1. World / Ego (Local ENU) Coordinate Frame

본 프로젝트에서 차량 동역학과 제어는 Ego(Local) 좌표계를 기준으로 한다.

•원점: 차량 기준 (ego)

•단위: meter, radian

축 정의:

•**x-axis**: 차량 전방 (forward)

•**y-axis**: 차량 좌측 (left)

•**z-axis**: 위쪽 (up)

각도 정의:

•**yaw (ψ)**: +x축 기준, 반시계 방향(CCW)이 양(+)

- ψ = 0 → 차량이 +x 방향을 향함
- ψ > 0 → 차량이 좌회전 방향

2. Vehicle State & Control Variables

차량 상태 벡터:

$$\mathbf{x} = (x,\ y,\ \psi\ v)$$

•x, y: 차량 위치 (ENU local)

•ψ: 차량 heading (yaw)

•v: 차량 속도 (m/s)

제어 입력:

$$\mathbf{u} = (\delta,\ a)$$

•δ: 전륜 조향각 (steering angle)

- $\delta > 0$: 좌회전

•a: 종방향 가속도

- $a > 0$: 가속
- $a < 0$: 감속

3. Vehicle (Body) Coordinate Frame

Pure Pursuit 및 오차 계산에서는 **차량 프레임(vehicle frame)**을 사용한다.

•원점: 차량 중심

•축 정의:

- **x_v**: 차량 전방
- **y_v**: 차량 좌측

World → Vehicle 변환:

$$\begin{aligned} x_v &= \cos\psi\,(x_t - x) + \sin\psi\,(y_t - y) \\ y_v &= -\sin\psi\,(x_t - x) + \cos\psi\,(y_t - y) \end{aligned}$$

4. Reference Path Convention

•참조 경로는 **(x, y) 점들의 시퀀스**로 표현된다.

•경로의 방향은 **인덱스 증가 방향**으로 정의된다.

•경로의 접선 방향을 기준으로:

- 좌측에 있으면 lateral error > 0
- 우측에 있으면 lateral error < 0

5. Errors Used in Control

•**Lateral Error (횡방향 오차)**

경로 접선 방향의 법선 성분

•**Heading Error (방향 오차)**

$$e_\psi = \psi_{path} - \psi_{vehicle}$$

•각도 오차는 항상 $[-\pi\ \pi]$범위로 정규화한다.

6. Pure Pursuit Convention

•Lookahead distance:

$$L_d = L_0 + kv$$

•Target point:

- 현재 차량 위치에서 경로를 따라 L_d만큼 앞선 점

•Curvature:

$$\kappa = \frac{2y_v}{L_d^2}$$

•Steering angle:

7. BEV (Bird's Eye View) Visualization

•BEV는 **World/Ego 좌표계(x,y)**를 위에서 내려다본 표현이다.

•내부 표현은 pixel 기반이지만,

- 축 눈금은 **meter 의미**를 가진다.

•BEV에서:

- +x → 위쪽(전방)
- +y → 왼쪽

1. 자율주행 시뮬레이션에서 센서의 역할

본 수업에서는 다음과 같은 **이상적 카메라 센서 모델**을 구현한다.

•입력: 월드 좌표계의 3D 점 $\mathbf{P}_{world} = (X_w, Y_w, Z_w)$

•출력: 카메라 이미지 좌표계의 픽셀 좌표 (u, v)

이를 위해 다음 4단계를 순차적으로 수행한다.

$$\textbf{World} \rightarrow \textit{Camera Coordinate} \rightarrow \textit{Image Plane} \rightarrow \textit{BEV (Top-down)}$$

2. 좌표계 정의 (Coordinate Frames)

World 좌표계 (Local ENU)

•X_w: 차량 진행 방향 (forward)

•Y_w: 차량 좌측 (left)

•Z_w: 위쪽 (up)

Camera 좌표계 (Computer Vision Convention)

•Z_c: 전방 (forward)

•X_c: 우측 (right)

•Y_c: 아래쪽 (down)

카메라 좌표계는 컴퓨터비전(OpenCV)에서 일반적으로 사용하는 정의를 따른다.

3. 강체 변환 (Rigid Body Transformation)

월드 좌표의 점을 카메라 좌표로 변환하기 위해 **강체 변환**을 적용한다.

(1) Homogeneous Coordinate

$$\widetilde{\mathbf{P}}_{world} = \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

(2) World → Camera 변환 행렬

$$\widetilde{\mathbf{P}}_{cam} = \mathbf{T}_{cam \leftarrow world}\, \widetilde{\mathbf{P}}_{world}$$

$$\mathbf{T}_{cam \leftarrow world} = \begin{bmatrix} \mathbf{R}_{cam \leftarrow world} & \mathbf{t}_{cam} \\ \mathbf{0}^{\top} & 1 \end{bmatrix}$$

•$\mathbf{R} \in \mathbb{R}^{3\times3}$: 회전 행렬

•$\mathbf{t} \in \mathbb{R}^{3}$: 카메라 위치 (translation)

4. Pinhole Camera Model

카메라 좌표계에서의 점:

$$\mathbf{P}_{cam} = (X_c, Y_c, Z_c)$$

(1) 이상적 투영 (Normalized Image Plane)

$$x_n = \frac{X_c}{Z_c}, y_n = \frac{Y_c}{Z_c}$$

(2) Camera Intrinsic Matrix

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

•f_x, f_y: focal length (pixel 단위)

•(c_x, c_y): principal point (image center)

(3) Pixel Projection

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix} = \begin{bmatrix} f_x \dfrac{X_c}{Z_c} + c_x \\ f_y \dfrac{Y_c}{Z_c} + c_y \\ 1 \end{bmatrix}$$

$Z_c > 0$인 점만 카메라 전방에 존재하며,

이미지 범위를 벗어난 점은 센서에서 관측되지 않는다.

5. BEV (Bird's Eye View) 변환 개념

BEV는 **카메라 시점의 왜곡된 시각 정보**를

지면 기준의 정사영(top-down) 좌표계로 변환한 표현이다.

BEV 좌표 변환 (단순화된 모델)

월드 좌표의 지면 점 (X_w, Y_w)를

미리 정의된 범위 $[x_{min}, x_{max}], [y_{min}, y_{max}]$에 대해

픽셀 좌표로 선형 매핑한다.

$$u_{bev} = \frac{X_w - x_{min}}{x_{max} - x_{min}} \cdot W$$

$$v_{bev} = \left(1 - \frac{Y_w - y_{min}}{y_{max} - y_{min}}\right) \cdot H$$

•W, H: BEV 이미지 해상도

•BEV는 **제어·계획 모듈에서 기하적 판단**에 유리하다.

$$\boxed{\mathbf{P}_{world} \xrightarrow{\mathbf{T}_{cam \leftarrow world}} \mathbf{P}_{cam} \xrightarrow{Pinhole} (u, v)}$$

1. 차량 상태(State)와 제어 입력(Control Input)

(1) 차량 상태 벡터

본 실습에서 차량 상태는 다음 4개의 변수로 정의된다.

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ \psi \\ v \end{bmatrix}$$

•x, y: 차량의 위치 (ENU, ego 기준)

•ψ: 차량의 heading (yaw)

•v: 차량의 속도

이 상태는 **센서나 플래너가 아닌, 차량 자체의 물리적 상태**를 의미한다.

(2) 제어 입력

차량에 가해지는 제어 입력은 다음 두 가지이다.

$$\mathbf{u} = \begin{bmatrix} \delta \\ a \end{bmatrix}$$

•δ: 조향각 (steering angle)

•a: 종방향 가속도 (acceleration)

즉, 차량은 **조향과 가속이라는 두 입력만으로 움직인다.**

2. Kinematic Bicycle Model의 Assumptions

Kinematic Bicycle Model은 다음을 가정한다.

- **차량은 미끄러지지 않는다 (no slip)**
- **타이어의 힘, 질량, 관성은 무시한다**
- **차량은 앞·뒤 바퀴를 하나로 묶은 단순한 자전거 모델로 표현된다**

이 모델은 고속·한계 주행에는 부적합하지만 **도심 주행·경로 추종·제어 개념 학습**에는 적합하다.

4. 차량 상태 업데이트 수식 (Dynamics)

Kinematic Bicycle Model의 연속 시간 방정식은 다음과 같다.

$$\dot{x} = v\cos\psi$$

$$\dot{y} = v\sin\psi$$

$$\dot{\psi} = \frac{v}{L}\tan\delta$$

$$\dot{v} = a$$

•L: 차량 wheelbase

•δ: 조향각

•a: 가속도

본 실습에서는 이를 **Euler 적분**으로 시간 이산화한다.

$x_{k+1} = x_k + \dot{x}\Delta t$ *(동일하게 y, yaw, v 업데이트)*

이 과정이 바로 step_kinematic_bicycle() 함수의 수학적 의미이다.

5. reference path와 차량 상태의 기하적 관계

차량은 **절대 좌표를 목표로 주행하지 않는다.**

항상 reference path와의 상대적 관계를 기반으로 제어된다.

이를 위해 다음 두 가지 오차를 정의한다.

(1) Lateral Error (횡방향 오차)

•차량이 경로의 왼쪽에 있는지, 오른쪽에 있는지를 나타내는 오차

•경로의 접선 방향에 대한 **법선 방향 거리**

$$e_{lat} = (\mathbf{p}_{veh} - \mathbf{p}_{path}) \cdot \mathbf{n}_{path}$$

(2) Heading Error (방향 오차)

•차량의 heading과 경로의 방향 차이

$$e_{\psi} = \psi_{path} - \psi_{veh}$$

이 두 오차가 **경로 추종 문제를 수치적으로 표현한 지표**이다.

6. 조향 제어 (Steering Control)

본 실습에서는 **Pure Pursuit 제어**를 사용한다.

•차량 앞쪽에 일정 거리 L_d만큼 떨어진 **목표 점**을 설정

•차량이 해당 점을 향해 원호를 그리도록 조향각을 계산

7. 종방향 제어 (Speed Control)

속도 제어는 단순한 **P 제어**를 사용한다.

$$a = k(v_{ref} - v)$$

•v_{ref}: 목표 속도

•v: 현재 속도

이를 통해 차량 속도는:

•불연속적으로 변하지 않고

•시간에 따라 자연스럽게 증가·감소한다.

8. BEV 기반 통합 시각화의 의미

Dynamic & Control 모듈은 다음 조건을 만족해야 한다.

•참조 경로만 주어지면 **독립 실행 가능**

•차량의 주행 궤적이 **BEV 상에서 시각적으로 확인 가능**

•조향 입력 변화에 따라 궤적이 직관적으로 달라짐