Loam (Lidar odometry and mapping)

↳ 라이다를 이용해 odometry 와 mapping 분야 함 (2개 알고리즘)

⇒ odometry → /mapping → odometry → mapping Sweep ⇒ Lidar가 한바퀴 도는것

↳ Sweep간 relative pose 측정 ↳ pose 보정 (에러율 ↑)

$\chi^{L}_{(k,i)}$ = Lidar coordinate system의 k번째 sweep 에서 측정된 점들중 i번째 점의 x,y,z 좌표

$\chi^{W}_{(k,i)}$ = Word Coordinate System

+ 기본문제 ⇒ lidar 가 움직이면 error 크게 생김
drift ⇒ 로봇위치 오차 생긴것

* Maping ⇒ Global에 등록함 (ICP와 다른점임) → Contribution = Mapping

↳ feature 가 많이 뽑아서 error 줄이는 것이다.

Feature Extraction

⇒ 2가지 feature 정의

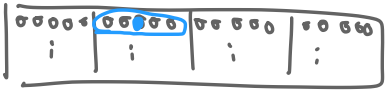
↳ 휘어진 정도

$$c = \frac{1}{|S| \, \|X^{L}_{(k,i)}\|} \left\| \sum_{j \in S, j \neq i} (X^{L}_{(k,i)} - X^{L}_{(k,j)}) \right\|$$

(curvature) 곡률

c 크면 edge 작으면 planar

⇒ 1개의 Scan line 4개의 Sub region으로 나눔



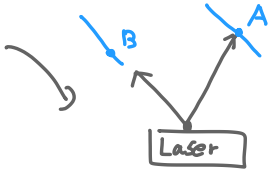
↳ 실제 코드 i-5, i-4, i-3, .. i+5 이런식으로 되어있다

추출되지 말아야할 feature

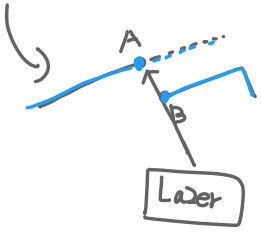
=> 이전에 추출되었고, 너무 가까운것

=> Laser 와 너무 평행한 평면

=> 다른 물체에 가려진 feature

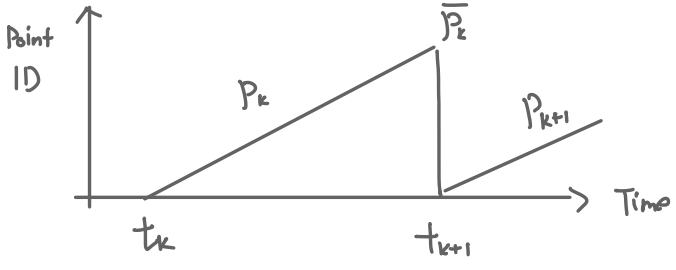


=> B 같은 경우는 추출되면 X



=> A 는 Edge 아니지만 Edge 처럼 보이기 때문에 추출 되지 말아야 한다.

Finding Feature Point Correspondence



$\mathcal{E}_{k+1}$: Edge feature

$\mathcal{H}^{+}_{k+1}$: Edge feature

$\parallel$

P_{k+1} ?

=> $t_k \sim t_{k+1}$ 동안 얻은 measurement 값 P_k를 transformation 가지고 t_{k+1}로 projection 한 point cloud 를 $\bar{P}_k$ 라고 한다

=> t_{k+1} 이후부터 다시 P_{k+1} 를 모으기 시작 t_{k+2}에서 $\bar{P}_{k+1}$ 등록

=> 등록 시점만 모았던 등록 시점으로 리셋

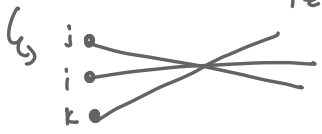
이후 $\bar{P}_k$ $\bar{P}_{k+1}$ 를 이용해서 KD tree를 만든다

↳ 이진 트리를 다차원으로 확장한 트리 => 

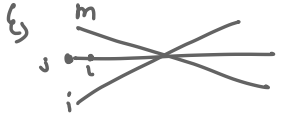
↳ 그이후 가까운 feature point 배응계 검색

⇒ 즉, Edge, Planar points 의 correspondence 찾아야 함

Edge ⇒ 2개의 point 이며 i 말고 j, L 를 이전에 scan line에서 2번이면에 추출한 가까운 edge feature



planar ⇒ 3개의 point m, j, L 정함



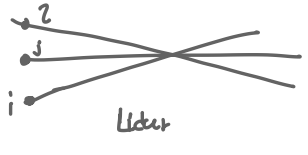
Sparse 한 경우는 유클리디안 보다는 Cos Similarity 나 외적으로 벡터 비교 더 쉽고 확실.

■ Correspondence 의 distance F 정의

⇒ 정의된 dist F 를 최소화하도록 최적화 해 lidar motion 추정

⇒ feature는 동일한 위치에서 안변하니 Euclidean dist 사용할까

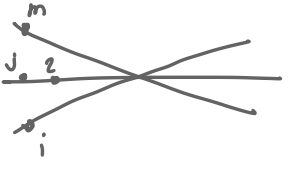
$$d_\varepsilon = \frac{\left|(\bar{X}^L_{(k+1,i)} - \bar{X}^L_{(k,j)}) \times (\bar{X}^L_{(k+1,i)} - \bar{X}^L_{(k,l)})\right|}{\left|\bar{X}^L_{(k,j)} - \bar{X}^L_{(k,l)}\right|}$$



↳ Cross product 는 sin θ 로 θ 가 0 이면 일직하고 d 값은 0 이된다

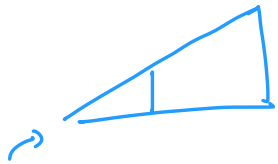
↳ |AB| sinθ

$$d_H = \frac{\left|(\bar{X}^L_{(k+1,i)} - \bar{X}^L_{(k,j)}) \times ((\bar{X}^L_{(k,j)} - \bar{X}^L_{(k,l)}) \times (\bar{X}^L_{(k,j)} - \bar{X}^L_{(k,m)}))\right|}{\left|(\bar{X}^L_{(k,j)} - \bar{X}^L_{(k,l)}) \times (\bar{X}^L_{(k,j)} - \bar{X}^L_{(k,m)})\right|}$$



Motion Estimation

=> Lidar Motion linear velocity 가정



기울기 같다

$$T^{L}_{(k+1,i)} = \frac{t_i - t_{k+1}}{t - t_{k+1}} T^{L}_{k+1}$$

~ ~ => Edge dist, planar dist, -> total dist

=> 목적 total dist minimize 한 transformation matrix T를 찾는것

↳ LM (Levenberg Marquart) 등등 사

Lidar Mapping

=> 연산량이 많아 Low frequency (1Hz, 2Hz) 로 실행

~> 파라미터링 Error 있는데 이를 최소화 하기위한 최적화

feature 는 odo 보다 비교적 많이 뽑음

KD-tree 이용해 correspondence 계산

Odometry 와 동일하게 LM 최적화로 pose 보강