ECE276A: Sensing & Estimation in Robotics
Lecture 4: Robot Motion and Observation Models

Nikolay Atanasov

natanasov@ucsd.edu

UC San Diego
JACOBS SCHOOL OF ENGINEERING
Electrical and Computer Engineering

Outline

Rigid Body Kinematics and Dynamics

Motion Models

Observation Models

Rotation Kinematics

- The trajectory $R(t)$ of continuous rotation motion satisfies:

$$R^\top(t)R(t) = I \quad \Rightarrow \quad \dot{R}^\top(t)R(t) + R^\top(t)\dot{R}(t) = 0.$$

- Since $R^\top(t)\dot{R}(t)$ is **skew-symmetric**, there exists $\boldsymbol{\omega}(t) \in \mathbb{R}^3$ such that:

$$R^\top(t)\dot{R}(t) = \hat{\boldsymbol{\omega}}(t)$$

- **Rotation kinematics**: the orientation of a rigid body $R(t) \in SO(3)$ rotating with angular velocity $\boldsymbol{\omega}(t) \in \mathbb{R}^3$ (in body-frame coordinates) satisfies:

$$\dot{R}(t) = R(t)\hat{\boldsymbol{\omega}}(t)$$

- **Discrete-time rotation kinematics**: if $\boldsymbol{\omega}(t) \equiv \boldsymbol{\omega}_k$ is constant for $t \in [t_k, t_{k+1})$ and $R_k := R(t_k)$, $\tau_k := t_{k+1} - t_k$:

$$R_{k+1} = R_k \exp(\tau_k \hat{\boldsymbol{\omega}}_k)$$

Quaternion Kinematics

- **Quaternion kinematics**: the orientation of a rigid body $\mathbf{q}(t) \in \mathbb{H}_*$ rotating with angular velocity $\boldsymbol{\omega}(t) \in \mathbb{R}^3$ (in body-frame coordinates) satisfies:

$$\dot{\mathbf{q}}(t) = \mathbf{q}(t) \circ [0, \boldsymbol{\omega}(t)/2]$$

- **Discrete-time quaternion kinematics**: if $\boldsymbol{\omega}(t) \equiv \boldsymbol{\omega}_k$ is constant for $t \in [t_k, t_{k+1})$ and $\mathbf{q}_k := \mathbf{q}(t_k)$, $\tau_k := t_{k+1} - t_k$:

$$\mathbf{q}_{k+1} = \mathbf{q}_k \circ \exp([0, \tau_k \boldsymbol{\omega}_k/2])$$

Pose Kinematics

- **Pose kinematics**: the pose of a rigid body $T(t) \in SE(3)$ moving with **twist** (generalized velocity) $\zeta(t) = \begin{bmatrix} \mathbf{v}(t) \\ \boldsymbol{\omega}(t) \end{bmatrix} \in \mathbb{R}^6$ (in body-frame coordinates) satisfies:

$$\dot{T}(t) = T(t)\hat{\zeta}(t) \qquad\qquad \begin{bmatrix} \mathbf{v} \\ \boldsymbol{\omega} \end{bmatrix}^{\wedge} := \begin{bmatrix} \hat{\boldsymbol{\omega}} & \mathbf{v} \\ \mathbf{0} & 0 \end{bmatrix}$$

- **Discrete-time pose kinematics**: if $\zeta(t) \equiv \zeta_k$ is constant for $t \in [t_k, t_{k+1})$ and $T_k := T(t_k)$, $\tau_k := t_{k+1} - t_k$:

$$T_{k+1} = T_k \exp(\tau_k \hat{\zeta}_k)$$

Pose Dynamics

- **Pose dynamics**: the pose $T(t) \in SE(3)$ and twist $\zeta(t) \in \mathbb{R}^6$ of a rigid body with mass $m \in \mathbb{R}_{>0}$ and moment of inertia $J \in \mathbb{R}^{3\times 3}$, moving with **wrench** (generalized force) $\mathbf{w}(t) = \begin{bmatrix} \mathbf{f}(t) \\ \boldsymbol{\tau}(t) \end{bmatrix} \in \mathbb{R}^6$ (in body-frame coordinates) satisfies:

$$\dot{T}(t) = T(t)\hat{\zeta}(t) \qquad\qquad M := \begin{bmatrix} mI & 0 \\ 0 & J \end{bmatrix}$$

$$M\dot{\zeta}(t) = \overset{\curlywedge}{\zeta}(t)^\top M\zeta(t) + \mathbf{w}(t) \qquad\qquad \overset{\curlywedge}{\begin{bmatrix} \mathbf{v} \\ \omega \end{bmatrix}} := \begin{bmatrix} \hat{\omega} & \hat{\mathbf{v}} \\ \mathbf{0} & \hat{\omega} \end{bmatrix}$$

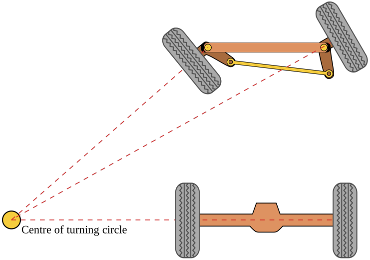
Outline

Rigid Body Kinematics and Dynamics

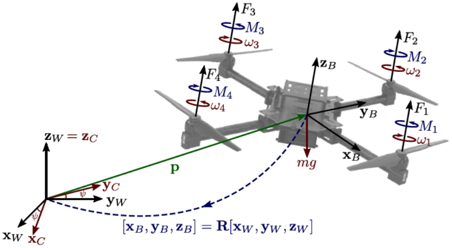
Motion Models

Observation Models

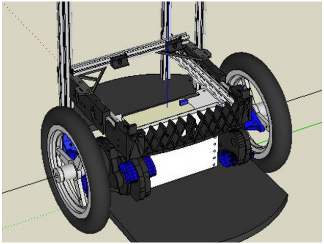
Motion Models



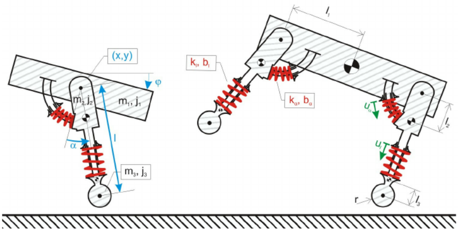
Ackermann Drive



Quadrotor



Differential Drive



Spring-loaded Gait

Motion Model

- Variables describing a robot system:
  - time t (continuous or discrete)
  - state $\mathbf{x}$ (e.g., position, orientation)
  - control input $\mathbf{u}$ (e.g., velocity, force)
  - disturbance $\mathbf{w}$ (e.g., tire slip, wind)
- A **motion model** is a function f relating the current state $\mathbf{x}$ and input $\mathbf{u}$ of a robot with its state change
  - Continuous-time: $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t))$
  - Discrete-time: $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$
- If the robot motion is affected disturbance $\mathbf{w}$ modeled as a random variable, then the state $\mathbf{x}$ is also a random variable described either:
  - in function form: $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{w}_t)$ or
  - with the probability density function $p_f(\cdot \mid \mathbf{x}_t, \mathbf{u}_t)$ of $\mathbf{x}_{t+1}$

Odometry-based Motion Model

- Consider a rigid-body robot with state $\mathbf{x}_t = T_t \in SE(3)$ capturing the robot pose in the world frame $\{W\}$ at time t
- **Odometry**: onboard sensors (camera, lidar, encoders, imu, etc.) may be used to estimate the relative pose of the robot body frame at time $t+1$ with respect to the body frame at time t:

$$\mathbf{u}_t = {}_tT_{t+1} = \begin{bmatrix} {}_tR_{t+1} & {}_t\mathbf{p}_{t+1} \\ \mathbf{0}^\top & 1 \end{bmatrix} \in SE(3)$$

- **Odometry-based motion model**: given the robot state $\mathbf{x}_t$ and input $\mathbf{u}_t$ at time t, the state at time $t+1$ satisfies:

$$T_{t+1} = \mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) := \mathbf{x}_t\mathbf{u}_t = T_t\,{}_tT_{t+1}$$

- Given an initial pose $\mathbf{x}_0$ and odometry measurements $\mathbf{u}_0, \ldots, \mathbf{u}_t$, the robot pose at time $t+1$ can be estimated as:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) = f(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})\mathbf{u}_t = \ldots = \mathbf{x}_0\mathbf{u}_0\mathbf{u}_1\cdots\mathbf{u}_t$$

- An odometry estimate is "drifting" (gets worse over time) because small measurement errors in each $\mathbf{u}_t$ are accumulated

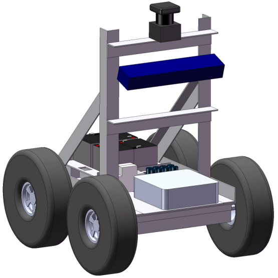
Differential-drive Kinematic Model

- **State**: $\mathbf{x} = (\mathbf{p}, \theta)$, where $\mathbf{p} = (x, y) \in \mathbb{R}^2$ is the position and $\theta \in (-\pi, \pi]$ is the orientation (yaw angle) in the world frame
- **Control**: $\mathbf{u} = (v, \omega)$, where $v \in \mathbb{R}$ is the linear velocity and $\omega \in \mathbb{R}$ is the angular velocity (yaw rate) in the body frame
- **Continuous-time model**:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = f(\mathbf{x}, \mathbf{u}) := \begin{bmatrix} v \cos\theta \\ v \sin\theta \\ \omega \end{bmatrix}$$

- Obtained from 2D pose kinematics with body twist $\zeta = (v, 0, \omega)^\top$:

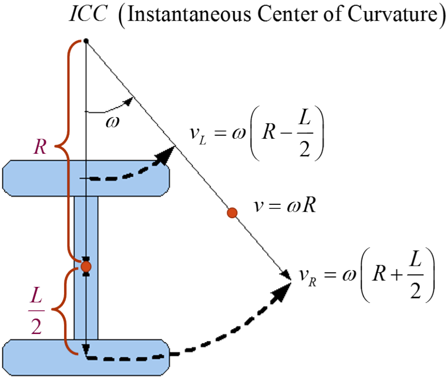
$$\begin{bmatrix} \dot{R}(\theta) & \dot{\mathbf{p}} \\ \mathbf{0} & 0 \end{bmatrix} = \begin{bmatrix} R(\theta) & \mathbf{p} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} 0 & -\omega & v \\ \omega & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$



Differential-drive Kinematic Model

- Let ℓ be the axle length (distance between wheels) and r be the radius of rotation, i.e., the distance from ICC to axle center
- The arc-length traveled is equal to the angle θ times the radius r

$$vt = r\theta \qquad \Rightarrow \qquad v = \frac{r\theta}{t} = r\omega$$



- Left wheel: $v_L = \omega(r - \ell/2)$
- Right wheel: $v_R = \omega(r + \ell/2)$
- Linear and angular velocity from wheel velocities:

$$\omega = \frac{v_R - v_L}{\ell}$$

$$r = \frac{\ell}{2}\left(\frac{v_L + v_R}{v_R - v_L}\right) = \frac{v}{\omega}$$

$$v = \frac{v_R + v_L}{2}$$

Discrete-time Differential-drive Kinematic Model

- **Euler discretization** over time interval of length τ_t:

$$\mathbf{x}_{t+1} = \begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} = f_d(\mathbf{x}_t, \mathbf{u}_t) := \mathbf{x}_t + \tau_t \begin{bmatrix} v_t \cos(\theta_t) \\ v_t \sin(\theta_t) \\ \omega_t \end{bmatrix}$$

- **Exact integration** over time interval of length τ_t:

$$\mathbf{x}_{t+1} = \begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} = f_d(\mathbf{x}_t, \mathbf{u}_t) := \mathbf{x}_t + \tau_t \begin{bmatrix} v_t \operatorname{sinc}\left(\frac{\omega_t \tau_t}{2}\right) \cos\left(\theta_t + \frac{\omega_t \tau_t}{2}\right) \\ v_t \operatorname{sinc}\left(\frac{\omega_t \tau_t}{2}\right) \sin\left(\theta_t + \frac{\omega_t \tau_t}{2}\right) \\ \omega_t \end{bmatrix}$$

- The exact integration is equivalent to the discrete-time pose kinematics:

$$\begin{bmatrix} R(\theta_{t+1}) & \mathbf{p}_{t+1} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} R(\theta_t) & \mathbf{p}_t \\ \mathbf{0} & 1 \end{bmatrix} \exp\left(\tau_t \begin{bmatrix} 0 & -\omega_t & v_t \\ \omega_t & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \right)$$

Differential-drive Kinematic Model

- What is the state after τ seconds if we apply constant linear velocity v and angular velocity ω at time t_0?
- To convert the continuous-time differential-drive model to discrete time, we solve the ordinary differential equations:

$$
\begin{aligned}
\dot{x}(t) &= v\cos\theta(t) \\
\dot{y}(t) &= v\sin\theta(t) \\
\dot{\theta}(t) &= \omega
\end{aligned}
\quad\Rightarrow\quad
\begin{aligned}
\theta(t_0+\tau) &= \theta(t_0) + \int_{t_0}^{t_0+\tau} \omega ds = \theta(t_0) + \omega\tau \\
x(t_0+\tau) &= x(t_0) + v\int_{t_0}^{t_0+\tau} \cos\theta(s) ds \\
&= x(t_0) + \frac{v}{\omega}\left(\sin\left(\omega\tau+\theta(t_0)\right) - \sin\theta(t_0)\right) \\
&= x(t_0) + v\tau\frac{\sin(\omega\tau/2)}{\omega\tau/2}\cos\left(\theta(t_0)+\frac{\omega\tau}{2}\right) \\
y(t_0+\tau) &= y(t_0) + v\int_{t_0}^{t_0+\tau} \sin\theta(s) ds \\
&= y(t_0) - \frac{v}{\omega}\left(\cos\theta(t_0) - \cos\left(\omega\tau+\theta(t_0)\right)\right) \\
&= y(t_0) + v\tau\frac{\sin(\omega\tau/2)}{\omega\tau/2}\sin\left(\theta(t_0)+\frac{\omega\tau}{2}\right)
\end{aligned}
$$

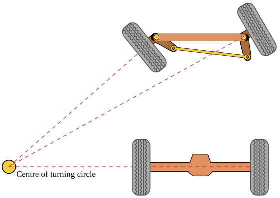
Ackermann-drive Kinematic Model

- **State**: $\mathbf{x} = (\mathbf{p}, \theta)$, where $\mathbf{p} = (x, y) \in \mathbb{R}^2$ is the position and $\theta \in (-\pi, \pi]$ is the orientation (yaw angle) in the world frame

- **Control**: $\mathbf{u} = (v, \phi)$, where $v \in \mathbb{R}$ is the linear velocity and $\phi \in (-\pi, \pi]$ is the steering angle in the body frame

- **Continuous-time model**:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = f(\mathbf{x}, \mathbf{u}) := \begin{bmatrix} v \cos\theta \\ v \sin\theta \\ \frac{v}{L} \tan\phi \end{bmatrix}$$



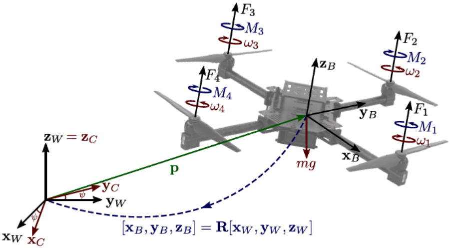
where L is the distance between the two wheel axles

- With the definition $\omega := \frac{v}{L} \tan\phi$, the model is equivalent to the differential-drive model and we can use the same discretized models

Quadrotor Dynamics Model

- **State**: $\mathbf{x} = (\mathbf{p}, R, \mathbf{v}, \boldsymbol{\omega})$ with position $\mathbf{p} \in \mathbb{R}^3$, orientation $R \in SO(3)$, body-frame linear velocity $\mathbf{v} \in \mathbb{R}^3$, body-frame angular velocity $\boldsymbol{\omega} \in \mathbb{R}^3$
- **Control**: $\mathbf{u} = (\rho, \boldsymbol{\tau})$ with body-frame thrust force $\rho \in \mathbb{R}$ and torque $\boldsymbol{\tau} \in \mathbb{R}^3$
- **Continuous-time dynamics model** with mass m, gravity acceleration g, moment of inertia $J \in \mathbb{R}^{3\times 3}$ and $\mathbf{e}_3 = (0, 0, 1)^\top$:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) = \begin{cases} \dot{\mathbf{p}} = R\mathbf{v} \\ \dot{R} = R\hat{\boldsymbol{\omega}} \\ m\dot{\mathbf{v}} = -\boldsymbol{\omega} \times m\mathbf{v} + \left(\rho\mathbf{e}_3 - mgR^\top\mathbf{e}_3\right) \\ J\dot{\boldsymbol{\omega}} = -\boldsymbol{\omega} \times J\boldsymbol{\omega} + \boldsymbol{\tau} \end{cases}$$



Outline

Rigid Body Kinematics and Dynamics

Motion Models

Observation Models

Observation Models

Inertial Measurement Unit

RGB Camera

Global Positioning System

2-D Lidar

Observation Model

- Variables describing a sensor:
  - sensor state $\mathbf{x}$ (e.g., position, orientation)
  - environment state $\mathbf{m}$ (e.g., object position, orientation, shape)
  - measurement $\mathbf{z}$ (e.g., image)
  - noise $\mathbf{v}$ (e.g., blur)
- An **observation model** is a function h relating the sensor state $\mathbf{x}$ and the environment state $\mathbf{m}$ with the sensor measurement $\mathbf{z}$:

$$\mathbf{z} = h(\mathbf{x}, \mathbf{m})$$

- If the sensor is affected by noise $\mathbf{v}$ modeled as a random variable, then the measurement $\mathbf{z}$ is also a random variable described either:
  - in function form: $\mathbf{z} = h(\mathbf{x}, \mathbf{m}, \mathbf{v})$ or
  - with the probability density function $p_h(\cdot \mid \mathbf{x}, \mathbf{m})$ of $\mathbf{z}$

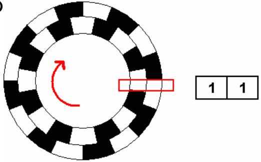
Common Sensor Models

- **Inertial or force sensor**: measures velocity, acceleration, or force, e.g., encoder, magnetometer, gyroscope, accelerometer
- **Position sensor**: measures position, e.g., GPS, RGBD camera, laser scanner
- **Bearing sensor**: measures angles, e.g., compass, RGB camera
- **Range sensor**: measures distance, e.g., radio received signal strength or time-of-flight

Encoder

- A magnetic encoder consists of a rotating gear, a permanent magnet, and a sensing element
- The sensor has two output channels with offset phase to determine the direction of rotation
- A microcontroller counts the number of transitions adding or subtracting 1 (depending on the direction of rotation) to the counter
- The distance traveled by the wheel, corresponding to one tick on the encoder is:

$$\text{meters per tick} = \frac{\pi \times (\text{wheel diameter})}{\text{ticks per revolution}}$$

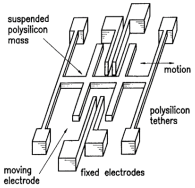


- The distance traveled during time τ for a given encoder count z, wheel diameter d, and 360 ticks per revolution is:

$$\tau v \approx \frac{\pi d z}{360}$$

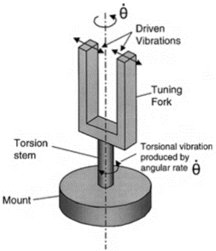
and can be used to predict position change in a differential-drive model

Inertial Measurement Unit



Surface Micromachined Accelerometer

- **IMU**: inertial measurement unit:
  - triaxial accelerometer (measures linear acceleration)
  - triaxial gyroscope (measures angular velocity)
- **Accelerometer**:
  - A mass m on a spring with constant k. The spring displacement is proportional to the system acceleration: $F = ma = kd \quad \Rightarrow \quad a = \frac{kd}{m}$
  - VLSI fabrication: the displacement of a metal plate with mass m is measured with respect to another plate using capacitance
  - Used for car airbags (if the acceleration goes above $2g$, the car is hitting something!)



- **Gyroscope**: uses Coriolis force to detect rotational velocity from the changing mechanical resonsance of a tuning fork

IMU Observation Model

- **State**: $(\mathbf{p}, R, \mathbf{v}, \boldsymbol{\omega}, \mathbf{a}, \boldsymbol{\alpha}, \mathbf{b}_g, \mathbf{b}_a)$ with position $\mathbf{p} \in \mathbb{R}^3$, orientation $R \in SO(3)$, body-frame linear velocity $\mathbf{v} \in \mathbb{R}^3$, body-frame angular velocity $\boldsymbol{\omega} \in \mathbb{R}^3$, body-frame linear acceleration $\mathbf{a} \in \mathbb{R}^3$, body-frame angular acceleration $\boldsymbol{\alpha} \in \mathbb{R}^3$, gyroscope bias $\mathbf{b}_g \in \mathbb{R}^3$, accelerometer bias $\mathbf{b}_a \in \mathbb{R}^3$
- **Extrinsic Parameters**: IMU position ${}_B\mathbf{p}_I \in \mathbb{R}^3$ and orientation ${}_BR_I \in SO(3)$ in the body frame are assumed known or obtained from calibration
- **Strapdown IMU**: the IMU frame and the body frame are identical, i.e, ${}_B\mathbf{p}_I = \mathbf{0}$ and ${}_BR_I = I$
- **Measurement**: $(\mathbf{z}_\omega, \mathbf{z}_a)$ with angular velocity measurement $\mathbf{z}_\omega \in \mathbb{R}^3$ and linear acceleration measurement $\mathbf{z}_a \in \mathbb{R}^3$:

$$
\begin{aligned}
\mathbf{z}_\omega &= {}_BR_I^\top \boldsymbol{\omega} + \mathbf{b}_g + \mathbf{n}_g \\
\mathbf{z}_\mathbf{a} &= {}_BR_I^\top \left(\mathbf{a} - gR^\top \mathbf{e}_3 + \hat{\boldsymbol{\alpha}}\, {}_B\mathbf{p}_I + \hat{\boldsymbol{\omega}}^2\, {}_B\mathbf{p}_I \right) + \mathbf{b}_a + \mathbf{n}_a
\end{aligned}
$$

Laser Sensors

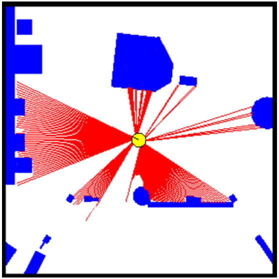
Single-beam Garmin Lidar

2-D Hokuyo Lidar

3-D Velodyne Lidar

LIDAR Model

- **LIDAR**: LIght Detection And Ranging
- Illuminates the scene with pulsed laser light and measures the return times and wavelengths of the reflected pulses
- Mirrors are used to steer the laser beam in the *xy* plane (and *zy* plane for 3D lidars)
- LIDAR rays are emitted over a set of known horizontal (azimuth) and vertical (elevation) angles $\{\alpha_k, \epsilon_k\}$ and return range estimates $\{r_k\}$ to obstacles in the environment $\mathbf{m}$
- Example: Hokuyo URG-04LX; detectable range: 0.02 to 4m; 240° field of view with 0.36° angular resolution (666 beams); 100 ms/scan



Laser Range-Azimuth-Elevation Model

- Consider a Lidar with position $\mathbf{p} \in \mathbb{R}^3$ and orientation $R \in SO(3)$ observing a point $\mathbf{m} \in \mathbb{R}^3$ in the world frame

- The point $\mathbf{m}$ has coordinates $\bar{\mathbf{m}} := R^\top(\mathbf{m} - \mathbf{p})$ in the lidar frame

- The lidar provides a spherical coordinate measurement of $\bar{\mathbf{m}}$:

$$\bar{\mathbf{m}} = R^\top(\mathbf{m} - \mathbf{p}) = \begin{bmatrix} r\cos\alpha\cos\epsilon \\ r\sin\alpha\cos\epsilon \\ r\sin\epsilon \end{bmatrix}$$

  where r is the range, α is the azimuth, and ϵ is the elevation

- **Inverse observation model**: expresses the lidar state $\mathbf{p}$, R and environment state $\mathbf{m}$, in terms of the measurement $\mathbf{z} = \begin{bmatrix} r & \alpha & \epsilon \end{bmatrix}^T$

- Inverting gives the **laser range-azimuth-elevation model**:

$$\mathbf{z} = \begin{bmatrix} r \\ \alpha \\ \epsilon \end{bmatrix} = \begin{bmatrix} \|\bar{\mathbf{m}}\|_2 \\ \arctan\left(\bar{\mathbf{m}}_y / \bar{\mathbf{m}}_x\right) \\ \arcsin\left(\bar{\mathbf{m}}_z / \|\bar{\mathbf{m}}\|_2\right) \end{bmatrix} \qquad \bar{\mathbf{m}} = R^\top(\mathbf{m} - \mathbf{p})$$

Common Observation Models

- **Position sensor**: state $\mathbf{x} = (\mathbf{p}, R)$, position $\mathbf{p} \in \mathbb{R}^3$, orientation $R \in SO(3)$, observed point $\mathbf{m} \in \mathbb{R}^3$, measurement $\mathbf{z} \in \mathbb{R}^3$:

$$\mathbf{z} = h(\mathbf{x}, \mathbf{m}) = R^\top (\mathbf{m} - \mathbf{p})$$

- **Range sensor**: state $\mathbf{x} = (\mathbf{p}, R)$, position $\mathbf{p} \in \mathbb{R}^3$, orientation $R \in SO(3)$, observed point $\mathbf{m} \in \mathbb{R}^3$, measurement $z \in \mathbb{R}$:

$$z = h(\mathbf{x}, \mathbf{m}) = \|R^\top (\mathbf{m} - \mathbf{p})\|_2 = \|\mathbf{m} - \mathbf{p}\|_2$$

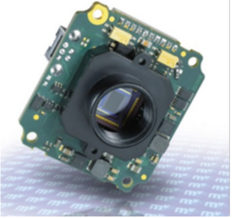
- **Bearing sensor**: state $\mathbf{x} = (\mathbf{p}, \theta)$, position $\mathbf{p} \in \mathbb{R}^2$, orientation $\theta \in (-\pi, \pi]$, observed point $\mathbf{m} \in \mathbb{R}^2$, bearing $z \in \mathbb{R}$:

$$z = h(\mathbf{x}, \mathbf{m}) = \arctan\left(\frac{m_2 - p_2}{m_1 - p_1}\right) - \theta$$

- **Camera sensor**: state $\mathbf{x} = (\mathbf{p}, R)$, position $\mathbf{p} \in \mathbb{R}^3$, orientation $R \in SO(3)$, intrinsic camera matrix $K \in \mathbb{R}^{3\times 3}$, projection matrix $P := [I,\ \mathbf{0}] \in \mathbb{R}^{2\times 3}$, observed point $\mathbf{m} \in \mathbb{R}^3$, pixel $\mathbf{z} \in \mathbb{R}^2$:

$$\mathbf{z} = h(\mathbf{x}, \mathbf{m}) = PK\pi(R^\top (\mathbf{m} - \mathbf{p})) \quad \text{projection:} \quad \pi(\mathbf{m}) := \frac{1}{\mathbf{e}_3^\top \mathbf{m}} \mathbf{m}$$

Camera Sensors



Global shutter

Stereo (+ IMU)

RGBD

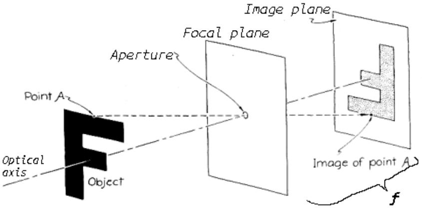
Event-based

Image Formation

- **Image formation model**: must trade-off physical accuracy and mathematical simplicity
- The values of an image depend on the shape and reflectance of the scene as well as the distribution of light
- **Image intensity** $I(u, v)$ describes the energy falling onto a small patch of the imaging sensor (integrated both over the shutter interval and over a region of space) and is measured in power per unit area (W/m^2)
- A camera uses a set of lenses to control the direction of light propagation by means of *diffraction*, *refraction*, and *reflection*
- **Thin lens model**: a simple geometric model of image formation that considers only <u>refraction</u>
- **Pinhole model**: a thin lens model in which the lens aperture is decreased to zero and all rays are forced to go through the optical center and remain undeflected (<u>diffraction</u> becomes dominant).

Pinhole Camera Model

- **Focal plane**: perpendicular to the **optical axis** with a circular aperture at the **optical center**



- **Image plane**: parallel to the focal plane and a distance f (**focal length**) in **meters** from the optical center

- The pinhole camera model is described in an **optical frame** centered at the optical center with the optical axis as the z-axis:
  - **optical frame**: x = right, y = down, z = forward
  - **regular frame**: x = forward, y = left, z = up

- **Image flip**: the object appears upside down on the image plane. To eliminate this effect, we can simply flip the image $(x, y) \rightarrow (-x, -y)$, which corresponds to placing the image plane $\{z = -f\}$ in front of the optical center instead of behind $\{z = f\}$.

Pinhole Camera Model

- **Field of view**: the angle subtended by the spatial extend of the image plane as seen from the optical center. If s is the side of the image plane in meters, then the field of view is $\theta = 2 \arctan\left(\frac{s}{2f}\right)$.
  - For a flat image plane: $\theta < 180^\circ$.
  - For a spherical or ellipsoidal imaging surface, common in omnidirectional cameras, θ can exceed 180°.

- **Ray tracing**: assuming a pinhole model and Lambertian surfaces, image formation can be reduced to tracing rays from points on objects to pixels. A mathematical model associating 3-D points in the world frame to 2-D points in the image frame must account for:
  1. **Extrinsics**: world-to-camera frame transformation
  2. **Projection**: 3D-to-2D coordinate projection
  3. **Intrinsics**: scaling and translation of the image coordinate frame

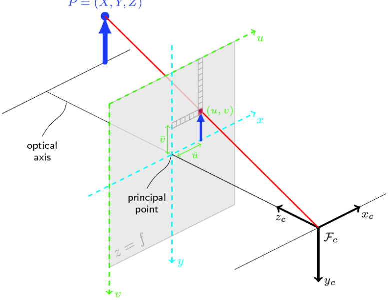
Extrinsics

- Let $\mathbf{p} \in \mathbb{R}^3$ and $R \in SO(3)$ be the camera position and orientation in the world frame
- **Rotation from regular to optical frame**: ${}_oR_r := \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix}$
- Let (X_w, Y_w, Z_w) be the coordinates of point $\mathbf{m}$ in the world frame. The coordinates of $\mathbf{m}$ in the optical frame are then:

$$\begin{pmatrix} X_o \\ Y_o \\ Z_o \\ 1 \end{pmatrix} = \begin{bmatrix} {}_oR_r & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} R & \mathbf{p} \\ \mathbf{0}^\top & 1 \end{bmatrix}^{-1} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} = \begin{bmatrix} {}_oR_r R^\top & -{}_oR_r R^\top \mathbf{p} \\ 0 & 1 \end{bmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$

Projection

- The 3D-to-2D perspective projection operation relates the optical-frame coordinates (X_o, Y_o, Z_o) of point $\mathbf{m}$ to its image coordinates (x, y) using similar triangles:



$$x = f\frac{X_o}{Z_o} \qquad y = f\frac{Y_o}{Z_o}$$

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \frac{1}{Z_o}\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{pmatrix} X_o \\ Y_o \\ Z_o \\ 1 \end{pmatrix}$$

- The above can be decomposed into:

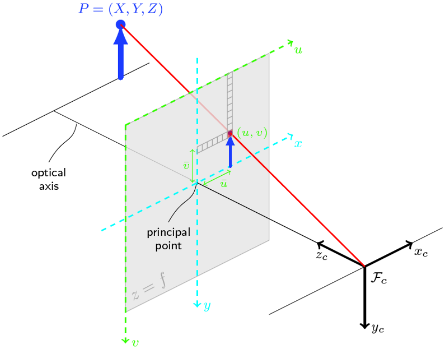
$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \underbrace{\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{image flip: } F_f}\underbrace{\begin{bmatrix} -f & 0 & 0 \\ 0 & -f & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{focal scaling: } K_f}\underbrace{\frac{1}{Z_o}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{canonical projection: } \pi}\begin{pmatrix} X_o \\ Y_o \\ Z_o \\ 1 \end{pmatrix}$$

Intrinsics

- Images are obtained in terms of pixels (u, v) with the origin of the pixel array typically in the upper-left corner of the image
- The relationship between the image frame and the pixel array is specified via the following parameters:
  - (s_u, s_v) [pixels/meter]: define the **scaling** from meters to pixels and the **aspect ration** $\sigma = s_u/s_v$
  - (c_u, c_v) [pixels]: coordinates of the *principal point* used to translate the image frame origin, e.g., $(c_u, c_v) = (320.5, 240.5)$ for a 640×480 image
  - s_θ [pixels/meter]: **skew factor** that scales non-rectangular pixels and is proportional to $\cot(\alpha)$ where α is the angle between the coordinate axes of the pixel array
- Normalized coordinates in the image frame are converted to pixel coordinates in the pixel array using the **intrinsic parameter matrix**:

$$\underbrace{\begin{bmatrix} s_u & s_\theta & c_u \\ 0 & s_v & c_v \\ 0 & 0 & 1 \end{bmatrix}}_{\text{pixel scaling: } K_s} \underbrace{\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{image flip: } F_f} \underbrace{\begin{bmatrix} -f & 0 & 0 \\ 0 & -f & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{focal scaling: } K_f} = \underbrace{\begin{bmatrix} fs_u & fs_\theta & c_u \\ 0 & fs_v & c_v \\ 0 & 0 & 1 \end{bmatrix}}_{\text{calibration matrix: } K} \in \mathbb{R}^{3\times 3}$$

Pinhole Camera Model Summary



- **Extrinsics**:

$$\begin{pmatrix} X_o \\ Y_o \\ Z_o \\ 1 \end{pmatrix} = \begin{bmatrix} {}_oR_r R^\top & -{}_oR_r R^\top \mathbf{p} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$

- **Projection and Intrinsics**:

$$\underbrace{\begin{pmatrix} u \\ v \\ 1 \end{pmatrix}}_{\text{pixels}} = \underbrace{\begin{bmatrix} fs_u & fs_\theta & c_u \\ 0 & fs_v & c_v \\ 0 & 0 & 1 \end{bmatrix}}_{\text{calibration: } K} \underbrace{\frac{1}{Z_o} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{canonical projection: } \pi} \begin{pmatrix} X_o \\ Y_o \\ Z_o \\ 1 \end{pmatrix}$$

Perspective Projection Camera Model

- The **canonical projection function** for vector $\mathbf{x} \in \mathbb{R}^3$ is:

$$\pi(\mathbf{x}) := \frac{1}{\mathbf{e}_3^\top \mathbf{x}} \mathbf{x}$$

- **Camera observation model**: state $\mathbf{x} = (\mathbf{p}, R)$ with position $\mathbf{p} \in \mathbb{R}^3$ and orientation $R \in SO(3)$ of the optical frame, intrinsic camera matrix $K \in \mathbb{R}^{3\times3}$, observed point $\mathbf{m} \in \mathbb{R}^3$, pixel $\mathbf{z} \in \mathbb{R}^2$:

$$\mathbf{z} = h(\mathbf{x}, \mathbf{m}) = PK\pi(R^\top(\mathbf{m} - \mathbf{p})) \qquad P := \begin{bmatrix} I & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{2\times3}$$

- The camera model can be written directly in terms of the camera optical frame pose $T \in SE(3)$ using homogeneous coordinates:

$$\underline{\mathbf{z}} = K\pi(PT^{-1}\underline{\mathbf{m}}) \qquad \underline{\mathbf{x}} := \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \qquad P := \begin{bmatrix} I & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{3\times4}$$

Radial Distortion and Other Camera Models

- **Wide field of view camera**: in addition to linear distortions described by the intrinsic parameters K, one can observe distortion along radial directions

- The simplest effective **model for radial distortion** is:

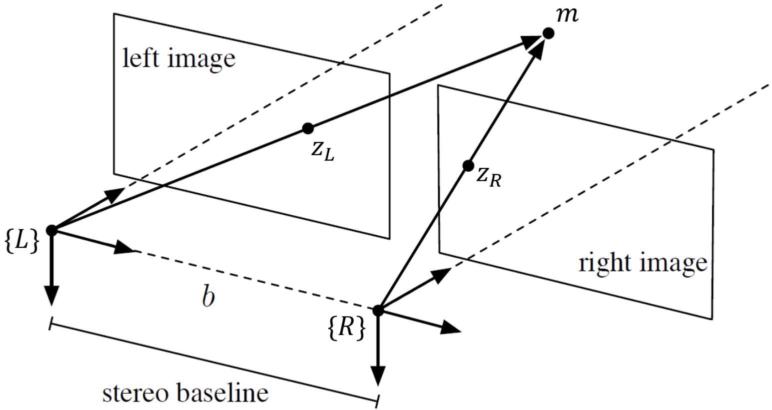
$$x = x_d(1 + a_1 r^2 + a_2 r^4)$$
$$y = y_d(1 + a_1 r^2 + a_2 r^4)$$

  where (x_d, y_d) are the pixel coordinates of distorted points and $r^2 = x_d^2 + y_d^2$ and a_1, a_2 are additional parameters modeling the amount of distortion

- **Spherical perspective projection**: if the imaging surface is a sphere $\mathbb{S}^2 := \{\mathbf{x} \in \mathbb{R}^3 \mid \|\mathbf{x}\| = 1\}$ (motivated by retina shapes in biological systems), we can define a spherical projection $\pi_s(\mathbf{x}) = \frac{\mathbf{x}}{\|\mathbf{x}\|_2}$ and use it in place of π in the perspective projection model

- **Catadioptric model**: uses an ellipsoidal imaging surface

Stereo Camera Model



Stereo Camera Model

- **Stereo Camera**: two perspective cameras rigidly connected to one another with a known transformation

- Unlike a single camera, a stereo camera can determine the depth of a point from a single stereo observation

- **Stereo Baseline**: the transformation between the two stereo cameras is only a displacement along the x-axis (optical frame) of size b

- The pixel coordinates $\mathbf{z}_L, \mathbf{z}_R \in \mathbb{R}^2$ of a point $\mathbf{m} \in \mathbb{R}^3$ in the world frame observed by a stereo camera at position $\mathbf{p} \in \mathbb{R}^3$ and orientation $R \in SO(3)$ with intrinsic parameters $K \in \mathbb{R}^{3\times3}$ are:

$$\underline{\mathbf{z}}_L = K\pi\left(R^\top(\mathbf{m}-\mathbf{p})\right) \qquad \underline{\mathbf{z}}_R = K\pi\left(R^\top(\mathbf{m}-\mathbf{p}) - b\mathbf{e}_1\right)$$

Stereo Camera Model

- Stacking the two observations together gives the stereo camera model:

$$\begin{bmatrix} u_L \\ v_L \\ u_R \\ v_R \end{bmatrix} = \underbrace{\begin{bmatrix} fs_u & 0 & c_u & 0 \\ 0 & fs_v & c_v & 0 \\ fs_u & 0 & c_u & -fs_u b \\ 0 & fs_v & c_v & 0 \end{bmatrix}}_{M} \frac{1}{z} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad \begin{bmatrix} x \\ y \\ z \end{bmatrix} = R^\top(\mathbf{m} - \mathbf{p})$$

- Because of the stereo setup, two rows of M are identical. The vertical coordinates of the two pixel observations are always the same because the epipolar lines in the stereo configuation are horizontal.

- The v_R equation may be dropped, while the u_R equation is replaced with a **disparity** measurement $d = u_L - u_R = \frac{1}{z} fs_u b$ leading to:

$$\begin{bmatrix} u_L \\ v_L \\ d \end{bmatrix} = \begin{bmatrix} fs_u & 0 & c_u & 0 \\ 0 & fs_v & c_v & 0 \\ 0 & 0 & 0 & fs_u b \end{bmatrix} \frac{1}{z} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad \begin{bmatrix} x \\ y \\ z \end{bmatrix} = R^\top(\mathbf{m} - \mathbf{p})$$