https://doi.org/10.7735/ksmte.2017.26.1.121

J. Korean Soc. Manuf. Technol. Eng.
ISSN 2508-5107(Online) / ISSN 2508-5093(Print)

Special Issue : SNUT Capstone Design

무선랜을 이용한 조립 작업 로봇의 협력 제어 시스템 구축

박상영[a], 이귀형[a*]

Foundation of Cooperative Control System of Assembly-Working Robots Using Wireless LAN

Sang-Young Park[a], Gui-Hyung Lee[a*]

[a] *Department of Mechanical System Design Engineering, Seoul National University of Science and Technology, 232, Gongneung-ro, Nowon-gu, Seoul 01811, Korea*

ARTICLE INFO

Article history:
Received 3 January 2017
Revised 4 February 2017
Accepted 15 February 2017

Keywords:
Cooperative control
Assemlby work
Mobile robot
Wireless LAN
ROS

ABSTRACT

In this study, we investigated a cooperative control system of assembly robots using wireless LAN. We developed two different types of robots to assemble three blocks on a workbench. Robot1 can assemble blocks on a workbench and Robot2 can carry blocks to Robot1. We constructed an ROS-based communication system and shared data. Three blocks and one workbench were recognized by camera-image processing By developing the UI using Windows programming language Visual C#, we evaluated the status of the robots and blocks and controlled the robots. The control system was developed by constructing all elements necessary for cooperative control, such as robot design and fabrication, motor control, ROS-based communication, and image processing. Thus, we completed fundamental tasks required for assembly.

1. 서 론

현대 산업이 날로 발전함에 따라 로봇에게 요구되는 기능과 역할이 훨씬 복잡해지고 있다. 이러한 복잡한 기능과 역할을 수행하는 데에 단일 로봇은 구조적 한계가 있어 다개체 로봇에 대한 연구가 진행되고 있다. 2010년 기준, 세계 130만 대의 산업용 로봇과 720만 대의 서비스 로봇이 활동 중이며 이들은 점차 네트워크로 연결되고 팀을 이루어 협업하는 시스템이 높은 효율성을 가질 것으로 보고 있다. 4차 산업 혁명을 눈 앞에 둔 시점에서 물류, 제조업에서 큰 효과를 기대하고 있다. 또 다품종 소량생산의 시장 변화에 따라 고성능 단일 로봇보다는 이종 다수 로봇의 협력이 유연하게 대응 가능할 것으로 보인다[1].

국내에서 진행 중인 다개체 로봇 연구는 주로 편대 이동 제어에 대한 연구이거나 대부분이 동일한 형태를 가지는 동종 로봇에 대한 연구이다. 또 로봇의 제작과 실험 없이 시뮬레이션에 그치는 실정이다[2,3]. 해외에서는 보다 실질적인 연구가 진행되어 왔으나 대부분이 동종 로봇이거나 youbot, Pioneer P3-DX와 같이 이미 제작되어 있는 로봇 플랫폼을 사용하거나 마커를 이용해 로봇의 위치를 인식하였다[4,5].

본 논문에서는 좀더 구체적으로 블록을 조립하는 작업을 선정하여 연구하였다. 서로 다른 형태의 로봇 두 대를 직접 설계 및 제작하고, 두 대의 로봇이 서로 협력하여 조립 작업을 수행하는 시스템을 구현하였다. 모바일 형태의 다개체 로봇을 연결하기 위해 무선랜을 이용하고 로봇 미들웨어이자 TCP/IP 기반의 통신 미들웨어

* Corresponding author. Tel.: +82-2-970-6325
Fax: +82-2-974-8270
E-mail address: ghlee@seoultech.ac.kr (Gui-Hyung Lee).

인 ROS로 전체 통신 시스템을 구축하였다. 블록과 작업대를 인식하기 위해 로봇에 부착된 카메라에서 영상 데이터를 수신하여 처리하였는데, 관심 물체를 검출하여 얻은 중심 좌표와 기울기 값을 토대로 조립 작업을 수행하였다. 쓰러진 블록을 바로 세우는 협력 작업을 구현하였고 로봇 한 대로 주변의 블록을 작업대에 조립하는 실험을 진행하였다. 조립 작업을 수행하는 로봇의 협력 제어를 위해 필요한 요소들로 시스템을 구축하였고, 실행 결과 기본 동작들이 잘 수행됨을 확인하였다.

2. 로봇의 제작 및 시스템 구성

2.1 조립 작업의 요구 사양

작업 목표는 작업 공간 내부 임의의 위치에 놓인 블록 3개를 작업대에 3층으로 조립하는 것이다. Fig. 1은 작업 공간을 도시한 것이다. 블록은 빨간색, 초록색, 파란색으로 3가지 색상을 갖고 작업대는 노란색이다. 검정색은 이동 불가 영역을 나타낸다. Fig. 2에 블록과 작업대를 도시하였는데, 돌기가 홈에 들어가는 형태로 조립 작업이 이루어진다. 작업 공간은 2 m×2 m이며 블록과 작업대의 크기는 각각 40 mm×40 mm×48 mm, 160 mm×40 mm×11 mm이다.

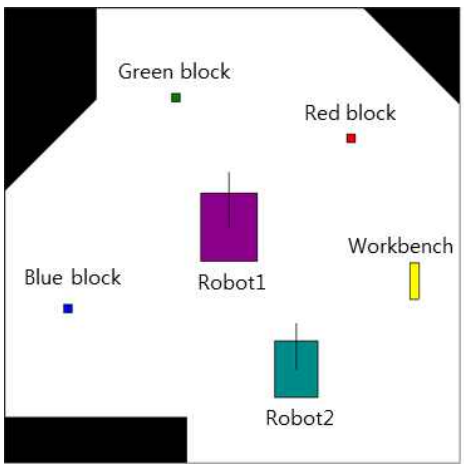


Fig. 1 Overall layout of workspace

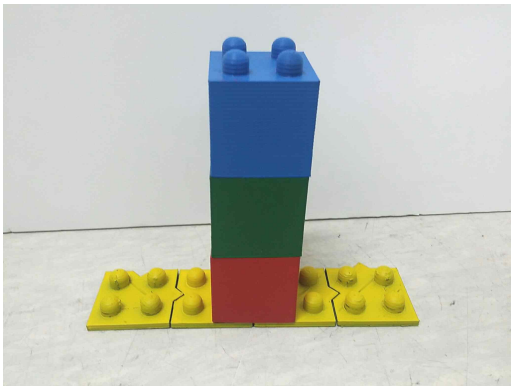


Fig. 2 Assembling of blocks on workbench

2.2 로봇의 설계와 제작

작업 로봇은 로봇1과 로봇2로 총 2대이며 주요 구성 요소에는 바퀴, 로봇팔, Kinect 센서가 있다. 두 로봇의 바퀴, 로봇팔, Kinect 센서는 동일하고 로봇팔은 서로 다른 형태를 가진다. Fig. 3은 제작된 로봇1의 사진을 도시하였다. 다소 좁은 작업 공간을 고려하여 크기가 250 mm×360 mm×302 mm인 소형으로, 질량은 4.2 kg이고 로봇팔 슬라이더의 스트로크는 125 mm이 되도록 제작하였다.

Fig. 4는 제작된 로봇2의 사진을 나타낸다. 다소 좁은 작업 공간을 고려하여 180 mm×240 mm×260 mm의 크기로 무게는 1.9 kg이다. 로봇2의 로봇팔은 단순 회전운동 구조로 설계하였다. 쓰러진 블록을 바로 세우기 위해 로봇1과 협력 작업이 필요하다. 로봇2의 센서는 정면을 향해야 하고, 쓰러진 블록을 90°에 가깝게 회전해야 하며, 로봇1의 그리퍼의 작업 공간에 들어갈 수 있도록 로봇팔을 설계, 제작하였다.

Khepera III Gripper의 메커니즘을 참고하여 로봇1의 로봇팔을

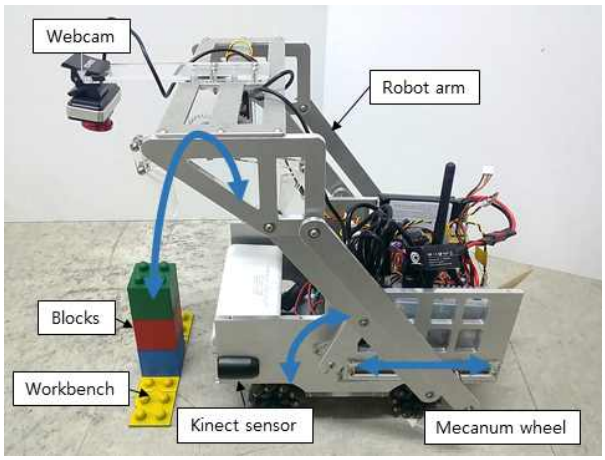


Fig. 3 Figure of Robot1

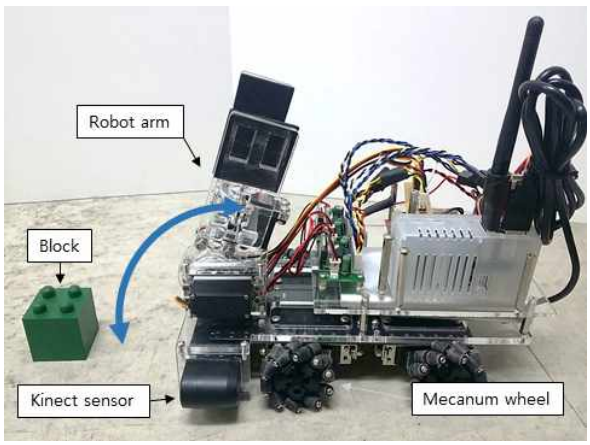


Fig. 4 Figure of Robot2

설계하였다[6]. 로봇팔의 구동 모터를 1개로 제한하였고 Fig. 5와 같이 원하는 평면 궤적을 따라 움직이도록 slider-crank(4절 링크) 구조로 설계하였다. 이때 end-effector는 블록을 여러 층으로 조립하기 위해 항상 수평을 유지하도록 하였다. 원하는 평면 궤적은 바닥에 위치한 블록에서부터 작업대의 3층에 위치한 블록까지이다. 블록과 작업대 인식을 위해 RGB 센서가, 로봇의 위치 보정을 위해 depth 센서가 필요하기 때문에 RGBD 센서인 Kinect 센서를 선정하였다. 다양한 Kinect 센서 중 ROS 구동을 고려해 Linux와 호환이 가능하고, 짧은 거리도 측정이 가능한 Primesense Carmine 1.09로 선정하였다.

2.3 통신 시스템

Fig. 6에 ROS의 통신 시스템을 도시하였다. ROS(robot operating system)는 일종의 통신 미들웨어로서, 소프트웨어를 모듈 단위인 노드(node) 단위로 나누어 구성하며 노드들은 TCP/IP 기반으로 토픽(topic)을 발행/구독(publish/subscribe)하는 방식으로 데이터를 송수신한다[7]. 모바일 형태의 다개체 로봇을 연결하는 통신 방법으로 무선랜이 유리하므로 TCP/IP 기반인 ROS로 전체 통신 시스템을 구축하였다. 본 논문에서는 별도의 PC가 server로서 master 역할을 하고 로봇들은 PC 없이 ROS 기반 임베디드 시스템을 구성하여 PC의 master와 통신하는 하이브리드 구조로 구성하였다[8]. 로봇에서는 영상 데이터를 수집하여 PC로 전달하고 PC에서는 영상처리, 가시화 등 전체 시스템을 제어하기 위해 필요한 고성능 연산과 알고리즘이 수행된다.

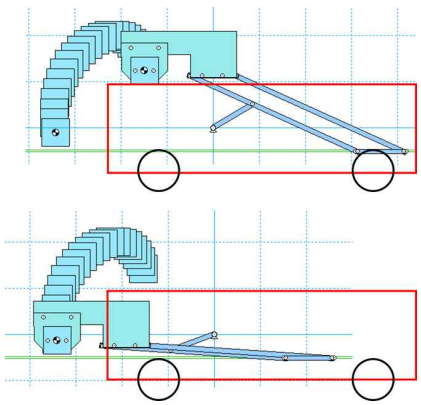


Fig. 5 Modeling of robot arm of Robot1

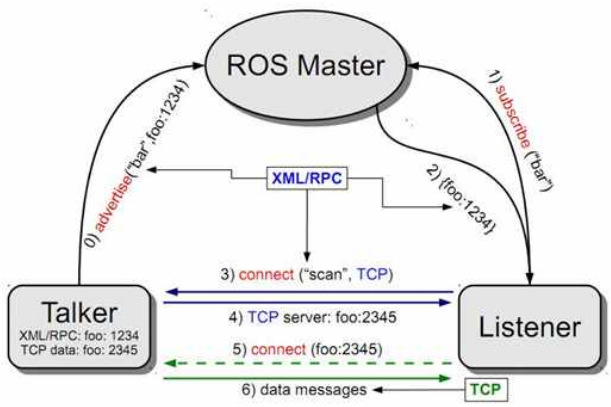


Fig. 6 Communication system of ROS[7]

2.4 제어 시스템

Fig. 7에 로봇의 하드웨어 구조를 도시하였다. 하위 제어기인 ATmega128이 모터 제어를 맡고, 상위 제어기인 Odroid-U3는 센서 데이터를 수집하며 PC와 무선랜으로 통신한다. Odroid-U3와 ATmega128 간에는 시리얼 통신으로 데이터를 주고받는다. 바퀴와 그리퍼 부분은 로봇1과 로봇2에 공통으로 구성되고, 로봇팔의 경우 로봇1은 DC geared encoder motor로, 로봇2는 Servo motor로 구동한다.

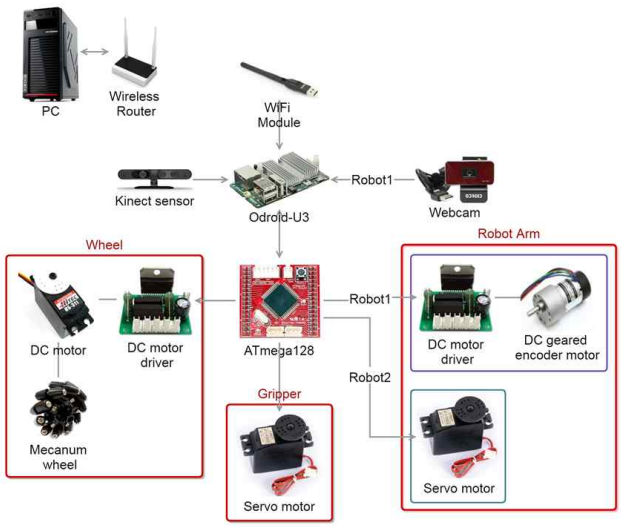


Fig. 7 Hardware structure diagram

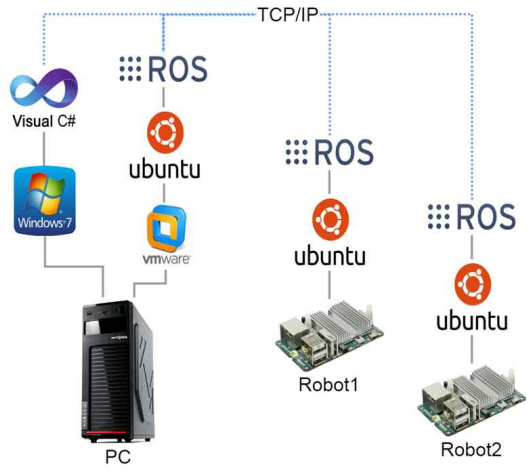


Fig. 8 Software structure diagram

Fig. 8에 소프트웨어 구조를 도시하였다. PC에서는 Windows7에서 Visual C#으로 작성한 UI가 구동되고 가상머신(VMware)에서 Ubuntu 위에 ROS(master)가 구동된다. 로봇1과 로봇2에서는 Ubuntu 위에 ROS가 구동되며 ROS와 C#은 모두 TCP/IP로 통신이 이루어진다. 본 논문에서 Ubuntu는 14.04LTS, ROS는 Indigo 버전으로 개발하였다. 로봇1, 2에서는 로봇에 부착된 카메라에서 수신한 영상 데이터와 로봇의 이동량을 PC의 ROS master로 전달한다. PC의 ROS master에서는 로봇들로부터 수신한 데이터를 가공하여 블록 위치, 로봇 위치, 행동 완료 상태를 파악하고 UI 프로그램에 전달한다. UI에서는 이러한 정보들을 보여주면서 사용자가 각 로봇을 제어할 수 있도록 하였다. 영상 데이터의 통신 부하를 줄이기 위해 영상 해상도를 320×240 pixel로 줄였으며, 영상 데이터의 수신 주기를 고려하여 ROS master에서 영상 처리 주기를 0.2초로 하였다.

3. 조립 작업의 구현

3.1 구동부 제어

바퀴는 Mecanum wheel을 사용하였다. 로봇의 속도(v_x, v_y, w_z)를 등속으로 제한하고 모터 구동 시간을 조절하여 엔코더 없이 오픈 루프로 위치 제어를 하였다. 실험적으로 일정 시간 동안 로봇의 이동량을 측정하여 로봇의 실제 속도를 계산하였다. 블록을 잘 잡는 위치로 정밀하게 움직여야 하기 때문에 제한 속도는 로봇이 안정적으로 움직이는 최저 속도로 하였다. 로봇1의 로봇팔은 부드러운 움직임을 만들기 위해 사다리꼴 속도 프로파일을 적용하고 각도 값에 대해 PID 제어를 하였다. 시리얼통신 명령어 처리 주기는 10 ms, PID 제어 주기는 20 ms로 하였다.

3.2 영상 처리

3가지의 색(빨강, 초록, 파랑)을 가진 블록과 노란색의 작업대를 검출하기 위해 RGB 데이터를 HSV(hue, saturation, value) 데이터로 변환하여 추출하였다. 이후 관심 물체 내외부의 노이즈를 제거하기 위해 모폴로지 연산을 수행하였다. 블록의 각도를 알기 위해 수평 모서리를 추출한 후 hough line 변환을 이용해 직선을 검출하여 직선의 기울기를 계산하였다. 수평 모서리 추출에는 수평 모서리의 y축 밝기 차가 큰 점을 이용하여 y방향 sobel 필터를 적용하였다. 모서리 검출에 대표적인 canny edge 알고리즘과 비교하였을 때 sobel 필터가 더 효과적으로 수평 모서리를 추출하였다. Fig. 9와 Fig. 10에 수평 모서리 검출을 수행한 결과를 도시하였다.

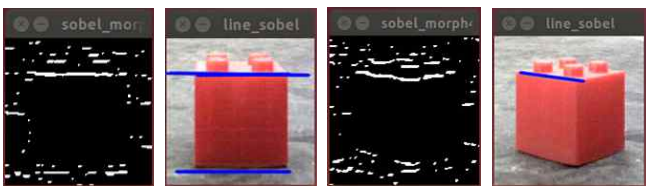


Fig. 9 Horizontal edge detection using y-direction sobel filter

Fig. 10 Horizontal edge detection using canny edge algorithm

정면을 바라보고 있는 Kinect 센서의 RGB 데이터로 블록의 위치와 각도를 측정하려 했으나 화면에 블록과 작업대의 전체가 잡히지 않는 문제가 있어 로봇1의 로봇팔 상단에 광각렌즈를 부착한 웹캠을 추가하였다. 관심 물체(블록과 작업대)의 contour를 감싸는 직사각형을 구해 관심 물체의 중심 좌표와 각도를 계산하였다. Fig. 11과 같이 작업대에 블록이 조립되어 작업대의 중심이 가려진 경우 나뉜 2개의 contour의 중앙을 잇는 직선을 추가하여 1개의 contour가 되도록 한 뒤 이 1개의 contour를 감싸는 직사각형을 구해 작업대의 중심 좌표와 각도를 계산하였다.

로봇에서부터 블록까지의 상대 위치 측정을 위해 RGB 데이터에서 관심 블록을 먼저 검출하고 depth 데이터로 블록까지의 거리를 측정하였다. Fig. 12에 Carmine 1.09의 RGB 센서와 depth

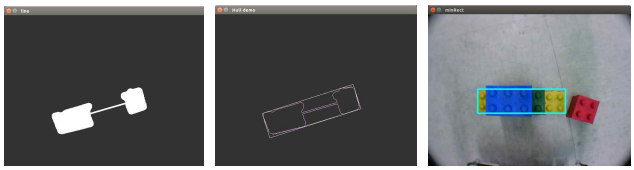
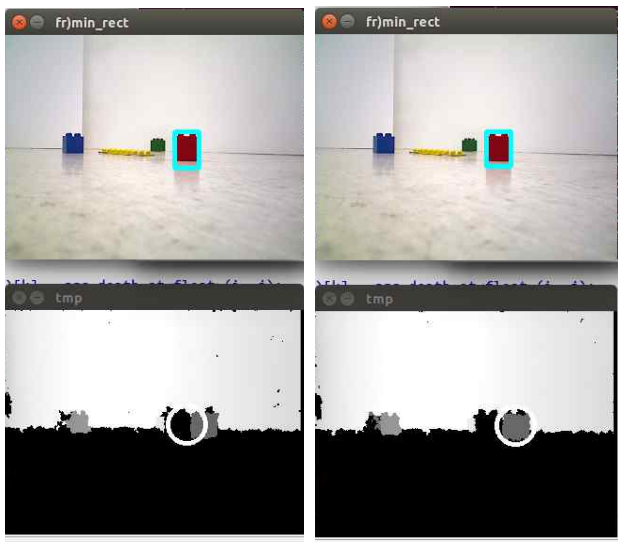


Fig. 11 Rectangular surrounding occluded workbench

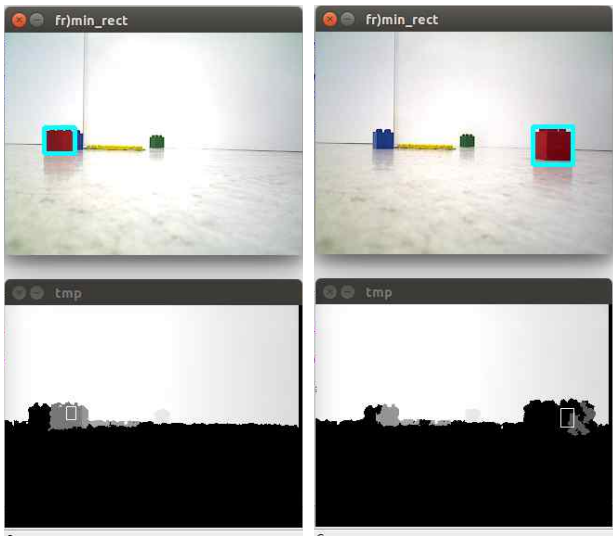


(a) (b)

Fig. 12 Correction of block center of depth image in RGB image

센서의 시점 차를 보정한 과정을 도시하였다. Fig. 12(a)와 같이 depth 이미지에서의 빨간색 블록 위치가 RGB 이미지에서의 빨간색 블록 위치와 달라 RGB 데이터에서 얻은 중심좌표에서 보정이 필요하였다. 여기서는 빨간색 블록을 감싸는 직사각형의 가로의 절반만큼을 더해 보정하였다. Fig. 13에 Kinect 센서의 depth 데이터에서 블록까지의 거리 값을 읽을 때 블록의 영역을 조정한 것을 도시하였다. Kinect 센서의 depth 데이터는 노이즈가 심하고 값이 없는 부분이 존재한다. 이 부분은 NaN(not a number)으로 처리되는데 중심좌표에 해당하는 픽셀 하나에 대해서만 거리 값을 읽어올 경우 NaN 값이 될 수 있는 문제가 있다. 따라서 Fig. 13과 같이 관심 물체를 벗어나지 않도록 기존 사각형의 가로/3×세로/2만큼을 새로운 영역으로 잡고 이 영역 내부에 NaN이 아닌 값에 대해서만 평균을 구해 물체까지의 거리 값을 얻었다.

Fig. 14에 전체 영상 처리가 이루어지는 img_proc 노드와 관련된 토픽들을 도시하였다. PC에서 로봇의 Kinect 센서와 웹캠의 영상 데이터를 수신하여 영상 처리가 이루어진다. PC에서 영상 처리를 한 후 모터 제어를 위한 명령어를 command 토픽으로 발신하고 로봇의 Odroid-U3가 이를 수신하여 ATmega128로 전달한다. ATmega128에서는 명령에 대한 응답을 보내고 Odroid-U3가 이 응답을 response 토픽으로 다시 PC로 전달한다. img_proc 노드에서는 이 응답을 반영하면서 영상 처리를 수행한다. 실제 실험을 진행하기 위해 Visual C#으로 작성한 UI 프로그램과 behavior, block_pose, flag_completion, robot_pose 토픽을 주고받는다. behavior는 로봇에게 할당된 현재 행동을, block _pose는 로봇에서 블록까지의 상대 위치와 각도를, robot_pose는 로봇의 위치를, flag_completion은 로봇의 동작 완료 상태를 나타낸다. 관심 물체의 중심 좌표와 기울기를 계산하고, 목표 위치에서 측정되는 관심 물체의 중심 좌표와 기울기와의 차이를 계산한다. x좌표, y좌표, 기울기의 차이를 비교하여 x축 이동, y축 이동, z축 회전, 로봇팔 구동 중 가장 적절한 동작으로 결정한다. 로봇이 목표 위치에 도달했으면 로봇팔을 구동하고 그렇지 않으면 중요도가 가장 높은 이동이 이루어지도록 하였다.



(a) (b)

Fig. 13 Setting block area of depth image

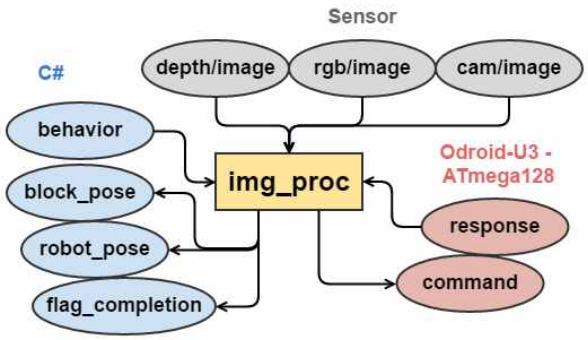


Fig. 14 Topics related to img_proc node

4. 실 험

4.1 실험 환경 구성

Fig. 15에 구성한 실험 환경을, Fig. 16에 블록과 로봇의 위치와

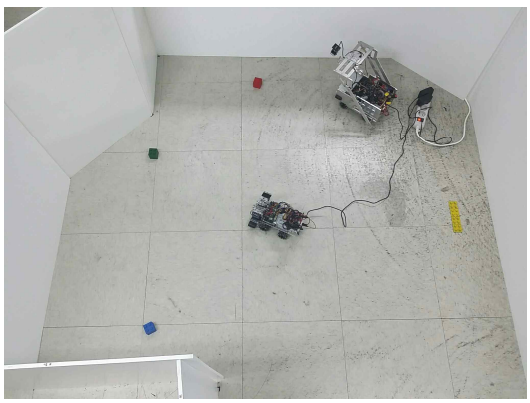


Fig. 15 Figure of experimental environment

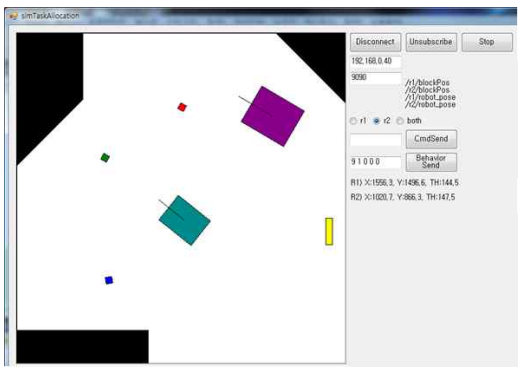


Fig. 16 Figure of UI program

상태를 나타내는 UI 프로그램을 도시하였다. 시뮬레이션을 실제 실험에 적용하기 위해 rosbridge_suite 패키지를 이용하여 ROS와 C#을 연동하였다. rosbridge는 JSON 기반으로 통신이 이루어진다[9]. Fig. 17에 ROS와 C# 간 주고받는 토픽을 도시하였다. block_pose는 로봇에서부터 블록까지의 상대 위치를, robot_pose는 로봇의 위치를, flag_completion은 로봇의 행동 완료 신호를, behavior는 C# UI에서 결정한 로봇의 현재 행동을 나타낸다.

4.2 기본 동작 실험

조립 작업을 위한 기본 동작들에 대한 실험을 진행하였다. 기본 동작에는 접근이동, 블록잡기, 블록놓기가 있다. 접근 이동 시 mecanum wheel을 오픈 루프로 위치 제어를 하기 때문에 오차가 발생한다. 이 오차를 보정하기 위해 영상처리로 얻은 관심 물체의 위치와 각도 값을 이용하였다. 광곽렌즈를 부착한 웹캠의 인식 영역은 대략 440 mm×310 mm이고 시야각이 65°이며 영상 해상도는 320×240 pixel이므로 분해능은 가로 1.31 mm/pixel, 세로 1.24 mm/pixel이다. 이를 통해 영상처리로 1.3 mm의 위치 정밀도를 기대할 수 있다. Carmine 1.09의 경우, 블록은 100 cm, 작업대는 85 cm까지 인식이 가능하였다. Figs. 18, 19, 20에 로봇의 위치 제어 시 발생하는 오차를 보정한 실험 과정을 도시하였다. Fig. 18(a)는 로봇1에서부터 100 cm 떨어진 지점을 도착 지점으로 잡고 영상처리로 위치 보정을 하도록 빨간 블록을 위치시킨 모습이다. 기존 로봇의 웹캠(USB-CAM 720P) 우측에 고성능 카메라 센서(Logitech C920)를 부착하여 2 mm 눈금을 눈으로 읽는 방법으로 로봇의 위치를 측정하였다. Figs. 18(b), (c)에 영상처리 없이 로봇이 10 cm 전진 이동한 실험 과정을 도시하였다. Fig. 18(b)는 시작 지점에서, Fig. 18(c)는 도착 지점에서 C920으로 얻은 영상이다. 눈금을 확인해보면 10 cm 전진 이동 시 실제로 약 9.4 cm 이동한 것을 확인할 수 있다.

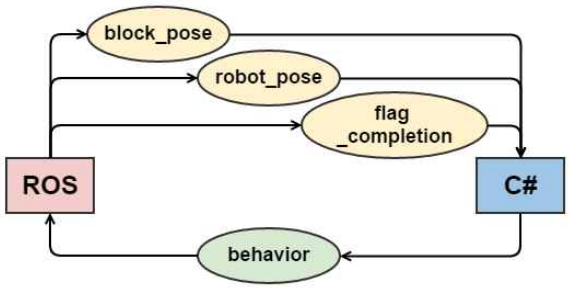
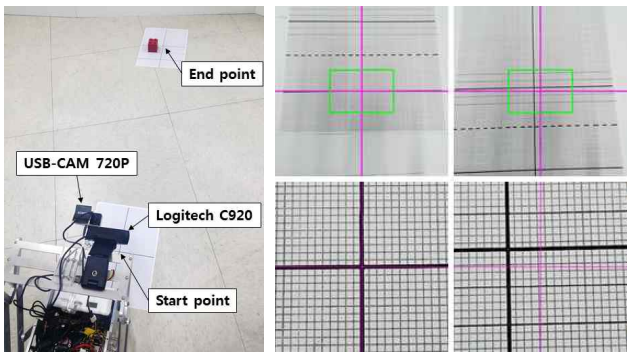


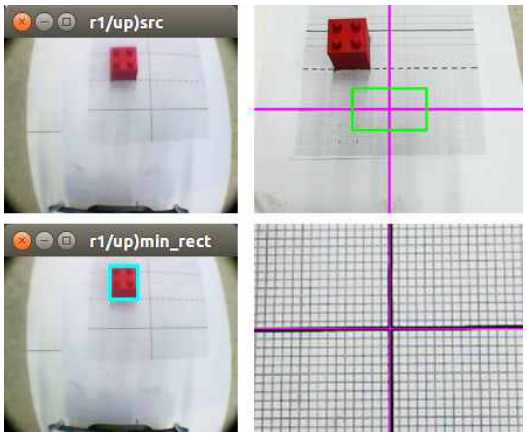
Fig. 17 Topics between ROS and C#



(a) (b) (c)

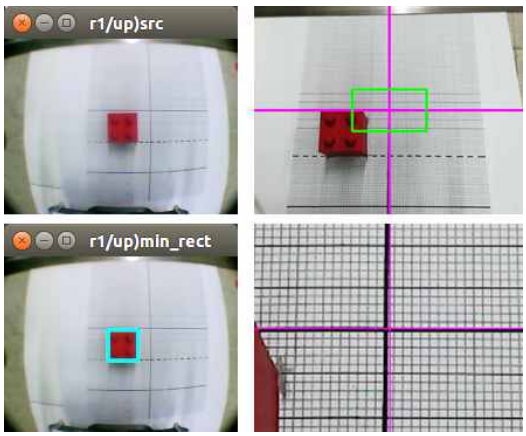
Fig. 18 Experiment to correct robot's position using image processing

Figs. 19, 20에 로봇이 10 cm 전진 이동할 때, 영상처리로 로봇의 위치 제어 시 발생하는 오차를 보정한 실험 과정을 도시하였다. Fig. 19는 시작 지점에서, Fig. 20은 도착 지점에서 얻은 영상을 나타내며, (a)는 USB-CAM 720P로 블록을 인식한 것을, (b)는 C920으로 눈금을 확인한 영상을 나타낸다. 로봇이 10 cm 떨어진 블록에 접근 이동 시 약 4.1초가 소요되었다. 위와 같은 방법으로 눈금을 이용하여 로봇의 이동과 관련된 오차 범위를 측정하였다. 로봇을 10 cm씩 x축(전진, 후진), y축(좌측, 우측) 병진 이동과



(a) (b)

Fig. 19 Images at start point



(a) (b)

Fig. 20 Images at end point

90° 회전(CCW, CW) 이동 시 실제 이동량을 측정하였다. Table 1에 각 방향에 대한 측정값들의 오차범위를 표준편차로 정리하였다. 또한 블록을 20 cm, 50 cm, 100 cm 떨어진 지점에 위치시킨 후 영상처리로 접근 이동하는 실험을 진행하였다. 20 cm 떨어진 지점에서는 블록을 45°로 위치시킨 추가 실험을 진행하였는데, 이 때 영상처리 결과의 오차범위는 0.2 mm 내외이었다. 실제 블록의 돌기지름이 9 mm, 홈 지름이 11 mm로 2 mm 오차한계를 넘지 않아야 하는데 0.2 mm 오차범위는 이를 충분히 만족한다. 1 pixel의 오차도 허용하지 않고 접근이동을 할 경우, 정밀도는 충분히 만족하나 소요시간은 약 2~3배 정도 증가하는데, 이는 정밀도와 소요 시간 사이에 trade-off가 발생한 것이다. 따라서, 소요 시간을 줄이기 위해 오차한계내로 오차범위를 맞추어 2~3 pixel의 허용오차를 두었다. Table 2에 이동에 소요된 시간을, Table 3에 블록잡기와 블록놓기에 소요된 시간을 정리하였다.

4.3 협업작업과 조립작업 실험

기본 동작들을 기반으로 협업작업과 조립작업에 대한 실험을 진행하였다. Fig. 21에 쓰러진 블록을 바로 세우는 협업작업 과정을 도시하였다. 블록은 로봇2로부터 15 cm, 로봇1로부터 20 cm 떨어진 지점에서 작업을 시작하였다. 로봇2가 먼저 쓰러진 블록을 집고 회전시킨다(1). 로봇1이 로봇2에 접근하고 로봇팔을 블록 근처로 이동시킨다(2). 로봇2의 로봇팔을 조금 더 들어올린다. 로봇1이 블록을 집는다. 로봇2가 블록을 놓는다(3). 로봇1과 로봇2의 로봇팔을 원위치로 놓는다(4). 로봇1 로봇팔의 작업 공간이 협소하여 로봇2의 로봇팔을 2번에 나눠 회전시키는 과정이 필요하였다. 두 로봇이 상대 로봇의 동작 완료 상태를 확인하기 위해 flag_completion 변수를 공유하였다. 로봇2의 그리퍼의 가용 범위는 32~85 mm이고, 로봇1의 그리퍼의 가용 범위는 ~95 mm이므로 블록의 가로 길이는 로봇2 그리퍼의 가용 범위를, 세로 길이는 로봇1 그리퍼의 가용 범위를 넘지 않아야 하며 높이는 40 mm가 적당하였다. 실험에서 소요된 시간은 기본 동작들에 소요되는 시간의 합으로 계산할 수 있으며, Table 4에 협업작업에 소요된 시간을 기본 동작들로 구분하여 정리하였다. 기본 동작들의 합으로는 76.9초가 소요되지만 실제로는 협업작업 간 두 로봇의 동작 완료를 대기하는 시간과 오차 등이 포함되어 83.2초가 소요되었다.

Fig. 22에 로봇1이 주변에 위치한 블록을 모두 조립하는 작업

Table 1 Standard deviation of each direction movement

Movement	x-axis translation	y-axis translation	z-axis rotation
std.	2.95 mm	4.81 mm	1.84°

Table 2 Time spent approaching to block (sec)

Block's position	Average	Std.
20 cm, 0°	20.8	1.30
20 cm, 45°	46.0	5.83
50 cm, 0°	30.4	3.05
100 cm, 0°	52.0	4.06

Table 3 Time spent gripping and putting block (sec)

Grip	Put
21.0	18.3

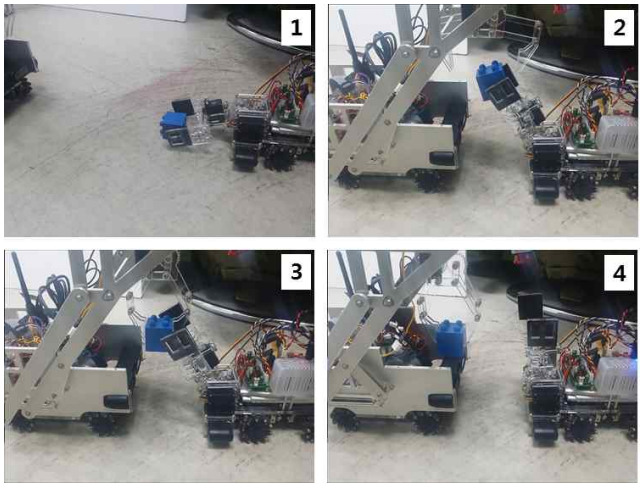


Fig. 21 Progression of cooperation task to erect block straight

Table 4 Time spent cooperative task (sec)

Case	Robot2's approach	Robot2's grip	Robot1's approach	Robot1's grip	Sum
Expectation	20.8	21.0	14.1	21.0	76.9
Real	22.3	20.5	15.8	24.6	83.2

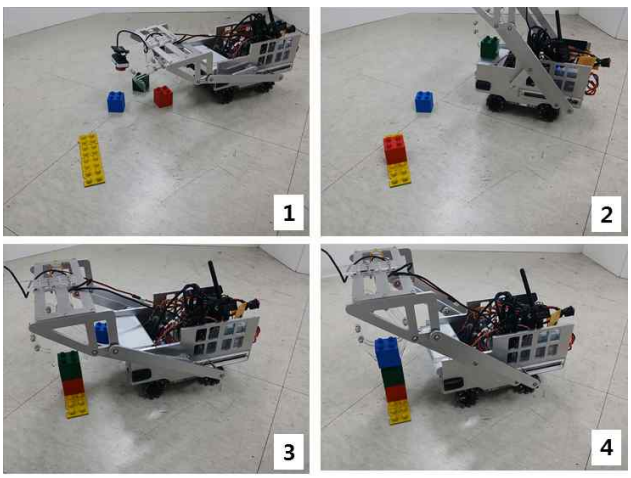


Fig. 22 Progression of assembly task by Robot1

Table 5 Time spent assembly task (sec)

Case	Expectation		Real	
	Approach	Grip/put	Approach	Grip/put
Red block (1-1)	30	21	42	21
Workbench (1-2)	30	19	40	19
Green block (2-1)	24	21	30	21
Workbench (2-2)	32	18	48	18
Blue block (3-1)	15	21	29	21
Workbench (3-2)	10	17	19	17
Sum	258		325	

과정을 도시하였다. 블록은 빨간색, 초록색, 파란색 순서대로 조립하며 쓰러진 블록이 없는 것으로 제한하였으며, 순서대로 로봇1에서부터 10 cm, 15 cm, 20 cm 떨어져 있고, 작업대는 30 cm 떨어져 있다. 로봇1의 웹캠에 관심 물체가 잡히지 않은 경우에는 Kinect 센서의 RGB 영상에서 관심 물체를 찾도록 하였다. 협업작업과 마찬가지로 조립작업에서도 소요된 시간을 기본 동작들에 소요되는 시간의 합으로 계산하여 Table 5에 정리하였다. 기본 동작들의 합으로는 253초가 소요되지만 블록과 로봇 간의 각도가 정확하게 고려되어 있지 않았으며, 실제로는 315초가 소요되었다.

영상 데이터에서 색을 구분하여 블록과 작업대만 검출하므로 로봇 간 충돌이 발생할 수 있다는 한계가 있다. 또 블록이 로봇에서 너무 멀리 떨어져 있거나 다른 물체에 가려진 경우 영상 데이터에서 검출되지 않아 작업을 완료할 수 없다는 한계가 있다. 두 로봇의 충돌 없이, 영상에서 블록과 작업대가 검출되지 않는 경우에도 작업을 수행할 수 있도록 로봇 주변 환경으로부터 로봇의 위치를 추정할 수 있는 기술이 필요하다. 또 능력이 다른 두 로봇에게 작업을 적절히 분담하여 작업시간을 줄이는 등 협업 알고리즘을 제안하고 성능을 평가하는 연구가 필요하다.

5. 결 론

본 논문에서는 무선랜을 이용한 조립 작업 로봇의 협력 제어 시스템 구축에 대해 연구하였다. 3개의 블록을 작업대에 조립하는 조립 작업을 수행하기 위해 서로 다른 형태의 로봇 두 대를 제작하였다. 로봇1은 조립 작업을 맡고 로봇2는 운반 작업을 맡아 보조 역할을 수행하도록 설계, 제작하였다.

ROS 기반으로 통신 시스템을 구축하여 다개체 로봇 간 데이터를 공유하였다. Linux와 다른 플랫폼인 윈도우프로그래밍 Visual C#에서 개발한 UI에서 로봇의 위치와 상태를 확인하여 제어하였다.

블록과 작업대는 카메라 영상처리로 인식하였고 y방향 sobel 필터를 적용해 블록의 수평 모서리로부터 블록의 각도를 측정할 수 있었다. 로봇에서부터 블록까지의 상대 위치 정보로 로봇의 위치를 보정하였다.

로봇 설계 및 제작, 구동부 제어, ROS 기반 통신 구축, 영상처리 등 협력 제어에 필요한 모든 요소를 구축하여 제어 시스템을 완성하였다. 실험에서는 접근이동과 블록잡기, 블록놓기 등 로봇들의 기본 동작들에 대한 거리오차와 작업시간을 측정하고, 블록의 조립 정밀도 2 mm를 만족하면서 로봇의 접근이동에 소요되는 시간을 단축시키도록 영상처리의 오차범위를 2~3 pixel로 조정하였다.

이를 바탕으로 서로 다른 형태의 로봇1, 2가 서로 협력하여 쓰러진 블록을 바로 세우는 협업작업을 수행할 수 있었으며, 이후 로봇1이 그 블록들을 작업대로 운반 조립하는 과정에 있어서 접근이동과 블록잡기, 블록놓기 등 필요한 기본 동작들을 차례로 수행함으로써 전체 작업도 수행할 수 있었다.

향후에는 블록이 멀리 떨어져 있거나 다른 물체에 가려져 있더라도 로봇이 블록까지 이동하기 위한 로봇의 위치 추정 기술에 대한 연구가 진행되어야 할 것이다. 또, 블록과 로봇의 다양한 초기 위치에 따라 해당 경우의 작업시간을 최소화하는 등 로봇1, 2가 최적으로 작업을 분담하는 협업 알고리즘을 고안 및 평가하고 실제 시스템에 적용하는 연구가 진행되어야 할 것이다.

후 기

이 연구는 서울과학기술대학교 교내학술연구비 지원으로 수행되었습니다.

References

[1] Choi, J. S., Lim, B. H., Lee, J. H., Yoon, Y. H., 2010, Cooperative technology and application of multi-agent robots, The Korean Institute of Electrical Engineers, 59:11, 16-26.

[2] Lee, S. M., Kim, H. K., Myung, H., 2013, Cooperative Particle Swarm Optimization-based Model Predictive Control for Multi-Robot Formation, Institute of Control, Robotics and Systems, 19:5, 429-434.

[3] Wi, S. G., Kim, Y. K., Choi, J. W., Lee, S. K., 2013, Hybrid Path Planning of Multi-Robots for Path Deviation Prevention, Institute of Control, Robotics and Systems, 19:5, 416-422.

[4] Ijspeert, A. J., Martinoli, A., Billard, A., Gambardella, L. M., 2001, Collaboration through the exploitation of local interactions in autonomous collective robotics: the stick pulling experiment,

Autonomous Robots, 11:2, 149-171.

[5] Knepper, R. A., Layton, T., Romanishin, J., Rus, D., 2013, IkeaBot: An Autonomous Multi-Robot Coordinated Furniture Assembly System, 2013 IEEE International Conference on Robotics and Automation (ICRA 2013), 855-862.

[6] K-Team, 2016, Kh4 gripper, <http://ftp.k-team.com/KheperaIV/gripper/Kh4_Gripper_UserManual.pdf>.

[7] Park, H. S., 2016, Robot software platform and modularization, The Journal of The Korean Institute of Communication Sciences-Information and communication, 33:8, 28-35.

[8] Hu, C., Hu, C., He, D., Gu, Q., 2015, A New ROS-based Hybrid Architecture for Heterogeneous Multi-Robot Systems, Altair Engineering Inc.

[9] ROS wiki, 2016, Rosbridge, <http://wiki.ros.org/rosbridge_suite>.