# 파티클 필터를 이용한 로봇 측위 시스템의 설계 및 구현

정종근*, 김혜숙*, 임용혁*, 김승연*, 김인철**
*경기대학교 컴퓨터과학과 학부생
**경기대학교 컴퓨터과학과 교수
e-mail:{ttigem, chia, iyh0924, kimsy0406, kic}@kgu.ac.kr

# Design and Implementation of a Robot Localization System Using Particle Filters

Jong-Geun Jung*, Hye-Suk Kim*, Yong-Hyuk Lim*, Seung-Yeon Kim*, In-Cheol Kim**
*Undergraduate Course, Dept of Computer Science, Kyonggi University
**Faculty, Dept of Computer Science, Kyonggi University

요 약

본 논문에서는 파티클 필터를 이용한 이동 로봇의 위치 추정 방법을 제안한다. 이동 로봇의 위치를 추정하기 위해알기 위해 이동 로봇에 설치되어 있는 초음파 센서를 이용하여 주변 환경과의 거리를 측정한다. 그리고 측정된 센서 값과 이동 동작의 불확실성을 고려하여, 위치 추정 오차를 줄이고자 가우스 확률분포와 파티클 필터 기법을 이용하여 이동 로봇의 위치를 추정한다. 본 논문에서는 구현된 시스템과 실험 결과를 소개한다.

## 1. 서론

일정한 작업 공간에서 돌아다니며 일을 수행하는 이동 로봇에게는 자신의 현재 위치가 어디인지를 파악하는 능력은 가장 기초적이면서도 필수적인 지능 행위이다. 비록 이동 로봇에게 자신의 작업 공간에 대한 정보 즉, 환경 지도가 미리 주어져 있다고 가정하더라도. 주변 장애물까지 거리와 같은 제한적인 센서 데이터에만 의존하여 이동 중인 자신의 현재 위치를 정확히 파악한다는 것은 매우 어려운 일이다. 특히 대부분의 이동 로봇에 존재하는 센서 데이터의 불확실성과 이동 모터 작동의 부정확성은 이 문제를 더욱 어렵게 만든다[1]. 따라서 이러한 많은 불확실성이 존재하는 가운데서도 이동 로봇의 위치를 추정해내기 위해서는 확률 기반의 측위 기법들이 매우 효과적이다. 본 논문에서는 가우스 확률 분포(Gaussian Probability Distribution)와 다수의 파티클(Particle)들을 이용해 이동 로봇의 위치를 추정하는 방법을 제안하고, 이 방법을 적용한 로봇 측위 시스템(Robot Localization System)의 구현에 대해 소개한다.

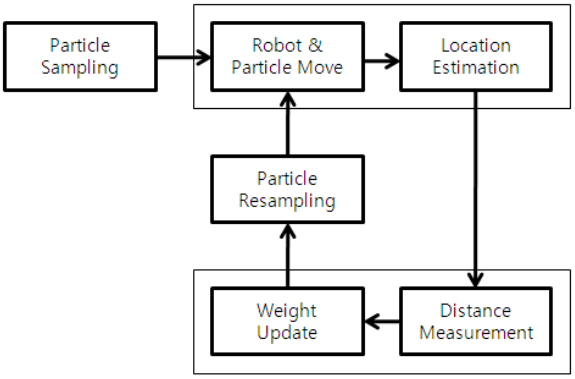
## 2. 파티클 필터링 측위 알고리즘

로봇 측위(Robot Localization)는 이미 자신이 놓여 있는 공간의 크기와 모양 등 세부적인 환경 정보 또는 환경 지도를 가지고 있는 경우와 그렇지 못한 경우로 나누어 볼 수 있다.

※ 본 연구는 경기도의 경기도지역협력연구센터사업의 일환으로 수행하였음

환경 지도를 이미 가지고 있는 경우, 로봇 측위는 측정된 센서 데이터와 환경 지도상의 물체들의 절대 위치를 이용해 환경 지도상에 로봇이 현재 어디쯤 위치하고 있는지를 추정하는 문제가 된다. 반면에, 아직 로봇이 충분히 경험하지 못한 새로운 공간인 경우, 즉 환경 지도를 가지고 있지 못한 경우, 로봇 측위는 로봇 스스로 주어진 공간을 돌아다니며 환경 지도를 작성하는 일(Mapping)과 병행할 수 밖에 없다. 이 경우의 로봇 측위 문제를 일반적으로 SLAM(Simultaneous Localization And Mapping)이라 부르며, 환경 지도의 불확실성으로 인해 로봇 측위는 더욱 어려워지게 된다.

정확한 환경 지도가 있는 경우에도, 센서 데이터의 불확실성과 이동 모터의 부정확성으로 인해 로봇 측위는 쉽지 않은 문제이다. 파티클 필터 측위법(Particle Filter Localization)은 몬테카를로 측위법(Monte Carlo Localization)으로도 불리며, 로봇 측위에 포함된 불확실성 해결에 도움이 되는 확률 기반의 측위 기법이다.




(그림 1) 파티클 필터를 이용한 로봇 측위

(그림 1)은 파티클 필터 측위 방법을 개념적으로 표현한 그림이다. 이 방법의 초기화 단계는 주어진 공간 내에 현재 로봇이 위치하고 있을 것 같은 다수의 파티클들을 임의로 샘플링하는 것으로 시작한다(Particle Sampling). 즉, 각 파티클은 로봇의 현재 위치를 추정한 하나의 후보위치를 암시한다. 파티클 생성을 통한 초기화가 끝나면, 로봇을 일정한 거리만큼 이동시키고, 로봇의 현재 위치를 암시하는 각 파티클도 로봇이 이동한 거리만큼 옮긴다(Robot & Particle Move). 이동 후 각 파티클의 새로운 위치를 계산하게 된다(Location Estimation). 각 파티클의 새로운 위치는 공간이동 후 로봇의 바뀐 위치를 암시한다. 각 파티클의 새로운 위치, 즉 로봇의 바뀐 위치가 실제 위치와 어느 정도 일치하는지 확인하기 위해 초음파 센서(Ultrasonic Sensor)와 같은 거리 측정 센서를 이용하여 주변의 벽이나 장애물까지 거리를 측정한다(Distance Measurement). 센서를 이용해 실제로 측정한 거리 측정치를 각 파티클의 가우스 확률 분포 함수에 적용하여, 로봇이 현재 해당 파티클에 위치하고 있을 확률을 구한다. 그리고 이 확률을 토대로 각 파티클의 가중치(weight)를 갱신한다(Weight Update). 갱신된 가중치를 참고하여, 가중치가 낮은 파티클들은 가중치가 높은 파티클들이나 새로운 파티클들로 교체가 된다(Particle Resampling). 로봇이 위치하고 있을 가능성이 매우 높은 소수의 파티클들로 충분히 수렴할 때까지, 이와 같은 과정을 반복한다.

<표 1> 파티클 필터링 알고리즘

1. **Inputs**:
   Distance $u_t$,
   Sensor reading $z_t$.,
   Sample set $S_t = \{(x_t(i), w_t(i)) \mid i = 1,...,n\}$

/* Update the current set of samples */
2. **for** i = 1 **to** n **do**
   // Compute new location
   i. $x_t$ = updateDist($x_t$, $u_t$)
   // Compute new weight
   ii. $w_t(i) = prob(z_t \mid x_t(i))$

/* Resample to get the next generation of samples */
3. $S_{t+1}$ = null
4. **for** i = 1 **to** n **do**
   i. Sample an index j from the distribution given by the weights in $S_t$
   // Add sample j to the set of new samples
   ii. Add $(x_t(j), w_t(j))$ to $S_{t+1}$

5. **return** $S_{t+1}$

<표 1>은 파티클 필터링 알고리즘을 나타내는 의사코드(Pseudo Code)이다. 파티클 필터링 알고리즘의 t 번째 단계를 표현하고 있는 이 의사코드에서는 n 개의 파티클들로 이루어진 샘플집합 $S_t$와 로봇의 이동 거리 $u_t$, 그리고 센서의 거리 측정치 $z_t$ 등을 입력으로 받는다. 이어서 각 파티클의 현재 위치 $x_t(i)$와 이동 거리 $u_t$를 토대로 이동 후 새로운 파티클의 위치를 계산하게 된다 (updateDist($x_t$, $u_t$)). 그리고 거리 측정치 $z_t$를 이용하여 확률 $prob(z_t \mid x_t(i))$를 계산함으로써 각 파티클의 가중치 $w_t(i)$를 갱신하게 된다. 새로 갱신된 가중치에 따라 다음 단계의 파티클 집합 $S_{t+1}$을 다시 샘플링하게 된다.

## 3. 시스템 설계

앞에서 설명한 파티클 필터 측위법을 적용한 실제 로봇 측위 시스템을 설계하고, Lego Mindstorm NXT 로봇과 Java 프로그래밍 API인 leJos NXT를 이용하여 구현하고자 한다.

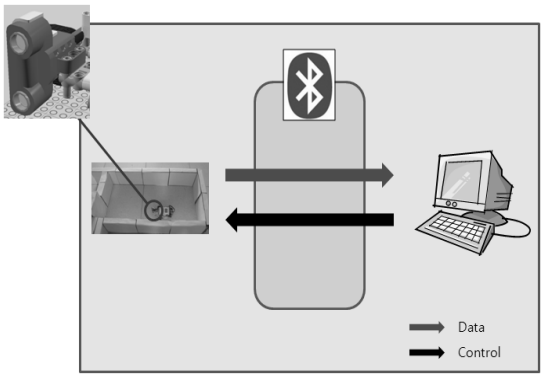
### 3.1 시스템 기능

전체 시스템의 개략적인 기능을 설명하면 다음과 같다. 초음파 센서를 장착한 로봇을 크기와 모양을 이미 알고 있는 일정한 공간속에 두고 조금씩 이동시키거나 회전시키면서 초음파 센서로 측정한 주변 벽까지 거리를 블루투스(Bluetooth)로 연결된 PC 서버로 전송하게 한다. 그러면 파티클 필터 측위 알고리즘이 내장된 PC 서버에서는 로봇에서 보내오는 거리 측정치에 기초하여 공간 내 현재 로봇이 놓여 있을 위치, 즉 파티클들의 위치와 가중치를 계산하여 화면에 그려준다. 로봇의 움직임은 측위의 효율성을 높이기 위해 로봇 스스로가 자율적으로 결정하도록 두지 않고, 공간의 특성을 잘 알고 있는 원격 PC 서버쪽의 사용자가 결정하여 이동 명령을 로봇에 전달하면 로봇은 그 명령대로 실행하는 것으로 가정한다. PC 서버쪽 화면에 표시되는 파티클들의 분포가 충분히 수렴되면, 로봇의 위치가 어느 정도 정확히 파악된 것으로 판단하여 전체 시스템의 동작을 정지한다.

(그림 2) 로봇 구성

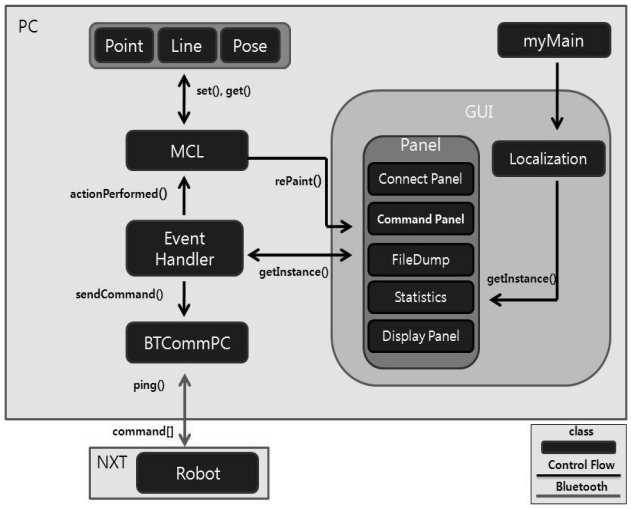
### 3.2 로봇 및 시스템 구성

로봇 구성과 전체 시스템 구성은 각각 (그림 2), (그림3)과 같다. 측위를 위한 로봇의 중요 센서는 초음파 센서(Ultrasonic Sensor)로 (그림 2)와 같이 로봇 앞쪽 장애물까지의 거리 측정을 위해 로봇의 전면부에 위치하도록 설계되었다.




(그림 3) 전체 시스템 구성

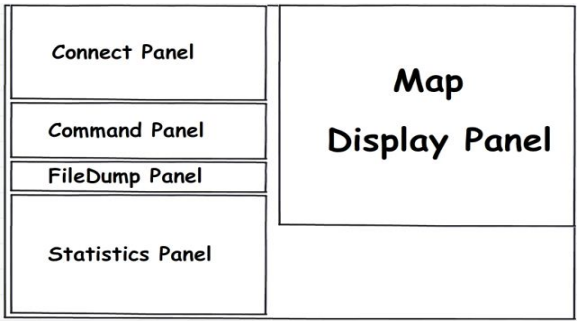
매순간 로봇이 움직여야 할 이동 거리 및 회전 각도는 사용자가 결정하며, 로봇은 바퀴 모터들을 제어하여 그만큼 이동 혹은 회전을 수행한다. 이러한 이동 로봇은 (그림 3)과 같이 블루투스를 통해 PC 서버에 연결된다. 로봇은 사용자의 명령에 따라 일정 거리만큼 이동한 후에, 초음파 센서로 장애물까지 거리를 측정한 다음, 이 거리 측정치를 서버에 전달하는 역할을 주로 수행한다. 반면에, PC 서버는 로봇이 보내준 거리 측정치를 파티클 필터 측위 알고리즘에 적용하여 로봇의 현재 위치를 추정할 뿐 아니라, 사용자에게 로봇이 현재 존재할 위치를 나타내는 파티클들을 지도상에 표시해주고, 사용자로부터 이동 명령을 받아 로봇에 전달하는 역할 등을 수행한다. (그림 3)에서 빨간색 화살표는 센서로 측정한 거리 측정치의 전송을 나타내고, 검은색 화살표는 로봇에 전달되는 사용자의 이동 명령을 나타낸다.




(그림 4) 프로그램 구성

### 3.3 프로그램 설계

(그림 4)는 파티클 필터를 이용한 로봇 측위 시스템의 프로그램 구성을 나타낸다. 프로그램은 myMain 클래스에서 시작한다. myMain에서 Localization을 호출하면 GUI환경을 담당하는 다섯 개의 패널(Panel)을 호출한다. 첫 번째로 Connect Panel) 로봇과 PC를 연결하는 클래스로 Connect 버튼을 누르면 연결 할 로봇의 주소를 넣는다. BTCommPC 클래스는 위 패널에서 정해진 주소를 가지고 로봇과 PC를 연결하여 데이터(data)와 컨트롤(control)을 주고 받을 수 있도록 하였다. Command Panel은 사용자가 로봇의 동작을 명령할 수 있는 클래스이다. 사용자가 내린 명령은 EventHandler의 _command배열에 상수 값으로 저장된다. 저장된 명령 값의 배열은 BTCommPC 클래스의 sendCommand 메소드를 통하여 로봇에 전달된다. 사용자가 내린 명령을 동작하면 로봇은 Ping 명령어를 통하여 Ultrasonic Sensor를 작동시킨다. 로봇이 측정한 값은 BTCommPC 클래스가 받아 EventHandler로 보낸다. FileDump 클래스는 로봇의 측정 값 및 프로그램에서 계산된 좌표 값, 로봇이 바라보고 있는 각도와 계산 값의 유효성에 대해 사용자가 지정한 경로에 텍스트 파일로 저장하도록 하였다. Statistics 클래스는 EventHandler 클래스를 통해 받아온 값을 저장하고 getDistance 메소드를 통하여 측정된 거리 값을 타 클래스에서 호출하여 사용할 수 있도록 하였다. 마지막으로 Display Panel에서는 MCL 클래스에서 계산된 로봇의 예상 위치를 그려준다. MCL은 실제 파티클 필터(Particle Filter) 알고리즘을 계산하는 클래스이다. 사용자가 내린 동작과 로봇이 측정한 값을 바탕으로 Sample Set의 가중치를 결정한다. 결정된 가중치는 다음의 Sampel Set을 결정한다. Sample Set이 결정되면 Display Panel의 repaint 메소드의 호출하여 사용자에게 Sample Set의 변화를 보여준다.

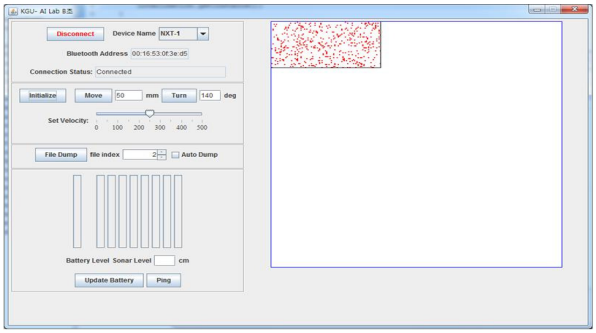



(그림 5) 화면 구성

### 3.4 사용자 인터페이스 설계

로봇 측위 시스템의 사용자 화면은 (그림 5)와 같이 연결 패널(Connect Panel), 명령 패널(Command Panel), 파일덤프 패널(FileDump Panel), 통계 패널(Statistics Panel), 출력 패널(Display Panel) 등 총 다섯 개의 패널들로 구성되도록 설계하였다. 연결 패널은 NXT의 MAC Address, 로봇과 PC의 블루투스 연결 여부 등을 표시해준다. 명령 패널은 초기화(Initialize) 버튼, 이동(Move) 버튼, 회전(Turn) 버튼 등으로 구성된다. 초기화 버튼은 임의로 파티클들을 생성한 다음, 출력 패널에 선택된 파티클들의 초기 위치를 보여준다. 사용자는 이동 버튼 및 회전 버튼을

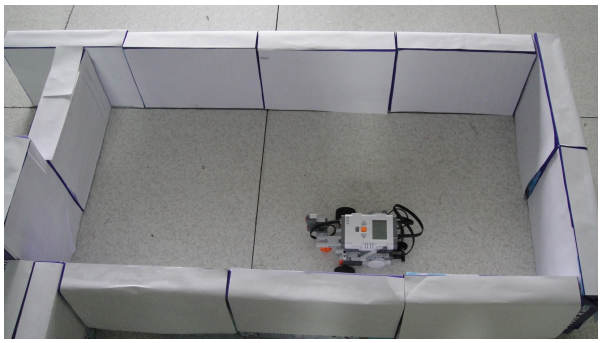
이용하여 로봇의 움직임을 원격 제어할 수 있다. 파일덤프 패널은 측정한 센서 값과 계산된 좌표 값, 각도와 이 값들의 유효성을 텍스트 파일에 저장 시킨다. 통계 패널은 로봇의 배터리 레벨과 센서 거리 측정치를 표시해준다. 출력 패널은 주어진 공간 내 현재 로봇의 위치를 추정한 파티클들을 표시해준다. 사용자는 측위가 진행되는 동안 출력 패널을 통해 변화되는 파티클들의 위치와 분포를 파악할 수 있다.




(그림 6) 로봇 측위 시스템의 사용자 화면

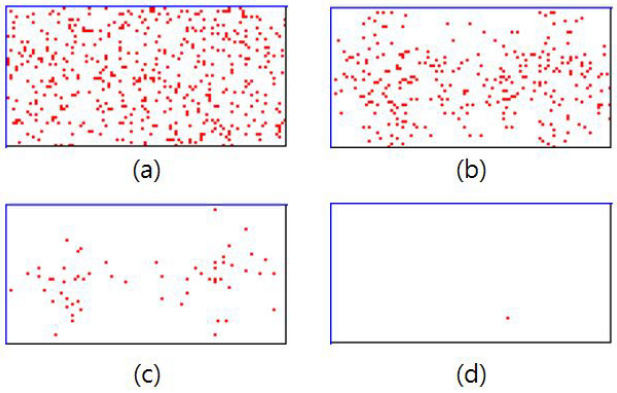
## 4. 구현 및 실험

(그림 6)은 최종적으로 구현된 로봇 측위 시스템의 사용자 화면을 보여주고 있다. 구현된 로봇 측위 시스템을 이용해 (그림 7)과 같은 직사각형 폐쇄 공간 안에서 로봇을 움직이면서 로봇의 위치를 추정하는 측위 실험을 전개해 보았다. (그림 8)은 파티클 필터 측위 과정이 수행되는 동안 로봇의 추정 위치를 나타내는 파티클들의 변화를 보여준다.



(그림 7) 로봇 측위 실험 환경

(그림 8)의 (a)는 초기화 단계에서 임의로 선택된 파티클들의 분포를 나타내고, (b)와 (c)는 초기화 이후 일정한 만큼의 로봇 이동이 이루어진 뒤, 변화된 파티클들의 분포를 보여준다. (d)는 최종적으로 하나의 파티클로 수렴된 결과를 보여주며, 이 파티클이 나타내는 로봇의 최종 추정 위치는 로봇의 실제 위치와도 매우 근접하다는 것을 확인할 수 있었다.




(그림 8) 파티클들의 변화

## 5. 결론

본 논문에서는 센서 데이터의 불확실성과 이동 모터의 부정확성을 극복하기 위해 가우스 확률 분포와 다수의 파티클들을 이용하는 로봇 측위 방법을 제안하였고, Lego Mindstorm NXT 로봇을 이용한 실제 로봇 측위 시스템의 구현에 대해 소개하였다. 그리고 실험을 통해 파티클 필터 로봇 측위 시스템의 효과를 확인할 수 있었다. 하지만 동시에 로봇 바퀴와 바닥 사이의 마찰 정도에 따라 로봇 이동 동작의 부정확성이 매우 커져서 측위 오차의 주원인이 됨을 알 수 있었다. 향후 연구를 통해 마찰에 의한 오차를 줄이거나 방향센서(Compass Sensor) 등을 추가함으로써 측위의 정확도를 개선시킬 수 있을 것으로 판단한다.

## 참고문헌

[1] Dieter Fox, Wolfram Burgard, Frank Dellaert and Sebastian Thrun, “Monte Carlo Localization: Efficient Position Estimation of Mobile Robots”, Proc. of AAAI-1999, 1999.

[2] Jeong Woo, et al, "Localization of Mobile Robot using Particle Filter", Proc. of SICE-ICASE-2006, 2006.

[3] Brian Bagnall, Maximum Lego NXT: Building Robots with Java Brains, Varinat Press, 2007.

[4] Brian Bagnall, et al, leJos: Java for Lego Mindstorms, http://lejos.sourceforge.net, 2009.

[5] Myles McNally, Frank Klassner and Christopher Continanza, "Exploiting MindStorms NXT: Mapping and Localization Projects for the AI Course", Proc. of FLAIRS-2007, 2007.