Universidad del Valle de Guatemala
Departamento de Ingeniería Mecatrónica
MT3006 - Robótica 2
MSc. Miguel Zea

Localización y mapeo en robótica móvil

En esta ocasión, dejaremos atrás a los manipuladores y la visión de computadora para considerar una serie de problemas canónicos en el área que denominaremos como robótica móvil de *consumer-grade*, debido que muchos de los algoritmos a considerar se encuentran actualmente operando en robots comerciales como el Roomba de iRobot o bien están muy cerca de ser adaptados a gran escala como las propuestas de carros autónomos de Google, Tesla, entre otras. Si bien este grado de certeza en los algoritmos implica que existen diversos recursos en software para su implementación, este no será el énfasis en este módulo sino más bien será desarrollar las destrezas necesarias para razonar y plantear correctamente este tipo de problemas. Por lo tanto, a pesar que llegaremos a implementar los algoritmos que desarrollemos, puede que para una aplicación en vida real existan soluciones más modernas o eficientes a los mismos problemas (cuando este sea el caso haremos una mención al nombre de los algoritmos más modernos al igual que se proveerá una referencia relevante que contenga más detalles al respecto).

Hasta el momento (y esta fue la norma también en los cursos de Control 1, Control 2 y Robótica 1) hemos tenido un interés significativamente mayor en resolver el problema de control no sólo para robots móviles sino que para cualquier tipo de robot/sistema general. Esto, sin embargo, nos presentará un desafío considerable al momento de querer implementar los algoritmos de control en un robot móvil real, ya que la mayoría de los controladores que hemos desarrollado asumen que el robot conoce perfectamente su estado (típicamente su posición y orientación). Desafortunadamente, este supuesto no puede alejarse más de la realidad ya que nuestros robots sólo tendrán a disposición las mediciones ruidosas e indirectas provistas por sus sensores a bordo. Por lo tanto, es necesario que consideremos el problema de sensado y estimación necesario para poder tomar la información provista por todos los sensores del robot y convertirla en una aproximación del estado actual del mismo, la cual ya podrá emplearse dentro de los algoritmos de control que hemos desarrollado. En este módulo entonces, identificaremos y desarrollaremos dos problemas canónicos de estimación en robótica móvil, el de localización y el de mapeo, al igual que consideraremos la combinación de ambos en lo que se conoce como el problema de mapeo y localización simultánea (SLAM, de *Simultaneous Localization and Mapping*).

Antes de definir y plantear la solución a los problemas de localización y mapeo, es importante hacer notar que estaremos trabajando bajo la perspectiva de observación/estimación de sistemas dinámicos con elementos estocásticos, por lo que conviene recordar la herramienta más importante que desarrollamos para este tipo de situaciones en el curso de Sistemas de Control 2: el filtro de Kalman (KF) y el filtro de Kalman Extendido (EKF). Recordemos que, si tenemos un sistema lineal como el de la Figura 1, entonces la ecuación de estado y la salida presentan la forma general

$$\dot{\mathbf{x}} = \mathbf{A}(t)\mathbf{x} + \mathbf{B}(t)\mathbf{u} + \mathbf{F}(t)\mathbf{w},$$

$$\mathbf{y} = \mathbf{C}(t)\mathbf{x} + \mathbf{v},$$

en donde $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_w(t))$, $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_v(t))$ corresponden al ruido de proceso y de observación respectivamente, y $\mathbf{F}(t)$ corresponde a una matriz que acopla al ruido con la dinámica del sistema (típicamente es igual a la identidad o a la matriz de actuadores $\mathbf{B}(t)$). Dentro de esta situación, el observador [1]:

$$\dot{\hat{\mathbf{x}}} = \mathbf{A}(t)\hat{\mathbf{x}} + \mathbf{B}(t)\mathbf{u} + \mathbf{L}^{\star}(t)\left(\mathbf{y} - \mathbf{C}(t)\hat{\mathbf{x}}\right),$$

con

$$\mathbf{L}^{\star}(t) = \mathbf{P}(t)\mathbf{C}^{\top}(t)\mathbf{Q}_v^{-1}(t)$$

y con $\mathbf{P}(t)$ como solución a la ecuación diferencial de Riccati (RE)

$$\dot{\mathbf{P}}(t) = \mathbf{A}(t)\mathbf{P}(t) + \mathbf{P}(t)\mathbf{A}^{\top}(t) - \mathbf{P}(t)\mathbf{C}^{\top}(t)\mathbf{Q}_v^{-1}\mathbf{C}(t)\mathbf{P}(t) + \mathbf{F}(t)\mathbf{Q}_w(t)\mathbf{F}^{\top}(t), \tag{1}$$

se conoce como el filtro de Kalman en tiempo continuo (cKF). Esta solución al problema de observación no sólo garantiza la estabilización de la media del error de estimación $\mathbf{e}(t) = \mathbf{x}(t) - \hat{\mathbf{x}}(t)$, sino que también minimiza su varianza $\mathbf{P}(t) = E\left\{(\mathbf{x} - \hat{\mathbf{x}})(\mathbf{x} - \hat{\mathbf{x}})^{\top}\right\}$.
La ventaja de este planteamiento es que nos permite adaptar el algoritmo a sistemas más simples y más complejos. Por ejemplo, si la dinámica del sistema es LTI y la varianza de los ruidos de proceso y observación no cambia, entonces la ecuación diferencial de Riccati se convierte en la ecuación algebráica de Riccati (ARE)

$$\mathbf{AP} + \mathbf{PA}^{\top} - \mathbf{PC}^{\top}\mathbf{Q}_w^{-1}\mathbf{CP} + \mathbf{FQ}_w\mathbf{F}^{\top} = \mathbf{0},$$

y el observador LTI resultante

$$\dot{\hat{\mathbf{x}}} = \mathbf{A}\hat{\mathbf{x}} + \mathbf{B}\mathbf{u} + \mathbf{L}^{\star}\left(\mathbf{y} - \mathbf{C}\hat{\mathbf{x}}\right),$$

con

$$\mathbf{L}^{\star} = \mathbf{PC}^{\top}\mathbf{Q}_v^{-1},$$

se conoce como el filtro de Kalman en estado estacionario (LQE) y es el análogo directo al regulador lineal cuadrático (LQR) pero para estimación en lugar de control. Por otro lado, si el sistema presenta dinámica no lineal

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{w}),$$

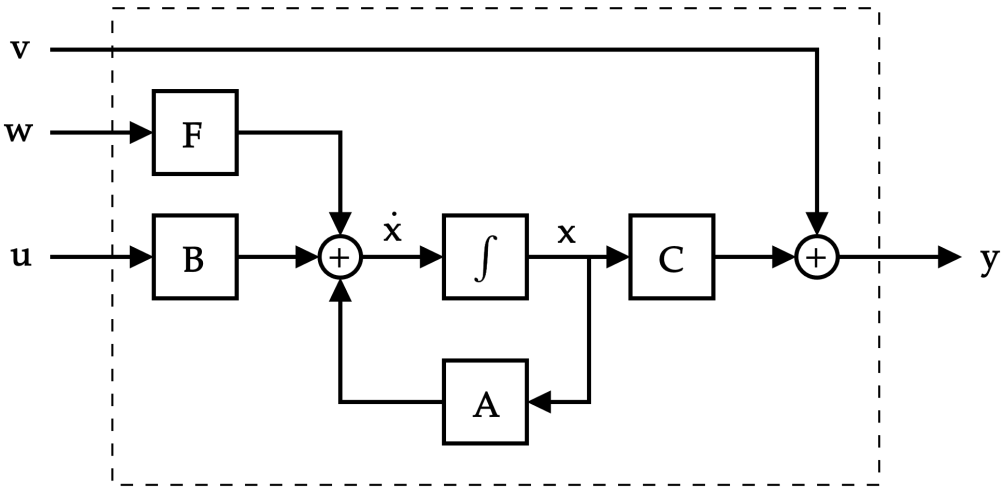

Figura 1: Sistema LTI con ruido de proceso y de observación.

$$\mathbf{y} = \mathbf{h}(\mathbf{x}) + \mathbf{v},$$

entonces podemos emplear linealización para obtener

$$\mathbf{A}(t) = \frac{\partial \mathbf{f}(\mathbf{x}^*(t), \mathbf{u}^*(t))}{\partial \mathbf{x}}, \qquad \mathbf{F}(t) = \frac{\partial \mathbf{f}(\mathbf{x}^*(t), \mathbf{u}^*(t))}{\partial \mathbf{w}}, \qquad \mathbf{C}(t) = \frac{\partial \mathbf{h}(\mathbf{x}^*(t))}{\partial \mathbf{x}},$$

y el observador de estado resultante

$$\dot{\hat{\mathbf{x}}} = \mathbf{f}\left(\hat{\mathbf{x}}, \mathbf{u}\right) + \mathbf{L}^\star(t)\left(\mathbf{y} - \mathbf{h}\left(\hat{\mathbf{x}}\right)\right),$$

con la misma ecuación diferencial de Riccati (1), se conoce como el filtro de Kalman extendido en tiempo continuo (cEKF). Este observador, a pesar de ya no ser óptimo en este caso, es uno de los estimadores más empleados para observación de sistemas no lineales, ya que presenta muy buenos resultados cuando se inicializa de forma correcta con

$$\hat{\mathbf{x}}(0) = E\left\{\mathbf{x}(0)\right\} = \mathbf{x}_0, \qquad \mathbf{P}(0) = E\left\{\mathbf{x}(0)\mathbf{x}(0)^\top\right\}.$$

Presentar al filtro de Kalman como un observador de estado fue lo más conveniente para nosotros en el curso de Sistemas de Control 2 ya que se acopló perfectamente a nuestra visión enfocada en sistemas dinámicos, sin embargo, en la práctica y en mucha de la literatura el filtro de Kalman se presenta típicamente de la forma en que fue concebido, como un filtro lineal óptimo en el sentido de mínimos cuadrados. Por esta razón, es más común ver el planteamiento del algoritmo en tiempo discreto. Bajo esta perspectiva, el filtro asume que se tiene un sistema dinámico lineal en tiempo discreto de la forma

$$\mathbf{x}_{k+1} = \mathbf{A}[k]\mathbf{x}_k + \mathbf{B}[k]\mathbf{u}_k + \mathbf{F}[k]\mathbf{w}_k,$$
$$\mathbf{y}_k = \mathbf{C}[k]\mathbf{x}_k + \mathbf{v}_k,$$

en donde $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_w[k])$ y $\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_v[k])$ son los ruidos de proceso y observación respectivamente. Es importante notar que estos ruidos no están correlacionados, es decir, $E\left\{\mathbf{w}_k\mathbf{v}_j^\top\right\} = 0$. A diferencia del filtro de Kalman en tiempo continuo, la variante discreta del filtro de Kalman típicamente se separa en dos etapas: *predicción* y *corrección*. Dentro de ambas etapas, se distinguen dos tipos de estimaciones, las que se hacen previo a emplear las mediciones de los sensores y las que se hacen después. Estos estimados reciben el nombre de *a-priori* y *a-posteriori* respectivamente, y se representan de la siguiente manera (empleando la notación de probabilidad condicional):

$$\hat{\mathbf{x}}_{k|k-1} \quad \text{(a-priori)}, \qquad \hat{\mathbf{x}}_{k|k} \quad \text{(a-posteriori)}.$$

Adicionalmente, también se asume que la condición inicial está siendo modelada por un vector aleatorio Gaussiano, es decir

$$E\left\{\mathbf{x}[k_0]\right\} = \mathbf{x}_0 \equiv \hat{\mathbf{x}}_{0|0}, \qquad \mathbf{P}_0 = E\left\{\mathbf{x}[k_0]\mathbf{x}^\top[k_0]\right\} \equiv \mathbf{P}_{0|0}.$$

Bajo esta nomenclatura entonces, se describe el algoritmo del filtro de Kalman (KF) separado en predicción y corrección en el diagrama de la Figura 2. En la literatura de estimación (en contraste a la de sistemas de control), se denomina al término $\mathbf{y}_k - \mathbf{C}[k]\hat{\mathbf{x}}_{k|k-1} \triangleq \mathbf{z}_k$ como la *innovación*, la cual representa la diferencia de las cantidades medidas con las estimadas empleando la información a-priori. Al emplear la innovación, se separa el cálculo de la *ganancia de Kalman* $\mathbf{L}_k$ en dos pasos:

$$\mathbf{S}_k = \mathbf{Q}_v[k] + \mathbf{C}[k]\mathbf{P}_{k|k-1}\mathbf{C}[k]^\top,$$

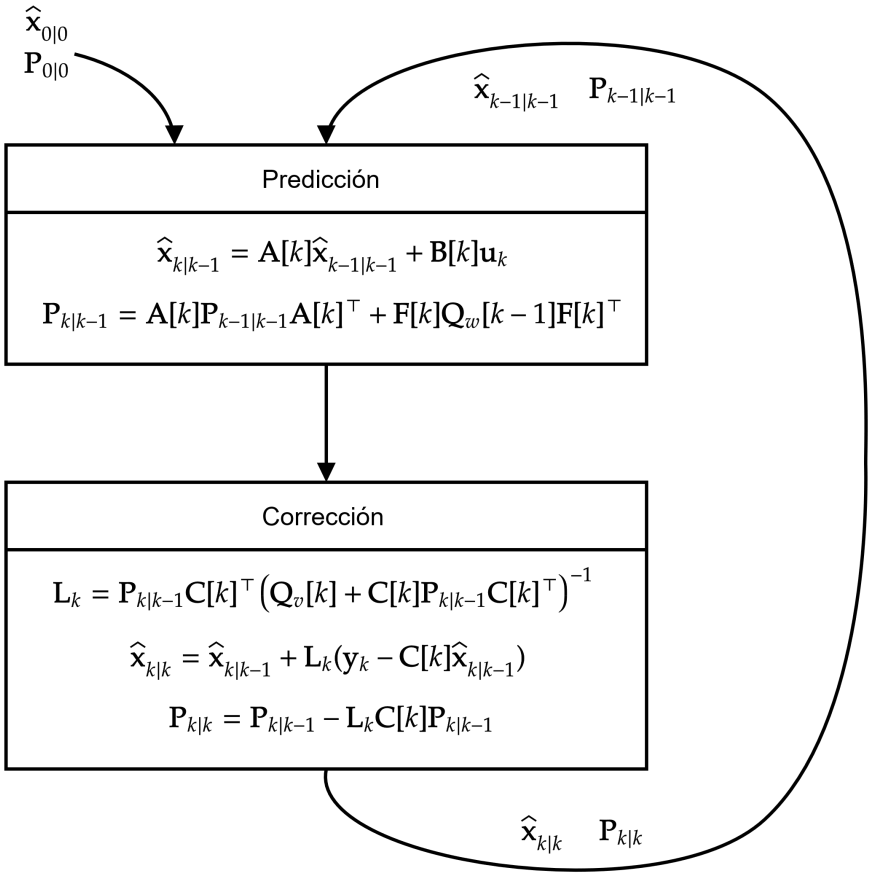

Figura 2: Algoritmo predicción-corrección para el filtro de Kalman discreto.

$$\mathbf{L}_k = \mathbf{P}_{k|k-1}\mathbf{C}[k]^\top \mathbf{S}_k^{-1},$$

en donde $\mathbf{S}_k$ es la matriz de covarianza de la innovación.

Para extender el algoritmo a sistemas discretos no lineales

$$\mathbf{x}_{k+1} = \boldsymbol{\mathcal{F}}\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k\right),$$

$$\mathbf{y}_k = \boldsymbol{\mathcal{H}}\left(\mathbf{x}_k\right) + \mathbf{v}_k,$$

basta con redefinir: la predicción de estado

$$\hat{\mathbf{x}}_{k|k-1} = \boldsymbol{\mathcal{F}}\left(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_{k-1}, \mathbf{0}\right),$$

la corrección de estado

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{L}_k\left(\mathbf{y}_k - \boldsymbol{\mathcal{H}}\left(\hat{\mathbf{x}}_{k|k-1}\right)\right),$$

y generar las matrices mediante los jacobianos de los mapas $\boldsymbol{\mathcal{F}}$ y $\boldsymbol{\mathcal{H}}$ de la forma

$$\mathbf{A}[k] = \frac{\partial \boldsymbol{\mathcal{F}}\left(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_{k-1}, \mathbf{0}\right)}{\partial \mathbf{x}}, \qquad \mathbf{B}[k] = \frac{\partial \boldsymbol{\mathcal{F}}\left(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_{k-1}, \mathbf{0}\right)}{\partial \mathbf{u}},$$

$$\mathbf{F}[k] = \frac{\partial \boldsymbol{\mathcal{F}}\left(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_{k-1}, \mathbf{0}\right)}{\partial \mathbf{w}}, \qquad \mathbf{C}[k] = \frac{\partial \boldsymbol{\mathcal{H}}\left(\hat{\mathbf{x}}_{k|k-1}\right)}{\partial \mathbf{x}},$$

Con estas modificaciones el algoritmo resultante se conoce como el filtro de Kalman extendido (EKF) y funcionará adecuadamente siempre y cuando su inicialización coincida lo mejor posible con las condiciones iniciales reales del sistema que genera las variables de estado a estimar. Una forma práctica (que aplica tanto para el KF como para el EKF) de inicialización es

$$\hat{\mathbf{x}}_{0|0} = \mathbf{x}_0, \qquad \mathbf{P}_{0|0} = \sigma_e^2 \mathbf{I},$$

en donde σ_e^2 representa la incertidumbre que tenemos con respecto al conocimiento de las condiciones iniciales reales (es decir, presenta un valor alto cuando no estamos seguros de las condiciones y un valor bajo cuando sí lo estamos).

El planteamiento del KF y el EKF probarán ser fundamentales para el resto del documento ya que los problemas de localización y mapeo se reducirán a la aplicación de ya sea un KF o un EKF.

Localización

La gran ventaja que los robots de base móvil presentan con respecto de sus contrapartes de base fija es que son capaces de cambiar su posición y orientación, relativa a un marco de referencia inercial, a voluntad. Esta ventaja, sin embargo, hace considerablemente más difícil la obtención del estado/-configuración del robot, algo que es muy sencillo de hacer para el caso de manipuladores de base fija. Esta discrepancia surge del hecho que la mayor parte de sensores relacionados a estimar la pose de un robot son *propioceptivos*, es decir, miden cantidades internas a los mismos. Esto causa que todas las mediciones de posición y orientación son relativas, lo cual no importa en caso que la base del robot se encuentre fija en el mundo pero es problemático en el caso móvil, en donde típicamente las tareas se codifican en coordenadas absolutas con respecto de cierto marco de referencia inercial. Otro problema es que si el robot sólo tiene acceso a mediciones internas a él, no tiene ninguna forma de verificar la validez de las mismas (con respecto del marco inercial) o de corregirlas en caso de ocurrir una situación imprevista o bien si los sensores presentan ruido significativo. Podemos pensar que desde la popularización de sensores *exteroceptivos*, que obtienen información del entorno externo al robot, como el GPS o la brújula digital (comúnmente encontrados dentro de cualquier smartphone moderno) obtener posiciones y orientaciones absolutas ya no es problema. Sin embargo, resulta que estos sensores no son suficientes ya que en muchos casos carecen de una precisión fina (los GPS comerciales por si solos tienen una precisión en el orden de metros), sus métodos de medición son poco robustos (los GPS deben comunicarse de forma inalámbrica con al menos cuatro satélites remotos, las brújulas digitales dependen de una medición débil del campo electromagnético de la Tierra), están sujetos a perturbaciones comunes (las mediciones de las brújulas digitales se ven perturbadas por la presencia de cualquier otro campo magnético adicional al de la Tierra) o bien no pueden emplearse en todos los ambientes de interés (los GPS no funcionan en interiores, bajo tierra o bajo de agua). Por lo tanto, lo que ha mostrado ser la mejor opción es la combinación de información obtenida con tanto sensores propioceptivos como exteroceptivos, mediante un proceso denominado *fusión de sensores*. El problema de localización en robótica móvil entonces se resume a uno de estimación de pose empleando fusión de sensores, específicamente, mediante el KF y el EKF.

En el resto del documento se emplearán cuatro marcos de referencia para representar las cantidades estimadas y las medidas por los sensores, estos son: $\{I\}$ el marco de referencia inercial o de la Tierra (se mantiene fijo y no toma en consideración la rotación de la misma, el marco que sí lo considera se denomina $\{E\}$ pero no lo emplearemos), $\{N\}$ el marco de navegación con respecto al cual nos interesa determinar la posición y orientación (este se considera fijo con respecto del inercial siempre

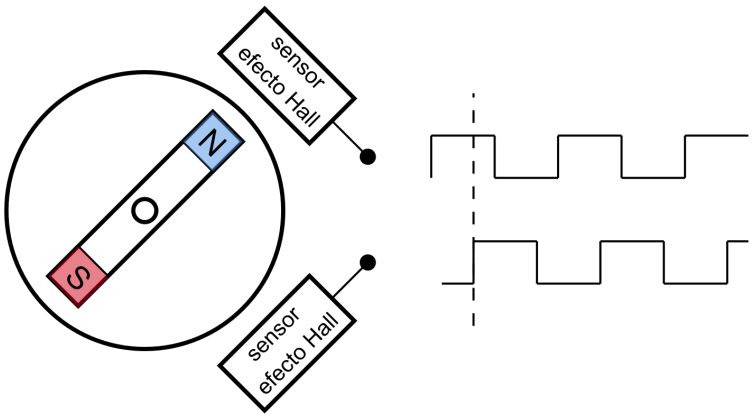

Figura 3: Funcionamiento básico de un encoder incremental magnético.

y cuando las distancias recorridas sean pequeñas con respecto al tamaño de la superficie terrestre), $\{S\}$ el marco del sensor y $\{B\}$ el marco de la base del robot móvil (que en muchos casos lo asumiremos idéntico a $\{S\}$ por simplicidad).

Estimando la posición

A pesar que existe una diversa cantidad de sensores para generar un estimado de la posición de un robot móvil, en esta sección nos enfocaremos en dos opciones populares para robots móviles con ruedas. Es importante notar que si bien la gran mayoría de propuestas de sensores y algoritmos funcionan para este tipo de robots, el caso contrario no es cierto en el sentido que muchas de las opciones disponibles para robots con ruedas no funcionan para robots con patas o robots aéreos y acuáticos.

Dead reckoning empleando odometría

El término *dead reckoning*, también conocido como integración de recorrido, se refiere al proceso de emplear la combinación de una posición previamente determinada junto a un estimado de la velocidad del robot para estimar la posición actual del mismo. En otras palabras, sin importar en dónde (dentro del marco inercial) empiece el robot su recorrido, podemos emplear mediciones de su velocidad para calcular la distancia recorrida con respecto a ese punto inicial desconocido.

En robótica móvil, un tipo de sensor que funciona bajo este principio son los llamados sensores de odometría, que emplean mediciones de movimiento para estimar la posición actual del robot con respecto a su posición inicial. Para robots con ruedas, estos sensores típicamente toman la forma de *encoders incrementales*, los cuales proveen un estimado de tanto la posición angular como la velocidad angular de los ejes de las ruedas. En la Figura 3, se ilustra el funcionamiento básico de un encoder incremental manético de dos polos, el cual usa un par de sensores de efecto Hall para detectar el paso de cada polo montado en un disco que gira junto al eje del motor (la mayoría de encoders se instala en el eje del motor *sin reducción*, ya que típicamente este eje realiza múltiples revoluciones por cada revolución del eje de la rueda). El desfase resultante entre las señales cuadradas resultantes provee un estimado de la velocidad del motor, mientras que cualquiera de las dos señales puede emplearse para estimar la posición angular del motor mediante un conteo de flancos.

De manera concreta, si asumimos un intervalo de tiempo corto Δt, las trayectorias que realizan las ruedas y el centro de un robot móvil diferencial pueden aproximarse como arcos, similar a la Figura 4. Bajo este supuesto, sabemos que la distancia recorrida en cada arco está dada por

$$D_L = r\theta_L = 2\pi r \frac{\Delta \text{tick}_L}{N}, \qquad D_R = r\theta_R = 2\pi r \frac{\Delta \text{tick}_R}{N},$$

$$D_c = \frac{D_R + D_L}{2} \equiv \delta_\rho,$$

en donde se asume que los encoders tienen N *ticks* o marcas/muescas/flancos por revolución (este número ya debe tomar en cuenta la relación de la caja reductora, por ejemplo, un encoder magnético de 8 polos con una reducción 8:1 presentará $N = 64$ marcas por revolución), y el cambio en los *ticks* $\Delta\text{tick} = \text{tick}_k - \text{tick}_{k-1}$ representa cuántas marcas se han contado desde el último período de muestreo. Recordemos que el estado de un robot diferencial está dado por ${}^N\boldsymbol{\xi} = {}^N\begin{bmatrix} x & y & \theta \end{bmatrix}^\top = \begin{bmatrix} \xi_1 & \xi_2 & \xi_3 \end{bmatrix}^\top$. Entonces, si conocemos la distancia recorrida por el centro δ_ρ en Δt segundos podemos propagar el estado del robot mediante

$${}^N\boldsymbol{\xi}_{k+1} = {}^N\boldsymbol{\xi}_k + \begin{bmatrix} \delta_\rho \cos \xi_{3,k} \\ \delta_\rho \sin \xi_{3,k} \\ \delta_\theta \end{bmatrix}, \qquad \delta_\theta = \frac{D_R - D_L}{\ell}.$$

Este modelo, sin embargo, no toma en cuenta el error inherente que presentan los estimados generados usando las mediciones de los encoders. El supuesto más grande bajo el que trabajan las expresiones que derivamos es que el suelo en donde se desplaza el robot es homogéneo y presenta un contacto perfecto con sus ruedas. De no ser este el caso, como en cualquier situación realista, los encoders presentarán *drift* y esto afectará a la pose estimada (piense en el caso en donde una rueda del robot se queda atorada en un agujero, esta continua girando pero el robot no cambia su pose). Para tomar estos errores en consideración se añade ruido blanco no correlacionado $w_\rho \sim \mathcal{N}(0, \sigma_{w_\rho})$, $w_\theta \sim \mathcal{N}(0, \sigma_{w_\theta})$ a ambos incrementos de desplazamiento lineal y angular para obtener el modelo de odometría del robot como [2]:

$$\underbrace{{}^N\boldsymbol{\xi}_{k+1}}_{\mathbf{x}_{k+1}} = \underbrace{\begin{bmatrix} \xi_{1,k} + (\delta_\rho + w_\rho)\cos \xi_{3,k} \\ \xi_{2,k} + (\delta_\rho + w_\rho)\sin \xi_{3,k} \\ \xi_{3,k} + \delta_\theta + w_\theta \end{bmatrix}}_{\boldsymbol{\mathcal{F}}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k)}, \tag{2}$$

el cual puede emplearse en la etapa de predicción de un EKF. Para poder estimar el estado correctamente, sin embargo, necesitamos la información de un sensor exteroceptivo como un GPS para completar el EKF y efectuar fusión de sensores.

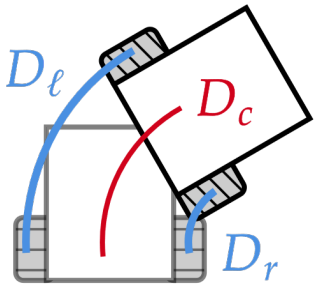

Figura 4: Arcos de desplazamiento para un robot móvil diferencial.

Dead reckoning empleando un acelerómetro

De las conclusiones más útiles en cálculo diferencial está el hecho que podemos encontrar la velocidad y la aceleración de una partícula como la primera y segunda derivada de la posición respectivamente. Esta relación funciona también en sentido contrario, en donde podemos obtener la posición integrando dos veces la aceleración. De esta cuenta, si tuviésemos a disposición un sensor capaz de medir la aceleración de un objeto, podríamos en teoría obtener la posición del mismo integrando (numéricamente) dos veces las mediciones de aceleración. Este es el principio fundamental del dead reckoning empleando un acelerómetro. Este método, desafortunadamente, nos presenta sólo un estimado de la posición del objeto ya que las mediciones de un acelerómetro real se comportan según el modelo [3]:

$$ {}^{S}\mathbf{a} = {}^{S}\mathbf{R}_{N}\left({}^{N}\mathbf{a} - {}^{N}\mathbf{g}\right) + {}^{S}\mathbf{b}_{a} + {}^{S}\mathbf{n}_{a}, \tag{3} $$

en donde ${}^{N}\mathbf{g}$ representa al vector de aceleración por la gravedad, ${}^{S}\mathbf{n}_{a} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{a})$ es el ruido que afecta las mediciones del acelerómetro y ${}^{S}\mathbf{b}_{a}$ es un término de deriva (o *bias*) tal que ${}^{S}\dot{\mathbf{b}}_{a} = {}^{S}\mathbf{b}_{ba} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{a})$. De esto podemos observar claramente que al integrar las mediciones de aceleración dos veces también estamos integrando el ruido y la deriva, los cuales corromperían de manera acumulativa en el tiempo a los estimados de posición.

Por el problema de acumulación de error planteado con anterioridad, en lugar de generar los estimados de posición de forma directa mediante integración numérica, podemos plantear un sistema dinámico que nos relacione los estimados de posición con las mediciones de aceleración, el cual luego podremos emplear dentro de la etapa de predicción de un filtro de Kalman. Para hacer esto, primero debemos asumir que el robot se comporta como una partícula dentro del marco de referencia de navegación. Segundo, asumimos que el marco de referencia del sensor coincide con el de la base del robot. Bajo estos supuestos entonces podemos plantear el siguiente sistema

$$ \begin{bmatrix} {}^{N}\dot{\mathbf{q}} \\ {}^{N}\ddot{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} {}^{N}\mathbf{q} \\ {}^{N}\dot{\mathbf{q}} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} {}^{N}\mathbf{a}, $$

en donde podemos encontrar la aceleración con respecto del marco de navegación despejando el término correspondiente en (3) y obteniendo

$$ {}^{N}\mathbf{a} = {}^{N}\mathbf{R}_{S}\left({}^{S}\mathbf{a} + {}^{S}\mathbf{g} - {}^{S}\mathbf{b}_{a}\right) - {}^{N}\mathbf{n}_{a} = {}^{N}\mathbf{R}_{S}\left({}^{S}\mathbf{a} - {}^{S}\mathbf{b}_{a}\right) + {}^{N}\mathbf{g} + {}^{N}\mathbf{w}_{a}. $$

Nótese que la rotación, al no modificar la norma de los vectores sobre los que opera, no modifica ni la media ni la varianza del ruido. Adicionalmente, tenemos que conocer la orientación del sensor ${}^{N}\mathbf{R}_{S}$ con respecto del marco de navegación para poder utilizar el modelo, lo cual implica que este debe ir acompañado de una estimación de orientación. Bajo estas consideraciones, obtenemos finalmente el modelo

$$ \begin{bmatrix} {}^{N}\dot{\mathbf{q}} \\ {}^{N}\ddot{\mathbf{q}} \\ {}^{S}\dot{\mathbf{b}}_{a} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -{}^{N}\mathbf{R}_{S} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} {}^{N}\mathbf{q} \\ {}^{N}\dot{\mathbf{q}} \\ {}^{S}\mathbf{b}_{a} \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ {}^{N}\mathbf{R}_{S} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} {}^{S}\mathbf{a} \\ {}^{N}\mathbf{g} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ {}^{N}\mathbf{w}_{a} \\ {}^{S}\mathbf{b}_{ba} \end{bmatrix}. $$

Para poder emplear esto dentro de un KF debemos primero discretizar el modelo. En la literatura esto típicamente se hace mediante el método de Tustin (esto se hace porque las expresiones resultantes para la posición y la velocidad tienen la forma conocida $v_1 = v_0 + at$ y $p_1 = p_0 + v_0 t + (1/2)at^2$) obteniendo

$$ \begin{bmatrix} {}^{N}\mathbf{q}_{k+1} \\ {}^{N}\dot{\mathbf{q}}_{k+1} \\ {}^{S}\mathbf{b}_{a,k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \Delta t\mathbf{I} & -(\Delta t^2/2){}^{N}\mathbf{R}_{S} \\ \mathbf{0} & \mathbf{I} & -\Delta t{}^{N}\mathbf{R}_{S} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} {}^{N}\mathbf{q}_{k} \\ {}^{N}\dot{\mathbf{q}}_{k} \\ {}^{S}\mathbf{b}_{a,k} \end{bmatrix} + \begin{bmatrix} (\Delta t^2/2){}^{N}\mathbf{R}_{S} & (\Delta t^2/2)\mathbf{I} \\ \Delta t{}^{N}\mathbf{R}_{S} & \Delta t\mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} {}^{S}\mathbf{a}_{k} \\ {}^{N}\mathbf{g} \end{bmatrix} + \ldots $$

$$... \begin{bmatrix} (\Delta t^2/2)\mathbf{I} & -(\Delta t^3/4)^N\mathbf{R}_S \\ \Delta t\mathbf{I} & -(\Delta t^2/2)^N\mathbf{R}_S \\ \mathbf{0} & \Delta t\mathbf{I} \end{bmatrix} \begin{bmatrix} {}^N\mathbf{w}_{a,k} \\ {}^S\mathbf{b}_{ba,k} \end{bmatrix},$$

lo cual se simplifica finalmente a

$$\underbrace{\begin{bmatrix} {}^N\mathbf{q}_{k+1} \\ {}^N\dot{\mathbf{q}}_{k+1} \\ {}^N\mathbf{b}_{a,k+1} \end{bmatrix}}_{\mathbf{x}_{k+1}} = \underbrace{\begin{bmatrix} \mathbf{I} & \Delta t\mathbf{I} & -(\Delta t^2/2) \\ \mathbf{0} & \mathbf{I} & -\Delta t \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} {}^N\mathbf{q}_k \\ {}^N\dot{\mathbf{q}}_k \\ {}^N\mathbf{b}_{a,k} \end{bmatrix}}_{\mathbf{x}_k} + \underbrace{\begin{bmatrix} (\Delta t^2/2)^N\mathbf{R}_S & (\Delta t^2/2)\mathbf{I} \\ \Delta t^N\mathbf{R}_S & \Delta t\mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}}_{\mathbf{B}} \underbrace{\begin{bmatrix} {}^S\mathbf{a}_k \\ {}^N\mathbf{g} \end{bmatrix}}_{\mathbf{u}_k} + \underbrace{\begin{bmatrix} {}^N\mathbf{w}_{p,k} \\ {}^N\mathbf{w}_{v,k} \\ {}^N\mathbf{b}_{ba,k} \end{bmatrix}}_{\mathbf{w}_k},$$

con ${}^N\mathbf{b}_{ba,k} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_a)$, ${}^N\mathbf{w}_{v,k} \sim \mathcal{N}\left(\mathbf{0}, (\Delta t - \Delta t^2/2)\mathbf{Q}_a\right)$ y ${}^N\mathbf{w}_{p,k} \sim \mathcal{N}\left(\mathbf{0}, (\Delta t^2/2 - \Delta t^3/4)\mathbf{Q}_a\right)$.

A pesar que este modelo es significativamente más complejo que el de odometría, tenemos la ventaja que funciona para cualquier tipo de robot o vehículo móvil.

Estimando la orientación

En robótica móvil con ruedas, típicamente el único ángulo que nos interesa es el de rumbo (*heading*, casi siempre corresponde al *yaw*). Dado que el modelo de odometría ya nos otorga un estimado de éste, basta con emplear un sensor exteroceptivo, como una brújula por ejemplo, para realizar una corrección. La estimación de orientación, sin embargo, es crítica para cualquier tipo de robot que se mueva en las tres dimensiones y en algunos casos es hasta de mayor importancia que la estimación de posición (para el control de satélites pequeños por ejemplo, el control es puramente de orientación). Por lo tanto, estudiaremos brevemente una propuesta de algoritmo para hacer precisamente esto.

IMUs y AHRS

Un AHRS (Attitude and Heading Reference System) emplea las mediciones otorgadas por los sensores de una IMU (Inertial Measurement Unit), típicamente un acelerómetro, un giroscopio y un magnetómetro de tres ejes cada uno, para obtener mediciones robustas de los ángulos de navegación: roll, pitch, yaw. En la actualidad, el estado del arte en algoritmos para implementar un AHRS es el denominado *filtro de Madgwick* [4], el cual plantea el problema de estimación con cuaterniones unitarios y optimización no lineal. Otras opciones incluyen filtros complementarios, el algoritmo DCM (Direction Cosine Matrix) y fusión de sensores empleando un KF. En esta sección desarrollaremos el último de los casos, específicamente el planteamiento que utiliza directamente los ángulos de navegación (a diferencia de un planteamiento alternativo que usa cuaterniones unitarios).

En este planteamiento del algoritmo, se usan las mediciones del giroscopio para la predicción y las mediciones del acelerómetro y del magnetómetro para la corrección. Para el giroscopio, emplearemos el modelo [3]:

$$\boldsymbol{\omega}_g = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = {}^S\boldsymbol{\omega}_{NS} + {}^S\mathbf{b}_g + {}^S\mathbf{n}_g, \tag{4}$$

en donde ${}^S\mathbf{n}_g \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_\omega)$ es el ruido Gaussiano que afecta a las mediciones del giroscopio y ${}^S\mathbf{b}_g$ es un término de deriva que cumple con ${}^S\dot{\mathbf{b}}_g = {}^S\mathbf{b}_{bg} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_\omega)$. Nótese que, en teoría, podemos recuperar los ángulos de navegación integrando las mediciones de velocidad angular que entrega el giroscopio. Sin embargo, similar al caso de estimación de posición con el acelerómetro, la

deriva y el error se acumulan corrompiendo rápidamente los estimados de los ángulos. Por lo tanto, la estrategia será plantear la estimación de ángulos mediante un modelo dinámico que podamos emplear en la parte de predicción de un KF. Este modelo resulta ser

$$
\begin{aligned}
{}^{N}\dot{\boldsymbol{\theta}}_S &= {}^{S}\boldsymbol{\omega}_{NS} = \boldsymbol{\omega}_g - {}^{S}\mathbf{b}_g - {}^{S}\mathbf{n}_g, \\
\Rightarrow \begin{bmatrix} {}^{N}\dot{\boldsymbol{\theta}}_S \\ {}^{S}\dot{\mathbf{b}}_g \end{bmatrix} &= \begin{bmatrix} \mathbf{0} & -\mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} {}^{N}\boldsymbol{\theta}_S \\ {}^{S}\mathbf{b}_g \end{bmatrix} + \begin{bmatrix} \mathbf{I} \\ \mathbf{0} \end{bmatrix} \boldsymbol{\omega_g} + \begin{bmatrix} -{}^{S}\mathbf{n}_g \\ {}^{S}\mathbf{b}_{bg} \end{bmatrix}.
\end{aligned}
$$

En la literatura, este modelo se discretiza típicamente empleando forward Euler de lo cual obtenemos que

$$
\underbrace{\begin{bmatrix} {}^{N}\boldsymbol{\theta}_{S,k+1} \\ {}^{S}\mathbf{b}_{g,k+1} \end{bmatrix}}_{\mathbf{x}_{k+1}} = \underbrace{\begin{bmatrix} \mathbf{I} & -\Delta t\mathbf{I} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} {}^{N}\boldsymbol{\theta}_{S,k} \\ {}^{S}\mathbf{b}_{g,k} \end{bmatrix}}_{\mathbf{x}_k} + \underbrace{\begin{bmatrix} \Delta t\mathbf{I} \\ \mathbf{0} \end{bmatrix}}_{\mathbf{B}} \underbrace{\boldsymbol{\omega_g}}_{\mathbf{u}_k} + \underbrace{\begin{bmatrix} {}^{S}\mathbf{w}_g \\ {}^{S}\mathbf{w}_{bg} \end{bmatrix}}_{\mathbf{w}_k},
$$

en donde ${}^{S}\mathbf{w}_g \sim \mathcal{N}(\mathbf{0}, \Delta t\mathbf{Q}_g)$ y ${}^{S}\mathbf{w}_{bg} \sim \mathcal{N}(\mathbf{0}, \Delta t\mathbf{Q}_g)$. Luego, emplearemos el acelerómetro para obtener estimados del roll y el pitch que figurarán como las mediciones en la parte de corrección del KF. Si el acelerómetro se encuentra estático en el marco de navegación, sus mediciones corresponden a los componentes del vector de gravedad, como se muestra en la Figura 5. Si sabemos que el vector de gravedad apunta siempre hacia el centro de la Tierra podemos encontrar los ángulos de pitch y roll mediante

$$
\phi = \arctan\left(\frac{a_y}{a_z}\right), \qquad \theta = \arctan\left(\frac{-a_x}{\sqrt{a_y^2 + a_z^2}}\right),
$$

en donde (a_x, a_y, a_z) corresponden a las mediciones en bruto del acelerómetro luego de un proceso de calibración, que consta típicamente en una conversión a gs ($1g = 9.8m/s^2$) y un ajuste para que mida exactamente $\pm 1g$ al estar ajustados sus ejes exactamente con el vector de gravedad. Si sabemos que ${}^{N}\boldsymbol{\theta}_S = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^\top$, entonces el modelo de medición empleando el acelerómetro resulta

$$
\mathbf{y}_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{x}_k + \begin{bmatrix} v_{\phi,k} \\ v_{\theta,k} \end{bmatrix},
$$

en donde las varianzas de los ruidos de observación $v_{\phi,k} \sim \mathcal{N}(0, \sigma_\phi)$, $v_{\theta,k} \sim \mathcal{N}(0, \sigma_\theta)$, pueden estimarse de las mediciones de aceleración. Para la corrección del yaw basta con agregar a las mediciones del acelerómetro, las mediciones del magnetómetro que se detallarán en la siguiente sección.

Corrigiendo los estimados de pose

En la sección anterior discutimos cómo generar modelos dinámicos que nos permitieran generar estimados del estado del robot empleando, en su mayoría, sensores propioceptivos. Estos modelos pueden luego emplearse en la etapa de predicción de un KF o un EKF, siempre y cuando sea posible completar el filtro con una etapa de corrección otorgada por las mediciones de uno o más sensores exteroceptivos (típicamente). En esta sección plantearemos entonces algunas de las opciones disponibles para efectuar esto.

Corrección de posición (velocidad y aceleración) empleando un GPS en ambientes abiertos

El sistema de posicionamiento global (GPS) es un sistema de radio-navegación satelital operado por la Fuerza Espacial de los Estados Unidos, el cual pertenece a un tipo de sistema de navegación

basado en *beacons* o guías, en donde se emplea la información del tiempo de vuelo de alguna señal de comunicación (radiofrecuencia, ultrasónica, infrarroja, etc.) entre el objeto a localizar y los beacons. Posteriormente se emplea triangulación para determinar la posición absoluta del objeto con respecto al marco de referencia "global" (no necesariamente tiene que corresponder al marco de la Tierra, sino simplemente a un marco inercial fijo). Si bien la precisión de la posición medida del GPS para aplicaciones militares está en la escala de centímetros, esta aumenta a metros en aplicaciones de nivel consumidor. Como ejemplo particular, el móudulo FGPMMOPA6B Global Top Technology Inc. presenta (por si solo) una precisión de 2.5-3 metros para las mediciones de posición (en espacios abiertos), aunque mejora significativamente para las mediciones de velocidad y aceleración. El modelo de medición para el GPS resulta entonces [3]:

$$\mathbf{y}_k = \begin{bmatrix} {}^I\mathbf{q} \\ {}^I\dot{\mathbf{q}} \\ {}^I\ddot{\mathbf{q}} \end{bmatrix} + \mathbf{v}_k = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} {}^I\mathbf{q} \\ {}^I\dot{\mathbf{q}} \\ {}^I\ddot{\mathbf{q}} \end{bmatrix} + \mathbf{v}_k = \mathbf{C}\mathbf{x}_k + \mathbf{v}_k,$$

en donde ${}^I\mathbf{q}, {}^I\dot{\mathbf{q}}$ y ${}^I\ddot{\mathbf{q}}$ corresponden a la posición, velocidad y aceleración del robot/vehículo con respecto al marco inercial $\{I\}$ de la Tierra, si este se representa como una partícula. Nótese que estas cantidades pueden transformarse fácilmente al marco de navegación $\{N\}$ si se conocen las coordenadas de este con respecto del marco inercial.
Finalmente, otra de las características problemáticas del GPS es su baja frecuencia de actualización (≤ 10 Hz), por lo que puede que deba manejarse un distinto período de muestreo que el de la etapa de predicción. Esto también nos permite considerar el caso cuando se pierde la señal del GPS al momento que el objeto entre a un espacio cerrado, o se encuentre bajo tierra o agua. La discrepancia en tiempos de muestreo causará que no siempre se ejecute una etapa de corrección luego de la de predicción en el filtro, aunque el filtro completo debería funcionar siempre y cuando la predicción no diverja de forma desmesurada entre las correcciones.

Corrección del ángulo de yaw/rumbo empleando un magnetómetro

Un magnetómetro es un sensor capaz de medir el campo magnético local, el cual está compuesto tanto por el campo magnético de la Tierra como el producido por la presencia de algún material magnético. El campo magnético local de la Tierra se denota por el vector $\mathbf{H}$, el cual forma un ángulo ϑ con respecto de la superficie terrestre local, como se muestra en la Figura 6.
El ángulo ϑ se conoce como inclinación magnética (*magnetic dip angle*) y establece la taza entre los componentes horizontal, el cual apunta hacia el polo norte magnético de la Tierra, y vertical del

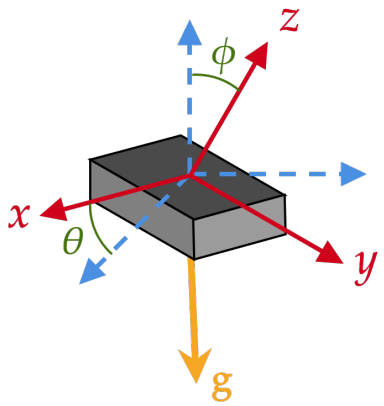

Figura 5: Vector de gravedad en los marcos del acelerómetro.

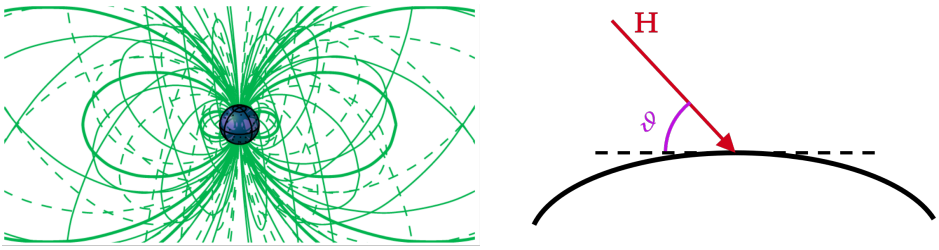

Figura 6: Campo magnético local de la Tierra.

campo magnético terrestre [3]. Los magnetómetros típicamente cuentan con tres ejes de medición, los cuales detectan los componentes H_x, H_y y H_z del vector de campo magnético $\mathbf{H}$, como se detalla en la Figura 7.
Como puede observarse, existen dos mediciones de rumbo, una que representa el norte magnético de la Tierra y otra que representa el norte real las cuales pueden calcularse de la siguiente manera

$$\psi_{\text{mag}} = \arctan\left(\frac{H_y}{H_x}\right), \qquad \psi_{\text{real}} = \psi_{\text{mag}} - \delta\left({}^{E}\mathbf{q}\right), \tag{5}$$

en donde δ es el ángulo de declinación magnética el cual depende de la posición ${}^{E}\mathbf{q}$ del sensor con respecto del marco de la Tierra. Este parámetro, junto con el ángulo de inclinación se conocen con certeza gracias a diversos estudios geofísicos.

Es importante hacer notar que las fórmulas (5) asumen que el vector de campo magnético se mide con respecto a un marco de referencia cuyo plano $x-y$ es paralelo a la superficie de la Tierra, es decir, asumen un vector de campo magnético ${}^{N}\mathbf{H}$. Para el caso de un robot móvil con ruedas esto puede asumirse con relativa certeza ya que el plano $x-y$ de la base $\{B\}$ siempre es paralelo al del marco de navegación $\{N\}$, por ende el modelo de medición para este caso resulta

$$y_k = \psi_k + v_{\psi,k},$$

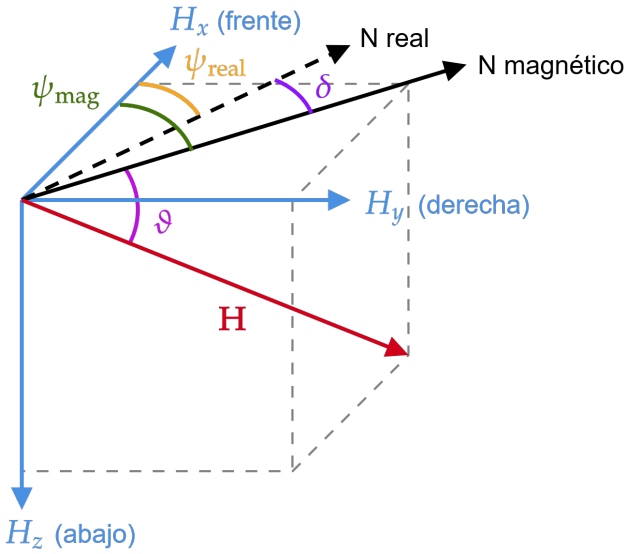

Figura 7: Componentes del vector de campo magnético.

con ψ cualquiera de los dos ángulos de rumbo y $v_\psi \sim \mathcal{N}(0, \sigma_\psi)$, el cual puede combinarse con el modelo de odometría y un GPS para estimar completamente la pose de un robot móvil con ruedas en ambientes abiertos, siempre y cuando el sensor haya sido calibrado para compensar el campo magnético generado por los componentes eléctricos y electromecánicos del robot. Para ambientes cerrados, la presencia de materiales magnéticos en la estructura de casas o edificios (o inclusive dentro de muebles) perturba las mediciones del campo magnético local y desafortunadamente no pueden eliminarse mediante la calibración del sensor, por lo que debe emplearse algún otro método para compensar por estos errores.

En caso que el marco de nuestro robot no tenga un plano paralelo a la superficie terrestre, las mediciones de campo del magnetómetro deben compensarse en orientación, por lo que el modelo de medición cambia a [3]:

$$\mathbf{y}_k = {}^S\mathbf{R}_N {}^N\mathbf{H}_k + \mathbf{v}_k,$$

con $\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{v,m})$, el cual puede combinarse con un acelerómetro, giroscopio y GPS para obtener la pose completa de cualquier robot móvil en ambientes abiertos.

Corrección de pose en ambientes cerrados empleando un mapa conocido

Como pudimos observar, tanto el GPS como el magnetómetro presentan problemas (es más, el GPS no es viable) al momento de querer emplear sus mediciones para efectuar una corrección en ambientes cerrados. Por lo tanto, una opción viable para efectuar corrección de pose para robots móviles dentro de ambientes cerrados es emplear la información provista por un mapa del ambiente cerrado en donde se encuentre navegando el robot. Este mapa puede ser conocido por el robot a-priori o puede irse construyendo como resultado de un proceso de mapeo, el cual se detallará en la siguiente sección.
Como caso particular, consideraremos a un robot móvil con ruedas equipado con un sensor LiDAR de distancia, dentro de un mapa conocido conformado por un conjunto de M obstáculos o *landmarks* puntuales (para un caso más general puede consultarse la sección 6.8 de [2]) ${}^N\mathbf{p}_i = \begin{bmatrix} x_i & y_i \end{bmatrix}^\top$, $i = 1, 2, ..., M$ como el que se muestra en la Figura 8. El sensor de distancia presenta el modelo de medición [2]:

$$\mathbf{y}_{i,k} = {}^B\begin{bmatrix}\kappa_i \\ \beta_i\end{bmatrix} = {}^B\boldsymbol{\mathcal{G}}\left({}^N\boldsymbol{\xi}_k, {}^N\mathbf{p}_i, \mathbf{v}_k\right) = \underbrace{\begin{bmatrix}\sqrt{(x_i - \xi_{1,k})^2 + (y_i - \xi_{2,k})^2} \\ \arctan\left(\frac{y_i - \xi_{2,k}}{x_i - \xi_{1,k}}\right) - \xi_{3,k}\end{bmatrix}}_{\boldsymbol{\mathcal{H}}_i(\mathbf{x}_k)} + \underbrace{\begin{bmatrix}v_\kappa \\ v_\beta\end{bmatrix}}_{\mathbf{v}_k}, \tag{6}$$

el cual presenta la distancia relativa del i-ésimo obstáculo/landmark mediante las coordenadas polares κ_i (*range*) y β_i (*bearing*), siempre y cuando éste se encuentre dentro del rango de medición ε del sensor. Recordemos que ${}^N\boldsymbol{\xi} = \begin{bmatrix}\xi_1 & \xi_2 & \xi_3\end{bmatrix}^\top = \begin{bmatrix}x & y & \theta\end{bmatrix}^\top$ corresponde al estado del robot móvil, por lo que puede juntarse este modelo de medición junto con una predicción mediante odometría (2) para generar un EKF que estime la pose completa del robot. **Nota:** se emplea la función arctan dentro del modelo del sensor para que sea sencillo encontrar su derivada, sin embargo, en software debe implementarse mediante la función atan2 para poder obtener ángulos en el rango $(-\pi, \pi)$.

Mapeo

Con el problema de estimación de pose resuelto, tenemos la garantía de poder saber en dónde se encuentra y cómo se encuentra orientado nuestro robot con bastante certeza, con lo cual es posible generar controladores que lleven al robot de una pose inicial a una pose final deseada. Esto, sin embargo, no toma en consideración que los robots por lo general navegan dentro de entornos llenos de obstáculos y de otros agentes autónomos (sean personas, animales o inclusive otros robots). El conocer completamente su pose, desafortunadamente, no le da al robot ninguna información sobre su entorno, por lo que es necesario dotar al robot de sensores exteroceptivos como sensores de distancia, cámaras, entre otros, para hacerlo capaz de detectar obstáculos o agentes distintos a el mismo. Dado que este tipo de sensores exteroceptivos miden cantidades relativas al marco $\{B\}$ del robot, no es necesario conocer la pose del mismo para poder esquivar a los obstáculos dentro del entorno de navegación, como pudimos observarlo en la perspectiva de robótica basada en *behaviors* que se estudió en el módulo anterior. Conocer la pose del robot, sin embargo, nos permitirá extender las capacidades de estos sensores para combinar la información relativa medida ${}^{B}\mathbf{p}_i$ con la información absoluta del robot ${}^{N}\mathbf{T}_B$ para obtener un estimado de la pose absoluta de los obstáculos/agentes ${}^{N}\mathbf{p}_i$. Si los obstáculos dentro del entorno de navegación se mantienen fijos, el obtener la posición de los mismos combinando la pose conocida del robot con mediciones relativas de distancia y orientación, permite la construcción en tiempo real de un mapa del entorno desconocido. Por esta razón, denominaremos *mapeo* a este proceso.

Landmarks puntuales asumiendo estimación perfecta de pose

Si bien el caso más general de mapeo busca construir una *occupancy grid* (como las empleadas en el curso de Robótica 1) del entorno desconocido, para luego poder emplear algoritmos de planificación, consideraremos el caso más sencillo de la Figura 8 en donde todos los obstáculos o *landmarks* (lugares/objetos hacia donde se quiere llegar, no esquivar) pueden aproximarse como puntos, de los cuales sólo nos interesa su posición. **Nota:** se describe el caso más general de mapeo en la sección 6.8 de [2]

Para el planteamiento de este problema, asumiremos lo inverso de lo que se supuso en la corrección de pose mediante un mapa conocido. Es decir, asumiremos que se conoce perfectamente la pose del robot (como resultado de una estimación de pose) pero se desconoce la cantidad M y las posiciones ${}^{N}\mathbf{p}_i,\ i = 1, 2, .., M$ de los obstáculos dentro del mapa. De nuevo, similar al problema de localización, nuestra intención es plantear un KF/EKF que resuelva el problema de estimación, por lo que requerimos tanto un modelo del proceso que genera las cantidades que nos interesa estimar al igual que un modelo de medición para los sensores que tendrán la tarea de corregir las predicciones. Se describirá cada una de las piezas requeridas para el KF/EKF a continuación [2]:

- **Cantidades a estimar/variables de estado:** lo que nos interesa estimar son las posiciones ${}^{N}\mathbf{p}_i$ de los M obstáculos dentro del mapa. Sin embargo, se tiene el problema que el robot no conoce a-priori ni el número de obstáculos ni la posición (o un estimado de) de los mismos. Por lo tanto, debe construirse y/o aumentarse el estimado del vector de estado conforme se van encontrando nuevos obstáculos, mientras navega el robot en el mapa desconocido. Bajo esta consideración, iniciaremos con un estimado vacío $\hat{\mathbf{x}}_{k|k} = \begin{bmatrix} \ \end{bmatrix}$ el cual se aumentará al momento que el robot encuentre el i-ésimo obstáculo mediante el mapa

$$\hat{\mathbf{x}}_{k|k}^{\text{new}} = \boldsymbol{\mathcal{Y}}\left(\hat{\mathbf{x}}_{k|k}, {}^{N}\boldsymbol{\xi}_k, \mathbf{y}_{i,k}\right) = \begin{bmatrix} \hat{\mathbf{x}}_{k|k} \\ {}^{N}\hat{\mathbf{p}}_i \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{x}}_{k|k} \\ {}_{N}\mathbf{g}\left({}^{N}\boldsymbol{\xi}_k, \mathbf{y}_{i,k}\right) \end{bmatrix}, \tag{7}$$

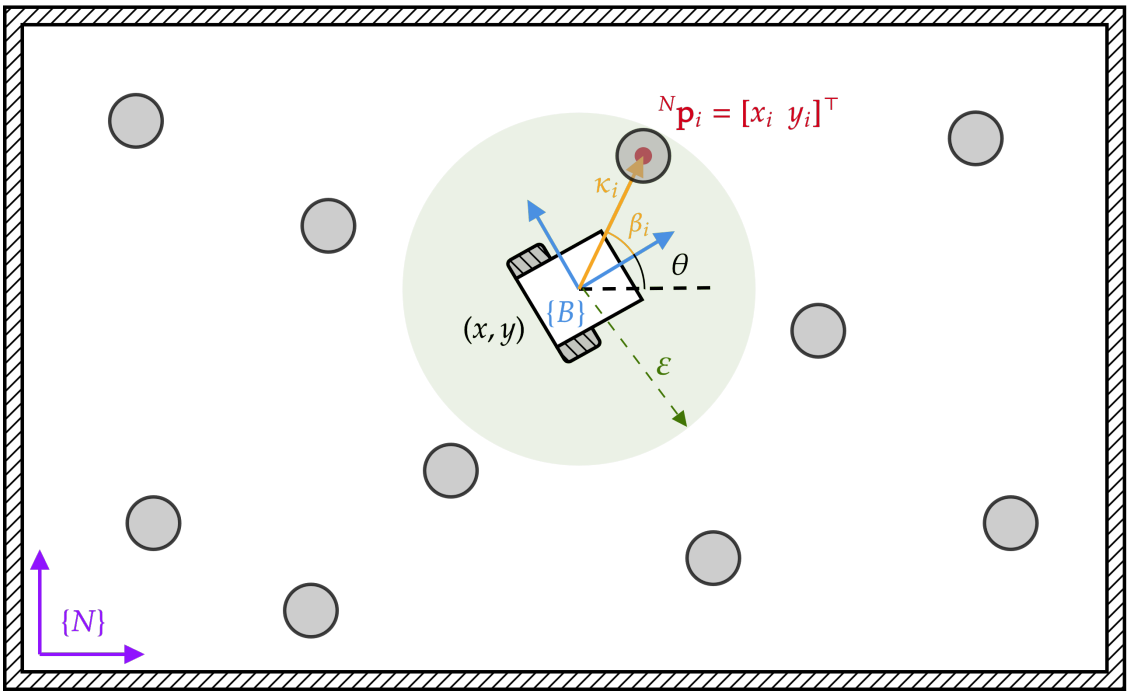

Figura 8: Robot móvil dentro de un ambiente/mapa cerrado.

$$ {}^N\mathbf{g}\left({}^N\boldsymbol{\xi}_k, \mathbf{y}_{i,k}\right) = \begin{bmatrix} \xi_{1,k} + \kappa_i \cos(\xi_{3,k} + \beta_i) \\ \xi_{2,k} + \kappa_i \sin(\xi_{3,k} + \beta_i) \end{bmatrix}, $$

en donde $\mathbf{y}_{i,k}$ corresponde a la distancia del obstáculo respecto al robot, ${}^N\boldsymbol{\xi}$ es de nuevo la pose del robot y ${}^N\mathbf{g}\left({}^N\boldsymbol{\xi}_k, \mathbf{y}_{i,k}\right)$ es la inversa de la función ${}^B\boldsymbol{\mathcal{G}}\left({}^N\boldsymbol{\xi}_k, {}^N\mathbf{p}_i\right)$ en (6), es decir, emplea la información de la pose del robot junto con las mediciones relativas del sensor de distancia para brindar un estimado de la posición absoluta del obstáculo. Es importante notar que al cambiar las dimensiones del estimado del estado, también cambian las dimensiones de todas las cantidades relacionadas al mismo, como el error de estimación $\mathbf{e}_k = \mathbf{x}_k - \hat{\mathbf{x}}_{k|k}$ y su matriz de covarianza $\mathbf{P}_{k|k}$, la cual debe aumentarse mediante

$$ \mathbf{P}_{k|k}^{\text{new}} = \mathbf{Y}_{\mathbf{y}} \begin{bmatrix} \mathbf{P}_{k|k} & \mathbf{0}_{n\times 2} \\ \mathbf{0}_{2\times n} & \mathbf{Q}_w \end{bmatrix} \mathbf{Y}_{\mathbf{y}}^\top, \tag{8} $$

con

$$ \mathbf{Q}_w = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}, \qquad \mathbf{Y}_{\mathbf{y}} = \frac{\partial \boldsymbol{\mathcal{Y}}\left(\hat{\mathbf{x}}_{k|k}, {}^N\boldsymbol{\xi}_k, \mathbf{y}_{i,k}\right)}{\partial \mathbf{y}} = \begin{bmatrix} \mathbf{I}_{n\times n} & & \mathbf{0}_{n\times 2} \\ \mathbf{G}_{\boldsymbol{\xi}} & \mathbf{0}_{2\times n-3} & \mathbf{G}_{\mathbf{y}} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{n\times n} & \mathbf{0}_{n\times 2} \\ \mathbf{0}_{2\times n} & \mathbf{G}_{\mathbf{y}} \end{bmatrix}, $$

$$ \mathbf{G}_{\boldsymbol{\xi}} = \frac{\partial {}^N\mathbf{g}\left({}^N\boldsymbol{\xi}_k, \mathbf{y}_{i,k}\right)}{\partial {}^N\boldsymbol{\xi}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, $$

$$ \mathbf{G}_{\mathbf{y}} = \frac{\partial {}^N\mathbf{g}\left({}^N\boldsymbol{\xi}_k, \mathbf{y}_{i,k}\right)}{\partial \mathbf{y}} = \begin{bmatrix} \cos(\xi_{3,k} + \beta_i) & -\kappa_i \sin(\xi_{3,k} + \beta_i) \\ \sin(\xi_{3,k} + \beta_i) & \kappa_i \cos(\xi_{3,k} + \beta_i) \end{bmatrix}, $$

en donde σ_x^2, σ_y^2 corresponden a las varianzas del ruido blanco que corrompe a los estimados ${}^N\hat{\mathbf{p}}_i,\ i = 1, 2, ..., M$ de la posición de los obstáculos, n es el tamaño del estimado del vector de estado previo de su aumento de tamaño y la matriz $\mathbf{G}_{\boldsymbol{\xi}}$ es cero a causa del conocimiento

perfecto de la pose del robot (lo cual cambiará para el problema de localización y mapeo simultáneo de la siguiente sección).

- **Modelo de predicción:** dado que la posición de los obstáculos se asume fija para todo tiempo, el modelo de predicción a emplear dentro del EKF resulta trivial

$$\hat{\mathbf{x}}_{k|k-1} = \hat{\mathbf{x}}_{k-1|k-1}, \qquad \mathbf{P}_{k|k-1} = \mathbf{P}_{k-1|k-1},$$

donde claramente $\mathbf{A} = \mathbf{I}, \mathbf{B} = \mathbf{F} = \mathbf{0}$.

- **Corrección basada en las mediciones:** el modelo del sensor empleado es de nuevo el LiDAR (6). Dado que este es no lineal, necesitamos su jacobiano $\mathbf{C}[k]$ con respecto del estado para poder calcular la ganancia de Kalman $\mathbf{L}_k$ y actualizar el estimado a-posteriori de la matriz de covarianza $\mathbf{P}_{k|k}$, por lo cual emplearemos

$$\mathbf{C}[k] = \begin{bmatrix} \mathbf{C}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{C}_M \end{bmatrix},$$

$$\mathbf{C}_i = \frac{\partial \boldsymbol{\mathcal{H}}_i\left(\hat{\mathbf{x}}_{k|k-1}\right)}{\partial \mathbf{p}_i} = \begin{bmatrix} \frac{x_i - \xi_{1,k}}{\sqrt{(x_i-\xi_{1,k})^2+(y_i-\xi_{2,k})^2}} & \frac{y_i - \xi_{2,k}}{\sqrt{(x_i-\xi_{1,k})^2+(y_i-\xi_{2,k})^2}} \\ -\frac{y_i - \xi_{2,k}}{(x_i-\xi_{1,k})^2+(y_i-\xi_{2,k})^2} & \frac{x_i - \xi_{1,k}}{(x_i-\xi_{1,k})^2+(y_i-\xi_{2,k})^2} \end{bmatrix},$$

o bien, de forma equivalente, podemos decir que el i-ésimo bloque columna de $\mathbf{C}[k]$ está formado por

$$\begin{bmatrix} \mathbf{0} & \cdots & \mathbf{C}_i & \cdots & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{2\times 2N}.$$

Nótese que las cantidades $\mathbf{y}_{i,k}, \boldsymbol{\mathcal{H}}_i\left(\hat{\mathbf{x}}_{k|k-1}\right) \equiv \boldsymbol{\mathcal{H}}_i\left(\hat{\mathbf{p}}_{i,k|k-1}\right)$ en (6) y $\mathbf{C}_i$ existen sólo si el i-ésimo obstáculo se encuentra dentro del rango de detección del sensor de distancia. Por lo tanto, para obstáculos fuera del rango del sensor deben hacerse $\mathbf{C}_i = \mathbf{0}$ y $\mathbf{y}_{i,k} - \boldsymbol{\mathcal{H}}_i\left(\hat{\mathbf{x}}_{k|k-1}\right) = \mathbf{0}$ para evitar que la etapa de corrección los tome en consideración.

Localización y mapeo simultáneo (SLAM)

Gracias a las propuestas de solución planteadas para los problemas de localización y mapeo, finalmente podemos llegar a uno de los problemas más importantes en robótica móvil, el de localización y mapeo simultáneo SLAM (*Simultaneous Localization And Mapping*). En este problema buscamos no solamente estimar la pose del robot móvil, típicamente un robot con ruedas, dentro de un entorno desconocido sino que al mismo tiempo buscamos crear un mapa de dicho entorno. Como puede verse, esta situación conforma un problema del huevo o la gallina, en donde se requiere del mapa para localizar al robot pero el mapa se construye empleando la pose del robot. Afortunadamente, el EKF presenta una solución viable a este problema, combinando tres de los casos detallados con anterioridad: dead reckoning empleando odometría, corrección de pose empleando un mapa conocido y mapeo de landmarks puntuales. Ahora el estimado del vector de estado se iniciará con la pose del robot y se irá aumentando con la posición de cada uno de los obstáculos detectados por el sensor de distancia conforme el robot navega en el entorno desconocido, empleando (7) y (8) igual que en el caso de mapeo pero con la diferencia que

$$\mathbf{G}_{\boldsymbol{\xi}} = \frac{\partial^N \mathbf{g}\left({}^N\boldsymbol{\xi}_k, \mathbf{y}_{i,k}\right)}{\partial^N \boldsymbol{\xi}} = \begin{bmatrix} 1 & 0 & -\kappa_i \sin(\xi_{3,k} + \beta_i) \\ 0 & 1 & \kappa_i \cos(\xi_{3,k} + \beta_i) \end{bmatrix}.$$

El estimado de la pose del robot se propaga mediante el modelo de odometría (2) mientras que los estimados a-priori de las posiciones de los obstáculos se quedan fijos como se asumió también en el caso de mapeo. En corrección se emplea el modelo del sensor de distancia para tanto la pose del robot como para las posiciones de los obstáculos, modificando el jacobiano de la siguiente forma

$$\mathbf{C}[k] = \begin{bmatrix} \mathbf{C}_{\boldsymbol{\xi}_1} & \mathbf{C}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{C}_{\boldsymbol{\xi}_2} & \mathbf{0} & \mathbf{C}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \\ \mathbf{C}_{\boldsymbol{\xi}_M} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{C}_M \end{bmatrix},$$

$$\mathbf{C}_{\boldsymbol{\xi}_i} = \frac{\partial \boldsymbol{\mathcal{H}}_i\left(\hat{\mathbf{x}}_{k|k-1}\right)}{\partial {}^N\boldsymbol{\xi}} = \begin{bmatrix} -\frac{x_i-\xi_{1,k}}{\sqrt{(x_i-\xi_{1,k})^2+(y_i-\xi_{2,k})^2}} & -\frac{y_i-\xi_{2,k}}{\sqrt{(x_i-\xi_{1,k})^2+(y_i-\xi_{2,k})^2}} & 0 \\ \frac{y_i-\xi_{2,k}}{(x_i-\xi_{1,k})^2+(y_i-\xi_{2,k})^2} & -\frac{x_i-\xi_{1,k}}{(x_i-\xi_{1,k})^2+(y_i-\xi_{2,k})^2} & -1 \end{bmatrix},$$

y aplicando las mismas consideraciones para los obstáculos fuera del rango del sensor que en el caso de mapeo. Dada la complejidad que surge al combinar los casos de localización y mapeo se presenta un pseudocódigo que detalla el algoritmo 1-2, denominado EKF-SLAM. **Nota:** dentro del algoritmo, el operador $\otimes$ se refiere al *producto de Kronecker*.

Algorithm 1 Localización y mapeo simultáneo basado en el filtro de Kalman extendido, parte 1

Require: $\mathbf{Q}_w = \begin{bmatrix} \sigma_\rho^2 & 0 \\ 0 & \sigma_\theta^2 \end{bmatrix}, \quad \mathbf{Q}_{v\mathcal{G}} = \begin{bmatrix} \sigma_\kappa^2 & 0 \\ 0 & \sigma_\beta^2 \end{bmatrix}, \quad \mathbf{Q}_{vg} = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}.$

Ensure: $\hat{\mathbf{x}}_{0|0} = {}^N\hat{\boldsymbol{\xi}}_{0|0} = E\left\{{}^N\boldsymbol{\xi}_0\right\}, \quad \mathbf{P}_{0|0} = \sigma_e^2\mathbf{I}, \quad M = 0.$

while robot navegando **do**

$\begin{bmatrix} \Delta\text{tick}_L & \Delta\text{tick}_L \end{bmatrix} \leftarrow$ medir_encoders(robot)

$\begin{bmatrix} \delta_\rho & \delta_\theta \end{bmatrix} \leftarrow$ calcular_odometria $\left(\Delta\text{tick}_L, \Delta\text{tick}_R, N_{\text{tick}}, r, \ell\right)$

$$\hat{\mathbf{x}}_{k|k-1} = \begin{bmatrix} {}^N\hat{\boldsymbol{\xi}}_{k|k-1} \\ \hat{\mathbf{p}}_{1,k|k-1} \\ \vdots \\ \hat{\mathbf{p}}_{M,k|k-1} \end{bmatrix} = \begin{bmatrix} \hat{\xi}_{1,k-1|k-1} + \delta_\rho \cos\left(\hat{\xi}_{3,k-1|k-1}\right) \\ \hat{\xi}_{2,k-1|k-1} + \delta_\rho \sin\left(\hat{\xi}_{3,k-1|k-1}\right) \\ \hat{\xi}_{3,k-1|k-1} + \delta_\theta \\ \hat{\mathbf{p}}_{1,k-1|k-1} \\ \vdots \\ \hat{\mathbf{p}}_{M,k-1|k-1} \end{bmatrix}$$

$$\mathbf{A} \leftarrow \begin{bmatrix} 1 & 0 & -\delta_\rho \sin(\hat{\xi}_{3,k-1|k-1}) \\ 0 & 1 & \delta_\rho \cos(\hat{\xi}_{3,k-1|k-1}) \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{F} \leftarrow \begin{bmatrix} \cos(\hat{\xi}_{3,k-1|k-1}) & 0 \\ \sin(\hat{\xi}_{3,k-1|k-1}) & 0 \\ 0 & 1 \end{bmatrix}$$

$$\mathbf{P}_{k|k-1} \leftarrow \begin{bmatrix} \mathbf{P}_{\xi\xi,k|k-1} & \mathbf{P}_{\xi p,k|k-1} \\ \mathbf{P}_{\xi p,k|k}^\top & \mathbf{P}_{pp,k|k-1} \end{bmatrix} = \begin{bmatrix} \mathbf{A}\mathbf{P}_{\xi\xi,k-1|k-1}\mathbf{A}^\top + \mathbf{F}\mathbf{Q}_w\mathbf{F}^\top & \mathbf{A}\mathbf{P}_{\xi p,k-1|k-1} \\ \mathbf{P}_{\xi p,k-1|k-1}^\top\mathbf{A}^\top & \mathbf{P}_{pp,k-1|k-1} \end{bmatrix}$$

obstaculos $\leftarrow$ leer_sensor_distancia(robot)

Nótese que uno de los pasos más importantes del algoritmo es discernir si cada uno de los obstáculos observados por el sensor corresponde a uno que ya se observó previamente o a uno nuevo. Un error en este paso probaría ser fatal tanto para el mapeo como la localización, ya que el supuesto que todos los procesos estocásticos del EKF son Gaussianos hace que el filtro tenga una única hipótesis

Algorithm 2 Localización y mapeo simultáneo basado en el filtro de Kalman extendido, parte 2

while ... **do**

obstaculos ← leer_sensor_distancia(robot)

for i in obstaculos **do**

if i es nuevo obstáculo **then**

$M \leftarrow M + 1$

$$\hat{\mathbf{x}}_{k|k-1} \leftarrow \begin{bmatrix} \hat{\mathbf{x}}_{k|k-1} \\ {}^{N}\hat{\mathbf{p}}_{i,0|0} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{x}}_{k|k-1} \\ \hat{\xi}_{1,k|k-1} + \kappa_i \cos\left(\hat{\xi}_{3,k|k-1} + \beta_i\right) \\ \hat{\xi}_{2,k|k-1} + \kappa_i \sin\left(\hat{\xi}_{3,k|k-1} + \beta_i\right) \end{bmatrix}$$

$$\mathbf{G}_{\boldsymbol{\xi}_i} = \begin{bmatrix} 1 & 0 & -\kappa_i \sin\left(\hat{\xi}_{3,k|k-1} + \beta_i\right) \\ 0 & 1 & \kappa_i \cos\left(\hat{\xi}_{3,k|k-1} + \beta_i\right) \end{bmatrix}$$

$$\mathbf{G}_{\mathbf{y}} = \begin{bmatrix} \cos\left(\hat{\xi}_{3,k|k-1} + \beta_i\right) & -\kappa_i \sin\left(\hat{\xi}_{3,k|k-1} + \beta_i\right) \\ \sin\left(\hat{\xi}_{3,k|k-1} + \beta_i\right) & \kappa_i \cos\left(\hat{\xi}_{3,k|k-1} + \beta_i\right) \end{bmatrix}$$

$$\mathbf{Y}_{\mathbf{y}} = \begin{bmatrix} \mathbf{I}_{n\times n} & & \mathbf{0}_{n\times 2} \\ \mathbf{G}_{\boldsymbol{\xi}} & \mathbf{0}_{2\times n-3} & \mathbf{G}_{\mathbf{y}} \end{bmatrix}$$

$$\mathbf{P}_{k|k-1} \leftarrow \mathbf{Y}_{\mathbf{y}} \begin{bmatrix} \mathbf{P}_{k|k-1} & \mathbf{0}_{n\times 2} \\ \mathbf{0}_{2\times n} & \mathbf{Q}_{vg} \end{bmatrix} \mathbf{Y}_{\mathbf{y}}^{\top}$$

$$\mathbf{z}_{i,k} \leftarrow \mathbf{y}_{i,k} - {}^{B}\boldsymbol{\mathcal{G}}\left({}^{N}\boldsymbol{\xi}_{k|k-1}, {}^{N}\mathbf{g}\left({}^{N}\boldsymbol{\xi}_{k|k-1}, \mathbf{y}_{i,k}\right), \mathbf{0}\right) = \mathbf{0}_{2\times 1}$$

else if i es obstáculo existente **then**

$$\mathbf{z}_{i,k} \leftarrow \mathbf{y}_{i,k} - \begin{bmatrix} \sqrt{\left(\hat{x}_{i,k|k-1} - \hat{\xi}_{1,k|k-1}\right)^2 + \left(\hat{y}_{i,k|k-1} - \hat{\xi}_{2,k|k-1}\right)^2} \\ \arctan\left(\frac{\hat{y}_{i,k|k-1} - \hat{\xi}_{2,k|k-1}}{\hat{x}_{i,k|k-1} - \hat{\xi}_{1,k|k-1}}\right) - \hat{\xi}_{3,k|k-1} \end{bmatrix}$$

else if i es obstáculo fuera de rango **then**

$\mathbf{z}_{i,k} \leftarrow \mathbf{0}_{2\times 1}$

$$\hat{\kappa}_i \leftarrow \sqrt{\left(\hat{x}_{i,k|k-1} - \hat{\xi}_{1,k|k-1}\right)^2 + \left(\hat{y}_{i,k|k-1} - \hat{\xi}_{2,k|k-1}\right)^2}$$

$$\mathbf{C}_i \leftarrow \begin{bmatrix} \frac{\hat{x}_{i,k|k-1} - \hat{\xi}_{1,k|k-1}}{\hat{\kappa}_i} & \frac{\hat{y}_{i,k|k-1} - \hat{\xi}_{2,k|k-1}}{\hat{\kappa}_i} \\ -\frac{\hat{y}_{i,k|k-1} - \hat{\xi}_{2,k|k-1}}{\hat{\kappa}_i^2} & \frac{\hat{x}_{i,k|k-1} - \hat{\xi}_{1,k|k-1}}{\hat{\kappa}_i^2} \end{bmatrix}$$

$$\mathbf{C}_{\boldsymbol{\xi}_i} \leftarrow \begin{bmatrix} -\frac{\hat{x}_{i,k|k-1} - \hat{\xi}_{1,k|k-1}}{\hat{\kappa}_i} & -\frac{\hat{y}_{i,k|k-1} - \hat{\xi}_{2,k|k-1}}{\hat{\kappa}_i} & 0 \\ \frac{\hat{y}_{i,k|k-1} - \hat{\xi}_{2,k|k-1}}{\hat{\kappa}_i^2} & -\frac{\hat{x}_{i,k|k-1} - \hat{\xi}_{1,k|k-1}}{\hat{\kappa}_i^2} & -1 \end{bmatrix}$$

$$\mathbf{C} \leftarrow \begin{bmatrix} \mathbf{C}_{\boldsymbol{\xi}_1} & \mathbf{C}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{C}_{\boldsymbol{\xi}_2} & \mathbf{0} & \mathbf{C}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{C}_{\boldsymbol{\xi}_M} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{C}_M \end{bmatrix}$$

$$\mathbf{L}_k \leftarrow \mathbf{P}_{k|k-1}\mathbf{C}^{\top}\left(\mathbf{C}\mathbf{P}_{k|k-1}\mathbf{C}^{\top} + \mathbf{I}_{M\times M} \otimes \mathbf{Q}_{v\mathcal{G}}\right)^{-1}$$

$$\mathbf{z} \leftarrow \begin{bmatrix} \mathbf{z}_{i,k} \\ \vdots \\ \mathbf{z}_{i,M} \end{bmatrix}$$

$$\hat{\mathbf{x}}_{k|k} \leftarrow \hat{\mathbf{x}}_{k|k-1} + \mathbf{L}_k\mathbf{z}$$

$$\mathbf{P}_{k|k} \leftarrow \mathbf{P}_{k|k-1} - \mathbf{L}_k\mathbf{C}\mathbf{P}_{k|k-1}$$

(por la única cresta de la distribución normal) sobre la posición de los obstáculos. Por esta razón se suelen emplear pruebas estadísticas de confianza (siendo la más popular una prueba χ^2 con base en la distancia de Mahalanobis, como se detalla en la sección 7.6 de [5]) o bien marcadores visuales (como los fiducial markers que discutimos en el módulo de control basado en visión) para asegurar una correcta identificación de los obstáculos o landmarks.

Si bien el EKF-SLAM funciona adecuadamente para situaciones como la descrita en la Figura 8, este muestra sus limitantes en mapas con un número elevado de obstáculos o con obstáculos más complejos como paredes por dos razones. Primero, resulta que si bien la matriz de covarianza en el caso de mapeo es *dispersa* (la mayor parte de sus elementos es cero), esta resulta ser *densa* (pocos elementos iguales a cero) en el problema de SLAM (ya que el estimado de la pose del robot está acoplado a los estimados de las posiciones de los obstáculos), por lo que se requiere de más memoria para poderla almacenar. Segundo, como tanto el estado del sistema como la covarianza aumentan de tamaño por cada nuevo obstáculo/landmark detectado, el algoritmo aumenta constantemente su complejidad computacional (número de operaciones requeridas). Combinando ambos problemas obtenemos un algoritmo que no es escalable para cualquier situación. Por lo tanto, existen otras propuestas que solucionan el problema de SLAM, siendo el FastSLAM ([6], [7] y [5]) una de las variantes más populares.

Referencias

[1] R. M. Murray y col., "Optimization-based control", *California Institute of Technology, CA*, págs. 111-128, 2009.

[2] P. Corke, *Robotics, vision and control: fundamental algorithms in MATLAB® second, completely revised*. Springer, 2017, vol. 118.

[3] M. Kok, J. D. Hol y T. B. Schön, *Using Inertial Sensors for Position and Orientation Estimation*. 2017.

[4] S. O. Madgwick, A. J. Harrison y R. Vaidyanathan, "Estimation of IMU and MARG orientation using a gradient descent algorithm", en *2011 IEEE international conference on rehabilitation robotics*, IEEE, 2011, págs. 1-7.

[5] S. Thrun, "Probabilistic robotics", *Communications of the ACM*, vol. 45, n.º 3, págs. 52-57, 2002.

[6] M. Montemerlo y S. Thrun, "Simultaneous localization and mapping with unknown data association using FastSLAM", en *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, IEEE, vol. 2, 2003, págs. 1985-1991.

[7] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit y col., "FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges", en *IJCAI*, 2003, págs. 1151-1156.