차 례

이동지능 소프트웨어 개발 학습모듈의 개요

학습모듈의 목표

목적지까지의 로봇 이동을 위하여 환경 지도의 작성, 로봇 위치의 매핑, 그리고 경로를 계획하고 구현할 수 있다.

선수학습

C 프로그래밍, 로봇 소프트웨어 플랫폼(ROS) 프로그래밍, 로봇 공학, 모션 컨트롤러

학습모듈의 내용 체계

학습	학습 내용	NCS 능력단위 요소	
		코드번호	요소 명칭
1. 지도 작성하기	1-1. 지도 작성	1903080319_16v2.1	지도 작성하기
2. 로봇 위치 매핑하기	2-1. 로봇 위치 매핑	1903080319_16v2.2	로봇 위치 매핑하기
3. 이동 경로 계획 구현하기	3-1. 이동 경로 계획 구현	1903080319_16v2.3	이동 경로 계획 구현하기

핵심 용어

지능로봇, 소프트웨어, 알고리즘, 인공지능, 작업지능, 이동지능, 인지지능, HRI지능

학습 1 지도 작성하기

학습 2	로봇 위치 매핑하기
학습 3	이동 경로 계획 구현하기

1-1. 지도 작성

학습 목표

- 작업 환경을 표현하는 여러 지도의 특징에 대해 알고 있다.
- 각 지도를 적합한 데이터 구조로 프로그래밍 할 수 있다.
- 각 지도간의 위치가 매핑되는 데이터 구조로 프로그래밍 할 수 있다.

필요 지식 /

① 작업 환경을 표현하는 여러 지도의 특징

로봇이 작업 공간에서 보행 또는 이동하면서 요구 서비스를 제공하기 위해서는 작업 환경에 대한 2차원적 또는 3차원적 공간 정보를 알아야 하고, 로봇과 작업 대상체, 목표 지점의 상대적·절대적 위치를 알아야 한다. 로봇이 현재 지점에서 목표 지점까지 자율적으로 이동하기 위해서는 다음의 과정이 필요하다.

- 로봇이 위치해 있는 환경의 지도 데이터 확보(지도가 없을 때는 지도 작성(mapping))
- 출발 지점부터 목표 지점까지 적절한 경로 계획(path planning)
- 생성된 경로를 추종하기 위한 족관절(보행로봇)/바퀴(모바일로봇)의 궤적 생성(trajectory planning) 및 추종 제어
- 센서 계측을 통한 로봇의 현재 위치 인식(localization)

만약 로봇이 이동해야 할 실내외 환경의 지도 데이터를 가지고 있지 않다면 로봇이 이동하면서 현재 위치를 인식하는 동시에 지도를 만드는 기술인 SLAM(simultaneous localization and mapping)기능이 필수적이다.

지도 작성(map building)은 공간 표현(spatial representation)이라고도 불리며 로봇이 계측한 센서 정보로부터 주변 사물의 기하학적 형상 정보 또는 특징 관련 정보를 추출하여 로봇이 해석할 수 있는 데이터로 변환하는 기술을 의미한다.

DUDEK and JENKIN(2000)에 의하면 로봇의 작업 환경을 표현하는 지도의 종류에는 크게 점유격자지도(occupancy grid map), 특징지도(feature map), 위상지도(topological map)가 있다.

1. 점유격자지도

점유격자지도(occupancy grid map)는 공간 분할(spatial decomposition)이라고도 하며, 로봇의 환경 공간을 일정한 간격의 2차원 격자(grid)로 나누고 로봇이 주행 가능한 빈 공간(free space)과 장애물이나 구조물, 또는 물체가 차지하고 있는 점유 공간(occupied space), 알 수 없는 공간(unknown space)으로 구분하여 표현하는 지도이다. 이 방법은 로봇의 이동 경로가 정확하다는 가정 하에 잡음이 섞인 거리 측정 센서 데이터를 이용하여 환경 정보를 작성하는 기술로서 직관적이고 임의의 공간 구조를 표현하는 것이 가능하므로, 1980년대 ELFES and MORAVEC(1985)에 의해 제안 된 이후 로봇용 지도로 폭넓게 사용되고 있다.

점유격자지도의 장점은 구축하고 관리하기가 쉽다는 것이고, 단점은 넓은 공간에서는 격자의 수가 크게 증가하므로 지도 데이터의 저장량도 비례해서 커진다는 것이다.

2. 특징지도

특징지도(feature map)는 로봇이 센서로 계측 가능한 환경 공간의 구조물이나 물체를 선(모서리)이나, 면(벽면), 다각형(형광등), 원(램프)과 같은 특징들로 표현한 지도로서 CHATILA and LAUMOND(1985)에 의해 제안되었다.

특징지도의 장점은 선이나 면, 원은 몇 개의 수식 파라미터로 표현할 수 있으므로 지도 데이터의 용량을 크게 줄일 수 있으며, 크기에 상관없이 작업 환경을 측정 정보(metric information)와 함께 효과적으로 표현할 수 있고, 이산화(discretization) 문제도 발생하지 않는 것이다.

특징지도의 단점은 기하학적으로 표현하기 어려운 비정형적(unstructured) 환경을 특징지도로 표현하는 것에는 적합하지 않으며, 특징점(feature point)이 증가할수록 계산량이 비례하여 증가하며, 환경 로봇의 위치나 시점에 따라 동일한 구조물이나 사물도 기울기나 모양이 다르게 측정되므로 실제 모습으로 보정하는 과정에서 오차나 오류가 발생할 수 있는 점이다.

3. 위상지도

위상지도(topological map)는 KUIPER and BYUN(1991)에 의해 제안된 것으로 여행용 지도와 같이 환경 공간 내의 장소들을 선으로 연결한 그래프 모양으로 나타냄으로써 영역 혹은 물체 간의 관계성을 효율적으로 표현한다. 그래프에서 각 에지(edge)는 로봇이 지나갈 수 있는 여유 공간을 의미하며, 노드는 별개의 상황, 장소, 또는 랜드마크(landmark)에 해당한다. 개별적 연결 가능한 노드(node)의 조합이 로봇이 이동 가능한 경로가 된다.

위상지도의 장점은 가정이나 빌딩 같이 다수의 공간과 영역으로 구성된 건물을 표현하기에 적합하고, 데이터가 간결하며(compact), 효율적인 경로 계획을 가능하게 하고, 공간의 복잡성이 낮다는 것이다. 단점은 측정 정보가 적고, 정확한 위치 인식이 어려우며, 장소 인식이 종종 모호하다는 사실이다.

2 각 지도를 적합한 데이터 구조로 프로그래밍

세 가지 지도를 적합한 데이터 구조로 프로그래밍하기 위해서는 각 지도를 제작하는 이론적 원리와 알고리즘을 이해할 필요가 있다.

1. 점유격자지도

점유격자지도에서 m은 환경지도를 의미하고, N개의 셀($m_1, m_2, \cdots, m_N$)로 구성된다고 하자. 각 셀은 이진수의 경우 0 또는 1, 혹은 비트열로 표현되는 경우 0에서 1사이의 확률값 $p(m_i)$을 가지는데, 0은 빈 공간, 1은 장애물이나 어떤 물체가 그 셀에 있음을 의미한다. 확률값을 쓰는 이유는 환경을 인식하는 센서의 불확실성 때문이며, 센서값은 시간에 따라 현재시간 t까지 누적되고 $\boldsymbol{z}_{1:t}$로 표현한다. 그리고 현재시간까지의 로봇의 위치 및 방향각은 상태변수 $x_{1:t}$로 표시한다. 지도 제작은 관찰된 로봇의 상태값에 따른 센서 정보를 이용하여 각 격자 셀의 점유상태를 확률적으로 표현하는 것으로 이해할 수 있으므로 다음 식으로 표현 가능하다.

$$p(m|x_{1:t}, \boldsymbol{z}_{1:t}) = \Pi_{i=1}^{N} p(m_i|x_{1:t}, \boldsymbol{z}_{1:t}) \qquad \text{식 (1)}$$

식에서 전체 격자 셀의 점유상태를 계산하는 것이 너무 많은 계산량과 메모리를 차지하므로 각 셀의 점유상태가 서로 독립이라는 가정 하에 전체 셀의 점유상태를 각 셀의 점유상태 확률의 곱으로 근사화했다.

점유격자지도를 생성하기 위해서 다음과 같이 조건부 확률의 성질을 이용한 베이즈 규칙(Bayes rule)을 많이 사용한다.

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} = \frac{p(y|x)p(x)}{\sum_i p(y|x_i)p(x_i)} \qquad \text{식 (2)}$$

즉, 확률변수 x는 관찰 가능하고 y는 조건 없이 관찰하기 어렵다고 했을 때, y를 조건으로 하는 확률값 $p(x|y)$를 직접 계산할 수 없으므로 계산 가능한 $p(y|x)$로부터 계산한다.

두 개의 확률변수가 조건으로 주어졌다면 식 (2)는 다음과 같이 수정된다.

$$p(x|y,z) = \frac{p(y|x,z)p(x|z)}{p(y|z)} \qquad \text{식 (3)}$$

이제 확률변수 $x_{1:t}$와 $\boldsymbol{z}_{1:t}$를 조건으로 두고 식 (3)에서 $y = z_t,\ z = \left[x_{1:t}\ \boldsymbol{z}_{1:t-1}\right]$로 대입하고 식 (1)에 적용하면 다음과 같은 관계식을 얻을 수 있다.

$$p(m_i|x_{1:t}, \boldsymbol{z}_{1:t}) = p(m_i|x_{1:t}, \boldsymbol{z}_{1:t-1}, z_t) = \frac{p(z_t|m_i, x_{1:t}, \boldsymbol{z}_{1:t-1})p(m_i|x_{1:t}, \boldsymbol{z}_{1:t-1})}{p(z_t|x_{1:t}, \boldsymbol{z}_{1:t-1})} \qquad \text{식}$$

(4)

선형시스템일 경우 가장 최근의 측정값은 최근 상태변수의 선형함수로 표현될 수 있다. 그러므로 식 (4)에서 $p(z_t|m_i,x_{1:t},\boldsymbol{z}_{1:t-1})$가 $p(z_t|m_i,x_t)$로 간략화되고 여기에 다음과 같이 베이즈 규칙을 적용한다. 상태변수 x_t는 m_i와 무관하므로 $p(m_i|x_t)$를 $p(m_i)$로 간략화하면 식 (4)는 다음과 같이 단순해진다.

$$p(z_t|m_i,x_t)=\frac{p(m_i|x_t,z_t)p(z_t|x_t)}{p(m_i|x_t)}=\frac{p(m_i|x_t,z_t)p(z_t|x_t)}{p(m_i)} \quad \text{식 (5)}$$

식 (5)를 식 (4)에 대입하면 식 (6)이 도출된다. 식 (6)은 현재까지 상태변수 $x_{1:t}$와 직전 샘플시간 전까지의 센서값 $\boldsymbol{z}_{1:t-1}$으로 계산한 각 셀의 점유확률에 현재까지의 센서값 $\boldsymbol{z}_{1:t}$를 이용하여 계산한 확률식을 곱함으로써 재귀적(recursive) 형태로 업데이트할 수 있음을 보인다.

$$p(m_i|\boldsymbol{x}_{1:t},\boldsymbol{z}_{1:t})=\frac{p(m_i|x_t,z_t)p(z_t|x_t)}{p(m_i)}\frac{p(m_i|\boldsymbol{x}_{1:t-1},\boldsymbol{z}_{1:t-1})}{p(z_t|\boldsymbol{x}_{1:t},\boldsymbol{z}_{1:t-1})} \quad \text{식 (6)}$$

식 (7)과 같은 로그오즈(log odds)를 식 (6)의 양변에 취하면 식 (8)과 같이 간단한 업데이트 식으로 변환할 수 있다.

$$l_o(x)=\log\left(\frac{p(x)}{1-p(x)}\right) \quad \text{식 (7)}$$

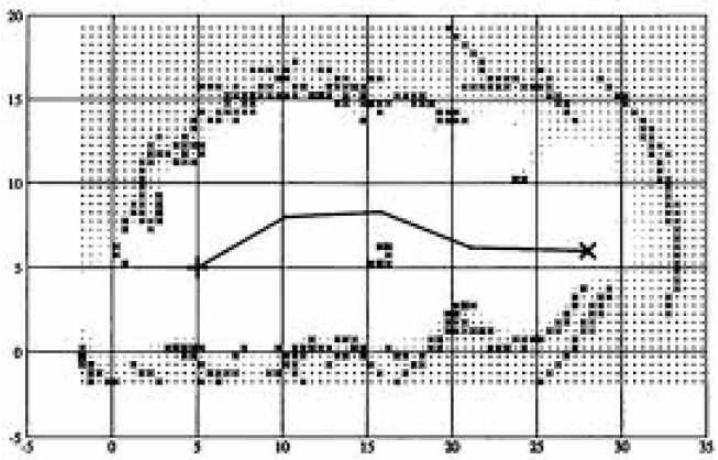
$$l_o(m_i|x_{1:t},\boldsymbol{z}_{1:t})=l_o(m_i|x_t,z_t)+l_o(m_i|x_{1:t-1},\boldsymbol{z}_{1:t-1}) \quad \text{식 (8)}$$

식 (8)을 통해 i번째 셀의 점유확률의 로그오즈 값을 계산하면 식 (7)을 이용하여 확률값을 계산할 수 있다.

식 (6)과 (8)에서 $p(m_i|x_t,z_t)$는 현재 시점의 격자 셀의 점유확률로서 로봇에 사용되는 센서의 특성에 따라 확률값이 다르게 계산된다. 거리측정센서 사용 시 현재 측정된 거리가 z_t이고 격자의 해상도(resolution)가 r, 현재 로봇의 위치에서 격자 셀까지의 거리를 $dist(x_t,m_i)$라고 할 때 점유확률을 계산하면 다음과 같다. 식에서 p_{prior}는 이전 확률값, p_{occ}는 점유된 공간일 확률값, p_{free}는 빈공간일 확률값을 나타낸다.

$$p(m_i|x_t,\boldsymbol{z}_t)=\begin{cases}p_{prior} & when\ \ z_t=z_{\max}\ \text{or}\ m_i is\ \ not\ \ covered\\ p_{occ} & when\ \ |z_t-dist(x_t,m_i)|<r/2\\ p_{free} & when\ z_t\geq dist(x_t,m_i)\end{cases} \quad \text{식 (9)}$$

[그림 1-1]은 MORAVEC의 점유격자지도 제작 예를 보인다.



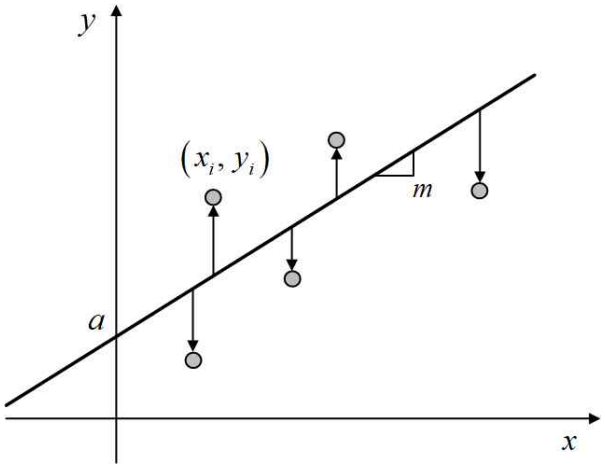
출처: 정완균・도낙주・이수용 외(2012).『실험로보틱스 Ⅱ, 이동로봇』. 한국로봇학회・제어로봇시스템학회・한국로봇산업진흥원. p.130.
[그림 1-1] MORAVEC의 점유격자지도

2. 특징지도

특징지도에는 실내 환경을 이루는 벽과 바닥을 직선의 조합으로 표현하기에 적당한 선지도(line map)와 다양한 도형을 간편하게 표현하기 위해 벡터 기반의 특징점을 모아 폐회로(closed loop)로 나타내는 방법 등이 사용되고 있다.

특징지도의 기본 원리를 이해하기 위해 거리센서로부터 벽이나 바닥을 인식한 정보가 주어지고 이를 하나의 직선으로 표현할 때 직선의 성분(기울기와 절편)을 찾아내는 원리에 대해 알아보자.

일반적으로 직선의 식은 기울기를 m, y절편을 a라고 하면 $y=mx+a$로 나타낼 수 있다. 이 두 값을 N개의 데이터로부터 계산하는 것은 모든 데이터와 직선까지의 y축 상의 거리의 합 $R(m,a)$를 최소화하는 값을 찾는 문제가 된다.



[그림 1-2] 수직 최소자승법으로 점들을 근사화하는 직선

$$R(m,a)=\sum_{i=1}^{N}(y_i-y)^2=\sum_{i}^{N}(y_i-mx_i-a)^2 \qquad \text{식 (10)}$$

$R(m,a)$를 최소화하는 m과 a는 극소값 계산을 위해 식 (10)으로부터 $\frac{\partial R}{\partial m}=0, \frac{\partial R}{\partial a}=0$을 계산하면 되고, 이를 행렬식으로 표현하면 다음과 같다.

$$\begin{bmatrix}\sum_i x_i^2 & \sum_i x_i \\ \sum_i x_i & N\end{bmatrix}\begin{bmatrix}m\\a\end{bmatrix}=\begin{bmatrix}\sum_i x_i y_i\\ \sum_i y_i\end{bmatrix} \qquad \text{식 (11)}$$

이 식에서 미지수 m와 a를 계산하기 위해 역행렬로 표현하면 다음과 같은데, 이런 원리로 직선의 파라미터를 계산하는 방법을 수직 최소자승법(least square method)이라고 한다.

$$\begin{bmatrix}m\\a\end{bmatrix}=\frac{1}{N\left(\sum_i x_i^2-\bar{x}^2\right)}\begin{bmatrix}\sum_i x_i y_i-\bar{x}\bar{y}\\ \bar{y}\sum_i x_i^2-\bar{x}\sum_i x_i y_i\end{bmatrix} \qquad \text{식 (12)}$$

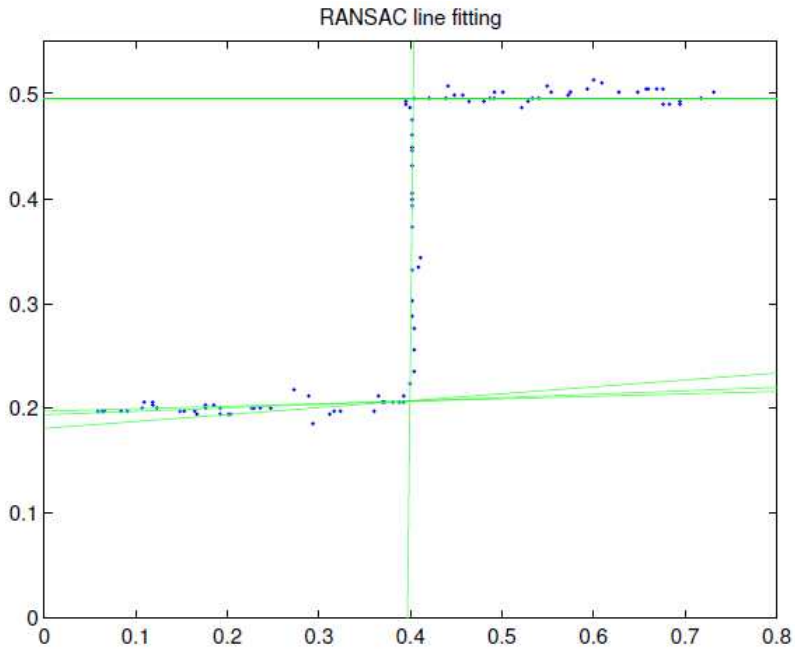
식에서 $\bar{x}$와 $\bar{y}$는 x_i와 y_i의 평균값으로서 계산식은 $\bar{x}=\frac{1}{N}\sum_{i=1}^{N}x_i$, $\bar{y}=\frac{1}{N}\sum_{i=1}^{N}y_i$이다.

만약 n개의 좌표점을 여러 개의 직선으로 표현해야 하는 경우 split-and-merge 알고리즘(PAVLIDIS & HOROWITZ, 1974), RANSAC(RANdom SAmple Consensus) 알고리즘(FISCHLER & BOLLERS, 1981) 등을 이용하면 된다.

이 중에서 많이 사용되고 있는 RANSAC을 설명하기 위해 센서로 얻은 환경 데이터가 N개의 2차원 점이라고 하자.

- 이 중에서 적당히 큰 숫자의 샘플 $(x_1, y_1), \cdots, (x_n, y_n)$을 추출 후 그중 두 점을 무작위로 선택하여 두 점을 지나는 직선 식 $y = f(x) = ax + b$을 구한다. 포물선의 경우 $f(x)$는 2차방정식 $f(x) = ax^2 + bx + c$으로 표현된다.
- 이 점을 제외한 점 (x_i, y_i)와 직선 간의 거리 $r_i = |y_i - f(x_i)|$가 문턱값 d_T 이하이면 해당 $f(x)$의 컨센서스 집합(consensus set)에 포함시킨다.
- 이 집합에 포함된 점의 개수가 n_c 이상이면 현재 $f(x)$를 후보로 저장한다.
- 이 과정을 N번 반복한 후 저장되어 있는 $f(x)$중 컨센서스 집합의 크기가 가장 큰 $f(x)$를 선택한다.

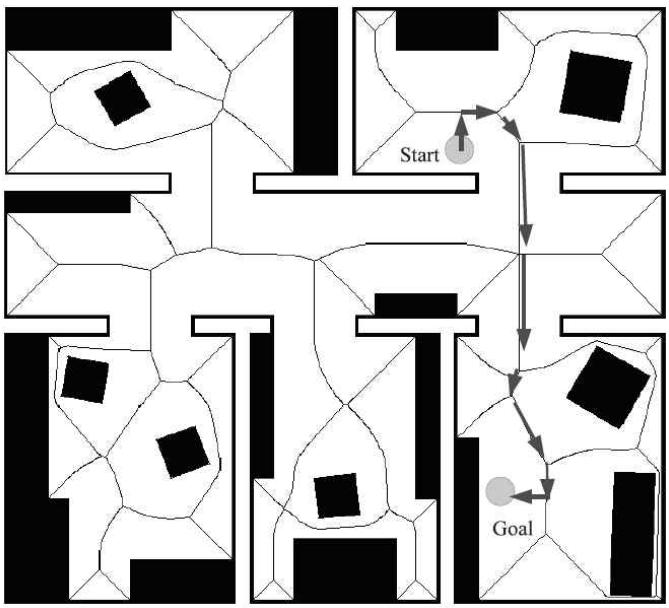
RANSAC을 적용하기 위해서는 컨센서스 집합을 만들 때 사용되는, $f(x)$와의 거리 문턱값 d_T와 집합에 포함된 점의 개수 n_c를 상황에 따라 적적히 설정하는 것이 중요하다.



출처: 정완균・도낙주・이수용 외(2012).『실험로보틱스 Ⅱ, 이동로봇』. 한국로봇학회・제어로봇시스템학회・한국로봇산업진흥원. p.147.
[그림 1-3] RANSAC 알고리즘을 적용한 직선 추출

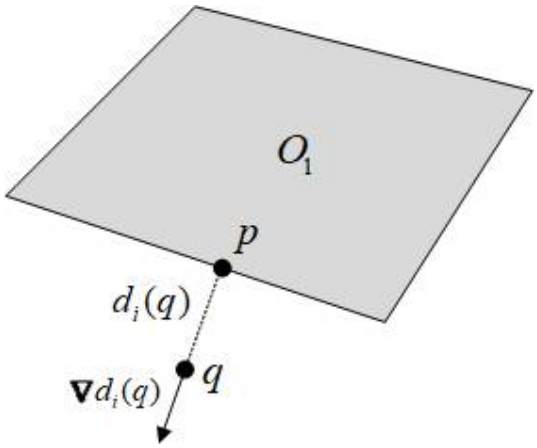
3. 위상지도

위상지도(topological map)는 장소와 장소를 선으로 연결한 그래프 모양의 지도이며, 많은 방법 중 보로노이지도(Voronoi diagram)와 시각지도(visibility map)에 대해 알아보자.



출처: 정완균・도낙주・이수용 외(2012).『실험로보틱스 Ⅱ, 이동로봇』. 한국로봇학회・제어로봇시스템학회・한국로봇산업진흥원. p.151.
[그림 1-4] 보로노이 지도 예

실내 환경은 가정의 경우 안방, 거실, 부엌 등 단순하고 한정된 공간으로 구성되어 있으므로 노드(node)와 에지(edge)로 구성된 그래프 형태로 환경을 표현할 수 있다. 그래프에서 각 노드는 복도 끝, 구석, 문 등 로봇이 위치할 수 있는 빈 공간을 의미하며, 에지는 [그림 1-4]처럼 두 개의 근접한 장애물로부터 거리가 같은 점들을 이어 붙인 것으로서 로봇이 이동 가능한 공간을 의미한다.



[그림 1-5] 장애물과 빈 공간의 한 점의 거리 및 그래디언트 방향

보로노이지도는 빈 공간에서 가까운 장애물과의 거리로 정의되기 때문에 모바일 로봇의 거리센서 데이터가 있는 경우 이를 이용하여 실시간으로 제작할 수 있다. 보로노이지도 제작을 위해서는 다음의 3단계가 필요하다.

- 장애물과의 거리를 측정 후 거리가 멀어지는 방향으로 이동하여 가까운 에지로 이동
- 에지에 다다르면 에지 트레이싱(tracing) 방법을 이용하여 에지를 따라 이동
- 에지를 따라 이동하다가 노드에 도착하면 새로운 노드인지 여부를 판별하여 기록

먼저 [그림 1-5]를 보면서 가까운 에지로 이동하는 원리에 대해 설명하면, 빈 공간의 한 점 q와 장애물 O_1 사이의 가장 가까운 거리는 다음과 같이 정의된다.

$$d_i(q) = \min d(q,p) \qquad \text{식 (13)}$$

그림에서 보듯이 장애물 O_1의 가장자리 중 점 q와 가장 가까이 있는 점이 p라고 하면 로봇은 에지에 도달하기 위해 다음 식의 값만큼 이동해야 한다. 식에서 α는 변화율(step length)을 의미한다.

$$\dot{q} = \alpha \frac{q-p}{d_i(q)} \qquad \text{식 (14)}$$

보로노이지도에서 에지는 두 개의 인접한 장애물 O_1, O_2와 거리가 같은 점의 집합이므로 수식으로 표현하면 다음과 같다.

$$G(q) = d_1(q) - d_2(q) = 0 \qquad \text{식 (15)}$$

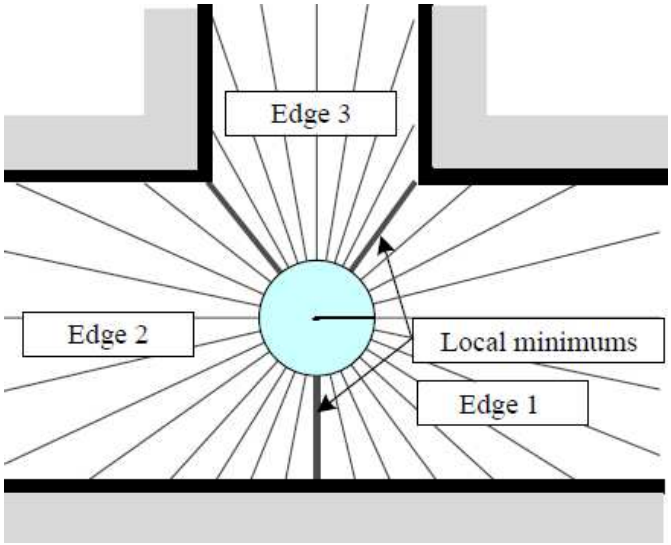
로봇이 에지를 따라가려면 식 (14)는 다음과 같이 수정되어야 한다.

$$\dot{q} = \alpha v + \beta(\nabla G(q))^{\dagger} G(q) \qquad \text{식 (16)}$$

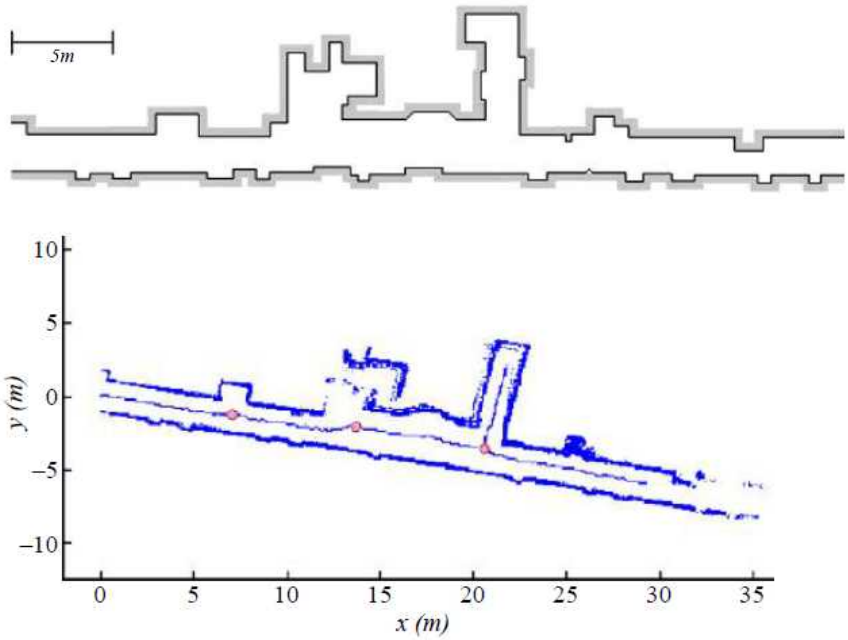
식에서 v는 보로노이지도 에지에 접한 벡터이고, β는 가중치, $(\nabla G(q))^{\dagger}$는 $\nabla G(q)$의 의사역행렬(pseudo-inverse)을 의미한다.
로봇이 보로노이지도 에지를 따라 이동하다면 보면 교차점 노드에 도달하게 되는데, 노드는 다음 수식으로 표현할 수 있다.

$$G(q) = \begin{bmatrix} d_1(q) - d_2(q) \\ d_1(q) - d_3(q) \end{bmatrix} \qquad \text{식 (17)}$$

로봇은 노드를 만났음을 인식하면 새로운 노드인지 여부를 판별한 후 새로운 노드이면 메모리에 기록한다.
거리측정센서(range finder)를 써서 보로노이지도를 만드는 경우 [그림 1-6]과 같이 양쪽 벽과의 거리가 최소가 되는 부분을 에지로 정의하고, 세 개 이상의 에지가 만나는 점을 노드로 정의한다. 이 방법의 문제점은 센서 노이즈에 의해 지역최소점(local minimum)을 잘못 계산하여 가짜 노드가 선택되거나 노드로서 의미 없는 점이 선택되는 것이다.
AHN(2008)은 에지가 되기 위한 조건으로 첫째, 두 개의 지역최소점으로 구성되어야 하며 둘째, 이 두 점이 이루는 영역이 문턱값 $A_{thresh}\left(=\frac{n}{2}\theta w\right)$ 이상이어야 한다고 정함으로써 가짜 노드가 검출되는 것을 최소화했다. 문턱값에서 θ와 w는 두 지역최소점의 각도와 넓이, n은 민감도 조절 시 할당될 수 있는 다중도계수(multiplicity factor)를 의미한다.



출처: Ahn, S., Doh, N. L., Chung, W. K., & Nam, S. Y. (2008). The robust construction of a generalized Voronoi graph (GVG) using partial range data for guide robots. Industrial Robot: An International Journal, 35(3), 259-265.
[그림 1-6] 보로노이지도에서 지역최소점을 이용한 에지와 노드 검출 원리



출처: Ahn, S., Doh, N. L., Chung, W. K., & Nam, S. Y. (2008). The robust construction of a generalized Voronoi graph (GVG) using partial range data for guide robots. Industrial Robot: An International Journal, 35(3), 259-265.
[그림 1-7] 실제 복도를 따라 거리측정센서로 그린 보로노이 지도와 설계도와의 비교

[그림 1-7]은 실제 복도를 따라 거리측정센서로 보로노이지도를 그린 것으로 가운데 둥근 점인 노드가 잘 검출된 것을 확인할 수 있다.

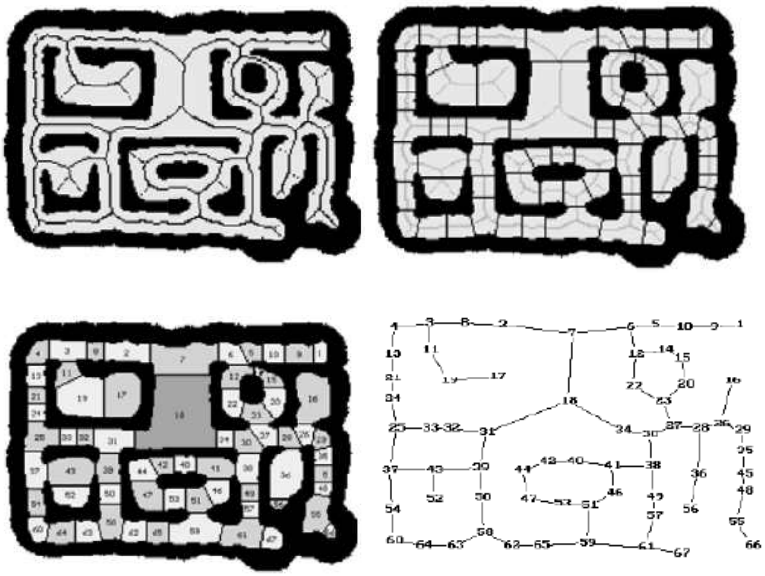
3 각 지도간의 위치가 매핑되는 데이터 구조로 프로그래밍

앞에서 각 지도를 제작하는 원리와 알고리즘을 이해했으므로 다음은 지도 간 결합 방법에 대해 알아보자. 이를 하이브리드지도(hybrid map)라고 부른다.

1. 하이브리드 지도

하이브리드 지도로 가장 많이 사용되는 결합법은 위상지도와 점유격자지도/특징지도의 결합이다. 위상지도는 로봇이 경로를 생성하고 판단을 할 때 전역 환경을 설명하는 데 사용되고, 점유격자지도 또는 특징지도는 위상지도의 측정 정보가 부족한 점을 보완하여 지역 환경을 표현하는 데 사용된다.

THRUN(1996)은 점유격자지도와 위상지도를 결합하는 방법을 제안했다. 결합 순서 면에서 점유격자지도가 먼저 작성되고, 그 위에 임계선(critical line)을 이용하여 점유격자지도를 응집(coherent) 영역으로 분할한 토폴로지 맵이 생성된다. 점유격자지도는 중간 정도의 복잡성 환경에서 쉽게 구성하고 유지할 수 있는 장점이 있고, 위상지도는 빠른 경로 계획이 가능한 장점이 있으므로 두 지도의 결합은 시너지 효과를 낸다. [그림 1-8]은 하이브리드 지도를 만드는 과정을 나타낸다.



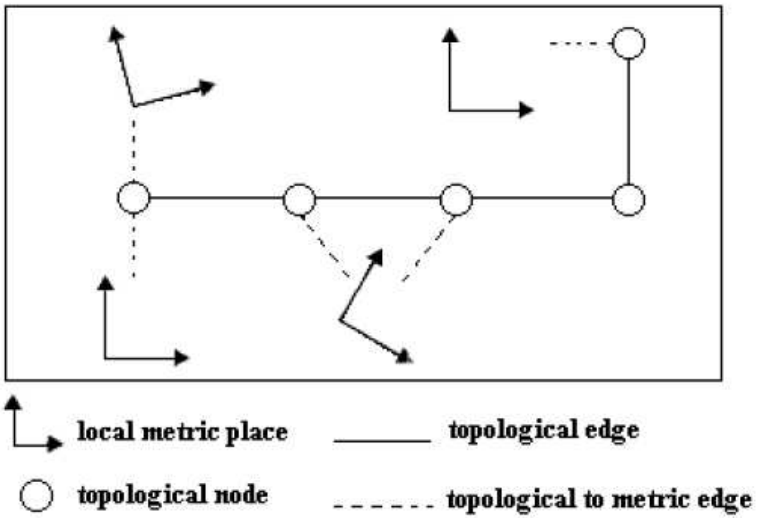
출처: Chen, C., & Cheng, Y. (2008). Research on map building by mobile robots. Proceedings of Second International Symposium on Intelligent Information Technology Application, 673-677.
[그림 1-8] 점유격자지도와 위상지도를 결합하는 하이브리드지도 제작 과정(위 왼쪽: 보로노이지도, 위 오른쪽: 임계선 표시, 아래 왼쪽: 위상지도 영역, 아래 오른쪽: 위상 그래프)

TOMATIS(2001)는 특징지도와 위상지도의 특성을 이용하여 다음 두 단계의 환경 모델 추상화를 표현했다.

1단계: 어떤 장소는 특징지도와 같은 지역 측정지도(metric map)로 정의하며 가까운 범위 안에서 이동이 가능하다.

2단계: 로봇이 한 장소에서 다른 장소로 이동할 때 지도시스템은 특징지도(측정지도)에서 위상지도로 전환한다. 그리고 로봇이 목적한 장소로 이동하면 다시 위상지도에서 특징지도로 전환한다.

[그림 1-9]에서 환경은 특징지도와 위상지도 노드에 의해 주어진 위치로 표현된다. 로봇이 노드에서 장소로 이동할 때 하이브리드 지도 시스템은 위상지도에서 특징지도로 또는 그 반대로 전환되는 것을 나타낸다.



출처: Chen, C., & Cheng, Y. (2008). Research on map building by mobile robots. Proceedings of Second International Symposium on Intelligent Information Technology Application, 673-677.
[그림 1-9] 특징지도와 위상지도를 결합하는 하이브리드지도

수행 내용 / 지도 작성하기

재료・자료

- 지도 파일
- ROS 사용법 소개 책자
- ROS 패키지 코드

기기(장비・공구)

- 컴퓨터, 프린터
- 지도파일 뷰어 소프트웨어

- 문서 작성용 소프트웨어
- 프로그램 개발용 소프트웨어

안전・유의 사항

- 로봇이 작업 공간에서 이동하면서 지도를 제작하는 경우 프로그램 오류나 오작동에 의해 로봇이 기물을 파손하거나 인명 피해를 일으킬 수 있으므로 작업 공간에 불필요한 장애물을 제거하고 로봇에 비상 정지 스위치나 버튼을 부착하여 실험을 시행한다.

수행 순서

[1] 작업 환경을 표현하는 여러 지도의 특징에 대해 파악하낟.

1. 로봇이 이동하면서 서비스를 수행할 작업 환경을 정하고, 지도 확보 후 로봇이 다닐 수 있도록 장애물을 제거한다.
2. 작업 환경의 지도 데이터를 CAD 파일 등의 형태로 제작한 것이 있으면 이 파일을 확보하고 뷰어 소프트웨어를 준비한다.
3. 지도 데이터가 있으면 이 지도가 점유격자지도, 특징지도, 위상지도 중 어느 것에 해당하는지 확인한다.
4. 로봇의 작업 내용과 작업 환경의 특성을 고려할 때 세 종류 지도 중 어떤 지도가 가장 적합한지 비교 검토하여 결정한다.
5. 이동지능을 수행할 로봇의 시뮬레이션 소프트웨어를 다운로드한다. 로봇 전용 소프트웨어가 없다면 범용 로봇 시뮬레이터를 다운로드하거나 직접 코드를 작성한다.

[2] 각 지도를 적합한 데이터 구조로 프로그래밍한다.

1. 로봇의 작업 환경 지도를 보기 위해 공개 소프트웨어인 ROS(robot operating system)와 관련 시각화(visualization) 소프트웨어를 설치한다.
   ROS는 다양한 로봇 응용프로그램의 운영체제급 로봇용 소프트웨어 플랫폼으로서 2007년 스탠포드 대학 AI Lab에서부터 개발이 시작되고 WILLOW GARAGE라는 회사가 이를 이어 받아 2010년에 ROS 1.0을 개발한 후 매년 새로운 버전이 알파벳 순서의 이름으로 발표되고 있으며 2017년에 ROS 2.0이 발표되었다. 현재 ROS는 Open Robotics에 의해 운영되고 있으며, BSD 라이선스를 기반으로 하고 있어 누구든지 무료로 다운로드하여 수정, 재사용, 재배포가 가능하다.
   (1) ROS를 사용하기 위해 컴퓨터 운영체제로 ROS가 공식적으로 지원하는 Ubuntu 16.04를 설치한다. ROS 관련 교재의 예제 코드를 사용하기 위해 다음 명령어로 ROS Kinetic Kame를 설치한다.

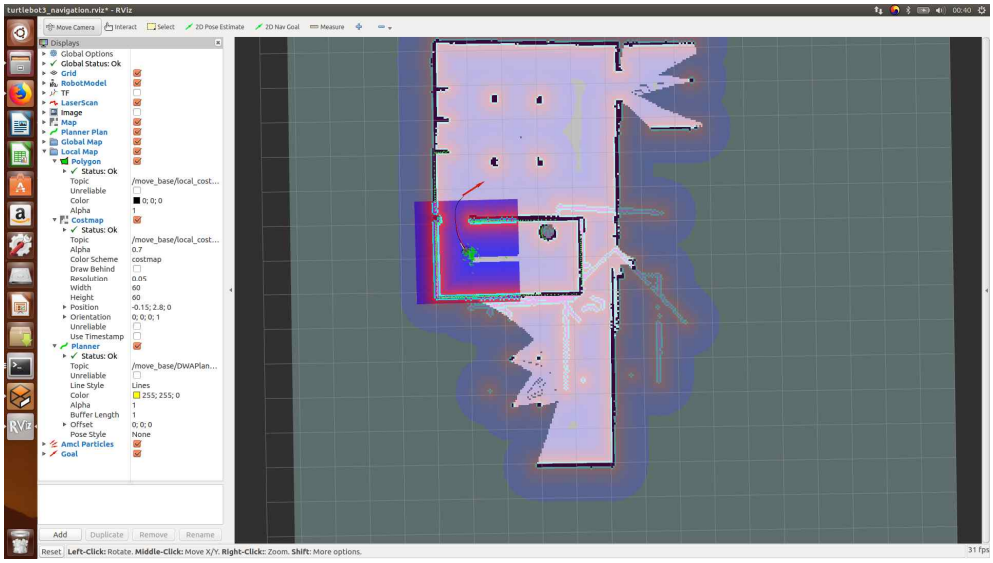
```
$ sudo apt-get install ros-kinetic-desktop-full
```

(3) 3차원 시각화 도구인 RViz를 다음 명령으로 설치한다.

RViz는 레이저 거리센서로부터 장애물과의 거리, Kinect 같은 3차원 거리센서의 포인트 클라우드 데이터 등을 3차원으로 표시할 수 있어서 로봇 시뮬레이션과 실험에 유용하다.

```
$ sudo apt-get install ros-kinetic-rviz
```

[그림 1-10]은 RViz를 실행한 화면의 예이다.



[그림 1-10] RViz 실행 화면

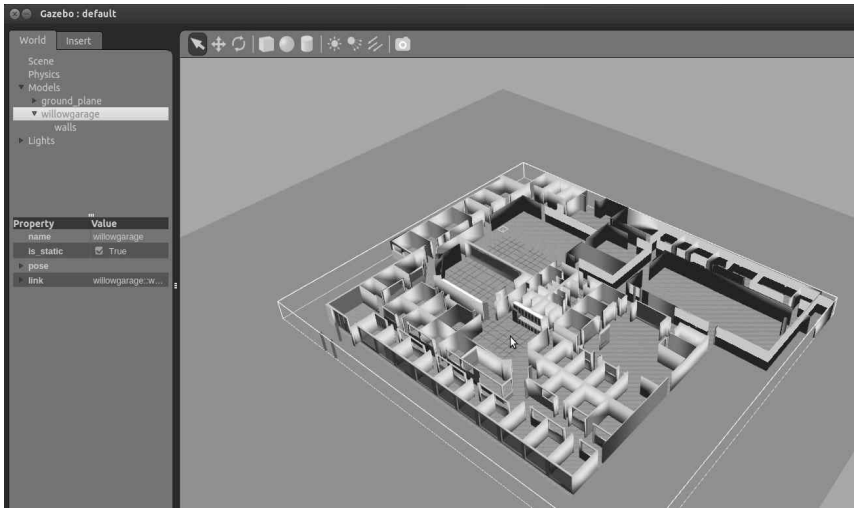
(4) 3차원 시뮬레이터인 Gazebo를 실행해 본다.

Gazebo는 3차원 시뮬레이션을 위한 로봇, 센서, 환경 모델 등을 지원하고 다양한 물리 엔진(Bullet, Simbody, DART)을 탑재하여 실제와 근사한 결과를 얻을 수 있는 3차원 시뮬레이터이다. Gazebo는 ROS Kinetic 버전에 기본적으로 설치되어 있으므로 다음 명령어를 사용하면 된다.

```
# gazebo
```

roslaunch 명령어를 사용하여 WILLOW GARAGE사의 예전 사무실 지도를 아래와 같이 gazebo로 볼 수 있다.

```
# roslaunch gazebo_ros willowgarage_world.launch
```



[그림 1-11] Gazebo로 WILLOW GARAGE사의 사무실 지도 보기

2. ROS의 map_server 패키지를 사용하여 지도를 열어본다.

Map_server 패키지는 ROS 서비스로 지도 데이터를 제공하는 map_server와, 동적으로 생성되는 지도를 파일로 저장하는 map_saver 명령줄 유틸리티를 제공한다.

ROS에서는 높은 가독의 데이터 직렬화 언어인 YAML(YAML Ain't Markup Language) 타입으로 지도의 메타데이터와 이미지 파일을 설명하며, 이미지 파일은 작업 환경의 점유 여부를 데이터로 코딩한 것이다. 지도와 관련된 YAML 포맷 중 image는 점유 데이터를 포함하고 있는 영상파일 이름, resolution은 각 픽셀이 나타내는 크기(이 경우 10cm), origin은 지도 원점의 x좌표, y좌표, yaw각을 나타낸다. 각 픽셀의 부호화(decoding)한 값이 occupied_thresh보다 크면 점유공간으로, free_thresh보다 작으면 빈 공간으로 간주한다. negate는 흑백을 반전하게 한다.

```
image: map.png
resolution: 0.1
origin: [0.0, 0.0, 0.0]
occupied_thresh: 0.65
free_thresh: 0.196
negate: 0
```

(1) 로봇이 만든 지도를 보려면 다음 명령어를 사용한다.

```
$ rosrun map_server map_server map.yaml
```

(2) 로봇을 이동시키면서 로봇의 오도메트리나 거리 스캔 정보를 기반으로 지도를 작성하려는 경우 다음 명령어를 사용한다.

코드에서 -f는 작성한 지도파일을 저장할 폴더 및 파일명을 지정하는 옵션이며 ~/map은 사용자 폴더(~/)에 map.pgm, map.yaml의 이름으로 저장하라는 뜻이다.

```
$ rosrun map_server map_saver -f ~/map
```

3. 구글이 오픈소스로 공개한 카토그래퍼(cartographer)를 사용해 본다.

카토그래퍼는 실시간 SLAM 기술로 지도를 제작하는 데 필요한 라이브러리로 2D, 3D 환경 및 ROS(robot operating system)를 지원한다. 카토그래퍼를 다운로드할 수 있는 github 사이트의 주소는 https://github.com/googlecartographer이다.

(1) 카토그래퍼를 빌드하기 위해 다음 코드를 실행한다.

```
# python, wstool, rosdep 설치
sudo apt-get update
sudo apt-get install -y python-wstool python-rosdep ninja-build

# catkin_ws에 새 작업공간 생성.
mkdir catkin_ws
cd catkin_ws
wstool init src

# cartographer_ros.rosinstall 파일과 의존성 위한 fetch 코드 병합
wstool merge -t src https://raw.githubusercontent.com/googlecartographer
/cartographer_ros/master/cartographer_ros.rosinstall
wstool update -t src

# proto3 설치
src/cartographer/scripts/install_proto3.sh

# deb 의존성 설치
sudo rosdep init
rosdep update
rosdep install --from-paths src --ignore-src --rosdistro=${ROS_DISTRO} -y

# 빌드와 설치
catkin_make_isolated --install --use-ninja
source install_isolated/setup.bash
```

(2) 카토그래퍼 데모를 다음 코드로 실행한다.

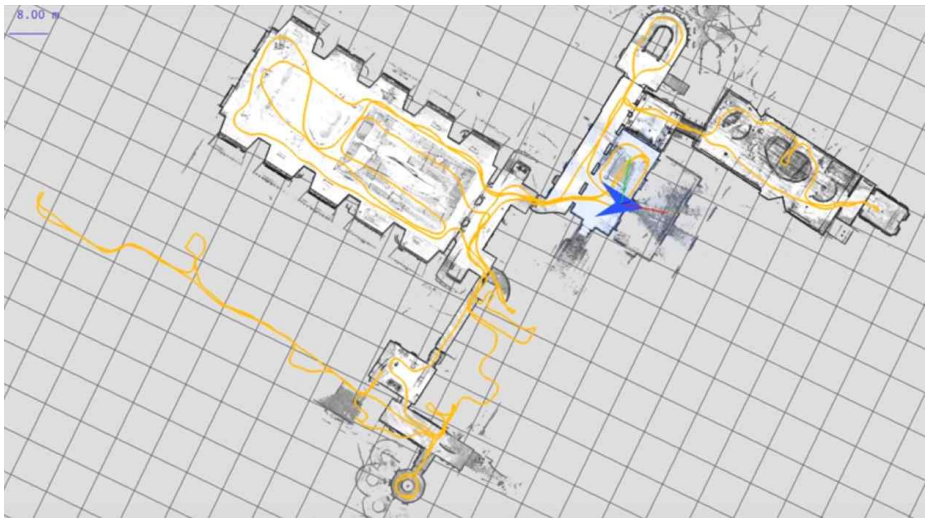
실행 결과로 그림 1-12와 같은 공간지도가 그려지는 것을 볼 수 있다.

```
# 2D 백팩 예제(backpack example) bag 파일 다운로드
wget -P ~/Downloads https://storage.googleapis.com/cartographer-public-data/bags/backpack_2d/cartographer_paper_deutsches_museum.bag

# 2D 백팩 데모 론치
roslaunch cartographer_ros demo_backpack_2d.launch bag_filename:=${HOME}/Downloads/cartographer_paper_deutsches_museum.bag

# 3D 백팩 예제(backpack example) bag 파일 다운로드
wget -P ~/Downloads https://storage.googleapis.com/cartographer-public-data/bags/backpack_3d/with_intensities/b3-2016-04-05-14-14-00.bag

# 3D 백팩 데모 론치
roslaunch cartographer_ros demo_backpack_3d.launch bag_filename:=${HOME}/Downloads/b3-2016-04-05-14-14-00.bag
```



출처: https://www.youtube.com/watch?time_continue=33&v=DM0dpHLhtX0

[그림 1-12] 구글 카토그래퍼로 작성한 공간지도

2-1. 로봇 위치 매핑

학습 목표

- 현재의 로봇 위치를 지도와 매핑할 수 있다.
- 지역적 로봇 위치 계산 오차를 보정할 수 있다.

필요 지식 /

1 현재의 로봇 위치를 지도와 매핑하기

로봇이 환경 지도를 가지고 있거나 만들어 가면서 목표 지점까지 이동하기 위해서는 현재의 로봇 위치를 정확히 알아야 하는데 이를 로봇의 위치인식(localization)이라고 한다.

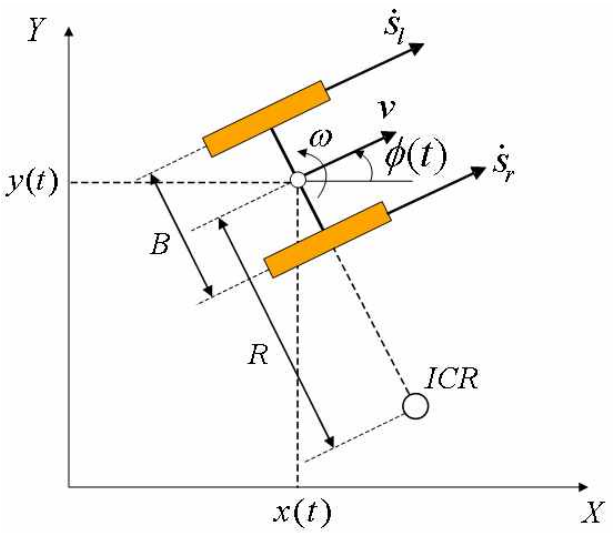
위치인식 방법에는 사용되는 센서에 따라 상대 위치인식 기법과 절대 위치인식 기법이 있다. 상대 위치인식 기법의 대표적인 예는 로봇 바퀴에 부착된 엔코더(encoder), 타코미터(tachometer)를 통해 측정된 바퀴의 회전수를 이용해서 로봇의 주행 거리를 계측하는 오도미터(odometer)나 자이로센서, 가속도센서를 이용하여 로봇의 변화된 위치를 계산하는 방법이다. 이 방법은 로봇의 바퀴나 본체에 부착하는 것이 용이하고 가격이 저렴하다는 장점이 있지만, 주행하면서 바퀴의 미끄러짐 때문에 오도미터 데이터로부터 회전 각속도 측정 시 오차가 발생하고 이 측정값을 적분하여 회전각 위치를 계산하는 과정에서 오차가 점차 누적되어 정확한 위치인식이 어렵다는 단점이 있다.

절대 위치인식 기법은 레이저 거리센서, 비전센서, 초음파센서, 적외선 스캐너, GPS(global positioning system), 지자기 센서, RFID(radio frequency identification) 등으로 측정한 거리, 방향, 영상 정보 등을 이용하여 로봇이 지도 상에서 어디에 있는지를 추정하는 방법이다. 이 방법은 상대 위치인식 기법처럼 적분법을 사용하지 않기 때문에 누적되는 오차가 작다는 장점이 있지만, 별도의 하드웨어 모듈을 추가해야 하고 특히 GPS는 실내에서 사용하기 어렵다는 단점이 있다.

1. 추측항법을 이용한 위치인식

추측항법(dead reckoning)은 오도미터나 자이로센서, 가속도센서 등의 관성센서(inertial measurement unit) 데이터를 이용하여 로봇의 현재 위치를 추정하는 기술인데, 이 방법을 쓰면 바퀴의 미끄러짐이나 센서 노이즈로 인해 실제 로봇의 위치에 오차가 발생하고 이 값이 계속 누적되기 때문에 다양한 센서 정보를 융합하여 정확한 위치로 보정해야 한다.

추측항법을 적용하기 위해서는 로봇의 위치와 오도메트리(odometry, 오도미터 데이터) 간 관계식이 필요하다. [그림 2-1]은 모바일 로봇에 많이 사용되는 차동형(differential) 이동로봇의 기구학 모델을 나타낸 것으로 ICR(instantaneous center of rotation)은 두 바퀴로 이동하는 로봇의 회전 중심점이며, B는 두 바퀴 사이의 거리, R은 ICR과 로봇 중심점 간의 거리를 의미한다. s_r과 s_l은 오른쪽과 왼쪽 바퀴의 이동거리, $\dot{s}_r$과 $\dot{s}_l$은 두 바퀴의 선형속도를 각각 나타내며, v, ϕ, ω는 로봇 중심점의 이동속도, 회전각, 회전 각속도를 각각 나타낸다.



[그림 2-1] 차동형 이동로봇의 기구학 모델

직전 샘플시간과 현재 샘플시간의 로봇 각도를 $\phi(k-1)$, $\phi(k)$이라고 하면 샘플시간 T 동안 평균 회전각은 $\frac{\phi(k-1)+\phi(k)}{2}=\phi(k-1)+\frac{\omega T}{2}$이고, 이동거리는 vT이다. 그러므로 현재 로봇의 상태벡터 $\boldsymbol{p}(k)=[x(k)\,y(k)\,\phi(k)]^T$는 직전 샘플시간의 로봇 상태벡터 $\boldsymbol{p}(k-1)$에 대해 다음과 같은 이산시간 상태방정식(discrete-time state equation)으로 표현할 수 있다(정완균 · 도낙주 · 이수용 외, 2012).

$$\boldsymbol{p}(k)=\boldsymbol{p}(k-1)+\begin{bmatrix} vT\cos\left(\phi(k-1)+\dfrac{\omega T}{2}\right) \\ vT\sin\left(\phi(k-1)+\dfrac{\omega T}{2}\right) \\ \omega T \end{bmatrix},\ \boldsymbol{p}(k)=\begin{bmatrix} x(k) \\ y(k) \\ \phi(k) \end{bmatrix} \qquad \text{식 (18)}$$

식 (18)을 제어입력 $\boldsymbol{u}(k)$에 대해 표현하기 위해 제어입력을 양바퀴의 선형 이동거리, 즉 $\boldsymbol{u}(k)=[s_r(k)\ s_l(k)]^T$라고 정의하자. 그러면 식 (18)에서 로봇의 회전속도와 선형속도는 다음의 관계식으로 표현된다.

$$\omega=\frac{\dot{s}_r-\dot{s}_l}{B},\quad v=\frac{\dot{s}_r+\dot{s}_l}{2} \qquad \text{식 (19)}$$

식 (19)를 식 (18)에 대입하여 정리하면 다음과 같이 실제 로봇 상태의 추측항법 기본식을 얻을 수 있다.

$$\boldsymbol{p}(k)=\boldsymbol{p}(k-1)+\begin{bmatrix} \dfrac{s_r+s_l}{2}\cos\left(\phi(k-1)+\dfrac{s_r-s_l}{2B}\right) \\ \dfrac{s_r+s_l}{2}\sin\left(\phi(k-1)+\dfrac{s_r-s_l}{2B}\right) \\ \dfrac{s_r-s_l}{B} \end{bmatrix},\ \boldsymbol{p}(k)=\begin{bmatrix} x(k) \\ y(k) \\ \phi(k) \end{bmatrix} \qquad \text{식 (20)}$$

이 식에 양쪽 바퀴에서 발생하는 외란 $\delta s_l, \delta s_r$의 영향을 넣기 위해 $s_r{}'=s_r+\delta s_r$, $s_l{}'=s_l+\delta s_l$을 새롭게 정의하여 식 (20)에 대입하면 다음과 같이 추측항법 최종식을 구할 수 있다.

$$\boldsymbol{p}(k)=\boldsymbol{p}(k-1)+\begin{bmatrix} \dfrac{s'_r+s'_l}{2}\cos\left(\phi(k-1)+\dfrac{s'_r-s'_l}{2B}\right) \\ \dfrac{s'_r+s'_l}{2}\sin\left(\phi(k-1)+\dfrac{s'_r-s'_l}{2B}\right) \\ \dfrac{s'_r-s'_l}{B} \end{bmatrix},\ \boldsymbol{p}(k)=\begin{bmatrix} x(k) \\ y(k) \\ \phi(k) \end{bmatrix} \qquad \text{식 (21)}$$

실제 실험 환경에서는 외란 $\delta\boldsymbol{u}=[\delta s_r\ \delta s_l]^T$를 알 수 없으므로, 이에 대한 영향을 배제하고 로봇의 상태를 추정(estimation)한 후 추정된 값의 신뢰도를 확률적인 방법으로 표현하는 것이 바람직하다. 그러므로 로봇 상태 추정을 위한 추측항법은 다음 식과 같이 간략화할 수 있다.

$$\hat{\boldsymbol{p}}_r = \begin{bmatrix} \hat{x}_r \\ \hat{y}_r \\ \hat{\phi}_r \end{bmatrix} = \begin{bmatrix} \hat{x}_r(k-1) \\ \hat{y}_r(k-1) \\ \hat{\phi}_r(k-1) \end{bmatrix} + \begin{bmatrix} \frac{s_r+s_l}{2}\cos\left(\hat{\phi}_r(k-1)+\frac{s_r-s_l}{2B}\right) \\ \frac{s_r+s_l}{2}\sin\left(\hat{\phi}_r(k-1)+\frac{s_r-s_l}{2B}\right) \\ \frac{s_r-s_l}{B} \end{bmatrix} \quad \text{식 (22)}$$

$$= \boldsymbol{f}(\hat{\boldsymbol{p}}_r(k-1), \boldsymbol{u(k-1)})$$

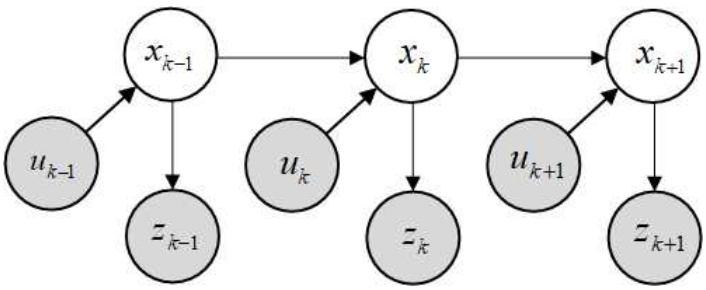
2. 베이즈 필터

로봇이 얻는 데이터에는 노이즈가 많기 때문에 환경의 상태(state)의 참 값에 대한 확신값(belief)이 필요하다. 또 로봇은 움직이면서 각종 센서 데이터로 환경의 상태를 측정하기 때문에 측정(observation)과 제어입력(control input)이 상호 작용하며 서로 영향을 미친다.

이산시간 상태 x가 '완전하다(complete)'는 것은 이전 시간에 발생한 모든 것에 대한 충분한 요약으로 정의되며, 이 경우 상태 변화를 표현하는 확률법칙은 다음과 같이 조건부확률 형태로 표현할 수 있다. 식에서 우변에 있는 항 $p(x_k|x_{k-1}, u_k)$을 상태천이확률(state transition probability)이라고 한다.

$$p(x_k|x_{0:k-1}, \boldsymbol{z}_{1:k-1}, u_{1:k}) = p(x_k|x_{k-1}, u_k) \quad \text{식 (23)}$$

식에서 $x_{0:k-1}$은 0초부터 직전 시간인 $k-1$번째 샘플시간까지의 상태변수를, $\boldsymbol{z}_{1:k-1}$은 직전 시간까지 측정한 센서값을, $u_{1:k}$는 현재까지의 제어입력을 의미한다. 편의상 로봇이 i번째 반복에서 제어입력 u_i를 먼저 수행한 다음에 측정값 z_i를 취득하는 것으로 약속하자.



[그림 2-2] 측정, 상태, 입력의 진행을 표현하는 동적 베이지안 네트워크 또는 은닉마코프모델(hidden Markov model)

이와 유사한 관계를 사용하여 다음 식과 같이 현재 측정값 z_k는 현재 상태 x_k로부터 예측할 수 있는데, 식에서 우변에 있는 항 $p(z_k|x_k)$를 측정확률(measurement probability)이라고 한다.

$$p(z_k|x_{0:k}, \boldsymbol{z}_{1:k-1}, u_{1:k}) = p(z_k|x_k) \quad \text{식 (24)}$$

로봇의 포즈(pose)는 로봇의 2차원 위치와 방향각으로 구성된 벡터값을 의미한다. 전역좌표계에서 로봇 자신이 현재 포즈(상태)를 정확히 측정하는 것은 어려우며 노이즈가 섞인 데이터로부터 추론해야 한다. 이렇게 로봇이 현재의 참 상태에 대해 추론한 값을 추정값이라고 하며, 다음과 같은 사후확률(posterior) 식으로 표현된다.

$$bel(x_k) = p(x_k|z_{1:k}, u_{1:k}) \quad \text{식 (25)}$$

한편, 현재 측정값 z_k를 모르는 상태에서 x_k를 예측(prediction)할 필요성이 있을 때 다음과 같은 사후확률 식으로 나타낼 수 있다.

$$\overline{bel}(x_k) = p(x_k|z_{1:k-1}, u_{1:k}) \quad \text{식 (26)}$$

$\overline{bel}(x_k)$는 현재 시간에서 측정값을 사용하지 않고 이전 상태 사후확률로부터 상태를 예측하는 것이며, $\overline{bel}(x_k)$로부터 $bel(x_k)$를 계산하는 것을 교정(correction) 또는 측정 업데이트(measurement update)라고 부른다.

베이즈 필터 알고리즘은 이와 같은 예측과 교정을 재귀적으로 계산하는 원리로 구성되어 있으며 의사코드(pseudo-code)로 설명하면 다음과 같다.

Bayes_filter$(bel(x_{t-1}), u_t, z_t)$
  for all x_t do
    $\overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1}$
    $bel(x_t) = \eta p(z_t|x_t) \overline{bel}(x_t)$
  endfor
  return $bel(x_t)$

3. 칼만 필터를 이용한 위치인식

칼만 필터(Kalman filter)는 피드백 제어와 유사한 형태로 프로세스의 상태를 추정하는 알고리즘으로서, 필터가 어떤 시점에서 프로세스의 상태와 그에 따른 출력값을 예측하는 식

으로 추정(estimate)한 후, 잡음이 섞여 있는 출력값을 측정하여 그 오차를 계산하고 이를 이용하여 교정하는 식을 그 다음 샘플 시간의 추정 모델에 반영함으로써 점차 그 추정 오차를 줄여가는 원리에 기반하며, 기본적으로 베이즈 필터의 일종이다. 전자의 경우를 사전(a priori) 추정을 이용한 시간 업데이트라 하고, 후자의 경우를 사후(a posteriori) 추정을 이용한 측정(measurement) 업데이트라고 한다(WELCH and BISHOP, 2001).

가장 단순한 형태인 이산시간 칼만필터를 설명하기 위해 다음과 같은 선형 이산시간 제어 프로세스의 상태 $\boldsymbol{x}_k$를 추정하는 문제를 생각하자.

$$\boldsymbol{x}_k = A\boldsymbol{x}_{k-1} + Bu_k + w_{k-1}, \quad z_k = H\boldsymbol{x}_k + v_k \qquad \text{식 (27)}$$

여기서 w_k와 v_k는 프로세스 잡음과 측정 잡음을 의미하며, 각각의 공분산(covariance)행렬은 P와 Q이다. z_k는 센서로 측정한 값을 의미한다.

먼저 칼만필터의 사전 추정(예측)을 위한 시간 업데이트에 관한 식을 쓰면 다음과 같다.

$$\hat{\boldsymbol{x}}_k^- = A\hat{\boldsymbol{x}}_{k-1} + Bu_k, \qquad \text{식 (28)}$$

$$P_k^- = AP_{k-1}A^T + Q$$

여기서 위첨자에 '-' 기호가 붙은 것은 해당 값이 사전 추정(예측)되었음을 의미한다.

그 다음으로 칼만필터의 사후 추정(교정)을 위한 측정 업데이트에 관한 식을 쓰면 다음과 같다.

$$K_k = P_k^- H^T\left(HP_k^- H^T + R\right)^{-1}, \qquad \text{식 (29)}$$

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + K_k\left(z_k - H\hat{\boldsymbol{x}}_k^-\right),$$

$$P_k = \left(I - K_k H\right)P_k^-$$

이 식은 그 다음 샘플타임에서 식 (28)의 k를 1 증가시켜 P_{k-1}을 P_k로 업데이트 한 후에 대입된다.

이외에도 비선형 시스템을 위한 EKF(extended kalman filter), 정확성을 보완한 UKF(unscencted kalman filter) 등이 많이 사용된다.

4. SLAM

SLAM은 로봇에게 환경 공간의 지도 데이터가 없는 경우 센서 정보를 이용하여 환경에 대한 지도를 작성하고, 그와 동시에 작성된 지도로부터 로봇의 현재 위치를 인식하는 기

술을 총칭한다. 이 기술은 자율 주행 자동차, 무인 비행체, AUV(autonomous underwater vehicle), 가정용 로봇 등에 점점 더 많이 적용되고 있다(송재복 · 황서연, 2014). [그림 2-3]은 물품 이송에 많이 사용되고 있는 모바일 로봇의 모습이다.

[그림 2-3] SLAM 기능이 있는 모바일 로봇

SLAM을 구성하는 알고리즘으로는 아래의 요소들이 있다.

- 위상지도 작성
- 거리 센서(레이저 스캐너, 초음파 센서, LiDAR, RADAR 등)와 비전 센서(스테레오 카메라, 모노 카메라, 전방향 카메라, 깊이지각 카메라), 오도미터 등의 다양한 센서들을 이용한 환경 센싱 및 센서 융합(sensor fusion)
- 잡음이 있는 상태에서 센싱 정확도를 높이기 위한 로봇 자신의 기구학 모델링
- 랜드마크를 포함한 다수의 애매한 물체 인식
- 이전에 지나간 영역을 인식하여 영역간의 위치정보를 결합하는 루프종료(loop closure)
- 지도를 최대한 효율적으로 작성하기 위한 경로 탐색
- 넓은 공간에서 증가하는 센싱 정보와 계산량을 효율적으로 관리하는 인공지능 기술

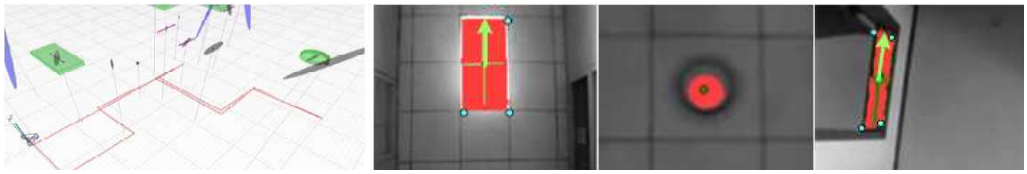
SLAM의 종류는 다음과 같이 구분된다.

(1) 칼만필터 기반 SLAM

선형 시스템에는 칼만필터를 적용하고, 비선형 시스템에는 EKF와 UKF가 사용되고 있다. 특히 EKF는 대부분의 비선형 시스템에서 추정 성능이 우수하므로 SLAM에 널리 사용되고 있다. 모노 카메라를 이용하는 경우 한 시점에서 특징에 대한 거리 정보를 얻기 위해 특징초기화(feature initialization, 여러 시점에서의 관측 정보를 통해 특징의 위치 계산) 과정이 필요하다. 특징의 개수가 N일때 EKF의 연산량은 $O(N^2)$이어서 연산량이 특징의 개수에 비해 기하급수적으로 증가한다.

비전센서 기반의 SLAM은 카메라의 방향 및 종류에 따라 다양한 방법들이 시도되고

있는데, 이동로봇의 측면에 모노 카메라를 붙이고 그 영상으로부터 코너 및 직선 특징을 추출하는 EKF-SLAM, 로봇의 전방을 향하도록 카메라를 붙이고 스케일 및 어파인(affine) 변화를 보정하고 SIFT(scale-invariant feature transform)를 이용하여 특징이나 물체를 추출하는 SLAM, 천장을 향하도록 카메라를 붙이고 형광등이나 조명, 코너 특징을 이용하는 SLAM 등이 있다.



[그림 2-4] 천장지향 모노 카메라 기반 SLAM

그 외에도 360° 화각을 가진 전방향 카메라와 레이저스캐너 정보를 융합한 SLAM, 초음파센서 데이터를 이용한 EKF-SLAM 등의 방법도 있다.

(2) 파티클필터 기반 SLAM

RBPF(rao-blackwellization particle filter) 기반 방법을 주로 사용하며, 파티클필터로 로봇의 위치를 추정하고 EKF로 특징지도를 작성하는 FastSLAM이 많이 사용되고 있으며 모노 카메라 또는 스테레오 카메라가 장착된 쿼드롭터로 FastSLAM을 수행했다(LEE, FRAUNDORFER and POLLEFEYS, 2011).

RBPF에 기반한 다중 로봇 SLAM도 제안되었는데, 각 로봇은 천장 지향 모노 카메라를 이용하여 출발점을 기준으로 특징지도를 작성해 나간 후 각 로봇이 만든 특징지도에서 중첩부분 감지 후 최적 변환을 통해 지도를 통합하여 제작했다(LEE and LEE, 2009).

(3) 그래프 기반 SLAM

그래프 기반 SLAM, 일명 GraphSLAM에서 노드는 로봇이 지나간 지점 또는 특징 위치를 나타내고 에지는 노드와 노드 사이의 구속조건을 표현하며, 전체 그래프에 누적된 오차를 최소화하면서 루프 종료를 감지한다(GUTMANN and KONOLIGE, 1999).

GraphSLAM은 센서 정보를 이용하여 그래프를 작성하는 단계(front-end)와 작성된 그래프의 오차를 최적화하는 단계(back-end)로 구분된다. 전방 모노 카메라를 이용하여 임베디드 시스템에서 GraphSLAM을 구현하기도 했는데, front-end에서는 5점 RANSAC 정보로 로봇의 위치를 예측 및 측정했으며, back-end에서는 그래프 최적화 및 노드 개수 최소화를 통해 연산량을 줄였다(EADE, FONG, and MUNICH, 2010).

(4) 비전 기반 SLAM

비전 분야에서는 영상특징 기반 3차원 복원 문제에 사용되는 BA(bundle adjustment)를 기반으로 한 SLAM에 대한 연구가 활발히 이뤄지고 있으며 카메라와 특징들의 위치를 동시 보정하는 용도로 활용되고 있다.

PTAM(parallel tracking and mapping)법은 모노 카메라를 이용하여 RANSAC 알고리즘으로 카메라 이동과 특징 위치를 예측하였고, 일정 거리 이상 이동 시 특징을 키프레임(key frame)에 등록하였다. 그리고 실시간성 확보를 위해 현재 카메라 위치에 인접한 5개의 키프레임들로 지역적 BA를 수행하여 각 키프레임과 특징들의 위치를 보정하였다(STRASDAT and DAVISON, 2011).

2 지역적 로봇 위치 계산 오차 보정

로봇이 환경 공간에서 경로를 따라 목표 지점까지 이동하는 동안 다음과 같이 두 종류의 오차가 발생한다.

- 구조적(structured) 오차: 정밀 측정으로 보정 가능한 바퀴 직경 오차, 축 간 거리 오차, 바퀴 정렬 불량, 엔코더 정밀도 오차 등
- 비구조적(unstructured) 오차: 확률적 기법으로만 모사 가능한 바퀴의 미끄러짐, 옆미끄러짐(skidding), 바퀴와 지면 간의 비점접촉(non-point contact) 등

로봇이 이런 오차에도 불구하고 정확히 목표 지점에 도달하기 위해서는 로봇에 설치되어 있는 각종 센서(엔코더, 관성 센서, 거리 측정 센서 등)에서 측정된 값들을 최대한 활용하여 로봇의 위치와 방향을 정확히 추정(estimate)할 수 있어야 한다.

이를 위해 로봇의 상태에 대해 실제 측정된 값을 $\boldsymbol{p}_r = [x_r\ y_r\ \phi_r]^T$, 추정한 값을 $\hat{\boldsymbol{p}}_r = [\hat{x}_r\ \hat{y}_r\ \hat{\phi}_r]$이라고 하고, $k-1$번째 샘플시간에서 측정된 모터입력을 $\boldsymbol{u}(k-1)$, 실제 외란(disturbance)으로 인가되는 입력을 $\delta\boldsymbol{u}$라고 정의하자. 외란입력 $\delta\boldsymbol{u}$는 정확히 측정하기 어려우므로, 평균이 0이고 공분산행렬이 Q인 정규분포로 가정한다.

이동로봇의 입력을 양쪽 바퀴의 선형 이동거리로 정의하면 $\boldsymbol{u}(k-1) = [s_r\ s_l]^T$의 관계를 가진다. [그림 2-1]을 보면 $\dot{s}_l = -\omega(R + B/2)$, $\dot{s}_r = -\omega(R - B/2)$의 관계가 있어서 이로부터 $\omega = (\dot{s}_r - \dot{s}_l)/B$의 관계식을 얻을 수 있다. 또 선형속도 식은 $v = \dfrac{\dot{s}_r + \dot{s}_l}{2}$로 표현되므로, 두 식을 이용하면 다음의 관계식이 성립한다.

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{B} & -\frac{1}{B} \end{bmatrix} \begin{bmatrix} \dot{s}_r \\ \dot{s}_l \end{bmatrix} \qquad \text{식 (30)}$$

이로부터 다음과 같이 제어 입력을 표현할 수 있다.

$$\boldsymbol{u}(k-1)=\begin{bmatrix}s_r\\ s_l\end{bmatrix}=\delta_t\begin{bmatrix}\dot{s}_r\\ \dot{s}_l\end{bmatrix}=\delta_t\begin{bmatrix}1 & \frac{B}{2}\\ 1 & -\frac{B}{2}\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix} \quad \text{식 (31)}$$

식 (30)을 식(18)에 대입하고 외란의 영향을 받은 두 바퀴의 이동거리를 s'_r, s'_l이라고 표시하면 로봇의 상태 예측을 위한 식을 다음과 같이 유도할 수 있다.

$$\boldsymbol{p}_r(k)=\boldsymbol{p}_r(k-1)+\begin{bmatrix}\frac{s'_r+s'_l}{2}\cos\left(\phi_r(k-1)+\frac{s'_r-s'_l}{2B}\right)\\ \frac{s'_r+s'_l}{2}\sin\left(\phi_r(k-1)+\frac{s'_r-s'_l}{2B}\right)\\ \frac{s'_r-s'_l}{B}\end{bmatrix}=\boldsymbol{f}\left(\boldsymbol{p}_r(k-1),\boldsymbol{u},\delta\boldsymbol{u}\right) \quad \text{식 (32)}$$

수행 내용 / 로봇 위치 매핑하기

재료 · 자료

- 로봇 소프트웨어 사용법 안내 자료
- 로봇 소프트웨어 패키지 코드
- 로봇 매뉴얼

기기(장비 · 공구)

- 컴퓨터, 프린터
- 문서 작성용 소프트웨어
- 프로그램 개발용 소프트웨어
- 모바일 로봇(선택 사항)

안전 · 유의 사항

- 오픈 소스를 사용할 경우 라이센스를 확인해서 사용규정을 위반하지 않아야 한다.

수행 순서

1 SLAM을 사용하여 현재의 로봇 위치를 지도와 매핑한다.

1. 오픈소스 로봇 소프트웨어인 ROS를 이용하여 SLAM을 구현하기 위해 ROS와 연동되는 로봇과 센서를 준비한다.

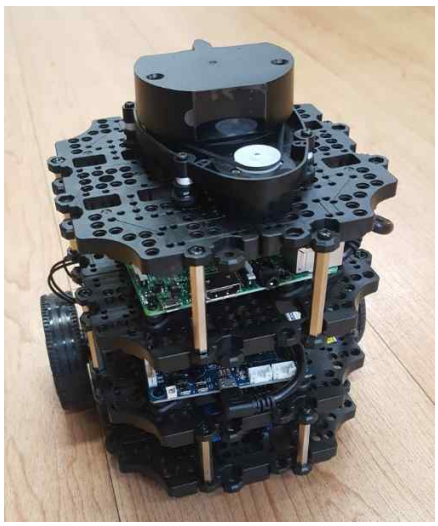
ROS를 설치하여 구동할 수 있는 로봇은 관련 위키 사이트(http://robots.ros.org/)에서 찾을 수 있다. 이 중에서 터틀봇(TurtleBot)은 ROS의 표준 하드웨어 플랫폼이며, 2017년에 터틀봇3가 개발되어 판매되었다. 터틀봇3의 자세한 정보와 내용, 예제들은 공식 위키 사이트(http://turtlebot3.robotis.com)에서 얻을 수 있다.

(1) ROS로 제어할 수 있는 대표적인 이동로봇인 로보티즈사의 터틀봇3를 조립한다.

터틀봇3는 모양에 따라 버거(Burger)와 와플(Waffle)이 있으며, 기본 구성품은 다음과 같다.

- 로봇 구동용 액추에이터
- Intel 프로세서 기반 고성능 SBC(single board computer)
- SLAM 및 네비게이션을 위한 360° LIDAR 센서(와플의 경우 RealSense 추가)
- 중간 제어기용 임베디드 보드인 OpenCR

OpenCR에는 3축 가속도 센서와 자이로 센서가 장착되어 있어 로봇의 방향이나 속도를 추정하는 데에 사용된다.



[그림 2-5] 터틀봇3 버거

(2) 터틀봇3의 ROS 패키지로는 turtlebot3, turtlebot3_msgs, turtlebot3_simulations, turtlebot3_applications가 있다. 터틀봇3의 개발환경 구축을 위해 터틀봇3를 조종하는 사용자 PC와 터틀봇3를 준비한다. ROS는 Kinetic Kame 버전을 양쪽 컴퓨터에 설치

하고, 세부 설치 방법은 http://emanual.robotis.com/docs/en/platform/turtlebot3/pc_setup/ 사이트를 참조한다.

(3) 터틀봇3를 조종하는 사용자 PC(원격 PC)에 관련 의존성 패키지 및 터틀봇3 패키지를 다음 명령줄로 설치한다.

```
$ sudo apt-get install ros-kinetic-joy ··· ros-kinetic-gmapping ros-kinetic navigation
$ cd ~/catkin_ws/src
$ git clone https://github.com/ROBOTIS-GIT/turtlebot3.git
$ git clone https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git
$ git clone https://github.com/ROBOTIS-GIT/turtlebot3_simulations
$ cd ~/catkin_ws && catkin_make
```

(4) 터틀봇3 로봇에 탑재된 SBC에 HDMI로 접속하여 다음과 같이 관련 의존성 패키지 및 터틀봇3 패키지, 센서 패키지를 설치한다.

```
$ sudo apt-get install ros-kinetic-joy ··· ros-kinetic-gmapping ros-kinetic navigation
$ cd ~/catkin_ws/src
$ git clone https://github.com/ROBOTIS-GIT/turtlebot3.git
$ git clone https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git
$ git clone https://github.com/ROBOTIS-GIT/hls_lfcd_lds_driver.git
$ cd ~/catkin_ws && catkin_make
```

(5) 사용자의 원격 PC와 터틀봇3를 무선으로 연동하기 위해 그림과 같이 네트워크 설정을 맞춰준다. 원격 PC와 터틀봇3의 IP 값을 확인할 때 ifconfig 명령을 이용한다.

출처: 표윤석(2017). 『ROS 로봇 프로그래밍』. 루비페이퍼. p.300.
[그림 2-6] 터틀봇3의 원격 제어 설정

무선 조종의 주체는 원격 PC이므로 양쪽 PC의 ROS_MASTER_URI는 동일하게 원격 PC의 IP주소로 설정해야 한다.

원격 PC에서 터틀봇3 SBC에 접속하기 위해 다음과 같이 SSH(Secure Shell)를 사용한다. SSH는 리눅스에서 다른 컴퓨터를 터미널 창에서 접속하여 원격 명령을 내릴 때 많이 사용된다. SSH의 사용법은 '사용자명@랩톱 IP' 이다.

```
$ sudo apt-get install ssh
$ ssh turtlebot@192.168.7.200
```

(6) 무선 네트워크를 통한 연결이 잘 되었는지를 확인하기 위해 다음 명령을 실행하여 원격 PC에서 터틀봇3 로봇을 조종한다.

```
$ roscore
```

터틀봇3 로봇에서 다음과 같이 turtlebot3_robot.launch 파일을 실행한다.

```
$ roslaunch turtlebot3_bringup turtlebot3_robot.launch --screen
```

원격 조종을 위해 원격 PC에서 다음과 같이 turtlebot3_teleop_key.launch 파일을 실행한다. 모든 과정이 잘 되었으면 키보드의 w, a, d, x 키로 병진속도와 회전속도를 변화시키며 로봇을 동작시킬 수 있다.

```
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch --screen
```

2. 로봇이 없는 경우 ROS의 3차원 시각화 도구인 RViz나 3차원 로봇 시뮬레이터인 Gazebo를 이용하여 터틀봇3를 시뮬레이션한다.

참고로 RViz와 달리 Gazebo에는 물리엔진이 포함되어 있기 때문에 각종 센서 정보를 가상으로 사용할 수 있다. 시뮬레이션을 위해 turtlebot3_simulations 메타패키지를 이용해야 하며, 이 패키지를 이용하려면 turtlebot3_fake 패키지를 설치해야 한다.

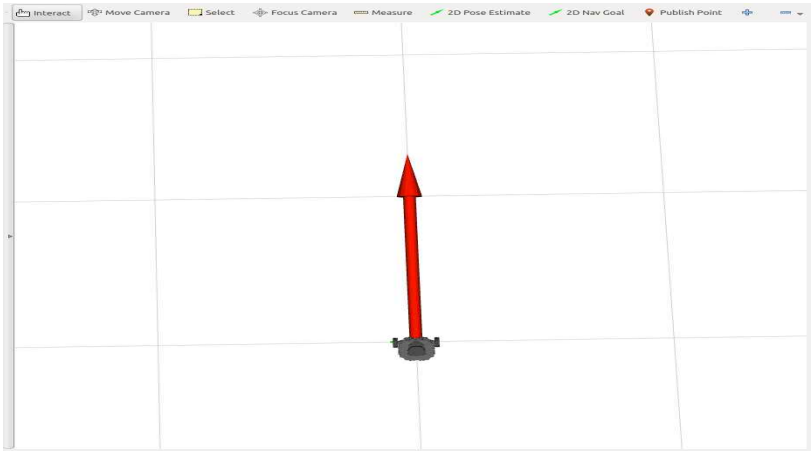
(1) RViz가 설치되어 있지 않다면 다음 명령으로 RViz를 설치하고 실행한다.

```
$ sudo apt-get install ros-kinetic-rviz
$ rviz
```

가상 로봇을 실행하기 위해 다음과 같이 turtlebot3_fake 패키지의 turtlebot3_fake.launch 파일을 실행한다.

```
$ export TURTLEBOT3_MODEL=burger
$ roslaunch turtlebot3_fake turtlebot3_fake.launch
```

그러면 [그림 2-7]처럼 터틀봇3의 3차원 모델이 중앙에 표시된다.



[그림 2-7] RViz에서 터틀봇3의 시뮬레이션

다음 명령으로 RViz 가상 환경에서 키보드의 w, a, d, x 키를 이용하여 로봇을 동작시킬 수 있다.

```
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

(2) 다음 명령으로 Gazebo를 실행한다.

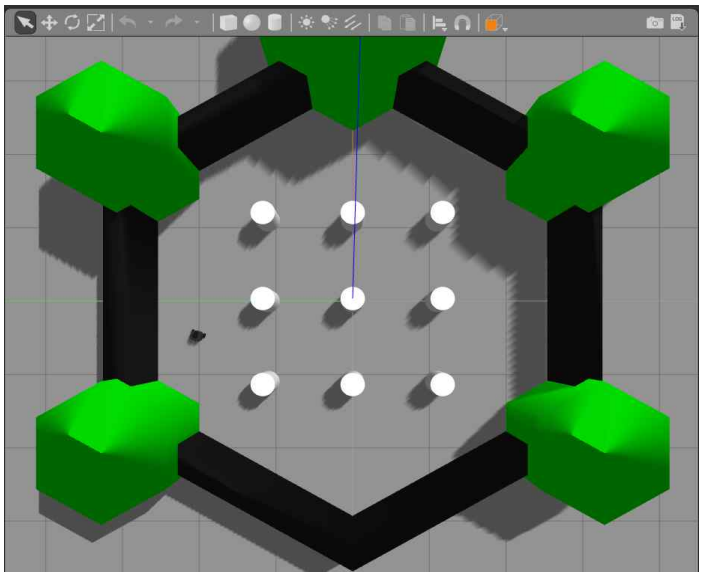
```
$ gazebo
```

그 다음에 아래와 같이 새 터미널을 열고 론치 파일을 실행한다.

```
$ export TURTLEBOT3_MODEL=burger
$ roslaunch turtlebot3_gazebo turtlebot3_world.launch
```

RViz에서 한 것처럼 turtlebot3_teleop_key 론치 파일을 실행해서 키보드로 터틀봇3를 가상 환경에서 움직일 수 있고, 아래와 같이 새 터미널을 열고 launch 파일을 실행하면 터틀봇3는 몸체의 충돌을 체크하고 위치를 계측하고 각종 센서를 사용하면서 랜덤하게 이동하며 장애물을 감지하거나 회피한다.

```
$ export TURTLEBOT3_MODEL=burger
$ roslaunch turtlebot3_gazebo turtlebot3_simulation.launch
```



[그림 2-8] Gazebo에서 터틀봇3의 시뮬레이션

3. 다음 순서대로 실행함으로써 로봇을 가상 환경에서 원격으로 조종하며 환경 지도를 작성한다.

(1) roscore 실행하여 로봇을 구동한다.

```
$ roscore
```

```
$ roslaunch turtlebot3_bringup turtlebot3_robot.launch
```

(2) Gazebo를 실행한다.

```
$ export TURTLEBOT3_MODEL=burger
$ roslaunch turtlebot3_gazebo turtlebot3_world.launch
```

(3) SLAM을 실행한다.

turtlebot3_slam 론치 파일을 실행하면 robot_state_publisher 노드(양 바퀴 및 각 조인트들의 3차원 포즈 정보를 TF로 퍼블리시)와 slam_mapping(지도 작성) 노드가 함께 실행된다.

```
$ export TURTLEBOT3_MODEL=burger
$ roslaunch turtlebot3_slam turtlebot3_slam.launch
```

(4) RViz를 실행한다.

```
$ export TURTLEBOT3_MODEL=burger
$ rosrun rviz rviz -d ‘rospack find turtlebot3_slam’ /rviz/turtlebot3_slam.rviz
```

(5) 터틀봇3를 원격 조종한다.

다음 명령을 입력하여 키보드 방향키로 로봇을 조종하며 지도 작성을 시작한다. 로봇이 이동하면 로봇의 오도메트리, 좌표계 변환(tf) 정보, 센서의 스캔 정보를 기반으로 지도가 작성되며, 지도 그림을 RViz에서 확인할 수 있다.

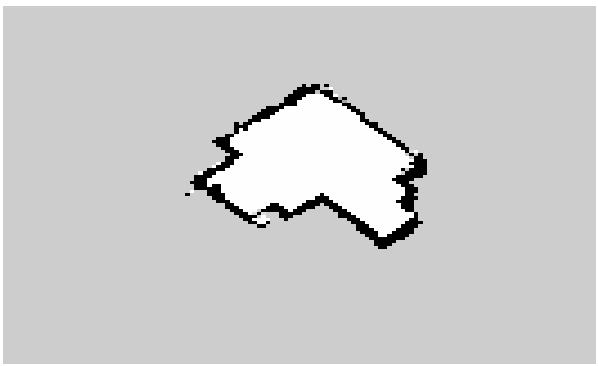
```
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

(6) 지도를 출력한다.

지도 작성 작업이 완료되면 다음 명령을 입력하여 작성한 지도를 저장한다. 그러면 지도 데이터가 담긴 map.pgm 파일과 지도 정보가 담긴 map.yaml 파일이 map_saver를 동작시킨 디렉토리에 저장된다. ‘~/’은 사용자 폴더를 의미하며, 이 폴더에 map.pgm, map.yaml의 이름으로 저장하라는 의미이다.

```
$ rosrun map_server map_saver -f ~/map
```

(7) 완성된 지도를 확인한다.



출처: RViz. 스크린샷.
[그림 2-9] 터틀봇3를 움직이며 SLAM 기능으로 생성한 지도

학습 3 이동 경로 계획 구현하기

3-1. 이동 경로 계획 구현

학습 목표

- 현재 인식된 위치를 기반으로 전역적인 로봇이동 경로를 생성할 수 있다.
- 지역적인 로봇이동 경로를 생성할 수 있다.

필요 지식 /

1 현재 인식된 위치를 기반으로 전역적 로봇이동 경로 생성

이동로봇의 경로 계획(path planning)이란 로봇이 목표 지점까지 벽 같은 구조물이나 장애물과의 충돌 없이 주어진 조건(최단 거리나 최소 시간 등)을 만족하며 도착할 수 있도록 로봇의 2차원 경로를 생성하는 기법을 의미한다.

이동로봇을 위한 경로를 생성할 때 다음의 사항들을 고려해야 한다.

- 이동로봇의 경로가 장애물과의 충돌을 발생시킬 가능성이 있는가?
- 이동로봇의 경로가 최단 경로인가?
- 얼마나 빠른 시간 내에 효율적으로 경로를 생성할 수 있는가?

장애물에는 가구 같은 정적인 장애물과 사람이나 동물 같은 동적인 장애물이 있다. 만약 환경에서 정적인 장애물만 있다면 출발점에서 목표 지점까지 전역적 이동 경로를 생성해서 그대로 따라가면 되지만, 동적인 장애물이 있을 경우에는 로봇이 장애물을 만날 때마다 전역적 경로를 수정한 지역적 이동 경로를 생성해서 충돌을 회피하며 목표 지점까지 이동해야 한다. 계층적 계획(hierarchical planning)법은 이렇게 로봇이 전역적 이동 경로와 지역적 이동 경로를 함께 만들어서 안정적이면서도 효율적으로 이동하는 방법이다.

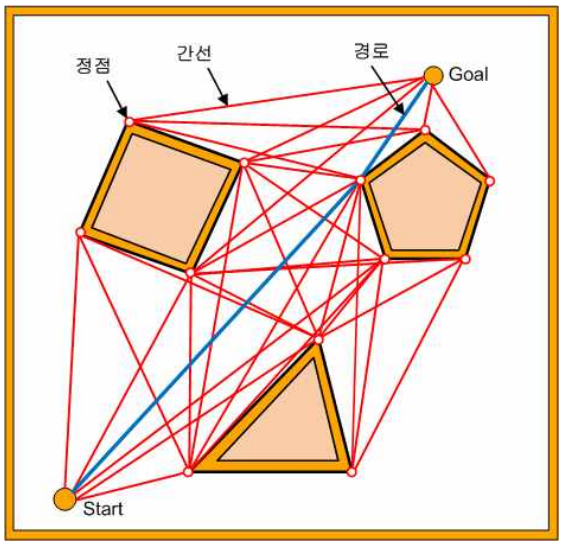
경로 생성 방법은 크게 결정론적 탐색(deterministic search)과 확률적 탐색(probabilistic search)으로 구분한다. 결정론적 탐색의 한 예는 LIKHACHEV(2009)가 환경을 격자(lattice)로 표현해서 그래프 탐색(graph search) 기법을 적용한 것인데, 이 방법의 단점은 환경을 격자로 이산화(discretization)할 때 어느 정도의 해상도가 적절한지 정하기가 어렵다는 것이다. 아래에서 설명되는 셀 분할(cell decomposition) 기반 경로 계획이 결정론적 탐색이라고 할 수 있다.

확률적 탐색에는 PRM(probabilistic roadmaps) 방법과 RRT(rapidly-exploring random trees) 방법이 있다. PRM 경로 생성 알고리즘은 오프라인 로드맵 구성 단계와 온라인 그래프 검색 단계로 구성된다. 먼저 랜덤 포인트를 샘플링하고 로컬 플래닝을 사용하여 인접한 샘플 포인트들을 연결함으로써 환경 공간의 로드맵을 구성한 다음, 할당된 출발점과 목표 지점에 그래프 탐색 기법을 적용하여 전체적 경로를 생성한다. PRM의 단점은 환경 공간이 빠르게 변하는 동적특성을 가질 경우 미리 로드맵을 구성하는 것이 쉽지 않다는 사실이다. RRT는 아래에서 설명한다.

1. 가시성 그래프를 이용한 경로 계획

가시성 그래프란 환경 공간에서 출발 지점과 목표 지점, 그리고 다각형으로 근사화된 장애물의 꼭지점들을 그래프의 정점(vertex)으로 설정했을 때, 모든 가능한 정점들의 연결 중 중간에 어떤 장애물도 존재하지 않는 가시선(visible edge)들이 존재하는데 이 가시선들을 모두 모아 놓은 그래프를 의미한다(SIEGWART and NOURBAKHSH, 2004).

이 그래프에 A* 알고리즘 같은 최적 경로 탐색 알고리즘을 적용하여 출발 지점으로부터 목표 지점까지 가장 짧은 가시선의 조합을 찾아내면 이것이 바로 이동로봇의 최단 거리 경로가 된다. 가시성 그래프를 이용한 방법의 단점은 생성된 경로가 장애물들의 꼭지점을 지나가게 되어 있으므로 로봇이 이동 시 장애물과 충돌할 확률이 높다는 것이다. [그림 3-1]은 가시성 그래프에서 최적 경로 탐색 알고리즘을 이용해서 최단 경로를 선택한 것을 보인다.

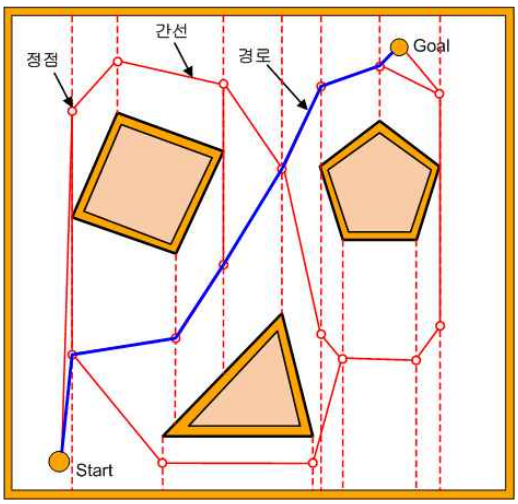


[그림 3-1] 가시성 그래프에서 최적 경로 탐색 알고리즘으로 선택한 최단 경로

2. 셀 분할 기반 경로 계획

셀 분할 기반 경로 계획은 환경 공간을 특정 규칙을 사용하여 여러 개의 셀들로 분할한 후 각 셀의 대표점(셀의 중심점 또는 셀 간 경계선의 중심점)을 연결성 그래프(connectivity graph)의 정점으로, 이웃한 셀들 간의 관계를 연결성 그래프의 에지로 표현한 다음, 출발 지점과 목표 지점이 속해 있는 셀을 포함하여 최단 경로를 생성하는 방법이다(LATOMBE, 1991).

가장 많이 사용되는 수직선 기반 셀 분할 방법은 환경 공간에 존재하는 다각형 장애물들의 꼭지점으로부터 지도의 위쪽 또는 아래쪽의 인접 경계선까지 수선을 그어 사다리꼴 모양의 볼록한(convex) 다각형 셀들을 만든다. 그리고 모든 셀의 수직 경계선의 중심점을 연결성 그래프의 정점(각 셀의 대표점)으로 잡고, 모든 인접한 정점 간에 만들어지는 연결성 그래프 에지에 두 노드 간의 거리를 가중치로 넣는다. 마지막으로 동일한 원리로 출발 지점과 목표 지점을 연결성 그래프의 정점과 에지에 편입한 다음 그래프 탐색 방법을 이용해서 최단 거리의 경로를 결정한다. 셀 분할 기반 경로 계획의 단점은 장애물의 배치에 따라 셀의 대표점 위치의 변동이 심할 수 있고, 이 경우 로봇의 경로가 매끄럽게 연결되지 않아 로봇이 지그재그 형태로 이동할 수 있다는 것이다.



[그림 3-2] 수직선 기반 셀 분할 방법에서 최적 경로 탐색 알고리즘을 이용해서 얻은 경로

3. 포텐셜 필드 기반 경로 계획

포텐셜 필드(potential field)는 위치에너지장을 의미하며, 포텐셜 필드를 이용한 이동로봇의 경로 계획에서는 환경 공간에 가상의 인력위치에너지(attractive potential)와 척력위치에너지(repulsive potential)의 합력으로 인해 에너지장(energy field)이 형성되고, 전체 위치에너지가 감소하는 방향으로 로봇이 이동하면 부드러우면서도 자연스러운 경로가 만들어지는 원리를 이용한다(SIEGWART, 2004).

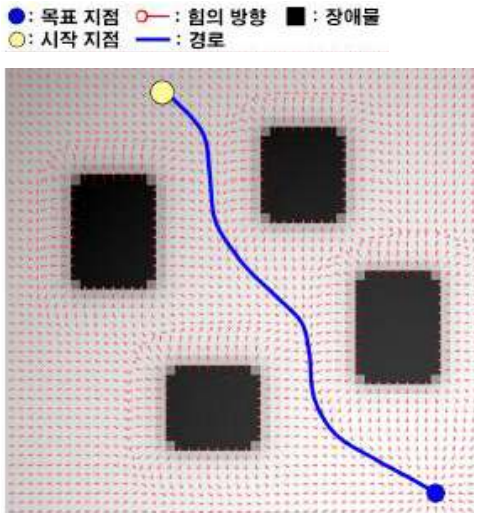
이동로봇의 자세를 p라고 하면 인력위치에너지 $U_{att}(p)$의 경우 목표 지점으로부터 멀어

질수록 그 값이 커지고 가까워질수록 0에 가깝게 작아져서 목표 지점이 이동로봇을 끌어당기는 역할을 한다. 반면에 척력위치에너지 $U_{rep}(p)$는 이동로봇과 장애물 간의 충돌을 피하기 위해 이동로봇이 특정 거리 이내에 위치한 모든 장애물과 가까워질수록 그 값이 커지고, 멀어질수록 그 값이 작아진다. 그리고 전체 위치에너지장 U은 $U_{att}(p)$와 $U_{rep}(p)$의 합이다.

이동로봇을 하나의 구체라고 가정하고 구체는 중력에 의해 위치에너지가 높은 곳에서 낮은 곳으로 굴러가는 원리를 적용하면, 포텐셜 필드 기반 경로 계획의 원리를 이해하기가 쉬울 것이다. 이때 위치에너지의 차이에 의해 에너지가 낮은 방향으로 구체(로봇)에 가해지는 가상의 힘은 다음의 관계식으로 표현된다.

$$F(p) = -\nabla U \qquad \text{식 (33)}$$

이동로봇의 최적 경로는 포텐셜 필드에서 이렇게 계산 된 가상의 힘 $F(p)$를 따라가는 방향으로 정하면 된다. 이 방법의 단점은 포텐셜 필드의 지역해(local minimum)가 존재할 경우 로봇이 여기에서 멈출 수도 있다는 사실이며, 이 경우 강제적으로 로봇을 지역해로부터 탈출(escape)시키는 알고리즘을 추가해야 한다.

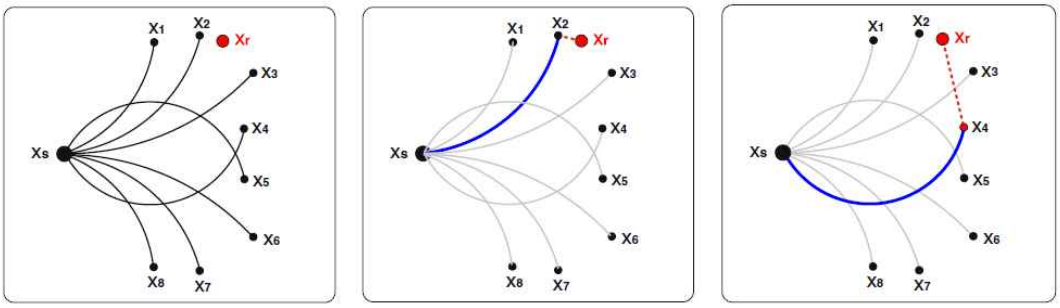


출처: 정완균・도낙주・이수용 외(2012).『실험로보틱스 Ⅱ, 이동로봇』. 한국로봇학회・제어로봇시스템학회・한국로봇산업진흥원. p.26.
[그림 3-3] 포텐셜 필드 기반으로 경로 계획을 수행한 예

4. RRT

RRT는 검색 공간에서 무작위 샘플을 사용하여 시작 구성 트리를 만든다. 각 샘플이 그려지면 트리와 가장 가까운 상태 사이에서 연결이 시도된다. 빈 공간을 완전히 통과하

고 제약 조건을 준수하면 연결이 가능하며, 그 결과 트리에 새 상태가 추가된다. RRT는 특정 지역에서 표본 추출의 확률을 높임으로써 트리를 그 방향으로 확장할 수 있는데, 이 성질을 이용하여 목표 지점에 대한 탐색을 유도한다. 이 확률이 높을수록 트리가 목표 쪽으로 가지를 밀집시키며 확장된다.



출처: Yang, K., Moon, S., Yoo, S., Kang, J., Doh, N.L., & Joo, S. (2014). Spline-based RRT path planner for non-holonomic robots. Journal of Intelligent Robot Systems, 73, 763-782.
[그림 3-4] RRT에서 가장 가까운 이웃점 선택 예

RRT에서 가장 가까운 위치에 이웃한 두 상태 간의 거리를 계산할 때 주의할 점이 있다. 만약 이동로봇이 전방향(omni-directional) 바퀴를 갖춰서 현재 포즈에서 어느 방향과 자세로도 이동할 수 있는 홀로노믹(holonomic) 시스템이라면 유클리디안 거리를 거리값으로 계산할 수 있다. 이 경우 [그림 3-4]의 가운데 그림처럼 x_r과 가장 가까운 상태는 x_2가 된다. 하지만 이동로봇이 홀로노믹이 아닌 경우(nonholonomic) [그림 3-4]의 오른쪽 그림처럼 x_s에서의 접근 방향도 고려하여 x_4 상태가 x_r에 가장 가깝다고 본다.

2 지역적 로봇이동 경로 생성

이동로봇은 주행 중 인식되는 장애물에 대한 적절한 경로를 스스로 생성하여 이동할 수 있어야 한다. 지역적인 이동 경로 생성 방법에는 크게 DWA(dynamic window approach), EB(elastic band), VFH(vector field histogram) 등이 있으며 센서 기반으로 주변 장애물을 실시간으로 인식하여 목표점까지 이동한다. 지역경로 계획 방법은 새로운 장애물 발견 시 실시간 처리가 가능하고, 로봇의 동역학 특성을 고려할 수 있는 장점이 있지만, 지역적인 제한으로 인해 지역최소문제에 빠져 목표점에 도달하지 못하는 경우가 발생할 수 있다

1. DWA

DWA는 사람들이 많이 오가는 공공장소에서 사용될 수 있는 온라인 충돌회피(collision avoidance) 알고리즘이다. DWA는 로봇의 동역학으로부터 직접적으로 도출되었으므로 로봇의 제한된 속도와 가속도에 의한 구속조건을 잘 만족시키도록 설계되었다.

DWA는 두 단계로 구성되어 있는데, 첫째는 유효한 탐색 공간을 생성하는 것이고, 둘째는 탐색 공간에서 최적해를 선택하는 것이다. FOX(1997)에 의하면 탐색 공간은 안전한 원형

궤적이며 로봇이 짧은 시간 안에 도달할 수 있어야 하며, 충돌 물체가 없어야 한다. 최적화 목표는 로봇이 어떤 장애물과도 충돌 없이 목표 지점에 도착하게 하는 방향각과 속도를 선택하는 것이다.

이를 위해 DWA 알고리즘은 다음과 같은 단계를 거친다.

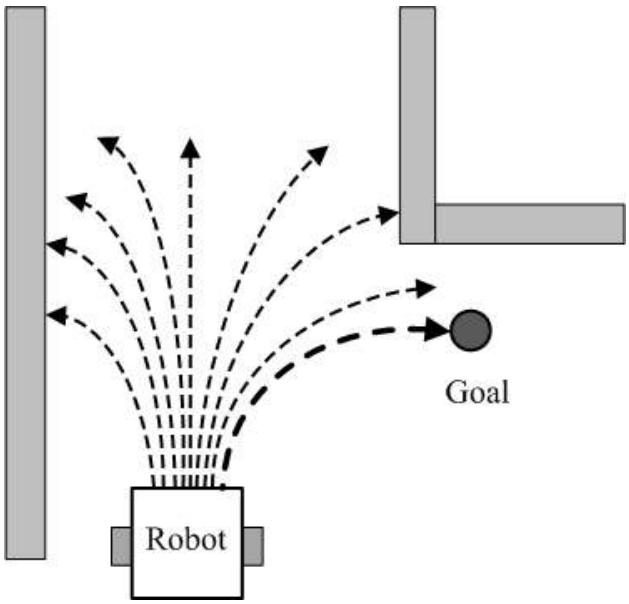
1단계: 로봇의 제어 공간(병진속도 v와 회전속도 ω를 축으로 하는 영역)을 이산시간(discrete time) 샘플링한다.

2단계: 각 샘플 된 병진·회전 속도에 대해 로봇이 현재 상태로부터 그 값을 입력으로 받아 단기간 전진하면 어떻게 되는지를 시뮬레이션한다.

3단계: 각 시뮬레이션 궤적에 대해 장애물과의 접근성, 목표물과의 접근성, 전역경로로부터의 이탈 정도, 로봇 속도 등을 고려하여 목적함수를 계산한다. 이 과정에서 조건 위반성(장애물과 충돌)이 있는 경우 해당 궤적은 제외한다.

4단계: 가능한 시뮬레이션 궤적 중 목적함수 값이 최대가 되게 하는 병진속도 v와 회전속도 ω를 선택한다.

[그림 3-5]는 DWA가 목표 지점에 도달하는 궤적을 선택한 예를 보인다.



[그림 3-5] DWA가 목표 지점에 도달하게 하는 최적 병진속도와 회전속도를 찾는 과정

2. 경로 추종 제어

경로 추종 제어란 이동로봇이 출발 지점에서부터 목표 지점까지 이동 시 목표 경로와의 오차를 최소화하며 잘 따라가도록 로봇의 해당 액추에이터(모바일 로봇의 경우 바퀴, 족형 로봇(legged robot)의 경우 하지관절모터)를 제어하는 기술을 의미한다. 족형 로봇의 경우 목표 보행 경로가 주어졌을 때 자세 안정도를 유지하면서 이 경로를 추종할 수 있도록

각 하지관절모터의 회전각 궤적을 추가적으로 생성해야 하는데, 이를 궤적 생성(trajectory generation)이라고 한다. 김종욱(2015)은 이족 휴머노이드 로봇의 전방향(omni-directional) 이족 보행을 위한 하지 관절 궤적 생성의 원리를 오픈소스 코드와 함께 체계적으로 설명했다.

(1) PID제어기를 통한 이동로봇의 경로 추종

PID제어기는 구조가 간단하고 제어 성능이 우수하므로 산업 현장에서 대부분의 이상의 제어시스템에 적용되고 있다. PID제어기의 입력은 제어 목표값(reference value)과 시스템 출력값(output value) 간의 차이 값인 오차(error)이며, P(proportional) 제어기는 오차에 이득(gain) K_P를 곱한 값을, I(integral) 제어기는 오차를 일정 시간 적분한 값에 이득 K_I를 곱한 값을, D(derivative) 제어기는 오차를 미분한 값에 이득 K_D를 곱한 값을 제어기의 출력으로 보낸다(한수희 · 이영삼 · 권보규 외, 2013).

PID제어기의 출력이자 대상 시스템(이동로봇)의 입력 신호가 되는 $u(t)$를 연속 시간 시스템의 수식으로 표현하면 다음과 같다.

$$u(t) = K_P e(t) + K_I \int e(\tau) d\tau + K_D \frac{de(t)}{dt} \qquad \text{식 (34)}$$

이동로봇의 이산시간(discrete time) 상태방정식이 식 (18)과 같이 주어져 있고, k번째 샘플시간에 목표 경로 지점의 상태벡터가 $[x_r(k)\, y_r(k)\, \phi_r(k)]^T$이고, 이때 로봇의 추정된 상태벡터를 $[x(k) y(k) \phi(k)]^T$라고 하면, 로봇의 오차벡터는 다음과 같이 정의된다.

$$\begin{bmatrix} x_e(k) \\ y_e(k) \\ \phi_e(k) \end{bmatrix} = \begin{bmatrix} x_r(k) \\ y_r(k) \\ \phi_r(k) \end{bmatrix} - \begin{bmatrix} x(k) \\ y(k) \\ \phi(k) \end{bmatrix} \qquad \text{식 (35)}$$

이 오차를 최소화하기 위해 PID제어기를 적용하면 로봇 중심점의 x축 속도와 y축 속도, 각속도의 제어식은 다음과 같이 설계할 수 있다.

$$v_x(k) = \frac{1}{\delta_t}\left(x_r(k+1) - x_r(k) + K_{P_x} x_e(k) + K_{D_x}(x_e(k) - x_e(k-1)) + K_{I_x} \sum_{i=0}^{k} x_e(i) \right)$$

$$v_y(k) = \frac{1}{\delta_t}\left(y_r(k+1) - y_r(k) + K_{P_y} y_e(k) + K_{D_y}(y_e(k) - y_e(k-1)) + K_{I_y} \sum_{i=0}^{k} y_e(i) \right)$$

$$\omega(k)=\frac{1}{\delta_t}\left(\omega_r(k+1)-\omega_r(k)+K_{P_\omega}\omega_e(k)+K_{D_\omega}(\omega_e(k)-\omega_e(k-1))+K_{I_\omega}\sum_{i=0}^{k}\omega_e(i)\right)$$

식 (36)

여기서 K_P, K_I, K_D는 PID제어기의 비례, 적분, 미분 제어 이득으로 상수 값들이다. PID 제어기의 이득값을 조절하는 방법은 지글러(ZIEGLER)와 니콜스(NICHOLS)가 1942년에 제안한 동조법(ZN법)이 있는데, 제어기 동조의 초기값으로 설정하기에 유용하다(ZIEGLER and NICHOLS, 1942). ZN법의 주파수 응답법은 비례 제어기만을 공정에 연결해서 이득 K_P를 서서히 증가시키다가 출력이 진동(oscillation)하기 시작하면(임계 안정상태) 멈추고, 그 때의 이득값 K_u와 진동 주기 T_u를 측정해서 이 값들로부터 제어기 이득을 결정하는 방법이다. <표 1-1>은 PID 제어기 식 (34)를 다음과 같이 표현할 때 이득값 K, 시간상수 T_d와 T_i가 K_u 및 T_u로부터 어떻게 정해지는지를 보여주는 ZN법 이득 조절표이다.

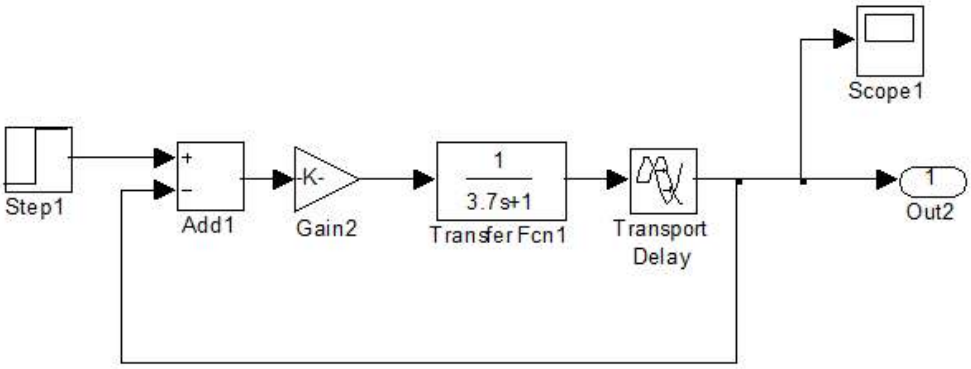
$$u(t)=K\left(e(t)+\frac{1}{T_i}\int_0^t e(\tau)\,d\tau+T_d\frac{de(t)}{dt}\right)$$

식 (37)

<표 1-1> 지글러-니콜스 주파수 응답법의 PID 파라미터

제어기 종류	K	T_i	T_d
P 제어기	$0.5K_u$	-	-
PI 제어기	$0.4K_u$	$0.8T_u$	-
PID 제어기	$0.6K_u$	$0.5T_u$	$0.125T_u$

[그림 3-6]은 PID 제어기 설계 시뮬레이션을 위해 Matlab SIMULINK 모델 파일을 구성한 화면이며, 입력은 단위 계단함수(step function)을 인가했다. 비례 제어기에 해당하는 Gain 블록의 이득을 2.84까지 증가시키자 [그림 3-7]과 같이 출력 값에 진동이 발생한다. 이득을 그 이상으로 증가시키면 계단 응답은 무한대로 발산한다.



[그림 3-6] 페루프 시스템 시뮬레이션을 위한 SIMULINK 모델

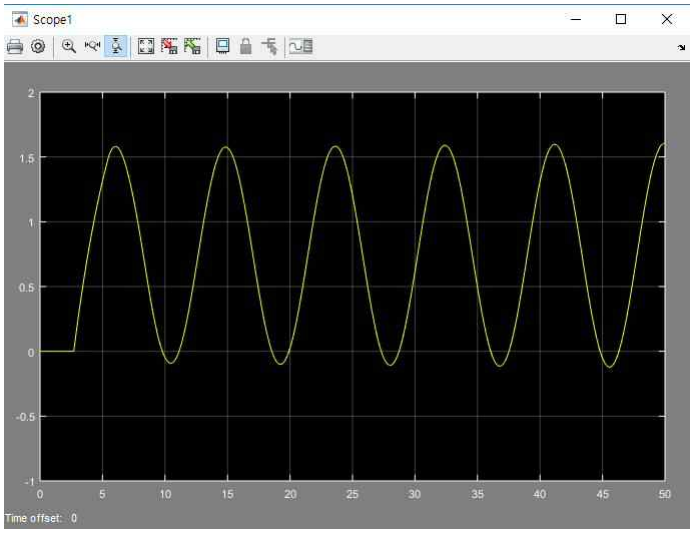
[그림 3-7]의 계단응답 파형으로부터 진동이 발생하기 시작하는 이득값 K_u는 2.84이고 진동주기 T_u는 약 8.8초임을 알 수 있다. PID 제어기에서 미분 제어기는 잡음에 민감한 문제가 있어 PI제어기를 설계하고자 한다면 <표 1-1>로부터 식 (37)의 이득값과 시간상수는 다음과 같이 정해진다.

$$K = 0.4K_u = 1.14,\ T_i = 0.8T_u = 7.04,\ \ T_d = 0 \qquad \text{식 (38)}$$

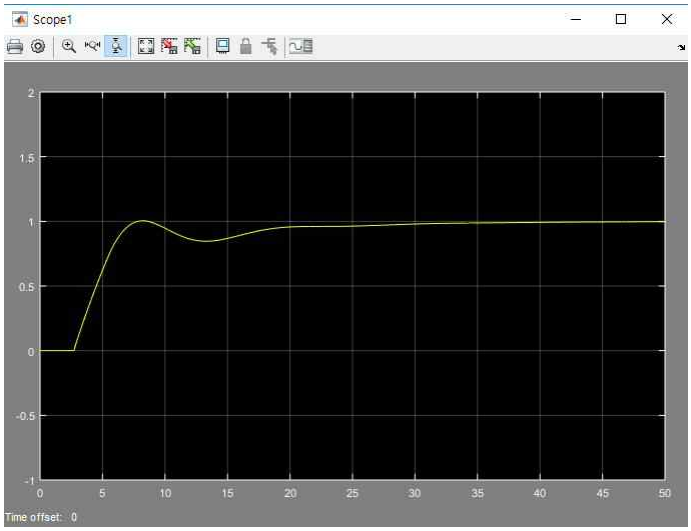
이 값들을 식 (37)에 대입하면 다음과 같은 PI 제어기가 최종적으로 설계된다.

$$u(t) = 1.14e(t) + 0.16\int_0^t e(\tau)d\tau \qquad \text{식 (39)}$$

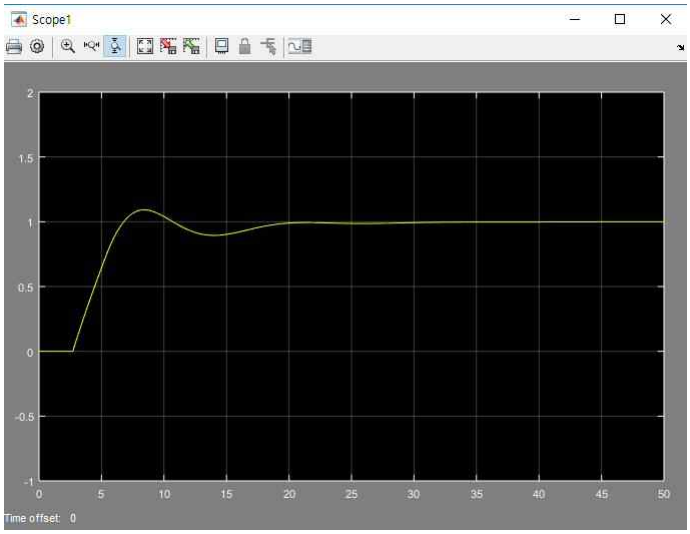
[그림 3-8]은 식 (39)로 표현된 PI제어기의 계단 응답 파형으로서 비교했을 때 목표값인 1.0을 잘 추종함을 알 수 있다. 그러나 자세히 보면 정상상태 오차가 빨리 줄어들지 않음을 알 수 있다. 그래서 적분 제어기 이득을 0.16에서 0.2로 미세하게 증가시키니 [그림 3-9]와 같이 오차가 좀 더 빨리 줄어드는 것을 볼 수 있지만, 오버슛이 0.09로 더 커졌음을 알 수 있다. 이처럼 ZN법은 PID 제어기의 이득을 설정하는 기본 방법으로 사용하고, 설계자의 필요에 따라 미세 조정(fine tuning)을 하면 원하는 제어 성능을 얻을 수 있다.



[그림 3-7] K가 2.84일 때의 계단 응답 파형



[그림 3-8] ZN법으로 동조한 PI 제어기 사용 시 출력 파형



[그림 3-9] 적분 제어기 이득을 0.2로 증가시켰을 때의 PI 제어 출력 파형

수행 내용 / 이동 경로 계획 구현하기

재료 · 자료

- 로봇 소프트웨어 사용법 소개 책자
- 로봇 소프트웨어 패키지 코드
- 로봇 매뉴얼

기기(장비 · 공구)

- 컴퓨터, 프린터
- 문서 작성용 소프트웨어
- 프로그램 개발용 소프트웨어
- 모바일 로봇(선택 사항)

안전 · 유의 사항

- 오픈 소스를 사용할 경우 라이센스를 확인해서 사용규정을 위반하지 않아야 한다.

수행 순서

1 현재 인식된 위치를 기반으로 전역적인 이동로봇 경로를 생성할 수 있다.

[그림 3-10]은 이동로봇이 목표 지점까지 가기 위한 이동지능 소프트웨어의 흐름도이다. ROS로 네비게이션을 구현하기 위해 '학습 2. 로봇 위치 매핑하기'의 수행 내용을 통해 SLAM 패키지를 다운로드하고 작업 환경의 지도 파일을 확보했다고 가정하고, 다음 단계를 거치며 전역적 이동로봇 경로를 생성한다.

1. 네비게이션 노드를 실행한다.

원격 PC에서 roscore를 실행한다.

```
$ roscore
```

터틀봇3 로봇에서 다음과 같이 turtlebot3_robot.launch 파일을 실행한다.

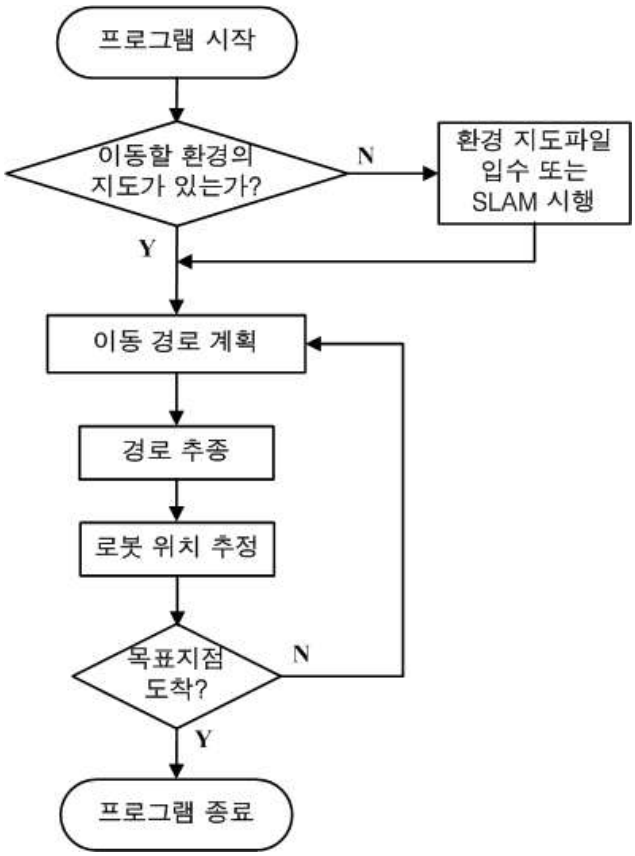
```
$ roslaunch turtlebot3_bringup turtlebot3_robot.launch
```

원격 PC에서 네비게이션 파일을 론치하여 네비게이션 패키지를 실행한다. 이 패키지에는 터틀봇3의 3차원 모델 정보, 양 바퀴 및 각 조인트들의 3차원 위치 및 방향 정보를 TF로 발행하는 robot_state_publisher 노드, 미리 작성되어 있는 지도를 불러오는 map_server 노드, AMCL(Adaptive Monte Carlo Localization) 노드, move_base 노드가 함께 실행된다.

```
$ export TURTLEBOT3_MODEL=burger
$ roslaunch turtlebot3_navigation turtlebot3_navigation.launch map_file:=$HOME/map.yaml
```

경로 생성을 위해 RViz를 다음 명령으로 실행한다.

```
$ rviz -d `rospack find turtlebot3_slam`/rviz/turtlebot3_navigation.rviz
```

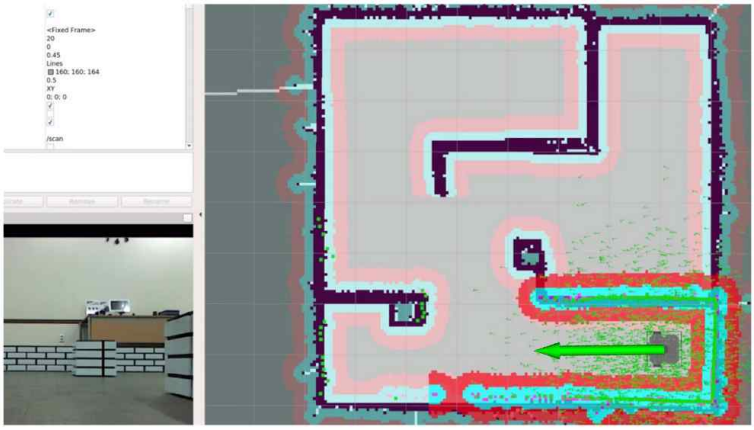


[그림 3-10] 이동지능 소프트웨어의 프로그램 흐름도

2. 초기 포즈를 추정한다.

원격 PC에서 RViz 메뉴 중 2D Pose Estimate 버튼을 누른다. 이렇게 하면 ROS 네비게이션 스택이 사용하는 localization 시스템을 초기화한다. 그 결과 큰 녹색 화살표가 나타나는데 이것을 실제 로봇이 있는 위치에 클릭하고, 로봇이 향하고 있는 방향이라는 것을 알려주기 위해 마우스 왼쪽 버튼을 누른 채 커서를 드래깅한다.

그 다음에 turtlebot3_teleop_keyboard 명령을 실행하여 주변 환경의 정보를 얻고 지도에서 로봇이 실제 위치와 방향을 확인한다. [그림 3-11]에서 보듯이 이 작업이 끝나면 녹색 화살표가 터틀봇3의 초기 포즈를 잘 나타낸다. 그림에서 작은 녹색 화살표들이 잔뜩 있는 것은 SLAM에 사용된 파티클 필터의 각 입자를 나타낸다.

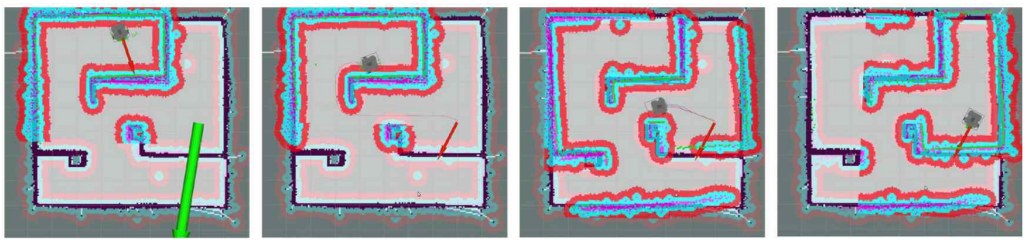


[그림 3-11] RViz에서 터틀봇3의 초기 포즈 추정

3. 목표 지점을 정한다.

원격 PC에서 RViz 메뉴 중 2D Nav Goal 버튼을 누른다. 그러면 큰 녹색 화살표가 나타나는데 이것을 목표 위치에서 클릭한다. 그 다음에 마우스 왼쪽 버튼을 누른 채 커서를 드래깅하여 로봇이 접근할 방향을 정한다.

그러면 로봇이 지도를 기반으로 장애물을 피하면서 목표 지점에 주어진 방향으로 도착하도록 전역경로를 생성한다. [그림 3-12]는 오른쪽 하단에 목표 지점의 위치와 방향을 설정했을 때 로봇이 전역경로를 생성하고 목표 지점에 도달하는 것을 보인다.



[그림 3-12] RViz에서 터틀봇3의 초기 포즈 추정

2 지역적인 이동로봇 경로를 생성한다.

DWA로 지역적 이동로봇 경로 생성을 위해 dwa_local_planner의 파라미터들을 적절히 설정한다.

```
DWAPlannerROS:

 # 로봇 파라미터 설정
 max_vel_x: 0.18            # x축 최대 속도(m/s)
 min_vel_x:-0.18            # x축 최소 속도(m/s)
 max_vel_y: 0.0             # y축 최대 속도(m/s)
 min_vel_y: 0.0             # y축 최소 속도(m/s)
 max_trans_vel: 0.01        # 최대 병진 속도(m/s)
 min_trans_vel: 0.05                # 최소 병진 속도(m/s), 음수이면 후진 가능
 max_rot_vel: 1.8           # 최대 회전 속도(rad/s)
 min_rot_vel: 0.7           # 최소 회전 속도(rad/s)
 acc_lim_x: 2.0             # x축 가속도 제한(m/s^2)
 acc_lim_y: 0.0             # y축 가속도 제한(m/s^2)
 acc_lim_theta: 2.0         # theta축 가속도 제한(rad/s^2)

 # 목표지점 허용오차
 yaw_goal_tolerance:0.15    # yaw축 목표지점 허용오차(radian)
 xy_goal_tolerance: 0.05    # x, y 거리 목표지점 허용오차(m)

 # 전방(forward) 시뮬레이션 파라미터
 sim_time: 3.5              # 전방향 시뮬레이션 궤적 시간
 vx_samples: 20             # x축 속도 공간에서 탐색하는 샘플 수
 vy_samples: 0              # y축 속도 공간에서 탐색하는 샘플 수
 vtheta_samples: 40                 # theta축 속도 공간에서 탐색하는 샘플 수

 # 궤적 평가 파라미터
# cost =
# path_distance_bias*(distance to path from the endpoint of the trajectory in m)
# + goal_distance_bias*(distance to local goal from the endpoint of the trajectory in m)
# + occdist_scale*(maximum obstacle cost along the trajectory in obstacle cost (0-254))
 path_distance_bias: 32.0       # 주어진 경로를 얼마나 잘 추종하는지에 대한 가중치
 goal_distance_bias: 24.0       # 목표 지점과 제어 속도에 근접한지에 대한 가중치
 occdist_scale: 0.04                    # 장애물 회피에 대한 가중치
 forward_point_distance: 0.325   # 로봇 중심점과 추가 평가점과의 거리
 stop_time_buffer: 0.2          # 로봇이 충돌 전 정지에 필요한 시간(sec)
 scaling_speed: 0.25            # 스케일링 속도(m/s)
 max_scaling_factor: 0.2        # 최대 스케일링 요소

 # 오실레이션 동작 방지 파라미터
 oscillation_reset_dist: 0.05       # 오실레이션 플래그가 리셋되기 전에 로봇이 움직이는 거
리
```