Robótica: Ejercicio 1

Año académico 2021-2022

Autor:	ID:	Email:
Barcelona Auria, Federico	666151	unizar@federico.barcelona

En el código desarrollado, la localización se representa con la clase Location. Esta clase utiliza internamente la matriz homogénea para realizar las transformaciones, por lo que no se han desarrollado funciones de loc() y hom() explícitas.

Todas las transformaciones se hacen internamente en los métodos de las clases `Location`, `Direction` y `Point`.

1. Calcular la localización del robot en W, si la localización relativa entre puerta y robot es ${}^{R}x_{P}$=(5.2,-3,-125gr).

   Teniendo en cuenta que la localización de la puerta en el mundo es ${}^{W}x_{P}$=(10, 2.5, 0gr), podemos calcular ${}^{W}x_{R}$ mediante ${}^{W}T_{P} * {}^{P}T_{R}$.
   Para calcular ${}^{P}T_{R}$, es necesario hacer la inversa de ${}^{R}T_{P}$, que se obtiene de la localización ${}^{R}X_{P}$.

```
door_in_world = Location.from_angle_degrees(Point(10, 2.5), 0)
door_seen_from_robot = Location.from_angle_degrees(Point(5.2, -3), -125)

robot_seen_from_door = door_seen_from_robot.inverse()
robot_in_world = robot_seen_from_door.seen_from_other_location(door_in_world)
```

```
########################################
Apartado 1
robot_in_world: (x: 10.525141336158466, y: -3.480319939355897, deg: 125.00000000000001, rad: 2.181661564992912)
########################################
```

2. ¿Cuál es la localización del cuadro C en W, cuando el robot lo ve en ${}^{R}x_{C}$=(4.17,0.73,-35gr)?

   Este apartado se resuelve de la misma forma que el anterior, pero utilizando la fórmula ${}^{W}T_{C}={}^{W}T_{R} * {}^{R}T_{C}$.

```
painting_seen_from_robot = Location.from_angle_degrees(Point(4.17, 0.73), -35)
painting_in_world = painting_seen_from_robot.seen_from_other_location(robot_in_world)
```

```
########################################
Apartado 2
painting_in_world: (x: 7.535346604243639, y: -0.4831667132070647, deg: 90.00000000000001, rad: 1.5707963267948968)
########################################
```

3. ¿Cuál es la localización del cuadro respecto a la puerta?

Para obtener la localización relativa del cuadro respecto de la puerta, se puede utilizar la localización de ambos en el mundo.
De esta forma, obtendremos ${}^{P}T_{C}={}^{P}T_{W} * {}^{W}T_{C}$, donde ${}^{P}T_{W}$ se obtiene con la inversa de ${}^{W}T_{P}$ ya provista por la localización ${}^{W}x_{P}$ en el dibujo.

```
world_seen_from_door = door_in_world.inverse()
painting_seen_from_door = painting_in_world.seen_from_other_location(world_seen_from_door)

########################################
Apartado 3
painting_seen_from_door: (x: -2.464653395756361, y: -2.9831667132070647, deg: 90.00000000000001, rad: 1.5707963267948968)
########################################
```

4. Calcular la velocidad lineal y angular que habrá que aplicar al robot para alcanzar la puerta siguiendo una trayectoria circular en 8 segundos. ¿Cuál será la localización final relativa a la localización inicial? ¿Cuál será la localización del robot en W?

El radio de curvatura (R) se calcula mediante las coordenadas de la localización relativa del robot con la puerta:

$$R = \frac{x^2+y^2}{2y}$$

```
def radious_of_curvature(self):
    return (self.origin.x ** 2 + self.origin.y ** 2) / (2.0 * self.origin.y)
```

El ángulo de curvatura, se calcula mediante la fórmula:

$$\Theta = atan2(2xy,\ x^2 - y^2)$$

```
def angle_of_curvature(self):
    return math.atan2(self.origin.x * self.origin.y * 2, self.origin.x ** 2 - self.origin.y ** 2)
```

La velocidad angular se calcula mediante la fórmula:

$$\omega = \frac{\Theta}{t}$$

```
def angular_velocity_to_arrive_in(self, duration: datetime.timedelta):
    return self.angle_of_curvature() / duration.seconds
```

Y la velocidad lineal se calcula mediante la fórmula:

$$v = \omega R$$

```
def linear_velocity_to_arrive_in(self, duration: datetime.timedelta):
    return self.angular_velocity_to_arrive_in(duration) * self.radious_of_curvature()
```

Para calcular la nueva posición final relativa a la inicial, se calcula primero el punto en el que va a quedar el robot:

$$x_k = R\,sin(\Theta_k);\; y_k = R(1 - cos(\Theta_k);$$

El ángulo será el propio de la localización relativa del movimiento, por lo que la localización final respecto a la inicial será:

```
def new_location_after_rotating(self, movement_ratio=1) -> 'Location':
    angle_of_curvature = self.angle_of_curvature() * movement_ratio
    new_x_position = self.radious_of_curvature() * math.sin(angle_of_curvature)
    new_y_position = self.radious_of_curvature() * (1 - math.cos(angle_of_curvature))
    return Location.from_angle_radians(Point(new_x_position, new_y_position), angle_of_curvature)
```

Finalmente, para calcular la localización final respecto del mundo, se puede utilizar la fórmula ${}^{W}T_{RR}={}^{W}T_{R} * {}^{R}T_{RR}$, donde ${}^{R}T_{RR}$ es la nueva localización calculada con `new_location_after_rotating()`.

```
print(f'radius of curvature: {door_seen_from_robot.radius_of_curvature()}')
print(f'angle of curvature: {math.degrees(door_seen_from_robot.angle_of_curvature())} degrees')
print(f'angular velocity: {door_seen_from_robot.angular_velocity_to_arrive_in(datetime.timedelta(seconds=8))}')
print(f'linear velocity: {door_seen_from_robot.linear_velocity_to_arrive_in(datetime.timedelta(seconds=8))}')

new_robot_seen_from_robot = door_seen_from_robot.new_location_after_rotating()
print(f'new_robot_seen_from_robot: {print_location(new_robot_seen_from_robot)}')

new_robot_in_world = new_robot_seen_from_robot.seen_from_other_location(robot_in_world)
print(f'new_robot_in_world: {print_location(new_robot_in_world)}')


##########################################
Apartado 4
radius of curvature: -6.006666666666668
angle of curvature: -59.963278737698666 degrees
angular velocity: -0.13081958053299386
linear velocity: 0.7857896137348499
new_robot_seen_from_robot: (x: 5.2, y: -3.0, deg: -59.963278737698666, rad: -1.046556644263951)
new_robot_in_world: (x: 10.0, y: 2.5, deg: 65.03672126230134, rad: 1.1351049207289612)
##########################################
```

5. ¿Cuál será la localización del robot respecto a su situación inicial transcurridos 4 segundos de haber iniciado el movimiento? ¿Cuál será la localización del robot en W?

Dado que el recorrido del robot es continuo, a los 4 segundos se habrá recorrido la mitad de distancia. Esto es equivalente a calcular el recorrido con la mitad del ángulo:

```
new_robot_seen_from_robot_with_half_rotation = door_seen_from_robot.new_location_after_rotating(movement_ratio=0.5)
print(f'new_robot_seen_from_robot_with_half_rotation: {print_location(new_robot_seen_from_robot_with_half_rotation)}')

new_robot_in_world_with_half_rotation = new_robot_seen_from_robot_with_half_rotation.seen_from_other_location(
    robot_in_world)
print(f'new_robot_in_world_with_half_rotation: {print_location(new_robot_in_world_with_half_rotation)}')


########################################
Apartado 5
new_robot_seen_from_robot_with_half_rotation: (x: 3.001666203960727, y: -0.803778579801407, deg: -29.981639368849333, rad: -0.5232783221319754)
new_robot_in_world_with_half_rotation: (x: 9.461873198575324, y: -0.5604704786904935, deg: 95.01836063115068, rad: 1.6583832428609366)
########################################
```