

# MPC 주행 원리 및 논문-코드 비교

## 1) MPC 개요

MPC(Model Predictive Control)는 미래 상태를 예측해 최적의 제어 입력을 계산하는 방법입니다.

- 현재 조향/속도 상태를 기준으로 미래 궤적을 예측하고
- 제약조건을 만족하는 입력 후보 중
- 비용함수가 가장 작은 입력을 선택합니다.

핵심 절차:

1. 미래를 여러 칸(시간)으로 나눠 예측한다.
2. 점수(비용)가 가장 작은 조작을 고른다.
3. 그중 첫 번째 조작만 실제로 적용한다.
4. 다음 순간에 다시 1~3을 반복한다.

이 방식을 **Receding Horizon(이동 예측구간)**이라 합니다.

---

## 2) 차량 모델(기본식)

단순화된 자전거 모델의 이산시간 식:

- $(x, y)$ : 차량 위치
- $()$ : 헤딩각
- $(v)$ : 속도
- $()$ : 조향각
- $(a)$ : 가속도
- $(L)$ : 축간거리(앞바퀴~뒷바퀴 거리)
- $(t)$ : 제어 주기

공식:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \psi_{k+1} \\ v_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ \psi_k \\ v_k \end{bmatrix} + \begin{bmatrix} v_k \cos(\psi_k) \\ v_k \sin(\psi_k) \\ v_k \tan(\delta_k) \\ a_k \end{bmatrix} \Delta t$$

의미: - 상태  $((x_k, y_k, \psi_k, v_k))$ 와 입력  $((\delta_k, a_k))$ 로 - 다음 상태  $((x_{k+1}, y_{k+1}, \psi_{k+1}, v_{k+1}))$ 를 계산합니다.

---

## 3) 비용함수(목적식)

MPC 는 아래 비용 (J)를 최소화합니다.

대표식:

$$[ J = \sum_{k=0}^{N-1} ( w_y e_{y,k}^2 + w_e e_{e,k}^2 + w_v e_{v,k}^2 + w_{\delta} \delta_k^2 + w_a a_k^2 ) ]$$

- (N): 예측 칸 수(예: 20 칸)
- ( $e_{y,k}$ ): 경로에서 옆으로 벗어난 오차
- ( $e_{e,k}$ ): 차의 방향 오차
- ( $e_{v,k}$ ): 목표 속도 오차
- ( $\delta_k = \delta - \delta_{k-1}$ ): 핸들 변화량
- (w): 중요도 가중치(큰 값일수록 더 중요)

해석(각 항의 역할): - 경로에서 벗어나면 벌점 - 방향이 틀어져도 벌점 - 속도가 목표와 다르면 벌점 - 핸들을 갑자기 확 꺾으면 벌점 - 너무 큰 가속/감속도 벌점

즉, 경로추종 정확도와 조향/가감속의 부드러움 사이 균형을 최적화합니다.

---

## 4) 제약조건

최적화는 다음 제한을 동시에 만족해야 합니다.

$$[ \{ k \} ] [ a \{ a_k \} ] [ v \{ v_k \} ] [ k \{ k \} ]$$

추가로 장애물/차선 경계 제약을 포함할 수 있습니다.

MPC는 점수 최소화 + 조건 만족을 동시에 풀어냅니다.

---

## 5) 반복 제어 순서

1. 센서로 현재 상태를 읽음 ((x, y, , v))
2. 앞으로 (N)칸의 조작 ((, a)) 후보를 최적화
3. 가장 좋은 해를 찾음(점수 (J) 최소화, 제약조건 만족)
4. 첫 번째 조작만 차량에 적용
5. 다음 0.1 초 후 다시 1 번부터 반복

이 과정을 매우 빠르게 반복해서 차가 계속 “미래를 보며” 주행합니다.

---

## 6) 핵심 요약

MPC는 예측 + 최적화 + 제약조건을 매 주기 반복해, 안전하고 부드러운 주행 입력을 계산하는 방식입니다.

---

## 7) 논문 MPC vs 지금 쓰는 MPC 차이점 (코드 기준 분석)

비교 기준: - 논문: papers/ltv\_mpc.md (Gutjahr 2017) - 현재 구현:  
src/bisa/src/ltv\_mpc/\*.cpp, src/bisa/src/mpc\_path\_tracker.cpp.cpp

### A. 같은 점 (핵심 뼈대는 거의 같음)

1. 상태 모델 구조가 같다
  - 둘 다 ( $x=[d_r, \dots, r_r]^T$ ), 입력 ( $u=$ ), 외란 ( $z=_r$ ) 형태
2. 비용함수 기본 형태가 같다
  - 둘 다 ( $d_r$ ), ( $(-r_r)$ ), ( $\cdot$ ), ( $u$ )를 제곱 벌점으로 사용
3. **QP(이차계획)**로 푸는 **LTV-MPC** 구조가 같다
  - 선형 시간가변 모델 + 이차 비용 + 제약조건

### B. 다른 점 (실제 주행에서 중요한 차이)

1. 예측 시간 간격( $T_s$ )과 호라이즌 설정이 다름
  - 논문 설명값: ( $T_s$  0.2s) 중심
  - 현재 코드: 기본 ( $T_s=0.05s$ ) (20Hz), 실제 설정은 cav\_config.yaml 에서 horizon: 60 사용
  - 의미: 지금 코드는 더 자주 재계산하고, 더 촘촘하게 반응함
2. 장애물/차선 제약 방식이 단순화됨
  - 논문: ( $d_{\{i\}}(k)$ ,  $d_{\{i\}}(k)$ )를 시간에 따라 바꾸며(동적 장애물 포함) 직접 제약
  - 현재 코드: 기본은 입력/곡률 제약 + 선택적 lateral\_bound(좌우 대칭 경계) 중심
  - 즉, 논문처럼 “장애물마다 시점별 경계”를 QP 에 넣는 구조는 현재 코드에 직접 구현되어 있지 않음
3. 곡률 제약의 물리 모델 차이
  - 논문: Kamm's circle 기반으로 ( $v(k)$ , )에 따라 () 상한/하한이 시간가변
  - 현재 코드: kappa\_min ~ kappa\_max 하드 박스 제약(대부분 고정값 또는 파라미터 환산값)
4. **Soft constraint** 적용 범위가 다름
  - 논문: 슬랙을 특정 출력(주로 ( $d_1, d_2, d_3$ ))과 짧은 구간 ( $N_S < N$ )에 선택 적용
  - 현재 코드: lateral\_bound 를 쓰면 전체 예측구간 ( $N$ )에 대해 슬랙 2 개( $\epsilon_u$ ,  $\epsilon_l$ )를 공통 적용
5. **MPC** 밖의 보정 로직이 많이 추가됨
  - 현재 mpc\_path\_tracker.cpp 에는 다음이 추가:
    - path fallback
    - path hold 보정
    - off-path recovery(이탈 복귀)
    - overshoot/oscillation guard
    - ( $v$ ,) rate limit
  - 논문은 주로 MPC 최적화 자체를 중심으로 설명하고, 이런 실전 보정층은 거의 없음
6. 속도 프로파일 생성 방식
  - 논문: “미래 속도 프로파일이 주어진다”는 전제

- 현재 코드: 경로 곡률로부터 내부적으로 ( $v$ )를 자동 생성하고, 이후 추가 규칙으로 감속 보정

## C. 한 줄 결론

현재 MPC 는 논문 **LTV-MPC** 의 핵심 수학 구조는 유지하면서, 실제 시뮬레이터 안정주행을 위해 장애물 제약은 단순화하고, 대신 **MPC** 외부 보정 로직을 크게 강화한 버전입니다.

---

## 8) 현재 코드의 Pure Pursuit(퓨어퍼슈트) 보조 역할 (수학적으로)

### A. 기본 퓨어퍼슈트 식

현재 코드의 보조 조향 핵심은 아래 식입니다.

$$[\_pp] = [$$

- ( $v$ ): 차량 속도
- ( $\delta$ ): 차량 진행 방향과 lookahead 목표점 방향의 각도 차
- ( $L_d$ ): 차량과 lookahead 점 사이 거리

해석: - ( $\delta$ )가 크면 ( $\delta$ )가 커져 ( $\_pp$ ) 증가 -> 더 빠르게 회전 - ( $L_d$ )가 작으면 ( $\_pp$ ) 증가 -> 더 강하게 경로 복귀 - ( $\delta$ )이면 ( $\_pp$ ) 0 -> 이미 맞춰진 상태라 조향 감소

### B. MPC 와 혼합되는 식

코드에서는 퓨어퍼슈트를 단독이 아니라 MPC 와 섞어서 사용합니다.

$$[\delta] = (1-\alpha)\delta_{mpc} + \alpha\delta_{pp}$$

- ( $\alpha$ ): 혼합 비율(상황에 따라 변함)
- 정상 주행: ( $\alpha$ ) 작음 -> MPC 중심
- 경로 이탈 증가: ( $\alpha$ ) 커짐 -> 퓨어퍼슈트 보정 강화

즉, MPC 해가 약해질 때도 경로 쪽으로 복귀시키는 보조 항이 생깁니다.

### C. Path-hold 단계의 추가 보정식

코드는 기하학 항(퓨어퍼슈트 형태) 외에 오차 기반 보정도 더합니다.

$$[\delta_{hold}] = [\delta_{geo}] + [\delta_{heading}] + [\delta_{cte}] \quad [\delta_{heading}] = k_h e_{\delta}, \quad \delta_{cte} = -k_d e_{\dot{\delta}} |v|$$

- ( $e_{\delta}$ ): heading 오차
- ( $e_{\dot{\delta}}$ ): 횡방향 오차(CTE)

그래서 실제로는: - 퓨어퍼슈트(기하 복귀) + heading 보정 + 횡오차 보정을 합쳐 - 경로 재진입과 진동 억제를 동시에 노립니다.

## D. 논문 대비 차이 핵심

- 논문(주요 내용): LTV-MPC 최적화 내부 구성 중심
- 현재 코드: MPC 바깥에 퓨어퍼슈트 기반 보조층을 추가해 실주행 안정성을 강화

한 줄 수식 요약:

$$[ = (1-),_{\text{mpc}} + ]$$

이 식이 현재 코드에서 퓨어퍼슈트가 “어떤 도움을 주는지”를 가장 직접적으로 보여줍니다.