Journal of Korea Multimedia Society Vol. 19, No. 4, April 2016(pp. 747-755)
http://dx.doi.org/10.9717/kmms.2016.19.4.747

설계 패턴을 이용한 모바일 파워 카트의 유연한 아키텍처 구현

이종민†, 김성우††, 권오준†††

Implementation of a Flexible Architecture for a Mobile Power Cart Applying Design Patterns

Jong Min Lee†, Seong Woo Kim††, Oh Jun Kwon†††

ABSTRACT

Automated guided vehicles have been used for a long time to increase work efficiency in the logistics field, but it is difficult to apply to a variety of logistics sites due to either the restricted movement mechanism or expensive devices. In this paper, we present a flexible software architecture that is hardware-independent for a mobile power cart of the follow mode and implement it using a ROS software platform. Through the SCV analysis for the system functionalities, we design a package to track a user movement and a package to control a new hardware platform. It has an advantage to use a variety of movement algorithms and hardware platforms by applying the strategy pattern and the template method pattern for the design of a software architecture. Through the performance evaluation, we show that the proposed design is maintainable in terms of a software complexity and it detects a user's movement by obtaining a user skeleton information so that it can control a hardware platform to move at a certain distance.

Key words: Mobile Power Cart, ROS, SCV Analysis, Follow Mode, Design Pattern

1. 서 론

오래전부터 사무실, 슈퍼마켓, 공장 창고 같은 물류 현장에서 작업자의 안전과 함께 효율적인 물품 운반을 위하여 무인반송차와 같은 기술이 도입되어 사용되고 있다[1]. 작업 효율을 높이기 위하여 물품 이동을 보다 쉽게 해주는 무인반송차 기술은 수동 원격 조정 방식, 자율 주행 및 충돌 회피가 가능한 위치 기반 추적 방식, 자기테이프 등을 설치해 이를 따라가는 유도선 추적 방식 등의 다양한 방식이 사용되고 있다[2]. 그러나 수동 원격 조정 방식이나 자기 유도 방식의 경우 이동 방식에 제한이 있고, 위치 기반 추적 방식은 고가의 센서를 필요로 하여 다양한 물류 현장에 적용하기 어려운 실정이다[2].

물류 현장에서는 작업자가 밀거나 당겨서 이동하는 수동 카트가 여전히 많이 사용되어 오고 있어 수동 카트에 앞서 언급한 기술을 적용할 경우 작업을 위한 단말과 적재 능력을 향상시켜 보다 효과적인

※ Corresponding Author : Jong Min Lee, Address: (47340) Eomgwang-ro 176, Busanjin-gu, Busan, Korea, TEL : +82-51-890-1736, FAX : +82-505-182-6905, E-mail : jongmin@deu.ac.kr
Receipt date : Jan. 29, 2016, Revision date : Feb. 17, 2016
Approval date : Mar. 3, 2016

† Dept. of Computer Software Engineering, Dong-Eui University
†† Dept. of Computer Software Engineering, Dong-Eui University (E-mail : libero@deu.ac.kr)
††† Dept. of Computer Software Engineering, Dong-Eui University (E-mail : ojkwon@deu.ac.kr)

물품 이동 및 관리가 가능해진다. 본 논문에서는 기존의 수동 카트와 프로그램에 의한 주행 기술을 결합한 형태를 모바일 파워 카트(mobile power cart)라고 명명한다. 본 논문에서는 작업자를 인식하여 특정 작업자의 이동 경로를 따라서 카트가 이동하는 추적 유형의 모바일 파워 카트를 구현하기 위한 소프트웨어 아키텍처에 대한 연구를 다룬다.

모바일 파워 카트를 구현하기 위한 방법으로 독자적인 소프트웨어를 개발하거나 OSRF(Open Source Robotics Foundation)의 ROS(Robot Operating System), 마이크로소프트사의 MRDS (Microsoft Robotics Developer Studio), 유럽에서 개발된 OROCOS 등의 여러 로봇 소프트웨어 플랫폼을 활용할 수 있다[3]. 이중 ROS 플랫폼은 2007년 스탠포드 대학 인공지능 연구소의 Switchyard라는 시스템에서 시작되어 현재 OSRF 재단에서 관리하고 있는 BSD 라이센스 기반의 로봇 운영 체제이다. ROS 플랫폼의 주목적은 로보틱스 연구 개발에서 코드 재사용을 가능하게 하는 것으로, 새 하드웨어 플랫폼 개발 시 세부 기능 구현에 기존 패키지를 적용하는 것이 쉬운 구조이다[4].

본 논문에서는 이러한 ROS 소프트웨어 플랫폼을 사용하여 3차원 깊이 카메라를 사용하는 작업자 추적 방식의 모바일 파워 카트를 효과적으로 구현할 수 있는 소프트웨어 아키텍처를 제안하고 이를 구현한다. 모바일 파워 카트에 필요한 시스템 기능에 대한 SCV (Scope, Commonality and Variability) 분석[5]을 통하여 구현된 소프트웨어가 하드웨어 독립적인 아키텍처를 가질 수 있도록 한다. 이를 통하여 제안한 추적 유형의 모바일 파워 카트 소프트웨어가 여러 종류의 하드웨어에 적용 가능하고, 경우에 따라서 이동 알고리즘을 실행 시간에 동적으로 쉽게 변경할 수 있음을 보여준다.

본 논문의 구성은 다음과 같다. 2절에서는 본 논문에서 제안하는 시스템 구현과 관련된 기술을 소개한다. 3절에서는 3차원 깊이 카메라를 사용하는 사용자 추적 방식의 모바일 파워 카트 구현에 필요한 소프트웨어 아키텍처를 제안하고 이를 구현하기 위한 방안을 기술한다. 4절에서 구현된 시스템에 대한 성능 평가를 보여주고, 마지막으로 5절에서는 결론을 기술한다.

2. 관련 연구

2.1 ROS 소프트웨어 플랫폼

ROS 소프트웨어 플랫폼은 모바일 조작을 위해 설계된 메시지 기반 P2P 로보틱스 미들웨어로, 네이밍 서비스 기반의 메시지 송수신을 통한 분산 처리 시스템이라고 할 수 있다. 계산이 행하여지는 프로세스를 노드라고 하며, 이러한 노드의 집합을 패키지라고 한다. 이러한 패키지를 여러 개 묶어 내비게이션 스택이나 ROS 튜토리얼 패키지처럼 그룹화된 패키지를 메타패키지라고 한다. 프로그래밍은 C++ 프로그래밍 언어 또는 Python 스크립트 언어를 사용할 수 있다.

ROS 플랫폼은 로봇 개발에 필요한 3차원 깊이 카메라, 1-2차원 거리 측정기, 음성 인식, 모션 캡쳐, RFID 등 다양한 센서를 지원한다. 3차원 깊이 정보를 생성할 수 있는 마이크로소프트사의 Kinect나 아수스사의 Xtion Pro Live 센서는 사용자에 대한 3차원 정보로부터 스켈레톤(skeleton)을 생성하여 사용자의 자세 인식이 가능하다. ROS 플랫폼은 2차원 거리 측정기 혹은 LiDAR 센서(Light Detection And Ranging)도 지원하는데, 본 논문에서는 RPLiDAR 센서를 사용한다. LiDAR 센서는 레이저와 레이더를 합친 개념으로 주로 지도 생성, 장애물 탐지, 차간 거리 계산, 디지털 지형 모델 생성 등 다양한 분야에서 사용할 수 있다.

ROS 플랫폼에서 움직이는 하드웨어 플랫폼을 3차원 모델링하기 위해서는 URDF(Unified Robot Description Format) 형식[6]의 XML 파일을 사용한다. Fig. 1은 본 논문에서 정의하는 모바일 파워 카트의 3차원 모델을 graphviz 도구를 사용하여 시각화한 그림이다. 모바일 파워 카트의 몸통(base_link)과 네 개의 바퀴(wheel_1 ~ wheel_4), RPLiDAR 센서(base_laser_link)가 연결되는 구조를 보여준다. Fig. 2는 Fig. 1과 같이 정의된 URDF 파일을 ROS 3차원 시각화 도구인 rviz에서 렌더링하여 보여주는 그림으로 몸통, 바퀴, RPLiDAR 센서의 현재 좌표와 자세(orientation)를 나타내는 tf를 같이 보여주고 있다.

2.2 설계 패턴

설계 패턴은 설계 단계에 적용 가능한 패턴으로 GoF 패턴[7]으로도 불린다. 설계 단계에서 소프트웨

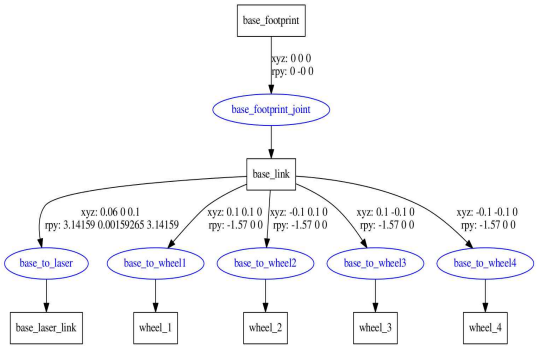


Fig. 1. Graphviz diagram of the URDF file of a mobile power cart.

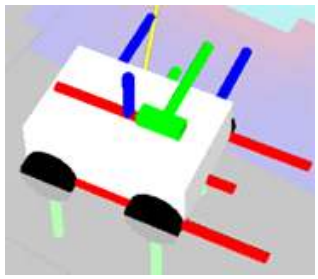


Fig. 2. 3D model of a mobile power cart with tf.

어 구조를 유연하게 함으로써 개발 및 유지 보수를 용이하게 한다. 본 절에서는 본 논문에서 사용되는 전략 패턴과 템플릿 메소드 패턴에 대하여 간략하게 기술한다.

상황에 따라서 서로 다른 알고리즘을 적용해야만 하는 경우 상속을 사용하면 자식 클래스에서 오버라이딩을 통하여 서로 다른 행위를 모델링할 수 있어 코드 재사용성 면에서 좋지 않다. 이러한 구조적 문제점을 해결하기 위해 클래스 내에 알고리즘을 캡슐화 하는 전략 패턴(strategy pattern)을 사용함으로써 기존 코드를 변경해야만 하는 문제를 해결할 수 있는 설계 패턴이다[7].

템플릿 메소드는 서브클래스에서 구상 행위를 제공하도록 오버라이딩 할 수 있는 추상 오퍼레이션을 사용하여 클래스의 오퍼레이션을 정의한다[7]. 추상 오퍼레이션의 구현에 따라서 약간씩 다른 행위를 하는 서브클래스 정의가 가능해지므로 동일한 알고리즘을 수행하지만 약간씩 내부적으로 다른 행위를 하는 경우에 적합한 설계 패턴이다.

3. 시스템 설계 및 구현

3.1 SCV 분석을 통한 아키텍처 설계

본 논문에서 구현하는 추적 유형의 모바일 파워 카트는 Fig. 3과 같은 시스템 기능을 가진다: 사용자 이동 추적(track user movement), 이동 행위 계산(Get movement behavior), 하드웨어 제어(control hardware), 시스템 환경 설정(configure the system). 대상 하드웨어(target hardware)는 본 논문의 대상인 모바일 파워 카트뿐만 아니라 기존의 상용 모바일 플랫폼인 터틀봇과 같은 다른 모바일 플랫폼에서도 적용할 수 있도록 유연한 소프트웨어 아키텍처 설계를 목표로 한다.

SCV 분석은 차이점을 가지는 유사한 시스템으로 구성되어 있는 소프트웨어 패밀리를 분석하고 구조화하는 데 사용되는 기법[5]으로 FAST 프로세스(Family-oriented Abstraction, Specification, and Translation process)에도 사용된다. SCV 분석은 개발하려는 제품 패밀리에 대하여 체계적인 방법으로 생각하고 식별할 수 있어 재사용 가능하고 쉽게 수정 가능한 설계가 가능하다.

모바일 파워 카트에 대한 SCV 분석 결과는 다음과 같다.

• 범위(scope): 물류 현장에서 사용자 추적을 통한 물품 이동을 위한 모바일 파워 카트 기능을 할 수 있는 터틀봇 등의 ROS 지원 모바일 하드웨어 플랫폼과 ROS 미지원 모바일 하드웨어 플랫폼

• 공통점(commonality): 3차원 깊이 카메라를 적용하여 사용자의 이동 방식 추적할 수 있는 기능

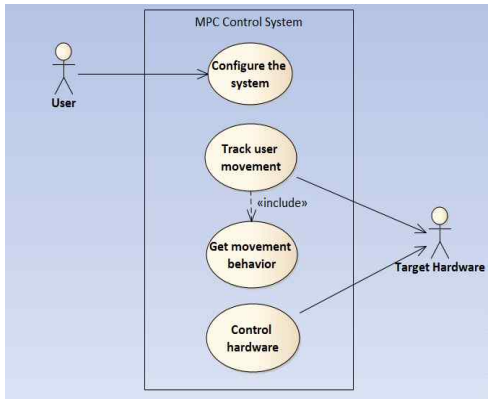


Fig. 3. A control system of a mobile power cart.

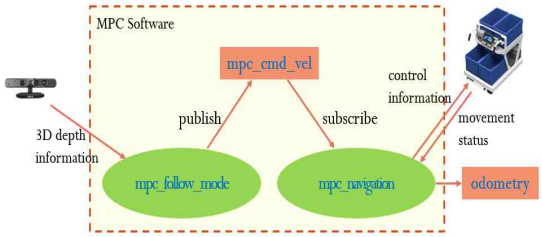


Fig. 4. Software architecture overview of the mobile power cart system.

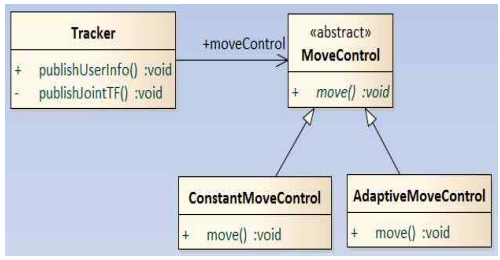


Fig. 5. A class diagram applying inheritance.

• 차이점(variability): 모바일 하드웨어 플랫폼 구동에 필요한 이동 제어/상태 정보는 하드웨어 플랫폼에 따라서 서로 다를 수 있으며, 이동 알고리즘 역시 변경 필요성 있음.

[하드웨어 구동 방식]

- 터틀봇과 같은 ROS 지원 장치: geometry_msgs/Twist 메시지에 따라서 직접 이동 제어 가능
- ROS 미지원 장치: geometry_msgs/Twist 메시지에 따라서 독자적인 이동 제어 방식 필요

[이동 알고리즘]

- 정속 이동, 적응 주행 등

Fig. 4는 SCV 분석을 통하여 설계한 모바일 파워 카트 시스템의 소프트웨어 아키텍처에 대한 그림이다. 모바일 파워 카트를 구현하기 위해 필요한 패키지는 mpc_follow_mode와 mpc_navigation이며, 기존의 openni2_tracker 패키지[8]를 mpc_follow_mode 구현에 사용한다. 다음은 이들 패키지 간의 관계를 알 수 있도록 각 패키지 기능을 요약한 것이다.

• openni2_tracker: 3차원 깊이 카메라를 사용하여 사용자 스켈레톤 정보 추출

• mpc_follow_mode: openni2_tracker에서 추출한 사용자 스켈레톤 정보를 분석하여 사용자의 이동을 감지하고 하드웨어 이동 제어를 위한 토픽 생성

• mpc_navigation: mpc_follow_mode에서 생성한 이동 제어 토픽을 하드웨어 플랫폼에 전달하여 이동하게 하고, 이동 관련 상태 정보를 발행

3.2 mpc_follow_mode 패키지

사용자 스켈레톤 정보를 추출하는 기능 구현은 기존의 openni2_tracker 패키지를 활용함으로써 ROS 플랫폼의 주목적인 재사용성을 높인다. openni2_tracker는 OpenNI2와 NiTE2 스켈레톤 추적 기능에 대한 ROS 래퍼(wrapper) 프로그램으로, 현재 추적 중인 사용자 스켈레톤에 대한 좌표와 자세 정보를 가지고 있는 tf 프레임을 발행한다[8]. 이전 버전의 ROS 개발 환경인 rosbuild 패키지로 개발되어 있으며, 클래스 구조가 아니라 일반적인 C 프로그래밍 언어 형태로 Table 1과 같은 함수를 가지고 있어 다른 기능을 추가하기 어려운 구조를 가진다.

Fig. 5는 mpc_follow_mode 패키지에서 사용자 이동을 추적하여 카트의 이동 제어를 하는 초기 설계 클래스 다이어그램이다. 기존의 openni2_tracker 패키지에 있던 소스 코드를 객체 지향적으로 수정, 3차원 깊이 카메라를 사용하여 사용자와의 거리와 각도를 추적하는 Tracker 클래스와 이러한 정보에 따라서 모바일 하드웨어 플랫폼의 이동 정보를 생성하여 mpc_cmd_vel 토픽으로 알려주는 책임을 수행하는 MoveControl 클래스로 구성된다.

MoveControl 클래스를 상속하는 ConstantMoveControl 클래스와 AdaptiveMoveControl 클래스는 하드웨어 플랫폼의 이동 제어정보를 생성하기 위하여 각각 정속 주행으로 추적하는 알고리즘과 사용자

Table 1. A list of functions in openni2_tracker

Function name	Description
main()	Create a nite::UserTracker object and update user states with the skeleton information
updateUserState()	Print the status message of user tracking
publishJointTF()	Broadcast the TF information of a given joint

이동 속도에 따라서 적응적으로 이동하는 알고리즘을 구현한다. 이와 같이 상속을 적용한 설계는 실행 시 이동 제어 정보를 생성하기 위한 알고리즘을 선택적으로 변경하여 적용할 수 없어 유연하지 않다.

이러한 문제점을 해결하기 위하여 Fig. 6과 같이 전략 패턴을 적용하였으며, 이와 관련된 클래스는 MoveControl 클래스, IMoveBehavior 추상 클래스, ConstantMove 클래스, AdaptiveMove 클래스이다. ConstantMove 또는 AdaptiveMove 클래스에서 IMoveBehavior 추상 클래스의 move() 오퍼레이션을 구현하고, MoveControl 객체에서 필요할 때 이 두 클래스의 객체를 생성하여 사용하면 move() 오퍼레이션에 사용되는 알고리즘을 실행 시 동적으로 변경 가능하며, 새로운 이동 제어 알고리즘 추가 및 사용에 있어 보다 유연한 구조가 된다. 이는 SCV 분석에서 식별된 차이점에 해당하는 부분을 구현한 것이다. Table 2는 MoveControl 클래스의 코드 일부를 보여주는 것으로 줄 번호 3은 MoveControl 헤더 파일에 선언된 변수 정의를 보여주며, 줄 번호 4~6은 하드웨어 이동 제어 정보를 계산하기 위한 함수로 줄 번호 9~12의 move 함수에서 호출된다.

MoveControl 클래스를 상속받아 구현된 TurtleBotMoveControl 클래스와 MpcMoveControl 클래스는 다양한 모바일 하드웨어 플랫폼을 구현하는 방법을 보여준다. 앞에서와 마찬가지로 이 부분 역시 SCV 분석에서 식별된 차이점에 해당한다. TurtleBot

Table 2. Code fragments of the 'MoveControl' class

```
1  /* For simplicity, parameters are omitted. */
2
3  /* deu_impl::IMoveBehavior *moveBehavior; */
4  void MoveControl::calculateVelocity(...) {
5      moveBehavior->calculateVelocity(...);
6  }
7
8  /* virtual void publishVelocity() = 0; */
9  void MoveControl::move(...) {
10     calculateVelocity(...);
11     publishVelocity();
12 }
```

MoveControl 클래스는 MoveControl 클래스에서 생성한 하드웨어 이동 제어 정보를 유진 로봇의 터틀봇 제어에 적합하게 수정하여 발행하며, MpcMoveControl 클래스는 다음에 설명한 mpc_navigation 패키지에서 정의한 하드웨어 플랫폼에 맞게 제어 정보를 발행한다. 이 두 클래스에서 오버라이딩하여 구현하는 publishVelocity 함수는 Table 2의 줄 번호 11에서 볼 수 있듯이 MoveControl::move 함수에서 사용되는 함수이다. 따라서 MoveControl::move 함수는 템플릿 메소드임을 알 수 있다.

3.3 mpc_navigation 패키지

터틀봇과 같은 기존의 ROS 지원 모바일 하드웨어 플랫폼의 경우 2차원 또는 3차원 지도에서 사용할

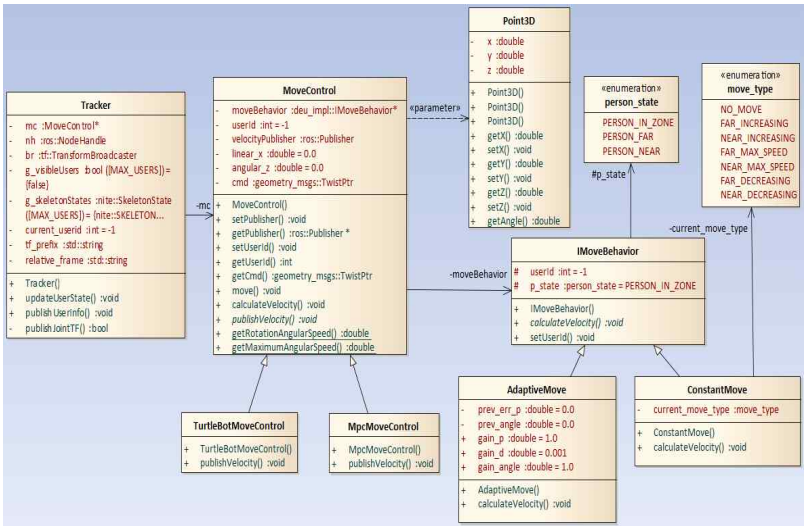


Fig. 6. A class diagram of the 'mpc_follow_mode' package applying design patterns.

수 있도록 하드웨어에 대한 3차원 모델링 정보와 하드웨어 제어 및 상태 정보 교환에 필요한 ROS 패키지가 존재한다. 터틀봇의 경우 turtlebot_bringup 패키지를 사용하면 하드웨어 제어가 가능하여 Fig. 6의 TurtleBotMoveControl 클래스에서 직접 터틀봇에게 제어 정보를 전달 가능하다. 그러나 ROS 미지원 하드웨어 플랫폼의 경우 3차원 모델링 정보와 하드웨어 제어 및 상태 정보 교환에 필요한 패키지 개발이 필요하다. mpc_navigation 패키지는 이와 같이 ROS 미지원 하드웨어 플랫폼에 적용하여 mpc_follow_mode 패키지와 같이 사용, 사용자 추적 유형의 모바일 파워 카트를 구현하기 위해 필요하다.

하드웨어에 대한 3차원 모델링은 URDF XML 파일을 사용하여 정의 가능하며, 2차원 또는 3차원 지도에 매핑하여 모바일 하드웨어 플랫폼의 이동을 계산하거나 시각화하는 데 중요하다. 2.1절 Fig. 1과 Fig. 2는 mpc_navigation 패키지에서 사용하는 3차원 모델링 결과를 보여준다.

mpc_navigation 패키지에서 구현하는 Fig. 7의 BaseController 클래스는 mpc_follow_mode 패키지에서 발행하는 mpc_cmd_vel 토픽 정보를 받아 대상 하드웨어의 이동을 제어(cmdVelCallBack 함수의 책임)하고, 하드웨어 구성 요소 각각의 현재 좌표와 자세 정보인 tf와 모바일 하드웨어 플랫폼의 시간에 따른 주행 거리 측정 정보(odometry)를 발행(publish Odometry 함수의 책임)한다.

Table 3은 BaseController 클래스의 소스 코드 중 일부로 cmdVelCallBack과 updateTransform 오퍼레이션은 각각 sendMoveInformation과 getEncoder Value 추상 오퍼레이션을 구현에 사용하여 템플릿

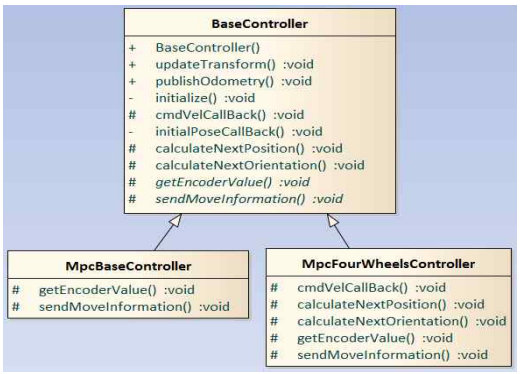


Fig. 7. A class diagram of the 'mpc_navigation' package.

Table 3. Code fragments of the 'BaseController' class

```
1  /* For simplicity, parameters are omitted. */
2
3  void BaseController::cmdVelCallBack(...) {
4      // calculate velocities of two wheels
5      // according to a given 'mpc_cmd_vel' topic
6      sendMoveInformation(...);
7  }
8
9  void BaseController::updateTransform() {
10     getEncoderValue();
11     // broadcast TF
12 }
```

메소드임을 알 수 있다. 이와 같이 템플릿 메소드 패턴을 적용하여 다양한 모바일 하드웨어 플랫폼에 대응할 수 있도록 설계한다. 그리고 BaseController 클래스를 상속받는 MpcFourWheelsController 클래스는 네 바퀴를 가지는 하드웨어 플랫폼을 구현할 때 필요한 클래스 구현을 가정한다.

4. 성능 평가

본 절에서는 모바일 파워 카트를 위해 구현한 ROS 패키지에 대하여 소프트웨어 복잡도 관점에서 분석하고, 구현된 모바일 파워 카트의 성능 시험에 대하여 기술한다. Table 4는 pmccabe 프로그램[9]을 사용하여 기존의 openni2_tracker와 mpc_follow_mode 패키지의 소프트웨어 복잡도를 비교 분석한 것이다. 주석을 제외한 실행 가능 문장 수와 소프트웨어 복잡도를 나타내는 McCabe 복잡도를 비교한 결과, mpc_follow_mode 패키지의 전체 문장 수와 McCabe 복잡도는 openni2_tracker 대비 각각 481%와 511%이나, 함수 당 평균 문장 수와 평균 McCabe 복잡도는 각각 51.4%와 54.8%를 보여 전체적으로 복잡도가 많이 감소하였다. Watson과 McCabe[10]에 의하면 McCabe 복잡도는 유지 보수 관점에서 최대 10을 넘지 않도록 제시하고 있다. 따라서 openni2_tracker의 평균 McCabe 복잡도가 9.3으로 10에 근접하고 있어 함수 수에 비해 복잡하게 코드가 작성되어 있음을 알 수 있다.

Table 5는 모바일 파워 카트 개발에 사용된 장치 목록을 보여준다. 모바일 파워 카트에 사용될 대상 하드웨어 플랫폼이 개발 중에 있어 터틀봇을 이용하여 패키지 개발을 진행 중이다. 소프트웨어는 Ubun-

Table 4. Comparison of complexity metrics

Metrics \ Package	openni2_tracker	mpc_follow_mode
the number of classes	1	8
the number of functions	3	28
total number of statements	83	399 (481%)
average number of statements per functions	27.7	14.25 (51.4%)
total McCabe complexity	28	143 (511%)
average McCabe complexity per functions	9.3	5.1 (54.8%)

Table 5. Development environment of a mobile power cart

Device	Description
TurtleBot	Kobuki Base. Max. linear velocity 0.7m/sec. Max. angular velocity 180°/sec.
ASUS Xtion Pro Live	RGB & Depth image sensor. Distance of Use: 0.8~3.5m
RPLiDAR	360° 2D laser scanner. 5.5Hz(2000 samples/sec). Max. range: 6m.

tu 14.04 운영체제에 ROS Indigo 버전을 설치하여 관련 ROS 패키지를 개발하였다. 사용자에 대한 3차원 깊이 정보 입력은 ASUS Xtion Pro Live 카메라를 사용하였으며, 현 위치 추정(localization), 실내 지도 생성 등에는 RPLiDAR 센서를 사용하였다.

Fig. 8은 RPLiDAR 센서를 사용하여 모델링한 모바일 파워 카트에 대한 현 위치 추정 기법 적용 결과를 rviz 시각화 도구를 사용하여 보여주는 그림이다. 모바일 파워 카트 주변의 노란 점은 RPLiDAR 센서가 검출한 장애물(obstacle)이며, 그 주변의 두꺼운 띠는 모바일 파워 카트가 뚫고 지나갈 수 없는 부푼 장애물(inflated obstacle)을 의미한다. 빨간 색 화살표는 모바일 파워 카트에서 생성되는 주행 거리 측정 정보이다. 검은 점은 전역 지도에서 사무실의 벽을 나타내는 것으로 RPLiDAR 센서로부터 입력된 정보로부터 현 위치 추정이 적절하게 되고 있음을 알 수 있다.

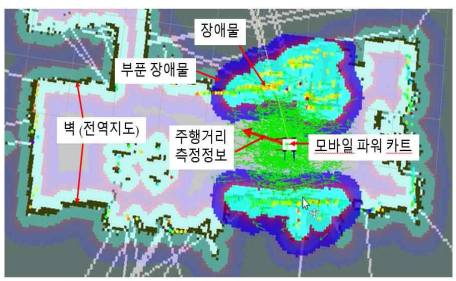


Fig. 8. Localization of a mobile power cart.

사용자 추적 정보를 생성하기 위한 mpc_follow_mode 패키지는 3차원 깊이 카메라를 사용하여 Fig. 9와 같이 사용자와 그 주변의 사람을 식별하여 스켈레톤 정보를 발행한다. 이 정보를 사용하여 사용자의 이동 여부를 식별한 다음 사용자의 이동을 따라 동일한 거리를 유지할 수 있도록 하드웨어 이동 제어를 위한 mpc_cmd_vel 토픽을 생성한다. 이 토픽은 선형 속도와 각속도로 구성되는데, 정속 이동 알고리즘(ConstantMove 클래스)과 적응 이동 알고리즘(AdaptiveMove 클래스)의 선형 속도는 계산 주기마다 각각 식(1), (2)와 같이 계산된다. 식(1)의 v_{step}은 한 주기에 증가하는 속도를 의미한다. 식(2)는 적응 이동 제어를 위해 모바일 파워 카트와의 거리를 오차 e로 하여 비례(proportional) 항 $K_P e$와 미분(derivative) 항 $K_D \Delta e$의 합으로 구성된 PD (Proportional

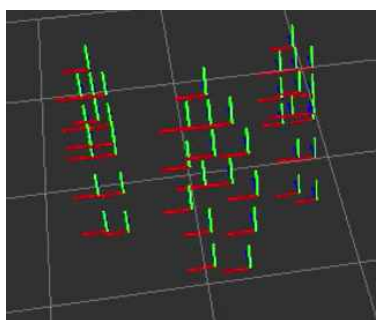
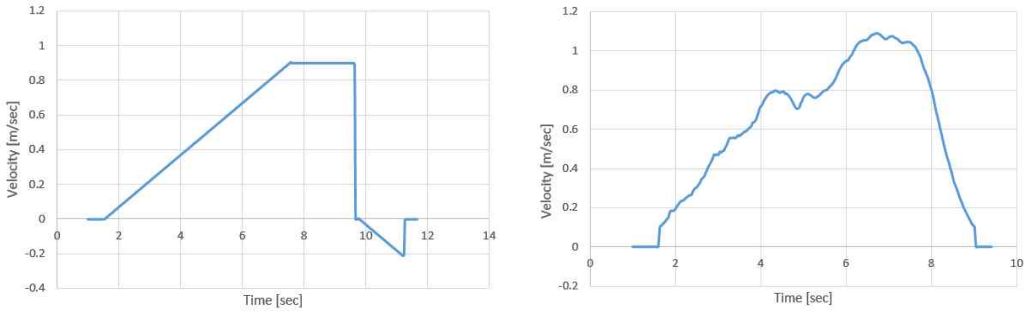


Fig. 9. Skeleton information generated by 'mpc_follow_mode'



(a) (b)

Fig. 10. Linear velocity of a mobile power cart. (a) ConstantMove and (b) AdaptiveMove.

Derivative) 제어 알고리즘을 보여준다. 이때 상수 K_P와 K_D는 각각 2.0과 0.001로 설정한다.

$$v_{const,x} = \min\{v_{const,x} + v_{step},\ 0.9\} \quad (1)$$

$$v_{adapt,x} = K_P e + K_D \Delta e \quad (2)$$

이에 따라서 모바일 파워 카트가 어떻게 이동하는지 알아보기 위하여 대상 하드웨어 플랫폼을 현재 사용 가능한 터틀봇으로 설정하고 사용자가 이동함에 따라서 이동 속도가 어떻게 변화하는지 측정해 보았다. 편의상 사용자가 직진으로 일정 시간 이동하다가 정지한다고 가정한다. Fig. 10은 정속 이동 알고리즘 ConstantMove 객체와 적응 이동 알고리즘 AdaptiveMove 객체를 사용했을 때의 이동 속도를 보여 준다. 성인의 평균 보행속도는 1.1m/sec[11]이며, 실험에 사용된 터틀봇의 최대 이동 속도는 0.7m/sec여서 정속 또는 이동 알고리즘에 의해 생성된 선형속도가 0.7m/sec을 초과하면 실제 터틀봇의 이동 속도는 0.7m/sec으로 제한된다.

Fig. 10(a)는 정속 이동 알고리즘 적용 시 실제 발행되는 선형 속도를 측정한 것이다. 정속 이동 알고리즘의 계산 주기는 30Hz, v_{step}은 0.005(m/sec)이므로 최대 속도인 0.9 m/sec에 도달하는데 6초 소요되며, 최대 속도에 도달하면 계속 유지된다. 1.56초에서 이동이 시작되어 7.56초에 최대 속도에 도달하며 9.6초에서 정지했는데, 이때 사용자와 모바일 파워 카트의 거리가 너무 가까워 적정 거리인 1.6~1.7m를 유지하기 위하여 11.25초에 이를 때까지 뒤로 이동했음을 보여준다. Fig. 10(b)는 적응 이동 알고리즘 적용 시 실제 발행되는 선형 속도를 측정한 것이다. 실험 결과로부터 모바일 파워 카트의 거리에 따라 지속적으로 이동 속도가 변화하는 궤환(feedback) 제어가 이루어지고 있음을 알 수 있다.

5. 결 론

물류 현장에서 작업 효율을 높이기 위한 무인반송차는 오래전부터 활용되어 오고 있었으나, 이동 방식과 고가의 센서 등의 제약으로 인해 다양한 물류 현장에 적용하기는 어려웠다. 본 논문에서는 다양한 물류 현장에서 쉽게 적용할 수 있도록 기존의 수동 카트와 프로그램에 의한 주행 기술을 결합한 추적 유형의 모바일 파워 카트 구현 기술을 제안하였다. SCV 분석을 통하여 사용자 이동을 추적하는 mpc_follow_mode 패키지와 새로운 하드웨어 플랫폼을 구동할 수 있는 mpc_navigation 패키지를 가지는 소프트웨어 아키텍처를 설계하고 구현하였다.

아키텍처 설계에 전략 패턴을 사용하여 다양한 이동 알고리즘을 개발, 적용할 수 있게 하고, 템플릿 메소드 패턴을 적용하여 여러 하드웨어 플랫폼 구동에서 사용할 수 있도록 하였다. 설계 패턴을 적용한 아키텍처가 적정한지 알 수 있도록 소프트웨어 복잡도를 개발에 사용한 openni2_tracker 패키지와 비교 분석한 결과 실행 가능 문장 수가 증가함에도 평균 McCabe 복잡도가 10 이하로 적정함을 보였다. 또한 성능 시험을 통하여 정속 이동 알고리즘과 적응 이동 알고리즘이 설계한 바와 같이 사용자 이동을 잘 추적함을 보였다. 따라서 본 논문에서 제안한 아키텍처를 활용하여 모바일 파워 카트를 다양한 하드웨어 플랫폼에 적용, 물류 현장에 활용할 수 있을 것으로 기대된다.

REFERENCE

[1] M.Y. Ali, S.G.M. Hossain, H. Jamil, and M.Z. Haq, "Development of Automated Guided Vehicles for Industrial Logistics Applications in Developing Countries Using Appropriate Technology," *Journal of Mechanical & Mechatronics Engineering*, Vol. 10, No. 2, pp. 13-17, 2010.

[2] S. Byun and M. Kim, "A Visual Based Guideline Interpretation Technique for AGV Navigation," *Journal of Korea Multimedia Society*, Vol. 15, No. 11, pp. 1319-1329, 2012.

[3] A. Elkady and T. Sobh, "Robotics Middleware: A Comprehensive Literature Survey and Attribute-Based Bibliography," *Journal of Robotics*, Article ID 959013, pp.1-15, 2012.

[4] ROS/Introduction - ROS Wiki, http://wiki.ros.org/ROS/Introduction (accessed Jan., 18, 2016).

[5] J. Coplien, D. Hoffman, and D. Weiss, "Commonality and Variability in Software Engineering," *Journal of IEEE Software Engineering*, Vol. 15, No. 6, pp. 37-45, 1998.

[6] urdf/XML/mode - Ros Wiki, http://wiki.ros.org/urdf/XML/model (accessed Jan. 18, 2016).

[7] E. Gamma, R. Helm, R.R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison Wesley, Boston, MA, 1994.

[8] Futureneer/openni2-tracker - C++ GitHub, https://github.com/futureneer/openni2-tracker (accessed Jan. 18, 2016).

[9] PMCCABE Overview, https://people.debian.org/~bame/pmccabe/overview.html (accessed Jan. 18, 2016).

[10] A.H. Watson and T.J. McCabe, *Structured Testing: A Testing Methodoloy Using the Cyclomatic Complexity Metric*, NIST Special Publication, Gaithersburg, MD, 1996.

[11] C.Y. Chung, M.S. Park, I.H. Choi, T.J. Cho W.J. Yoo and J.Y. Kim, "Three Dimensional Gait Analysis in Normal Korean: A Preliminary Report," *Journal of the Korean Orthopaedic Association*, Vol. 40, No. 1, pp. 83-88, 2005.

이 종 민

1992년 경북대학교 컴퓨터공학과 공학사
1994년 한국과학기술원 전산학과 공학석사
2000년 한국과학기술원 전자전산학과 공학박사

1997년~2002년 삼성전자 무선사업부 책임연구원
2012년 Visiting scholar, The University of Alabama
2002년 3월~현재 동의대학교 컴퓨터소프트웨어공학과 교수
관심분야 : 모바일컴퓨팅, 병렬컴퓨팅, 사물인터넷, 소프트웨어공학

권 오 준

1986년 경북대학교 전자공학과 공학사
1992년 충남대학교 전산학과 공학석사
1998년 포항공과대학교 전산학과 공학박사

1986년 1월~2000년 2월 한국전자통신연구원 선임연구원
2000년 3월~현재 동의대학교 컴퓨터소프트웨어공학과 교수
관심분야 : 정보보안, 컴퓨터네트워크, 패턴인식

김 성 우

1991년 한국과학기술원 전기및전자공학과 공학사
1993년 한국과학기술원 전기및전자공학과 공학석사
1999년 한국과학기술원 전기및전자공학과 공학박사

1999년~2002년 한국전자통신연구원 선임연구원
2008년 Research Associate, California State University at Fresno
2002년 3월~현재 동의대학교 컴퓨터소프트웨어공학과 교수
관심분야 : 임베디드소프트웨어, 사물인터넷, 지능형로봇