지능 로봇 - II
Intelligent Robots

이건명
충북대학교 소프트웨어학과

학습 내용

- 로봇 시스템에서 구성요소 간의 동신 방식에 대해서 알아본다.
- 로봇 소프트웨어 개발 프레임워크에 대해서 알아본다.
- 동시적 위치추정 및 지도작성(SLAM)에 대해서 알아본다.
- 항법(navigation) 방법에 대해서 살펴본다.

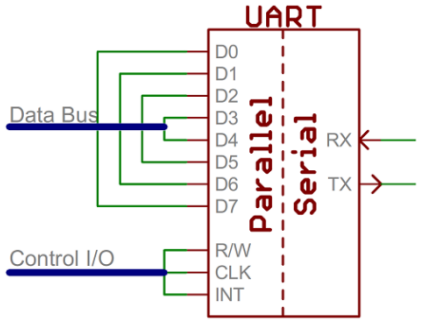
1. 요소간의 통신 방식

- **로봇 요소간의 통신 방식**
  - **하드웨어간 통신**
    - **UART**
    - **RS-232/422/485**
    - **SPI/I2C**
    - **USB**
    - …
  - **소프트웨어간 통신**
    - 공유 메모리 (shared memory)
    - 소켓 (socket)

요소간의 통신 방식

❖ **UART**(Universal Asynchronous Receiver Transmitter)

- GND(기준전압), TX(송신선), RX(수신선) 등의 통신선 이용한 **직렬통신**
  - **병렬 데이터 ↔ 직렬 데이터 변환**
  - UART 기능은 보통 MCU 등에 내장



- **비동기 방식**이기 때문에, 미리 송/수신쪽에서 **전송 속도 설정**
- **TTL level 전압** 사용(5/3.3V, 0V)
- **디지털 신호**를 직접 전달
- 근거리 통신에 사용
  - 동일한 보드 상의 MCU간의 통신 등
  - 장거리 통신에 부적합

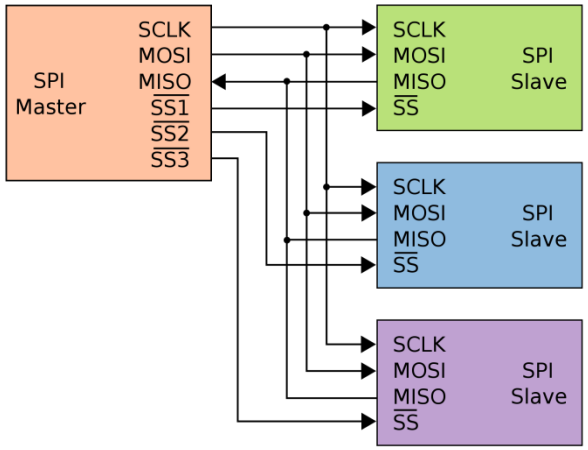
요소간의 통신 방식

- **RS-232/422/485**
  - TTL level 전압으로 통신하는 UART 통신을 보완한 방식
  - **12V/-12V** 또는 **25V/-25V** 전압으로 신호 전송
    - UART보다 먼거리까지 통신 가능
  - **TTL level 신호 ↔ RS-232 level 신호 변화 장치 사용**
    - MAX232 칩 사용

  - 사용하기 편리하나, 전송속도가 느리고, 전송 거리도 짧음
    - RS-232는 1:1 통신 지원
    - RS-422는 1:n 통신 지원
    - RS-485는 n:n 통신 지원

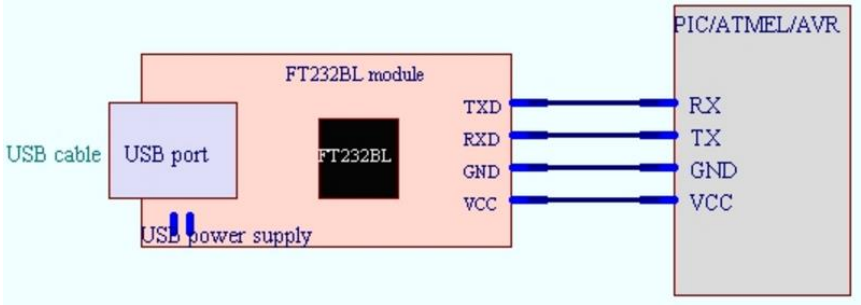
요소간의 통신 방식

- **SPI / I2C**
  - SPI (Serial Peripheral Interface)
  - I2C (Inter-Integrated Circuit)
  - **근거리 통신** 규격
  - **1:n 통신** 지원
    - 동일한 보드 내의 MCU간 연결, MCU와 센서 연결
    - MCU가 master가 되고, 센서들이 slave가 됨



요소간의 통신 방식

- **USB** (Universal Serial Bus)
  - **다양한 병렬, 직렬 방식**의 **기기 연결** 지원
  - 물리계층 규약과 프로토콜을 포함한 규격
    - USB 통신을 하기 위해서는 해당 **디바이스 드라이버**(device driver) 필요
    - KINECT 등의 카메라, 스마트기기 등이 USB 지원
  - **UART** 통신 모듈을 **USB**로 **연결** 가능
    - **변환 모듈** 사용 : FT232
    - USB를 통해 Serial Port 생성 : 별도 디바이스 드라이버 설치



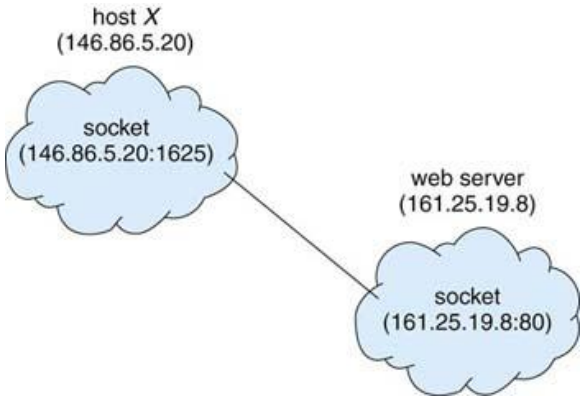
http://www.nbglin.com/ft232.htm

요소간의 통신 방식

❖ **소프트웨어 요소 간의 통신**

- 프로세스간의 통신(Inter-Process Communication, IPC) 이용
- **공유 메모리**(shared memory) 사용
  - 같은 메모리 공간에 대한 접근
  - 동일 컴퓨터 내에서만 가능

- **소켓**(socket) 통신
  - IP 주소(IP address)와 포트 번호(port number)로 식별되는 **통신 단말(endpoint)** 제공
  - 클라이언트-서버(client-server) 구조 사용



2. 로봇 제어 패러다임과 구조

❖ 로봇 제어 패러다임

- 감지(sense), 계획(plan), 행동(act) 기능의 구성 방식에 따라 구별
  - 감지 기능
    - 센서를 통해 정보를 수집하고 다른 기능 모듈에 전달
  - 계획 기능
    - 정보를 받아들여 로봇이 수행할 단일 작업 또는 일련의 작업 생성
  - 행동 기능
    - 모터 제어기 등에 출력 명령을 전달하여 행동을 수행하도록 하는 것

- **계층형 패러다임**
- **반응형 패러다임**
- **혼합형 패러다임**

❖ 로봇 제어 구조

- 제어 시스템을 구성하는 원칙적인 방법 및 제약조건 명세
- 특정 패러다임을 따르는 참조모델

계층적 패러다임

❖ 계층적 패러다임

- **감지**하고, **계획**을 **수립**하고, 이를 바탕으로 **행동**을 하는 과정의 **반복**

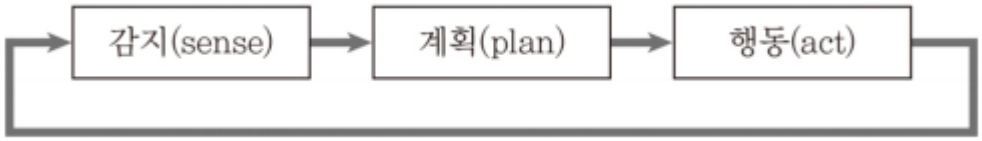


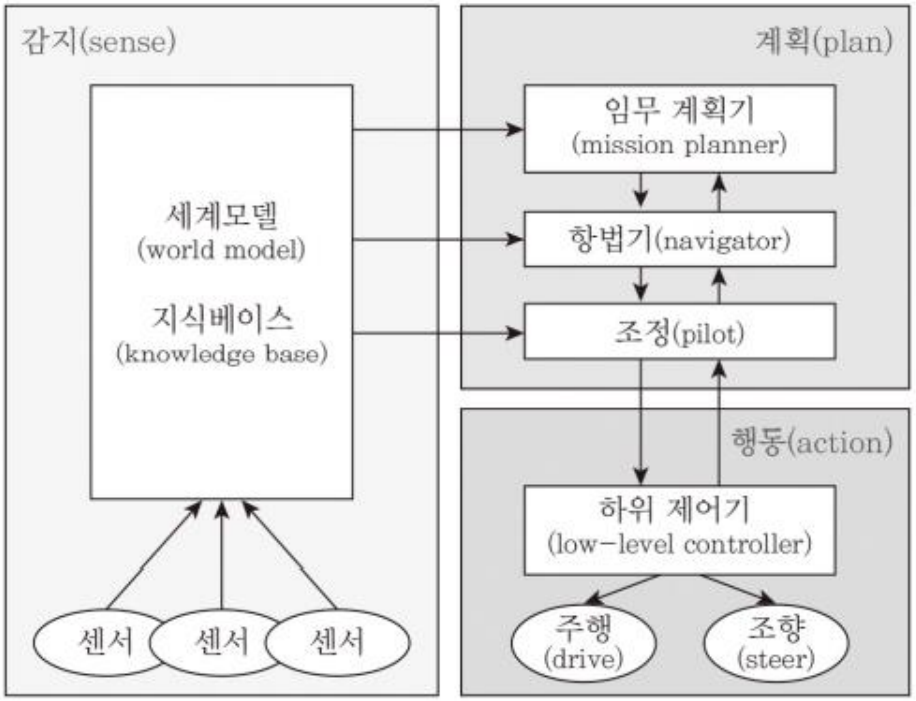
표 10.1 계층적 패러다임에서 단계별 입력 및 출력

기본 기능	입력	출력	
감지	센서 데이터	감지된 정보	로봇 및 환경에 대한 정보
계획	감지되거나 인식된 정보	지시명령 (directives)	계획수립에 따른 지시명령
행동	지시명령	조작된 명령	당장 수행할 동작의 제어기 명령어

- 대표적인 로봇 소프트웨어 구조
  - 중첩 계층 제어기 **NHC 구조**
  - NIST 실시간 제어 시스템 **RCS 구조**

계층적 패러다임

❖ 중첩 계층 제어기 NHC 구조



항법 문제에 적합

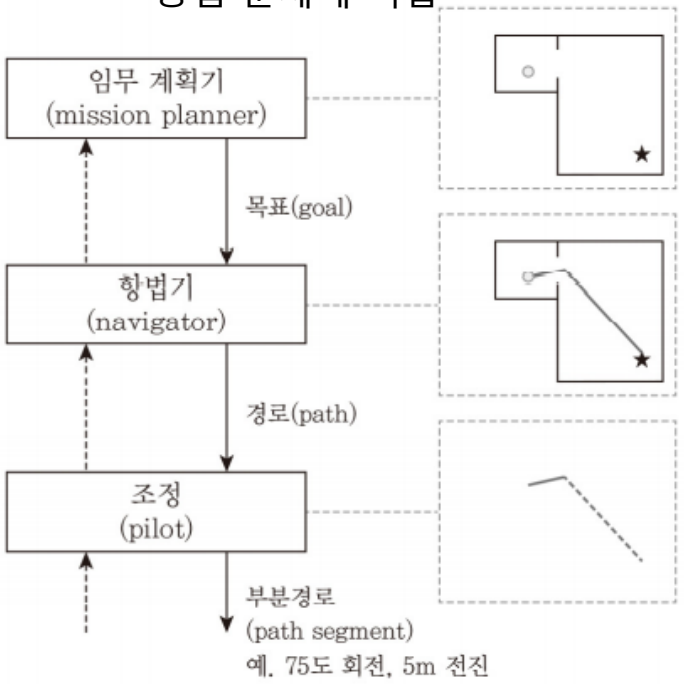
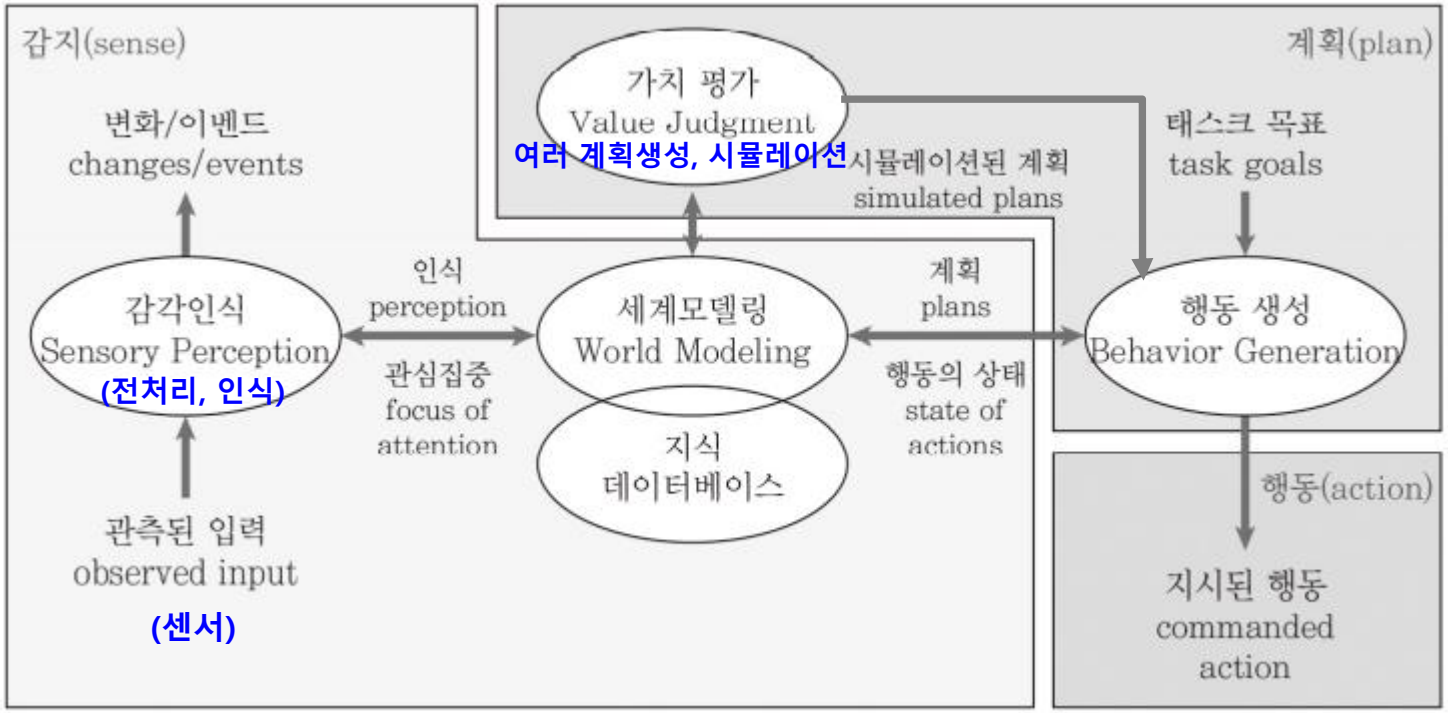


그림 10.14 NHC 구조에서 계획 모듈의 구성

계층적 패러다임

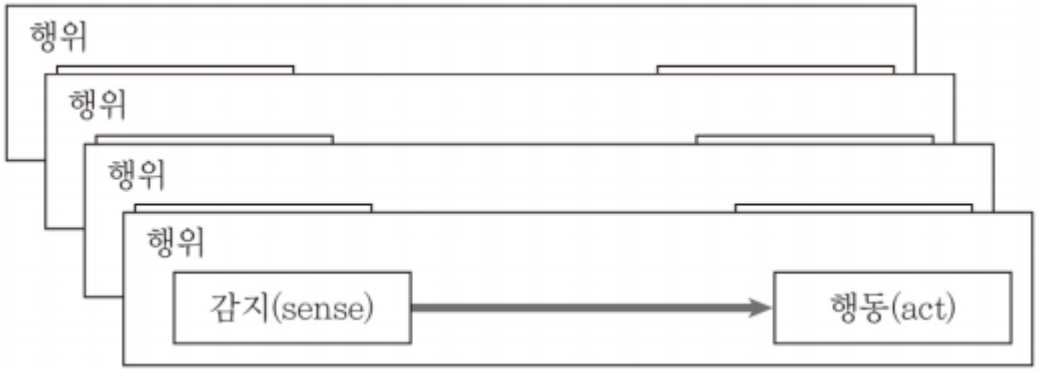
❖ NIST 실시간 제어 시스템 RCS 구조



반응형 패러다임

❖ 반응형 패러다임

- 계획수립을 하는 단계가 없이 센서를 통해 **감지된 상황별**로 바로 어떤 **행동**을 할지 대응
- **로봇 제어 프로그**램을 **병렬적인 행위들의 집합**으로 구성
  - **행위**: 감지된 상황별로 로봇이 수행할 행동을 대응시켜놓은 것



기본 기능	입력	출력
감지	센서 데이터 →	감지된 정보
행동	감지된 정보 →	조작자 명령

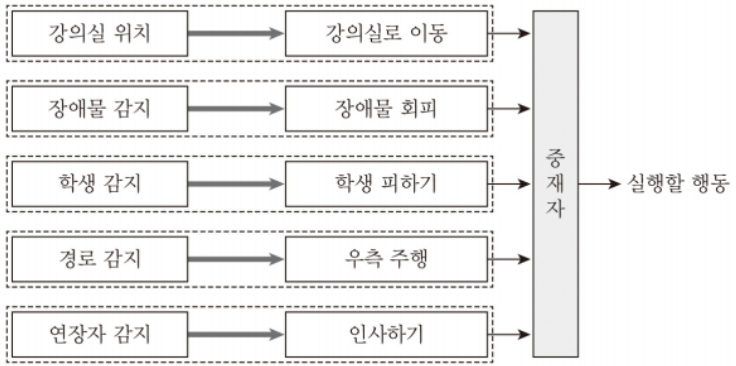
반응형 패러다임

❖ **반응형 패러다임** – cont.

- 로봇 행동에 대한 별도의 지식을 표현하지 않고, 상황별로 수행할 행동만을 지정

- **중재자**
  - 두 가지 이상의 **행위**가 실행 가능하면서 **충돌**이 생기면 **실행할 행동 결정**
  - 중재 전략
    - **경쟁적 방식** : 우선순위 부여 또는 다수결 투표
    - **협력적 방식** : 실행 가능한 두 개 이상의 동작을 동시에 사용
    - **혼합적 방식** : 상황에 따라 경쟁적 중재 방식과 협력적 중재 방식 사용

- **반응형 시스템**
  - 반응형 패러다임을 따르는 시스템
  - 중재 방식
    - 경쟁적 중재 방식인 **포섭 구조**
    - 협력적 중재 방식인 **전위장**



반응형 패러다임

❖ **포섭 구조**(subsumption architecture)

- **행위 모듈**을 **계층적인 구조**로 구성

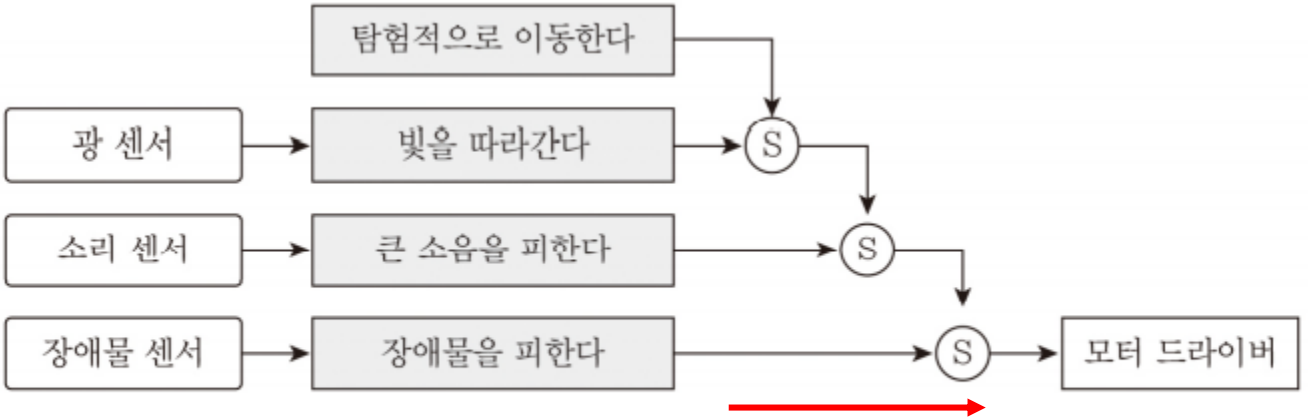


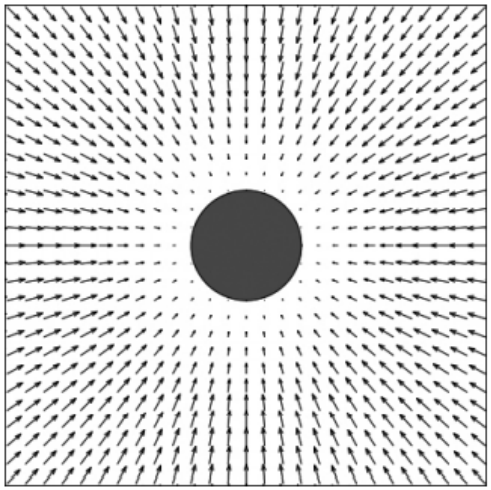
그림 10.19 **포섭구조로 표현된 반응형 시스템**

Ⓢ는 억제노드이고, 상위 계층의 모듈이 하위 계층 모듈보다 실행 우선순위가 높다.

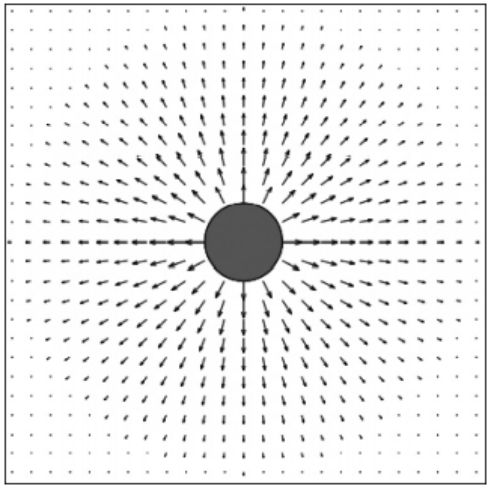
반응형 패러다임

❖ **전위장(potential field) 방법**

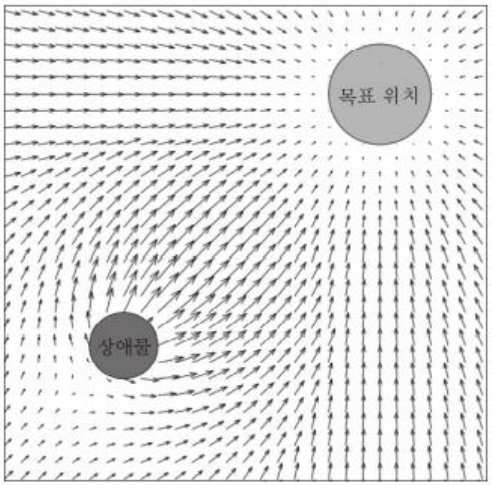
- 로봇이 위치하는 2차원 공간을 일정 크기의 그리드로 분할하여 표현
- 로봇의 이동에 **관련된 행위별**로 그리드의 **각 셀(cell)**에서의 **이동 벡터 계산**
- 각 셀에 대해서 행위별로 계산된 이동 벡터를 더하여 최종 이동 벡터를 결정



목표 위치를 향하라



장애물을 피하라



인력장과 척력장을 합성하여 얻은 벡터 필드

혼합형 패러다임

❖ **혼합형 패러다임** (hybrid paradigm)

- **우선 계획수립**을 해 놓은 다음에 이를 이용하여 반응형 패러다임처럼 **로봇 동작**
  - **계획 모듈**
    - 태스크를 완수할 수 있는 **부분 태스크**들로 분해하는 **임무 계획수립**을 한 다음, 각 부분 태스크를 수행할 **행위**를 결정
  - **로봇**
    - **실행 가능한 행위**들이 **중재**를 통해서 **병렬 수행**

표 10.3 혼합형 패러다임에서 단계별 입력 및 출력

기본 기능	입력	출력
계획	감지되거나 인식된 정보	지시명령 (directives)
감지·행동 (행위)	센서 데이터	조작된 명령

3. 로봇 제어 코드 구현

❖ **로봇 제어 코드 구현**

- 다양한 기능 모듈로 구성
- 개념적으로는 독립적으로 각각 기능

- 각 모듈이 소프트웨어적으로 **병렬**로 **실행**되고 있는 것처럼
  각 **모듈**을 **스레드**(thread)로 구현

- **단일 프로세스**를 사용하여 운영체제에서 **시분할 처리하는 것처럼**
  모듈을 일정시간씩 돌아가면서 처리

4. 로봇 소프트웨어 개발 프레임워크

❖ 로봇 소프트웨어 개발 프레임워크

- 배경
  - 로봇은 다양한 기능을 사용하여 서비스 구현
  - 다양한 신규 서비스 개발
  - 소프트웨어의 **모듈화**(modularization)와 **재사용성**(reusability) 관심
  - 소프트웨어 모듈간의 **통신** 문제

- **미들웨어**(middleware)
  - **소프트웨어 컴포넌트**나 **응용 프로그램**이 쉽게 **통신**할 수 있도록 하는 소프트웨어
  - 로봇 소프트웨어 개발에 미들웨어를 제공하여 소프트웨어의 통신 기능 제공

- **개발 프레임워크**(framework)
  - 미들웨어 + 각종 개발 도구 + 라이브러리 제공
  - 로봇 소프트웨어 개발 생산성 향상
    - 기존 개발된 것을 최대한 사용하며 자신의 관심분야에 집중

로봇 소프트웨어 개발 프레임워크

❖ **로봇 소프트웨어 개발 프레임워크**

- **ROS** (Robot Operating System)
  - www.ros.org
  - 가장 많이 사용되고 있는 프레임워크
- MSROS (Microsoft Robot Development Studio)
- OPRoS (Open Platform for Robotic Service)
  - 우리나라에서 만든 프레임워크
- OpenRTM
  - 일본 주도
- OROCOS
  - 유럽 주도

로봇 소프트웨어 개발 프레임워크 : ROS

❖ **ROS**

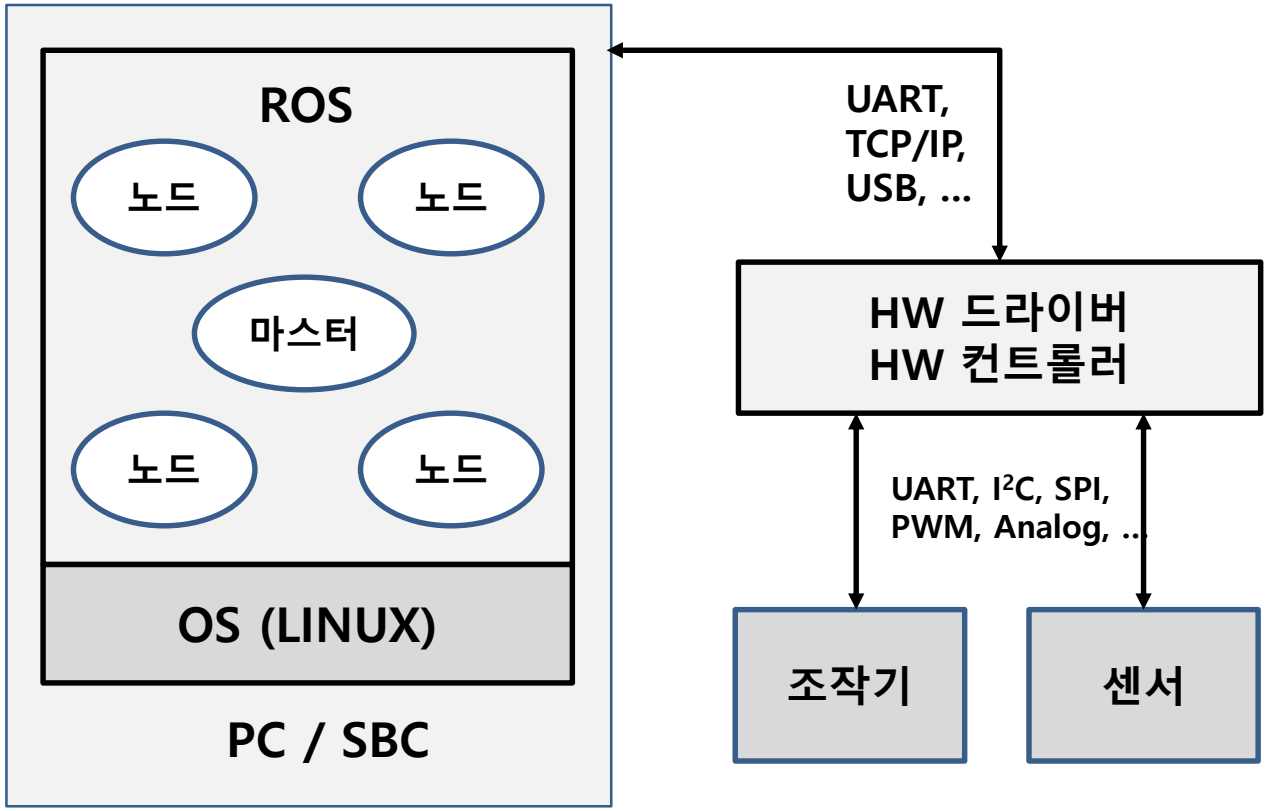
- 로봇 운영체제(Robot Operating System)이라는 이름이지만,
  로봇 소프트웨어 개발 오픈소스 프레임워크
- Stanford AI research에서 프로토타입 제작, 2007년 Willow Gragage에서 공식 개발
  - 현재 Open Source Robotics Foundation에서 관리
- 많은 기반구조, 도구, 기능 제공
  - 빌드(build) 도구, 시뮬레이터, 시각화 도구 등
- 다른 개발자의 소프트웨어 사용 및 자신의 소프트웨어 공유 용이
- 큰 사용자 커뮤니티

- 지속적으로 버전 갱신
- **Linux Ubuntu**만 공식 운영체제로 지원 : 타 운영체제 환경 사용 가능

❖ 교재 16장 참조

로봇 소프트웨어 개발 프레임워크 : ROS

❖ ROS



로봇 소프트웨어 개발 프레임워크 : ROS

❖ ROS 미들웨어

- **메시지 전달**(message passing) 기반의 통신 지원
  - **노드**(node)
    - 실행 가능한 프로그램
    - ROS에서 최소 단위 실행 **프로세스**(process)
  - **토픽**(topic)
    - 노드간에 메시지를 교환하는 **이름을 붙인 버스**(named bus)
  - **마스터**(master)
    - 노드 간의 연결 및 메시지 통신을 위한 **네임서버**(name server) 역할을 하는 노드
    - 노드들의 이름, 토픽과 서비스 등록
  - **발행**(publish)
    - **토픽의 내용**을 해당 메시지 형태의 데이터로 **전송**하는 것
  - **발행자**(publisher)
    - 발행하기 위해 **토픽**과 **자신의 정보**를 마스터에 등록하는 노드
    - 구독자 노드에 메시지 전송

로봇 소프트웨어 개발 프레임워크 : ROS

❖ **ROS 미들웨어**

- **메시지 전달**(message passing) 기반의 통신 지원 – cont.
  - **구독**(subscribe)
    - **토픽의 내용**을 정해진 메시지 형태의 데이터로 **수신**하는 것
  - **구독자**(subscriber)
    - 구독하기 위해 토픽 및 자신의 정보를 마스터에 등록하는 노드
    - **발행자 정보**를 **마스터로 부터** 수신
    - **발행자 노드와 직접 접속**하여 메시지 수신
      - » **소켓** 통신 이용

로봇 소프트웨어 개발 프레임워크 : ROS

❖ **ROS 미들웨어**

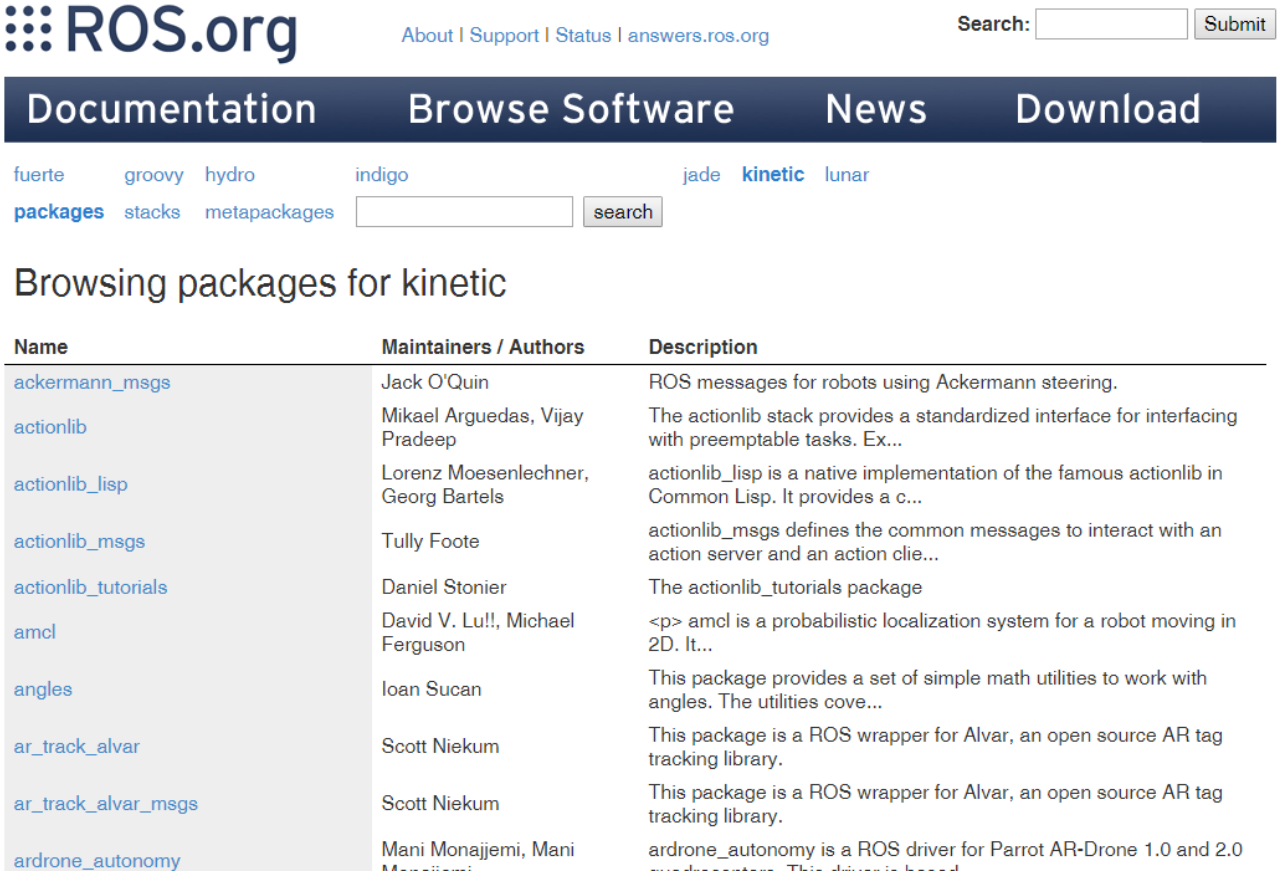
- **원격 프로시저 호출**(remote procedure call)
  - 프로세스 간의 **동기적**(synchronous) **요청**(request)/**응답**(response)
  - 서비스(service) 사용 구현

  - **서비스**(service)
    - 일회성 동기적 메시지 통신
  - **서비스 서버**
    - 요청을 입력으로 받아, 정해진 서비스 프로시저를 실행하고 응답하는 노드
  - **서비스 클라이언트**
    - 서비스를 요청하는 노드

로봇 소프트웨어 개발 프레임워크 : ROS

❖ ROS의 패키지

- 다양한 기능의 소프트웨어를 패키지 형태로 제공
- ROS wiki, github 등에 공개
  - 5000개 이상 패키지



5. 로봇 계획수립

- **움직임 계획수립**(motion planning)
  - 로봇이 **원하는 움직임 작업**을 수행할 수 있도록, 움직임에 관련된 제약조건을 만족하게 하면서 평가 기준을 최적화하는 **일련의 이산적인 움직임**들을 찾는 것

- **경로 계획수립**(path planning)
  - 주어진 시작 위치에서 목표 위치로 가기 위한 **관절 공간**이나 **이동 공간** 내에서의 **시간적 순서**에 따른 **위치**들을 찾는 것

- **궤적 계획수립**(trajectory planning)
  - 주어진 **경로**와 **제약조건** 및 **로봇의 기계적인 특성**을 고려하여 **매 시점**의 **관절** 또는 **바퀴**의 **위치, 속도, 가속도** 등의 값을 결정하는 것

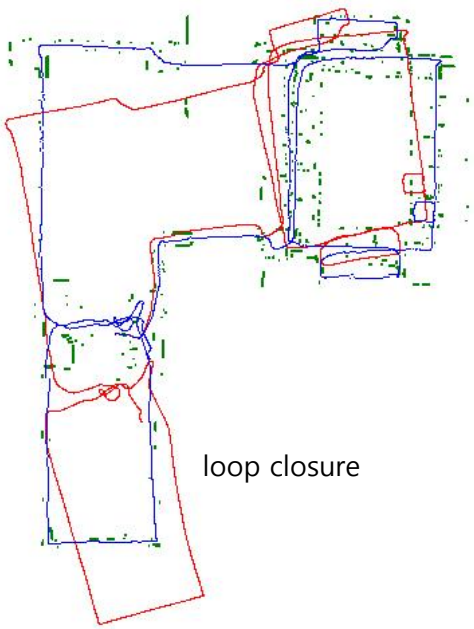
6. 동시적 위치추정 및 지도작성

❖ **동시적 위치추정 및 지도작성(SLAM)**

- **Simultaneous Localization And Mapping**
  - 로봇이 알려지지 않은 정적인 환경(unknown, static environment)을 이동하면서 지도를 함께 작성하는 것
  - 주로 실내환경에 적용

- **주어진 정보**
  - 로봇의 제어신호
  - 근처의 특징 관측정보

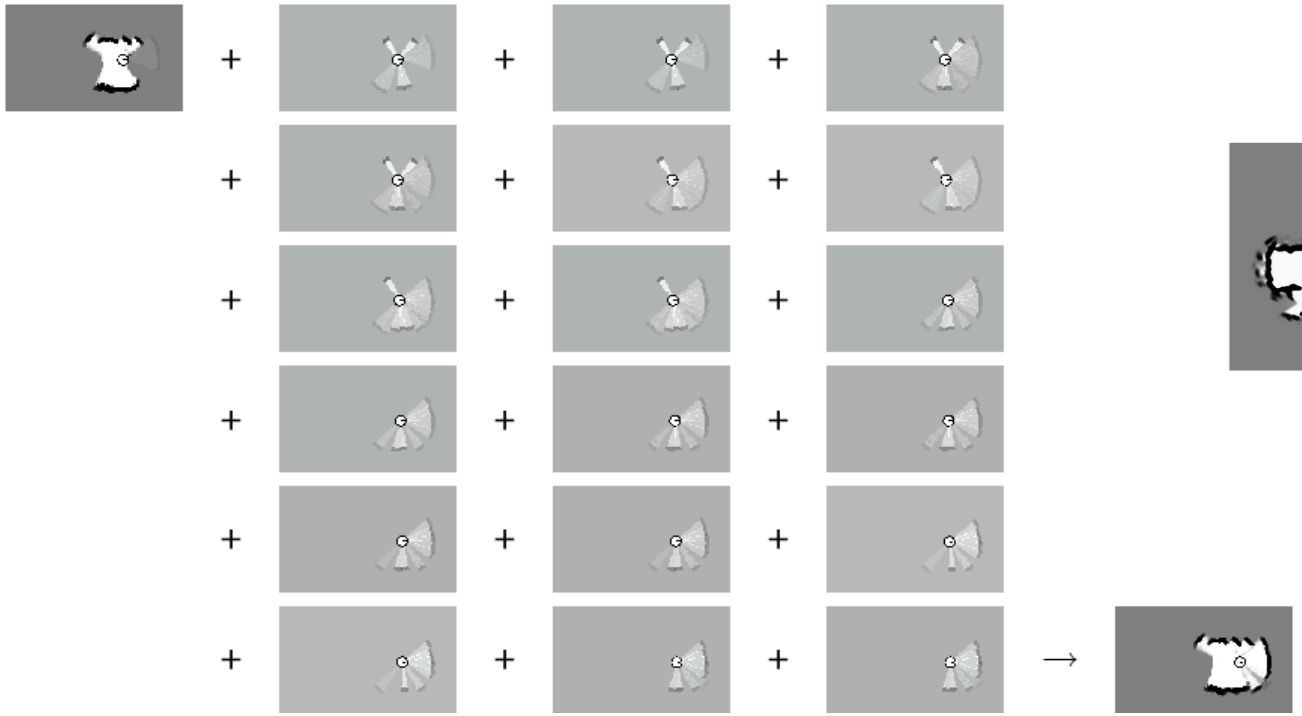
- **추정 정보**
  - 특징 지도
  - 로봇의 경로

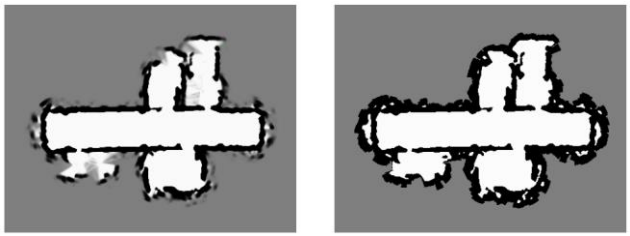


동시적 위치추정 및 지도작성

❖ **지도 작성(map making)**

- 점유 격자 지도(Occupancy Grid Maps)
  - 공간을 격자**(grid)**로 표현
  - 각 격자가 장애물에 의해 점유되었는지 여부의 확률 추정
  - 로봇의 위치는 안다고 가정





동시적 위치추정 및 지도작성

❖ **위치 추정**

- 지도가 없는 상태 또는 지도가 만들어지고 있는 상태에서 위치 추정

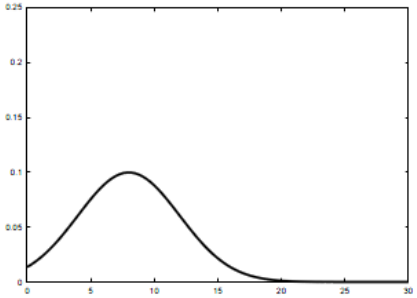
- **추측 항법(dead reckoning)**
  - 바퀴의 회전량을 사용하여 상대적인 위치를 추정하는 방법
  - 엔코더 사용
  - 관성 센서 등을 사용 정확도 보완

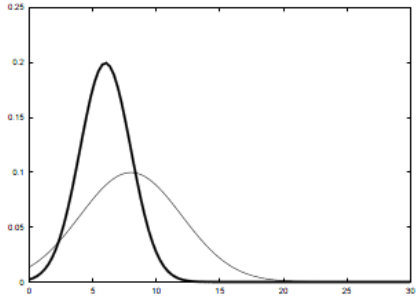
동시적 위치추정 및 지도작성

❖ Kalman Filter

- 시간에 따른 **관측 데이터를 사용**하여 **미지의 변수**(**unknown variable**) **의 추정치를 구하는 알고리즘**
- 잡음(Noise)을 **가우시안**(Gaussian) **함수**로 표현

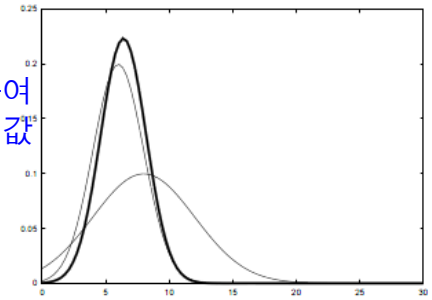
초기상태
belief

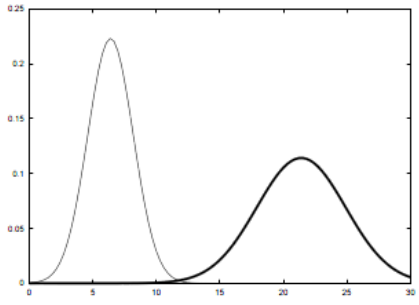




측정값
measurement

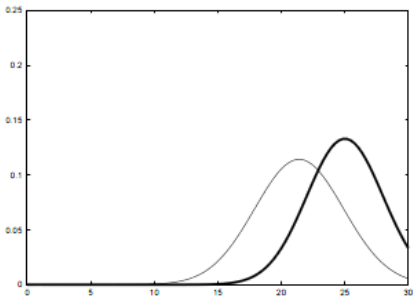
Kalman filter를 이용하여
초기상태 belief와 측정값
을 결합한 belief

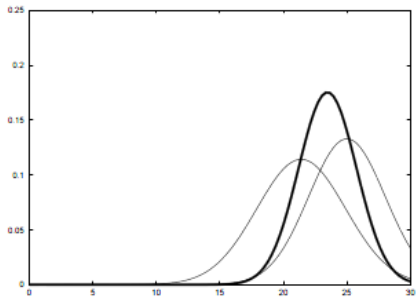




오른쪽 이동후 belief
(불확실성 확대)

새로운 측정값





최종 belief

동시적 위치추정 및 지도작성

❖ Kalman Filter

- 로봇의 **다음 상태** (next state)

$$\mathbf{x}_k = \mathbf{F}_k\mathbf{x}_{k-1} + \mathbf{B}_k\mathbf{u}_k + \mathbf{w}_k \qquad \mathbf{w}_k \sim N(0, \mathbf{Q}_k)$$

- $\mathbf{F}_k$ is the state transition model which is applied to the previous state $\mathbf{x}_{k-1}$;
- $\mathbf{B}_k$ is the control-input model which is applied to the control vector $\mathbf{u}_k$;
- $\mathbf{w}_k$ is the process noise which is assumed to be drawn from a zero mean multivariate normal distribution with covariance $\mathbf{Q}_k$.

- **관측값** (measurement)

$$\mathbf{z}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k \qquad \mathbf{v}_k \sim N(0, \mathbf{R}_k)$$

$\mathbf{H}_k$ is the observation model which maps the true state space into the observed space

- **k-번째 시점의 상태 표현**
  - **평균 추정위치**와 **공분산**(covariance)로 표현
    - $\hat{\mathbf{x}}_{k|k}$, the *a posteriori* state estimate at time k given observations up to and including at time k;
    - $\mathbf{P}_{k|k}$, the *a posteriori* error covariance matrix (a measure of the estimated accuracy of the state estimate).

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_k\hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}_{k-1}\mathbf{u}_{k-1}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k\mathbf{P}_{k-1|k-1}\mathbf{F}_k^{\mathrm{T}} + \mathbf{Q}_k$$

동시적 위치추정 및 지도작성

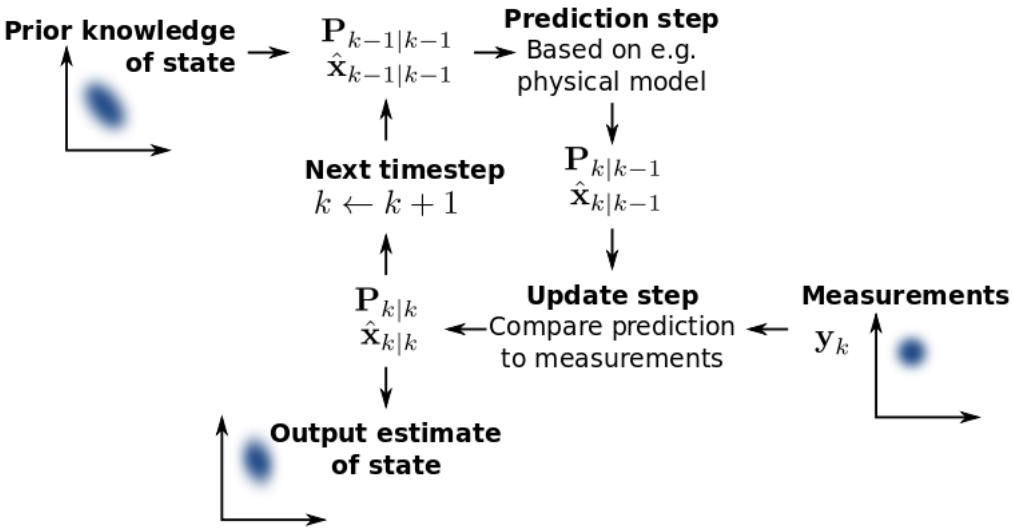
❖ Kalman Filter

Predict

Predicted (*a priori*) state estimate	$\hat{\mathbf{x}}_{k\mid k-1} = \mathbf{F}_k\hat{\mathbf{x}}_{k-1\mid k-1} + \mathbf{B}_{k-1}\mathbf{u}_{k-1}$
Predicted (*a priori*) estimate covariance	$\mathbf{P}_{k\mid k-1} = \mathbf{F}_k\mathbf{P}_{k-1\mid k-1}\mathbf{F}_k^{\mathrm{T}} + \mathbf{Q}_k$

Update

Innovation or measurement residual	$\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k\mid k-1}$
Innovation (or residual) covariance	$\mathbf{S}_k = \mathbf{H}_k\mathbf{P}_{k\mid k-1}\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k$
Optimal Kalman gain	$\mathbf{K}_k = \mathbf{P}_{k\mid k-1}\mathbf{H}_k^{\mathrm{T}}\mathbf{S}_k^{-1}$
Updated (*a posteriori*) state estimate	$\hat{\mathbf{x}}_{k\mid k} = \hat{\mathbf{x}}_{k\mid k-1} + \mathbf{K}_k\tilde{\mathbf{y}}_k$
Updated (*a posteriori*) estimate covariance	$\mathbf{P}_{k\mid k} = (I - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k\mid k-1}$



동시적 위치추정 및 지도작성

❖ **Kalman Filter를 이용한 위치로 속도 재기**

- 열차의 **위치정보**만 가지고 측정하지 않은 **속도 알아내기**
- 이동 거리를 시간으로 나누어 계산한 속도에는 **많은 잡음 포함**
- **상태변수**

$$\mathbf{x} = \begin{bmatrix} \text{위치} \\ \text{속도} \end{bmatrix}$$

- **동작을 나타내는 시스템 모델**

$$\mathbf{x}_{k+1} = \mathrm{F}\mathbf{x}_{\mathbf{k}} + w_k = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \mathbf{x}_k + w_k$$ 상태변수

$$z_k = \mathrm{H}\mathbf{x}_k + v_k = \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{x}_k + v_k$$ 측정값

$$w_k \Leftarrow N(0,Q), \quad Q = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}$$

$$v_k \Leftarrow N(0,R), \quad R = 10$$

동시적 위치추정 및 지도작성

❖ Kalman Filter를 이용한 위치로 속도 재기

```
function [pos vel] = DvKalman(z)
% 칼만 필터 알고리즘
persistent F H Q R
persistent x P
persistent firstRun

if isempty(firstRun)
    firstRun = 1;
    dt = 0.1;
    F = [1 dt; 0 1];    % 동작 모델 행렬
    H = [1 0];          % 관측 모델 행렬
    Q = [1 0; 0 3];     % 동작 모델에 대한 공분산행렬
    R = 10;             % 관측 모델에 대한 분산
    x = [0 20]';        % 처음 상태
    P = 5*eye(2);       % 처음 상태에서의 공분산행렬
end

xp = F*x;           % 동작 모델에 따른 상태
Pp = F*P*F' + Q;    % 동작 모델에 따른 상태의 공분산(covariance) 행렬

K = Pp*H'*inv(H*Pp*H' + R); % Kalman gain
x = xp + K*(z - H*xp); % 관측치를 이용한 보정된 상태
P = Pp - K*H*Pp;       % 보정된 상태의 공분산

pos = x(1);             % 보정된 위치
vel = x(2);             % 보정된 속도
```

동시적 위치추정 및 지도작성

❖ **Kalman Filter를 이용한 위치로 속도 재기**

```
function z = GetPos()
% 열차속도가 80m/s에서 약간씩 변한다고 가정하고 위치 정보 생성
persistent Posp Velp
if isempty(Posp)
    Posp = 0;   % 처음 위치
    Velp = 80;  % 기준 속도
end

dt = 0.1;  % 시간 간격
w = 5*randn;   % 속도에 대한 변화
v = 10*randn;  % 위치에 대한 잡음

z = Posp + Velp*dt + v;  % 측정된 위치

Posp = Posp + Velp*dt;  % 위치의 참값
Velp = 80+w;  % 속도의 참값
end
```

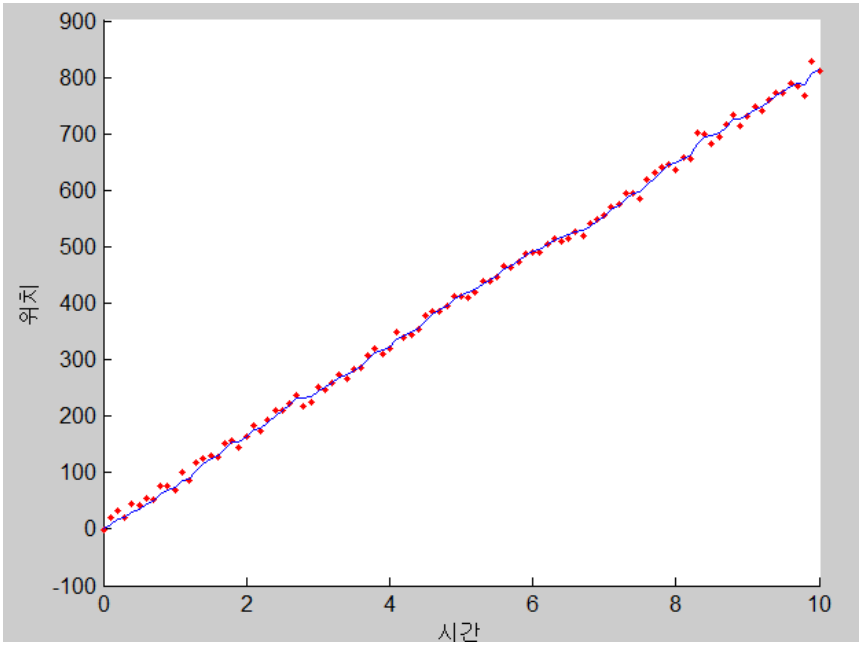
동시적 위치추정 및 지도작성

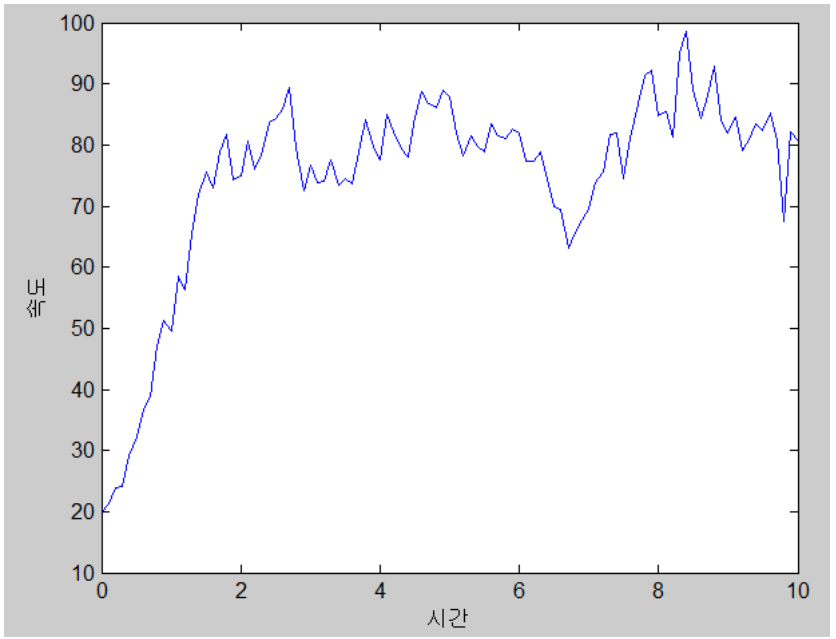
❖ Kalman Filter를 이용한 위치로 속도 재기

```
clear all

dt = 0.1;  % 시간 간격
t=0:dt:10;
Nsamples = length(t);
Xsaved = zeros(Nsamples, 2);
Zsaved = zeros(Nsamples, 1);


for k = 1:Nsamples
    z = GetPos();           % 위치 측정
    [pos vel]  = DvKalman(z);   % Kalman Filter를 이용한 상태 추정
    Zsaved(k) = z;
    Xsaved(k,:) = [pos vel];
end

figure
hold on
plot(t, Zsaved(:), 'r.');
plot(t, Xsaved(:,1))
xlabel('시간'); ylabel('위치')

figure
plot(t, Xsaved(:,2))
xlabel('시간'); ylabel('속도')
```

동시적 위치추정 및 지도작성

❖ **Kalman Filter를 이용한 위치로 속도 재기**





Object Tracking :
http://www.youtube.com/watch?v=Jq8HcIar68Y&src_vid=GBYW1j9lC1I&feature=iv&annotation_id=annotation_888715

동시적 위치추정 및 지도작성

- **Extended Kalman Filter (EKF)**
  - **상태 변화 및 관측값**이 선형함수(linear function)으로 표현되지 않고, **비선형 함수**로 표현되는 경우 적용

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_k) + \mathbf{w}_k$$
$$\mathbf{z}_k = h(\mathbf{x}_k) + \mathbf{v}_k$$

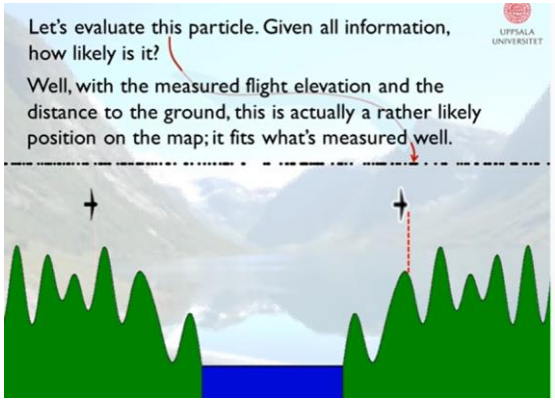
  - **부분적으로 선형화**하여 처리

- **Unscented Kalman Filter (UKF)**
  - EKF에 대한 근사적인 방법

동시적 위치추정 및 지도작성

❖ 파티클 필터(Particle Filter)

- **로봇의 위치**를 **표본(파티클)들의 분포**로 표현
- **다음 과정 반복**
  1. **파티클 이동**
     - 각 파티클을 동적 시스템 방정식을 이용해서 이동
  2. **확률 계산**
     - 주어진 지도 및 상황에서 **해당 위치가 참**일 **확률** 계산
  3. **재 표본 추출**
     - 계산된 **확률에 따라 파티클을 다시 표본 추출**하여 파티클 집합 구성



http://www.youtube.com/watch?v=aUkBa1zMKv4

7. 항법

❖ **항법**(navigation)
- 로봇이 정해진 목적지로 이동하는 것

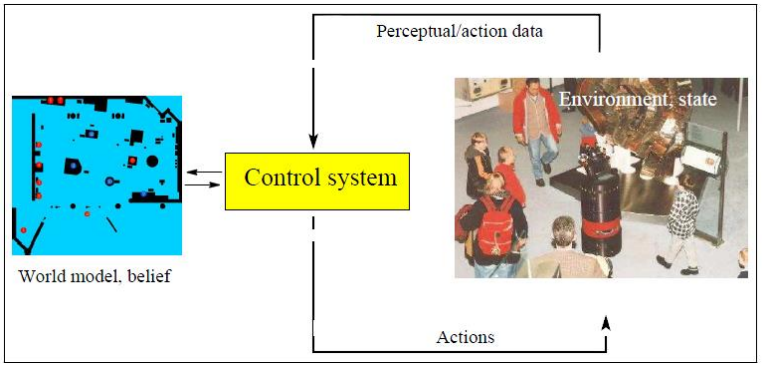
❖ **항법을 위한 필요 요소**
- **지도** : 정밀지도
  - **LDM(Local Dynamic Map)** : 정적/동적 정보를 저장하는 주행정보지도
- 로봇의 **위치 계측** 및 **추정**하는 기능
- 벽, 물체 등의 **장애물**을 계측하는 기능
- 목적지까지의 **최적 경로**를 계산하고 **주행**하는 기술

항법

❖ 항법에서의 위치 추정(Localization)

- **지도**가 주어진 상태에서 센서 데이터를 참고하여 **로봇의 위치**를 결정

- **아이콘 기반 위치 추정 기법**(Iconic localization method)
  - 센서 관측정보를 **직접 지도**에 **대조**하여 위치 결정
- **특징기반 위치 추정 기법**(Feature-based localization method)
  - 코너(corner), 랜드마크(landmark) 등의 특징을 추출하여 위치 결정
- **확률적 위치 추정 기법**(Probabilistic localization method)
  - 확률 분포를 이용하여 장소에 대한 분포 정보수신
  - **Kalman 필터, 파티클 필터** 등

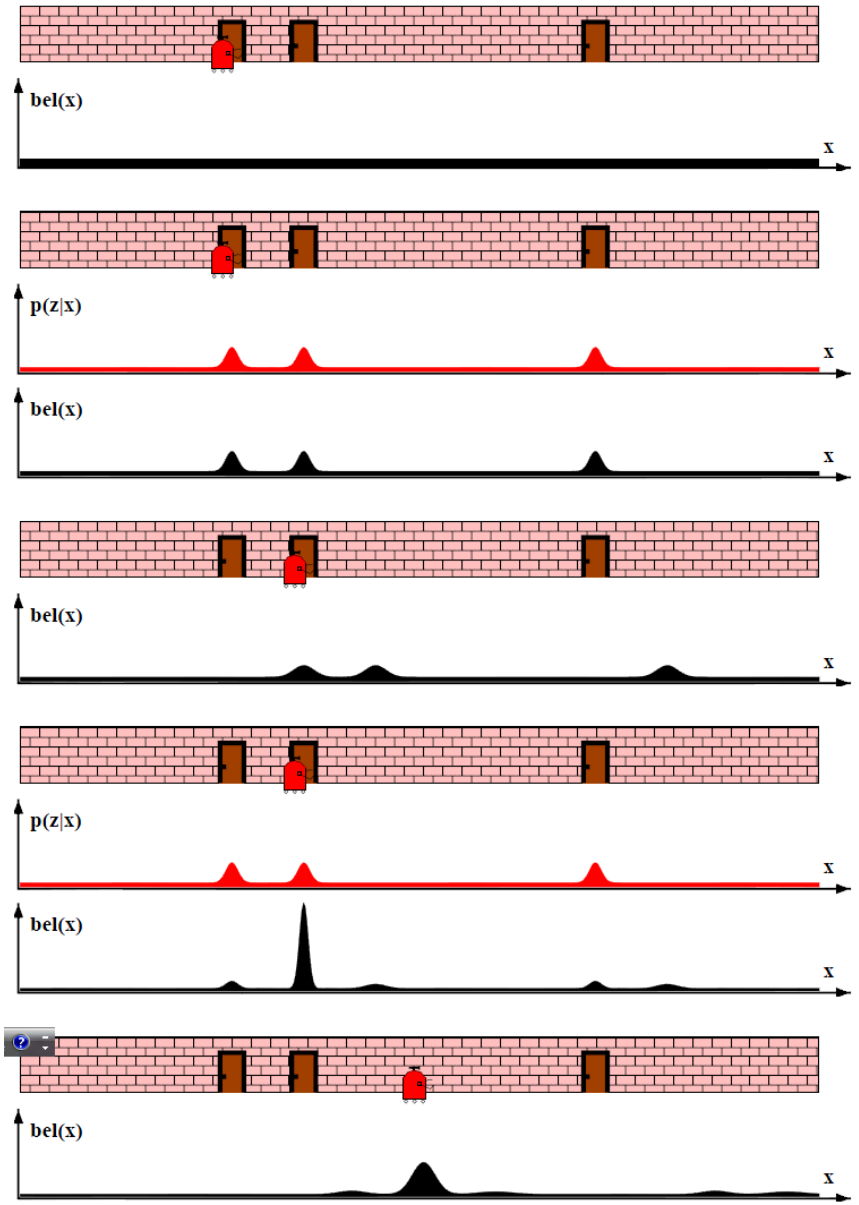


항법

❖ 확률적 위치 추정

- $bel(x_t) \quad = \quad p(x_t \mid z_{1:t}, u_{1:t})$

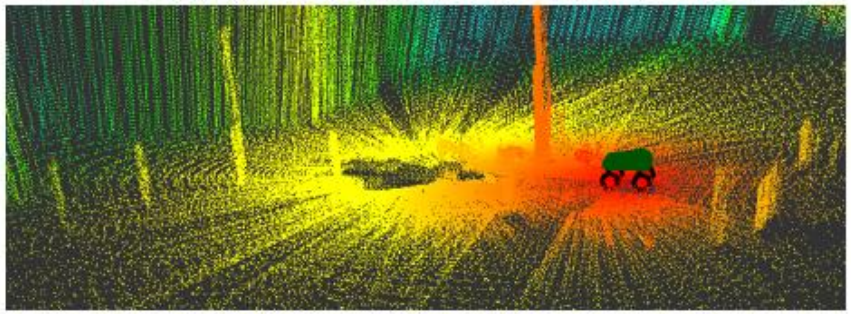
all past measurements $z_{1:t}$
all past controls $u_{1:t}$



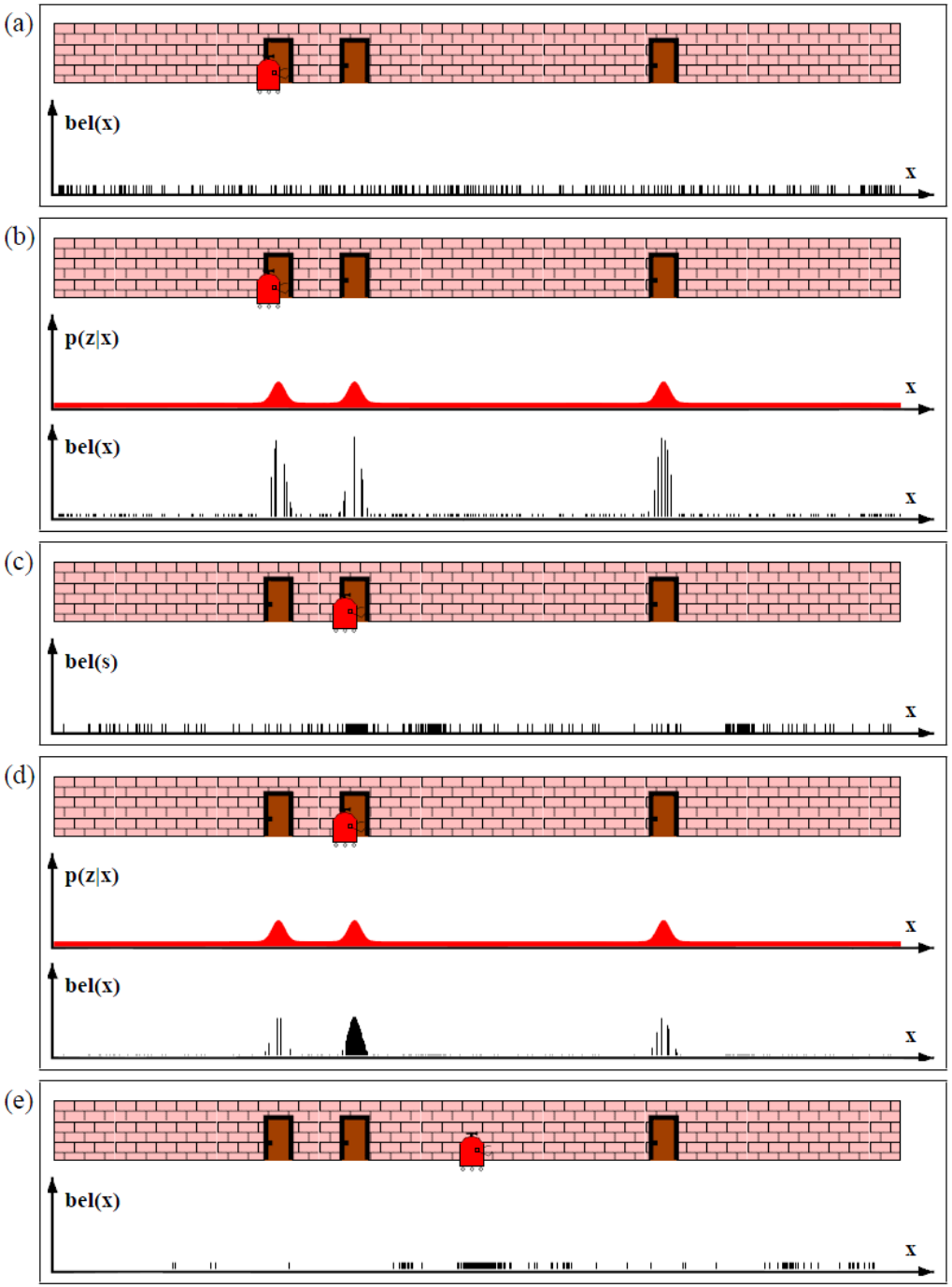
항법

❖ **파티클 필터**

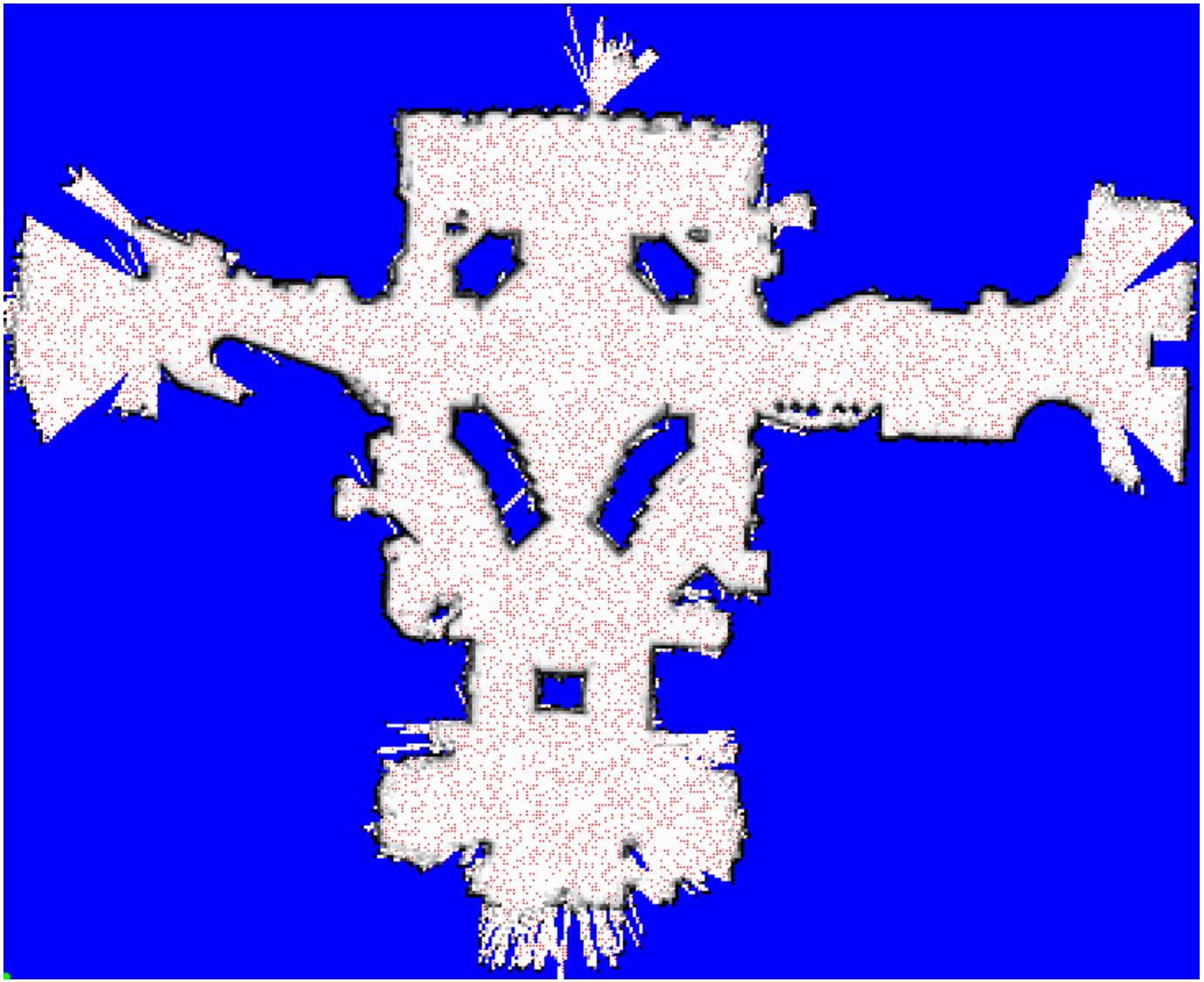
- 지도(map) 정보는 알고 있다고 가정
- x는 로봇이 위치로서 가능한 위치 후보
- z는 로봇이 센싱한 정보

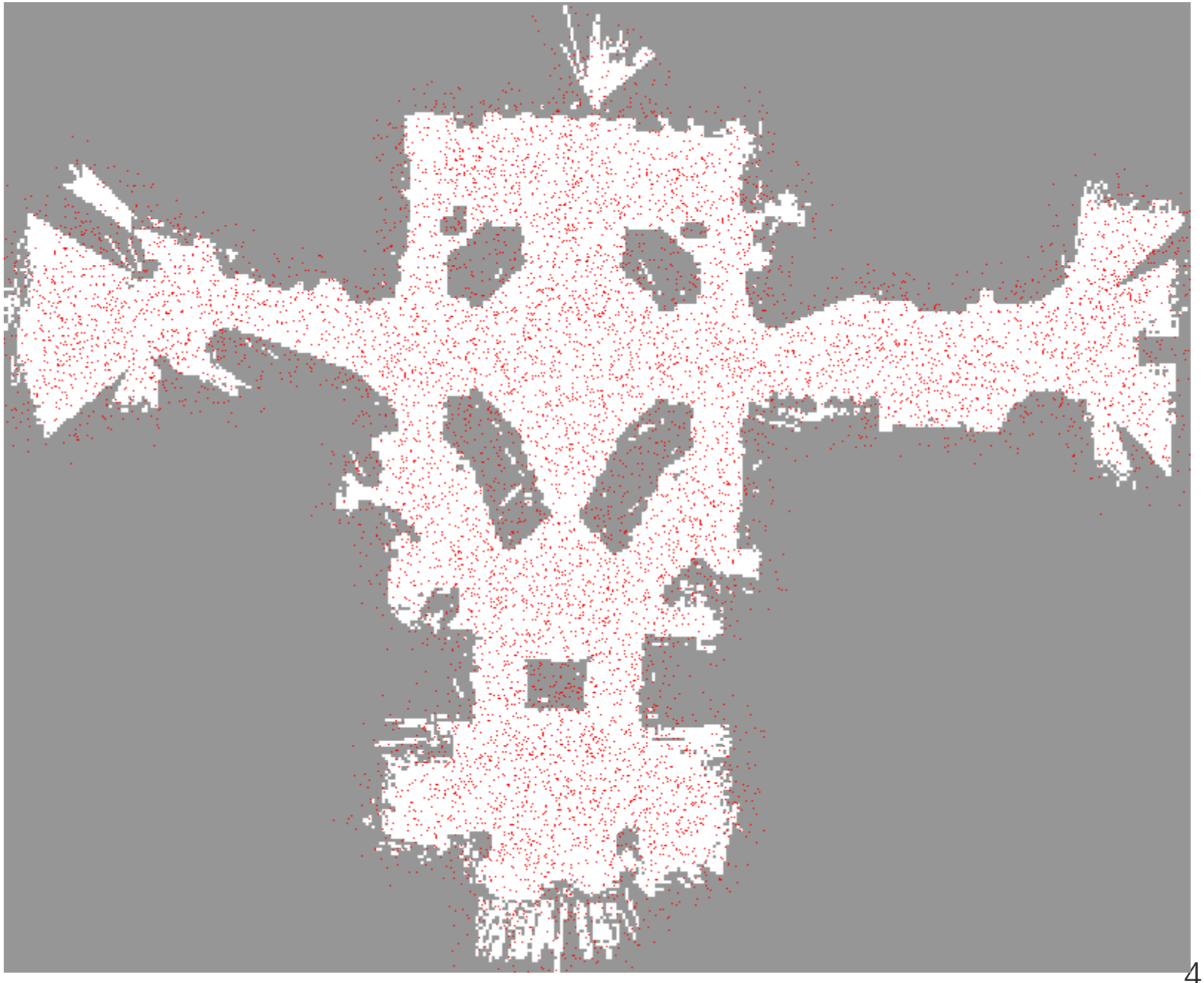


3차원 point cloud data

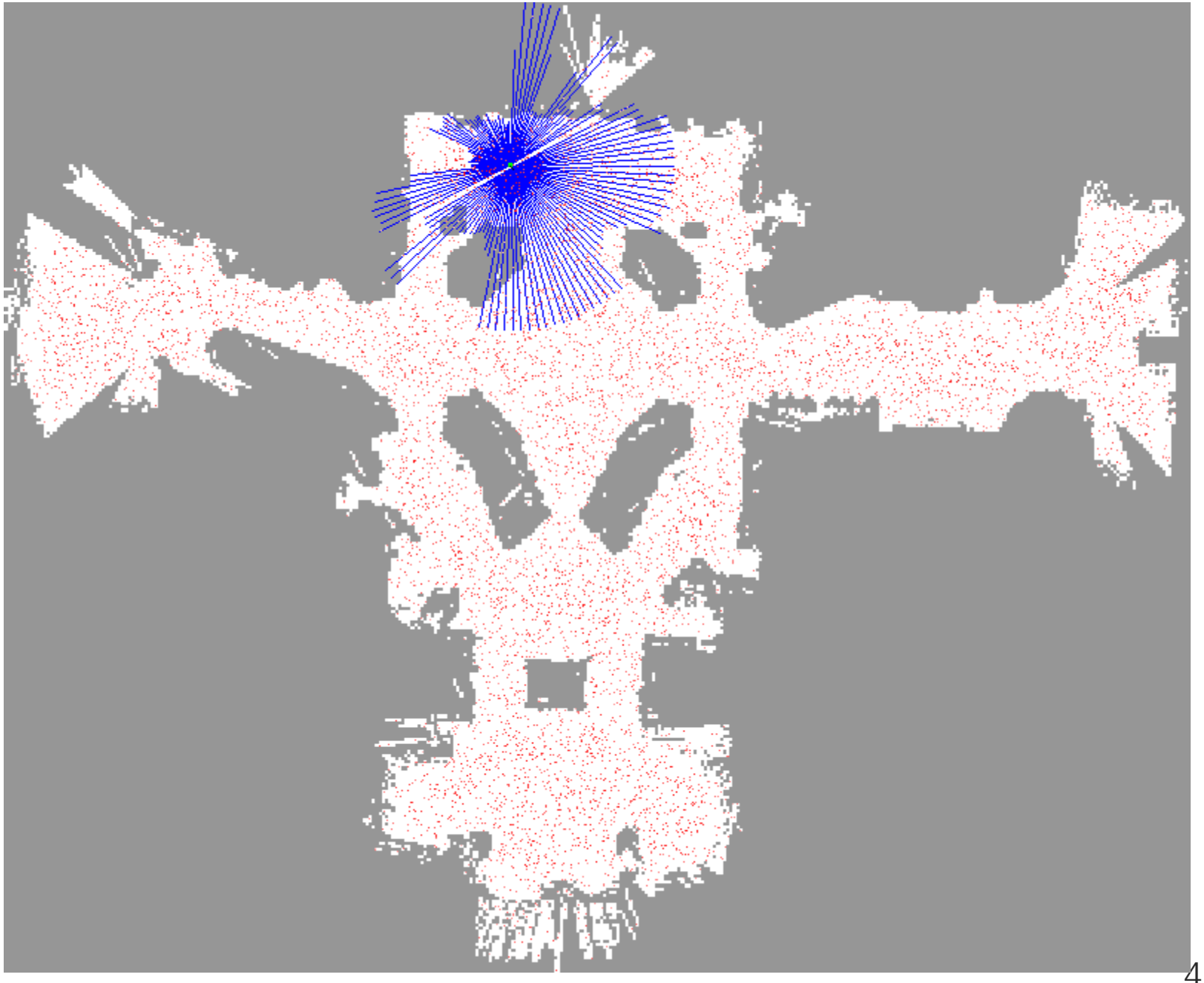


파티클 필터를 이용한 위치 추정

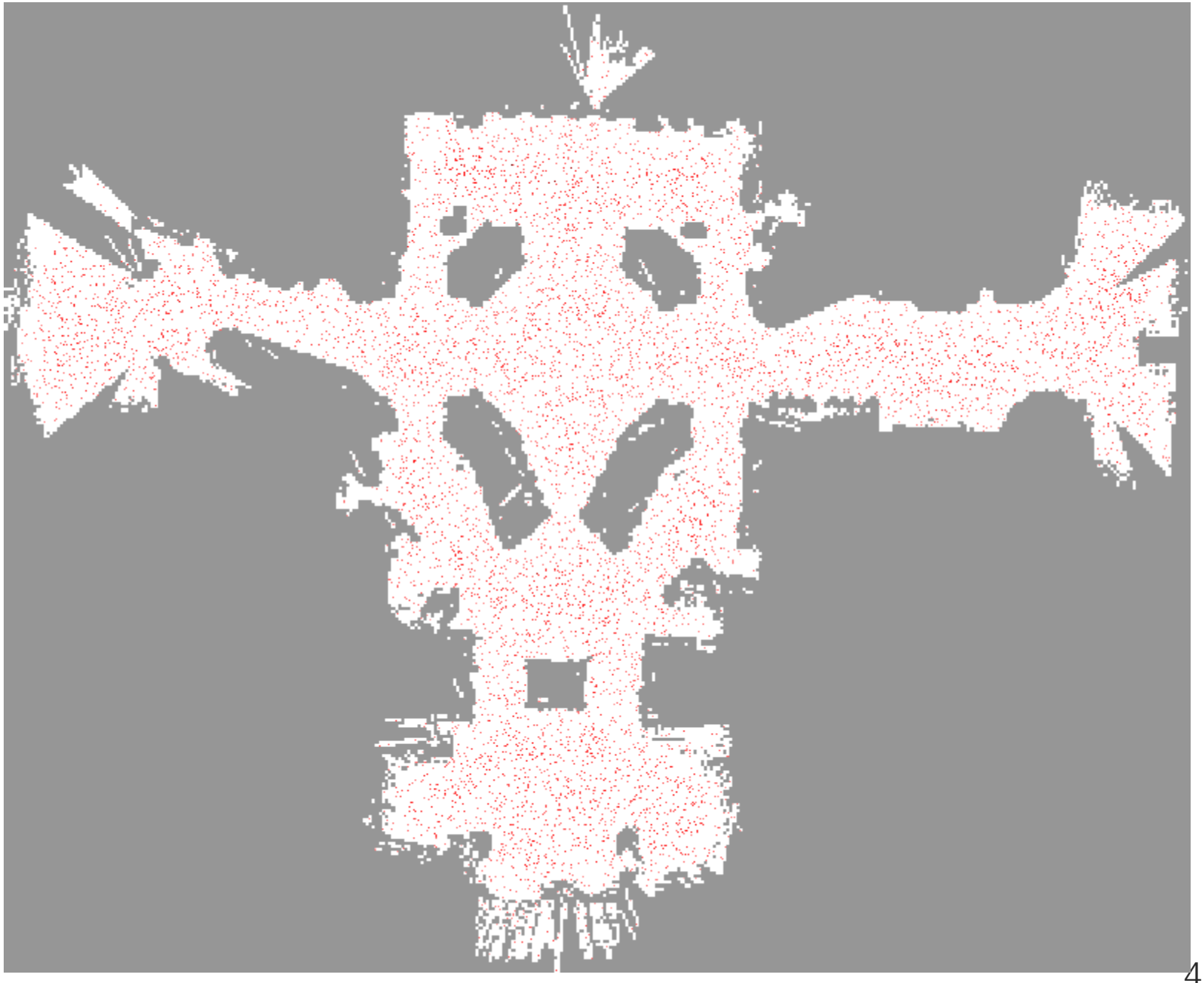




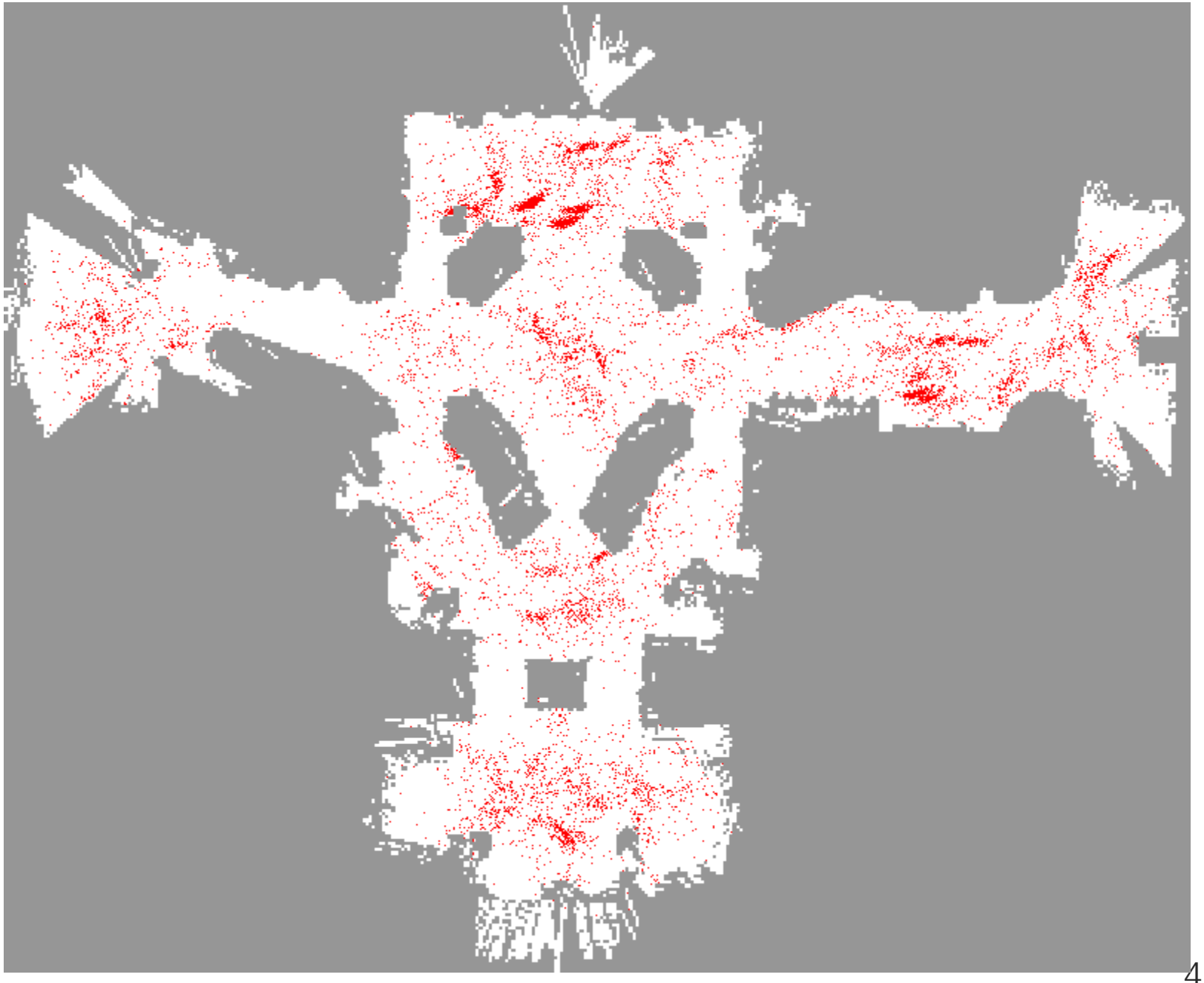
파티클 필터를 이용한 위치 추정



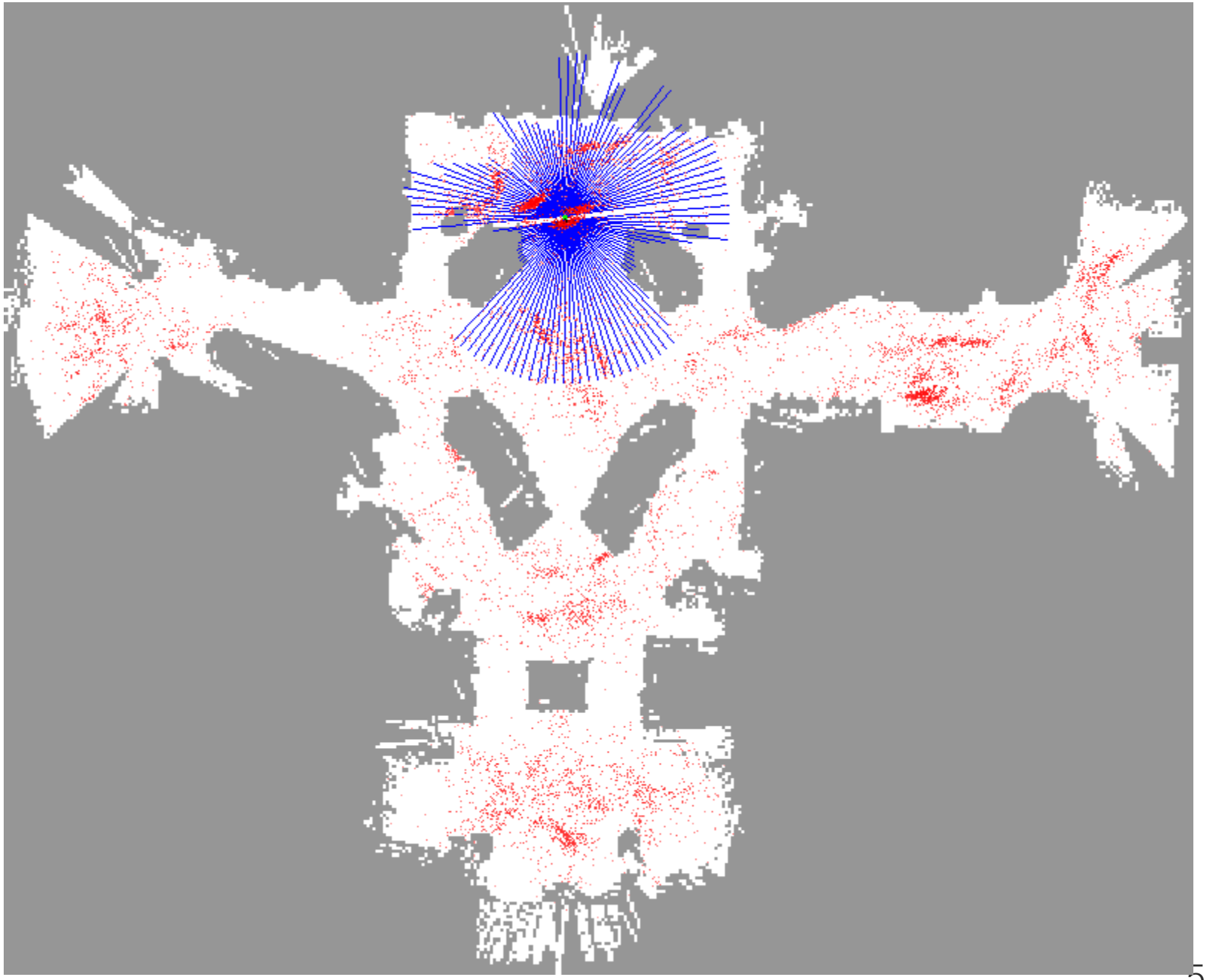
파티클 필터를 이용한 위치 추정



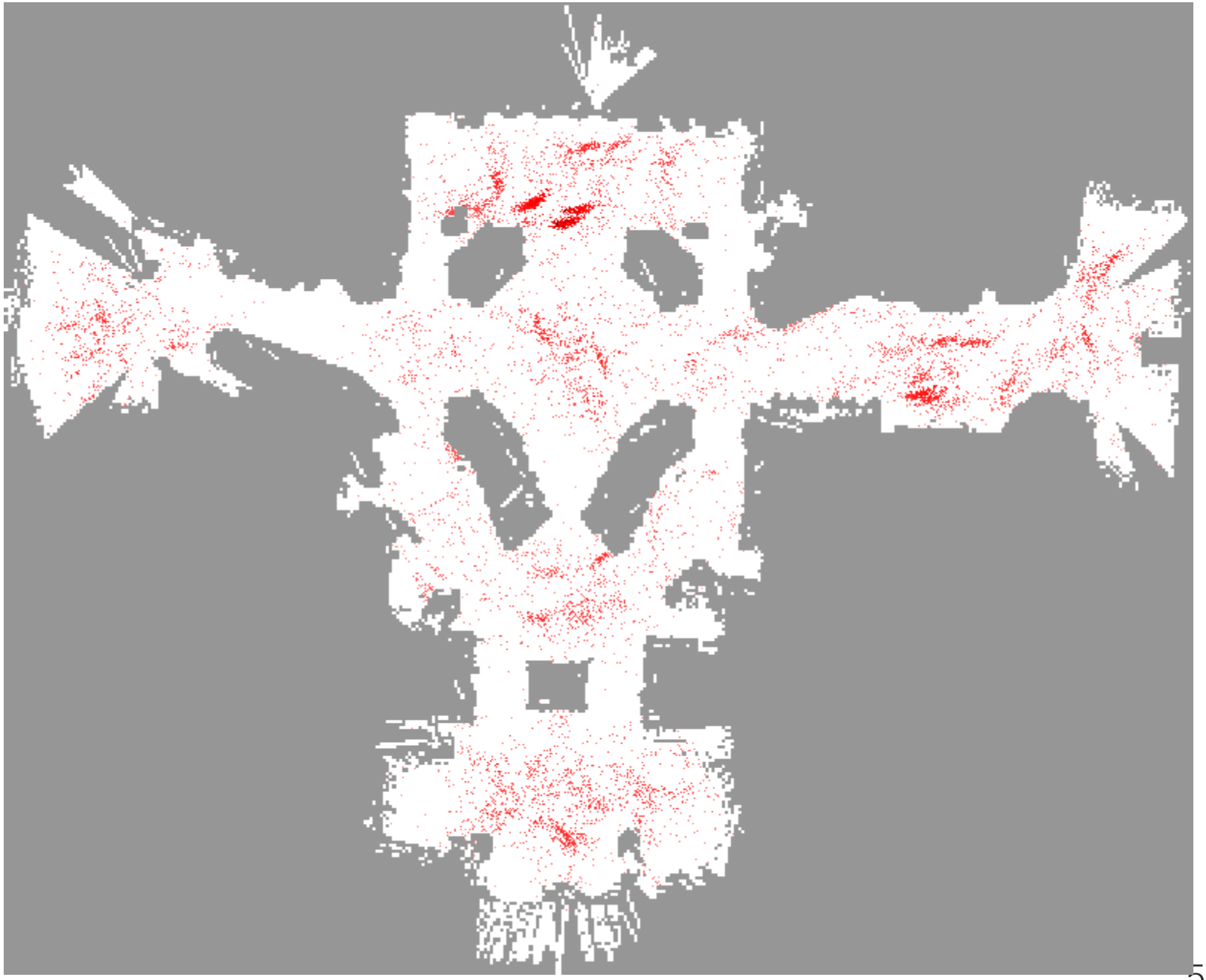
파티클 필터를 이용한 위치 추정



파티클 필터를 이용한 위치 추정



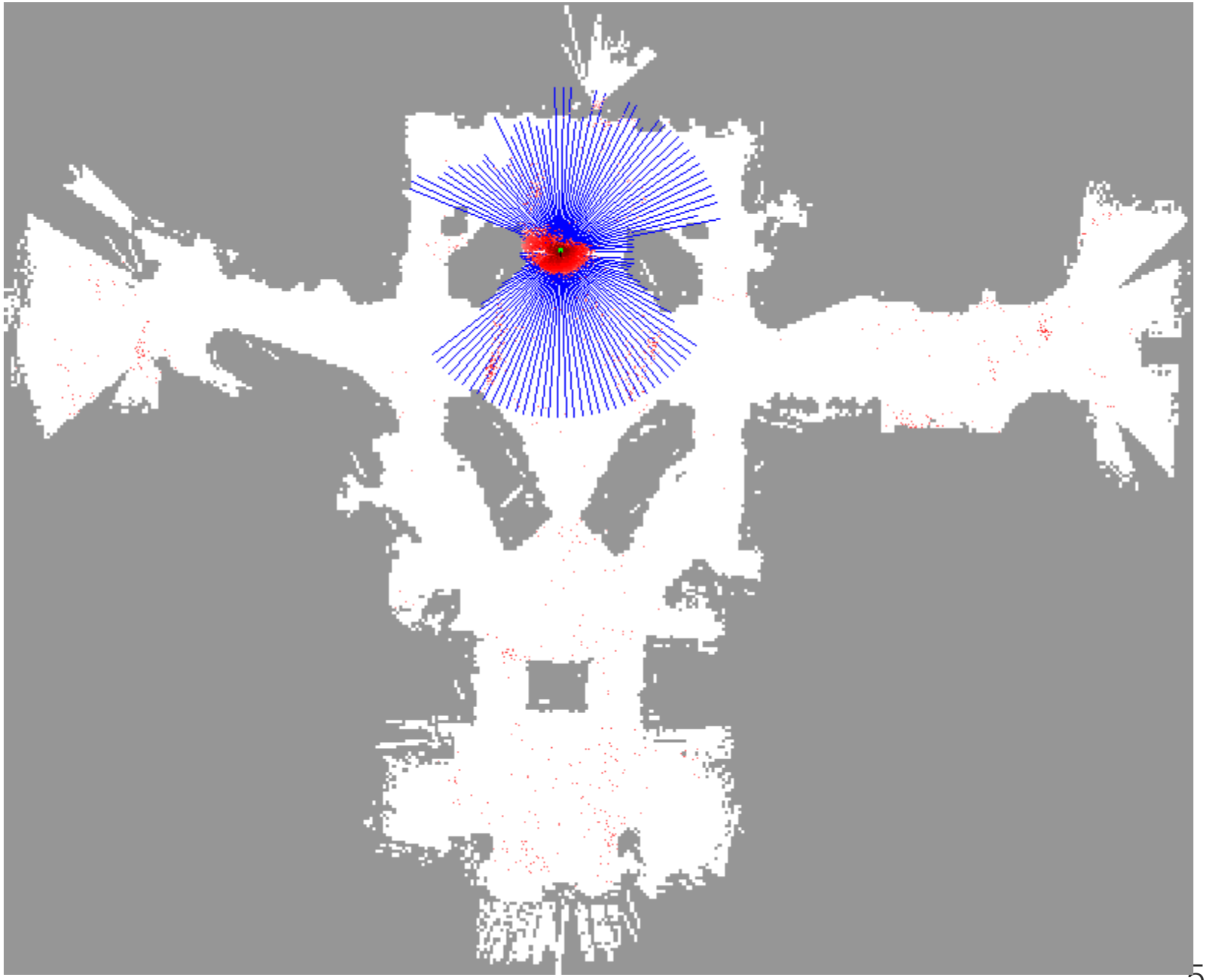
파티클 필터를 이용한 위치 추정



파티클 필터를 이용한 위치 추정

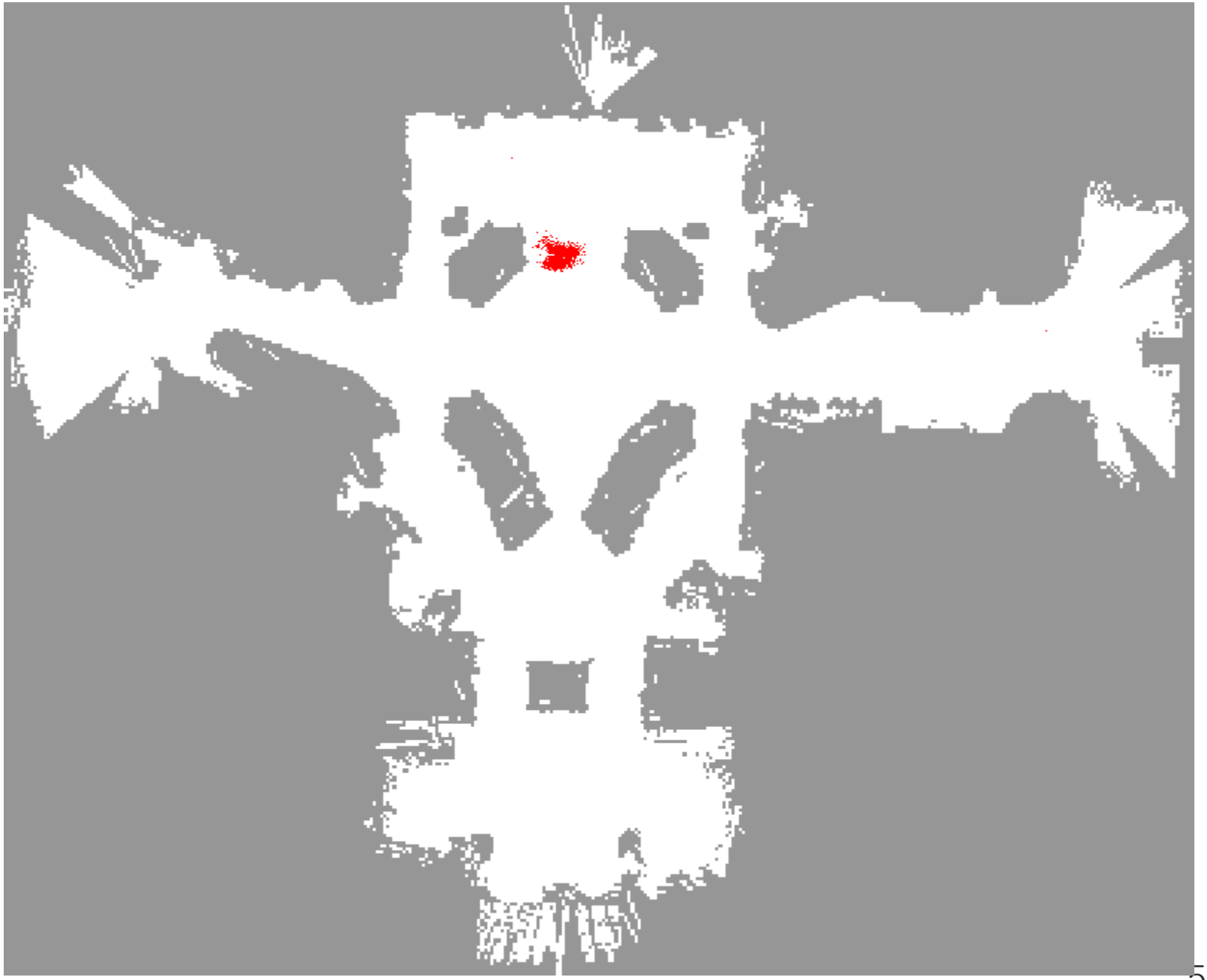
파티클 필터를 이용한 위치 추정

파티클 필터를 이용한 위치 추정

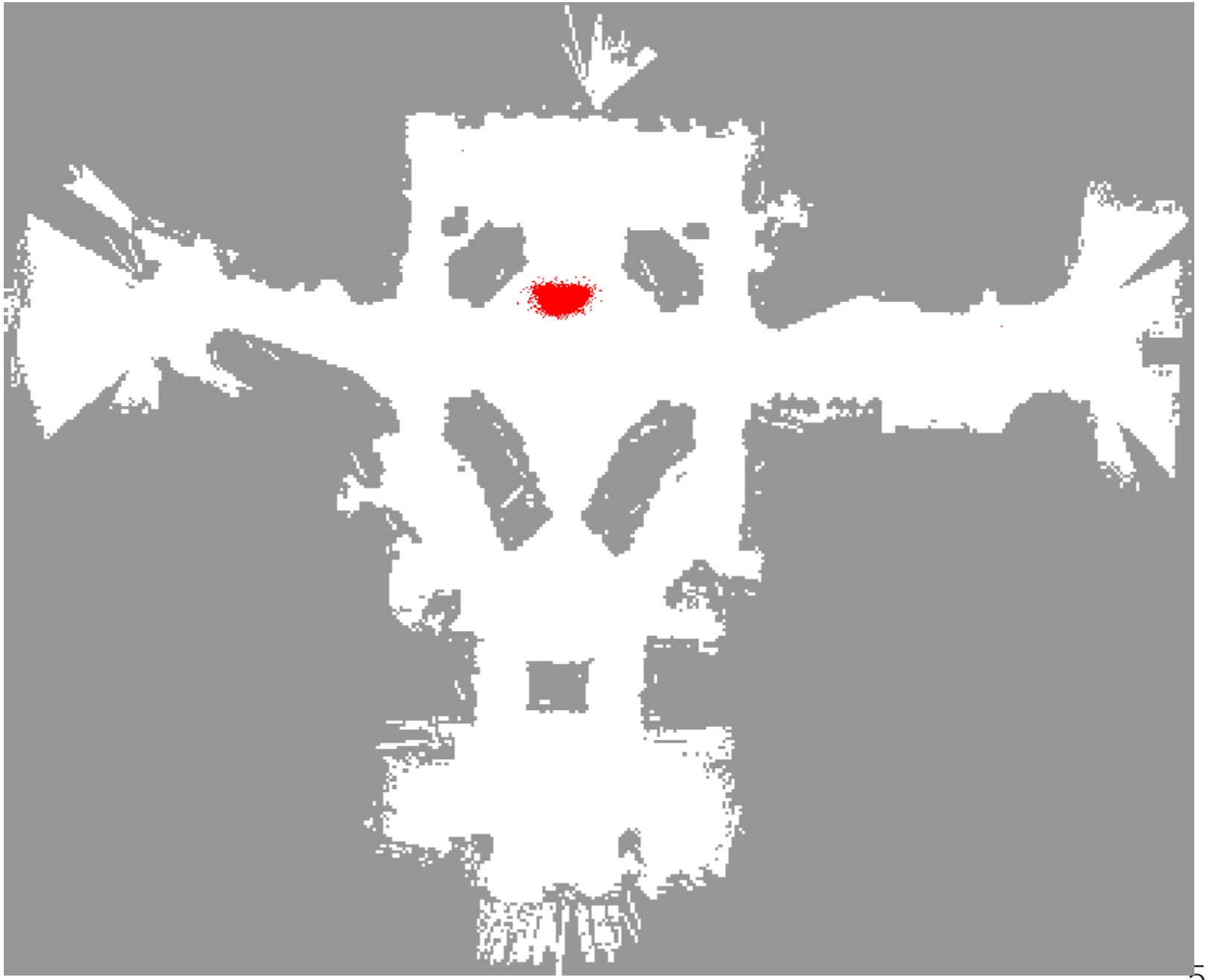


파티클 필터를 이용한 위치 추정

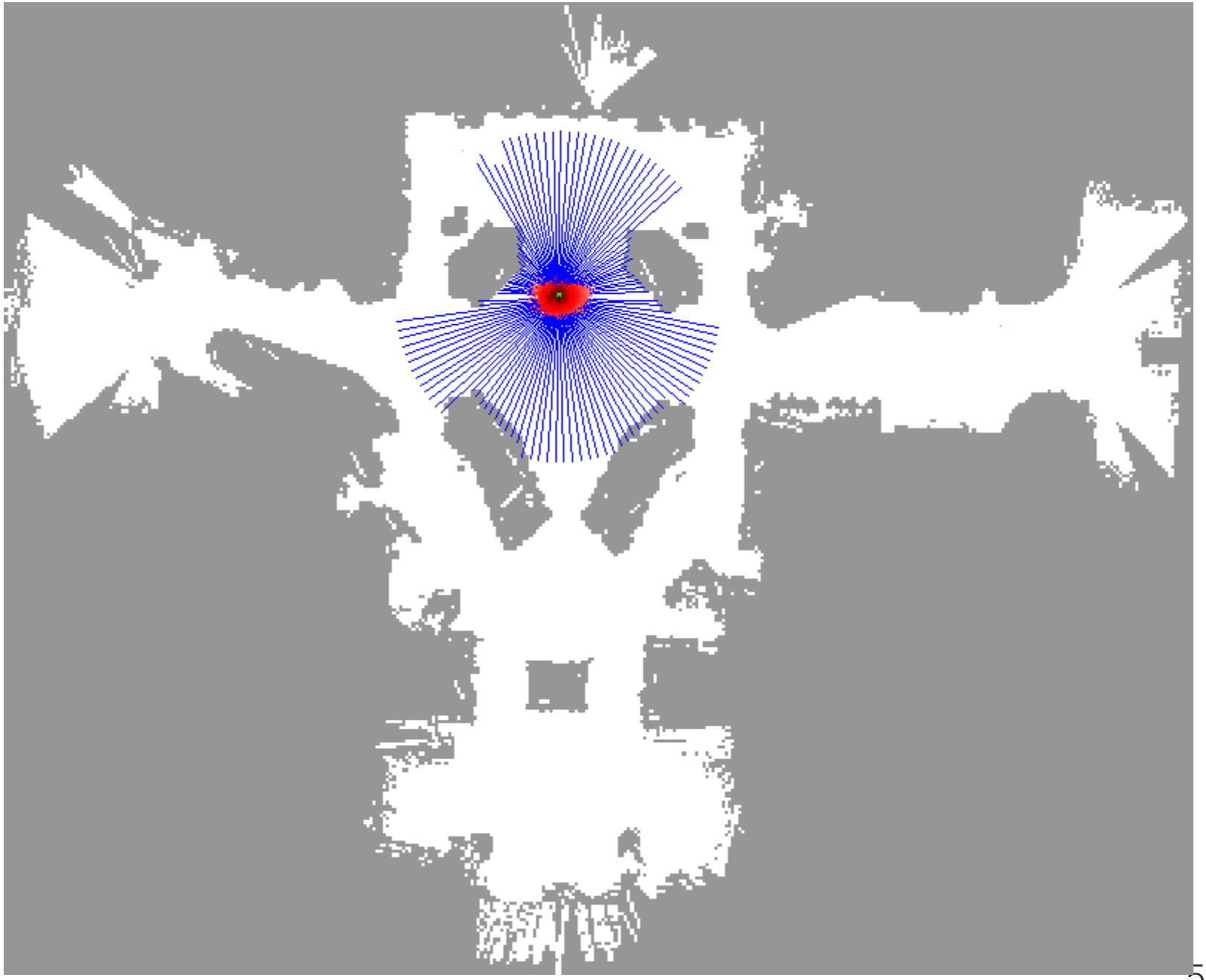
파티클 필터를 이용한 위치 추정



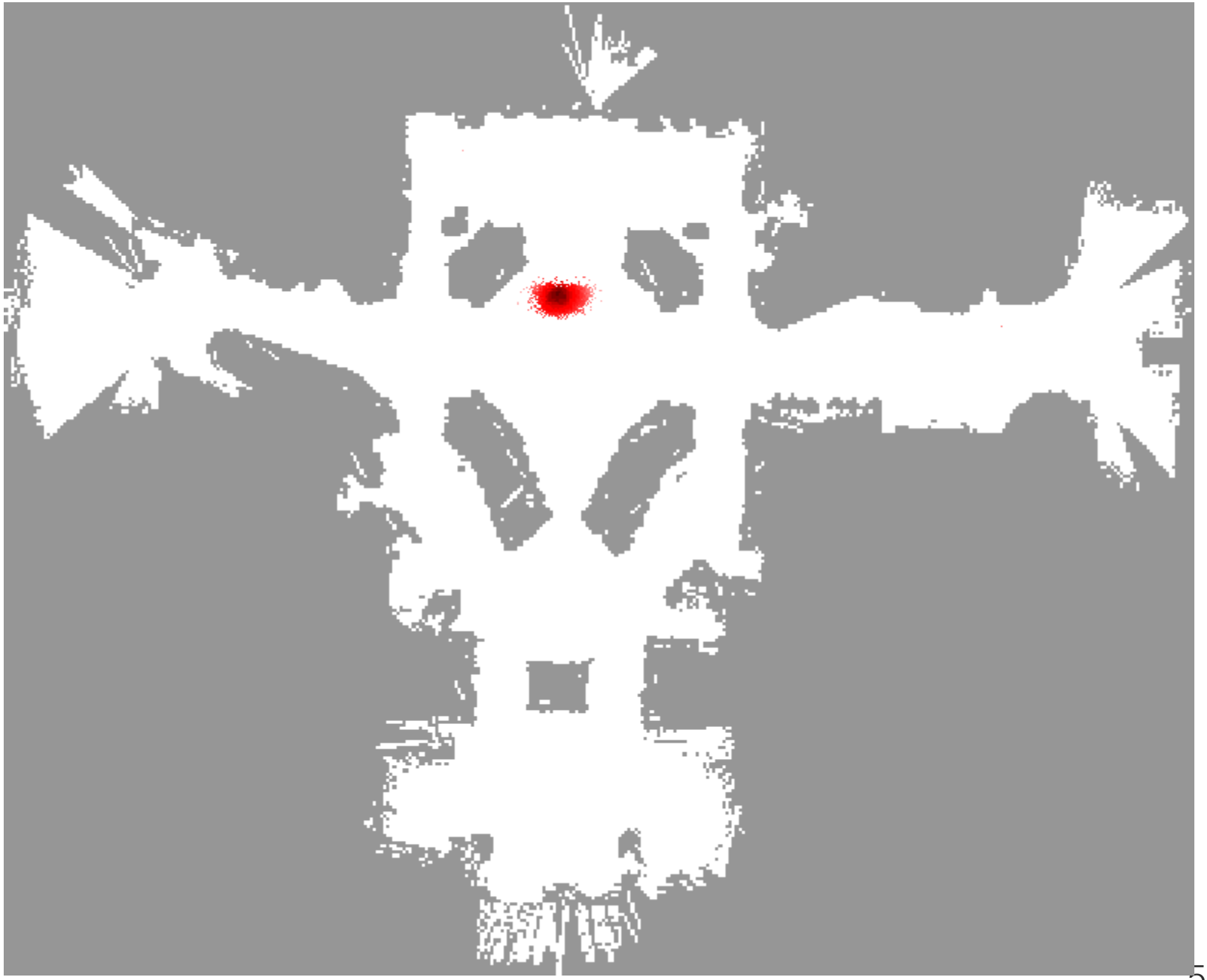
파티클 필터를 이용한 위치 추정



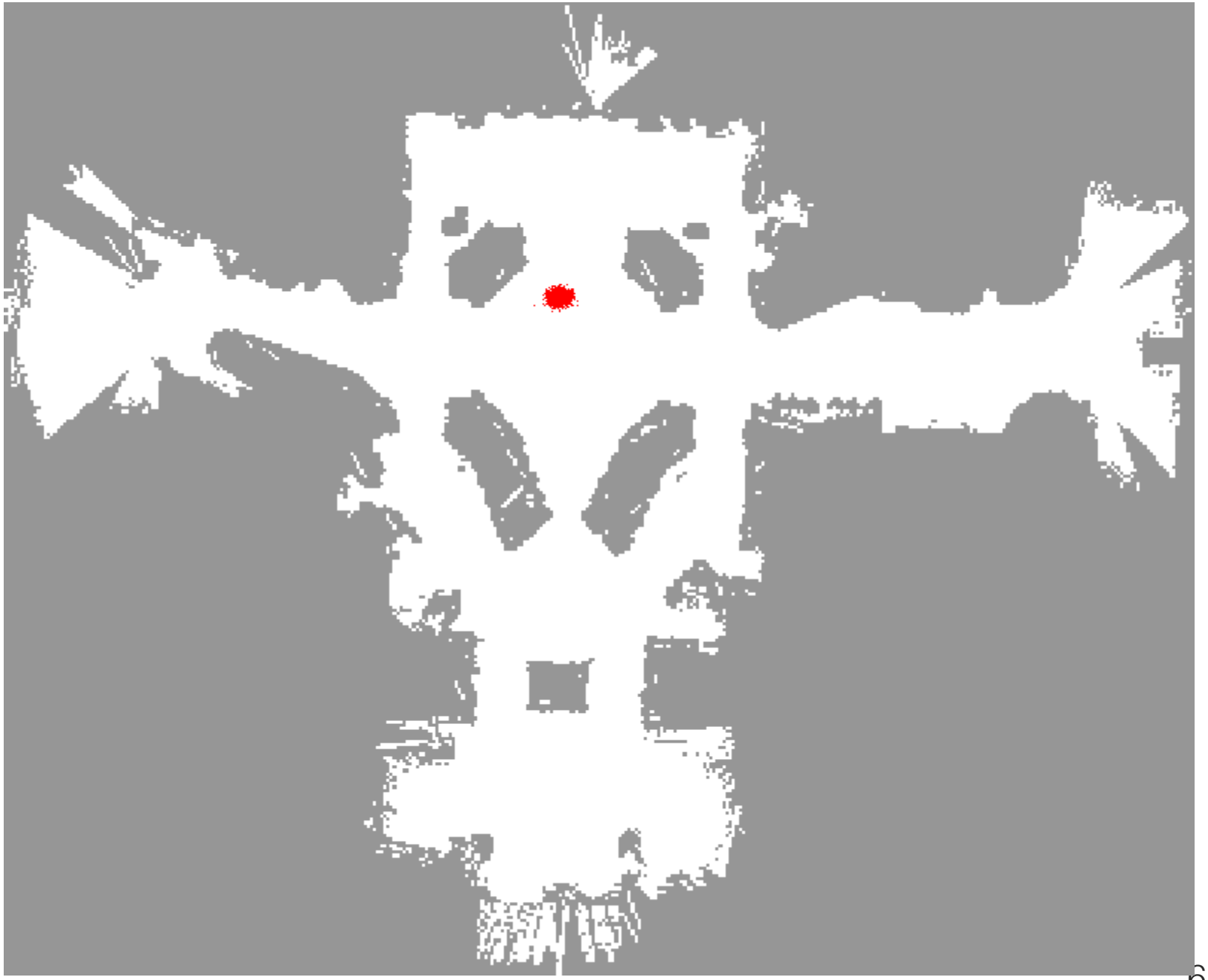
파티클 필터를 이용한 위치 추정



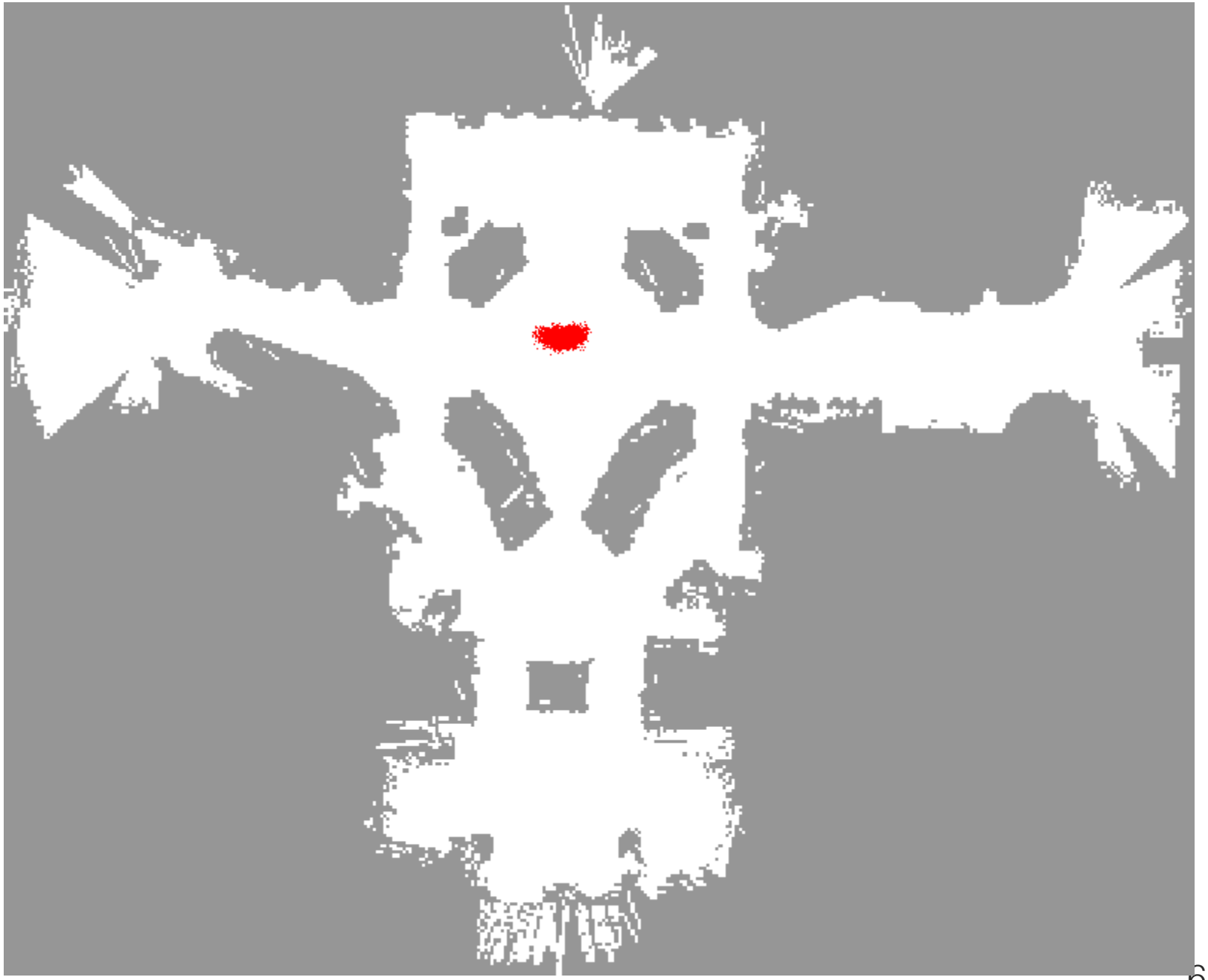
파티클 필터를 이용한 위치 추정



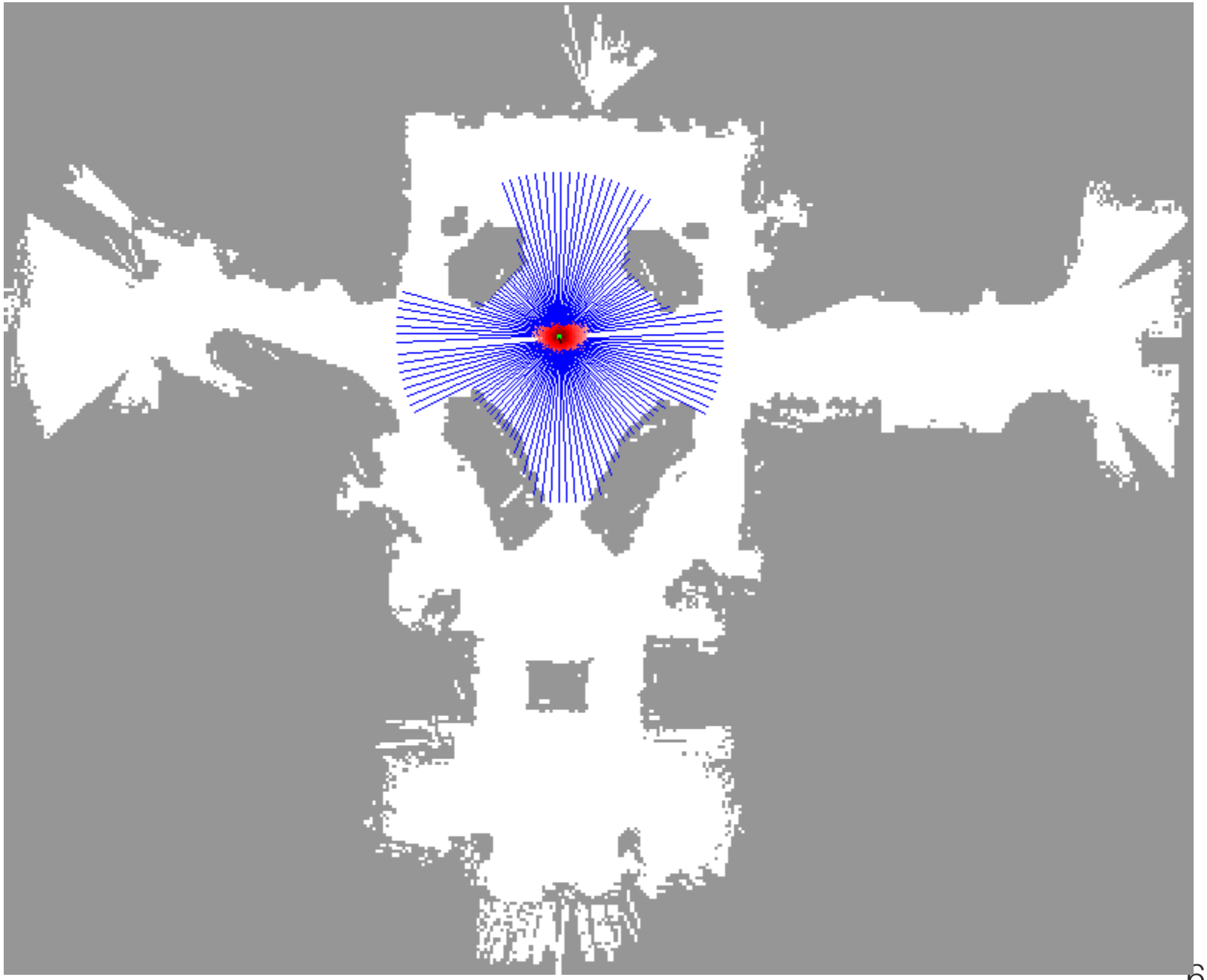
파티클 필터를 이용한 위치 추정



파티클 필터를 이용한 위치 추정



파티클 필터를 이용한 위치 추정



파티클 필터를 이용한 위치 추정