Universidad Nacional de Rosario

Facultad de Ciencias Exactas, Ingeniería y Agrimensura

Escuela de Ingeniería Electrónica

Robótica Móvil

Trabajo Práctico 1: Transformaciones

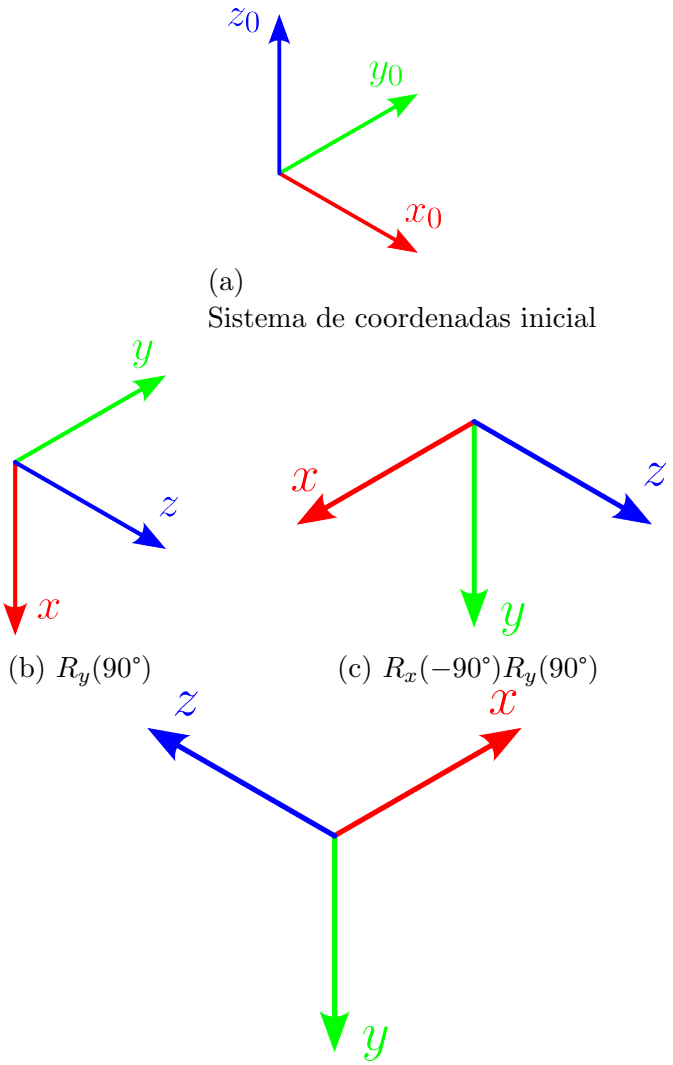
Nombre	**Legajo**	**Mail**
SANSONI, Uriel	S-5494/1	urisansoni@gmail.com
FINA, Facundo	F-3628/5	facundofina02@gmail.com

22 de septiembre de 2025

Índice

1. Ejercicio 1

Partiendo del sistema de coordenadas inicial dado en la Fig. 1a, realizamos las rotaciones correspondientes al método **extrínseco**, es decir siempre rotando sobre los ejes x_0, y_0 y z_0 originales, obteniendo nuestro resultado final en la Fig. 1d



(a)
Sistema de coordenadas inicial

(b) $R_y(90°)$

(c) $R_x(-90°)R_y(90°)$

(d) $R_z(180°)R_x(-90°)R_y(90°)$

Fig. 1: Matriz rotacional

Podemos concluir entonces:

$$\xi = R_z(180°)R_x(-90°)R_y(90°) = \begin{array}{c} \begin{matrix} x & y & z \end{matrix} \\ \begin{bmatrix} 0 & 0 & -1 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \end{array} \tag{1}$$

2. Ejercicio 2

En este ejercicio debemos calcular la matriz resultante dada por los siguientes ángulos de Euler ($\alpha = \frac{4\pi}{7}, \beta = \frac{\pi}{2}, \gamma = \frac{-\pi}{3}$), utilizando la rotación intrínseca.

Para ello nos armamos las matrices de rotación sobre cada uno de los ejes y luego resolvemos premultiplicando.

$$Rotacion\ Roll,\ sobre\ el\ eje\ x \quad \rightarrow \quad R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix}$$

$$Rotacion\ Pitch,\ sobre\ el\ eje\ y \quad \rightarrow \quad R_y(\beta) = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix}$$

$$Rotacion\ Yaw,\ sobre\ el\ eje\ z \quad \rightarrow \quad R_z(\gamma) = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Resolviendo:

$R = R_x(\alpha)R_y(\beta)R_z(\gamma) =$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix} \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

$$= \begin{bmatrix} \cos\beta\ \cos\gamma & -\cos\beta\ \sin\gamma & \sin\beta \\ \sin\alpha\ \sin\beta\ \cos\gamma + \cos\alpha\ \sin\gamma & -\sin\alpha\ \sin\beta\ \sin\gamma + \cos\alpha\ \cos\gamma & -\sin\alpha\ \cos\beta \\ \cos\alpha\ \sin\beta\ \cos\gamma - \sin\alpha\ \sin\gamma & -\cos\alpha\ \sin\beta\ \sin\gamma - \sin\alpha\ \cos\gamma & \cos\alpha\ \cos\beta \end{bmatrix} \tag{3}$$

$$= \begin{bmatrix} \cos(\beta)\cos(\gamma) & -\cos(\beta)\sin(\gamma) & \sin(\beta) \\ \sin(\alpha+\gamma) & \cos(\alpha+\gamma) & -\sin(\alpha)\cos(\beta) \\ -\cos(\alpha+\gamma) & \sin(\alpha+\gamma) & \cos(\alpha)\cos(\beta) \end{bmatrix} \tag{4}$$

Reemplazando numéricamente:

$$R = R_x\left(\frac{4\pi}{7}\right) R_y\left(\frac{\pi}{2}\right) R_z\left(-\frac{\pi}{3}\right) = \begin{bmatrix} \cos(\frac{\pi}{2})\cos(-\frac{\pi}{3}) & -\cos(\frac{\pi}{2})\sin(-\frac{\pi}{3}) & \sin(\frac{\pi}{2}) \\ \sin\left(\frac{4\pi}{7}-\frac{\pi}{3}\right) & \cos\left(\frac{4\pi}{7}-\frac{\pi}{3}\right) & -\sin(\frac{4\pi}{7})\cos(\frac{\pi}{2}) \\ -\cos\left(\frac{4\pi}{7}-\frac{\pi}{3}\right) & \sin\left(\frac{4\pi}{7}-\frac{\pi}{3}\right) & \cos(\frac{4\pi}{7})\cos(\frac{\pi}{2}) \end{bmatrix}$$

$$R = R_x\left(\frac{4\pi}{7}\right) R_y\left(\frac{\pi}{2}\right) R_z\left(-\frac{\pi}{3}\right) = \begin{bmatrix} 0 & 0 & 1 \\ 0{,}680173 & 0{,}733052 & 0 \\ -0{,}733052 & 0{,}680173 & 0 \end{bmatrix}$$

Si ahora deseamos a partir de R calculada, reconocer los ángulos de Euler que obtuvieron dicha matriz de rotación, utilizando 4 podemos llegar al siguiente sistema de ecuaciones:

$$r_{13} = \sin(\beta) \quad \rightarrow \quad \beta = \arcsin(r_{13}) = \arcsin(1) = \frac{\pi}{2} \tag{5}$$

$$-\frac{r_{12}}{r_{11}} = -\frac{\cos(\beta)\sin(\gamma)}{\cos(\beta)\cos(\gamma)} = \tan(\gamma) \quad \rightarrow \quad \gamma = \arctan(-\frac{r_{12}}{r_{11}}) \tag{6}$$

$$-\frac{r_{23}}{r_{33}} = -\frac{\sin(\alpha)\cos(\beta)}{\cos(\alpha)\cos(\beta)} = \tan(\alpha) \quad \rightarrow \quad \alpha = \arctan(-\frac{r_{23}}{r_{33}}) \tag{7}$$

de 6 y 7, podemos deducir que, como en este caso particular r_{11}, r_{12}, r_{23}, y r_{33}, son iguales a 0 el sistema queda indeterminado. Por lo tanto no podemos obtener los valores de α y γ, esto es debido a que el angulo $\beta = \frac{\pi}{2}$ y $\cos(\frac{\pi}{2}) = 0$. Este fenómeno es denominado **Gimbal Lock**.

3. Ejercicio 3

El escenario planteado gráficamente lo podemos observar en la siguiente Figura:

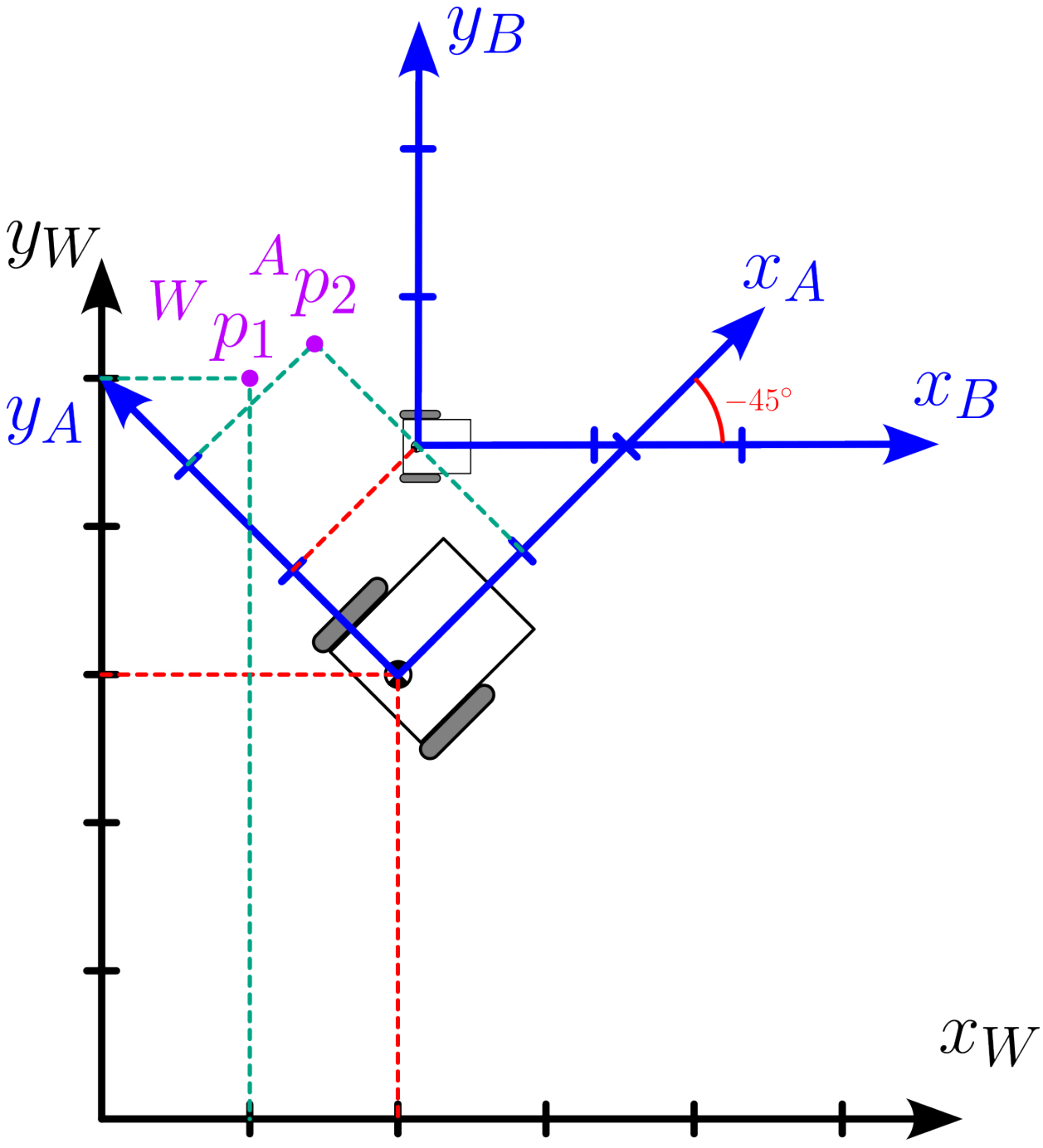


Fig. 2: Gráfico del escenario planteado

Para poder expresar el punto p_1 en el sistema de coordenadas del robot A, debemos aplicar la siguiente transformación:

$$\texttt{Transformación rototraslacional} \rightarrow {}^{W}\begin{bmatrix} x \\ y \end{bmatrix} = {}^{W}R_{A}\,{}^{A}\begin{bmatrix} x \\ y \end{bmatrix} + {}^{W}t_{A} \tag{8}$$

donde si realizamos el despeje obtenemos la ecuación 11

$$\Rightarrow {}^{W}\begin{bmatrix} x \\ y \end{bmatrix} - {}^{W}t_{A} = {}^{W}R_{A}\,{}^{A}\begin{bmatrix} x \\ y \end{bmatrix} \tag{9}$$

$$\Rightarrow {}^{W}R_{A}{}^{-1}\left({}^{W}\begin{bmatrix} x \\ y \end{bmatrix} - {}^{W}t_{A}\right) = {}^{A}\begin{bmatrix} x \\ y \end{bmatrix} \tag{10}$$

$$\Rightarrow {}^{A}\begin{bmatrix} x \\ y \end{bmatrix} = {}^{A}R_{W}\left({}^{W}\begin{bmatrix} x \\ y \end{bmatrix} + {}^{A}t_{W}\right) \tag{11}$$

Valores:

$${}^{A}R_{W} = {}^{A}\begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}_{W}, \qquad {}^{W}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 5 \end{bmatrix}, \qquad {}^{A}t_{W} = \begin{bmatrix} -2 \\ -3 \end{bmatrix} \tag{12}$$

Reemplazando los valores de la ec. 12 en la ec. 11, podemos calcular el punto visto en las nuevas coordenadas:

$$\Rightarrow {}^{A}\begin{bmatrix} x_{p_1} \\ y_{p_1} \end{bmatrix} = {}^{A}\begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}_{W}\left({}^{W}\begin{bmatrix} 1 \\ 5 \end{bmatrix} + \begin{bmatrix} -2 \\ -3 \end{bmatrix}\right) \tag{13}$$

$$\Rightarrow {}^{A}\begin{bmatrix} x_{p_1} \\ y_{p_1} \end{bmatrix} = {}^{A}\begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}_{W}\begin{bmatrix} -1 \\ 2 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2} \\ \frac{3\sqrt{2}}{2} \end{bmatrix} = \begin{bmatrix} 0{,}71 \\ 2{,}12 \end{bmatrix} \tag{14}$$

Análogamente para referir el punto p_2 al sistema de coordenadas del robot B, reutilizamos la ec 11 reemplazando los correspondientes nuevos valores:

$$\Rightarrow {}^{B}\begin{bmatrix} x \\ y \end{bmatrix} = {}^{B}R_{A}\left({}^{A}\begin{bmatrix} x \\ y \end{bmatrix} + {}^{B}t_{A}\right) \tag{15}$$

Valores:

$$ {}^{B}R_{A} = {}^{B}\begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}_{A}, \qquad {}^{A}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \qquad {}^{B}t_{A} = \begin{bmatrix} -1 \\ -1 \end{bmatrix} \tag{16} $$

Resolviendo:

$$ \Rightarrow {}^{B}\begin{bmatrix} x_{p_2} \\ y_{p_2} \end{bmatrix} = {}^{B}\begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}_{A} \left({}^{A}\begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} -1 \\ -1 \end{bmatrix} \right) \tag{17} $$

$$ \Rightarrow {}^{B}\begin{bmatrix} x_{p_2} \\ y_{p_2} \end{bmatrix} = {}^{B}\begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}_{A} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{bmatrix} = \begin{bmatrix} -0{,}71 \\ 0{,}71 \end{bmatrix} \tag{18} $$

Por último, si queremos saber cual es la pose del robot B con respecto al mundo, debemos realizar la transformación A $\rightarrow$ W de la ec 8:

$$ {}^{W}\begin{bmatrix} x_{_B} \\ y_{_B} \end{bmatrix} = {}^{W}R_{A} \, {}^{A}\begin{bmatrix} x_{_B} \\ y_{_B} \end{bmatrix} + {}^{W}t_{A} \tag{19} $$

Valores:

$$ {}^{W}R_{A} = {}^{B}\begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}_{A}, \qquad {}^{A}\begin{bmatrix} x_{_B} \\ y_{_B} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \qquad {}^{W}t_{A} = \begin{bmatrix} 2 \\ 3 \end{bmatrix} \tag{20} $$

Resolviendo:

$$ \Rightarrow {}^{W}\begin{bmatrix} x_{_B} \\ y_{_B} \end{bmatrix} = {}^{W}\begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}_{A} \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 0 \\ \sqrt{2} \end{bmatrix} + \begin{bmatrix} 2 \\ 3 \end{bmatrix} \tag{21} $$

$$ \Rightarrow {}^{W}\begin{bmatrix} x_{_B} \\ y_{_B} \end{bmatrix} = \begin{bmatrix} 2 \\ 3 + \sqrt{2} \end{bmatrix} = \begin{bmatrix} 2 \\ 4{,}41 \end{bmatrix} \tag{22} $$

Con respecto a la orientación del robot B en referencia al mundo, lo que debemos hacer es la suma de las orientaciones de ambos robots:

$$ \Rightarrow {}^{W}O_{B} = {}^{W}O_{A} + {}^{A}O_{B} = 45^{\circ} + (-45^{\circ}) = 0^{\circ} \tag{23} $$

Resultando la pose del motor B con respecto al mundo: ${}^{W}\begin{bmatrix} 2 \\ 4{,}41 \end{bmatrix}_{0^{\circ}}$

4. Ejercicio 4

Un esquema de la situación planteada se puede observar en la Figura 3:

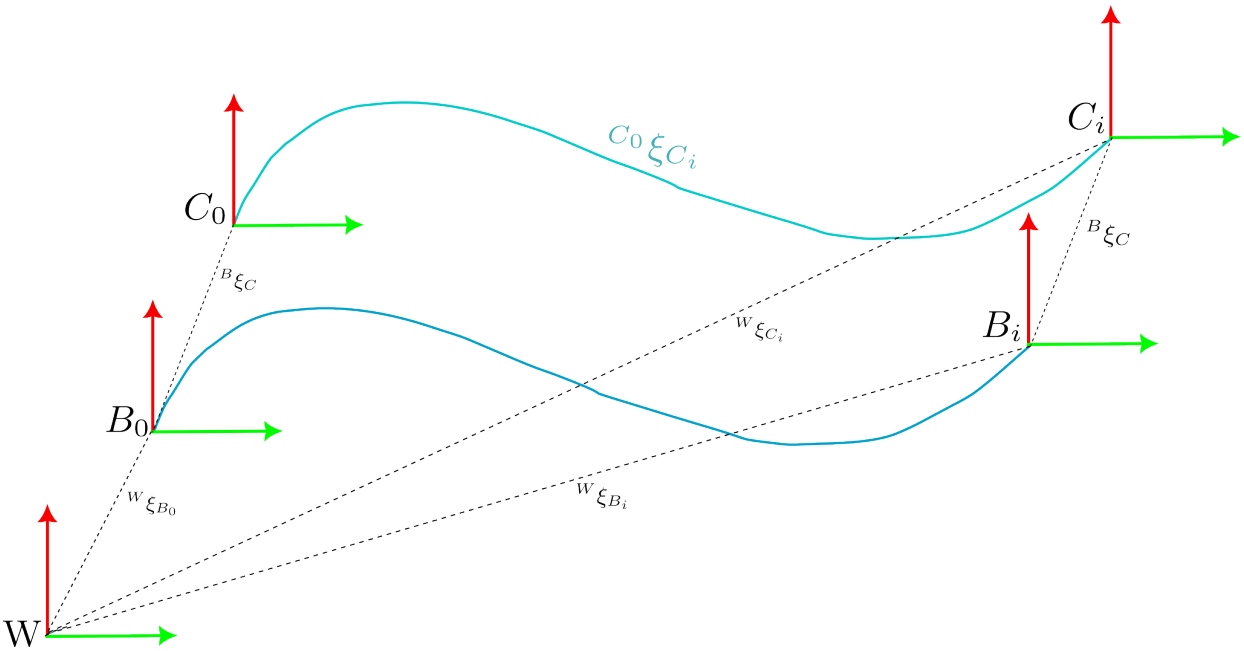


Fig. 3: Gráfico del escenario planteado

Para poder calcular la pose actual de la cámara con respecto al mundo (${}^{W}\xi_{C_i}$), como conocemos la pose inicial del robot en el mundo (${}^{W}\xi_{B_0}$) y la pose fija de la cámara en él (${}^{B}\xi_{C}$), dadas las poses de la cámara con respecto a la posición inicial misma (${}^{C_0}\xi_{C_i}$), debemos multiplicar cada una de estas transformaciones:

$$ {}^{W}\xi_{C_i} = {}^{W}\xi_{B_0}\, {}^{B}\xi_{C}\, {}^{C_0}\xi_{C_i} \qquad ... \qquad \forall i \tag{24} $$

Una vez teniendo la pose actual de la cámara en el mundo, si queremos conocer la pose actual del robot con respecto al mundo (${}^{W}\xi_{B_i}$), debemos multiplicar por la inversa de la transformación B $\rightarrow$ C, la cual es fija.

$$ {}^{W}\xi_{B_i} = {}^{W}\xi_{C_i}({}^{B}\xi_{C})^{-1} \tag{25} $$

$$ {}^{W}\xi_{B_i} = {}^{W}\xi_{B_0}\, {}^{B}\xi_{C}\, {}^{C_0}\xi_{C_i}\, {}^{C}\xi_{B} \qquad ... \qquad \forall i \tag{26} $$

5. Ejercicio 5

Mediante el *ground-truth* del archivo `mav0/state_groundtruth_estimate0/data.csv` que otorga la pose del robot con respecto al mundo, y en conjunto con la transformación ${}^{B}\xi_{C}$ de IMU a la cam0 (cámara izquierda) pudimos crear un script de Python que genere la trayectoria de la cámara.

Dicho script calcula la pose de la cámara izquierda en cada iteración dentro de un bucle for, el mismo se detalla a continuación:

```
# leemos el dataset de la IMU
trayectoria_csv = pd.read_csv("data.csv")

# leemos matriz de transformacion de la cam0 con repecto a la IMU
with open("sensor.yaml", "r") as f:
    sensor_data = yaml.safe_load(f)
T_data = sensor_data["T_BS"]["data"]

# pose de la cam0 con respecto a la IMU
T_BC = np.array(T_data).reshape((4,4))

# abrimos el grafico
plt.ion()
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# escala para los vectores
scale = 0.25
xs, ys, zs = [], [], []

for i in range(len(trayectoria_csv["#timestamp"]):
```

dentro del bucle leemos la posición actual de la IMU, y su orientación (matriz de rotación) la obtenemos desde cuaterniones mediante la funcion `quat2mat` de la libreria transform3d:

```
    t_ns = trayectoria_csv["#timestamp"].iloc[i]
    #item 5b) timestamp en segundos
    t_s = t_ns*1e-9
    x = trayectoria_csv[" p_RS_R_x [m]"].iloc[i]
    y = trayectoria_csv[" p_RS_R_y [m]"].iloc[i]
    z = trayectoria_csv[" p_RS_R_z [m]"].iloc[i]
```

```
qw = trayectoria_csv[" q_RS_w []"].iloc[i]
qx = trayectoria_csv[" q_RS_x []"].iloc[i]
qy = trayectoria_csv[" q_RS_y []"].iloc[i]
qz = trayectoria_csv[" q_RS_z []"].iloc[i]

# Convertimos el cuaternion a matriz de rotacion 3x3
R = quat2mat([qw, qx, qy, qz])

r = np.array([
    [x],
    [y],
    [z]
])

# pose de la IMU con respecto al mundo
T_WB = np.block([
    [R, r],
    [np.zeros((1,3)), np.ones((1,1))]
])
```

Aplicamos la transformacion ${}^{B}\xi_{C}$:

```
# pose de la cam0 con respecto al mundo
T_WC = T_WB @ T_BC
```

Una vez obtenida la trayectoria de la cámara solo resta graficar, para ello hicimos una simulación del movimiento de la misma, una de esta se puede observar en la Figura 4

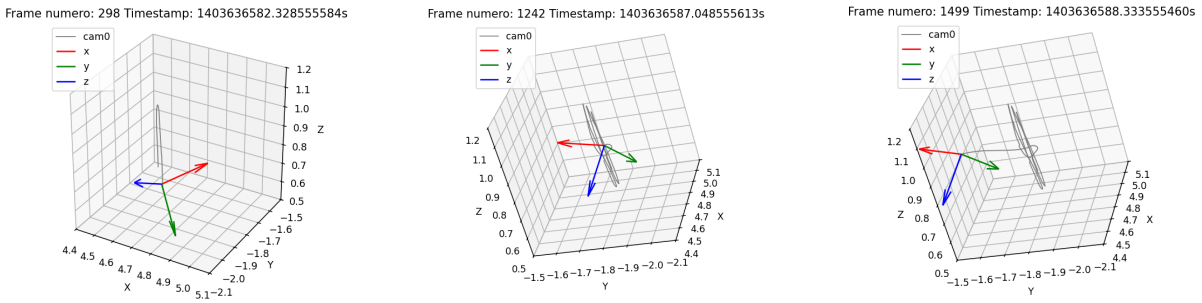


Fig. 4: Trayectoria de la cámara izquierda (cam0)

Esta animación de gráficos fue creada con las funciones `plot` y `quiver`, como detalla el siguiente script:

```
 X=T_WC[0,3]
 Y=T_WC[1,3]
 Z=T_WC[2,3]

 # acumulamos posiciones para marcar la trayectoria
 xs.append(X)
 ys.append(Y)
 zs.append(Z)

 # limpiamos vectores de frames anteriores
 ax.cla()

 # trayectoria acumulada
 ax.plot(xs, ys, zs, color='gray', linewidth=1, label='cam0')

 # dibujamos las flechas
 ax.quiver(X, Y, Z, T_WC[0,0], T_WC[1,0], T_WC[2,0], color='r',
length=scale, label='x')
 ax.quiver(X, Y, Z, T_WC[0,1], T_WC[1,1], T_WC[2,1], color='g',
length=scale, label='y')
 ax.quiver(X, Y, Z, T_WC[0,2], T_WC[1,2], T_WC[2,2], color='b',
length=scale, label='z')
```

Para comparar el ground-truth original con el obtenido, de la pose de la cámara T_WC extrajimos la matriz de rotación R y la convertimos a cuaterniones con la función `mat2quad`, también extrajimos las coordenadas X Y y Z de su ultima columna. En la tabla 1 y 2 se puede realizar la comparación de los datos imprimidos por el progrma.

```
 # Convertimos La matriz de rotacion 3x3 a cuaternion
 # Extraemos R (rotacion 3x3)
 R_WC = T_WC[0:3, 0:3]
 Q = mat2quat(R_WC)
 QW, QX, QY, QZ= Q

 # Imprimimos nuevos datos
 print(f"#timestamp: {t_s} x: {x} y: {y} z: {z} qw: {qw} qx: {qx
} qy: {qy} qz: {qz}")
 print(f"#timestamp: {t_s} x: {X} y: {Y} z: {Z} qw: {QW} qx: {QX
} qy: {QY} qz: {QZ}")
```

indice	#timestamp [ns]	x	y	z	qw	qx	qy	qz
1	14036365808385558	4.688319	-1.786938	0.783338	0.534108	-0.153029	-0.827383	-0.082152
2	14036365808435555	4.688177	-1.78677	0.78735	0.53464	-0.15299	-0.826976	-0.082863
3	14036365808485556	4.688028	-1.786598	0.791382	0.535178	-0.152945	-0.826562	-0.083605
4	14036365808535554	4.687878	-1.786421	0.795429	0.535715	-0.152884	-0.826146	-0.084391
5	14036365808585558	4.687727	-1.78624	0.799484	0.536244	-0.152821	-0.825731	-0.085213

Tabla 1: Ground-truth original (IMU)

indice	#timestamp [s]	x	y	z	qw	qx	qy	qz
1	1403636580.8385558	4.666121	-1.848356	0.761367	0.445604	-0.692875	-0.475716	0.308311
2	1403636580.8435555	4.665910	-1.848140	0.765315	0.446477	-0.692558	-0.475443	0.308182
3	1403636580.8485556	4.665690	-1.847918	0.769279	0.447377	-0.692231	-0.475168	0.308034
4	1403636580.8535554	4.665469	-1.847688	0.773252	0.448307	-0.691891	-0.474902	0.307855
5	1403636580.8585558	4.665244	-1.847452	0.777229	0.449257	-0.691550	-0.474639	0.307644

Tabla 2: Ground-truth obtenido (cam0)

Por ultimo, graficamos ambos ground-truths (IMU y cam0) en un mismo gráfico, como se puede ver el la Figura 5.

```
# trayectorias acumuladas
ax.plot(XS, YS, ZS, color='gray', linewidth=1, label='cam0')
ax.plot(xs, ys, zs, color='orange', linewidth=1, label='IMU')

# dibujamos las flechas
ax.quiver(X, Y, Z, T_WC[0,0], T_WC[1,0], T_WC[2,0], color='r', length=scale, label='x')
ax.quiver(X, Y, Z, T_WC[0,1], T_WC[1,1], T_WC[2,1], color='g', length=scale, label='y')
ax.quiver(X, Y, Z, T_WC[0,2], T_WC[1,2], T_WC[2,2], color='b', length=scale, label='z')

ax.quiver(x, y, z, R[0,0], R[1,0], R[2,0], color='r', length=scale)
ax.quiver(x, y, z, R[0,1], R[1,1], R[2,1], color='g', length=scale)
ax.quiver(x, y, z, R[0,2], R[1,2], R[2,2], color='b', length=scale)
```

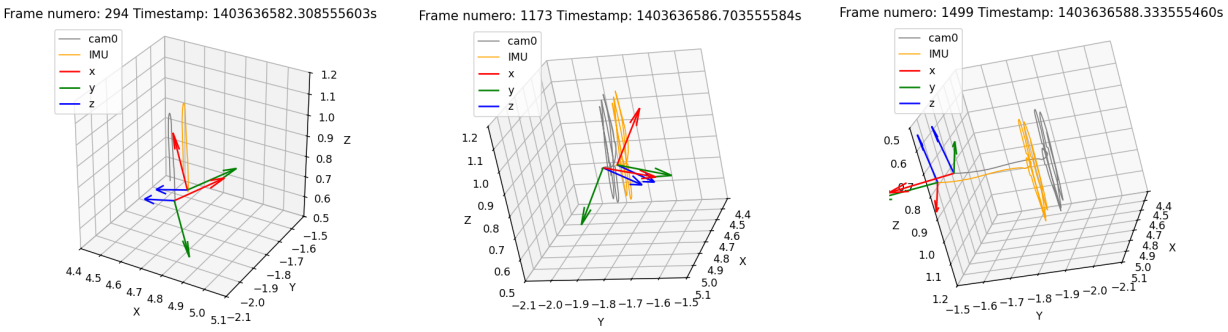


Fig. 5: Trayectoria de la cámara izquierda (cam0) y la IMU (Body)

6. Repositorio github

`https://github.com/urisanso/ScriptsTP1_FINA_SANSONI`