

# **RESUMEN TEORÍA MODELADO Y SIMULACIÓN DE ROBOTS:**

## **ÍNDICE DE CONTENIDOS:**

### **TEMA 1: INTRODUCCIÓN**

- 1.1 INTRODUCCIÓN**
- 1.2 UTILIDAD DE UN SIMULADOR**
- 1.3 GEMELOS DIGITALES**
- 1.4 SIMULADOR**
- 1.5 SIMULADORES PARA ROBOTS**

### **TEMA 2: MOTORES DE FÍSICAS**

- 2.1 MOTORES DE FÍSICAS**
- 2.2 DINÁMICA Y LEYES DE NEWTON**
- 2.3 SISTEMA DE REFERENCIA INERCIAL**
- 2.4 CUANTIZACIÓN**
- 2.5 TIPOS DE MOTORES DE FÍSICAS**
- 2.6 TRASLACIÓN, ORIENTACIÓN Y ROTACIÓN**
- 2.7 ÁNGULOS DE EULER**
- 2.8 CUATERNIONES**
- 2.9 FUERZAS SOBRE OBJETOS**

### **TEMA 3: URDF (UNIFIED ROBOT DESCRIPTION FORMAT)**

- 3.1 URDF**
- 3.2 XML**
- 3.3 JOINTS**
- 3.4 SDF**

### **TEMA 4: CONTROLANDO UN ROBOT EN PYBULLET**

- 4.1 VELOCIDAD**
- 4.2 MOMENTO DE FUERZA**
- 4.3 FRICCIÓN**
- 4.4 RESTITUCIÓN**
- 4.5 INERCIA**
- 4.6 MOMENTO ANGULAR VS FUERZA DE TORQUE**

### **TEMA 5: MOTOR GRÁFICO**

- 5.1 REPRESENTACIÓN GRÁFICA Y ESCENAS 3D**
- 5.2 REPRESENTACIÓN DE OBJETOS**
- 5.3 RENDERIZADO (DEFINICIÓN Y TIPOS)**
- 5.4 MODELO PIN-HOLE DE CÁMARA**
- 5.5 OPTIMIZACIONES**
- 5.6 RASTERIZADO, RAY-TRACING Y PATH-TRACING**

### **TEMA 6: CINEMÁTICA INVERSA**

- 6.1 CINEMÁTICA DIRECTA E INVERSA**
- 6.2 SINGULARIDAD**
- 6.3 JACOBIANO INVERSO**
- 6.4 MÍNIMOS CUADRADOS AMORTIGUADOS**
- 6.5 END EFFECTOR**

## **TEMA 7: END EFFECTORS**

### **7.1 END EFFECTOR**

### **7.2 PRISMATIC**

### **7.3 GEOMETRÍA ACKERMANN**

### **7.4 EJE MOTOR**

## **TEMA 8: BLENDER**

### **8.1 BLENDER**

### **8.2 PIPELINE RENDER**

### **8.3 FORMATOS**

### **8.4 INSTALACIÓN DEL ENTORNO**

## **TEMA 9: SIM2REAL**

### **9.1 SIM2REAL**

### **9.2 DATA AUGMENTATION**

### **9.3 DOMAIN RANDOMIZATION**

### **9.4 TRANSFER LEARNING**

### **9.5 FINE-TUNING**

## **TEMAS 1: INTRODUCCIÓN**

### **1.1 INTRODUCCIÓN**

**Modelado de un robot:** Permite modelar el esquema general del robot desde diferentes puntos de vista (modelado 3D, dinámico, cinemático y de sensores y actuadores).

**Simulador de robots:** Ayuda a testear el modelo del robot en un entorno virtual de forma "offline".

**Robots industriales:** Realizan tareas repetitivas, poseen brazos articulados con 3 o más grados de libertad y no necesitan demasiado contexto del mundo que les rodea.

**Robots industriales (disciplinas):** Físicas, mecánicas, teoría de control, microcontroladores, adquisición de datos, procesamiento de señales y sistemas de tiempo real.

**Robots móviles:** Realizan mucho más que tareas repetitivas, poseen un comportamiento inteligente, son capaces de aprender y adquieren conocimiento del mundo que les rodea.

**Robots móviles (disciplinas):** Las mismas que los robots industriales, además de comunicaciones, visión computacional, programación distribuida / concurrente, planificación de tareas y Machine / Deep Learning.

### **1.2 UTILIDAD DE UN SIMULADOR**

**Necesidad:** No siempre se puede acceder al robot, a su entorno o a ambos. Son necesarios para entrenar, mejorar o probar nuevas funcionalidades antes de salir a producción y también para experimentar y testear algoritmos cinemáticos sin poseer conocimientos previos, lo que se traduce en un ahorro de tiempo y dinero.

**Ventajas:** Permiten iterar más rápidamente en el robot y su algoritmo para realizar mejoras, donde el aumento de algoritmos basados en aprendizaje automático permite realizar multitud de simulaciones hasta encontrar los parámetros óptimos (conducción autónoma y aprender a andar).

### **1.3 GEMELOS DIGITALES**

**Gemelo Digital:** Representación o modelo virtual con precisión de un componente físico (robot, coche, turbina o actuador) que tiene como objetivo realizar pruebas e investigaciones para después pasarlas al objeto físico y así poder comunicarse con su otro gemelo (no confundir gemelo digital con simulador).

**Ventajas:** I+D (mejor y más eficiente investigación y desarrollo), **mayor eficiencia** (refleja y supervisa los sistemas de producción), anticipación de los problemas antes de que ocurran en el producto físico y toma de decisiones basada siempre en datos.

### **1.4 SIMULADOR**

**Simulador:** Componente físico o virtual que reproduce un comportamiento determinado de los objetos y de su entorno con el objetivo de proporcionar una experiencia lo más cercana posible a la realidad en un entorno seguro y controlado. Están basados en modelos matemáticos, físicos y computacionales que recrean situaciones específicas y que cada vez son más utilizados en industrias gracias a su aporte y valor.

**Tipos:** Vuelo, médicos, conducción, militares, realidad virtual, negocio, educativo, deportivo, desastres y procesos industriales.

**Características:** Posee un render realista, simula cuerpos rígidos, motor de físicas, modela entornos, mundos, robots reales, sensores y actuadores, se integra con middlewares externos e interacciona con el hardware.

**Simuladores Open Source:** Gazebo, CARLA, Webots, Mujoco, PyBullet, Pyro, Simbad, Airsim, OpenSim, UsarSim, OpenRave, Blender y EyeSim.

**Simuladores Propietarios:** Marilou, Cogmation, Ms Robotics Studio, Modelica, Robologix, Simulink, MATLAB y EasyRob.

### **1.5 SIMULADORES PARA ROBOTS**

**Gazebo Garden (Gazebo Transport):** Sensores, **físicas** (DART, Dynamic Animation and Robotics Toolkit) y **Render** (OGRE y OptiX de NVIDIA).

**Gazebo:** Conducción autónoma en coches de carreras, Reinforcement / Deep learning y DeepRacer.

**CARLA:** Simulador utilizado para conducción autónoma Open Source muy extendido en investigación.

**AirSim:** Simulador multiplataforma de código abierto para drones y vehículos terrestres como automóviles basado en drone exploration y Reinforcement Learning.

**Otros:** UWSim, Trick (NASA), RoboDK y ROAMS (JPL NASA).

## TEMA 2: MOTORES DE FÍSICAS

### 2.1 MOTORES DE FÍSICAS

**Motor de físicas:** Software que simula las leyes físicas en un entorno virtual proporcionando algoritmos para imitar aspectos del mundo real como la gravedad, la colisión de objetos, la dinámica de fluidos, y otras fuerzas y reacciones físicas como la dinámica de cuerpos rígidos (colisiones) y blandos, el movimiento de un fluido y la elasticidad.

**Real Time:** Utiliza modelos simplificados para poder simular en tiempo real las físicas del escenario reduciendo exactitud y precisión (videojuegos y otro tipo de simuladores interactivos).

**High Precision:** Requieren un mayor procesamiento para calcular las físicas de forma precisa (área científico).

**Usos:** **Científico** (modelado de patrones meteorológicos, diseño de aviones y simulación de nuevos materiales en ruedas de coches) y **videojuegos** (detección de colisiones, cuerpos rígidos / blandos y la velocidad prevalece sobre la exactitud y la precisión).

**Limitaciones:** No puede presentar objetos utilizando puntos / texturas / fuerzas infinitas sobre el objeto, hay errores de redondeo si la precisión es baja y las fluctuaciones no modeladas pueden cambiar el resultado.

### 2.2 DINÁMICA Y LEYES DE NEWTON

**Dinámica:** Parte de la física que estudia las relaciones de movimiento por los movimientos de los cuerpos y las causas que los provocan (concretamente las fuerzas que actúan sobre ellos), por lo que es necesaria la observación de un escenario para poder entender los movimientos que se producen, donde un objeto sólo cambia de posición si se aplica una fuerza sobre él.

**Primera Ley de Newton (Ley de Inercia):** Todo cuerpo persevera en su estado de reposo o movimiento uniforme y en la misma dirección y velocidad a no ser que sea obligado a cambiar su estado por fuerzas aplicadas sobre él.

$$\Sigma F = 0 \Leftrightarrow dv/dt = 0$$

**Segunda Ley de Newton (Ley Fundamental de la Dinámica):** El cambio de movimiento es directamente proporcional a la fuerza motriz aplicada y ocurre según la línea recta a lo largo de la cual la fuerza se imprime.

**Momento:**  $p = mv$  y  $F = dp/dt$  ; **Principio de superposición de fuerzas:**  $F = \sum_{i=1}^K F_i$

**Tercera Ley de Newton (Principio de Acción y Reacción):** Con toda acción ocurre siempre una reacción igual y contraria, donde las acciones mutuas de dos cuerpos siempre son iguales y dirigidas en sentido opuesto.

### 2.3 SISTEMA DE REFERENCIA INERCIAL

**Sistema de referencia inercial:** Aquel en el que un cuerpo permanece en reposo o en movimiento rectilíneo uniforme si no actúan fuerzas externas sobre él (trenes a velocidad constante y satélites en órbita estable).

**Sistema de referencia no inercial:** Incluye fuerzas ficticias para poder aplicar las leyes de Newton ya que está sometido a aceleraciones o giros (automóvil, montaña rusa, avión, fuerza centrífuga / de Coriolis e inercia).

**Tierra:** Sistema no perfectamente inercial debido a que orbita alrededor del sol, provocando una aceleración que cambia la velocidad y el movimiento, además de la fuerza centrífuga que va hacia afuera desde el eje de rotación, y la Fuerza de Coriolis, que afecta a la dirección del movimiento, la cual es medible en escalas de tiempo y espacios grandes y afecta a corrientes marinas y otros fenómenos meteorológicos (huracanes).

### 2.4 CUANTIZACIÓN

**Cuantización:** Proceso utilizado para reducir la precisión de las representaciones numéricas y los cálculos para mejorar el rendimiento computacional.

**Características:** Limita el rango de todos los posibles valores que una variable puede representar (discreto y finito), un alto grado de cuantización lleva a errores en la simulación y a comportamientos físicos incorrectos y permite ajustar el equilibrio entre la exactitud de la simulación y la carga computacional del sistema.

**Physics Processing Unit (PPU):** Microprocesador / Acelerador hardware dedicado y especializado en cálculos de físicas usado en proyectos especializados. Por otro lado, las GPUs están especializadas en operaciones gráficas y dan buen rendimiento aunque no son óptimas para las físicas.

**Problemas NP-hard:** Complejidad de la simulación de contactos dinámicos en cada paso de la simulación (aplicación de heurísticos y optimización convexa).

## } 2.5 TIPOS DE MOTORES DE FÍSICAS

**ODE (Open Dynamics Engine):** Soporta modelos de colisiones y de cuerpos rígidos y posee algunas limitaciones en su modelo de fricción y en su soporte para joint-damping (amortiguación).

**Bullet:** Provee binding para Python, simula detección de colisiones y dinámica de cuerpos rígidos y blandos e incorpora diferentes primitivas de colisión.

**DART (Dynamic Animation and Robotics Toolkit):** Librería desarrollada en C++ que ofrece estructuras y algoritmos cinemáticos y dinámicos para cuerpos rígidos y blandos, proporciona estabilidad y exactitud para representar articulaciones en cuerpos rígidos, está enfocado en la optimización de operaciones y soporta ODE y Bullet para detectores de colisión.

**Symbody:** Motor de alta calidad para simular mecanismos articulados como estructuras y esqueletos.

**Unity:** Motor de videojuego multiplataforma creado por Unity Technologies, no es open-source pero sí friendly-community, permite obtener código como referencia y usa PhysX de NVIDIA como motor de físicas.

**Unreal:** Motor de videojuegos creado por Epic Games, es open-source aunque no permite la comercialización de sus productos de software sin una licencia, posee motores de renderizado y de físicas, además de sistemas de animación e IA, editores de mundos, red y multijugador, y usa PhysX de NVIDIA como motor de físicas, el cual va migrando poco a poco a Chaos.

**PhysX de NVIDIA:** Motor privativo de físicas creado por NVIDIA diseñado para permitir interacciones realistas y simulaciones físicas en tiempo real dentro de videojuegos y aplicaciones con gráficos 3D, es capaz de simular cuerpos rígidos, fluidos, telas, dinámicas de vehículos y partículas, detección de colisiones, es multiplataforma y óptimo para el hardware NVIDIA.

## } 2.6 TRASLACIÓN, ORIENTACIÓN Y ROTACIÓN

**Traslación:** Definida por las proyecciones del vector de posición del punto sobre cada uno de los ejes (x,y,z).

**Orientación:** Conjunto de posibles elecciones para colocar el objeto sin cambiar un punto fijo de referencia.

**Rotación:** Movimiento de cambio de orientación de un cuerpo o un sistema de referencia de forma que una línea (eje de rotación) permanezca fija.

## } 2.7 ÁNGULOS DE EULER

**Ángulos de Euler:** Conjunto de tres coordenadas angulares ( $\alpha, \beta, \gamma$ ) que sirven para especificar la orientación de un sistema de referencia de ejes ortogonales móvil respecto a otro fijo.

**Bloqueo de Cardán (Gimbal Lock):** Pérdida de un grado de libertad en un espacio 3D que ocurre cuando 2 de los 3 gimbals están alineados en paralelo generando un espacio de movimiento en 2D (triple-roll wrist).

## } 2.8 CUATERNIONES

**Cuaterniones:** Proporcionan una notación matemática extendida de los números complejos para representar orientaciones y rotaciones de objetos 3D, son más simples y evitan el Bloqueo de Cardán sin ser tan intuitivos en comparación con los ángulos de Euler.

**Usos:** Aplicaciones de gráficos por ordenador, robótica, navegación y mecánica orbital de satélites.

**Cuaternión (fórmula):**  $q = w + xi + yj + zk$ , dados los ángulos de Euler  $\alpha$  (roll),  $\beta$  (pitch) y  $\gamma$  (yaw).

**Componente w:**  $w = \cos(\alpha/2) * \cos(\beta/2) * \cos(\gamma/2) + \sin(\alpha/2) * \sin(\beta/2) * \sin(\gamma/2)$

**Componente x:**  $x = \sin(\alpha/2) * \cos(\beta/2) * \cos(\gamma/2) - \cos(\alpha/2) * \sin(\beta/2) * \sin(\gamma/2)$

**Componente y:**  $y = \cos(\alpha/2) * \sin(\beta/2) * \cos(\gamma/2) + \sin(\alpha/2) * \cos(\beta/2) * \sin(\gamma/2)$

**Componente z:**  $z = \cos(\alpha/2) * \cos(\beta/2) * \sin(\gamma/2) - \sin(\alpha/2) * \sin(\beta/2) * \cos(\gamma/2)$

## } 2.9 FUERZAS SOBRE OBJETOS

**Gravedad:** Causa una aceleración en los objetos hacia el centro de la Tierra ( $9.8 \text{ m/s}^2$ ), varía con la altitud, es despreciable para las simulaciones y no depende de la forma / peso / altura a la que está el objeto.

**Aceleración:** Magnitud física vectorial que indica la variación de velocidad por unidad de tiempo, donde la aceleración de un cuerpo es proporcional a la fuerza que actúa sobre él.

**Aceleración:**  $F = ma \Rightarrow a = F/m$  ; **Aceleración media:**  $\bar{a} = \Delta v / \Delta t$

**Momento de Fuerza / Torque:** Pseudomagnitud vectorial generada al aplicar una fuerza sobre un objeto realizando una rotación sobre alguno de sus ejes ( $T = r \times F$ ).

**Fricción:** Fuerza que existe entre dos superficies de contacto opuestas al deslizamiento, donde la fuerza de rozamiento tiene una dirección paralela a la superficie de apoyo y el coeficiente de rozamiento depende exclusivamente de la naturaleza de los cuerpos en contacto.

**Fricción Estática:** Resistencia a superar para poner un cuerpo en movimiento respecto a otro en contacto.

**Fricción Dinámica:** Resistencia constante opuesta al movimiento una vez comenzado (modelo de Coulomb).

## **TEMAS 3: URDF (UNIFIED ROBOT DESCRIPTION FORMAT)**

### **3.1 URDF**

**URDF:** Especificación XML capaz de describir un modelo de un robot con la que se pueden representar descripciones cinemáticas / físicas, representaciones visuales y modelos de colisión, todo ello en un formato mayormente usado para la creación de modelos de robots.

**Visores:** RViz, Blender, PyBullet y mymodelrobot.

**Limitaciones:** Modelado de dinámicas complejas, simulación de sensores, descripción de entornos y definición de eventos ambientales (viento / luz).

### **3.2 XML**

**Modelo visual y de colisión (elementos):** Cajas, cilindros, esferas (más eficientes computacionalmente) y mallas / meshes (superficies creadas mediante un método 3D generado por sistemas de vértices posicionados en un espacio virtual más precisas para el modelo).

**<robot>:** Ofrece una descripción completa del robot / objeto y es obligatorio en el fichero (nodo padre).

**<link>:** Describe un cuerpo rígido con inercia, propiedades visuales y modelos de colisión (<inertial>, <visual> y <collision>).

**<gazebo>:** Añade extensiones de simulación para Gazebo.

### **3.3 JOINTS**

**<joint>:** Describe la cinemática y dinámica de la articulación e incluso puede definir los límites de los movimientos, está formado por 2 links (parent y child) y puede contener restricciones de movimiento.

**Tipos:** **revolute** (rota sobre uno de sus ejes y se pueden especificar sus límites superiores e inferiores), **continuous** (similar al revolute pero sin límites), **prismatic** (se desliza a lo largo del eje), **fixed** (bloquea el movimiento en todos los grados), **floating** (permite movimiento en 6-DOF) y **planar** (permite movimiento en un plano perpendicular al eje).

### **3.4 SDF**

**SDF:** Formato desarrollado específicamente para descripciones detalladas de simulaciones y robots en Gazebo, solventa algunas de las limitaciones de URDF (sensores, actuadores, luces, cámaras e interacciones con entornos físicos complejos), permite modelar la dinámica del robot, entornos y elementos estáticos / dinámicos, su estructura está basada en XML y es mucho más flexible y extensible que URDF.

## TEMA 4: CONTROLANDO UN ROBOT EN PYBULLET

### 4.1 VELOCIDAD

**Control / Movimiento del robot:** Realizado a través de los joints y fuerzas externas, donde cada joint de tipo 'revolute' y 'prismatic' está implementado con motor en PyBullet, por tanto, se pueden comandar velocidades y fuerzas a ese motor.

**Velocidades / Fuerzas del robot:** La velocidad asignada a un joint no implica que el robot completo adquiera esa velocidad, ya que siempre estarán limitadas a las restricciones especificadas en los URDF, donde la mayoría de las acciones comandadas a un joint se realizan a través de las funciones `setJointMotorControl2` y `setJointMotorControlArray`.

### 4.2 MOMENTO DE FUERZA

**Momento de fuerza respecto a un punto O (M):** Pseudofuerza resultante al multiplicar el vector posición (d) por el vector fuerza (F). Además, da a conocer en qué medida existe la capacidad en una fuerza / sistema de fuerzas para cambiar el estado de la rotación del cuerpo alrededor de un eje que pase por dicho punto.

$$M = r * F$$

**Signo:** Sentido horario (M negativo) y sentido antihorario (M positivo).

### 4.3 FRICCIÓN

**Fricción:** Fuerza que existe entre dos superficies en contacto, que se opone al deslizamiento.

**Fuerza de rozamiento:** Posee una dirección paralela a la superficie de apoyo, cuyo coeficiente depende exclusivamente de la naturaleza de los cuerpos en contacto.

**Fuerza Normal:** Fuerza perpendicular a la superficie de contacto del objeto y una reacción a la fuerza que el objeto ejerce sobre la superficie de acuerdo con la tercera ley de Newton.

$$F_g = m \times g \Rightarrow F_{paralela} = F_g \times \sin(\theta) ; F_{perpendicular} = F_g \times \cos(\theta)$$

$$F_n = F_{perpendicular} ; F_f = \mu \times F_n ; F_{resultante} = F_{paralela} - F_f$$

**Fricción lineal:** Fricción en la superficie de contacto y de desplazamiento, donde la fuerza normal es la reacción a la fuerza aplicada sobre un objeto perpendicularmente a la superficie de soporte. En simulaciones se modela cuando un elemento se mueve o arrastra a través de una superficie plana o inclinada.

$$F_{resultante} = F_{paralela} - F_f$$

**Fricción giratoria / alrededor de la normal:** Resistencia contra el movimiento rotacional alrededor del punto de contacto entre dos superficies, se manifiesta cuando el objeto gira sobre sí mismo sin necesariamente trasladarse, es muy utilizada en simulación de ruedas, poleas o incluso robots con partes rotativas que interactúan con superficies de contacto, y debe tener en cuenta el torque, la superficie de contacto y las velocidades angulares / aceleraciones.

**Fricción de rodadura:** Se produce cuando un objeto rueda sobre la superficie de otra rueda / suelo oponiéndose al movimiento de rodadura de un objeto, y es fundamental para poder simular el movimiento de vehículos / ruedas / esferas que se mueven rodando en vez de deslizándose.

$$f = u_r * N$$

### 4.4 RESTITUCIÓN

**Restitución:** Define de qué manera los objetos se recuperan después de una colisión, donde la dinámica de colisión hace uso del coeficiente [0..1], en el que 0 significa que los objetos se quedan pegados después de la colisión y la velocidad final es 0, y 1 significa que los objetos rebotan sin perder energía cinética y la velocidad final es igual a la inicial.



## 4.5 INERCIA

**Inercia:** Propiedad que tienen los cuerpos de permanecer en su estado de reposo o movimiento relativos. Es la resistencia que opone el objeto a modificar su estado de movimiento, donde un objeto tiene más inercia cuando es más difícil / costoso cambiar el estado físico del mismo.

**Momento de Inercia (I):** Medida que representa la resistencia de un objeto a cambios en su velocidad angular, depende de la geometría del cuerpo rígido y la posición del eje de giro y se puede calcular como el sumatorio de los productos de la masa de partículas por el cuadrado de la distancia  $r$  de cada partícula al eje.

$$I = \sum_i m_i r_i^2$$

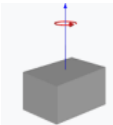
**Momento angular / cinético (L):** Magnitud física que representa la cantidad de movimiento de rotación de un objeto.

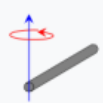
$$L = r \times p = r \times mv$$

**Principio de conservación del momento angular:** Afirma que si el momento de las fuerzas exteriores es 0, el momento angular total se conserva (permanece constante) para un cuerpo rígido que gira alrededor de su eje.

$$L = I * \omega$$

**Tensor / Matriz de inercia:** Tensor simétrico que describe la inercia rotacional de un cuerpo sólido rígido con todos los ejes de rotación posibles.

$$E_{rot} = \frac{1}{2} \begin{pmatrix} \Omega_x & \Omega_y & \Omega_z \end{pmatrix} \begin{pmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{pmatrix} \begin{pmatrix} \Omega_x \\ \Omega_y \\ \Omega_z \end{pmatrix}$$


$$I = \begin{bmatrix} \frac{1}{12}m(h^2 + d^2) & 0 & 0 \\ 0 & \frac{1}{12}m(w^2 + d^2) & 0 \\ 0 & 0 & \frac{1}{12}m(w^2 + h^2) \end{bmatrix}$$


$$I = \begin{bmatrix} \frac{1}{3}ml^2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{1}{3}ml^2 \end{bmatrix}$$

## 4.6 MOMENTO ANGULAR VS FUERZA DE TORQUE

**Momento angular:** Medida de la cantidad de rotación de un objeto que depende tanto de la masa / velocidad del objeto como de su distancia al eje de rotación, se relaciona con el estado de rotación de un objeto (cuánto y cómo está girando), un cambio en el momento angular de un objeto es el efecto de aplicar un torque sobre él en un sistema cerrado sin influencias externas, el momento angular se conserva y se mide en  $\text{kg} * \text{m}^2 / \text{s}$ .

**Fuerza de torque:** Describe cuánto puede cambiar el estado de movimiento rotacional de un objeto debido a una fuerza aplicada (causa de ese cambio en el estado de rotación y puede considerarse como la causa que puede alterar el momento angular de un objeto), el torque describe cómo las fuerzas externas / internas están tratando de cambiar ese estado de movimiento rotacional y se mide en  $\text{kg} * \text{m}^2 / \text{s}$ .

## TEMA 5: MOTOR GRÁFICO

### 5.1 REPRESENTACIÓN GRÁFICA Y ESCENAS 3D

**Simulador:** Motor de físicas, motor gráfico, sensores, entorno, controladores y API.

**Motor gráfico:** Renderizado / Gestión de modelos 3D, iluminación / sombras, materiales / texturas, animación / cinemática, gestión de cámaras, postprocesado / efectos visuales, simulación atmosférica / clima, optimización / rendimiento, integración con motores de físicas y renderizado de sensores.

**Representación gráfica 3D:** Proceso de visualización de objetos y escenas en 3D utilizando ordenadores.

**Ejemplos:** **Imágenes 2D** (ancho y altura) y **3D** (incorporan profundidad que permite una percepción más realista y con perspectiva del escenario) vitales para la simulación en Robótica.

**Escena 3D:** **Objetos 3D** (desde simples figuras geométricas hasta complejos modelos de robots), **cámaras** (punto de perspectiva), **luces** (iluminan la escena, crean brillos y dan realismo) y **sombras** (definen la interacción entre las luces y los objetos).

### 5.2 REPRESENTACIÓN DE OBJETOS

**Mallas poligonales:** Compuestas por vértices, aristas y caras, son la forma más común de representar objetos 3D y permiten detallar complejidad y forma con precisión.

**Superficies de subdivisión:** Basadas en la subdivisión de polígonos para suavizar las superficies, y son muy útiles cuando se requiere un alto nivel de detalle.

**Aplicaciones en Robótica:** Diseño y prototipado de robots, simulación, visualización de datos e impresión 3D.

**Herramientas de modelado 3D:** Blender, Autodesk Maya, SolidWorks y Unreal Engine.

### 5.3 RENDERIZADO (DEFINICIÓN Y TIPOS)

**Renderizado:** Proceso computacional por el cual se genera una imagen 2D foto-realista normalmente a partir de un modelo 3D (iluminación, texturas, sombreado y cámara).

**Renderizado en Tiempo Real:** Se utiliza en videojuegos y simulaciones para mantener una alta tasa de fotogramas por segundo (FPS) para interacciones dinámicas, estableciendo un compromiso entre la calidad de los detalles y velocidad de ejecución.

**Renderizado pre-calculado:** Se usa en la industria cinematográfica y representaciones arquitectónicas, obteniendo mucho mayor realismo y calidad de detalles.

**Render Farms:** Tareas altamente paralelizables, distribución de carga, escalabilidad, optimización de recursos, integración de pipelines y alta disponibilidad.

### 5.4 MODELO PIN-HOLE DE CÁMARA

**Modelo Pin-Hole de cámara:** Simula el comportamiento de la luz y cómo ésta se proyecta para formar imágenes sin utilizar lentes mediante la propagación rectilínea de la luz, permitiendo sacar información espacial desde la luz y renderizar la escena.

**Parámetros:** **Extrínsecos** (posición y orientación) e **intrínsecos** (distancia focal  $f_x$  y  $f_y$ , centro óptico  $U_0$  y  $V_0$  y skew / distorsión).

$$Pix_{img} = K * R * T * P_w = H * P_w$$

$$\begin{bmatrix} u_p \\ v_p \\ 1 \end{bmatrix}_{img} = \begin{bmatrix} f_x & 0 & U_0 & 0 \\ 0 & f_y & V_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_p \\ Y_p \\ Z_p \\ 1 \end{bmatrix}_w = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \end{bmatrix} \begin{bmatrix} X_p \\ Y_p \\ Z_p \\ 1 \end{bmatrix}_w$$

### 5.5 OPTIMIZACIONES

**Optimización de Frustum:** Lo que cae fuera del campo visual no se renderiza.

**Optimización de Culling:** No se renderiza ni la parte oculta ni la parte ocluida de los objetos.

**OpenGL:** API estándar industrial para gráficos de alto rendimiento 2D y 3D, es multiplataforma y aprovecha la aceleración hardware mediante pipelines (procesamiento de vértices mediante geometría 3D y procesamiento de píxeles mediante colores y texturas).

**Otras APIs:** **DirectX / Direct3D** (Microsoft), **Vulkan** (proporciona más control sobre GPU), **Metal** (Apple) y

**WebGL** (derivado de OpenGL).

## **} 5.6 RASTERIZADO, RAY-TRACING Y PATH-TRACING**

**Rasterizado:** Procedimiento mediante el cual una imagen definida en un formato de gráficos vectoriales se transforma en una matriz de píxeles o puntos, cuya visualización se realiza en un dispositivo de salida digital (pantalla, impresora y bitmaps).

**Ray-Tracing:** Técnica avanzada de renderizado en gráficos por ordenador capaz de simular el camino que la luz sigue en el mundo real para generar imágenes muy realistas, además de reflexiones, refracciones, sombras suaves, dispersión de luz e iluminación global, cuyo soporte en tiempo real lo poseen muchas GPUs.

**Path-Tracing:** Extensión del Ray-Tracing que utiliza el principio de simulación de Monte Carlo para la iluminación global, estudia la compleja interacción de la luz con las superficies y cómo se dispersa e ilumina en otras partes de la escena, y es ideal para el realismo, donde la calidad de la imagen es prioritaria.

**Frame Generation (NVIDIA DLSS 3 / 4):** Permite generar cuadros adicionales para mejorar la fluidez y FPS sin aumentar la carga de la GPU (tecnología presente en la serie RTX 4000 / 5000).

## TEMA 6: CINEMÁTICA INVERSA

### 6.1 CINEMÁTICA DIRECTA E INVERSA

**Cinemática:** Rama de la mecánica que estudia el movimiento de los cuerpos sin considerar las fuerzas que lo causan (posición, velocidad y aceleración de las partes del robot), permitiendo la planificación de trayectorias y la realización de tareas específicas.

**Cinemática Directa:** Calcula la posición y orientación del elemento final conociendo ángulos y geometría.

**Cinemática Inversa:** Calcula los ángulos y geometría para alcanzar la orientación y posición del elemento final conociendo la posición y orientación del elemento final. Es la técnica que permite determinar el movimiento de una cadena de articulaciones para lograr que un actuador final se ubique en la posición deseada (se considera un problema complejo, ya que la resolución de ecuaciones no deriva en una única solución).

**Aplicaciones:** Control / Simulación de robots, planificación de movimientos e interacción con el entorno.

**Características:** Se espera una solución en tiempo real (menos articulaciones en movimiento y más óptima en términos de batería), no siempre es posible ofrecer una solución (limitaciones físicas de articulaciones), y es utilizado para articulaciones en brazos robóticos, bípedos y cuadrúpedos.

### 6.2 SINGULARIDAD

**Singularidad:** Define un mal funcionamiento de un sistema robótico que ocurre porque los robots son controlados por matemáticas, ya que muchas ecuaciones matemáticas tienden a infinito. Ocurre cuando hay múltiples soluciones en el sistema cinemático y no se escoge la óptima, asumiendo que hay una, donde los efectos más comunes de singularidad son una posición incorrecta y aumento drástico de la velocidad de las articulaciones.

**Matriz Jacobiana:** Utilizada para representar la relación entre las variables de entrada y las variables de salida en un sistema multivariable (normalmente en sistemas no lineales), y ofrece una descripción del cambio de las variables de salida con respecto a pequeños cambios en las variables de entrada.

**Usos:** Relación entre velocidades, aproximaciones lineales, cambio de coordenadas, estimación de velocidad / propiedades de fluidos en el espacio, resolver problemas de cinemática inversa y entender deformaciones de objetos cuando se mueven sus articulaciones.

**Algoritmos iterativos:** **Jacobiana Inversa** (converge rápidamente pero es costosa computacionalmente),

**Pseudoinversa** (inestable cerca de la singularidad), **Jacobiana Traspuesta** (velocidad muy alta cerca de la singularidad) y **Mínimos Cuadrados Amortiguados** (reduce la velocidad cerca de la singularidad).

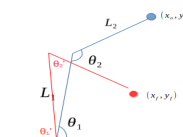
### 6.3 JACOBIANO INVERSO

**Jacobiano Inverso:**  $[\Delta\theta_1 \ \Delta\theta_2] = J^{-1}[\Delta x \ \Delta y]$ , donde  $\Delta x = x_f - x_o$   $\Delta y = y_f - y_o$

**Ecuaciones de posición:**  $x_o = L_1 \cos(\theta_1) + L_2 \cos(\theta_1 + \theta_2)$   $y_o = L_1 \sin(\theta_1) + L_2 \sin(\theta_1 + \theta_2)$

**Obtención de los nuevos ángulos:**  $\theta_1' = \theta_1 + \Delta\theta$   $\theta_2' = \theta_2 + \Delta\theta$

$$J = \begin{bmatrix} \frac{\partial x}{\partial \theta_1} & \frac{\partial x}{\partial \theta_2} \\ \frac{\partial y}{\partial \theta_1} & \frac{\partial y}{\partial \theta_2} \end{bmatrix} = \begin{bmatrix} -L_1 \sin(\theta_1) - L_2 \sin(\theta_1 + \theta_2) & -L_2 \sin(\theta_1 + \theta_2) \\ L_1 \cos(\theta_1) + L_2 \cos(\theta_1 + \theta_2) & L_2 \cos(\theta_1 + \theta_2) \end{bmatrix}$$



### 6.4 MÍNIMOS CUADRADOS AMORTIGUADOS

**Mínimos Cuadrados Amortiguados:** Parte del enfoque clásico de mínimos cuadrados para mejorar la estabilidad y la robustez de la solución en condiciones problemáticas como las singularidades, y añade el factor de amortiguamiento  $\lambda$  que reduce efectos de las singularidades.

$$[\Delta\theta_1 \ \Delta\theta_2] = J^T (J J^T + \lambda^2 I)^{-1} [\Delta x \ \Delta y]$$

**Ventajas:** Robustez ante las singularidades, mejora de la estabilidad y flexibilidad.

### 6.5 END EFFECTOR

**End Effector:** Define la herramienta conectada al final de un brazo mecánico, además de ser el "punto de contacto" entre el robot y los objetos o el entorno de trabajo (robot Da Vinci).

## **7 TEMA 7: END EFFECTORS**

### **7.1 END EFFECTOR**

**End Effector:** Define la herramienta conectada al final de un brazo mecánico, además de ser el “punto de contacto” entre el robot y los objetos o el entorno de trabajo (robot Da Vinci). En simulación, normalmente no se diseña la parte mecánica.

**Características:** Agiliza desarrollo, enfoque en dinámicas y cinemáticas y optimización de recursos computacionales.

### **7.2 PRISMATIC**

**Worm gear:** Transmisión del movimiento en ángulo recto, reduce velocidad y aumenta torque.

### **7.3 GEOMETRÍA ACKERMANN**

**Características:** Las ruedas de un mismo eje giran ángulos distintos y permite que las ruedas tracen trayectorias circulares con distintos radios durante una curva (paralelogramo con dimensiones y características especiales).

### **7.4 EJE MOTOR**

**Características:** Hace mover un eje al que están conectados a 2 ruedas (en simulación es un joint infinite o revolute).

## **TEMAS 8: BLENDER**

### **8.1 BLENDER**

**Blender:** Software multiplataforma dedicado al modelado, iluminación, renderizado de gráficos tridimensionales, simulación de cuerpos rígidos / blandos y fluidos.

### **8.2 PIPELINE RENDER**

**Cycles:** Motor de renderizado basado en RT y PT con multi-core GPU y soporte GPU (CUDA).

**Eevee:** Basado en técnicas de rasterización y es útil para la visualización interactiva y el trabajo de previsualización.

### **8.3 FORMATOS**

**URDF:** Representación XML del modelo del robot.

**SDF o SMURF:** Representación XML de un modelo de robot, su entorno y acciones de control.

**Stereolithography (.stl):** Formato (CAD) que define geometría de objetos 3D, excluyendo información como color, texturas o propiedades físicas.

**Wavefront (.obj):** Formato adoptado en gráfico 3D que define exclusivamente la geometría 3D.

**Collada (.dae):** Estándar basado en XML que describe geometría, sombras, efectos, animaciones y cinemática.

### **8.4 INSTALACIÓN DEL ENTORNO**

**Anaconda (miniconda):** Framework para creación de entornos aislados del sistema base que poseen diferente configuración y versiones de python.

**Blender 3.3 LTS:** Software multiplataforma, dedicado especialmente al modelado, renderizado, la animación y creación de gráficos tridimensionales.

**Phobos:** Plugin para blender que permite modelar robots y exportarlos para utilizarlos en entornos como ROS, GAZEBO, etc.

## **TEMAS 9: SIM2REAL**

### **9.1 SIM2REAL**

**Sim2Real:** Proceso de transferir conocimientos o algoritmos desde simuladores y mundos virtuales para su implementación en el mundo real, reduciendo costos, tiempo, optimización, entrenamiento y la minimización de la brecha entre simulación y realidad. Ese gap (brecha) hace que la transferencia del conocimiento no sea completa y el robot no funcione correctamente (sim2sim y real2real).

**Características:** **Complejidad del entorno** (variabilidad, dinámica y diversidad de escenarios), **fidelidad de la simulación** (exactitud de los modelos físicos y representación sensorial), generalización del modelo y limitaciones del procesamiento en RT.

### **9.2 DATA AUGMENTATION**

**Data Augmentation:** Aumenta los datos y la simulación mediante la generación de datos sintéticos variados mediante simulaciones para entrenar modelos de machine learning y deep learning. Esto incluye alterar dinámicamente condiciones de iluminación, texturas y configuraciones de objetos para cubrir un amplio espectro de situaciones posibles.

### **9.3 DOMAIN RANDOMIZATION**

**Domain Randomization:** Introduce variabilidad aleatoria en las propiedades de los entornos simulados durante el entrenamiento (aparición visual / física y configuración del entorno), lo que hace que el modelo sea menos sensible a las diferencias entre la simulación y el mundo real.

**Simulación + Realidad:** Uso de datasets simulados y reales al mismo tiempo.

### **9.4 TRANSFER LEARNING**

**Transfer Learning:** Aprovecha el conocimiento adquirido en una tarea o dominio para aplicarlo en otra tarea o dominio diferente, y evita tener que entrenar modelos desde cero.

### **9.5 FINE-TUNING**

**Fine-Tuning:** Técnica específica dentro de transfer learning que consiste en tomar un modelo preentrenado (simulación) y ajustar sus parámetros con un conjunto más pequeño y representativo de datos reales. Se basa en comenzar el aprendizaje desde un punto de partida y no desde cero de nuevo (no hay cambio de dominio ni de tarea).