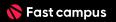
4-15 Stochastic Trajectory
Optimization for
Motion Planning
(STOMP)



강의 요약

01

문제 정의

- 목적 함수
 - → Smooth + obs
- 설계 변수
- → I raj.
 - → 시작점, range

02

Gradient Descent

- Configuration 의 관계를 고려한 (Covariance) 업데이트
- Smoothness 에서 차용한 M matrix

03

직관적 이해

- 초기값을 당기고 펴는 과정
- 조금씩 찾아가는 과정

$$\min_{oldsymbol{x} \in \mathcal{X}} f(oldsymbol{x}) \ g_i(oldsymbol{x}) \leq 0, \ h_j(oldsymbol{x}) = 0$$

충돌이 일어나지 않는 최단 경로를 찾기

- $oldsymbol{egin{align*} egin{align*} oldsymbol{egin{align*} oldsymbol{oldsymbol{egin{align*} oldsymbol{egin{align*} oldsymbol{egin{align*} oldsymbol{egin{align*} oldsymbol{egin{align*} oldsymbol{egin{align*} oldsymbol{egin{align*} oldsymbol{egin{align*} oldsymbol{egin{align*} oldsymbol{egin{align*} oldsymbol{ol}}}}}}}}} oldsimbol{oldsymbol{oldsymbol{oldsymbol{oldsymbo$
- $oldsymbol{eta}$ 목적 함수 (objective function) $J(\mathbf{q}) = \underbrace{J_{\mathrm{smooth}}(\mathbf{q})}_{\mathrm{bbs}} + \underbrace{J_{\mathrm{obs}}(\mathbf{q})}_{\mathrm{obs}}$

trajectory smoothness

obstacle avoidance

● 제약 조건 (constraints)

$$egin{cases} q_0 &= q_{ ext{start}}, \ q_{N-1} &= q_{ ext{goal}}, \ orall i &= 1, \dots, N-2: \quad \ell \, \leq \, q_i \, \leq \, u, \ \|\, q_i - q_{i-1}\| \, \leq \, v_{ ext{max}} \, \Delta t, \quad i = 1, \dots, N-1, \ \|\, q_{i+1} - 2 \, q_i + q_{i-1}\| \, \leq \, a_{ ext{max}} \, (\Delta t)^2, \quad i = 1, \dots, N-2, \ q_i \, \in \, C_{ ext{free}} \quad ext{for } i = 0, \dots, N-1 \end{cases}$$

- 솔루션을 찾는 방법: 경사 하강법 (Gradient Descent) 기법을 응용
 - → 반복이 핵심!

$$\mathbf{q}^{(k+1)} \ = \ \mathbf{q}^{(k)} \ - \ lpha \, M^{-1} \left[
abla J_{\mathrm{smooth}} ig(\mathbf{q}^{(k)} ig) \ + \
abla J_{\mathrm{obs}} ig(\mathbf{q}^{(k)} ig)
ight]$$

$$\min_{oldsymbol{x} \in \mathcal{X}} f(oldsymbol{x}) \ g_i(oldsymbol{x}) \leq 0, \ h_j(oldsymbol{x}) = 0$$

충돌이 일어나지 않는 최단 경로를 찾기

- $oldsymbol{egin{align*} oldsymbol{eta}} oldsymbol{eta} & ext{ days design variables)} \ oldsymbol{oldsymbol{q}} = \left[\,q_0,\,q_1,\,\ldots,\,q_{N-1}\,
 ight]^ op, \quad q_i \in \mathbb{R}^d \ egin{align*} oldsymbol{eta} \end{array}$
- ullet 목적 함수 (objective function) $J({f q}) = J_{
 m smooth}({f q}) + J_{
 m obs}({f q})$

trajectory smoothness

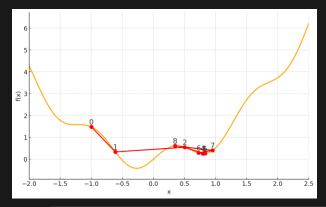
obstacle avoidance

● 제약 조건 (constraints)

$$egin{cases} \left\{ egin{aligned} q_0 &= q_{ ext{start}}, \ q_{N-1} &= q_{ ext{goal}}, \end{aligned}
ight. \ \left. orall i = 1, \ldots, N-2: \quad \ell \, \leq \, q_i \, \leq \, u, \ \left\| \, q_i - q_{i-1}
ight\| \, \leq \, v_{ ext{max}} \, \Delta t, \quad i = 1, \ldots, N-1, \ \left\| \, q_{i+1} - 2 \, q_i + q_{i-1}
ight\| \, \leq \, a_{ ext{max}} \, (\Delta t)^2, \quad i = 1, \ldots, N-2, \ q_i \, \in \, C_{ ext{free}} \quad ext{for } i = 0, \ldots, N-1 \end{cases}$$

- 솔루션을 찾는 방법: 경사 하강법 (Gradient Descent) 기법을 응용
 - → 반복이 핵심!

$$\mathbf{q}^{(k+1)} \ = \ \mathbf{q}^{(k)} \ - \ lpha \, M^{-1} \left[
abla J_{\mathrm{smooth}} ig(\mathbf{q}^{(k)} ig) \ + \
abla J_{\mathrm{obs}} ig(\mathbf{q}^{(k)} ig)
ight]$$



1. Deterministic

(Local Minimum Problem)

$$\min_{oldsymbol{x} \in \mathcal{X}} f(oldsymbol{x}) \ g_i(oldsymbol{x}) \leq 0, \ h_j(oldsymbol{x}) = 0$$

충돌이 일어나지 않는 최단 경로를 찾기

- $oldsymbol{egin{align*} oldsymbol{egin{align*} oldsymbol{oldsymbol{egin{align*} oldsymbol{egin{align*} oldsymbol{ol}}}}}}}}}}}}}}}} olengtare{oldsymbol{olds$
- 목적 함수 (objective function)

$$J(\mathbf{q}) = \underbrace{J_{\mathrm{smooth}}(\mathbf{q})}_{\mathrm{trajectory smoothness}} + \underbrace{J_{\mathrm{obs}}(\mathbf{q})}_{\mathrm{obstacle avoidance}}$$

2. Differentiable (정형적 성격)

● 제약 조건 (constraints)

$$egin{cases} \left\{ egin{aligned} q_0 &= q_{ ext{start}}, \ q_{N-1} &= q_{ ext{goal}}, \end{aligned}
ight. \ \left. egin{aligned} \forall i = 1, \ldots, N-2: \quad \ell \, \leq \, q_i \, \leq \, u, \ \left\| \, q_i - q_{i-1}
ight\| \, \leq \, v_{ ext{max}} \, \Delta t, \quad i = 1, \ldots, N-1, \ \left\| \, q_{i+1} - 2 \, q_i + q_{i-1}
ight\| \, \leq \, a_{ ext{max}} \, (\Delta t)^2, \quad i = 1, \ldots, N-2, \ q_i \, \in \, C_{ ext{free}} \quad ext{for } i = 0, \ldots, N-1 \end{cases}$$

- 솔루션을 찾는 방법: 경사 하강법 (Gradient Descent) 기법을 응용
 - → 반복이 핵심!

$$\mathbf{q}^{(k+1)} = \mathbf{q}^{(k)} - \alpha M^{-1} \left[\nabla V_{\mathrm{smooth}} (\mathbf{q}^{(k)}) + \nabla J_{\mathrm{obs}} (\mathbf{q}^{(k)}) \right]$$

- 2 1 0 -2.0 -1.5 -1.0 -0.5 0.0 0.5 1.0 1.5 2.0 2.5
- 1. Deterministic

(Local Minimum Problem)

$$\min_{oldsymbol{x} \in \mathcal{X}} f(oldsymbol{x}) \ g_i(oldsymbol{x}) \leq 0, \ h_j(oldsymbol{x}) = 0$$

충돌이 일어나지 않는 최단 경로를 찾기

- $oldsymbol{egin{align*} oldsymbol{eta}} oldsymbol{eta} & oldsymbol{eta} & oldsymbol{eta} \left[\, q_0, \, q_1, \, \ldots, \, q_{N-1} \,
 ight]^ op, \quad q_i \in \mathbb{R}^d \ \end{pmatrix}$
- 목적 함수 (objective function)
- 제약 조건 (constraints)

```
egin{cases} \left\{ egin{aligned} q_0 &= q_{	ext{start}}, \ q_{N-1} &= q_{	ext{goal}}, \end{aligned} 
ight. \ orall i &= 1, \ldots, N-2: \quad \ell \, \leq \, q_i \, \leq \, u, \ \left\| \, q_i - q_{i-1} \right\| \, \leq \, v_{	ext{max}} \, \Delta t, \quad i = 1, \ldots, N-1, \ \left\| \, q_{i+1} - 2 \, q_i + q_{i-1} \right\| \, \leq \, a_{	ext{max}} \, (\Delta t)^2, \quad i = 1, \ldots, N-2, \ q_i \, \in \, C_{	ext{free}} \quad 	ext{for } i = 0, \ldots, N-1 \end{cases}
```

• 솔루션을 찾는 방법: 경사 하강법 (Gradient Descent) 기법을 응용

$$\min_{oldsymbol{x} \in \mathcal{X}} f(oldsymbol{x}) \ g_i(oldsymbol{x}) \leq 0, \ h_j(oldsymbol{x}) = 0$$

충돌이 일어나지 않는 최단 경로를 찾기

- $oldsymbol{egin{align*} egin{align*} oldsymbol{q} & ext{distributions} \ oldsymbol{q} = \left[\,q_0,\,q_1,\,\ldots,\,q_{N-1}\,
 ight]^ op, & q_i \in \mathbb{R}^d \ \end{pmatrix}}$
- 목적 함수 (objective function)
- 제약 조건 (constraints)
- $egin{cases} \left\{ egin{aligned} q_0 &= q_{ ext{start}}, \ q_{N-1} &= q_{ ext{goal}}, \end{aligned}
 ight. \ \forall i = 1, \ldots, N-2: \quad \ell \, \leq \, q_i \, \leq \, u, \ \left\| \, q_i q_{i-1} \right\| \, \leq \, v_{ ext{max}} \, \Delta t, \quad i = 1, \ldots, N-1, \ \left\| \, q_{i+1} 2 \, q_i + q_{i-1} \right\| \, \leq \, a_{ ext{max}} \, (\Delta t)^2, \quad i = 1, \ldots, N-2, \ q_i \, \in \, C_{ ext{free}} \quad ext{for } i = 0, \ldots, N-1 \end{cases}$

- 솔루션을 찾는 방법: 경사 하강법 (Gradient Descent) 기법을 응용
 - → 1. Stochastic Gradient Descent 를 활용한다면?

$$\min_{oldsymbol{x} \in \mathcal{X}} f(oldsymbol{x}) \ g_i(oldsymbol{x}) \leq 0, \ h_j(oldsymbol{x}) = 0$$

충돌이 일어나지 않는 최단 경로를 찾기

- 설계 변수 (design variables) $\mathbf{q} = \left[\,q_0,\,q_1,\,\ldots,\,q_{N-1}\,
 ight]^ op,\quad q_i\in\mathbb{R}^d$
- 목적 함수 (objective function)
 ⇒ 2. 미분 불가능한 함수라면?
- 제약 조건 (constraints)

```
egin{cases} q_0 &= q_{	ext{start}}, \ q_{N-1} &= q_{	ext{goal}}, \ orall i &= 1, \dots, N-2: \quad \ell \, \leq \, q_i \, \leq \, u, \ \|\, q_i - q_{i-1}\| \, \leq \, v_{	ext{max}} \, \Delta t, \quad i = 1, \dots, N-1, \ \|\, q_{i+1} - 2 \, q_i + q_{i-1}\| \, \leq \, a_{	ext{max}} \, (\Delta t)^2, \quad i = 1, \dots, N-2, \ q_i \, \in \, C_{	ext{free}} \quad 	ext{for } i = 0, \dots, N-1 \end{cases}
```

- 솔루션을 찾는 방법: 경사 하강법 (Gradient Descent) 기법을 응용
 - → 1. Stochastic Gradient Descent 를 활용한다면?

목적 함수: 미분이 가능하지 않아도 됨 (자유도가 높음)

● 장애물 회피 (Obstacle)

- 경로의 부드러움 (Smoothness)
- 속도 제한 비용 (Velocity limit)

목적 함수: 미분이 가능하지 않아도 됨 (자유도가 높음)

● 장애물 회피 (Obstacle)

$$c_{
m obs}({f q}_t) = egin{cases} lpha \cdot (d_{
m safe} - d({f q}_t))^2 & ext{if } d({f q}_t) < d_{
m safe} \ 0 & ext{otherwise} \ d({f q}_t)$$
: configuration ${f q}_t$ 에서 가장 가까운 장애물까지 거리 $lpha$: penalty scale $d_{
m safe}$: 안전 거리 threshold

● 경로의 부드러움 (Smoothness)

$$c_{ ext{smooth}} = \sum_{t=2}^{T-1} \|\mathbf{q}_{t+1} - 2\mathbf{q}_t + \mathbf{q}_{t-1}\|^2$$

● 속도 제한 비용 (Velocity limit)

$$c_{ ext{vel}} = \sum_{t=1}^{T-1} \max(0, \|\dot{\mathbf{q}}_t\| - v_{ ext{max}})^2$$

$$c(\xi) = w_{ ext{obs}} \cdot c_{ ext{obs}} + w_{ ext{smooth}} \cdot c_{ ext{smooth}} + w_{ ext{vel}} \cdot c_{ ext{vel}}$$

목적 함수: 미분이 가능하지 않아도 됨 (자유도가 높음)

● 장애물 회피 (Obstacle)

$$c_{
m obs}({f q}_t) = egin{cases} lpha \cdot (d_{
m safe} - d({f q}_t))^2 & ext{if } d({f q}_t) < d_{
m safe} \ 0 & ext{otherwise} \ d({f q}_t) : ext{configuration } {f q}_t$$
에서 가장 가까운 장애물까지 거리 $lpha$: penalty scale $d_{
m safe}$: 안전 거리 threshold

→ Binary Collision Check 도 활용 가능

• 경로의 부드러움 (Smoothness)

$$c_{ ext{smooth}} = \sum_{t=2}^{T-1} \|\mathbf{q}_{t+1} - 2\mathbf{q}_t + \mathbf{q}_{t-1}\|^2$$

● 속도 제한 비용 (Velocity limit)

$$c_{ ext{vel}} = \sum_{t=1}^{T-1} \max(0, \|\dot{\mathbf{q}}_t\| - v_{ ext{max}})^2$$

$$c(\xi) = w_{ ext{obs}} \cdot c_{ ext{obs}} + w_{ ext{smooth}} \cdot c_{ ext{smooth}} + w_{ ext{vel}} \cdot c_{ ext{vel}}$$

$$\min_{oldsymbol{x} \in \mathcal{X}} f(oldsymbol{x}) \ g_i(oldsymbol{x}) \leq 0, \ h_j(oldsymbol{x}) = 0$$

충돌이 일어나지 않는 최단 경로를 찾기

- $oldsymbol{\Phi}$ 설계 변수 (design variables) $oldsymbol{\mathbf{q}} = \left[\,q_0,\,q_1,\,\ldots,\,q_{N-1}\,
 ight]^ op,\quad q_i\in\mathbb{R}^d$
- 목적 함수 (objective function)

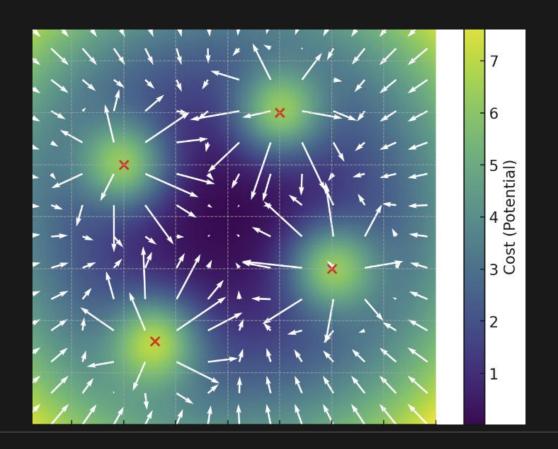
$$c(\xi) = w_{ ext{obs}} \cdot c_{ ext{obs}} + w_{ ext{smooth}} \cdot c_{ ext{smooth}} + w_{ ext{vel}} \cdot c_{ ext{vel}}$$

● 제약 조건 (constraints)

```
egin{cases} q_0 &= q_{	ext{start}}, \ q_{N-1} &= q_{	ext{goal}}, \ orall i &= 1, \dots, N-2: \quad \ell \, \leq \, q_i \, \leq \, u, \ \|\, q_i - q_{i-1}\| \, \leq \, v_{	ext{max}} \, \Delta t, \quad i = 1, \dots, N-1, \ \|\, q_{i+1} - 2 \, q_i + q_{i-1}\| \, \leq \, a_{	ext{max}} \, (\Delta t)^2, \quad i = 1, \dots, N-2, \ q_i \, \in \, C_{	ext{free}} \quad 	ext{for } i = 0, \dots, N-1 \end{cases}
```

- 솔루션을 찾는 방법: 경사 하강법 (Gradient Descent) 기법을 응용
 - → 1. Stochastic Gradient Descent 를 활용한다면?

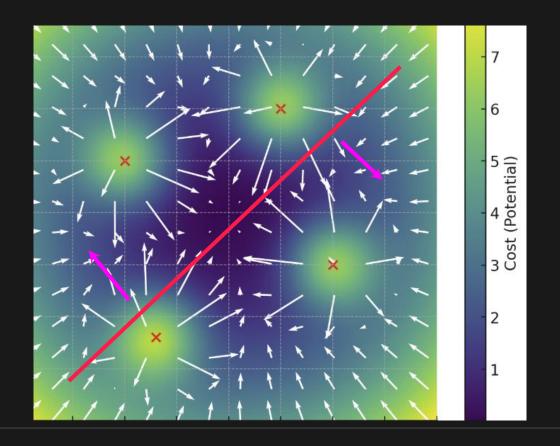
경사 하강법: Deterministic vs. Stochastic





경사 하강법: Deterministic vs. Stochastic

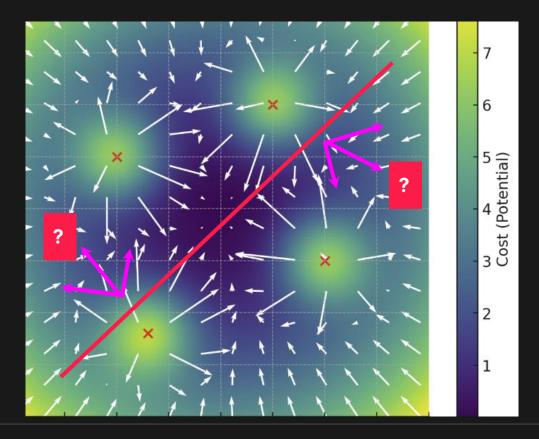
● Deterministic: 수식에 의해 명확하게 업데이트 방향이 결정이 됨





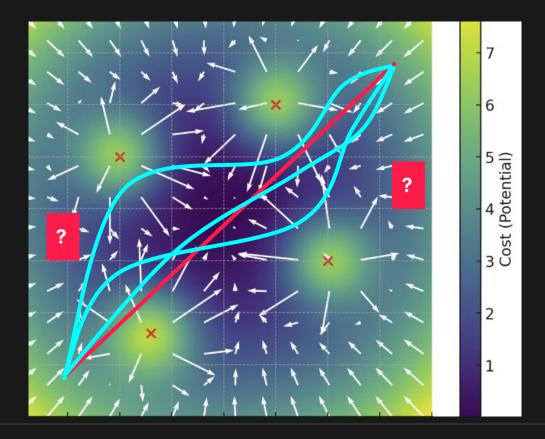
경사 하강법: Deterministic vs. Stochastic

- Deterministic: 수식에 의해 명확하게 업데이트 방향이 결정이 됨
- Stochastic: 핵심은 샘플링을 통해 경사하강을 진행한다는 점



경사 하강법: Deterministic vs. Stochastic

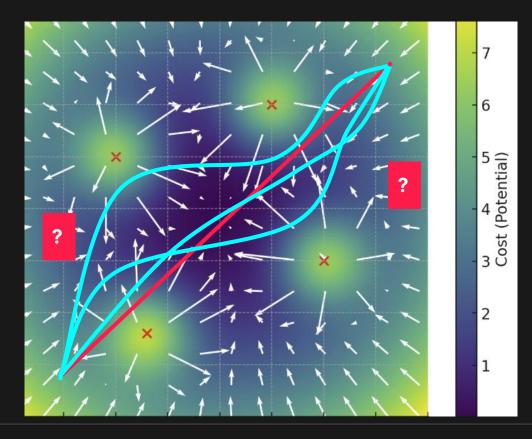
- Deterministic: 수식에 의해 명확하게 업데이트 방향이 결정이 됨
- Stochastic: 핵심은 샘플링을 통해 경사하강을 진행한다는 점



경사 하강법: Deterministic vs. Stochastic

- Deterministic: 수식에 의해 명확하게 업데이트 방향이 결정이 됨
- Stochastic: 핵심은 샘플링을 통해 경사하강을 진행한다는 점

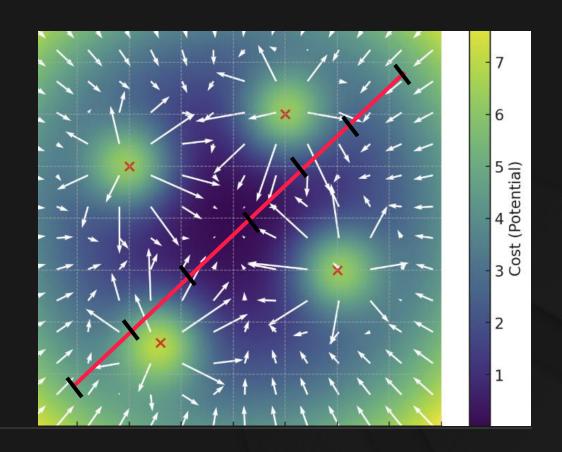
"여러 개의 노이즈를 trajectory에 더해보고, 그 중 성능이 좋은 (cost가 낮은) trajectory 쪽으로 이동한다"



"여러 개의 노이즈를 trajectory에 더해보고, 그 중 성능이 좋은 (cost가 낮은) trajectory 쪽으로 이동한다"

초기 trajectory $oldsymbol{\xi} = [\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_T]$

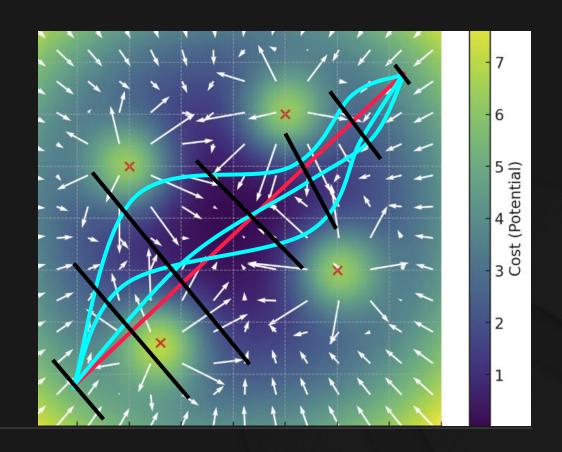
$$\xi_k = \xi + \epsilon_k \quad ext{where } \epsilon_k \sim \mathcal{N}(0, \Sigma)$$



"여러 개의 노이즈를 trajectory에 더해보고, 그 중 성능이 좋은 (cost가 낮은) trajectory 쪽으로 이동한다"

초기 trajectory $oldsymbol{\xi} = [\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_T]$

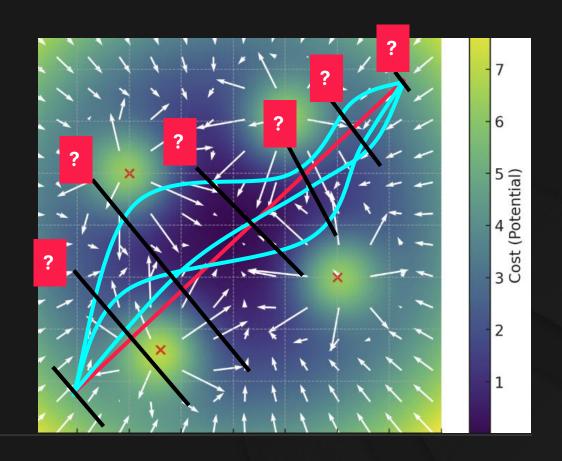
$$\xi_k = \xi + \epsilon_k \quad ext{where } \epsilon_k \sim \mathcal{N}(0, \Sigma)$$



"여러 개의 노이즈를 trajectory에 더해보고, 그 중 성능이 좋은 (cost가 낮은) trajectory 쪽으로 이동한다"

|초기 trajectory $oldsymbol{\xi} = [\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_T]$

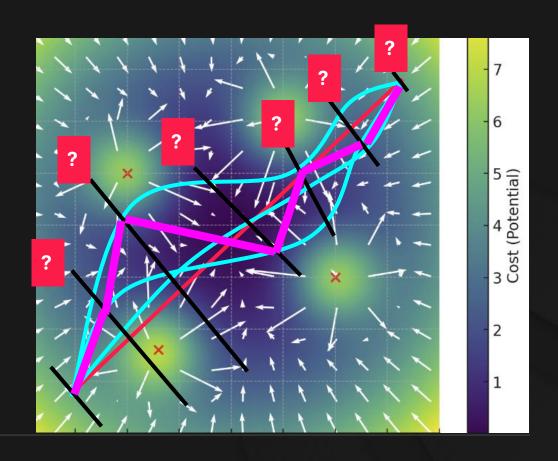
$$\xi_k = \xi + \epsilon_k \quad ext{where } \epsilon_k \sim \mathcal{N}(0, \Sigma)$$



"여러 개의 노이즈를 trajectory에 더해보고, 그 중 성능이 좋은 (cost가 낮은) trajectory 쪽으로 이동한다"

|초기 trajectory $oldsymbol{\xi} = [\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_T]$

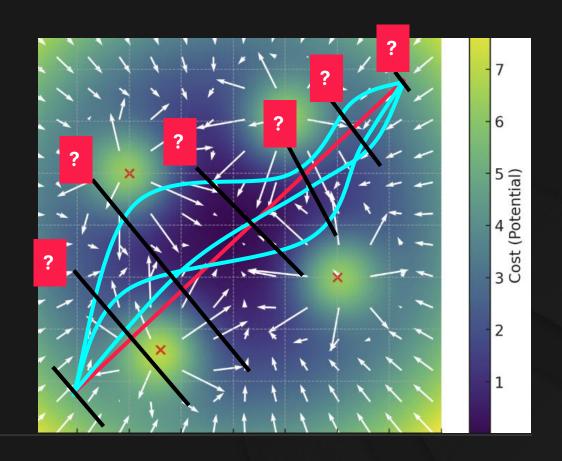
$$\xi_k = \xi + \epsilon_k \quad ext{where } \epsilon_k \sim \mathcal{N}(0, \Sigma)$$



"여러 개의 노이즈를 trajectory에 더해보고, 그 중 성능이 좋은 (cost가 낮은) trajectory 쪽으로 이동한다"

|초기 trajectory $oldsymbol{\xi} = [\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_T]$

$$\xi_k = \xi + \epsilon_k \quad ext{where } \epsilon_k \sim \mathcal{N}(0, \Sigma)$$



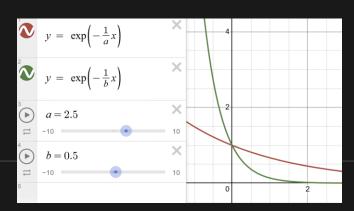
"여러 개의 노이즈를 trajectory에 더해보고, 그 중 성능이 좋은 (cost가 낮은) trajectory 쪽으로 이동한다"

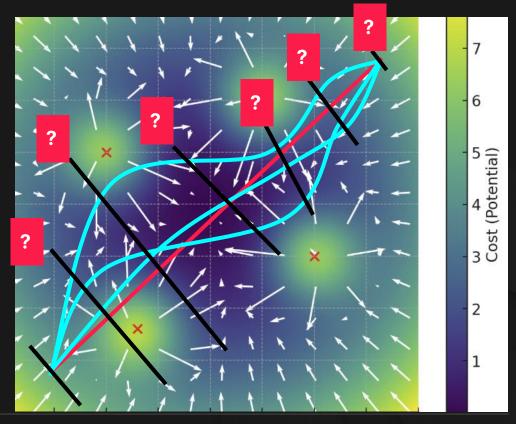
초기 trajectory $oldsymbol{\xi} = [\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_T]$

$$\xi_k = \xi + \epsilon_k \quad ext{where } \epsilon_k \sim \mathcal{N}(0, \Sigma)$$

$$w_t^{(k)} = rac{\exp\left(-rac{1}{\lambda}c_t^{(k)}
ight)}{\sum_{k=1}^K \exp\left(-rac{1}{\lambda}c_t^{(k)}
ight)}$$

각 trajectory 마다, timestep 마다 계산





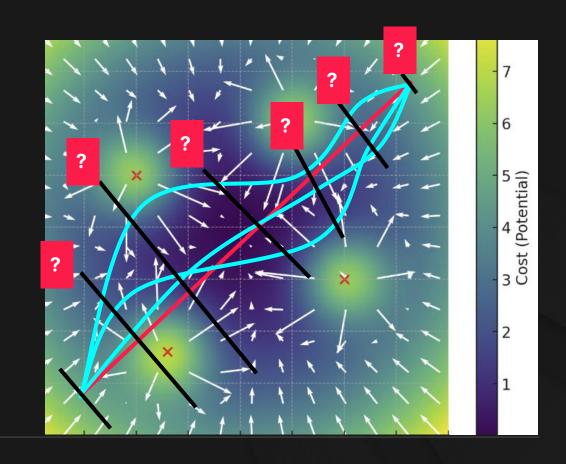
"여러 개의 노이즈를 trajectory에 더해보고, 그 중 성능이 좋은 (cost가 낮은) trajectory 쪽으로 이동한다"

초기 trajectory $\mathbf{\xi} = [\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_T]$

$$\xi_k = \xi + \epsilon_k \quad ext{where } \epsilon_k \sim \mathcal{N}(0, \Sigma)$$

$$w_t^{(k)} = rac{\exp\left(-rac{1}{\lambda}c_t^{(k)}
ight)}{\sum_{k=1}^K \exp\left(-rac{1}{\lambda}c_t^{(k)}
ight)}$$
 각 trajectory 마다, timestep 마다 계산

$$\xi \leftarrow \xi + \sum_{k=1}^K w^{(k)} \epsilon_k$$
 노이즈가 기여한 만큼 업데이트



"여러 개의 노이즈를 trajectory에 더해보고, 그 중 성능이 좋은 (cost가 낮은) trajectory 쪽으로 이동한다"

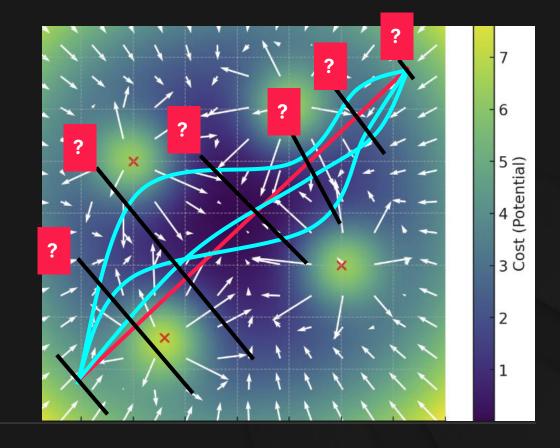
초기 trajectory $\mathbf{\xi} = [\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_T]$

$$\xi_k = \xi + \epsilon_k \quad ext{where } \epsilon_k \sim \mathcal{N}(0, \Sigma)$$

 $w_t^{(k)} = rac{\exp\left(-rac{1}{\lambda}c_t^{(k)}
ight)}{\sum_{k=1}^K \exp\left(-rac{1}{\lambda}c_t^{(k)}
ight)}$

각 trajectory 마다, timestep 마다 계산

$$\xi \leftarrow \xi + \sum_{k=1}^K w^{(k)} \epsilon_k$$
 노이즈가 기여한 만큼 업데이트



반복!

$$\min_{oldsymbol{x} \in \mathcal{X}} f(oldsymbol{x}) \ g_i(oldsymbol{x}) \leq 0, \ h_j(oldsymbol{x}) = 0$$

충돌이 일어나지 않는 최단 경로를 찾기

- $oldsymbol{\Phi}$ 설계 변수 (design variables) $oldsymbol{\mathbf{q}} = \left[\,q_0,\,q_1,\,\ldots,\,q_{N-1}\,
 ight]^ op,\quad q_i\in\mathbb{R}^d$
- 목적 함수 (objective function)

$$c(\xi) = w_{ ext{obs}} \cdot c_{ ext{obs}} + w_{ ext{smooth}} \cdot c_{ ext{smooth}} + w_{ ext{vel}} \cdot c_{ ext{vel}}$$

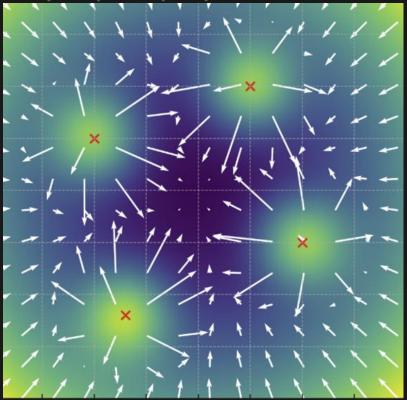
● 제약 조건 (constraints)

$$egin{cases} \left\{ egin{aligned} q_0 &= q_{ ext{start}}, \ q_{N-1} &= q_{ ext{goal}}, \end{aligned}
ight. \ orall i &= 1, \ldots, N-2: \quad \ell \, \leq \, q_i \, \leq \, u, \ \left\| \, q_i - q_{i-1} \right\| \, \leq \, v_{ ext{max}} \, \Delta t, \quad i = 1, \ldots, N-1, \ \left\| \, q_{i+1} - 2 \, q_i + q_{i-1} \right\| \, \leq \, a_{ ext{max}} \, (\Delta t)^2, \quad i = 1, \ldots, N-2, \ q_i \, \in \, C_{ ext{free}} \quad ext{for } i = 0, \ldots, N-1 \end{cases}$$

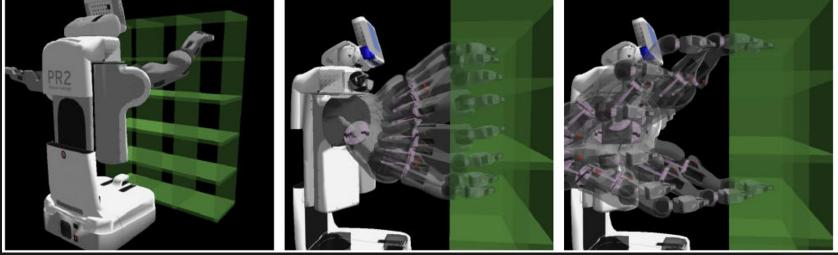
• 솔루션을 찾는 방법: 경사 하강법 (Gradient Descent) 기법을 응용

초기 trajectory
$$\xi = [\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_T]$$
 $\xi_k = \xi + \epsilon_k \quad ext{where } \epsilon_k \sim \mathcal{N}(0, \Sigma)$ $w_t^{(k)} = \dfrac{\exp\left(-\frac{1}{\lambda}c_t^{(k)}\right)}{\sum_{k=1}^K \exp\left(-\frac{1}{\lambda}c_t^{(k)}\right)}$ $\xi \leftarrow \xi + \sum_{k=1}^K w^{(k)} \epsilon_k$

C-Space (Joint Space)



Work Space (Task Space)



CHOMP vs. STOMP

CHOMP

- Deterministic
- 미분가능 여부: 필수
- 계산 속도: 비교적 빠름
- 안정성: 민감함 (local minimum, initialization)
- 활용: 비교적 부드러운 환경



STOMP

- Stochastic
- 미분가능 여부: 불필요
- 계산 속도: 샘플 개수에 따라 다름
- 안정성: 비교적 robust (exploration)
- 활용: 복잡한 환경, 비선형 목적함숙

강의 요약

01

문제 정의

미분 불가능한 목적함수 사용 가능 (자유도가 높다) 02

Gradient Descent

노이즈를 추가한 초기 경로에 가중치를 부여하여 업데이트 (Stochastic) 03

직관적 이해

- 초기값을 당기고 펴는 과정
- 조금씩 찾아가는 과정