Universidad de San Andrés, Depto. de Ingeniería

# Práctica 4 - Filtros EKF

I-402 - Principios de la Robótica Autónoma

Prof. Dr. Ignacio Mas, Tadeo Casiraghi y Bautista Chasco

14 de abril de 2025

**Fecha límite de entrega:** 09/05/25, 23:59hs.
**Modo de entrega:** Enviar por el Aula Virtual del Campus el código comentado y los gráficos (.jpg ó .pdf) y/o animaciones.

## 1. Filtro de Kalman Extendido

En este ejercicio se implementará un filtro de Kalman Extendido (EKF) basándose en una estructura de código provista por la cátedra.

### 1.1. Notas preliminares

La estructura provista contiene los distintos componentes del filtro EKF, para que el esfuerzo del desarrollo sea en los detalles de la implementación del filtro propiamente dicho. El archivo comprimido que se provee incluye las carpetas:

- **data** contiene la descripción del mundo y las lecturas del sensor
- **code** contiene la estructura del filtro EKF con funciones para ser completadas

Se puede ejecutar el filtro desde una terminal con `python ekf_framework.py sensor_data.dat world.dat` (asumiendo que se encuentra todo en la misma carpeta). El filtro no correrá correctamente hasta que no se completen las funciones correspondientes.
Algunos consejos adicionales:

- Para leer los datos del mundo y del sensor, se usan diccionarios. Las funciones `read_sensor_data` y `read_world_data` leen los datos de los archivos correspondientes y arman diccionarios con *timestamps* como keys primarios. Para acceder a los datos del sensor en el diccionario, puede usarse:

```
data_dict[timestamp,'sensor']['id']
data_dict[timestamp,'sensor']['range']
```

Para la información de odometría, el diccionario puede accederse como:

```
data_dict[timestamp,'odom']['r1']
data_dict[timestamp,'odom']['t']
data_dict[timestamp,'odom']['r2']
```

Para acceder a la pocisión de las *landmarks* en `world_dict`, puede usarse:

```
world_dict[id]
```

## 1.2. Paso de predicción EKF

Supongamos que tenemos un robot diferencial operando en el plano, por lo que su estado esta definido por $\langle x, y, \theta \rangle$. Su modelo de movimiento esta definido como el Modelo de Odometría visto en clase en el capítulo de *modelos de movimiento*.

1. Calcular la matriz jacobiana $G_t$ de la función de movimiento $g$ sin ruido.
2. Implementar el paso de predicción del filtro EKF en la función `prediction_step` usando el jacobiano $G_t$ calculado en el punto anterior. Asumir que el ruido del modelo de movimiento esta definido por:

$$Q_t = \begin{pmatrix} 0{,}2 & 0 & 0 \\ 0 & 0{,}2 & 0 \\ 0 & 0 & 0{,}02 \end{pmatrix}.$$

## 1.3. Paso de corrección EKF

1. Calcular la matriz jacobiana $H_t$ de la función de medición $h$ sin ruido de un sensor que sólo mide distancia (range).
2. Implementar el paso de corrección del filtro EKF en la función `correction_step` usando el jacobiano $H_t$. Asumir que el ruido de medición está caracterizado por la matriz diagonal cuadrada $R_t$:

$$R_t = \begin{pmatrix} 0{,}5 & 0 & 0 & \dots \\ 0 & 0{,}5 & 0 & \dots \\ 0 & 0 & 0{,}5 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \in \mathbb{R}^{size(z_t) \times size(z_t)}.$$

Luego de implementar los pasos de predicción y corrección, se generarán gráficos del estado del robot para cada paso de tiempo.
Nota: Con el comando *ffmpeg* (si está instalado en tu sistema operativo) se puede generar una animación de los gráficos de la carpeta *plots* de la siguiente manera:
`ffmpeg -r 10 -i ekf_plot_%03d.png ekf.mp4`