ROS 시스템 기반의 마라톤 보조 로봇 설계 및 개발

변병문*, 장희덕*, 이하영*, 고상민*
*경북대학교 전자공학부
bm4655@naver.com, gmlejr456@naver.com, dlgkdud4460@naver.com,
rudrhd7107@gmail.com

Design and development of marathon assistant robot based on ROS system

Byung-Moon Byun*, Hee-Deok Jang*, Ha-Young Lee*,
Sang-Min Ko*
*Dept. of Electronics Engineering, KyungPook National University

요 약

마라토너들이 대회에서 높은 성적을 받기 위해서는, 다른 선수를 위해 속도를 조율하여 선수들을 위해 전략적 희생을 하는 페이스메이커의 존재가 필수적이다. 그러나 기존의 페이스메이커는 사람이 수행한다는 점에서 한계점을 가지고 있다. 본 논문에서는 페이스메이커 기능을 수행할 수 있는 로봇을 설계 및 구현하여, 기존의 한계를 극복하고자 하였다.

1. 서론

장거리 주행을 하는 마라톤에서 페이스메이커의 존재는 필수적이다. 페이스메이커는 단어가 지닌 의미 그대로 마라토너들의 페이스를 조절해준다. 마라토너의 앞과 옆에서 함께 주행하여 러너의 속도를 조율해줌과 동시에 의지를 북돋워 장거리 경주에서 최고의 기록을 달성할 수 있게 해준다. 그뿐만 아니라 페이스메이커는 등에 완주 시간대를 표시하거나 풍선을 달아 일종의 시간 기준점의 역할을 한다. 이처럼 페이스메이커는 마라톤에서 다양한 역할을 수행할 뿐만 아니라 기록 단축에 있어 중요한 요소로 여겨진다.

그러나 현재, 페이스메이커는 사람이 직접 수행한다는 점에서 한계점을 가지고 있다. 페이스메이커도 사람이기 때문에 주행속도가 일정하지 않으며 완주에 있어서 오차가 발생한다. 기록을 1초라도 줄이기 위해서는 오차가 없는 정확한 주행 시간을 토대로 경주와 연습이 이루어져야 한다. 이러한 기존의 사람이 수행하는 페이스메이커의 한계점을 극복하기 위해 본 논문에서는 마라토너를 위한 '페이스메이커(사용자)', '페이스메이커(기본)' 이 두 가지 주요 기능을 탑재한 페이스메이커 로봇을 설계, 개발하였다.

로봇은 크게 두 가지의 기능을 수행할 수 있다. 첫째, 지정한 경로를 일정 속도로 주행할 수 있다. 사용자가 주행 코스의 지도 정보를 입력하면(혹은 사용자가 지정한 경로를 로봇이 1회 주행하여 SLAM을 통해 코스 지도를 작성하면) 입력된 코스를 사용자가 설정한 속도로 일정하게 주행한다.-{페이스메이커(기본) 기능} 둘째, 사용자를 인식하여 인식한 사용자를 기준으로, 일정 거리를 유지하며 사용자 앞에서 주행하는 페이스메이커의 기능을 한다. 픽시캠과 픽시캠의 이미지 정보를 활용할 수 있는 어플리케이션을 통해 사용자를 인식 및 설정한 뒤, 카메라 화면에 나타나는 사용자의 픽셀 크기(가로 x 세로)에 맞춰 거리를 감지한다. [1] 사용자가 로봇과 가까워지면 픽셀 크기는 증가할 것이며, 이에 따라 로봇은 사용자와의 거리를 늘리기 위해 속력을 올리게 되고 사용자와 로봇의 거리는 일정 거리를 유지하게 된다. 반대로, 로봇과 사용자가 멀어진다면 사용자의 픽셀 크기는 작아질 것이고, 이에 따라 로봇은 속도를 줄여 사용자와 가까워지게 된다.-{페이스메이커(사용자) 기능}

본 논문에서 로봇은 크게 H/W 부분과 S/W 부분으로 나누어지며 이들 사이에서 통신은 ROS 시스템을 기반으로 이루어져 있다. H/W는 라즈베리 파이, OpenCR 다이나믹셀, 픽시캠 그리고 LiDAR 등으로 구성되어 있다. 라즈베리 파이의 설치된 라즈비안 운영체제에서 실행되는 ROS 마스터 노드를 중심으로 모든 H/W 부분과 S/W 의 통신이 이루어진다. OpenCR에 부착된 여러 센서들은 OpenCR에 설치된 펌웨어를 기반으로 제어가 이루어진다. 센서들로부터 전달된 데이터는 OpenCR에서 ROS 노드의 토픽 형태로 라즈베리파이로 전달된다. LiDAR는 로봇 주위의 2차원 데이터를 수집하여, 이를 바탕으로 SLAM을 통한 지도를 생성할 수 있다. 마지막으로 픽시캠은 내장된 색조 기반의 객체 인식을 이용하여 사용자 인식을 수행한다. 전용 프로세서를 기반으로 이미지 센서와 결합하여 비교적 빠르고 가볍게 물체를 학습할 수 있다는 장점이 있다. S/W 부분에는 로봇을 제어하기 위한 어플리케이션과 어플리케이션을 동작시키기 위한 디바이스가 있다. 이 어플리케이션은 안드로이드 기반의 어플리케이션으로, 픽시

캠으로 받은 이미지 데이터를 통해 사용자를 등록하여 주고, 로봇을 직접 조작하여 커스텀 주행을 가능하게 한다.

앞서 말한 기능들을 활용함으로써, 사람이 수행하던 페이스메이커에서 로봇이 수행하는 페이스메이커로 주체를 옮겨 기존 페이스메이커의 한계점을 극복하고 마라토너들의 주행 기록 단축을 효과를 가져올 수 있다.

2. 관련연구

2.1 ROS(로봇 운영체제, Robot Operating System)

로봇 운영체제(ROS)는 로봇 동작에 있어서 각 프로세스 간 메시지전달과 패키지 관리에 사용되는 로봇 전용 메타 운영 체제이다. [2], [3] 그뿐만 아니라 개발환경에 필요한 다양한 라이브러리와 디버깅 도구를 통하여 개발 시간을 단축시켜준다. 이기종과의 통신이 가능하다는 장점을 필두로 현재 다양한 기업, 연구실에서 ROS 기반의 연구가 활발하게 진행되고 있다. 본 논문에서는 ROS 시스템을 기반으로 로봇의 여러 센서로부터 데이터를 받아 제어하고, 어플리케이션과 통신을 가능하게 하는 역할을 수행한다.

2.2 SLAM(Simultaneous Localization and mapping)

SLAM은 로봇이 임의의 공간에서 주위를 이동하며 주변을 탐색하여, 수집한 정보를 바탕으로 그 공간의 지도를 그리고 동시에 로봇 자신의 위치를 정하는 방식이다. 주변 데이터 수집에는 비전 센서, LiDAR 등이 사용되며 Odometry 방식을 기초로 하여 주위의 특징점을 찾아 주변을 Localization 하여 현재 위치를 찾아내는데 사용된다. 현재는 Raw data를 가공하는 전처리 알고리즘과 함께 Feature Detection, Optimisation 알고리즘 등 SLAM과 동시에 수행되는 다양한 연구들이 진행되고 있으며 딥러닝을 적용한 기술도 등장하고 있다. 또한 하드웨어적으로는 멀티스레드를 처리할 수 있는 GPU 기반의 병렬처리 코어 및 저전력 반도체를 이용한 SLAM 기술이 연구 개발 중이다. 향후 자율주행 자동차, 자율주행 드론, 자율주행 로봇 등 자율주행 기술에 기반이 되는 기술로 자리매김 중이다. 본 논문에서는 G-mapping 방식을 사용하여 주행할 트랙의 지도를 그리고 이를 바탕으로 로봇의 현재 위치를 스스로 파악하여 자율주행을 가능하게 한다. [4]

2.3 다이나믹셀(Dunamixel)

다이나믹셀은 감속기, 제어기, 구동부, 통신부 등 로봇의 관절에 필요한 기능들이 하나로 합쳐진 일체형 구동장치이다. 고유의 ID에 따라 패킷 통신으로 제어되어 동시다발적인 정밀한 수행이 가능하다. [5] 정밀한 제어 및 강한 출력으로 로봇의 관절 및 SLAM 지도 작성용 로봇에 사용된다. 다이나믹셀은 아두이노와 연결하여 사용할 수 있지만, 본 논문에서는 로보티즈에서 제공하는 ROS 통신에 최적화된 OpenCR에 연결하여 사용한다.

2.4 어플리케이션(Application)

어플리케이션 기반의 많은 연구들이 진행되고 있을 뿐만 아니라 어플리케이션을 활용한 연구들 또한 활발히 진행되고 있다. [6] 본 논문에서는 안드로이드 기반의 어플리케이션을 제작하여 ROS Master 노드와의 통신을 이용한다. 이를 바탕으로 로봇에 대한 위치 정보 및 주행 데이터를 실시간으로 전송받고, 주행 경로를 설정하여 로봇이 일정 경로를 주행할 수 있게 한다.

2.5 PUMA-BeatBot

라인 트레이서 및 컴퓨터 비전 시스템을 이용한 육상 단거리 전용 페이스 조절 로봇이다. 시속 30마일에 이르는 자율 경주용 로봇으로 2016년 하계 올림픽을 앞두고 PUMA에서 개발한 제품이다. 그러나 트랙 중심의 시스템과 단거리 전용이라는 한계점이 존재한다.

3. 설계 및 구현

3.1 전체 시스템 개요

시스템의 전체적인 구조는 H/W 및 S/W 부분과 두 부분을 연결하는 ROS 시스템으로 구성되어 있다. (그림 1) H/W 부분에는 로봇의 주변 환경 및 사용자 감지와 차체 구동을 위한 센서들이 부착되어 있고, 차체 구동은 ROS 통신을 통해 받아들인 정보를 바탕으로 라즈베리 파이에 의해 제어된다. S/W 부분에는 사용자 디바이스의 어플리케이션을 통해 사용자가 원하는 동작을 하기 위한 명령을 ROS 시스템을 통해 H/W부분으로 전달한다.

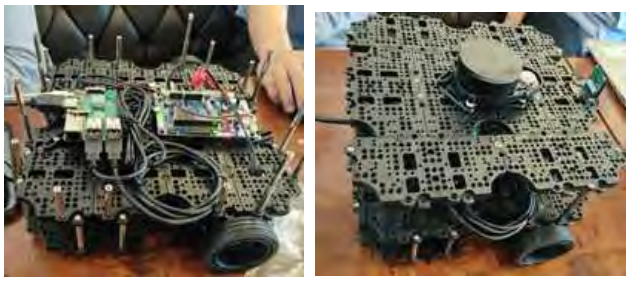
본 논문에서는 H/W 와 S/W를 연결하는 통신시스템으로 ROS 시스템의 통신을 사용하였으며, 마스터 노드를 통해 S/W 어플리케이션으로부터 전달된 동작 명령을 H/W 부분의 라즈베리파이에서 처리하여 부착된 센서들을 제어하였다.

(그림 1) 전체 시스템 개요도

3.2 H/W 부분 설계

모든 센서들은 OpenCR과 연결되어 있고, OpenCR과 연결된 라즈베리 파이에서 ROS 시스템을 통해 각 센서들의 데이터를 전달받거나 동작 명령을 내릴 수 있다. LiDAR는 주위 환경을 주기적으로 감지하여 장애물을 탐지하기 위해 로봇 최상단에 위치시켰고, 픽시캠은 뒤따라 오는 사용자를 인식하고 사용자와의 거리를 측정해야 하므로 로

봇의 후방에 배치하였다. 구동을 위한 서보모터는 전륜2륜에 다이나믹셀을 사용하였고, 로봇의 구동 안정성과 무게중심을 위해 가장 무거운 배터리를 최하단에 배치하였다. (그림 2)



(그림 2) H/W 부분

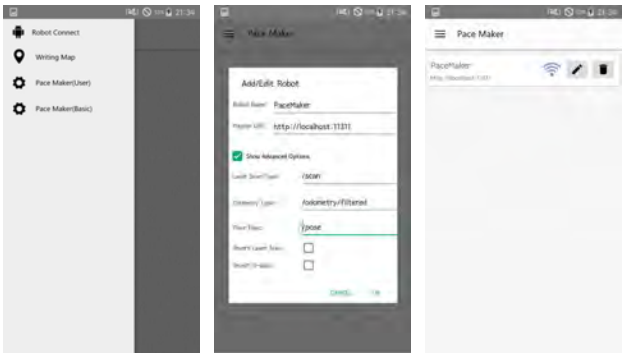
3.3 S/W 부분 설계

어플리케이션을 통한 로봇 제어를 위해서는 로봇과 ROS 시스템 연결이 우선시 된다. ROS 시스템 연결이 성공하면 LiDAR를 통해 감지한 로봇 주변의 위치를 실시간으로 시각화할 수 있다. 주행을 위한 지도는 '지도작성-로봇조작' 기능을 통해 사용자가 로봇을 구동시켜 주변 환경 지도를 그려주어야 한다. 지도가 준비되면 사용자와의 거리를 유지하며 경로를 지정하는 '페이스메이커(사용자)' 기능과 지정한 경로만을 주행하는 '페이스메이커(기본)' 기능을 통해 로봇을 구동시킨다. 주행에 필요한 경유 지점과 목표 지점은 사용자의 화면 터치입력을 받고, 주행속도는 다이나믹셀의 최대속도로 설정되어 있다. '페이스메이커(사용자)' 기능에 필요한 사용자 인식은 '사용자지정' 기능을 통해 사용자의 색을 기반으로 사용자를 인식할 수 있고, 주행에 필요한 사용자와의 거리는 100cm로 기본 설정되어 있다. (그림 3)

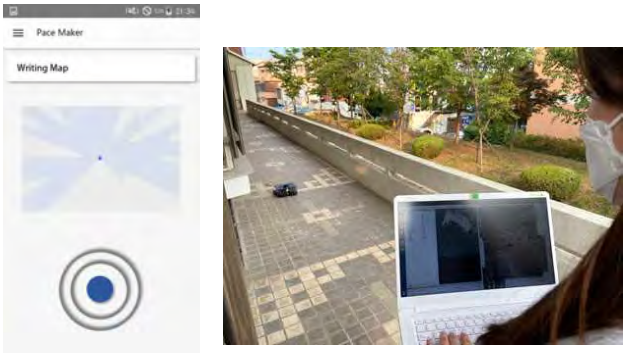
(그림 3) S/W 부분 시스템 구조도

3.4 동작 구현

어플리케이션을 첫 화면은 (그림 4)의 왼쪽과 같은 화면으로 시작된다. 왼쪽 위의 메뉴 배너를 통해 로봇 연결, 지도작성, 페이스메이커(사용자), 페이스메이커(기본)의 기능을 사용할 수 있다. H/W 부분과 S/W 부분을 연결하는 ROS 시스템을 사용하기 위해서는 'Robot Connect' 기능을 사용하여 접속할 마스터 노드의 URI를 설정해주어야 한다. 마스터 노드와 연결이 정상적으로 이루어지면 (그림 4)의 오른쪽과 같이 연결되었음을 확인할 수 있고, 연결되어야지만 '지도작성', '페이스메이커' 기능들을 사용할 수 있다.

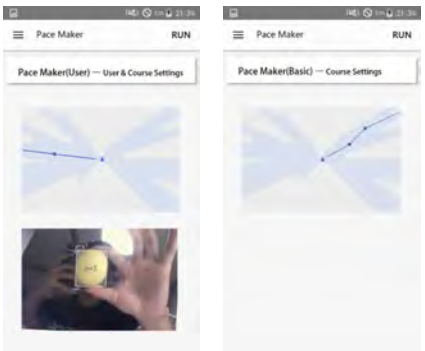


(그림 4) 어플리케이션 시작 및 마스터 노드 설정 화면



(그림 5) '지도작성-로봇조작' 기능 화면(외쪽)과 로봇의 동작 모습(오른쪽)

(그림 5)의 왼쪽 화면은 '지도작성-로봇조작' 기능을 통해 로봇의 주변 환경을 그리는 그림이고 오른쪽은 어플리케이션을 통해 조작되는 로봇의 모습과 노트북으로 로봇의 정상작동을 모니터링 하는 과정이다. 어플리케이션 하단부의 원형 조이스틱을 드래그하여 로봇을 동작시킬 수 있고, 로봇이 감지하는 주의 환경은 어플리케이션과 노트북에서 모니터링하며 결과를 확인할 수 있다. 로봇이 움직일수록 주위 환경지도는 점점 넓어지고 주행을 위한 지도를 얻을 수 있다.



(그림 6) '페이스메이커(사용자)' 기능(왼쪽)과 '페이스메이커(기본)' 기능 화면(오른쪽)

(그림 6)의 왼쪽 화면은 페이스메이커(사용자) 동작 화면으로 상단에 있는 지도에 경유 지점과 목표 지점을 설정한 뒤 오른쪽 위의 'RUN' 버튼을 통해 '경로지정-주행 기능' 을 동작시킬 수 있다. 하단에는 픽시캠의 실시간 화면으로 학습할 색을 드래그하여 사용자의 색을 기반으로 사용자를 인식하고 사용자와의 거리를 계산할 수 있다. (그림 6)의 오른쪽 화면은 페이스메이커(기본) 화면으로, '경로지정-주행' 기능 화면을 보여주며 위의 '페이스메이커(사용자)' 기능과 같이 'RUN' 버튼을 통해 로봇을 동작시킬 수 있다.

(그림 7) 사용자와 거리를 유지한 채 주행하는 로봇(왼쪽)과 지정한 경로를 일정속도로 주행하는 로봇(오른쪽)

'페이스메이커(사용자) 기능을 동작시키면 로봇은 사용자와 일정 거리를 유지하기 위해 속도를 조절하며 경로를 주행하게 된다. 사용자와의 거리조절을 위해 로봇 동작을 위한 다이나믹셀의 선속도와 각속도의 입력값은, 사용자와의 거리가 설정한 거리(100cm)의 ±20cm를 넘어서게 되면, 사용자와의 거리가 범위(+80cm~+120cm) 내로 측정되기 전까지 입력값의 60%를 속도 값으로 재설정한다. 사용자와의 거리가 멀어지면 로봇의 속도가 감소하는 것을 확인할 수 있고, '페이스메이커(사용자)' 기능이 정상작동함을 확인할 수 있다. 그러나 로봇의 주행속도를 최대 속도로 설정하여도 로봇이 주행하는 경로의 요철과 마찰로 실제 로봇의 속도는 설정값에 미치지 못한다는 한계가 있었다.

4. 결론

본 논문에서는 LiDAR, 픽시캠 등의 센서를 활용하여 자율주행을 구현할 수 있는 H/W 부분을 어플리케이션 S/W 부분으로 제어하여, 페이스메이커 기능을 수행할 수 있는 로봇을 설계 및 구현하였다. LiDAR는 주변 환경을 탐지하여 장애물을 구별하여 충돌 없는 주행을 가능하게 하였고, 픽시캠은 이미지 데이터를 사용하여 사용자를 객체로 인식하고 사용자와의 거리를 계산함으로써 일정 거리를 유지할 수 있게 해주었다. 실제 자동차의 자율주행에 활용되는 여러 기술을 모바일 로봇에 적용하여, 특정 상황의 한계점을 극복해내는 기술로 활용하였고, 어플리케이션을 통한 기능 수행으로 사용자가 사용하기 쉬운 플랫폼을 제공해 주었다.

향후 연구내용으로는 로봇의 구동 속도를 높여 실제 달리는 사람만큼의 구동 속도를 구현하고, 또한 색조 기반의 객체 인식방법을 벗어난 강화된 사용자 객체인식 방법을 통하여 사용자와 로봇 간의 거리 유연성을 증가시킬 것이다. 개선된 로봇 시스템은 마라톤을 준비하는 사람들의 기록향상을 가져올 수 있어, 실제 마라톤 경주에서도 활용될 수 있을 것으로 기대된다.

'본 논문은 과학기술정보통신부 정보통신창의인재양성사업의 지원을 통해 수행한 ICT멘토링 프로젝트의 결과물입니다.'

참고문헌

[1] M. F. Ahmad, H. J. Rong, S. S. N. Alhady, W. Rahiman and W. A. F. W. Othman, "Colour tracking technique by using pixy CMUcam5 for wheelchair luggage follower," 2017 7th IEEE International Conference on Control System, Computing and Engineering (ICCSCE), Penang, 2017, pp. 186-191, doi: 10.1109/ICCSCE.2017.8284402.

[2] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., ... & Ng, A. Y. (2009, May). ROS: an open-source Robot Operating System. In ICRA workshop on open source software (Vol. 3, No. 3.2, p. 5)

[3] 표윤석, 조현철, 정려운, 임태훈 저, "ROS 로봇 프로그래밍", 루비페이퍼, 2017년 08월 18일

[4] B. L. E. A. Balasuriya et al., "Outdoor robot navigation using Gmapping based SLAM algorithm," 2016 Moratuwa Engineering Research Conference (MERCon), Moratuwa, 2016, pp. 403-408, doi: 10.1109/MERCon.2016.7480175.

[5] Araújo, Mateus Amarante. "Aplicação do ROS no controle de movimento de robôs equipados com motores Dynamixel em Linux de Tempo Real." (2017).

[6] Rogers, R., Lombardo, J., Mednieks, Z., & Meike, B. (2009). Android application development: Programming with the Google SDK. O'Reilly Media, Inc..