.

$m_R =$ Masa del robot.

$F = m_R\ddot{x}_R$.

Se tiene dos llantas

$$F = F_{wI} + F_{wD} = m_R \ddot{x}_R = m_R \dot{v}_R$$

$\Rightarrow$

$$\frac{1}{R_{wI}}\left(J_{wI}\ddot{\theta}_{wI} + B_{wI}\dot{\theta}_{wI}\right) + \frac{1}{R_{wD}}\left(J_{wD}\ddot{\theta}_{wD} + B_{wD}\dot{\theta}_{wD}\right) ???????$$

$\Rightarrow$

$$V_R(s) = \omega_{wI}(s)\left(\frac{J_{wI}s + b_{wI}}{m_R R_{wI} s}\right) + \omega_{wD}(s)\left(\frac{J_{wD}s + b_{wD}}{m_R R_{wD} s}\right)$$

$$V(s) = \begin{bmatrix} \frac{J_{wI}s + b_{wI}}{m_R R_{wI} s} & \frac{J_{wD}s + b_{wD}}{m_R R_{wD} s} \end{bmatrix} \begin{bmatrix} \omega_{wI}(s) \\ \omega_{wD}(s) \end{bmatrix}$$

1.22 Movimiento Rotacional del Robot

$J_R =$ Momento de inercia del robot.

$R_R =$ Radio del robot.

$\ddot{\theta}_R =$ Aceleración rotacional del robot.

$\tau_R =$ Torque del robot.

$B_R =$ Constante de fricción.

$$\tau_R = J_R\ddot{\theta}_R + B_R\dot{\theta}_R = (F_{wI} - F_{wD})R_R = \left(\frac{J_R\ddot{\theta}_R + B_R\dot{\theta}_R}{R_{wI}}\right)R_R - \left(\frac{J_R\ddot{\theta}_R + B_R\dot{\theta}_R}{R_{wD}}\right)R_R$$

Transformando

$$\tau_R(s) = \omega_R(s)(sJ_R + R_R)$$

$$\omega_R(s) = \frac{\omega_{wI}(s)R_R}{R_{wI}}\left(\frac{J_{wI}s + \omega_{wI}}{J_R s + B_R}\right) - \frac{\omega_{wD}(s)R_R}{R_{wD}}\left(\frac{J_{wD}s + \omega_{wD}}{J_R s + B_R}\right)$$

$$\omega_R(s) = \begin{bmatrix} \frac{R_R}{R_{wI}}\left(\frac{J_{wI}s + \omega_{wI}}{J_R s + B_R}\right) \\ -\frac{R_R}{R_{wD}}\left(\frac{J_{wD}s + \omega_{wD}}{J_R s + B_R}\right) \end{bmatrix}^T \begin{bmatrix} \omega_{wI}(s) \\ \omega_{wD}(s) \end{bmatrix}$$

1.23 Movimiento del Robot

Juntando el moviento lineal con el rotacional

$$\begin{bmatrix} V_R(s) \\ \omega_R(s) \end{bmatrix} = \begin{bmatrix} \frac{J_{wI}s + b_{wI}}{m_R R_{wI} s} & \frac{R_R}{R_{wI}}\left(\frac{J_{wI}s + \omega_{wI}}{J_R s + B_R}\right) \\ \frac{J_{wD}s + b_{wD}}{m_R R_{wD} s} & -\frac{R_R}{R_{wD}}\left(\frac{J_{wD}s + \omega_{wD}}{J_R s + B_R}\right) \end{bmatrix}^T \begin{bmatrix} \omega_{wI}(s) \\ \omega_{wD}(s) \end{bmatrix}$$

Por medio de la matriz inversa se despeja $\begin{bmatrix} \omega_{wI}(s) \\ \omega_{wD}(s) \end{bmatrix}$

$$\begin{bmatrix} \omega_{wI}(s) \\ \omega_{wD}(s) \end{bmatrix} = A^{-1} \begin{bmatrix} V_R(s) \\ \omega_R(s) \end{bmatrix}$$

El modelo de motor despreciando la fricción

$$\begin{bmatrix} \omega_{wI}(s) \\ \omega_{wD}(s) \end{bmatrix} = \begin{bmatrix} \frac{k_{01} V_1(s)}{s+\alpha_1} \\ \frac{k_{0D} V_D(s)}{s+\alpha_D} \end{bmatrix}$$

Voltaje que se tiene que aplica a los motores

$$\begin{bmatrix} V_1(s) \\ V_D(s) \end{bmatrix} = \begin{bmatrix} \frac{\omega_1(s)(s+\alpha_1)}{k_{01}} \\ \frac{\omega_D(s)(s+\alpha_D)}{k_{0D}} \end{bmatrix}$$

1.24 Cinemática del Robot

Número de tics del encoder para una vuelta complera de la llanta.

$$S_L = 2\pi r \frac{ticksL}{ticks_per_rev}$$

Análogo para S_R

$$S = \frac{(S_L + S_R)}{2}$$

$$S_L = \phi(c + \frac{d}{2})$$
$$S_R = \phi(c - \frac{d}{2})$$

$$S_L - S_D = \phi d \quad \Rightarrow \quad \phi = \frac{S_L - S_R}{d}$$

$$V_R = 2\pi r \dot{\theta}_R = 2\pi r \omega_R$$

$$V_L = 2\pi r \dot{\theta}_L = 2\pi r \omega_L$$

Velocidad angular

$$\omega = \frac{2\pi r}{d}(\omega_L - \omega_D)$$

$$v = v_R + v_L$$

$$\begin{bmatrix} V \\ \omega \end{bmatrix} = 2\pi r \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{-1}{d} & \frac{d}{2} \end{bmatrix} \begin{bmatrix} V \\ \omega \end{bmatrix}$$

$$\begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix} = \frac{1}{2\pi r} \begin{bmatrix} 1 & \frac{-d}{2} \\ 1 & \frac{d}{2} \end{bmatrix} \begin{bmatrix} V \\ \omega \end{bmatrix}$$

1.25 Control ON - Off

El motor se prende o se apaga dependiendo de si la velocidad del motor esta abajo o arriba de la velocidad deseada.

$R(t) =$ Función que controla el motor

$V_{act}(t) =$ Velocidad del moto

$V_{des}(t) =$ Velocidad deseada

$k_c =$ Valor de la constante del control

$$R(t) = \begin{cases} k_c & si \quad V_{act}(t) < V_{des}(t) \\ 0 & si \quad V_{act}(t) \geq V_{des}(t) \end{cases}$$

1.26 Control Proporcional

$$R(t) = k_p(v_{des}(t) - v_{act}(t))$$

1.27 Control Integral Proporcional

La idea del controlador I es reducir el error del estado estable del controlador P.

$$R(t) = k_p[e(t) + \frac{1}{T_1}\int_0^t e(t)\,dt]$$

Discretizando: regla trapezoidal

$$R_n = k_p e_n + \frac{k_p}{T_1} t_\Delta \sum_{i=1}^{n} \frac{e_i - e_{i-1}}{2}$$

Si se calcula

$R_n - R_{n-1}$

$$R_n - R_{n-1} = k_p(e_n - e_{n-1}) + \frac{k_p}{T_I} t_\Delta (\frac{e_n - e_{n-1}}{2})$$

$$k_I = \frac{k_p}{t_I} t_\Delta \quad \Rightarrow \quad R_n = R_{n-1} + k_p(e_n - e_{n-1}) + k_1 (\frac{e_n - e_{n-1}}{2}$$

Reduce el error en estado estable.

1.28 Control Derivativo Proporcional

$$R(t) = k_p \left[(t) + T_D \frac{d}{dt} e(t) \right]$$

CONTROL PID

$$R(t) = k_p \left[e(t) + \frac{1}{T_1} \int_0^1 e(t)\, dt + T_D \frac{d}{dt} e(t) \right]$$

En el discreto

$$R_n = k_p e_n + \frac{k_p}{T_I} t_\Delta \sum_{i=1}^{n} \frac{e_i - e_{i-1}}{2} + \frac{1}{t_{Delta}} k_p T_D (e_n - e_{n-1})$$

$$k_1 = \frac{k_p}{T_1} t_\Delta; \quad k_D = \frac{1}{t_\Delta} k_p T_\Delta$$

$$R_n - R_{n-1} = k_p(e_n - e_{n-1}) + k_I \frac{(e_n - e_{n-1})}{2} + k_D (e_n - 2e_{n-1} + e_{n-2})$$

$$R_n = R_{n-1} + k_p(e_n - e_{n-1}) + k_I \frac{(e_n - e_{n-1})}{2} + k_D (e_n - 2e_{n-1} + e_{n-2})$$

1.29 Red Neuronal

$$y_i = f\left(\sum_{i=0}^{n-1} x_i w_i\right)$$

Cada salida esta conectada a una entrada de las demás neuronas (formando una red). El aprendizaje se manifiesta mediante la variación de los pesos de la matriz ω_k.

El sistema debe entrenarse por un humano experto.

2 Robots Móviles y Agentes Inteligentes

2.1 Introducción

La palabra *robot* tiene su origen en la obra *Rossum's Universal Robots* del escritor checo *Karel Čapeck*, publicada en 1921, y su significado es "trabajo duro". Existen varias definiciones de robot y en general varían dependiendo del campo de aplicación o del área de investigación.

De acuerdo con la Real Academia Española, un robot es una máquina o ingenio electrónico programable capaz de manipular objetos y realizar operaciones antes reservadas sólo a las personas. Esta definición incluye conceptos como "proglamable" y "manipulación de objetos", sin embargo, no es la más adecuada para el tipo de robots que se desarrollarán en este curso.

En las secciones subsecuentes se tocarán temas últiles para brindar autonomía a un robot móvil, pero, ¿qué es un robot autónomo?. De acuerdo con [**?**], un robot autónomo es aquel que sólo necesita descripciones de alto nivel para llevar a cabo una determinada tarea y que no requiere de intervención humana para tal fin, es decir, las descripciones de entrada deberán especificar más el qué que el cómo. En este curso se abordará sólo el tema de la navegación autónoma.

2.1.1 Primitivas de la robótica

Las tareas que puede llevar a cabo un robot pueden clasificarse en tres grandes conjuntos conocidos como primitivas de la robótica: sensar, planear y actuar.

Sensar: Se refiere a la extracción de información ya sea del ambiente externo o interno del robot.

Planear: Es la generación de subtareas y toma de decisiones a partir de la información obtenida de los sensores y/o de alguna representación del conocimiento generada anteriormente. La planeación genera *directivas*.

Actuar: Es la modificación del ambiente por alguno de los dispositivos del robot. Los comandos enviados a los *actuadores* del robot se generan a partir de la información sensada o de las directivas generadas por la planeación.

Como se verá más adelante, la forma en que se relacionan las diferentes primitivas define los diferentes *paradigmas de la robótica*.

2.1.2 Componentes básicos de un robot móvil

Cada una de las primitivas tiene un correlato en hardware: sensores, microcomputadoras y actuadores.

Sensores: Son los transductores utilizados para extraer información del ambiente. Se pueden clasificar en propioceptivos o exteroceptivos dependiendo de si extraen información del ambiente interno o externo del robot respectivamente. Los encoders para los motores o los medidores de nivel de bateria son ejemplos de sensores propioceptivos mientras que las cámaras y micrófonos son ejemplos de sensores exteroceptivos.

Otra forma de clasificar los sensores es en activos y pasivos. Los primeros son aquellos que necesitan emitir energía para realizar el sensado, contrario a los segundos, que no requieren de dicha emisión. Ejemplos de sensores pasivos son las cámaras RGB, micrófonos o sensores de contacto. Una cámara infrarroja o un sensor láser son ejemplos de sensores activos.

Microcomputadoras: En esta categoría entran los CPUs, GPUs, microprocesadores, microcontroladores, DSPs, etc. La elección de alguno de estos está en función del tipo de procesamiento que se desee realizar.

Actuadores: Son los dispositivos que sirven para realizar alguna modificación al ambiente. En el caso de los robots móviles autónomos, los más utilizados son los eléctricos.

2.1.3 Paradigmas de la robótica

Los paradigmas de la robótica se definen en función de la forma en que se relacionan las primitivas y existen tres: el jerárquico o tradicional, el reactivo y el híbrido.

Jerárquico: En este paradigma las tres primitivas se realizan en forma secuencial, como se muestra en la figura **??**. Tiene las siguientes características:

- Fuerte dependencia de una representación interna del ambiente.
- El tiempo de respuesta es lento comparado con el paradigma reactivo.
- El costo computacional es alto.
- Con este paradigma se pueden resolver tareas con algo nivel cognitivo.
- Dada la dependencia de una representación del ambiente, la capacidad de predicción es alta.

Reactivo: En este paradigma el sensado y la actuación se conectan directamente sin que haya de por medio una planeación, como se muestra en la figura **??**. Sus características en general son contrarias a las del paradigma jerárquico:

- No requiere de una representación del ambiente.
- El tiempo de respuesta es rápido comparado con el paradigma jerárquico.

- Bajo costo computacional.
- En general no es posible resolver tareas que requieran de un alto nivel cognitivo.
- La capacidad de predicción es baja.

Híbrido: Tiene como objetivo utilizar las ventajas de ambos paradigmas, es decir, emplear comportamientos reactivos para que el robot responda rápidamente ante cambios en el ambiente sin perder la alta capacidad cognitiva y de predicción que brinda el paradigma jerárquico. La figura **??** muestra la forma en que se relacionan las primitivas en este paradigma.

2.2 El problema de la planeación de movimientos

2.2.1 Tareas en la planeación de movimientos

El problema de la planeación de movimientos comprende cuatro tareas principalmente: navegación, cobertura, localización y mapeo. Para estas cuatro tareas es necesario tener una descripción o representación de los puntos en el espacio que ocupa el robot. A esta descripción se le llama *configuración* y el *espacio de configuraciones* es el conjunto de todas las configuraciones posibles que puede tener el robot.

La navegación es el problema de encontrar un movimiento libre de colisiones desde una configuración inicial a una final. La cobertura se refiere al problema de mover un sensor o una herramienta de modo que se asegure que se cubren todos los puntos de un espacio determinado. La localización consiste en determinar la configuración del robot dado un mapa y un conjunto de lecturas de los sensores. El mapeo consiste en la exploración y sensado de un ambiente desconocido de modo que se obtenga una representación de dicho ambiente que sea últil para alguna de las otras tareas. Al problema de la localización y mapeo simultáneos se le conoce como SLAM (por sus siglas en inglés).

En este curso se abordarán únicamente las tareas de navegación y localización.

2.2.2 Características del robot

Para poder construir un planeador de movimientos es necesario conocer primero las características del robot a utilizar. La primera de ellas es el número de *grados de libertad* que se refiere al número de parámetros independientes que definen la configuración del robot. En el caso de un robot móvil que sólo se mueve en el plano se tienen tres grados de libertad: dos coordenadas de posición (x, y) y una orientación θ. Los espacios de configuración pueden describirse empleando variedades y los grados de libertad definen la *forma* de dicha variedad.

Otra característica importante son las restricciones de movimiento. Si el robot se puede mover en cualquier dirección en el espacio de configuración (en ausencia de obstáculos) se habla de un robot omnidirecciones. Si existen restricciones de velocidad, como en el caso de un automóvil que sólo se puede desplazar en la dirección en la que apuntan las llantas delanteras, entonces se habla de un robot con restricciones *no holonómicas*. Es importante aclarar que una restricción es no holonómica cuando sólo se puede representar en términos de velocidades pero no de posición, por ejemplo, un robot diferencial sólo se puede mover con una velocidad

perpendicular al eje que une las llantas, sin embargo, con la planeación adecuada, es posible alcanzar cualquier configuración. El caso contrario es la restricción de movimiento al plano XY que se puede expresar como $\dot{z} = 0$ (en términos de la velocidad) o como $z = Cte$ (en términos de la posición).

Finalmente, es importante tomar en cuenta si el modelo del robot será dinámico o solamente cinemático. En el primer caso las señales de entrada pueden ser fuerzas o pares y en el segundo se asume que las velocidades se pueden manipular arbitrariamente y por lo tanto pueden considerarse como señales de entrada.

2.2.3 Características de los algoritmos

Una vez que se ha definido la tarea a realizar y con base en las características del robot, puede elegirse un algoritmo para resolver determinado problema. Si lo que nos interesa es que el robot ejecute una terea en un tiempo mínimo, con el menor gasto de energía posible o recorriendo la distancia más corta, entonces se requiere de un algoritmo *óptimo*.

El *costo computacional* de un algoritmo se refiere a la cantidad de recursos necesarios para resolverlo, es decir, cantidad de memoria y tiempo de ejecución. La complejidad se expresa como función de la cantidad de datos de entrada y se suele clasificar como exponencial, polinomial, logarítmica, factorial, etc, dependiendo de la función que pueda acotar ya sea el peor caso o un promedio de los distintos casos. Los datos de entrada pueden ser el número de grados de libertad o el número de estados en un planeador, por ejemplo.

Otra característica importante de los algoritmos es su *completitud* (se empleará esta palabra a falta de una mejor traducción del vocablo *completeness* en inglés). Se dice que un algoritmo es completo si garantiza encontrar la solución al problema si es que ésta existe. Algunas veces, para disminuir la complejidad de un algoritmo se maneja una completitud para una resolución dada, es decir, el algoritmo garantiza encontrar una solución si se maneja un cierto nivel de discretización. Un ejemplo de esto se verá en la sección 2.3.2, en el que una ruta se puede calcular sólo si el espacio navegable se discretiza.

Una última característica que está más relacionada con la implementación del planeador que con los algoritmos en general, es aquella que se refiere al momento en que se realiza la planeación. Si el planeador construye un plan previo a la ejecución se dice que éste es *fuera de línea*. Si el planeador actualiza el plan conforme lo ejecuta, entonces se dice que es *en línea*.

2.3 Planeación de rutas

2.3.1 Celdas de ocupación

El primer paso para resolver la tarea de navegación es la planeación de una ruta con base en un mapa construido previamente. Como se mencionó en la sección 2.2.1, un mapa es una representación del ambiente que contiene información útil para tomar decisiones. En este curso se emplerán mapas basados en *celdas de ocupación*. En esta representación, el espacio se discretiza con una resolución determinada y a cada celda se le asigna un número $p \in [0, 1]$ que indica

su nivel de ocupación. En un enfoque probabilístico este número se puede interpretar como la certeza que se tiene de que una celda esté ocupada.

2.3.2 El algoritmo A*

Si se tiene un mapa del ambiente con celdas de ocupación, el problema de la planeación de rutas se puede resolver aplicando un algoritmo de búsqueda en grafos. En este caso cada celda representa un nodo en el grafo y se considera que está conectada únicamente con aquellas celdas vecinas que pertenezcan al espacio libre. Para determinar los nodos vecinos se puede utilizar conectividad cuatro u ocho.

A* es un algoritmo de búsqueda que explora la ruta con el menor costo esperado. Para un nodo n, el costo esperado $f(n)$ se calcula como

$$f(n) = g(n) + h(n)$$

donde $g(n)$ es el costo de la ruta desde el nodo origen hasta el nodo n y $h(n)$ es una heurística que determina *un* costo que se esperaría tener desde el mismo nodo n hasta el nodo objetivo. Este costo esperado de hecho subestima el valor real, es decir, se debe cumplir que $h(n) \leq g(n) \quad \forall\, n \in Grafo$.

En la búsqueda por A* se manejan dos conjuntos principales: la *lista abierta* y la *lista cerrada*. La lista abierta contiene todos los nodos que han sido visitados pero no expandidos y la cerrada, aquellos que han sido visitados *y* expandidos (también llamados nodos conocidos). El algoritmo **??** muestra los pasos en pseudocódigo para implementar A*.

2.3.3 Suavizado por descenso del gradiente

Dado que la ruta que arroja A* se calcula a partir de celdas de ocupación, las vueltas siempre serán ángulos rectos, en caso de que se haya utilizado conectividad cuatro, o bien vueltas a 45^o, en caso de que se haya utilizado conectividad ocho. En cualquier caso, no es deseable tener "esquinas" en las rutas por varias razones: la primera de ellas es que la función de posición no es diferenciable en dichas esquinas y, además, este tipo de vueltas ocasionan cambios bruscos en las señales de control, lo que finalmente ocasiona daños a los actuadores del robot. Por otro lado, las restricciones de movimiento en algunos tipos de bases (como es el caso de los automóviles) pueden impedir ejecutar vueltas en ángulo recto.

Por lo anterior, es conveniente suavizar la ruta calculada por A* de modo que la nueva ruta sea lo suficientemente parecida a la original pero al mismo tiempo, lo suficientemente suave para evitar curvas muy pronunciadas. La figura **??** muestra un ejemplo de suavizado con dos casos extremos: la ruta roja es una ruta igual a la original, la azul es una ruta sin vueltas pero que ha dejado de ser una ruta útil, y la verde, que es un *promedio ponderado* de las dos anteriores.

Para poder obtener una ruta como la verde de la figura **??**, plantearemos una función de costo que tome en cuenta el parecido con la ruta original y el *nivel de suavizado* que se desee. Estos dos criterios están en conflicto: entre más suavizado, menor será el parecido con la ruta original y viceversa, es decir, la función de costo será grande con mucho suavizado y también debe ser muy grande si la ruta es muy parecida a la orignal. La idea es que dicha función de costo tenga

un mínimo en un punto intermedio de modo que este punto corresponda a una ruta como la verde de la figura **??**.

Las funciones cuadráticas son una buena opción como función de costo ya que tienen un mínimo global y además son diferenciables. Para el suavizado de la ruta se propone la función de costo

$$V = \frac{1}{2}\alpha \sum_{i=1}^{n} (p_i - q_i)^2 + \frac{1}{2}\beta \sum_{i=1}^{n-1} (p_i - p_{i+1})^2 \tag{2.1}$$

donde $Q = \{q_1 \dots q_n\}$ son todos los puntos $q = [xy]$ que forman la ruta original calculada con A* y $P = \{p_1 \dots p_n\}$ son los puntos de la ruta ya suavizada. La constante $\alpha \geq 0$ es un parámetro de diseño para indicar cuánto peso se le quiere dar al parecido de la ruta original con la suavizada y $\beta \geq 0$ indica el peso que se le da al suavizado en sí. Con $\alpha = 0$ se obtendría una línea recta, como la ruta azul de la figura **??**, y con $\beta = 0$ se obtendría una ruta igual a la original.

La ruta suavizada se obtiene encontrando el argumento que minimiza la función 2.1, sin embargo, dada la cantidad tan grande de variables, es difícil encontrarlo analíticamente. Una forma más sencilla es mediante el médoto del descenso del gradiente, que consiste en mover el argumento pequeñas cantidades proporcionales al gradiente de la función V y en sentido contrario a éste. Dado que el cambio entre cada iteración es proporcional a ∇V, se puede asumir que cuando el cambio en el argumento es menor que una tolerancia, entonces se ha alcanzado el mínimo.

El algoritmo 2.1 contiene los pasos en pseudocódigo para implementar descenso del gradiente. La constante $\delta > 0$ debe ser lo suficientemente pequeña para evitar inestabilidad en el algoritmo, sin embargo, se debe considerar que entre más pequeña sea ésta, mayor será el costo computacional.

Algorithm 2.1: Descenso del gradiente.

Data: Función V cuyo mínimo se desea encontrar, condición inicial $p(0)$, tolerancia tol.
Result: Argumento p que minimiza V.

1 $i = 0$
2 $p_i \leftarrow p(0)$
3 **while** $\|\nabla V(p_i)\| > tol$ **do**
4 $\quad p_{i+1} \leftarrow p_i - \delta \nabla V(p_i)$
5 $\quad i \leftarrow i + 1$
6 Regresar p_i

La ecuación 2.1 toma como argumentos las posiciones tanto de la ruta original como de la suavizada, sin embargo, dado que sólo varían los puntos de la nueva ruta, se puede considerar que el gradiente ∇V está dado por: Nótese que se está derivando con respecto a p (puntos de la ruta suavizada), no con respecto a q (puntos de la ruta original). Recuerde que cada punto de la ruta tiene coordenadas $[xy]$, por lo que el descenso de gradiente se tiene que aplicar a ambas coordenadas.

2.4 Modelo cinemático y control de posición

En la sección 2.3 se explicó el modo en que se puede calcular una ruta desde una configuración inicial hasta una final. El siguiente paso es hacer que el robot en efecto siga dicha ruta y para ello se empleará la teoría de control.

2.4.1 Modelo en variables de estado

Para poder diseñar una ley control que haga que el robot siga la ruta deseada es necesario primero tener un modelo del robot. Para los fines de este curso, un modelo cinemático es suficiente.

Considérese un robot diferencial como el de la figura **??** en el que la configuración está dada por tres valores $[x_r, y_r, \theta_r]$. Considerando sólo la parte cinemática y asumiendo que no existe deslizamiento en las llantas, el modelo del robot está dado por

$$\dot{x_r} = \frac{v_l + v_r}{2}\cos\theta_r \tag{2.2}$$

$$\dot{y_r} = \frac{v_l + v_r}{2}\sin\theta_r \tag{2.3}$$

$$\dot{\theta_r} = \frac{v_r - v_l}{L} \tag{2.4}$$

donde v_l y v_r son las velocidades lineales de las llantas izquierda y derecha respectivamente, consideradas como señales de entrada, y L es el diámetro del robot medido de eje a eje de las llantas. Se considera que el centro del robot está en el centro de dicho eje.

Nótese que no se está modelando la parte dinámica del robot, esto es, se considera que el estado del robot está dado por los mismos tres valores $[x_r, y_r, \theta_r]$ y que las velocidades de las llantas se pueden fijar de manera arbitraria. En realidad, esto no sucede así. La verdadera señal de control es el voltaje que se fija en las terminales de los motores, sin embargo, se puede considerar que las dinámicas tanto eléctrica como mecánica de dichos motores son lo suficientemente rápidas para suponer que un voltaje en el motor se reflejará *rápidamente* en una velocidad angular.

2.4.2 Estabilidad de sistemas no lineales

Cuando se tiene un sistema lineal invariante con el tiempo de la forma

$$\dot{x} = Ax + Bu \tag{2.5}$$

$$z = Cx + Du \tag{2.6}$$

es sencillo probar su estabilidad: basta con obtener los valores propios de la matriz A y si todos tienen parte real estrictamente menor que cero, entonces se puede asegurar que el sistema (2.5)-(2.6) es exponencialmente estable, o bien, se puede obtener la función de transferencia equivalente y verificar que todas las raíces del polinomio denominador tengan parte real estrictamente negativa.

Sin embargo, el sistema dado por las ecuaciones (2.2)-(2.4) es no lineal y por lo tanto su estabilidad no se puede analizar con las herramientas descritas en el párrafo anterior. La estabilidad

en el sentido de Lyapunov o estabilidad de puntos de equilibrio es una herramienta que nos permite analizar la estabilidad de sistemas no lineales.

Considérese un sistema dinámico de la forma

$$\dot{x} = f(x) \tag{2.7}$$

donde $x \in R^n$ es el vector de estados. Se dice que x_e es un punto de equilibrio del sistema (2.7) si se cumple que

$$x(0) = x_e \implies x(t) = x_e \qquad \forall t > 0 \tag{2.8}$$

es decir, si el sistema "comienza" en el punto x_e, entonces permanece en ese punto para todo tiempo finito.

La estabilidad en el sentido de Lyapunov se define para puntos de equilibrio y existen varios tipos. Para fines de este curso sólo se utilizará la estabilidad asintótica. Suponga que se tiene un sistema de la forma (2.7) con un punto de equilibrio $x_e = 0$. Se dice que x_e es asintóticamente estable, es decir, las trayectorias del sistema convergen a él conforme el tiempo tiende a infinito, si existe una función $V : R^n \to R$ tal que

$$V(0) = 0 \qquad \text{y} \qquad V(x) > 0 \quad \forall x \neq 0 \tag{2.9}$$

$$\dot{V}(x) < 0 \tag{2.10}$$

La ecuación (2.9) define a V como una función positivia definida, es decir, como una función escalar de variable vectorial cuyo valor es cero sólo para el vector cero, y estrictamente mayor que cero para cualquier valor diferente de cero. Esta función puede considerarse como una representación de la energía del sistema, solo que expresada en un sistema de coordenadas diferentes a las usuales.

La equación (2.10) representa la variación de energía del sistema conforme pasa el tiempo y, si ésta es siempre negativa, significa que la energía siempre disminuirá y en algún momento llegará a cero, es decir, el sistema dejará de "moverse". Esta derivada se tiene que evaluar a lo largo de las trayectorias del sistema (2.7).

Nótese que se está suponiendo que el sistema tiene un punto de equilibrio en cero, sin embargo, esto no provoca pérdida de generalidad dado que cualquier punto de equilibrio se puede trasladar a cero mediante una transformación de coordenadas. Por otro lado, en la ecuación (2.7) la dinámica sólo es función de los estados y no aparecen señales de entrada, sin embargo, si se diseña un control realimentado, las señales de entrada quedarán en términos de los estados y la dinámica en lazo cerrado podrá expresarse en la forma (2.7).

Regresando a la aplicación del robot móvil, si se quiere que el robot alcance una posición deseada, las velocidades deben ser tales que el robot tenga un punto de equilibrio asintóticamente estable en dicha posición deseada.

2.4.3 Leyes de control

La idea del control es diseñar las señales v_l y v_r de modo que se garantice que el robot llegue a la posición (x_g, y_g) aun en presencia de incertidumbres (como las dinámicas no modeladas) y perturbaciones. Se proponen dos leyes de control para las cuales es necesario definir primero un par de variables.

Considérese el esquema de la figura **??**. El ángulo deseado θ_g corresponde al ángulo del vector de error de posición $[x_g - x_r, y_g - y_r]$, esto es

$$\theta_g = (y_g - y_r, x_g - x_r)$$

de donde se define el error de ángulo

$$e_\theta = \theta_g - \theta_r = \text{atan2}\,(y_g - y_r, x_g - x_r) - \theta_r$$

El error de distancia r es simplemente la magnitud del vector de error de posición:

$$r = [(x_g - x_r) + (y_g - y_r)]^{1/2}$$

En la primera ley de control las velocidades se calculan tomando en cuenta tanto el error de posición como el de ángulo:

$$v_l = -k_\theta e_\theta + k_d r e^{-\psi e_\theta^2} \tag{2.11}$$

$$v_r = k_\theta e_\theta + k_d r e^{-\psi e_\theta^2} \tag{2.12}$$

Nótese que los primeros términos tienen igual magnitud pero signo opuesto, lo que provoca una velocidad angular que es proporcional al error de ángulo. Los segundos términos, al tener el mismo signo y misma magnitud, provocan una velocidad lineal que es proporcional al error de distancia, es decir, el robot se irá deteniendo conforme se acerque al punto meta. La exponencial sirve para hacer pequeña la velocidad lineal cuando el error de ángulo es grande, es decir, este término logra que el robot comience a avanzar hasta que esté apuntando en la dirección correcta.

En esta ley de control se tienen tres parámetros de diseño: $k_\theta > 0$, $k_d > 0$ y $\psi > 0$. Las dos primeras determinan la rapidez con que el robot girará y avanzará hacia el punto meta. La tercera es muy importante. Un valor de ψ muy grande hará que la velocidad lineal decrezca muy rápido cuando crece el error de ángulo, es decir, el robot comenzará a avanzar hasta que esté apuntando casi sin error hacia la meta. Por el contrario, una ψ muy pequeña hará que el robot describa curvas muy grandes.

La segunda ley de control sólo toma en cuenta el error de ángulo:

$$v_l = v_{max} e^{-\frac{e_\theta^2}{\alpha}} + \frac{D}{2}\omega_{max}\left(\frac{2}{1 + e^{-\frac{e_\theta}{\beta}}} - 1\right) \tag{2.13}$$

$$v_r = v_{max} e^{-\frac{e_\theta^2}{\alpha}} - \frac{D}{2}\omega_{max}\left(\frac{2}{1 + e^{-\frac{e_\theta}{\beta}}} - 1\right) \tag{2.14}$$

En estas ecuaciones se tienen cuatro parámetros de diseño: v_{max} y ω_{max} son las velocidades lineales y angulares máximas respectivamente que el robot alcanzará durante su movimiento.

La constante α tiene una función parecida a la ψ en las ecuaciones (2.11)-(2.12), solo que en este caso, al estar dividiendo, una *alpha* más pequeña provocará un descenso más pronunciado en la magnitud de la velocidad lineal. El valor de β determina qué tanto decrece la velocidad angular conforme decrece el error de ángulo.

2.5 Campos potenciales artificiales

En la sección 2.3 se explicó cómo planear una ruta cuando se tiene una representación del ambiente y la seccion 2.4 se dedicó al diseño de leyes de control que permitan seguir la ruta planeada. Sin embargo, ¿qué sucede cuando el robot tiene que evadir obstáculos que no están en el mapa y que no fueron contemplados en la planeación de la ruta?

Un forma de evadir obstáculos es mediante campos potenciales artificiales. Una función potencial es una función real diferenciable $U : R^n \to R$ que puede verse como una función de energía y que, por lo tanto, su gradiente $\nabla U(q)$, donde q es la posición del robot, representa fuerza.

El gradiente de la función, dado por

$$\nabla U(q) = \left[\frac{\partial U}{\partial q_1}, \ldots, \frac{\partial U}{\partial q_n}\right] \tag{2.15}$$

es un vector que apunta en la dirección de máximo cambio de U. Si esta función potencial se diseña de modo que tenga un mínimo en la posición meta y máximos locales en la posición de cada obstáculo que se desee evadir, entonces se puede mover al robot mediante el algoritmo de descenso del gradiente y esto hará que el robot continúe moviéndose hasta encontrar un punto q^* en el cual $\nabla U(q^*) = 0$.

Los valores q que satisfacen $\nabla U(q^*) = 0$ son llamados puntos críticos y pueden ser máximos locales, mínimos locales o puntos silla. Para determinar la naturaleza de un punto crítico, se puede utilizar la matriz Hessiana, dada por

$$H(q) = \begin{bmatrix} \frac{\partial^2 U}{\partial q_1^2} & \cdots & \frac{\partial^2 U}{\partial q_1 \partial q_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 U}{\partial q_n \partial q_1} & \cdots & \frac{\partial^2 U}{\partial q_n^2} \end{bmatrix} \tag{2.16}$$

Si $H(q)$ evaluada en q^* es no singular, significa que q^* es un punto crítico aislado. Si $H(q^*)$ es positiva definida, entonces q^* es un mínimo local, si es negativa definida, se trata de un máximo local, y en otro caso, q^* es un punto silla. Si $H(q^*)$ es singular, entonces significa que q^* no es un punto aislado, es decir, que la función potencial U es "plana" en la región que contiene a q^*.

En general, si el robot se mueve siempre en sentido contrario al gradiente, en algún momento llegará a un mínimo local. Puede suceder que la condición inicial del robot corresponda a un máximo local o a un punto silla, en cuyo caso el robot no se moverá, pues en estos puntos el gradiente es cero. Esta situación es muy improbable, además, cualquier perturbación hará que el robot "se mueva" de ese punto y entonces se dirigirá al mínimo local, es decir, los puntos silla y los máximos son puntos inestables.

2.5.1 Diseño mediante campos atractivos y repulsivos

Como se expuso previamente, la función potencial $U(q)$ debe diseñarse de modo que sea diferenciable, que tenga un mínimo en el punto meta, máximos locales en cada obstáculo, y todos sus puntos críticos deben ser aislados. Una forma sencilla de logralo es diseñando por separado dos campos: uno atractivo y otro repulsivo. La idea es que el punto meta debe generar una "fuerza atractiva" mientras que cada obstáculo debe generar una "fuerza repulsiva".

Algunas veces no es necesario diseñar primero el campo potencial y luego obtener su gradiente, sino que se puede diseñar directamente la fuerza correspondiente al gradiente. La fuerza atractiva se debe diseñar de modo que siempre se aleje del punto meta y las fuerzas repulsivas siempre deben tener una dirección que apunte hacia el obstáculo que se desea evadir, además, las fuerzas repulsivas deben crecer conforme la distancia a dicho obstáculo disminuye y ser muy pequeñas o cero, si el robot está lo suficientemente lejos.

Es importante recordar que las fuerzas tanto atractiva como repulsiva corresponden al gradiente de la función potencial, es decir, representan la dirección de máximo cambio, sin embargo, lo que se desea es alcanzar un mínimo, por lo que al aplicar el algoritmo de descenso del gradiente, el robot se moverá en sentido contrario a estas fuerzas.

En este curso, la fuerza atractiva que se utilizará está dada por

$$F_a = \zeta \frac{(q - q_g)}{\|q - q_g\|} \qquad \zeta > 0 \tag{2.17}$$

donde q es la posición actual del robot y q_g es la posición a donde se desea llegar. Esta fuerza tiene una magnitud ζ constante pero su dirección apunta siempre en sentido contrario al punto meta.

La fuerza de repulsión está dada por:

$$F_r = \begin{cases} \eta \left(\sqrt{\frac{1}{\|q_{oi} - q\|} - \frac{1}{d_0}} \right) \frac{q_{oi} - q}{\|q_{oi} - q\|} & \text{si} \quad \|q_{oi} - q\| < d_0 \\ 0 & \text{en otro caso} \end{cases} \tag{2.18}$$

donde q_{oi} es la posición del objeto i y d_0 es una distancia de influencia tal que, si el robot está a una distancia mayor que ésta del objeto i, entonces la fuerza de repulsión es cero. Nótese que la magnitud de esta fuerza se incrementa conforme el robot se acerca al objeto y la raiz cuadrada produce una transición suave cuando el robot pasa por d_0.

La fuerza resultante se calcula como

$$F = F_a + \frac{1}{n} \sum_{i=1}^{n} F_r \tag{2.19}$$

es decir, la fuerza atractiva más el promedio de las fuerzas repulsivas producidas por cada obstáculo.

Una desventaja de generar campos potenciales mediante la suma de un campo atractivo más uno repulsivo es la posibilidad de tener mínimos locales. Esto se puede solucionar con la generación de funciones potenciales mediante otros métodos como el algoritmo de *wavefront*. Este algoritmo no tiene el problema de los mínimos locales pero tienen la desventaja de necesitar de una discretización del espacio.

2.5.2 Movimiento por descenso del gradiente

Una vez diseñado el campo potencial y calculado el respectivo gradiente, sólo basta con mover al robot de acuerdo con el algoritmo 2.2. La constante $\alpha > 0$ debe ser lo suficientemente pequeña para evitar oscilaciones pero sin que ello conlleve un costo computacional muy alto.

Aunque en el algoritmo se indica que el robot seguirá moviéndose mientras la magnitud del gradiente sea mayor que cero, en una implementación real se fija una tolerancia y cuando la magnitud es menor que ésta, se detiene al robot.

Algorithm 2.2: Descenso del gradiente para mover al robot a través de un campo potencial.

Data: Posición inicial q_s, posición final q_g, posiciones q_{oi} de los obstáculos
Result: Secuencia de puntos $\{q_0, q_1, q_2, \dots\}$

$q_0 \leftarrow q_s$
$i \leftarrow 0$
while $\|\nabla U(q_i)\| > 0$ **do**
    $q_{i+1} \leftarrow q_i - \alpha \nabla U(q_i)$
    $i \leftarrow i + 1$

Finalmente, para que el robot se mueva hacia cada punto q_i generado por el descenso del gradiente, se pueden utilizar las leyes de control (2.13)-(2.14).

2.6 Localización

El problema de la localización consiste en determinar la configuración del robot dadas las lecturas de los sensores y una representación del ambiente. La localización puede ser global o local. En este curos únicamente se cubrirán dos algoritmos para localizar al robot: el Filtro de Kalman Extendido y el método del histograma.

2.6.1 El Filtro de Kalman Extendido

El Filtro de Kalman es un algoritmo que sirve para estimar un conjunto de variables a partir de un conjunto de mediciones que contienen ruido y de un modelo del sistema cuyos procesos también contienen ruido. Este filtro puede ser considerado como un filtro bayesiano, pues la estimación de las variables es en realidad la estimación de una distribución de probabilidad conjunta dada una distribución previa y un conjunto de mediciones.

Desde el punto de vista de la teoría de control, el Filtro de Kalman es un observador en el que la ganancia se calcula con base en las propiedades estadísticas del ruido tanto en las mediciones como en el proceso.

El Filtro de Kalman Extendido (EKF por sus siglas en inglés) es la versión del Filtro de Kalman para sistemas no lineales. En el EKF se hace una linealización alrededor de las estimaciones actuales y se aplican los mismos pasos que se usan en sistemas lineales.

Para realizar la estimación de la posición del robot, primero se requiere una versión discreta del modelo cinemático. Discretizando el modelo (2.2)-(2.4), y considerando que se tiene ruido de proceso, se tiene:

$$x_{k+1} = x_k + \Delta t \frac{v_l + v_r}{2} \cos\theta_k + \nu_1 \tag{2.20}$$

$$y_{k+1} = y_k + \Delta t \frac{v_l + v_r}{2} \sin\theta_k + \nu_2 \tag{2.21}$$

$$\theta_{k+1} = \theta_k + \Delta t \frac{v_r - v_i l}{L} + \nu_3 \tag{2.22}$$

donde Δt representa el periodo de muestreo y $\nu = [\nu_1\ \nu_2\ \nu_3]^T$ es ruido gaussiano sin correlación temporal, media cero y matriz de covarianza Q.

La estimación de la posición del robot por filtro de Kalman consta de los siguientes pasos:

Predicción: Con base en el modelo cinemático y el modelo de observación, se predice el siguiente estado y la salida considerando que el ruido es cero tanto en el modelo de transición de estados como en el modelo de obsevación:

$$\begin{aligned}
\hat{X}(k+1|k) &= \hat{F}(X(k|k), u(k)) \\
\hat{Z}(k+1|k) &= \hat{X}(k+1|k) \\
P(k+1|k) &= J(k)P(k|k)J^T(k) + Q
\end{aligned}$$

donde la matriz P es la covarianza del error de estimación y J es el Jacobiano de la función F con respecto a X.

Actualización: Con base en el error de observación y la covarianza del error de estimación se calcula el estado siguiente estimado.

$$\begin{aligned}
S(k+1) &= H(k+1)P(k|k+1)H^T(k+1) + R \\
K(k+1) &= P(k+1|k)H^T(k+1)S(k+1)^{-1} \\
\hat{X}(k+1|k+1) &= \hat{X}(k+1|k) + K(k+1)(Z(k+1) - \hat{Z}(k+1|k)) \\
P(k+1|k+1) &= (I - P(k+1)H(k+1))P(k+1|k)
\end{aligned}$$

donde H es el Jacobiano del modelo de observación, que en este caso es la identidad dado que se considera que se mide el estado completo del robot. La matriz K es conocida como ganancia de Kalman.

2.6.2 Método del histograma

La localización por el método de histograma permite estimar la posición del robot con base en observaciones del ambiente pero tomando en cuenta las probabilidades de error tanto en la observación como en el movimiento. Se asume que el robot sólo puede estar en configuraciones discretas bien definidas por lo que el método es factible sólo cuando se tienen pocos grados de libertad, como en el caso del robot diferencial que se mueve sólo en un plano.

Considere la posición del robot como un variable aleatoria discreta X que puede tomar los valores $x_i = \{x_1 \ldots x_n\}$ y las observaciones que puede realizar el robot como otra variable aleatoria

O con posibles valores $o_i = \{o_1 \dots o_m\}$. Las observaciones del robot pueden ser cualquier tipo de *marca* que se pueda extraer a partir de la información de los sensores.

Se asume que la posición inicial se desconoce por completo, por lo que el primer paso es asignar a cada posición la misma probabilidad

$$P(x_i) = \frac{1}{n} \tag{2.23}$$

es decir, inicialmente se tiene una distribución uniforme que indica total incertidumbre sobre la posición actual del robot. El objetivo es construir una función de densidad de probabilidad que, conforme el robot se mueva y realice observaciones, tienda hacia una distribución unimodal cuyo máximo indique la posición actual del robot.

Observación.

Para comenzar a estimar la posición del robot es necesario realizar observaciones el ambiente. Para ello, a cada nodo v del grafo G se le asigna una marca que "debería" ser observada si el robot se encontrara en ese nodo.

Con cada observación se debe modificar la probabilidad de cada nodo, dado que se identificó determinada marca. Considérese la posición del robot como una variable aleatoria V que puede tomar los valores $v_i = \{$entrada-izquierda, sala-frente,...$\}$ y sea $O = \{o_j\}$ la variable aleatoria que representa la marca observada.

Una vez hecha una observación, se desea actualizar la probabilidad de cada nodo v_i, dada la observación o_j, esto es

$$P(V = v_i | O = o_j) \tag{2.24}$$

que de acuerdo con el teorema de Bayes se puede calcular como

$$P(v_i|o_j) = \frac{P(o_j|v_i)\,P(v_i)}{P(o_j)} \tag{2.25}$$

En la ecuación anterior, el término $P(o_j|v_i)$ es la probabilidad de que se observe una determinada marca si se asume que el robot está en el nodo v_i, es decir, la probabilidad de o_j en la hipótesis v_i. Esta probabilidad se calcula de forma estadística en la etapa de entrenamiento. Si se observa la marca o_j existen dos posibles casos: que la marca corresponda al nodo en el que se asume el robot se encuentra o, que no corresponda. Llamaremos al primer caso un acierto verdadero (AV), y al segundo, un falso positivo (FP). En este trabajo se consideraron las probabilidades

$$P(AV) = 0{,}6 \tag{2.26}$$

$$P(FP) = 0{,}25 \tag{2.27}$$

El término $P(v_i)$ denota la probabilidad a priori del nodo v_i y el término $P(o_j)$ puede calcularse, de acuerdo con el teorema de la probabilidad total, como

$$P(o_j) = \sum_{i}^{n} P(o_j|v_i)\,P(v_i) \tag{2.28}$$

donde n es el número total de nodos del grafo G que representa el ambiente.

Movimiento Como se explicó anteriormente, se asume que el robot puede ejecutar tres movimientos: frente, izquierda y derecha. Para modelar la incertidumbre en el movimiento, se asume que existen dos posibles casos: el robot alcanza el nodo deseado o, el robot se queda en el nodo en que ya estaba. Llamaremos al primer caso movimiento correcto (MC) y al segundo, movimiento con falla (MF). Al igual que en el modelo de observación, las probabilidades de ambos casos se calcularon mediante experimentos en la etapa de entrenamiento. Los valores usados fueron:

$$P(MC) = 0{,}7 \tag{2.29}$$
$$P(MF) = 0{,}3 \tag{2.30}$$

Para calcular la probabilidad de cada nodo después de un movimiento, se usa nuevamente el teorema de la probabilidad total. La probabilidad a posteriori de un nodo v_i es la probabilidad de que el robot haya llegado a ese nodo por un movimiento correcto o bien, por un movimiento con falla, es decir

$$P\left(v_i^t\right) = \sum_k P\left(v_i^t | v_k^{t-1}\right) P\left(v_k^{t-1}\right) \tag{2.31}$$

donde los superíndices t y $t-1$ indican el tiempo actual y el tiempo anterior, respectivamente. Si se considera que el robot solo puede llegar a un nodo por un movimiento correcto (viniendo desde otro nodo) o por uno con falla (el robot permanece en el mismo nodo), entonces la ecuación se puede escribir como

$$P\left(v_i^t\right) = P\left(v_i^t | v_k^{t-1}\right) P\left(v_k^{t-1}\right) + P\left(v_i^t | v_i^{t-1}\right) P\left(v_i^{t-1}\right) \tag{2.32}$$

El término $P\left(v_i^t | v_k^{t-1}\right)$ indica la probabilidad de que el robot llegue al nodo v_i desde el nodo v_k (un movimiento correcto), es decir

$$P\left(v_i^t | v_k^{t-1}\right) = P(MC) \tag{2.33}$$

y de modo similar

$$P\left(v_i^t | v_i^{t-1}\right) = P(MF) \tag{2.34}$$

2.7 La plataforma ROS

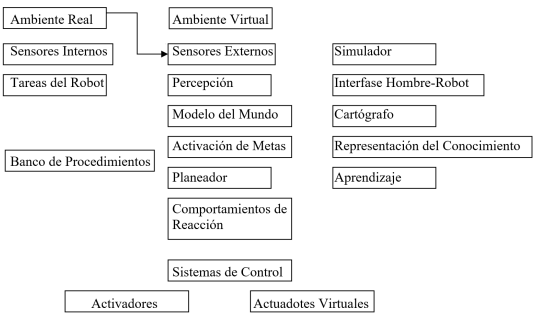
3 Robots Móviles y Agentes Inteligentes

3.1 Máquina de Estado

- Entradas
- Máquinas
- Salidas

3.2 Máquina de Estado Aumentada

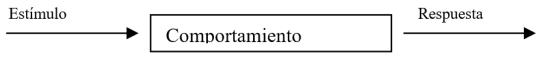
Una entrada (o salida) de una máquina puede ser sustituida por la salida de otra máquina de estados. Esa bloquea la entrada (o salida) ordinaria y pone su salida.



3.3 Arquitecturas Reactivas (o por Comportamientos)

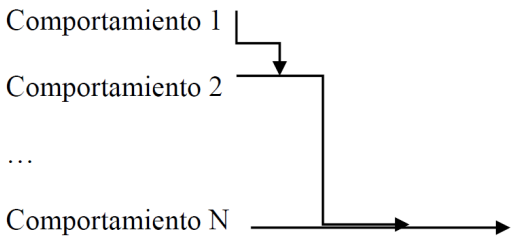
Diagramas de Stimulus Response (SR)

La idea es que la respuesta sea instantánea. En los comportamientos puros, la respuesta únicamente depende del estímulo.



3.3.1 Organización de los comportamientos:

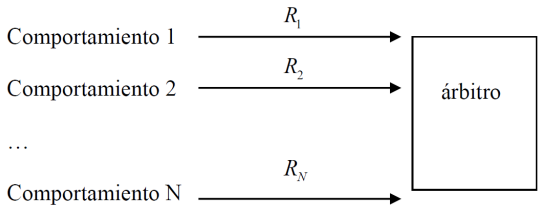
Esquema 1



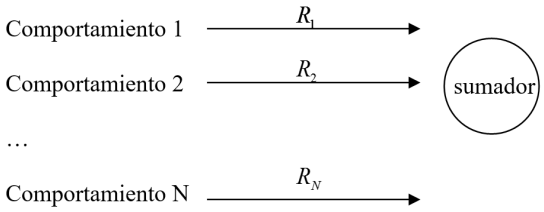
Algunos comportamientos bloquean la salida de otros. Solamente la salida de uno es el que se usa, todos se activan. El diagrama anterior decide la prioridad.

La salida generalmente es un vector que, por ejemplo, indica la magnitud y dirección del movimiento del robot.

Esquema 2



El árbitro decide cuál es la respuesta.



El sumador hace un promedio ponderado por*ganancias*.

$$\sum_{i=1}^{N} g_i R_i$$

La ganancia depende del diseñador y va a hacer que el robot se comporte de diferente forma (robot “audaz” contra robot “conservador”).

Un diagrama del espacio se puede hacer con campos vectoriales (diagramas de flechitas tipo sistemas dinámicos), por ejemplo uno que sea de las fuerzas de atracción y otro de repulsión y sumarlos.

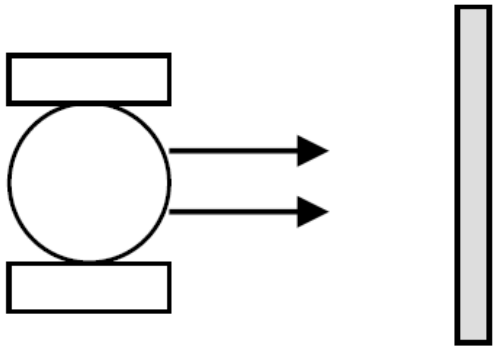
Puedo tener una complejidad grande combinando comportamientos, pues pueden controlar diferentes cosas.

3.4 Máquinas de Estado

3.4.1 Algoritmo para el Comportamiento de Evadir Obstáculos

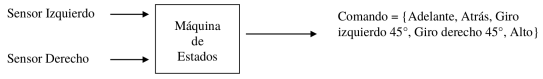
Un robot circula con:

- Dos motores, uno a cada lado.
- Dos sensores adelante, para detectar obstáculos (por ejemplo, infrarrojos, ultrasonido, etc.)

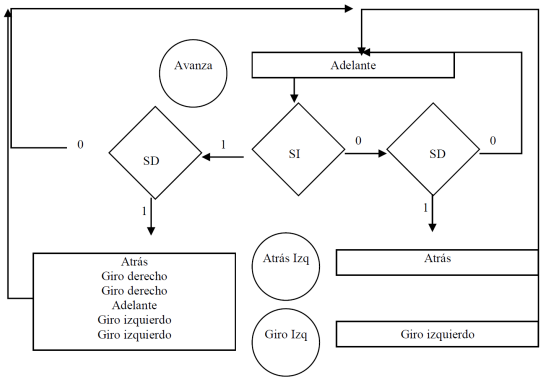


Obviamente:

- Si los sensores no detectan el obstáculo, seguir avanzando.
- Si el sensor derecho lo detecta y el otro no, hacerlo tantito para atrás y girar hacia la izquierda para seguir avanzando
- Si el sensor izquierdo lo detecta y el otro no, hacerlo tantito para atrás y girar hacia la derecha para seguir avanzando
- Si los dos detectan, hacerlo para atrás y giro 90°.



Notación de diagrama de flujo



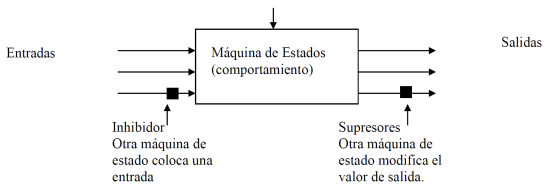
Si se quisiera programar esto con una subrutina, hace todo. En la teoría de comportamientos necesito una respuesta inmediata y necesito que cuando vuelva a entrar se guarde el estado.

3.5 AFSM (Augmented Finite State Machine)

La máquina de estados aplicada a robots representa el comportamiento. Una máquina tradicional de estados tiene entradas y salidas. Rodney Brooks (MIT) inventó la máquina de estados finitos aumentada.

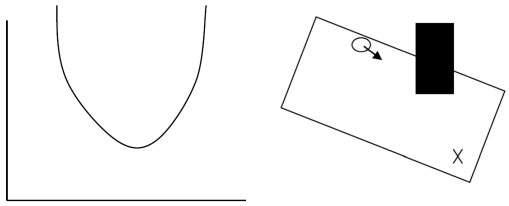
Básicamente el concepto es que alguna de las entradas es inhibida por otra máquina de estados (su salida se convierte la entrada) y en las salidas están los supresores, que bloquean la salida de la máquina y colocan un valor. Por último hay una entrada de "reset" que hace que se coloque en un estado en específico.

Con AFSM podemos tener diferentes capas y hacer una respuesta **jerárquica**.



3.6 Campos Potenciales

Se modela el robot como una canica. Los obstáculos ahora son montañas que ejercen fuerzas de repulsión. El destino es un hoyo. El robot se mueve a través de un campo potencial por la pendiente más pronunciada hasta que lo lleva al destino.



$$y = y_0 + (x - x_0)^2$$

El mínimo se encuentra diferenciando.

$$\frac{\partial y}{\partial x} = 2(x - x_0)$$

$$x^* = x_0$$

Supongamos que no se conoce la ecuación; entonces se utiliza un método iterativo. Empiezo con un punto x_{n-1} cualquiera y dependiendo del valor de la pendiente me muevo hacia la izquierda o la derecha.

Es una relación de recurrencia:

$$x_n = \int(x_{n-1}) = x_{n-1} - \partial \frac{\partial y}{\partial x} \quad \text{donde } \partial \text{ es una constante.}$$

Por ejemplo:
Si consideramos $\delta = \frac{1}{2}$ se tiene $x_n = x_{n-1}\frac{-1}{2}(2(x_{n_1} - x_0)) = x_0$.
Esto da pie a considerar la situación general.

Se utiliza Steepest Descent.

Posición del robot en n: $q_n = [x_n, y_n]^T$

$$q_n = q_{n-1} - \delta f(q_{n-1})$$

donde $f(z) = \frac{F(z)}{\|F(z)\|} = U(z)$ y $U(z)$ es el campo potencial del medio ambiente en donde navega el robot $U(z) = U_{atr}(z) + U_{rep}(z)$ (dos componentes, uno atractor y un repulsor).

Al final se tienen fuerzas de atracción más fuerzas de repulsión.

Se define:

$U_{atr} = \frac{1}{2}\varepsilon_1 \|q - q_{dest}\|^2$(campo de tipo parabólico) donde q_{dest} es la posición del destino. Se ve claro que es parabólico:

$$U_{atr} = \frac{1}{2}\varepsilon_1 (x - x_{dest})^2 + (y - y_{dest})^2$$

La fuerza de atracción en q es:

$$\nabla U_{atr} = \varepsilon_1 \begin{bmatrix} (x - x_{dest}) \\ (y - y_{dest}) \end{bmatrix} = \varepsilon_1 [q - q_{dest}] = F_{atr}(q)$$

La fuerza de atracción se va a convertir en aceleración. Para fuerzas muy grandes se transforma en aceleraciones muy grandes. Esto no se quiere, por lo que se tiene un umbral. Esta fórmula solamente aplica para $\|q - q_{dest}\| < d_1$.

¿Qué pasa en $\|q - q_{dest}\| > d_1$?

$$U_{atr} = E_2 \|q - q_{dest}\|$$

Esto es un campo cónico:

$$U_{atr} = E_2 \sqrt{(x - x_{dest})^2 + (y - y_{dest})^2)}$$

$$\nabla U_{atr} = E_2 \begin{bmatrix} \frac{(x - x_{dest})}{\sqrt{(x - x_{dest})^2 + (y - y_{dest})^2}} \\ \frac{(y - y_{dest})}{\sqrt{(x - x_{dest})^2 + (y - y_{dest})^2}} \end{bmatrix} = \frac{E_2}{\|q - q_{dest}\|} \begin{bmatrix} x - x_{dest} \\ y - y_{dest} \end{bmatrix} = E_2 \frac{q - q_{dest}}{\|q - q_{dest}\|}$$

Campos Repulsivos

¿Cómo representar el obstáculo? Se podría considerar un solo vector en el centroide, una serie de puntos que ejercen repulsión, etc. Depende del diseñador.

$$U_{rep}(q) = \frac{1}{2}\eta \left(\frac{1}{\|q - q_{obs}\|} - \frac{1}{d_0} \right)^2$$

Esto aplica cuando $q - q_{obs}$. En caso contrario, se toma como 0. Esto es para que no lo tomes en cuenta si está demasiado lejos.

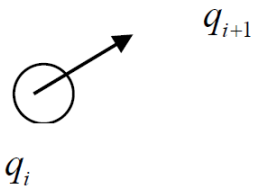
$$\nabla U_{rep}(q) = -\frac{\eta}{\|q - q_{obs}\|^2} \left(\frac{1}{\|q - q_{obs}\|} - \frac{1}{d_0} \right) \left(\frac{q - q_{dest}}{\|q - q_{dest}\|} \right) = F_{rep}(q)$$

En general,

$$F(q) = F_{atr}(q) + \sum_{i=1}^{n} F_{rep}(q)$$

$$f(q) = \frac{F(q)}{\|F(q)\|} = \nabla U(q)$$

$$q_{i+1} = q_i - \delta_i f(q_i)$$



Ejemplo 3.1

Robot:(1,1)
Obstáculo: hexágono centrado en (2,2)
Objetivo:(5,4)
$q_0 = (1,1)$
Se va a tomar $d_0 = 5, \varepsilon_1 = 1, \eta = 2, \delta_0 = 1$ y se va a utilizar la parabólica

Se tiene:

$$\begin{aligned} F_{atr}(q_0) &= \varepsilon_1(q_0 - q_{dest}) \\ &= (-4,-3) \end{aligned}$$

$$F(q_0) = -\frac{\eta}{\|q_0 - q_{obs}\|^2}\left(\frac{1}{\|q_0 - q_{obs}\|} - \frac{1}{d_0}\right)\left(\frac{q_0 - q_{obs}}{\|q_0 - q_{obs}\|}\right)$$

$$= \frac{-2}{\sqrt{2}}\left(\frac{1}{\sqrt{2}} - \frac{1}{5}\right)\left(\frac{1}{2\sqrt{2}}\right)(-1,-1)$$

$$= -(0.3585, 0.3585)$$

La fuerza total es:

$$F(q_0) = (-3.64, -2.64) \qquad f(q_0) = (-0.8091, -0.5868)$$

Finalmente:

$$\begin{aligned} q_1 &= q_0 - \delta_0(q_0) \\ &= (1,1) + (0.8091, 0.5868) \\ &= (1.8091, 1.5868) \end{aligned}$$

Obsérvese que no hemos considerado la **orientación inicial del robot**.

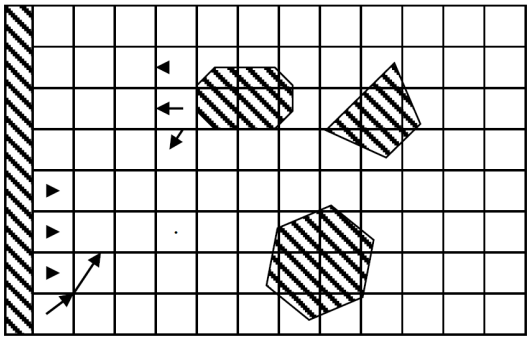
¿Cómo encontrar las constantes? Empíricamente

Nota: Si no conocemos obstáculos no podemos calcular las fuerzas de repulsión. En estos casos, se tiene que usar la información de los sensores para identificar al obstáculo y, con base en eso, calcular estas fuerzas.

Cálculo de los vectores de repulsión usando la posición de los obstáculos fijos

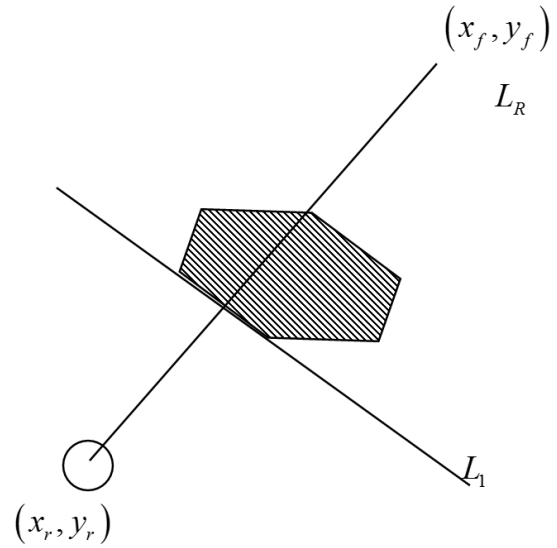
Los obstáculos conocidos se representarán por polígonos que tienen nodos ordenados en el sentido de las manecillas del reloj.

El espacio se divide en celdas y en cada celda se calcula la fuerza total de repulsión.



Etc.

¿Cómo encontrar la fuerza de repulsión en cada celda?



El punto x_r, y_r es el centro de la celda en donde se está (se asume que el robot está en el centro de la celda).

El punto (x_f, y_f) es un punto arbitrario (se quiere calcular para la mayor cantidad de direcciones posibles. No es en línea sino que se hace offline).

$$L_1 :: y = m_1 x + b_1$$
$$L_R : y = m_R x + b_R$$

Nótese que

$$x_f = d\cos\phi + x_r$$
$$y_f = d\sin\phi + y_r$$

Donde el ángulo es entre el eje x y la línea L_R.

Obs: se puede encontrar

$$m = \frac{y_1 - y_0}{x_1 - x_0} \qquad b = \frac{x_1 y_0 - y_1 x_0}{x_1 - x_0}$$

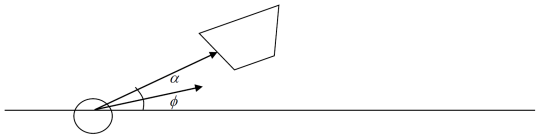
Los puntos de intersección son

$$x_{int} = \frac{b_r - b_1}{m_1 - m_r} \qquad y_{int} = m_r x_{int} + b_r$$

Obsérvese que se tiene que cumplir que $x_{int} \in (x_0, x_1)$ y$y_{int} \in (y_0.y_1)$

3.7 Cómo calcular el campo potencial para el caso de objetos desconocidos

Lo anterior se hace offline y se alimenta al robot. Si hay objetos desconocidos:



Tarea 1
Usar Roc2. Poner obstáculos pickable en el medio. Programar un robot para encontrar esos obstáculos y que los lleve a un lugar. Usar nada más la parte rectangular del mundo. El robot se mueve aleatoriamente. Tener una pila que se vaya bajando.

¿Qué se quiere?
De $(x_{i-1}, y_{i-1}, \phi_{i-|})$ (donde está el robot) necesito llegar a (x_i, y_i, ϕ_i).

Obsérvese que:
$x_i = x_{i-1} + x_i\cos(\phi_i) y_i = y_{i-1} + x_i\sin(\phi_i)$

donde:

$\phi = \phi_{i-1} + \phi_i$
$\phi = \arctan\left(\frac{y_i - y_{i-1}}{x_i - x_{i-1}}\right)$

Esto se conoce como **computación directa**.

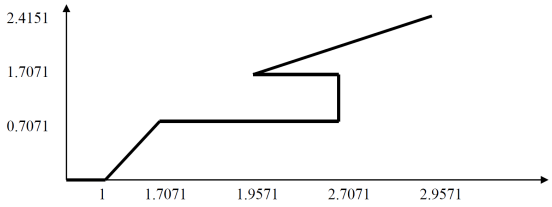
Otra opción es:

$$x_i' = \frac{(x_i - x_{i-1}) + (y_i - y_{i-1})}{\cos(\phi_i) + \sin(\phi_i)}$$

$\phi_i = \phi_i - \phi_{i-1}$ y obviamente también se tiene que $x_i = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}$

Esto se llama **computación inversa** y es la que nos interesa.

Ejemplo



Sup: Los puntos los encontré con campos potenciales (recordar que el campo potencial da el vector de movimiento:

$q_1 = (x_i, y_i) = q_{i-1} - \delta f(q_{i-1})$

$i = 1$
$x_0 = 0, \quad y_0 = 0, \quad \phi_0 = 0$
$x_i = 1 \quad y_i = 0$

Por lo tanto: $\phi_1 = arctan\left(\frac{y_1 - y_0}{x_1 - x_0}\right) = 0 \quad y \quad x_1 = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2} = 1$. Y finalmente:
$\phi_i = \phi_i - \phi_{i-1} = 0$

Ad nauseam.
Los valores son:

i	ϕ_i	x_i
1	0°	1
2	45°	1
3	-45°	1
4	90°	1
⋮	⋮	⋮

3.7.1 Trayectorias

El robot casi siempre avanza en línea recta.
¿A qué velocidad debe ir el robot?

El robot está en $(x_{i-1}, y_{i-1}, \phi_{i-1})$y quiero llegar a (x_i, y_i) .¿Cómo voy a recorrer la distancia x_i?
Tengo t_0 y el tiempo t_f (final).

Se quiere una función $f(t)$ que sea suave. Se asume que el robot está parado en ambos extremos.

$f(0) = 0, f(t_f) = x_i'$ [posiciones inicial y final]

$f'(0) = 0, f'(t_f) = x_i$ [velocidades inicial y final]

Asumiendo una relación cúbica:

$f(t) = a_0 + a_i t + a_2 t^2 + a_3 t^3 \, f'(t_f) = x_i$

$f'(t) = a_1 + 2a_2 t + 3a_3 t^2$

$f''(t) = 2a_2 + 6a_3 t$

Se encuentran:

$a_0, a_1 = 0, a_2 = \dfrac{3x_i}{t_f^2}, a_2 = \dfrac{-2x_i}{t_f^3}$. La ecuación es entonces:

$$f(t) = \frac{3x_i}{t_f^2}t^2 - \frac{2x_i'}{t_f^2}t^3$$

$$f'(t) = \frac{6x_i}{t_f^2}t - \frac{6x_i'}{t_f^2}t^2$$

$$f''(t) = \frac{6x_i}{t_f^2} - \frac{12x_i'}{t_f^2}t$$

Ejemplo 3.2

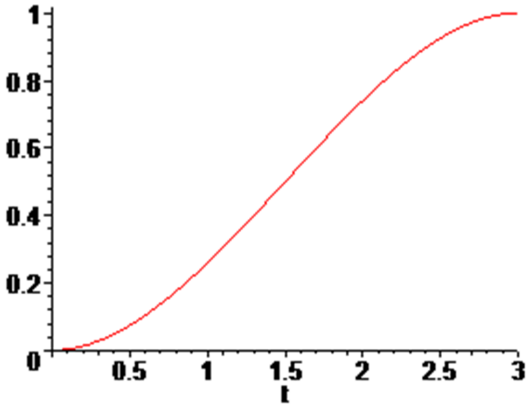
Suponer $t_f = 3\,x_1 = 1$
Entonces

$$f(t) = \frac{1}{3}t^2 - \frac{2}{27}t^3$$

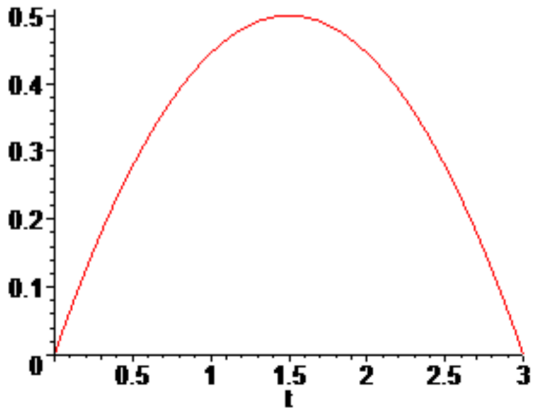
$$f'(t) = \frac{2}{3}t - \frac{6}{27}t^2$$

$$f''(t) = \frac{2}{3} - \frac{12}{27}t$$

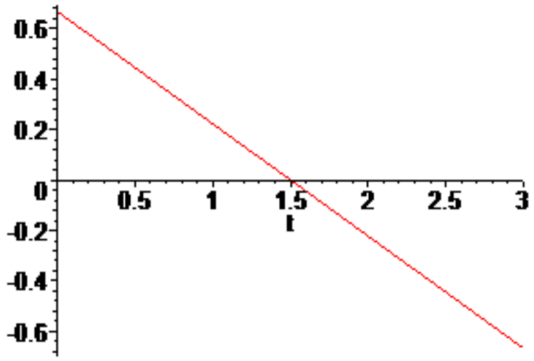
$f(t)$



$f'(t)$



$f''(t)$



En Roc2 para mover es:

mv x_i' ϕ_i t_f

donde la primera está dada en decímetros y el ángulo en radianes y el tiempo en segundos.

En Roc2 las funciones cinemáticas se encuentran en el archivo Roc2UserMv.

Cómo relajar el supuesto de la velocidad final 0

Suavizar la trayectoria.

Una forma: Le ajustas una parábola a la esquinita $y = y_0 + (x - x_0)^2$ con $x_b \leq x \leq x_f$ puntos pre-escogidos.

Otra forma: linearizar (segmentitos de recta).

Rotar y trasladar

polygon type-object $(x0, y0, x1, y1, x2, y2, \ldots ynyn)$ position *room* type-object name *xcyctheta*) ← esta función traslada a xc, yc y rota theta Type-object define un objeto genérico, podemos definir varios objetos (de ese tipo).

Dados (x_p, y_p)

$x = x_p \cos\alpha - y_p \sin\alpha + x_c$

$y = y_p \cos\alpha + x_p \sin\alpha + y_c$

Donde x_c, y_c es el punto fijo sobre el que se está girando.

Ejemplo **3.3**

Polygon mesa 100022220
Position salon mesa1 mesa_principal 44 $\frac{\pi}{4}$

Entonces:

i	x_i	y_i
0	4	4
1	2.59	5.414
2	4	6.28
3	5.4142	5.4142

Ancho del robot Se necesita considerar el ancho del robot. Por lo menos, ensanchar los obstáculos conocidos en el radio del robot. ¿Cómo hacer con los desconocidos? De la lectura del sonar, restarle el radio del robot.

3.8 Escalamiento de polígonos

Suponer (0, 2), (1,3), (2, 2), (1,1) un polígono.

1. Encontrar el centroide
2. Moverlo al origen
3. Multiplicar
4. Volverlo a mover al centroide.

Con $\lambda = 2$

$P_esc = \lambda(P - C) + C$

$P = \{(-1,2),(1,4),(3,2),(1,0)\}$

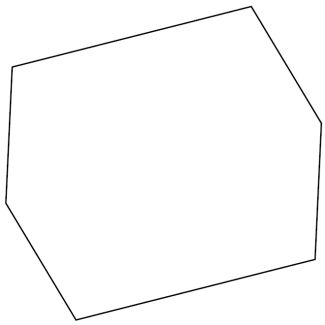
Siguiente tarea
El mundo tiene mesas y otros dos objetos tipo, puestas y rotadas.
Objetos fijos: sillones, mesas, cajas, etc. (al menos 3)

Descomponer el mundo en celdas.

Calcular los vectores de repulsión en cada celda. Si el centroide de la celda queda dentro del obstáculo, está ocupada Hacer una "estrella" hacia todas las direcciones y calcular si la intersección con algo es menor o igual a d0.
La repulsión total es la suma.

Tres archivos: 1) objetos tipo 2) posiciones objetos réplica 3) archivo final .WRL con la descripción del mundo.

Verificación de si un punto está dentro o fuera de un polígono



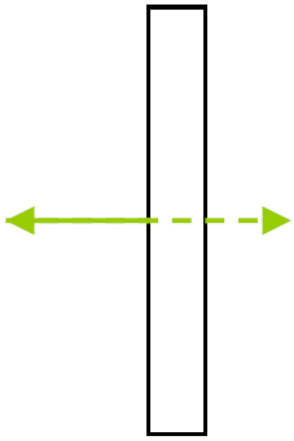
Ecuación paramétrica de la recta $x = x_0 + (x_1 - x_0)t$
$y = y_0 + (y_1 - y_0)t$
$t \in (0,1)$

Los puntos de la ecuación general

$$(y_0 - y_1)x + (x_1 - x_0)y + (x_o y_1 - x_1 y_0) = 0$$

Puntos que cumplen esta ecuación, están en la línea.
Puntos que evalúan >0, caen a la izquierda.
Puntos que evalúan <0, caen a la derecha.

Problema de los campos potenciales
Por ejemplo un mínimo local



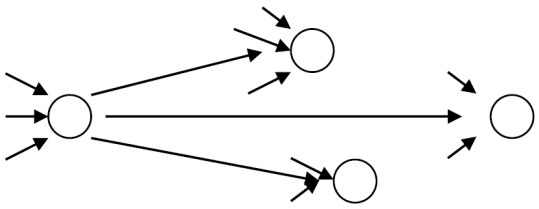
El vector resultante es 0 y el robot se queda inmóvil o en un ciclo.

3.9 Comportamientos con Redes Neuronales

Redes neuronales: 1943 → McCullock

Una red neuronal es un modelo matemático de cómo funciona una neurona.

Una neurona está conectada a otras neuronas.



La entrada a la red neuronal es una imagen (por ejemplo, la cámara del robot). Se puede entregar la información en bruto o procesada.

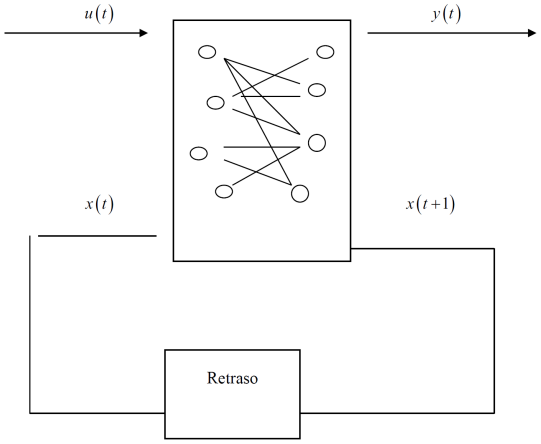
Las conexiones se van uniendo por medio de aprendizaje.

Se entrena a la red para que siga los comportamientos, por ejemplo un coche autónomo. Para entrenarlo, se puede poner a un conductor y hacer que la red aprenda los comportamientos de

acuerdo a las imágenes. La respuesta de la red es: para una cierta imagen, se giró a la izquierda violentamente; para otra imagen, se giró ligeramente; etcétera.

También se puede hacer con información de sonares, etc.

Puedo hacer máquinas de estado con redes neuronales.



Modelo

Por cada neurona se tiene

$$u = \sum_{i=1}^{n} w_i y_i + \theta$$

Entradas: $y_j \rightarrow$ vienen de otras neuronas Pesos: w_j Umbral de activación: θ

La salida de la neurona es la *función de activación*

$$a = f(u)$$

Generalmente $f(u) = \dfrac{1}{1+e^{\frac{-u}{T}}}$ (un sigmoide)

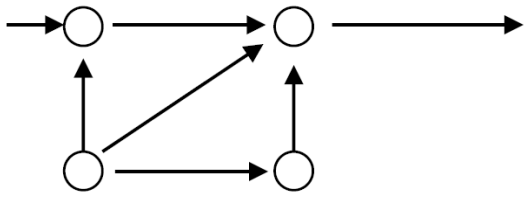
T es una constante de temperatura.

Resulta que f es la solución de la ecuación diferencial

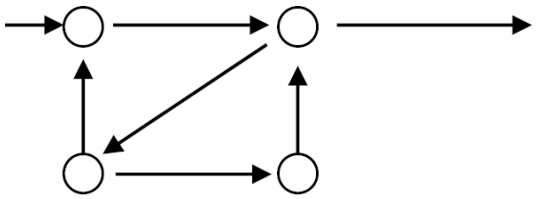
$$\frac{dy}{du} = \frac{y(1-y)}{T}$$

3.10 Topología de Redes Neuronales

¿Qué pasa con la topología de redes neuronales (muchas neuronas)?



Topología acíclica (no hay retroalimentación de la salida hacia otra interna) → el dígrafo es acíclico.



Topología cíclica (con retroalimentación) → el dígrafo es cíclico Estos fueron los primeros modelos de redes neuronales.

3.11 Modelo del Perceptrón

$u(x) = \sum_{i=1}^{n} w_i x_i + w_0$ y

$$y(x) = \begin{cases} 1, & u(x) \geq 0 \\ 0, & u(x) < 0 \end{cases}$$

Este modelo es utilizado para *detección y clasificación*.

Si las características de la entrada x es cercano a w tal que su producto interno $x^T w$ es mayor que un umbral $-w_0$, entonces la salida es 1, indicando la detección de un objetivo.

Ejemplo 3.4

Dos neuronas, una que representa una mesa y otra una silla. Las dos tienen las mismas entradas, que son características invariantes a distancia y orientación. La salida debe de ser 1 para la de la silla y 0 para la de la mesa.

N Entrenar a una red neuronal es encontrar los pesos.

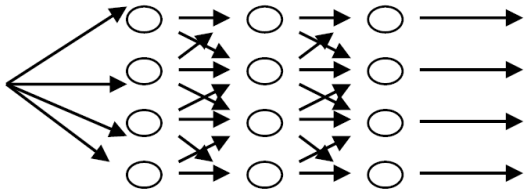
Minsky y Shannon

Shannon propone la matemática booleana y notación binaria en las computadoras. Wiener inventó el término *cibernética* (el estudio de los sistemas – eléctricos, mecánicos, sociales, etc. – y su control).

LEER ASIMOV Y PHILIP DICK

3.12 Perceptrón multicapas

Capa de entradas Capa intermedia Capa de salida



La capa de entrada tiene M neuronas. La capa intermedia se conoce como *capa oculta* y tiene H neuronas. La capa de salida tiene N neuronas.

El objetivo es encontrar los pesos que minimicen el error cuadrático medio.

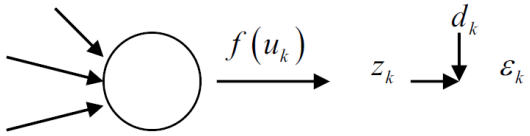
Cada salida de las neuronas de salidas es un objetivo de clasificación.

Ejemplo 3.5

¿Cómo encontrar los pesos?
Tres entradas, una neurona de entrada.
$u = Wx$ donde $x = [1 \quad x_1 \quad x_2]^T$ y $W = [W_0 \quad w_1 \quad W_2]$

Encontrar unos pesos que dado un objetivo d_k e vaya actualizando con la característica que $\varepsilon \to 0$. Se tendrán K muestras de entrenamiento (x_k, d_k); salidas z_k y un error total $E = \sum_{i=1}^{K} \varepsilon_k^2$, donde $\varepsilon = d_k - z_k$.

$d_k \in (0,1)$ generalmente.



Obsérvese que $z_k = f(Wx_k)$.

El objetivo final es minimizar la función E respecto a W.

La solución es no lineal si f es una función sigmoide, por lo que generalmente se realiza la minimización con iteraciones:

$W_{t+1} = W_t + \Delta W_t$

Por método de *steepest descent:* $\Delta W_t = -\eta \dfrac{dE}{dW}$

Solución

$$\frac{dE}{dW_i} = -2\sum_{k=1}^{K}(d_k - z_k)(\frac{-dZ_k}{dw_i})$$

Obsérvese que

$$\frac{-dz_k}{dw_i} = \frac{df(u)}{dW_i} = \frac{df(u)}{du}\frac{du}{dw_i} = f'(u)\frac{du}{dw_i} = f'(u)x_i$$

Por lo que

$$\frac{dE}{dW_i} = 2\sum_{k=1}^{K}(d_k - z_k)f'(u)x_i$$

Se denota $\delta_k = [d_k - x_k]f'(u_k)$ y entonces

$$w_i(t+1) = w_i(t) + \eta\sum_{i=1}^{K}\delta_k x_i(k)$$

En particular si $f(u)$ es el sigmoide, entonces

$\delta_k = [d_k - z_k]\, z_k\,(1 - z_k)\,\alpha := \dfrac{\partial E}{\partial u}$

$$\frac{\partial E}{\partial u} = -2\sum_{i=1}^{K}[d_k - z_k]\frac{dZ_k}{du}$$

y ademas

$$\frac{dz_k}{du} = \frac{f(u)[1 - f(u)]}{T}$$

.

Por lo tanto,

$$\frac{\partial E}{\partial u} = -2\sum_{i=1}^{K}[d_k - z_k]f(u)[1 - f(u)] = \frac{-2}{T} - 2\sum_{i=1}^{K}[d_k - z_k]\, z_k\,[1 - z_k].$$

QED

El proceso se debe hacer hasta que $E < \varepsilon$

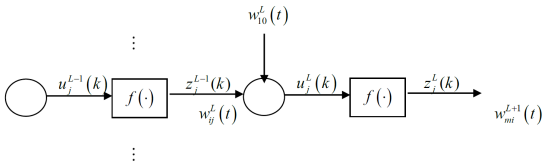
3.13 Cómo extender para un perceptrón multicapas

Se tienen:

K muestras de entrenamiento L capas

W_{ij}^{L} es el peso de la neurona i a la neurona j en la capa L y en la iteración t.

Propagación de error:



El objetivo es (de nuevo) encontrar $w_i j$ de cada capa que minimicen el error cuadrático medio.

Pero dado que

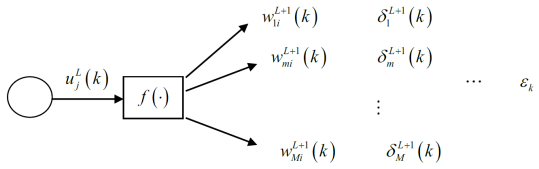
$$\frac{du_i^L(k)}{dw_{ij}^L} = \frac{d}{dw_{ij}^L} \sum_{m=1}^{M} \; w_{im}^L \; z_m^{L-1}(k)$$

Se puede reescribir la parcial como:

$$\frac{dE}{dw_{ij}^L} = -2 \sum_{k=1}^{K} \delta_i^L(k) z_j^{L-1}(k)$$

Se observa que el error δ_i^L en esta capa depende del error en la capa anterior, por lo que se necesita utilizar una derivación tipo Bellman para tener:

$$\delta_i^L(k) = \frac{\partial E}{\partial u_i^L(k)}$$



Obsérvese que tiene la siguiente recursión:

$$\begin{aligned}
\delta_i^L(k) = \frac{\partial E}{\partial u_i^L}(k) &= \sum_{m=1}^{M} \frac{dE}{du_m^{L+1}} \frac{du_m^{L+1}(k)}{\partial u_i^L(k)} \\
&= \sum_{m=1}^{M} \delta_m^{L+1}(k) \left[\frac{d}{\partial u_i^L(k)} \frac{j=1}{J} w_{mj}^L f(u_j^L(k)) \right] \\
&= f'(u_i^L(k)) \sum_{m=1}^{M} \delta_m^{L+1}(k) w_{mj}^L
\end{aligned}$$

Esto se conoce como *backward propagation*.

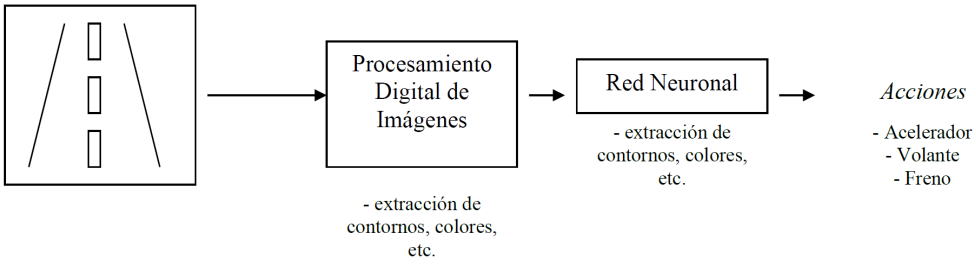
La fórmula de actualización será entonces:

$$w_{ij}^{L}(t+1) = w_{ij}^{L}(t) + \eta \sum_{k=1}^{K} \delta_i^L(k)\, z_j^{L-1}(k) + \underbrace{\mu\left[w_{ij}^{L}(t) - w_{ij}^{L}(t-1)\right]}_{\text{momento}} + \underbrace{\varepsilon_{ij}^{L}(t)}_{\text{ruido aleatorio}}$$

La η da la velocidad de convergencia. Sin embargo, si es muy grande, se puede dar la situación que se oscile demasiadas veces antes de llegar al mínimo. El error aleatorio se suma para encontrar el mínimo global.

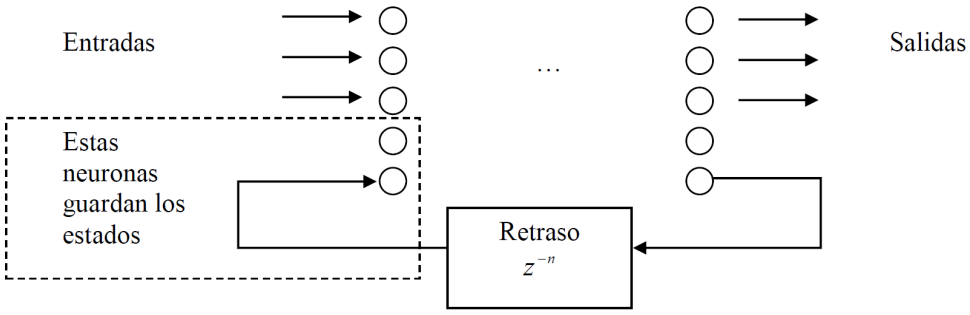
¿Cómo se usa esto para los robots?

En Carnegie-Mellon University, se realizó un experimento en el cual un robot pudo manejar 90 % autónomamente en una autopista larga de Estados Unidos.



El entrenamiento se haría primero manejando un humano y la red registrando qué se hizo y la foto correspondiente. Otra forma: máquina de estado con redes neuronales.

Máquina de Estados con Redes Neuronales



3.14 Aprendizaje

El aprendizaje en un robot puede ser varias cosas:

1. Aprender "audacia": esto depende de las constantes (afinación de constantes) Para campos potenciales: $\mu, \delta, d_1, d_0, \eta$.

2. Aprender nuevos comportamientos

a) Crear nuevos algoritmos de máquinas de estado.

b) Modificación de las máquinas de estado

Hay dos cosas: una, lo que aprendemos, y otro, el conocimiento que se obtiene a través de la evolución. Por ejemplo, un bebé sabe cómo evitar un obstáculo, no tiene que aprender.

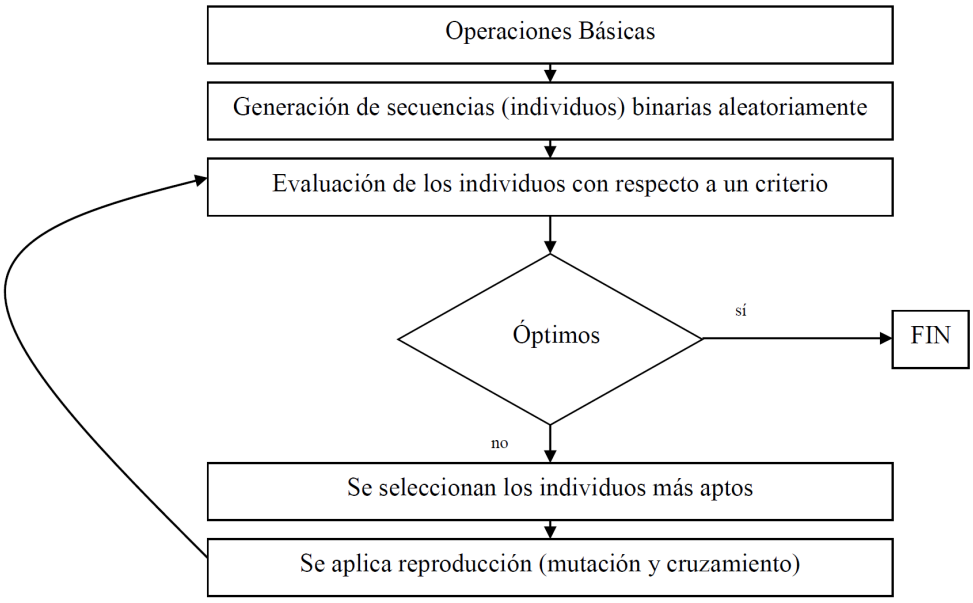
Adquisición de Comportamientos Básicos (evolución)

ALGORITMOS GENÉTICOS Los algoritmos genéticos nos permitirán utilizar el concepto de evolución para: - Afinación de constantes - Generación de algoritmos de máquina de estado.

Adquisición de Conocimiento Nuevo a partir de Comportamientos Básicos (heredados)

REDES NEURONALES Aprendizaje por medio de ejemplos (tú le enseñas qué debe de hacer en determinadas circunstancias).

¿Qué es un algoritmo genético?



Las operaciones básicas sirven para evaluar a los individuos, es decir se genera una medida para evaluar a cada secuencia de bits.

Individuo 1	1110 1111 1000 0111 0001
Individuo 2	0001 1001 1000 1111 0000
.	
.	
.	
Individuo N	0110 0001 1110 0010 1111

Ahora suponer que se cruza el individuo 1 con el 2 y tiene dos descendientes:

Individuo 1 $\otimes$ Individuo 2	1110 **1001** 1000 0111 0001
	0001 **1111** 1000 1111 0000

Para hacerlo "realista" se tienen que tomar partecitas (bloques de bits) del mismo lugar (por ejemplo del segundo bloque).

Mutación:

Individuo 1	1110 1111 **0111** 0111 0001

1. Afinación de Constantes en Campos Potenciales

Se tiene un **cromosoma** que codifica las constantes de un robot que navega usando campos potenciales.

μ	δ	d_1	d_0	η

Por ejemplo si $\mu = 3{,}5$ y se quiere tener cuatro bits para la parte entera y cuatro para la decimal:

00111000 Pues es $(0)2^3 + (0)2^2 + (1)2^0 + (1)2^{-1}$

(verificar que esto es cierto)

Si se tienen n individuos y se evalúan en un medio ambiente, la función podría ser:

$$f(ind_i) = \frac{k_1}{d(x_*, f_f)} + \frac{k_2}{pasos} + \ldots + k_n choques$$

En cada generación se cambia el origen y el destino.

En la última generación ya se tienen las constantes adecuadas.

2. Generación de algoritmos de máquinas de estado

Cromosoma que codifica la máquina de estados.

# de estados	Estado 1	Estado 2	…	Estado N

Cada estado, a su vez, es un conjunto de variables de salida y decisiones (condicionales – "if"s) con las variables de entradas.

Por ejemplo, el estado 1 se codificaría de la siguiente manera:

# de Variables de Salida	Variable 1	Valor	Variable 2	Valor	…	Variable N	Valor	# Variables Condicionales	Variable a sensar 1	Cota inferior	Cota superior	Transición (estado receptor)

...sigue...

Variable a sensar 2	Cota inferior	Cota superior	Transición (estado receptor)	…	Variable a sensar M	Cota inferior	Cota superior	Transición (estado receptor)	Transición si falso

Ejemplo 3.6

Codificar el estado (1) en:
$(1)Var5 = 7 \rightarrow (2)$

La solución es el binario de:

1	5	7	0	2

Es decir:

001	101	111	000	010

Ojo: necesito una máquina que realmente evalúe este cromosoma y lo ejecute como una máquina de estados.

Ejemplo 3.7

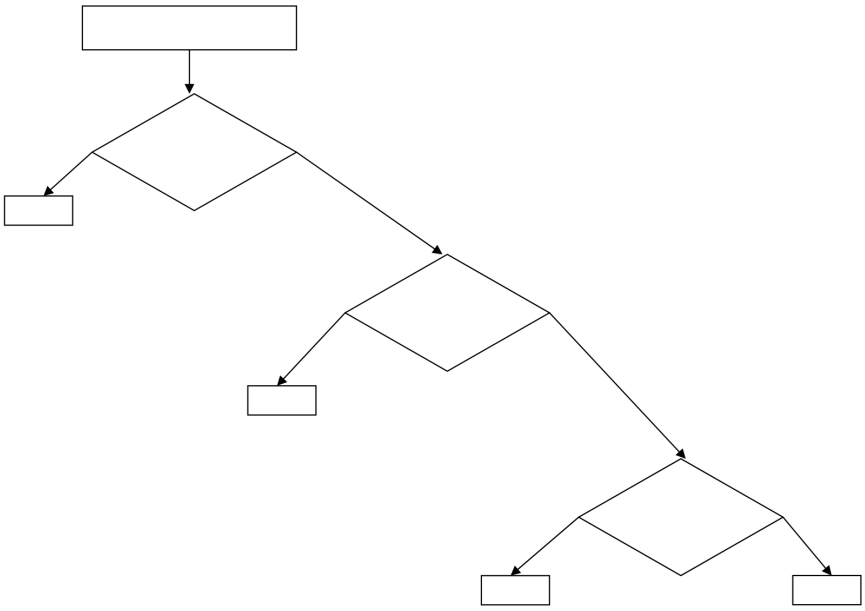
Codificar la máquina:
$(1)var1 = 3, var4 = 8 \rightarrow 10 < S3 < 25?(2)|(18)$

Solución: el binario de:

3	2	1	3	4	8	1	3	10	25	2	18

Ejemplo 3.8

Codificar lo siguiente:



N LO IMPORTANTE ES QUE EL CROMOSOMA SE PUEDA EJECUTAR. Por ejemplo, si hay condicionales de ambos lados no se puede con esta estructura.

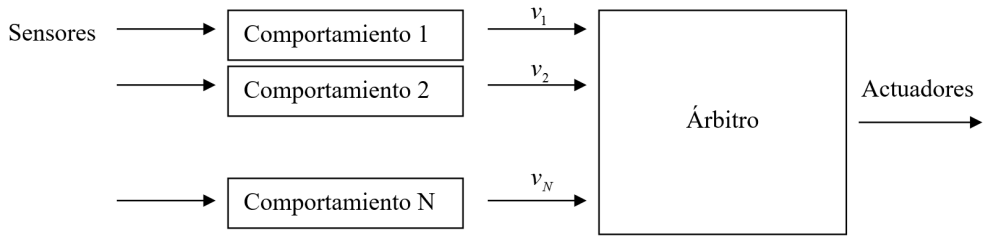
N Al hacer mutaciones o cruzamientos, hay que tener cuidado. Por ejemplo si se cruzan dos individuos de dos estados, y me da uno de un estado, habría que cortar el hijo para que la información posterior nada más tenga un estado. Igual si cambia a tres, añadir un estado. O cruzar condicionalmente... por ejemplo si cruzo el número de estados, podría tener que cruzar también la información de los estados etcétera.

Ejemplo 3.9

Codificar

$$
\begin{aligned}
&(1)M1 = 0.5; M2 = 0.3 \rightarrow 1.0 < S3 < 1.3? \\
&\quad T : (2)M1 = 0.1, M2 = 0.43.1 < S2 < 4.1 \\
&\quad\quad T : (2) \\
&\quad\quad F : 2.3 < S5 < 4? \\
&\quad\quad\quad T : (3)M1 = 0.7, M2 = 0.7 \rightarrow 6 < S2 < 7.0? \\
&\quad\quad\quad\quad T : (3) \\
&\quad\quad\quad\quad F : 2.1 < S3 < 4.0? \\
&\quad\quad\quad\quad\quad T : (3) \\
&\quad\quad\quad\quad\quad F : (1) \\
&\quad\quad\quad F : (2) \\
&\quad F : 3.1 < S1 < 4.7? \\
&\quad\quad T : (3) \\
&\quad\quad F : 1.1 < S2 < 1.2? \\
&\quad\quad\quad T : (2) \\
&\quad\quad\quad F : (1)
\end{aligned}
\tag{3.1}
$$

3.15 Utilización de algoritmos genéticos para organizar los comportamientos (arbitraje)



Recuérdese que

$$c_f = \sum_{i=1}^{N} g_i \mathbf{b}_i$$

donde g_i son las constantes de ganancia y $\mathbf{b}_i$ son los vectores de movimiento por los diferentes comportamientos.

Además se tiene una función de desempeño global que indica si el robot cumplió el objetivo.

- Una forma es encontrar las constantes g i con algoritmos genéticos, basándose en la función de desempeño global.

- Otra forma de organizar es poniendo prioridades.

3.15.1 Método de Utilidad Múltiple (Utility Manifold)

Se quiere que el árbitro decida cuál de los comportamientos se debe activar. Todos los comportamientos dan salidas siempre. Cada comportamiento dirá su aplicabilidad, dependiendo de:

- los sensores internos
- los sensores externos

Esto es un valor en (0,1)

1. Función de activación del comportamiento

$f_{H_i}(S_I, S_E, V)$

donde

S_I son los sensores internos (batería, odómetro, etc.) S_E son los sensores externos (sonares, infrarrojo, etc.) V son variables de estado $x_1, ..., x_r$

Se modela:

$f_{B_i}(S_E, S_I, x) = a_{i0} + a_{i1}S_E + a_{i2}S_I + a_{i3}x_i + a_{i4}S_E^2 + a_{i5}S_I^2 + a_{i6}x_i^2 + a_{i7}S_E S_I + a_{i8}S_I x_i + a_{i9}S_E x_i$,
para $i = 1, \ldots, n$ los n comportamientos

Se quiere obtener a_{ij} con algoritmos genéticos.

Se definen:

$$x_i = \begin{cases} b_{i1} + b_{i2}\, e^{-|b_{i3}|t_i}, & B_i \text{ está activo} \\ 0, & e.o.c. \end{cases}$$

Para cada comportamiento se tiene una función de activación. Se tienen diferentes individuos y, dentro del cromosoma, se tiene:

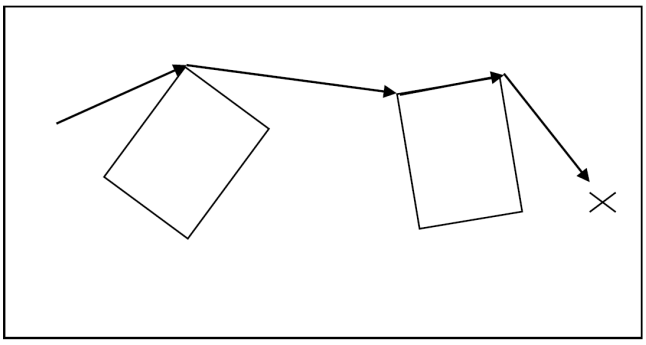
a_{i0}	a_{i1}	$\ldots$	a_{i9}	b_{i1}	b_{i2}	b_{i3}

(en binario)

Resumen Los algoritmos genéticos se utilizan para introducir a los robots los comportamientos básicos. Esto NO corre en línea, es para “alambrar” los “instintos” al robot.

3.16 Planeación

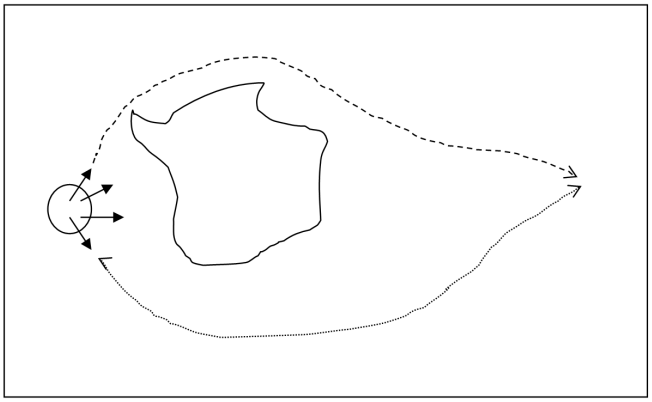
¿Cómo planear la ruta de un punto a otro?



(en binario)

Se quisiera encontrar camino más rápido.

Supóngase que se tiene un campo potencial y un obstáculo ¿Cómo se podría ajustar los campos potenciales?:



Una idea podría ser tratar de encontrar *varios* caminos por campos potenciales, dependiendo de un cierto ángulo de desvío θ.

Esto genera un árbol de opciones (caminos) que es muy costoso recorrer con fuerza bruta. ¿Cómo recorremos este árbol para que sea inteligente?

El objetivo es encontrar una mejor ruta para recorrerla.

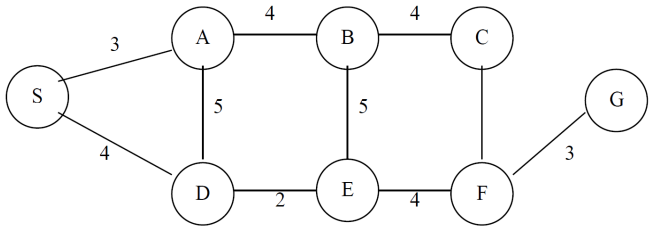
3.16.1 Búsqueda de la mejor ruta

El problema de búsqueda se puede formular como sigue:

Dado
x_0 un punto inicial (o nodo) x_f un punto objetivo (o nodo) un grafo con pesos (red topológica)
se quiere encontrar una ruta óptima que llegue al punto objetivo desde el punto inicial y recorrerla.

Ejemplo **3.10**

Observe la siguiente imagen



Depth-first

Funciona para árboles

Es necesario convertir la red en árbol. Se ponen únicamente los caminos únicos (no recursivos). Se ponen los pesos acumulados en cada nivel

Este approach se vuelve gigantesco, por lo que es necesario hacer un approach inteligente:

• Se va a realizar la búsqueda recorriendo el árbol por preorden hasta que sea nulo (en cuyo caso se saca de la lista y se mete otro hijo del padre) o sea el objetivo.

S
A | S
B | A | S
C | B | A | S
E | B | A | S
D | E | B | A | S
F | E | B | A | S
G | F | E | B | A | S

Aquí ya se encontró el destino por lo que se encontró un camino de nodos. Ésta es la forma de encontrar un camino.

La aplicación es:

1. El planeador encuentra los nodos del grafo que hay que visitar
2. Cada uno se convierte en un punto de atracción sucesivo en el algoritmo de campos potenciales

Hill Climbing

Tiene una función heurística. Generalmente la función del costo (o peso) al nodo más la distancia remanente al destino

$f(d(q_j, q_F), w_{ij})$

Es el costo de pasar de un nodo a otro más (por ejemplo) la distancia euclideana en R 3 del punto físico asociado al nodo y el punto físico asociado al destino.

Breadth-Search

Se va guardando en una lista ligada En cada iteración se van cargando los hijos de cada uno de los nodos de cada nivel.

Planeación usando espacio-estado

Reglas nombre Precondiciones → Operadores Lista de Borrado Lista Aditiva

Por ejemplo: Ponerse los zapatos:

Regla Colocar_Calcetin_Derecho Precondiciones

Colocar Zapato Derecho (comando)

Pie derecho está desnudo

Calcetín está disponible

→

Operador

Poner_Calcetin_PieDerecho

Lista de Borrado

Calcetín está disponible

Pie derecho está desnudo

Regla Colocar_Zapato_Derecho { Precondiciones

No hay pie derecho desnudo

Colocar Zapato Derecho (comando)

Zapato derecho disponible

→

Operador

Poner zapato derecho

Lista de borrado

Zapato derecho disponible

Colocar zapato derecho }

Se pueden organizar jerárquicamente:

Inicio

a) Colocar_Calcetin_derecho
a. Colocar_zapato_derecho
i. Colocar_calcetin_izquierdo
ii. Colocar_zapato_izquierdo

b. Colocar_calcetin_izquierdo
i. Colocar_zapato_izquierdo
1. Colocar_zapato_derecho
ii. Colocar zapato derecho
1. Colocar zapato_izquierdo

b) Colocar_Calcetin_izquierdo
a. Colocar zapato izquierdo
i. Colocar calcetín derecho
ii. Colocar zapato derecho

b. Colocar calcetín derecho
i. Colocar zapato derecho
1. Colocar zapato izquierdo
ii. Colocar zapato izquierdo

1. Colocar zapato derecho

¿Por cuál camino irnos? Debemos ponerle pesos y después trabajarlo con un algoritmo de búsqueda.

Ejemplo 3.11

Bloques
B A C E D Se quiere tener una representación espacio-estado con los operadores.
B D C E A

Operadores:

Goto(x,y,z) Ir a (x,y,z)
Pickup(w) Recoger w de la mesa
Putdown(w) Soltar w en la mesa
Stack(u,v) Poner el u encima del v
Unstack(u,v) Quita el bloque u del v (y se lo queda en la mano)

La representación del estado del mundo:

Location(w,x,y,z) el bloque w está en las coordenadas x,y,z
On(x,y) el bloque x está encima de y

Clear(x) no hay algo encima de x
Gripping(x) el brazo del robot tiene el bloque x
(*gripping(void)* es que no tiene algo)
OnTable(w) el bloque w está en la mesa

A partir de esto se deduce que

$$\forall x(CLEAR(x)) \Rightarrow \neg\exists y(on(y,x)))$$

$$\forall x(GRIPPING(void) \Leftrightarrow \neg GRIPPING(x)$$

$$\forall y \forall x(ONTABLE(y) \Rightarrow \neg ON(y,x))$$

Estado 1

OnTable(A)
OnTable(C)
OnTable(D)
On(B,A)
On(E,D)
Gripping(void)
Clear(B)
Clear(C)
Clear(E)

Regla Pickup(x) { Precondiciones
Clear(x)
OnTable(x)
Gripping(void)
$\rightarrow$ Lista de adición
Gripping(x)
Lista de borrado
OnTable(x)
Gripping(void)
}

Observación: esto genera una *notación de predicados*

$$\forall x(PICKUP(x) \Rightarrow GRIPPING(x)) \leftarrow (GRIPPING(void) \hat{} CLEAR(x) \hat{} ONTABLE(x))$$

Regla PutDown(x) {
Precondiciones
Gripping(x)
$\rightarrow$ Lista Adición
Gripping(void)
OnTable(x)

Lista de Borrado
Gripping(x)
}

Regla Stack(x,y) {
Precondiciones
Gripping(x)
Clear(y)
→ Lista de adición
Gripping(void)
On(x,y)
Lista de Borrado
Gripping(x)
Clear(y)
}

Árbol de Estados

B A C E D

1) B A E C D

2)

A E C D

Etcétera (todas las posibilidades)

Tercera evaluación

Comportamiento de potenciales Comportamiento de máquinas de estado Árbitro

(con variables de aplicabilidad entre 0 y 1)

Máquina de Inferencia

Hechos + Reglas → Máquinas de Inferencia → Agenda Los hechos activan reglas. Las reglas pueden generar otros hechos que, a su vez, pueden activar nuevas reglas.

Por ejemplo esto puede evitar que en 100000 if's anidados se tenga que ejecutar el último. En lugar de que las reglas busquen los hechos que los activan, es al revés (puesto que los hechos generalmente no cambian).

Las reglas producen hechos NUEVOS.

El sistema **CLIPS** es un lenguaje de programación orientado a plantillas que permite definir sistemas expertos. Tiene una máquina de inferencia con encadenamiento hacia delante para eficiencia. Permite definir hechos, reglas, etc.

3.17 Lenguaje Natural

La idea es hablarle a un robot en lenguaje cotidiano y que nos "entienda". Entender es hacer un análisis semántico para obtener un significado y transformarlo en una acci ón.

Lo que yo le dije, que lo haga.

La definición operacional de **entender** es que el sistema realice las acciones que el usuario le pide. Entender algo es transformar lo que se pide en una **representación**, la cual ha sido escogida para que corresponda a un conjunto de acciones disponibles que pueden ser ejecutadas.

1. Señal de entrada
   Puede ser voz, imagen o combinación de ambas (o cualquier otra forma de entrada, por teclado). Voz: por medio de un micrófono que transforma una señal acústica en una señal eléctrica. Es necesario aplicar técnicas de procesamiento digital de señales para obtener las características de esa señal acústica. Existen algoritmos (como el de cuantización vectorial y sus variantes) para procesar e identificar las palabras.
2. Las palabras de entrada son revisadas para ver si ellas están agrupadas de acuerdo con reglas gramaticales, significando que ellas forman oraciones gramaticalmente correctas.
3. El significado de cada palabra y de la oración es asignado. Este paso es el más difícil de los tres.

Dependencia Conceptual

Primitivas Conceptuales

La representación de varias acciones es la misma. Por ejemplo: “Juan dio el libro a Pedro”, “Juan prestó el libro a Pedro”, “Juan regaló el libro a Pedro”, significan cosas diferentes, pero la acción de transferir el libro de una persona a otra es la misma.

Cada primitiva tiene:

a) Un actor (realiza el acto)

b) Un acto (realizado por el actor y hecho hacia un objeto)

c) Un objeto (ente sobre el que es realizada la acción)

d) Una dirección (en la cual el acto es realizado)

e) Un estado (en el cual el objeto se encuentra)

Las primitivas son:

PTRANS Es la transferencia física de un objeto.

(PTRANS (Actor NIL) (Objeto NIL) (FROM NIL) (TO NIL))

Por ejemplo

“Robot, lleva las flores al patio” Se representaría como:

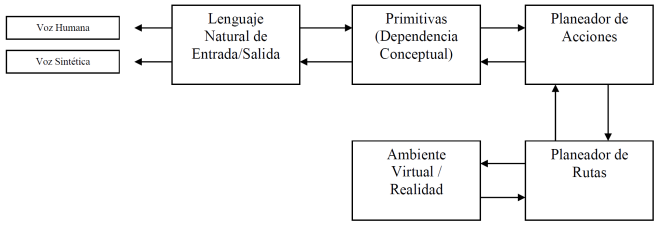
(PTRANS (Actor Robot) (Objeto flores) (From Posicion_flores) (To Patio)) “Robot, ve a la cocina” (PTRANS (Actor Robot) (Objeto Robot) (From Posicion_robot) (To Cocina))

Notar que el verbo indica qué primitiva utilizar

MTRANS es la transferencia de información mental. Roberto le dijo a Susana que era bonita (MTRANS (ACTOR Roberto) (OBJETO “que Susana es bonita”) (FROM cerebro_roberto)

(TO cerebro_susana) INGEST es la ingestión de un objeto por un animal, actor o ente Juan toma leche (INGEST (ACTOR Juan) (OBJECT Leche) (FROM NIL) (TO Boca_Juan)) PROPEL es la aplicación de fuerza física a un objeto María mató a una araña aventándole un zapato (PROPEL (ACTOR María) (OBJECT Zapato) (FROM Brazo_Maria) (TO Araña)) Observar: para matar a la araña con el PROPEL, primero tuvo que mover el brazo. MOVE es el movimiento de una parte del cuerpo

Voz → Lenguaje natural de entrada y salida → PRIMITIVAS (Dependencia Conceptual) → Planeador de acciones → Planeador de Rutas → Ambiente Virtual / Realidad



Proyecto Final

1)

1. Red topológica con nodos
2. Encontrar y ejecutar la mejor ruta
3. 6 cubos en un array de 2x3

2) Encontrar y ejecutar la mejor ruta 3) 6 cubos en un array de 2x3

A F |
B E |
C D |
Cuarto 1 cuarto 2

Poder ejecutar sentencias tipo “lleva A al cuarto 2” (por ejemplo el A no se puede agarrar, hay que quitar C y B y ponerlos en otro lado)

Con CLIPS Podemos poner nodos topológicos.

3.18 Creación de Mapas

La idea básica es crear un modelo del medio ambiente (una red topológica), basándose en la información en los sensores. Detectar con los sensores los puntos que delimitan el “espacio libre” (es decir, por donde puede navegar el robot).

3.18.1 Diagramas de Voronoi

La idea es encontrar el diagrama de Voronoi de los puntos y entones el robot puede caminar sobre las líneas.

Considérese un punto $(x,y) \in C$ (el espacio libre). Los puntos bases de (x,y) son los puntos (x',y') más cercanos en el espacio ocupado $\bar{C}$. El diagrama de Voronoi es el conjunto de puntos en el espacio libre que tiene cuando menos dos puntos bases diferentes a la misma distancia.

Puntos Críticos

Los puntos críticos (x,y) son puntos en el diagrama de Voronoi tales que minimizan un margen local. Existe $\varepsilon > 0$ tal que el margen de todos los puntos en la bola abierta de (x,y) no es menor.

Líneas críticas

Son obtenidas conectando cada punto crítico con sus puntos base. Las líneas críticas particionan el espacio libre en regiones disjuntas.

3.18.2 Cuantización Vectorial

Hacer clustering sobre el espacio libre.

Clustering:

Considérense m vectores, $v_i = (x_i, y_i)$.

1. Encontrar un centroide $c_1 = \frac{1}{m}\sum_{j=1}^{m} v_j$
2. Generar dos centroides nuevos a partir del centroide C_i anterior

$$c_{i+1} = c_i + \varepsilon_1$$

$$C_{i+2} = C_i \varepsilon_2$$

   Donde $\varepsilon_k = 1 + \delta_k$, con δ_k pequeño en magnitud absoluta

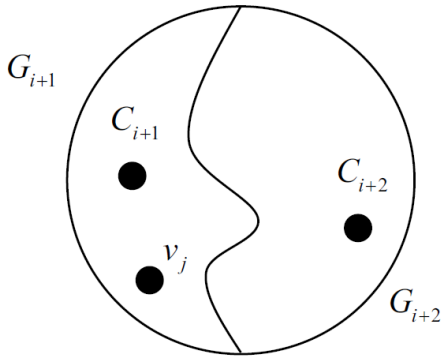
3. Agrupar los vectores en dos grupos de acuerdo al centroide más cercano.

$$d_T = d(v_j, c_{i+r}) r = 1,2$$

Asignar a los grupos:

$$v_j \in G_{i+1} \Leftrightarrow d_1 < d_2$$

$$v_j \in G_{i+2} \Leftrightarrow d_1 \leq d_2$$



Localización del Robot

Si el odómetro fuera perfecto, se puede saber siempre en dónde se encuentra el robot. Sin embargo, en la realidad no son exactos y no se sabe. Se puede utilizar la visión para localizar puntos de referencia predefinidos. Para hacer esto, se necesita que el sistema de visión sea invariante al tamaño y a transformaciones (rotaciones). Ya sea con un sistema de visión o de localización, se puede hacer triangulación. Si se tienen varias marcas, suponiendo que se pueden encontrar las distancias a las marcas, se puede trazar un círculo de radio la distancia. Se establece entonces:

$$(z-z_i)^2+(z-z_i)^2+(z-z_i)^2=d_i^2 \quad ,\forall_i$$

El sistema de ecuaciones me determina (x,y,z), es decir, la posición del robot.

Esto podría funcionar pero los sensores tienen errores y por lo tanto nunca se intersecan los círculos. Por lo tanto, la localización del robot es probabilística

4. Si $D_g^t - D_g^{t-r} > \varepsilon$ entonces recalcular los centroides

$$c_{i+r}=\frac{1}{m_{G_{i+r}}}=\sum_{j=1}^{m_{G_{i+r}}} v_j, \;\; r=1,2$$

donde $D_g^t=\sum_{j=1}^{t} min(d_1(j),d_2(j))$ y t es la iteración

Si es mayor, ir al punto 3. Si es menor:

Repetir 2 hasta que se tenga el número de centroides deseado.

3.18.3 Cadenas de Markov y Modelos Ocultos de Cadenas de Markov

Una cadena de Markov es como una máquina de estado en la que las transiciones son probabilísticas.

$$a_{ij}=P[pasardeiaj]$$

Los cambios de estado están indexados al tiempo y se denominan q_t.

Un proceso estocástico es un **proceso markoviano de primer orden** si la probabilidad de que la cadena de Markov se encuentre en un determinado estado j es:

$$P[q_t = j \mid q_{t-1} = i_1,\ q_{t-2} = i_2, \ldots, q_0 = i_t] = P[q_t = j | q_{t-1} = i]$$

Se tiene:

$$a_{ij} > 0 \forall i, j$$

$$\sum_{j=1}^{N} a_{ij} = 1 \forall i$$

<falta una clase aquí>

Problema 1
$O = (O_1, O_2, \ldots, O_N)$
$\lambda = (A, B, \Pi)$
$P(O|\lambda)$

Con lo visto anteriormente, son aproximadamente $(2T-1)N^T + N^T - 1$ operaciones. Por ejemplo, con 5 estados y 100 observaciones, esto es aproximadamente 10^{72} operaciones. Por tanto, necesitamos obtener algún procedimiento más eficiente.

Procedimiento hacia delante

$\alpha_t(i) = P(O_1, O_2, \ldots,\ O_t, q_t = i|\lambda)$

Es la probabilidad de observar la secuencia parcial $O_1, O_2, \ldots, O_t$ y el estado i en el tiempo t.

Esta probabilidad se puede encontrar de manera inductiva:

1. Inicialización

   $\alpha_1(i) = \Pi_i b_i(O_1), 1 \leq i \leq N$

   O_i es la observación al tiempo i.

   Por ejemplo, se puede tener: $O_1 = v_4$ que es una mesa. Las probabilidades son fijas pero las observaciones están cambiando respecto al tiempo.

2. Inducción

   $$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1})\ 1 \leq t \leq T-1,\ 1 \leq j \leq N$$

   La probabilidad depende de la probabilidad anterior, las probabilidades de transición y la probabilidad de la observación en el estado j.

3. Término

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) = \sum_{k=1}^{N} P(O_1, O_2, \ldots, O_t, q_t = i|\lambda)$$

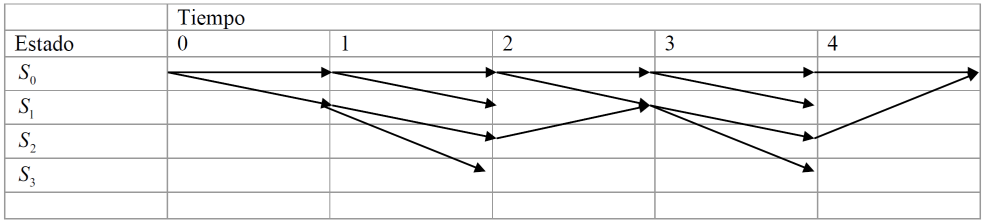
Para calcular $\alpha_t(j) 1 \leq t \leq T,\ 1 \leq j \leq N$ se requieren N^2T cálculos (en lugar de $2TN^T$)

Problema 2
Encontrar la mejor secuencia de estados $q = (q_1, q_2, \ldots, q_T)$dada la secuencia de observaciones $O = (O_1, O_2, \ldots, O_T)$ En términos del robot: dado lo que vio, en dónde se estuvo moviendo.

Diagrama de Trellis

Diagrama de Trellis

Dado una máquina de estados con transiciones:



Para cada estado hay una serie de caminos, hay que encontrar el mejor camino. Sin embargo, para cada tiempo, la búsqueda crece en forma exponencial (el número de caminos crece).

Hay que hacerlo con el **Algoritmo de Viterbi**, que va eliminando todos los caminos menos los más probables.

3.19 Algoritmo de Viterbi

Sea $\delta_t(i)$la mejor ruta en el tiempo t que toma en cuenta las primeras t observaciones y termina en el estado i.

$$\delta_t(i) = \max_{q_1,\ldots,q_{t1}} P[q_1 q_2 \ldots q_{t-1}, q_t = i, O_1, O_2, \ldots, O_t|\lambda]$$

1) Inicialización

$\delta_i(i) = \pi_i b_i(O_1)$ para toda i

2) Recursión

$$\delta_t(j) = \left(\max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] \right) b_j(O_t)$$

$2 \leq t \leq N, 1 \leq j \leq N$

$\Psi_t(j) = \underset{1 \leq i \leq N}{argmax} \left[\delta_{t-1(i)a_{ij}} \right]$ (es decir, los nodos que conforman la ruta)

3) Término

$$p^* = \max_{1 \le i \le N} [\delta_T(i)]$$

$$q_T^* = \operatorname*{argmax}_{1 \le i \le N} [\delta_T(i)]$$

4) Ruta

La ruta se obtiene:

$$q_t^* = \Psi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2, \ldots, 1$$

Problema 3
Estimación de Parámetros

Se quiere encontrar $\lambda = (A, B, \pi)$ que satisfaga un cierto criterio de optimalidad.

Modelo inicial $\rightarrow$ (de datos de entrenamiento) Segmentación secuencia de estados $\rightarrow$ Reestimación del modelo $\rightarrow$ Convergencia del modelo? Si sí, regresar a segmentación, si no, dar parámetros del modelo.

$b_j(k)$ es el número de observaciones con el símbolo k en el estado j dividido por el número de observaciones en el estado j.

a_{ij} es el número de transiciones del estado i al estado j dividido por el número de transiciones de i a cualquier estado. π_i es el número de veces que se empezó en el estado i dividido entre el número de veces de entrenamientos.