

정윤석 / Yunsuk Jeung

Profile

- 서울대학교 김호영 교수님 지도하에 있는 마이크로 유체역학 및 소프트 메터 연구실 소속 석사과정 생이며, 반도체 공정 및 메타모핑 기계시스템 연구단 소속으로 연구하고 있습니다.

Project Experience

- Nano/bio-mechanical system technology using micro/nanoscale porous flow control(2018)
- 2018-2019 SEMES-SNU Research project(2018-2019)
- Research Center for Metamorphing Mechanical Systems(2019-2020)

Main Research Theme

- Line/plane extraction based LiDAR Odometry

Interests/Knowledge

- Microfluidics
- Flow visualize with highspeed camera
- Line/plane extraction from LiDAR pointcloud and odometry calculation
- Project Management

Skills

- C++, MATLAB, Solidworks, ROS(C++), LiDAR Data Processing Using PCL Library

Education

- 2013-2018 B.S. in Mechanical Engineering, SNU
- 2018-2019 M.S. in Mechanical Engineering, SNU
- 2019-2020 Ph.D. candidate in Mechanical Engineering, SNU

목차

1. SEMES-SNU Research Project - Fingering instability during a semiconductor cleaning process
2. Main Research Theme - Feature extraction from LiDAR Pointcloud and Odometry estimation
3. Appendix

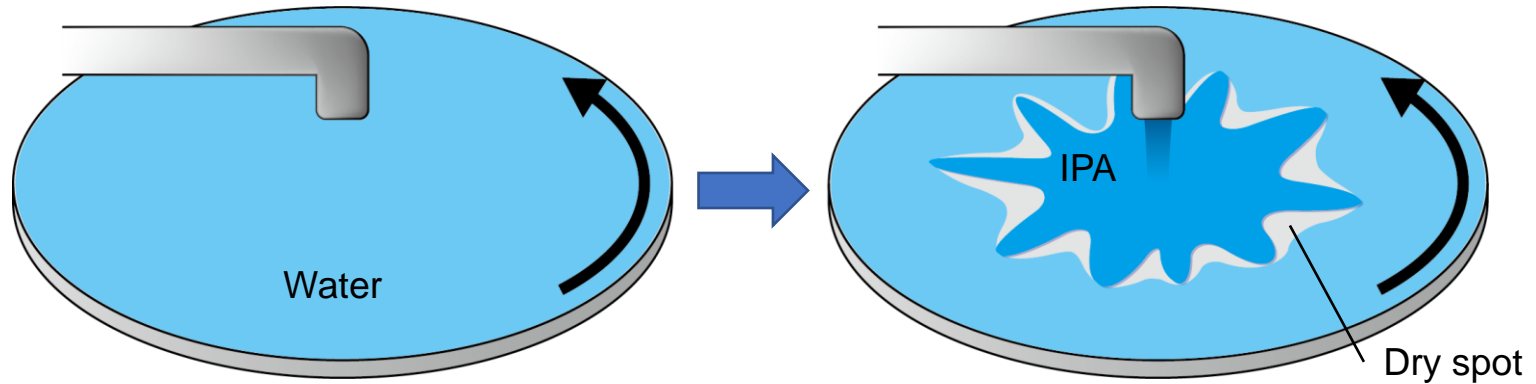
SEMES-SNU Research Project

Fingering instability during a semiconductor cleaning process

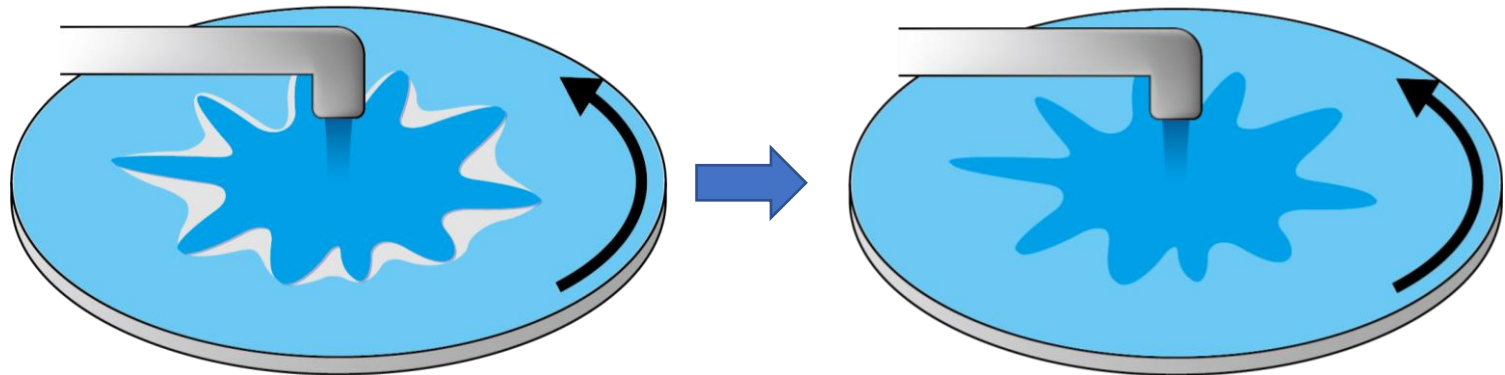
: Analyzing physics of cleaning process, project management

300 mm wafer에서의 fingering 메커니즘 및 개선안 도출

- 연구 개요 : 반도체 세정 공정에서 물이 덮여진 웨이퍼에 IPA를 분사 시 finger 모양의 dry spot이 생성되어 공정의 효율이 떨어지는 문제 발생



- 연구 목표 : Dry spot의 생성 원인을 파악하고 해결 방안 제시

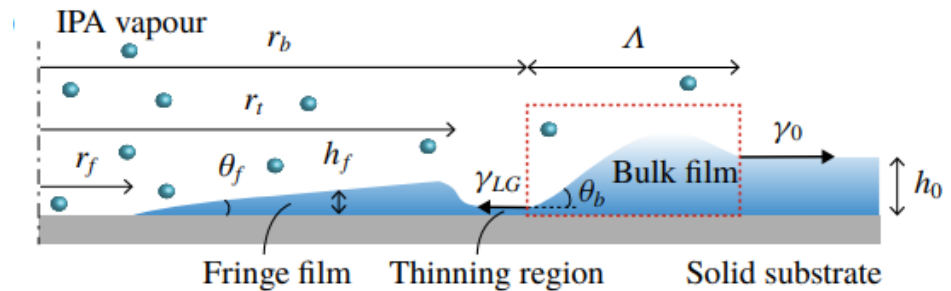
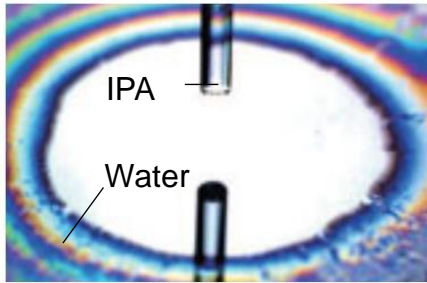


300 mm wafer에서의 fingering 메커니즘 및 개선안 도출

• 연구 방향 설정

- 주어진 예산 내에서 실제 세정 공정을 재현할 수 있도록 장비 선정
- 선행 연구를 바탕으로 가설 설정

S. Kim et al., *J. Fluid Mech.*, 2019



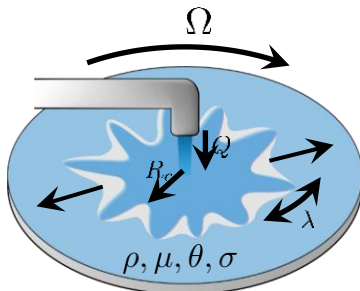
IPA 증기가 물 표면에 달라붙어 표면장력 감소 → Dry spot 발생

- 선행 연구를 바탕으로 주요 parameter 선정

Liquid property: 표면장력, 밀도, 점도

Controllable parameter: 접촉각, 유량, 회전속도

➔ R_c 를 최대화 하여 dry spot 제거



$$R_c = f(Q, \rho, \mu, \theta, \sigma, \Omega)$$

R_c : Dry spot이 생성되는 위치

Q : 액체 유량

θ : 접촉각

ρ : 밀도

σ : 표면장력

μ : 점도

Ω : 회전 속도

300 mm wafer에서의 fingering 메커니즘 및 개선안 도출

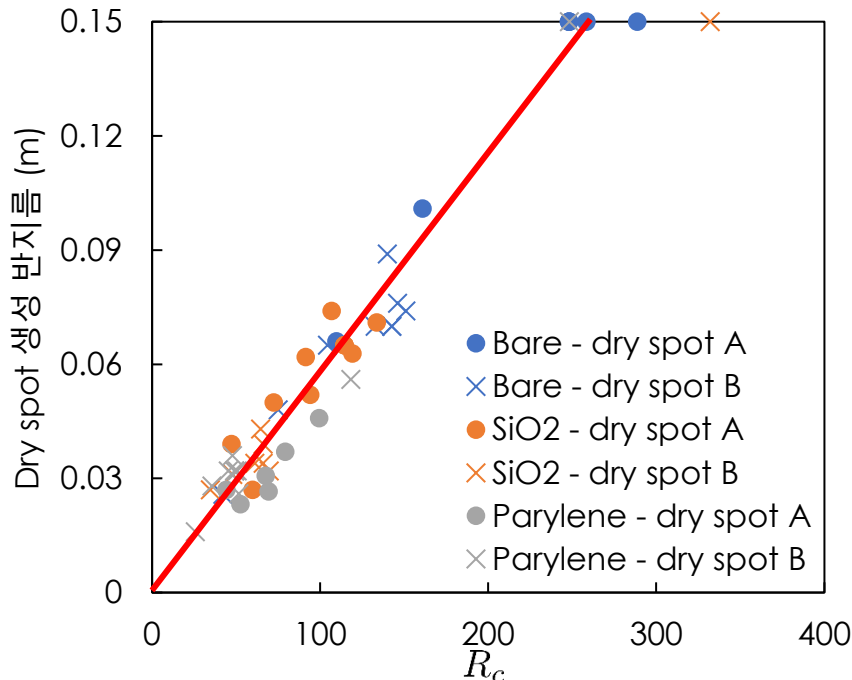
• 이론 모델 검증

- 주요 parameter를 이용한 이론 모델 생성

$$R_c \sim \rho_d^{\frac{1}{4}} \left(\frac{1}{\mu_d} \right)^{\frac{1}{4}} \mu_{cv}^{\frac{3}{4}} Q_g^{\frac{1}{2}} \Omega^{\frac{1}{2}} \left(\frac{1}{\Delta\sigma} \right)^{\frac{3}{4}} \left(\frac{1}{h_c} - \frac{1}{h_0} \right)^{\frac{3}{4}}$$

$$h_0 = \left(\frac{3}{2\pi} \frac{Q_d \mu_d}{\rho_d \Omega^2 r_0^2} \right)^{\frac{1}{3}}, h_c = \left(\frac{3}{2\pi} \frac{Q_d \mu_d}{\rho_d \Omega^2 r_f^2} \right)^{\frac{1}{3}}, r_f = 0.13 \frac{\rho_d \Omega^{\frac{1}{2}} Q_d^{\frac{5}{4}}}{\mu_d^{\frac{1}{4}} \sigma_d^{\frac{3}{4}} (1 - \cos \theta)^{\frac{3}{4}}}$$

- 실험과 비교



정확한 이론 모델을 통해 변수를
조절하여 dry post 제거 가능

Main Research Theme:

Feature extraction from LiDAR pointcloud and
Odometry estimation

: Feature extraction from Lidar pointcloud & LiDAR Odometry

Keyword: Feature(line, plane) extraction / LiDAR odometry

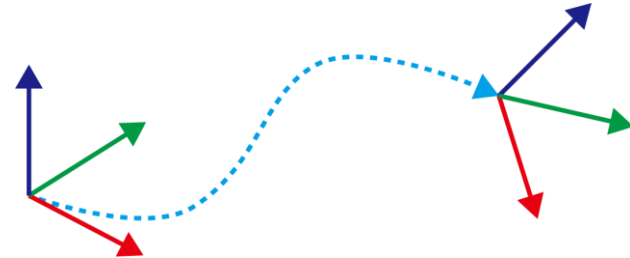
Research Motivation

- 연구 동기 및 개요 : 실내 3D LiDAR Odometry

- 연구 동기 : LiDAR를 이용한 실내 Odometry algorithm 구현

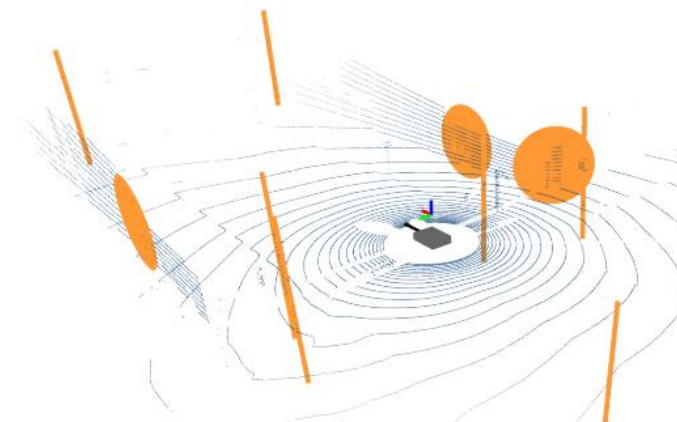
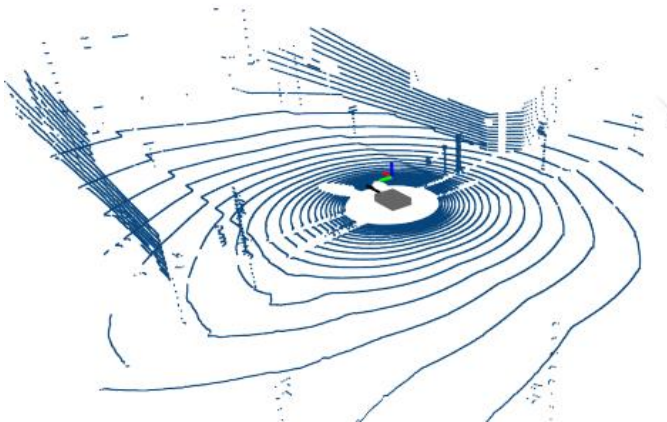


Velodyne VLP-16



$$Z_{Lidar, 1:t} \longrightarrow \hat{x}_{1:t}$$

- 연구 개요 : 실내 환경의 주요 구성 요소인 Line-plane feature extraction algorithm을 이용하여 Odometry 계산



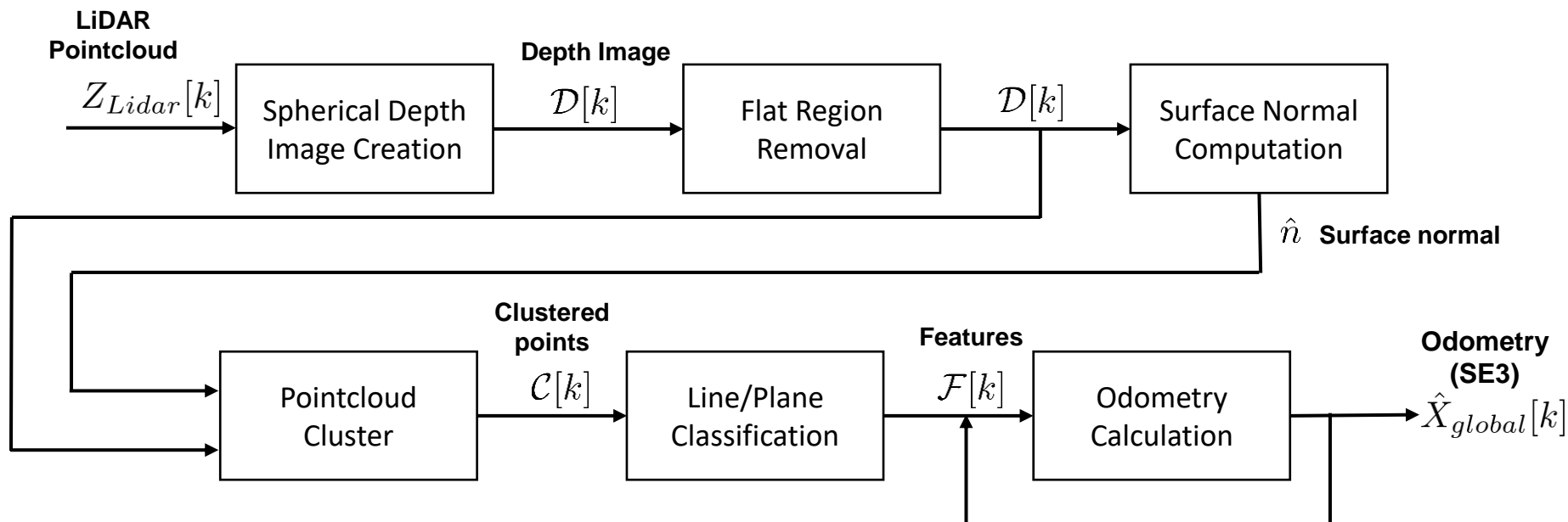
J. Serafin et al., *IROS*, 2016

LiDAR Odometry Algorithm Block Diagram

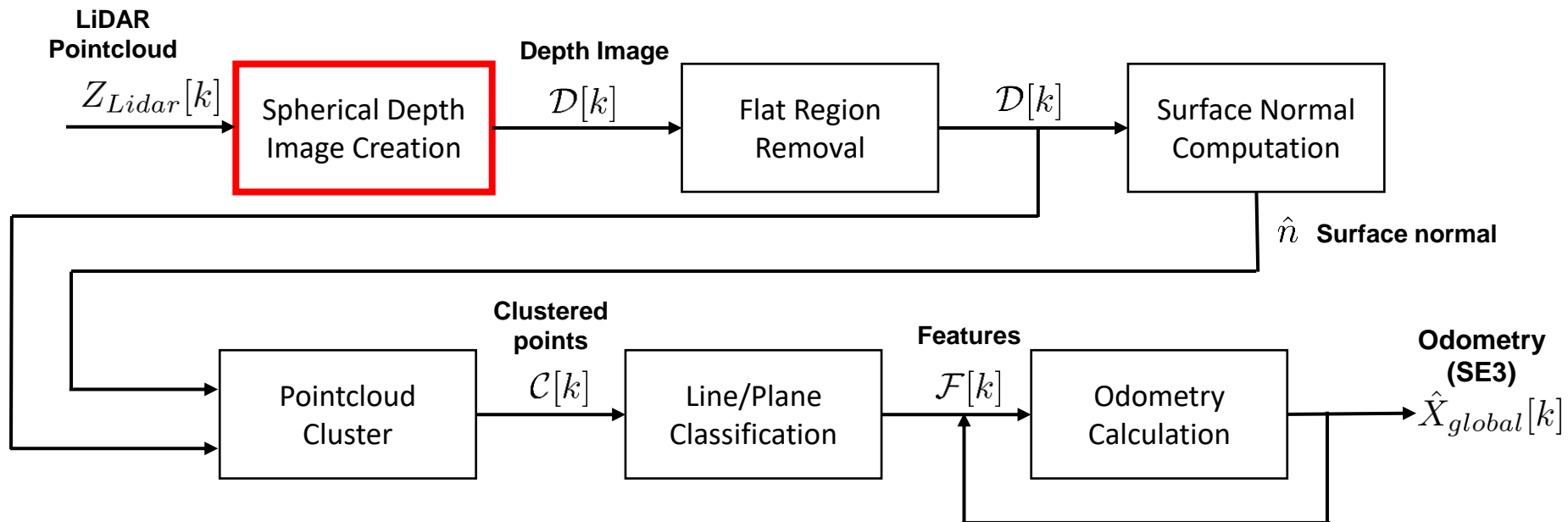
- Lidar Odometry Algorithm : Line & plane matching

- 알고리즘 개요 :

- 1) LiDAR로 수집된 Pointcloud를 spherical depth image로 변환 후 flat region 제거
- 2) Covariance matrix와 Adaptive window size를 이용한 surface normal 계산
- 3) Extract line and plane
- 4) Find odometry with matching features



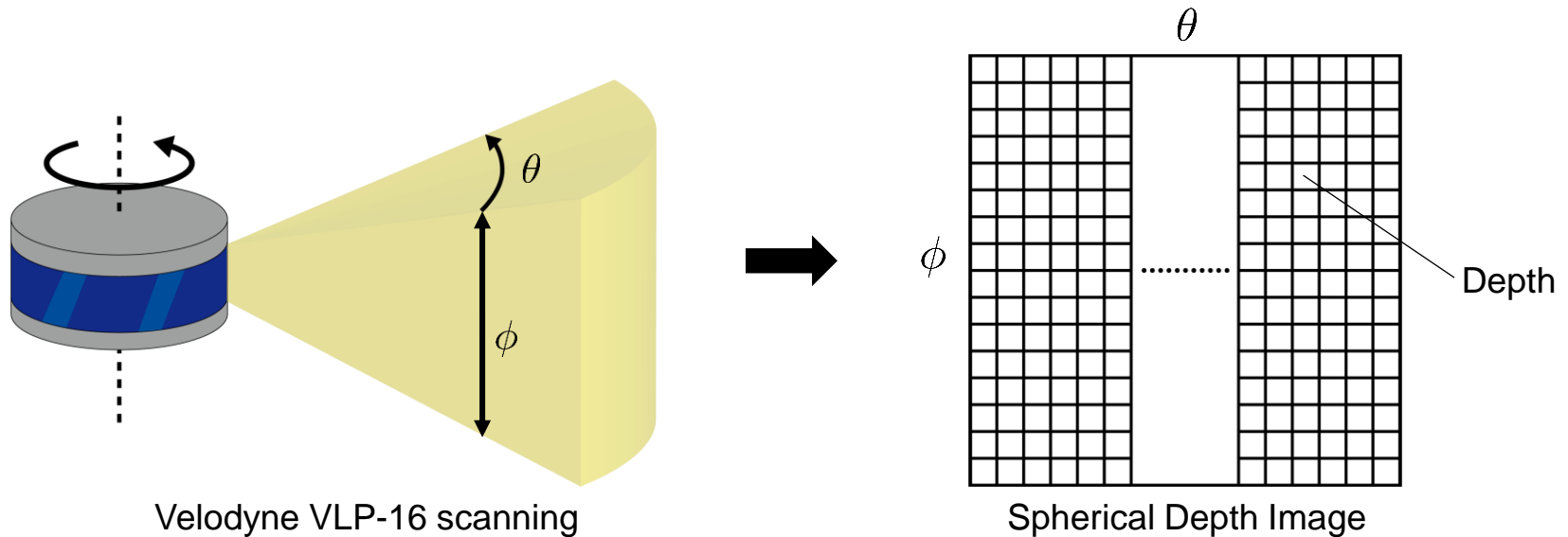
Spherical Depth Image Creation



Spherical Depth Image Creation

- Spherical Depth image

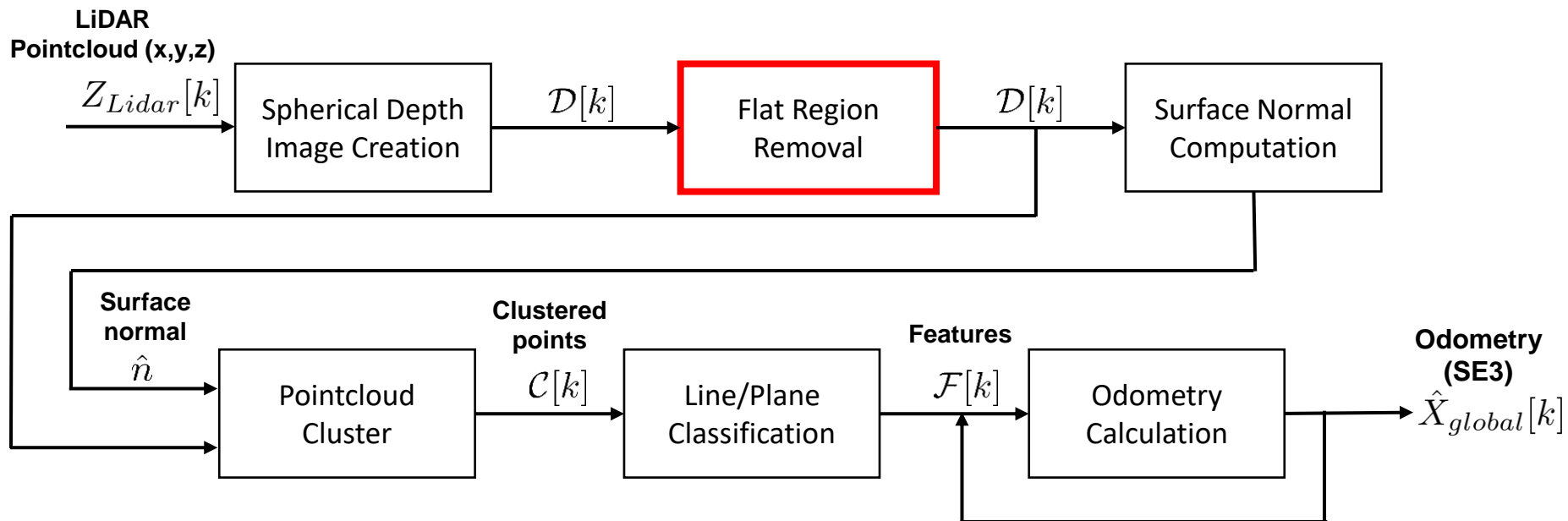
- Velodyne LiDAR의 scan 방식을 반영하여 polar coordinate 사용



$$p = (x, y, z) \quad \longrightarrow \quad \Phi(p) = s \quad \longrightarrow \quad s = (\theta, \phi, d)$$

- Spherical depth image를 이용함으로써 LiDAR로 스캔한 3차원 데이터를 2차원 배열과 depth로 표현 가능
- VLP-16: 16채널 0.2 deg의 분해능으로 16 x 1800 의 image 사용

Flat region removal



Flat region removal

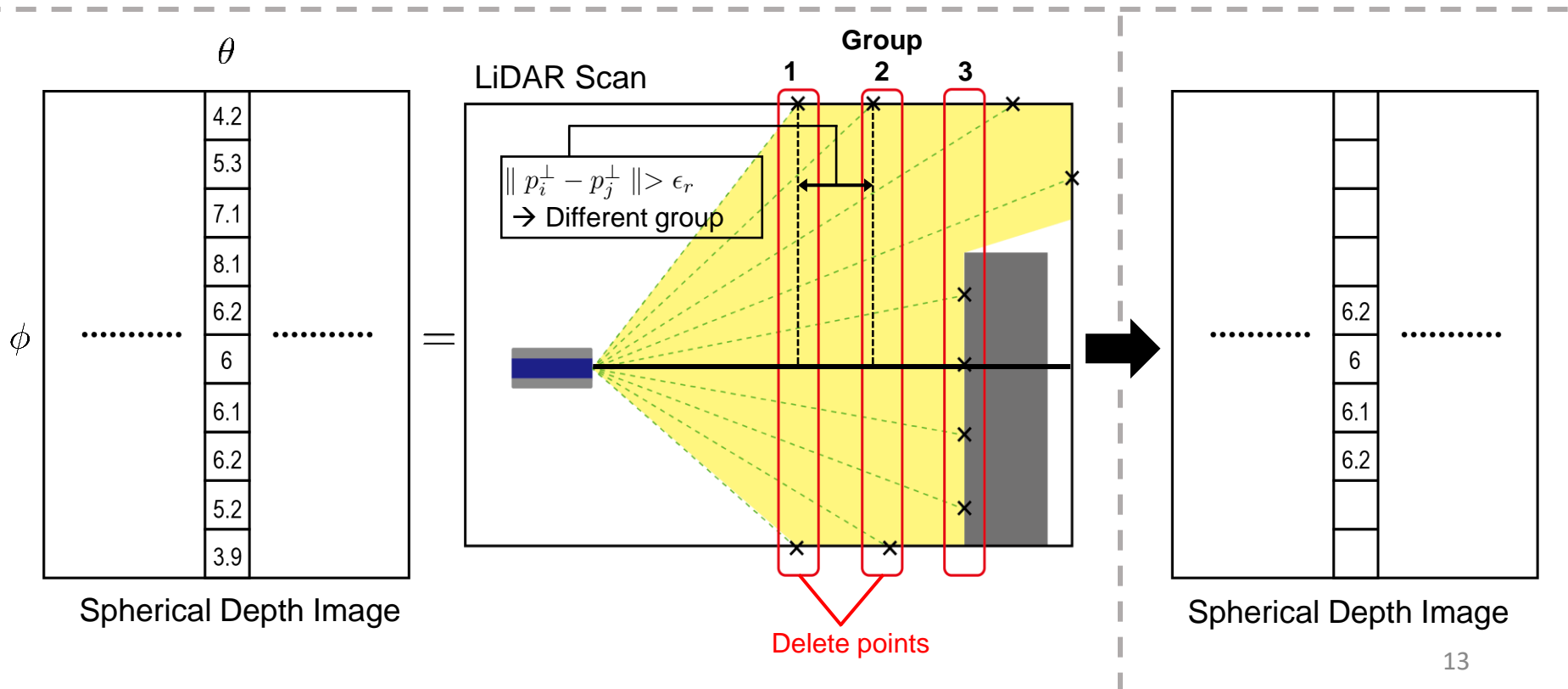
• Flat region removal

- 중요하지 않은 ground를 제거함으로써 계산 속도 향상 및 odometry 정확도 향상
- Points를 LiDAR Local Coordinated XY평면에 정사영 시킨 후 vertical points 판단

1) $\|p_i^\perp - p_j^\perp\| < \epsilon_r \rightarrow$ Group points (p^\perp : xy 평면에 정사영 된 좌표)

2) Number of points in group $> \epsilon_t \rightarrow$ Label vertical points

ϵ_r	0.5 m
ϵ_t	3

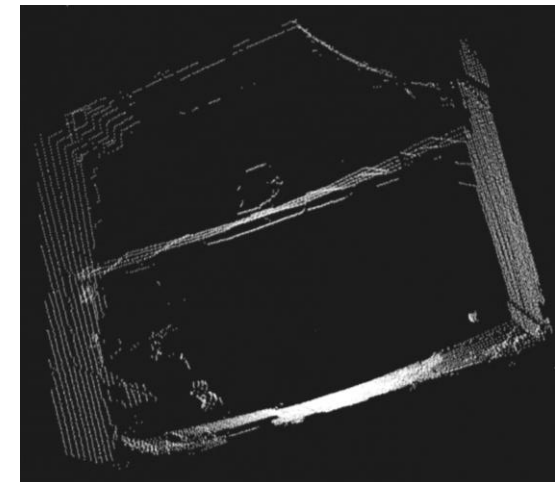
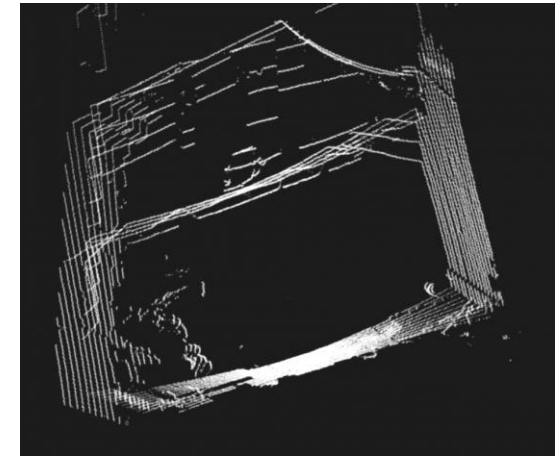


Flat region removal

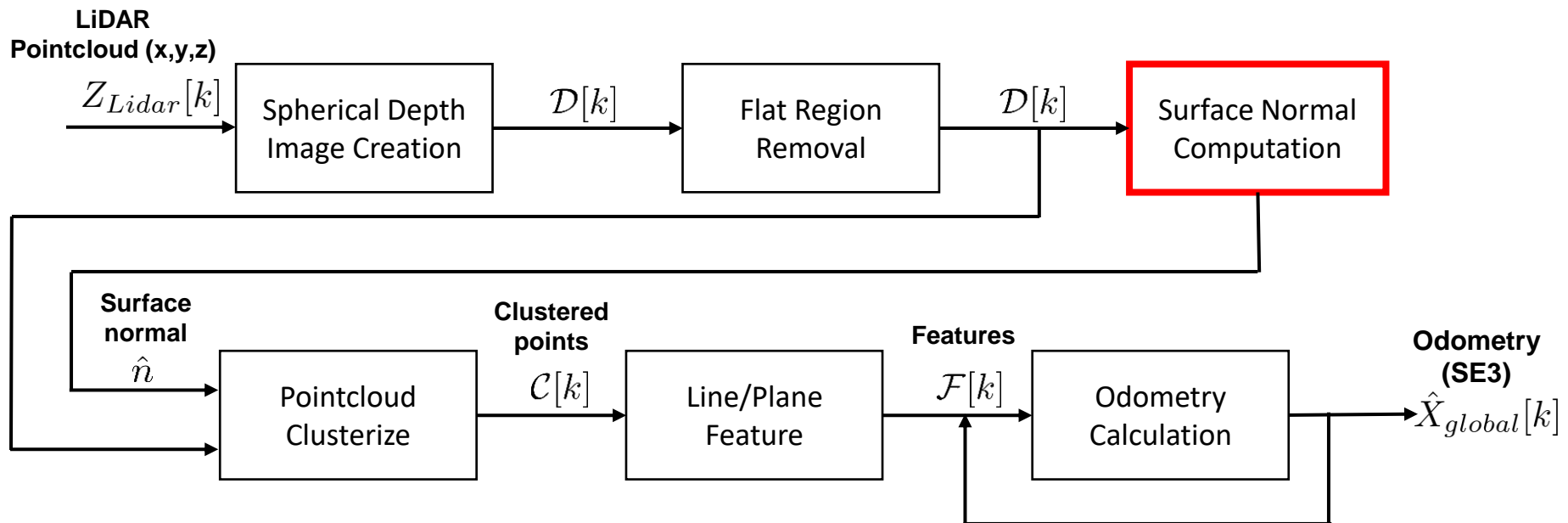
- Algorithm & Result

```
foreach column  $u$  in  $D^p$  do
  foreach row  $v$  in  $D^p$  do
    if  $D_{uv}^P \neq 0$  &&  $p_{uv}$  not labeled vertical then
       $n \leftarrow 0$ 
       $p \leftarrow p_{uv}$ 
      foreach row  $w$  in  $D^p$  greater than  $v$  do
        If  $\|p_{uv}^\perp - p_{uw}^\perp\| < \epsilon_r$  then
           $n \leftarrow n + 1$ 
           $p \leftarrow p \cup p_{uw}$ 
        end
      end
      If  $n > \epsilon_t$  then
        label all points in  $p$  as vertical
      else
        delete  $p_{uv}$  from  $P$ 
         $D_{uv}^P = 0$ 
      end
    end
  end
end
```

P : Pointcloud
 D^p : Spherical depth image
 p^\perp : 2D projection on the ground



Surface Normal computation



Surface Normal computation

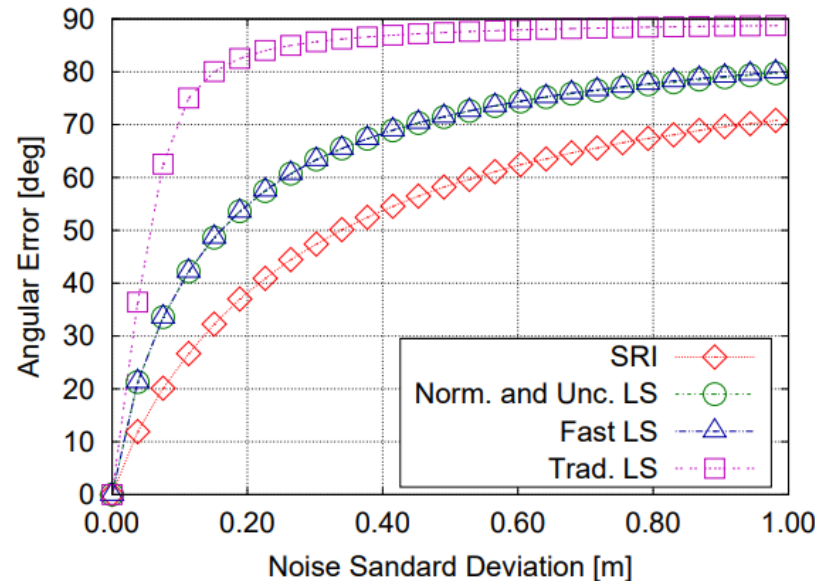
- Feature 분류를 위한 surface normal 계산
 - Integral Image를 이용한 Covariance matrix 계산
 - Covariance matrix \rightarrow eigenvalue decomposition \rightarrow surface normal
 - Feature를 구분하기 위하여 depth change를 바탕으로 adaptive window size 결정
- Covariance Matrix (SRI) vs Least Square (LS) for surface normal

M. Size	Methods	Time (ms) $\pm \sigma$	SUF
3×3	Trad. LS	192.25 ± 5.92	1
	Norm. LS	227.52 ± 4.04	0.85
	Unc. LS	18.87 ± 0.18	10.19
	Fast LS	7.02 ± 0.05	27.39
	SRI	3.66 ± 0.01	52.53
5×5	Trad. LS	195.00 ± 3.04	1
	Norm. LS	237.2 ± 4.04	0.82
	Unc. LS	19.14 ± 0.16	10.18
	Fast LS	7.04 ± 0.03	27.70
	SRI	$3.71 \pm < 0.01$	52.56
7×7	Trad. LS	436.12 ± 7.2	1
	Norm. LS	483.5 ± 3.1	0.9
	Unc. LS	19.14 ± 0.14	15.57
	Fast LS	7.07 ± 0.03	42.14
	SRI	4.33 ± 0.01	68.81
9×9	Trad. LS	436.12 ± 2.58	1
	Norm. LS	489.96 ± 4.8	0.89
	Unc. LS	19.35 ± 0.16	22.54
	Fast LS	7.18 ± 0.12	60.74
	SRI	4.40 ± 0.01	99.12

TABLE II

ACTUAL COMPUTATION TIMES WITH SPEED UP FACTORS (SUF) OVER THE

H. Badino et al., *ICRA*, 2011



→ Covariance Matrix shows faster and more accurate performance

Surface Normal computation

- Feature 분류를 위한 surface normal 계산

- *Integral Image를 이용한 Covariance matrix 계산
- Covariance matrix → eigenvalue decomposition → surface normal
- Feature를 구분하기 위하여 depth change를 바탕으로 adaptive window size 결정

- Integral image

- $\mathcal{I}_o(m, n) = \sum_i^m \sum_j^n \mathcal{O}(i, j)$ (\mathcal{I}_o : integral image, \mathcal{O} : original image)
- 특정 영역 픽셀 값의 합을 4개의 픽셀로 계산 가능 → 계산 속도 향상

*Integral image

10	15	7	9	5	6
9	32	65	45	12	7
7	24	66	65	41	34
9	11	70	89	44	37
32	78	91	78	48	65
64	12	89	58	65	45

Original image of x

$$89+44+37+78+48 \\ +65+58+65+45 = 529$$

10	25	32	41	46	52
19	66	138	192	209	222
26	97	235	354	412	459
35	117	325	533	635	719
67	227	526	812	962	1111
131	303	691	1035	1250	1444

Integral image of x

$$1444-459-691+235 = 529$$

Faster computation

1. Depth image와 같은 크기의 integral image 생성
2. Point의 좌표 (x, y, z) 를 바탕으로 9개의 integral image 생성

$$\mathcal{I}_x, \mathcal{I}_y, \mathcal{I}_z, \mathcal{I}_{xx}, \mathcal{I}_{xy}, \mathcal{I}_{xz}, \mathcal{I}_{yy}, \mathcal{I}_{yz}, \mathcal{I}_{zz}$$

3. Covariance matrix 계산

$$C = \begin{bmatrix} C_{xx} & C_{xy} & C_{xz} \\ C_{xy} & C_{yy} & C_{yz} \\ C_{xz} & C_{yz} & C_{zz} \end{bmatrix} - \begin{bmatrix} C_x \\ C_y \\ C_z \end{bmatrix} \begin{bmatrix} C_x \\ C_y \\ C_z \end{bmatrix}^T$$

C_k : 관심 영역 내 k의 평균

ex) $c_x = 529/9 \approx 59$

S. Holzer et al., IROS, 2012

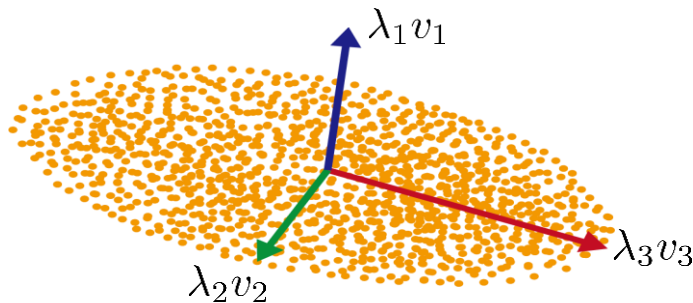
Surface Normal computation

- Covariance matrix eigenvalue decomposition

$$C = \begin{bmatrix} \text{cov}(x, x) & \text{cov}(x, y) & \text{cov}(x, z) \\ \text{cov}(y, z) & \text{cov}(y, y) & \text{cov}(y, z) \\ \text{cov}(z, x) & \text{cov}(z, y) & \text{cov}(z, z) \end{bmatrix}$$

Eigenvalue decomposition ($\lambda_3 \geq \lambda_2 \geq \lambda_1$)

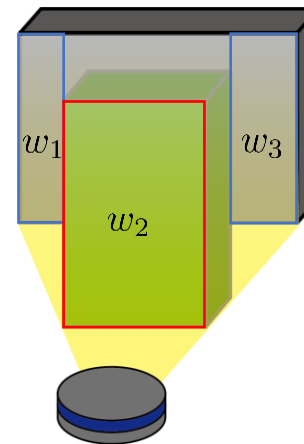
$$C = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \begin{bmatrix} v_1^T \\ v_2^T \\ v_3^T \end{bmatrix}$$



- v_3 : 가장 분산이 큰 방향
- v_2 : v_3 과 수직이면서 가장 분산이 큰 방향
- v_1 : v_1 과 v_2 와 수직 → surface normal

- Adaptive window size

- Depth image에서 depth 변화를 감지하여 window size 결정
- $|d_i - d_j| > \epsilon_w \rightarrow \text{change window}$
 d : Depth of image pixel
- $\epsilon_w = 0.6$



Depth image

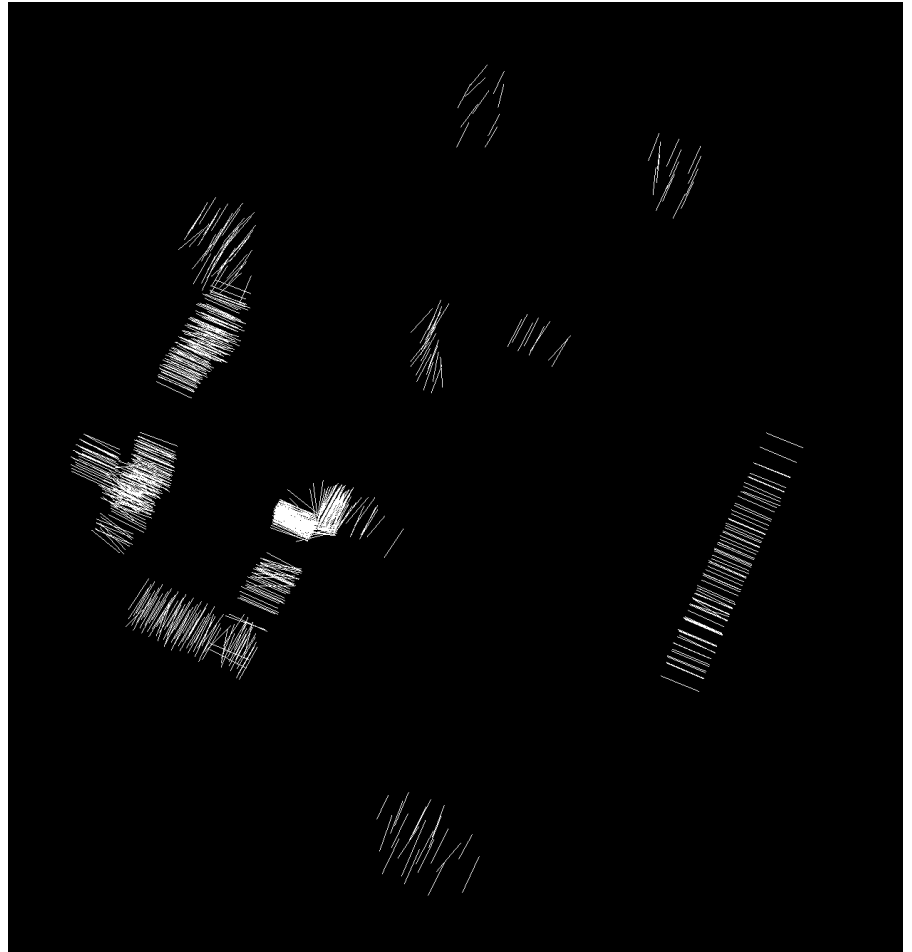
3	3	1	1	1	3	3
3	3	1	1	1	3	3
3	3	1	1	1	3	3
3	3	1	1	1	3	3
3	3	1	1	1	3	3

w_1 w_2 w_3

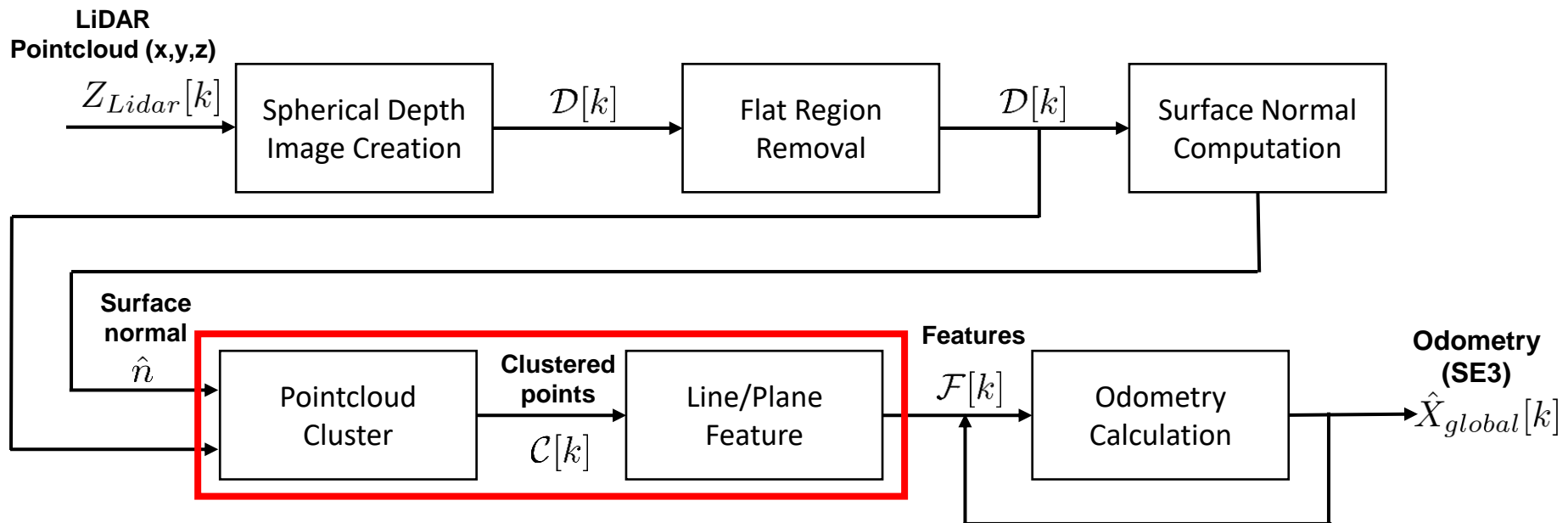
- 각 window 에서의 covariance matrix를 eigenvalue decomposition하여 surface normal 계산

Surface Normal computation

- Result



Feature extraction



Feature extraction

- Cluster pointcloud

- Neighbor points (p)간의 거리 및 surface normal(n) 의 각도 확인 후 point cluster

$$\| p_s - p_i \| = d$$

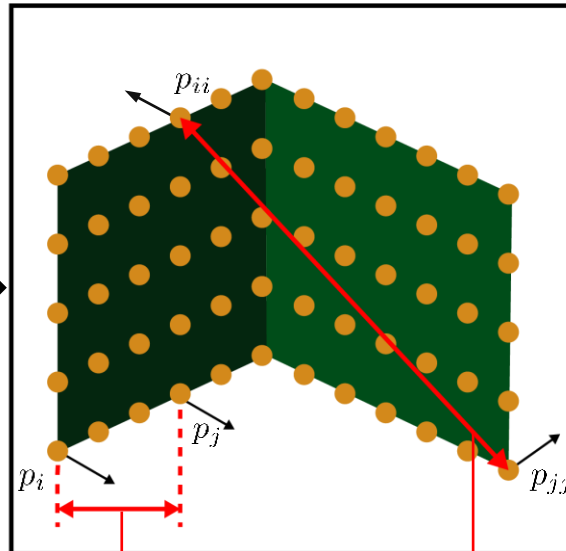
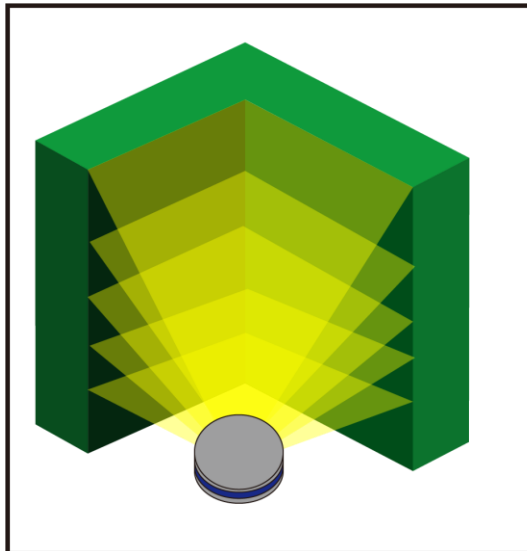
If $d < \epsilon_d \ \&\& \ n_s \cdot n_i > \epsilon_n$ **then**

$C_s \leftarrow C_s \cup p_i$

end

Seed point: p_s, n_s
Neighbor point: p_i, n_i
Clustered pointcloud: \mathcal{C}

ϵ_d	1.0 m
ϵ_n	0.9

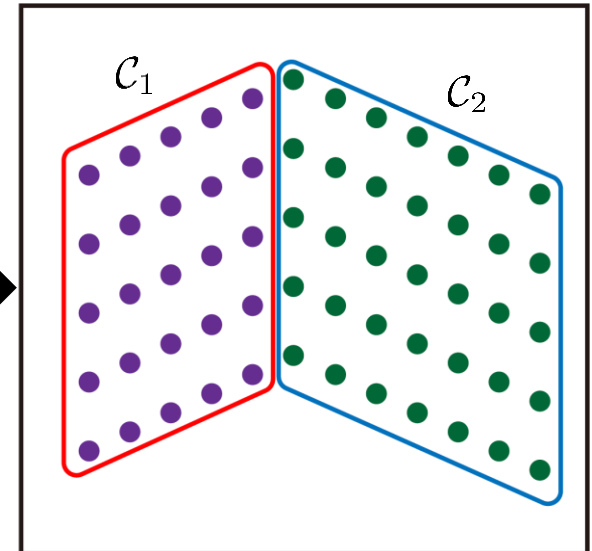


$$\begin{matrix} d = 0.2 \\ n_i \cdot n_j = 1 \end{matrix}$$

$p_i, p_j \rightarrow$ Same cluster

$$\begin{matrix} d = 1.2 \\ n_{ii} \cdot n_{jj} = 0 \end{matrix}$$

$p_{ii}, p_{jj} \rightarrow$ Different cluster



Feature extraction

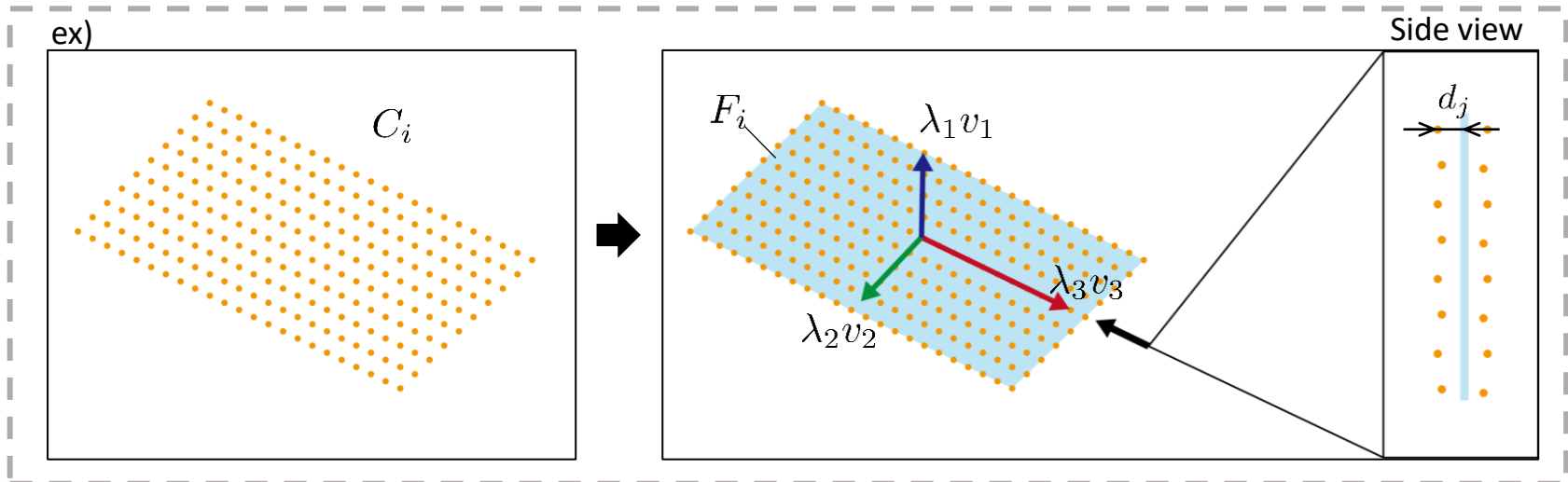
• Line/Plane 분류

- C_i 의 Gaussian distribution $\mathcal{N}_i^c(\mu_i^c, \Sigma_i^c)$ 계산 후 covariance matrix(CM)의 eigenvalue deposition
- Feature 생성 (F_i)

$$\frac{\lambda_1^c + \lambda_2^c}{\lambda_1^c + \lambda_2^c + \lambda_3^c} < \epsilon_l \Rightarrow F_i: \text{Line with origin } \mu_i^c, \text{ direction with } v_3^c$$

$$\frac{\lambda_1^c}{\lambda_1^c + \lambda_2^c + \lambda_3^c} < \epsilon_p \Rightarrow F_i: \text{Plane with origin } \mu_i^c, \text{ normal with } v_1^c$$

ϵ_r	0.02
ϵ_t	0.03



- Valid feature 확인

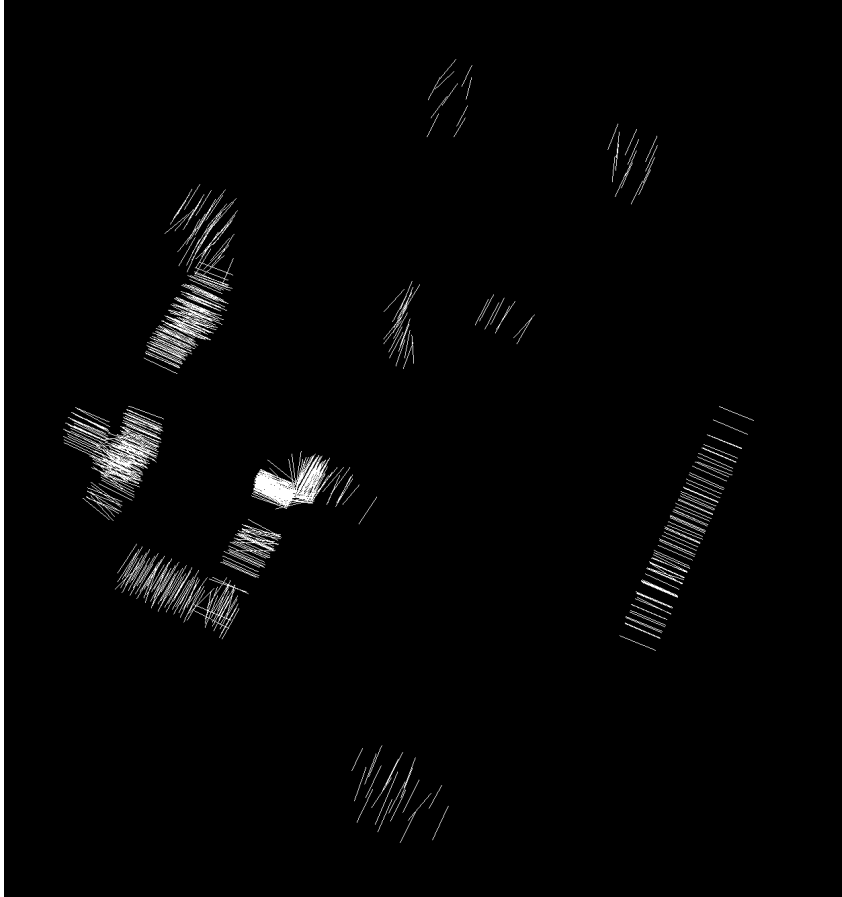
$$e_i = \frac{1}{N_i} \sum d_j, (N_i : \# \text{ of points in } C_i) \quad e_i < \epsilon_f \Rightarrow C_i, F_i \text{ 저장}$$

$$\epsilon_f = 0.1 \text{ m}$$

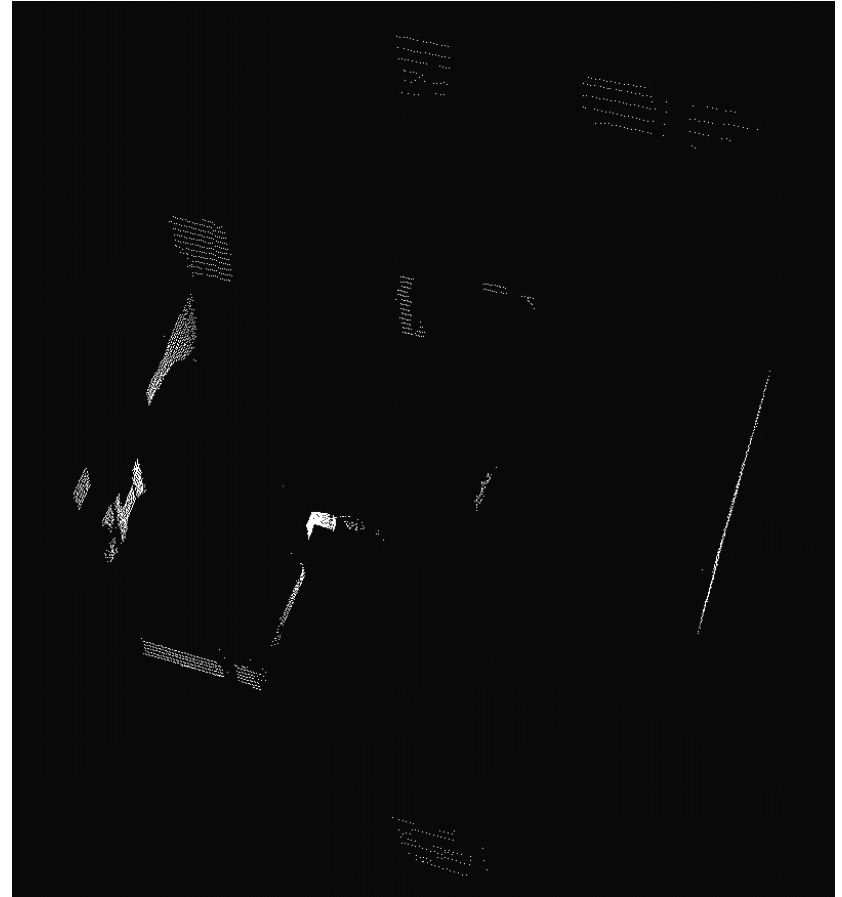
Feature extraction

- **Result**

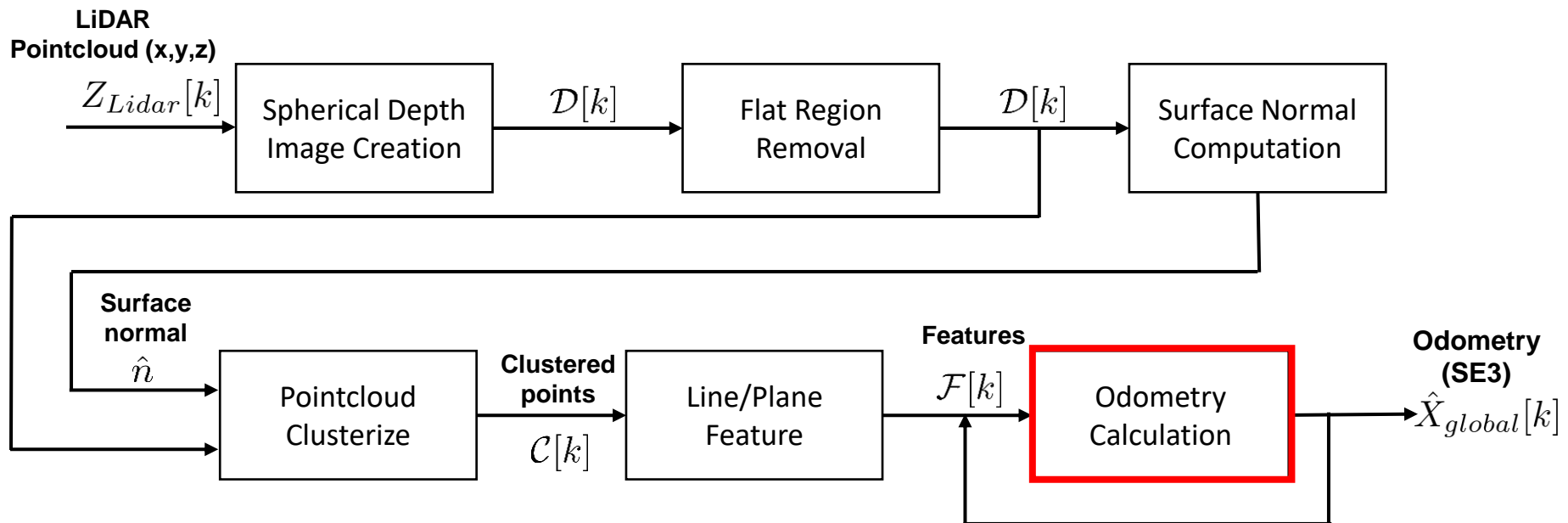
Points & Surface normal



Clustered points to features



LiDAR Odometry computation



LiDAR Odometry computation

- Existing Odometry algorithm

- \mathcal{F}^k 와 \mathcal{F}^{k+n} 사이의 matching features 찾기

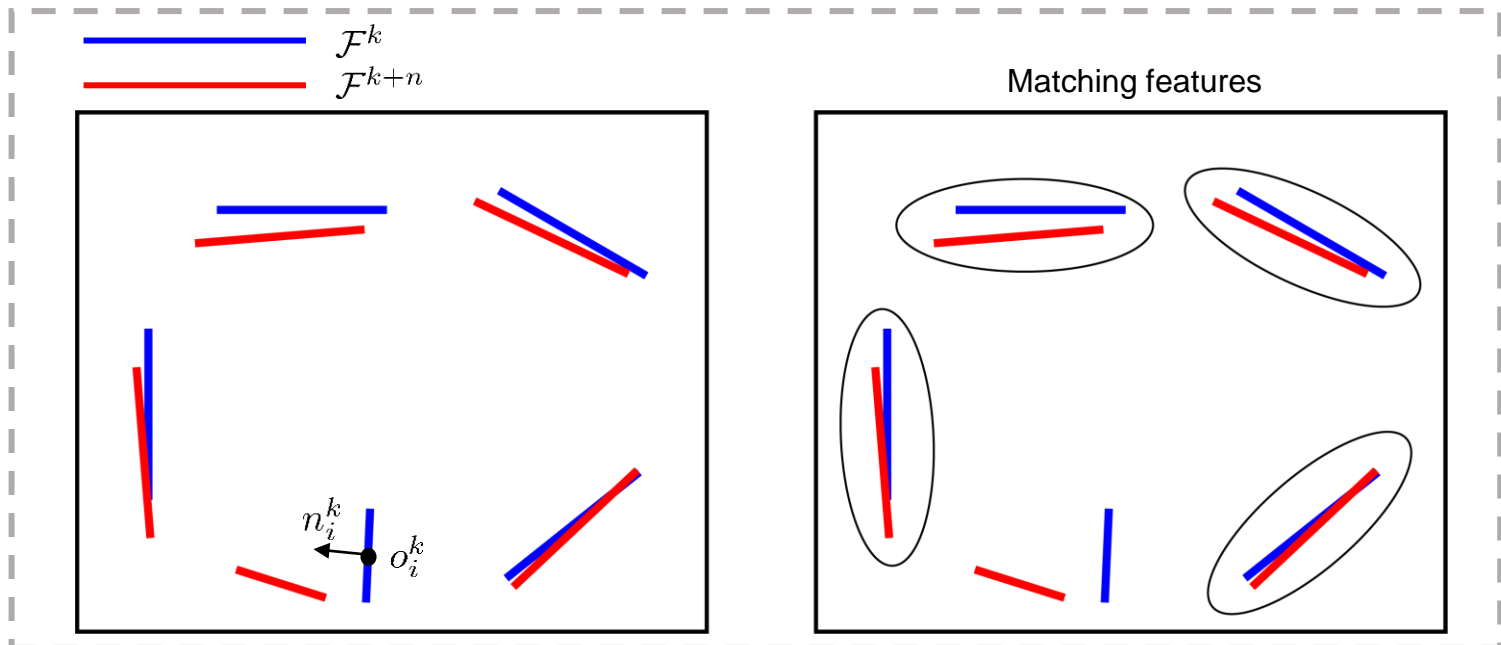
\mathcal{F}^k : k 번째 input pointcloud \mathcal{P}_k 에서 추출한 feature (F_i^k) 의 집합 $\rightarrow F_i^k \in \mathcal{F}^k$

o_i^k : F_i^k 의 중심 좌표

n_i^k : F_i^k 의 normal vector (feature 가 line 경우 direction vector)

If $\|o_i^k - o_j^{k+n}\| < \epsilon_{df} \ \&\& \ n_i^k \cdot n_j^{k+n} > \epsilon_{nf}$ then
 $F_{(k,i)}, F_{(k+n,j)} \rightarrow$ Matching features
 end

ϵ_{df}	0.5 m
ϵ_{nf}	0.85



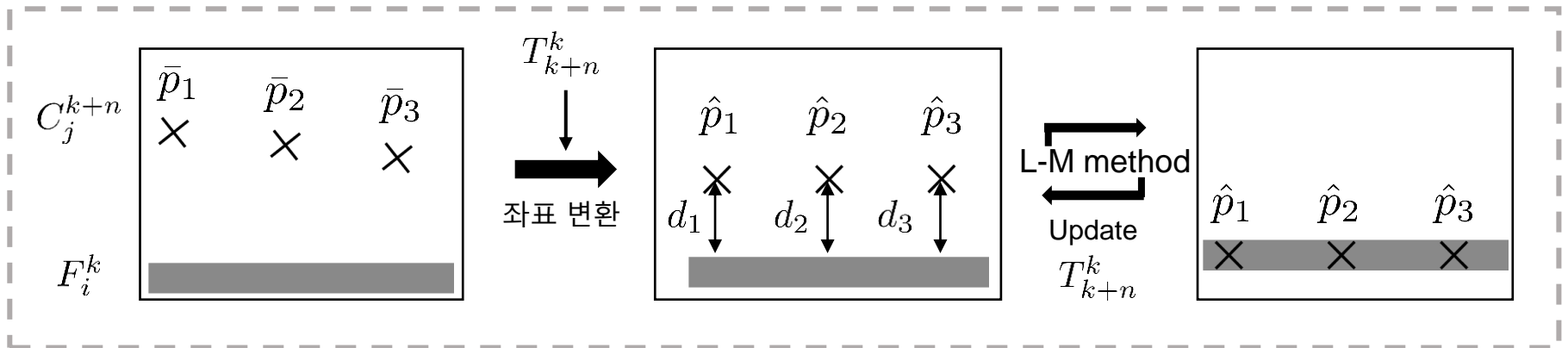
LiDAR Odometry computation

Existing Odometry algorithm

2. *L-M method를 이용한 T_{k+n}^k 최적화

- T : 6-DOF motion LiDAR, $T = [\tau^T \theta^T]$, $\tau = [t_x \ t_y \ t_z]^T$, $\theta = [\theta_x \ \theta_y \ \theta_z]^T$
- k_{th} feature 와 $(k+n)_{th}$ feature 사이의 6-DOF motion : T_{k+n}^k
- T_{k+n}^k 를 이용하여 C_j^{k+n} 의 좌표 변환 ($\bar{p} \rightarrow \hat{p}$), $C_j^{k+n} : F_j^{k+n}$ 에 대응하는 cluster

$$\bar{p}_m \in C_j^{k+n}, R = e^{\hat{\theta}_{k+n}^k}, t = \tau_{k+n}^k \longrightarrow \hat{p}_m = R\bar{p}_m + t$$
- Matching feature 와의 거리 $f(\bar{p}_m, T_k) = d$, for each m, stack f $\rightarrow \mathbf{f}(T_{k+n}^k) = \mathbf{d}$
- $T^* = \arg \min_T F(T)$, $F(x) = \frac{1}{2} \| \mathbf{f}(T) \|^2 \longrightarrow T \leftarrow T - (J^T J + \lambda \text{diag}(J^T J))^{-1} J^T \mathbf{d}$



LiDAR Odometry computation

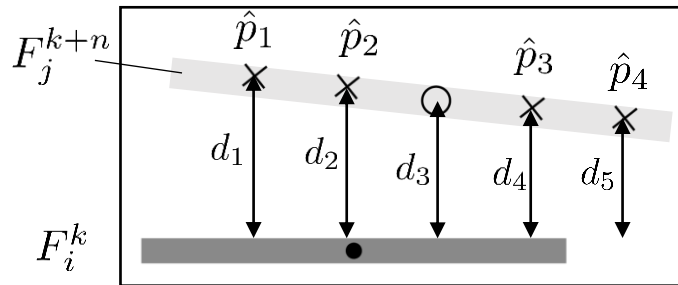
• Modified algorithm

2. *L-M method를 이용한 T_{k+n}^k 최적화 중

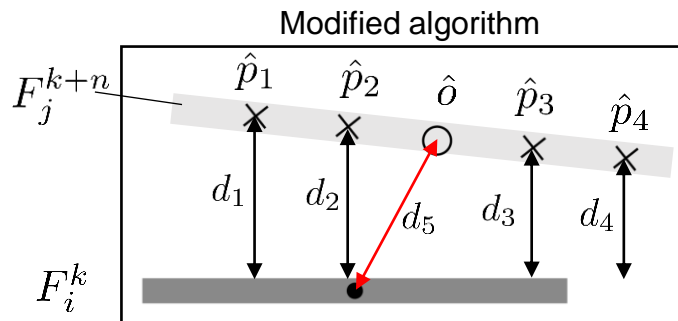
*APPENDIX 참조

- Matching feature 와의 거리 $f(\bar{p}_m, T_k) = d$, for each m, stack f $\rightarrow f(T_{k+n}^k) = d$
- Cost 감소를 위해 C_j^{k+n} 대신 F_j^{k+n} 위의 좌표 5개 사용
 \rightarrow Feature 수가 적을 시 정확한 Odometry를 찾을 수 없는 문제 발생 ... ①
- Matching feature의 중심간 거리 함수 추가 ... ②

①



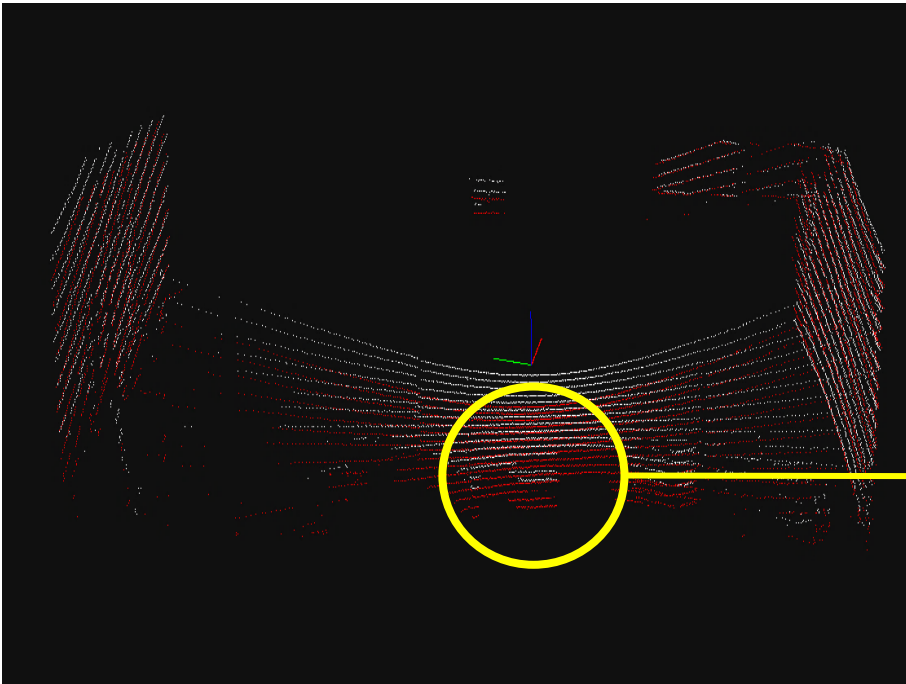
②



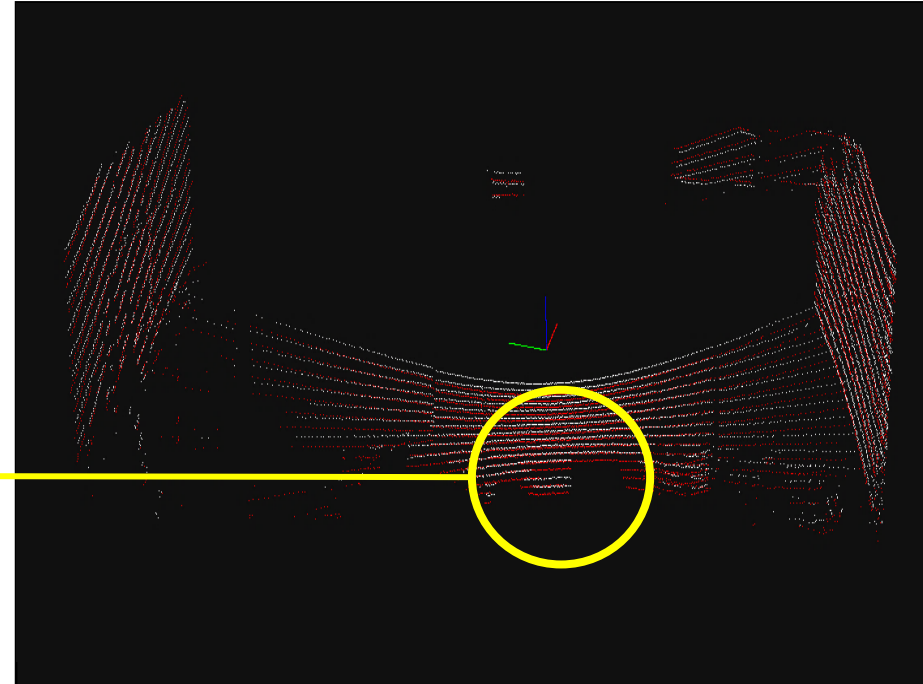
LiDAR Odometry computation

- Result

중심간의 거리 함수 추가 전



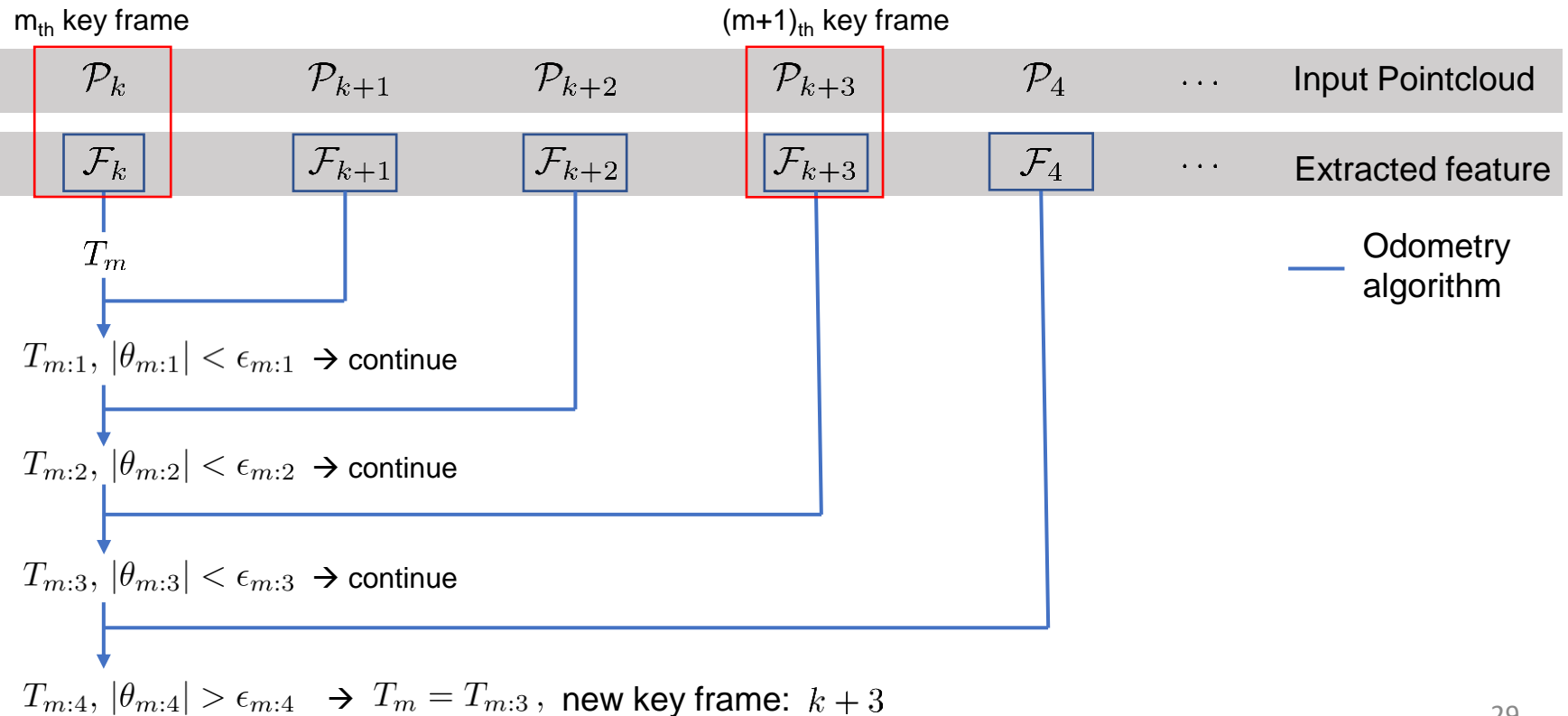
중심간의 거리 함수 추가 후



LiDAR Odometry computation

• Key frame selection

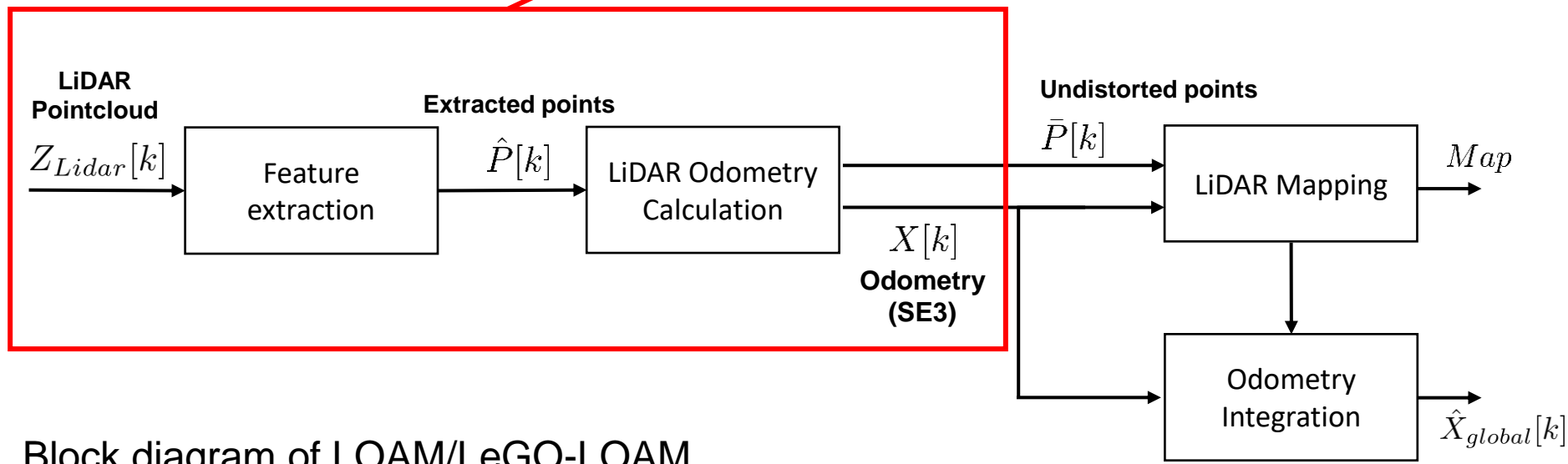
- Odometry 계산 시 $|\theta_k|$ 가 증가할수록 drift 증가 ($T_k = [\tau_k^T \ \theta_k^T]$)
- 초기값 $T_m = [0 \ 0 \ 0 \ 0 \ 0 \ 0]^T$ 으로 시작, Odometry & initial guess update with subframes
- $|\theta_m| > \epsilon_f \rightarrow$ key frame 변경, $\epsilon_f = 0.06$



Odometry Result

- **Experimental process**

- Velodyne VLP-16 datasets in indoor environments 사용하여 Odometry 계산
- General한 성능 지표로 쓰이는 LiDAR SLAM 알고리즘과의 비교 (거리 오차)
LOAM/LeGO-LOAM의 **frame-frame odometry algorithm**과 비교

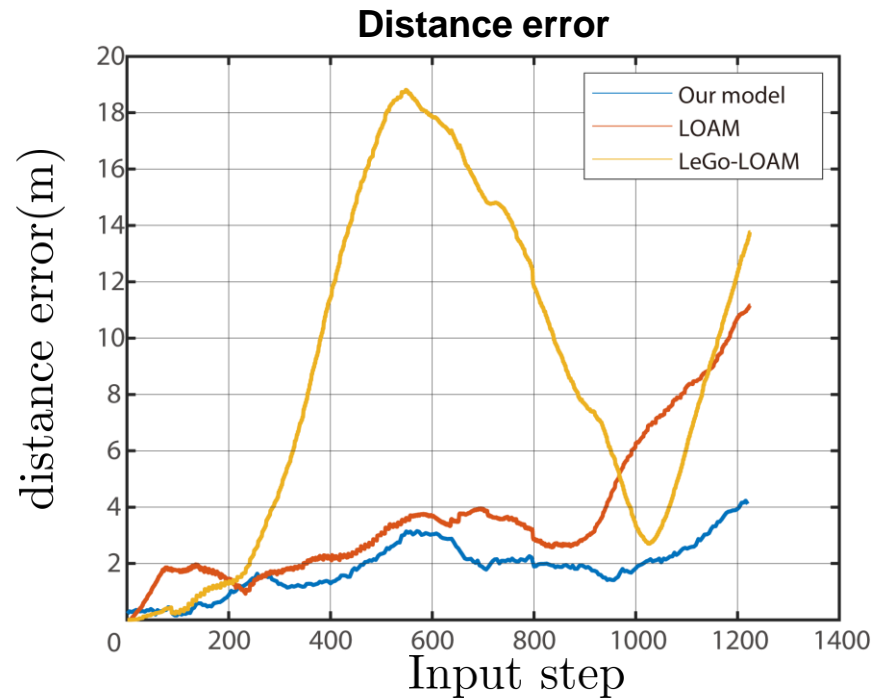
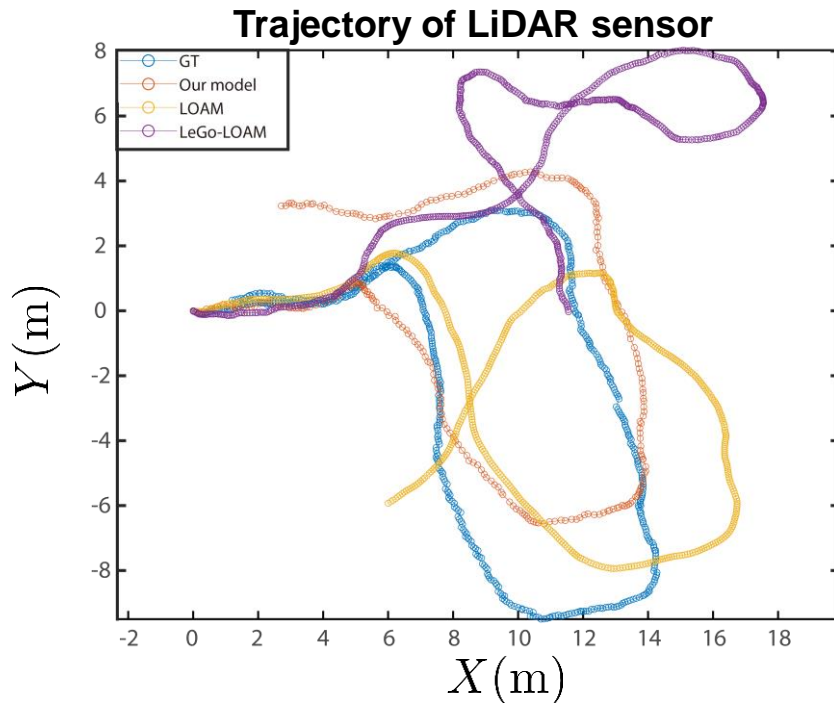


Block diagram of LOAM/LeGO-LOAM

Odometry Result

- Velodyne VLP-16 datasets in indoor environments 사용 결과
 - vs LOAM/LeGO-LOAM의 frame-frame odometry algorithm

Ground truth와의 거리 오차 비교



→ Our model shows **higher accuracy**

Conclusion

- **Summary**

- 실내 환경에 적합한 3D LiDAR odometry algorithm 을 선정하여 구현
- Velodyne VLP-16 datasets in indoor environments 사용하여 성능평가
- LOAM/LeGO-LOAM의 frame-frame odometry와 비교 (거리 오차)
정확도가 높은 Odometry 알고리즘 구현에 성공

- **Future works**

- Odometry algorithm의 속도 향상
- loop closer, back-end 추가
- Real-time 3D LiDAR Odometry algorithm 구현

Appendix

Integral Image to Covariance Matrix

For a given rectangular region R (본 연구에서 adaptive window, $R \subset \mathcal{D}$)

$$C_R = \frac{1}{n} \sum_{k=1}^n (f_k - \mu)(f_k - \mu)^T \quad f_k : \text{Features (본 연구에서 x, y, z 좌표)} \quad D : \text{Depth image}$$

$$C_R(i, j) = \frac{1}{n} \sum_{k=1}^n (f_k(i) - \mu(i))(f_k(j) - \mu(j)) = \frac{1}{n} \left[\sum_{k=1}^n f_k(i)f_k(j) - \frac{1}{n} \sum_{k=1}^n f_k(i) \sum_{k=1}^n f_k(j) \right]$$

➡
$$C_R = \begin{bmatrix} c_{xx} & c_{xy} & c_{xz} \\ c_{xy} & c_{yy} & c_{yz} \\ c_{xz} & c_{yz} & c_{zz} \end{bmatrix} - \begin{bmatrix} c_x \\ c_y \\ c_z \end{bmatrix} \begin{bmatrix} c_x \\ c_y \\ c_z \end{bmatrix}^T$$

10	15	7	9	5	6
9	32	65	45	12	7
7	24	66	65	41	34
9	11	70	89	44	37
32	78	91	78	48	65
64	12	89	58	65	45

Original image of x
 $89+44+37+78+48$
 $+65+58+65+45 = 529$

10	25	32	41	46	52
19	66	138	192	209	222
26	97	235	354	412	459
35	117	325	533	635	719
67	227	526	812	962	1111
131	303	691	1035	1250	1444

Integral image of x
 $1444-459-691+235 = 529$

Faster computation

1. Depth image와 같은 크기의 integral image 생성
2. Point의 좌표 (x, y, z) 를 바탕으로 9개의 integral image 생성

$$\mathcal{I}_x, \mathcal{I}_y, \mathcal{I}_z, \mathcal{I}_{xx}, \mathcal{I}_{xy}, \mathcal{I}_{xz}, \mathcal{I}_{yy}, \mathcal{I}_{yz}, \mathcal{I}_{zz}$$

3. Covariance matrix 계산

$$C_R = \begin{bmatrix} c_{xx} & c_{xy} & c_{xz} \\ c_{xy} & c_{yy} & c_{yz} \\ c_{xz} & c_{yz} & c_{zz} \end{bmatrix} - \begin{bmatrix} c_x \\ c_y \\ c_z \end{bmatrix} \begin{bmatrix} c_x \\ c_y \\ c_z \end{bmatrix}^T$$

c_k : 관심 영역 내 k의 평균
 ex) $c_x = 529/9 \approx 59$

L-M method

Problem: for given a set of m empirical pairs (x_i, y_i)

$$\hat{\beta} \in \operatorname{argmin}_{\beta} S(\beta) \equiv \operatorname{argmin}_{\beta} \sum_{i=1}^m [y_i - f(x_i, \beta)]^2$$

$\hat{\beta} : T$ (LiDAR 6-DOF motion)

x_i : Odometry 계산에 쓰이는 point p

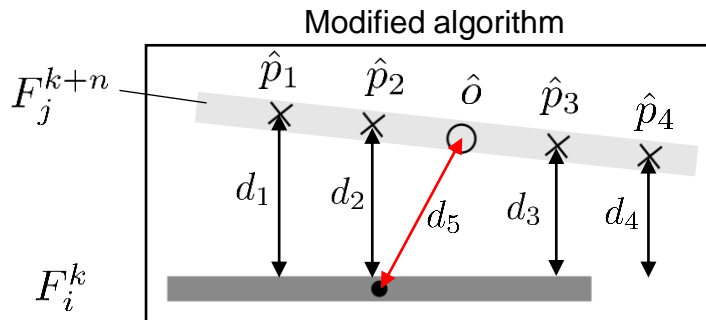
$y_i = 0$ (두 feature 간 거리가 0이 되는 T 를 찾는게 목적)

$f(x_i, \beta)$: Given point x_i 와 matching feature와의 거리

$$[\mathbf{J}^T \mathbf{J} + \lambda \operatorname{diag}(\mathbf{J}^T \mathbf{J})] \delta = \mathbf{J}^T [\mathbf{y} - \mathbf{f}(\beta)] \quad \longrightarrow \quad T \leftarrow T - (J^T J + \lambda \operatorname{diag}(J^T J))^{-1} J^T d$$

$$\blacksquare \quad T^* = \operatorname{argmin}_T F(T), \quad F(x) = \frac{1}{2} \| \mathbf{f}(T) \|^2 \quad \rightarrow \quad T \leftarrow T - (J^T J + \lambda \operatorname{diag}(J^T J))^{-1} J^T d$$

한 세트의 matching feature 간의 자코비안 계산 $J_i = \frac{\partial f_l(x_i, T)}{\partial T}$



$f_l (l = 1, 2, 3, 4) : p_j$ 와 F_i^k 의 거리

$f_l (l = 5) : F_j^{k+n}$ 의 중심과 F_i^k 의 중심 사이 거리

Publication

American physical society March Meeting 2018 Los Angeles

Elastic hoops jumping on water: A model system for fishing spiders

Han Bi Jung, Yunsuk Jeung, Ho-Young Kim

ASME-JSME-KSME Joint Fluids Engineering Conference 2019

Liquid jet impingement on a rotating substrate

Yunsuk Jeung, Jae Hong Lee, Ho-Young Kim

제 11회 한국유체공학학술대회

회전하는 고체표면과 액체 젯(jet)의 충돌 현상 연구

정윤석, 이재홍, 김호영

Patent

액막의 종류와 Wafer 소수성 정도에 따른 Fingering 제어 방법 (1020200105261)

액적 포집 시스템 및 제어 방법 (1020190155451)

기판 처리 방법 및 기판 처리 장치 (1020190056565)

해외특허 출원 : 일본 (2020-084408), 중국 (202010406079.3), 미국 (15/931,911)