제19회 임베디드SW경진대회 개발완료보고서

[지능형 휴머노이드]

1. 개발 개요

1.1 요약 설명

팀 명	아이스
목 표	영상처리를 이용하여 안정적인 라인트레이싱과 시민과 텍스트, 재난구역을 빠르게 인식할 수 있는 알고리즘 구현
소스코드	https://github.com/dana4056/2021ESWContest_robot_2007.git
시연동영상	https://youtu.be/16l5idp9xDI

1.2 개발 목적 및 목표

1.2.1 라인 트레이싱

이번 대회에서 가장 큰 비중을 차지하는 부분인 보행을 적용하여 라인 트레이싱을 해야 한다. 이 부분에서 휴머노이드의 2족 보행은 4족 보행 로봇에 비하여 안정도가 확실히 떨어진다고 느낄 수 있었다. 영상처리를 통해서 라인을 따라 안정적으로 보행할 수 있도록 하고 방향 전환이 필요할 시에 방향을 맞추는 것에 중점을 두고 로봇을 제어한다.

1.2.2 텍스트 인식

미션이 진행되는 과정에서 텍스트 인식이 원활하지 않으면 대회를 진행하는 데 있어서 어려움을 겪는다. 따라서, 로봇의 앞쪽으로 빛이 많이 들어오거나 빛이 적게 들어와 어두운 환경에서도 빠르고 정확한 텍스트 인식이 되어야 한다. 빠르고 정확한 텍스트 인식을 할 수 있는 알고리즘을 구성하고 구현하는 것이 목적이다.

1.2.3 색 인식

정확한 색 인식이 되어야 미션(색을 인식하여 확진 구역과 안전 구역 구분, 방 이름과 일치하는 색의 시민 인식)을 문제없이 진행할 수 있다. 빛은 정도에 따라 밝기가 달라져 색 인식에 많은 영향을 미치기 때문에, 텍스트를 인식할 때와 같이 로봇 앞쪽으로 빛이 많이 들어와서 밝거나 빛이 적게 들어와 어두운 환경에서도 빠르고 정확하게 색 인식을 할 수 있어야 한다. 빠르고 정확한 색 인식을 할 수 있는 알고리즘을 모색하고 구현하는 것이 목적이다.

1.2.4 안정적인 파지

시민 미션을 수행하면서 시민으로 인식되는 물체를 파지할 때, 로봇이 자신보다 낮은 곳(바닥)에 있는 시민을 파지하기 위해 앉아야 한다. 그리고 파지한 상태에서 일어나고 걸어야 한다. 이 부분에 있어서 앉았다 일어나는 경우와 보행하는 경우에 시민을 놓치지 않고 안정적으로 파지할 수 있도록 정확한 관절 제어를 하는 것이 목표이다.

2. 개발 방향 및 전략

2.1 기술적 요구 사항

2.1.1 개발 환경 구축

2.1.1.1 VNC를 이용한 환경 구축

VNC란 Virtual Network Computing의 약자로 컴퓨터 또는 보드 사이에 서버와 클라이언트의 구조를 가지는 것인데, 원격지인 클라이언트에서 서버로 접속하여 컴퓨터 또는 보드를 제어하는 것이다. VNC를 설치하고 난 후에는 네트워크에 연결된 컴퓨터에 원격으로 접속해서 컴퓨터의 화면을 보면서 제어할 수 있다. 이 원격 접속은 일대일과 일대 다수 모두에서 가능하다. 그뿐만 아니라 리눅스, 윈도우 환경 혹은 모바일 기기, 임베디드 기기에서 모두 사용 가능하다.

따라서, VNC를 이용하면 로봇의 카메라와 제어 보드, 두뇌보드가 돌아가는 모습을 원격으로도 보고 제어할 수 있다. 또한 일대일이 아닌 일대 다수 접속이 가능하여 한 사람뿐만 아니라 모든 팀원이 접속하여 보고 바로 원격제어할 수 있는 장점이 있다.

본 팀은 Windows PC와 무선 공유기를 이용하여 VNC 연결을 하였다. 설정 방법은 어렵지 않았다. 네트워크 설정 변경에 들어가서 Wi-Fi속성 – 홈 네트워킹 연결에 라즈베이파이와 연결된 네트워크를 선택하고 나면 손쉽게 VNC 연결을 할 수 있다.

2.1.1.2 영상처리를 위한 패키지

2.1.1.2.1 OpenCV

OpenCV란, Open Source Computer Vision의 약자로 다양한 영상 처리에 사용할 수 있는 오픈소스 라이브러리이다. 쉽게 말하여, 컴퓨터가 사람의 눈처럼 인식할 수 있게 데이터를 처리해 주는 역할을 한다. 현재 대회에서 사용 중인 언어인 Python의 인터페이스를 지원하며 다양한 OS를 지원한다. OpenCV는 알고리즘상으로 계산 효율성과 실시간 응용 프로그램에 중점을 두고 설계됐기 때문에 OpenCV 상에서 제공하는 API를 사용하여 실시간으로 프로세싱이 가능하게 했다.

영상처리를 위하여 전처리하는 과정, 영상 데이터를 정제하는 과정, 특징점을 검출해 내는 과정, 또 다음은 영상 데이터를 출력하는 과정 모두 OpenCV 라이브러리를 사용하여 해결했다.

2.1.1.2.2 Numpy

Numpy란, Numerical Python의 약자로, 파이썬의 고성능 과학 계산용 패키지로서 Matrix와 Vector 같은 Array 연산을 할 때 사용하며 표준 라이브러리처럼 사용하고 있다. Numpy는 반복문 없이 데이터 배열에 대한 처리를 지원하기 때문에 일반 List에 비해 빠른 연산 속도를 가지며, 메모리를 효율적으로 사용한다.

본 대회의 특성상 영상처리가 중점인데 이 과정에서 비교적 무거운 3차원의 영상 데이터를 다루기 때문에 Numpy를 사용하지 않는다면 프로그램 수행이 상당히 느려질 것이다. 하

지만 Numpy를 사용함으로써 무거운 알고리즘이라 할 수 있는 객체 인식, 글자 인식도 무난히 수행할 수 있었으며, 다양한 영상 데이터를 처리하는 데 많은 도움이 되었다.

2.1.2 UART 통신

범용 비동기화 송수신기라고 불리는 UART는 디바이스 사이에서 데이터를 주고받을 때, 병렬 데이터의 형태를 직렬 방식으로 전환하여 데이터를 전송하는 하드웨어 장치이다. UART 통신은 동기식 통신과 다르게 데이터 송수신의 타이밍을 위하여 Clock 라인을 사용하지 않는다. 따라서 데이터 자체만 보내고, 데이터 해석을 위하여 데이터 프레임에 규칙을 정한다. 보통 EIA RS-232, RS-422, RS-485와 같은 통신 표준을 사용한다. 본 대회에서 사용하는 MF-RAPI4 로봇에서는 두뇌보드인 라즈베리파이와 제어 보드인 MR-C3024FX 사이의 데이터 통신이 필요하다. UART에는 디바이스 간에 통신을 하기 위해 송신 라인과 수신 라인을 따로 두어 동시에 송수신이 가능하다. 이러한 방식을 full-duplex(전이중) 방식이라고 한다.

이러한 UART 통신의 이론들을 제어 보드에서는 로보 베이직을 통해, 두뇌 보드에서는 파이썬을 통해 구현할 수 있다. 따라서 두뇌 보드에서 영상처리를 통해 로봇의 행동을 결정하고자 하면 serial 통신을 통해 제어 보드에게 명령을 한다(TX). 그러면 제어 보드에서는 그 명령을 받고(RX) 그에 해당하는 모션을 취한다. 또 반대로 제어 보드에서 일정한 모션을 완료했다고 두뇌 보드에게 송신(TX)을 하면 두뇌 보드는 그 메시지를 수신(RX)하고 미션 흐름에 따라 다른 영상처리를 하게 된다.

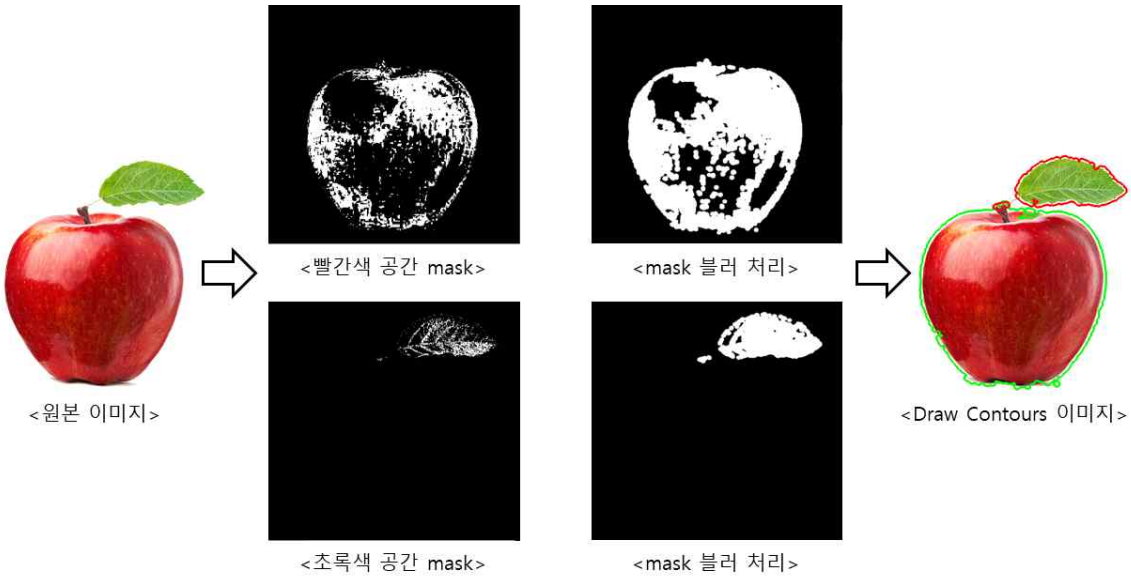
2.1.3 영상처리 이론

2.1.3.1 ROI

ROI(Region of Interest)는 영상에서 사용자가 관심 있는 영역을 뜻한다. 영상처리에서 가장 기초가 되는 기술이자 중요한 기술이다. 영상 안의 지정한 위치만 처리하거나 특정 오브젝트를 쉽게 찾기 위해 사용한다. 이 기술을 사용함으로써 불필요한 영역의 연산을 최소화할 수 있어 이미지 처리 속도가 빨라진다.

openCV의 도형 그리기 함수 중 하나인 fillPoly(img, points, color)를 사용하여 다각형의 꼭짓점을 따라 다각형 안을 원본 영상으로 채워 필요한 영역만 나타낼 수 있도록 한다. 아래 이미지는 도로와 풍경이 함께 있는 사진에서 도로 부분만 관심 영역으로 지정하고 추출한 모습이다.

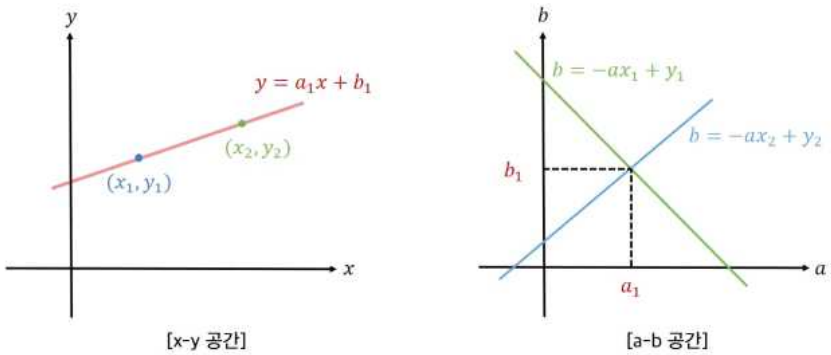
2.1.3.2 경계 그룹 찾기

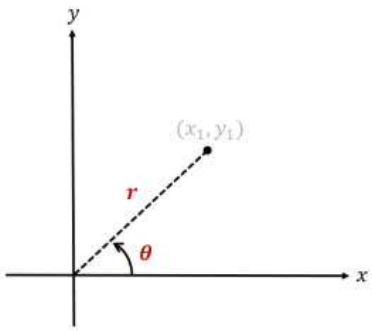


Contour은 객체의 외곽선을 검출하는 기술이다. 외곽선은 비슷한 색 공간(HSV 또는 RGB)을 가진 부분의 가장자리 좌표를 모두 연결하여 얻을 수 있다. 이 기술은 영상에 있는 물체의 모양을 분석하거나 객체 인식 등에 활용할 수 있다.

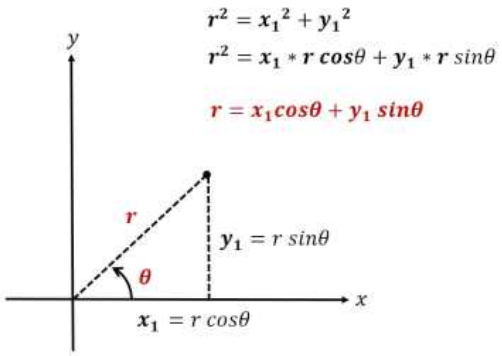
먼저, 인식하고자 하는 객체의 색 공간을 찾아 색 공간이 비슷한 부분의 픽셀을 제외하고 검은색으로 바꾸는 이진화 처리를 하여 mask 이미지를 만든다. 외곽선을 쉽게 추출하기 위해 mask 이미지를 블러 처리하여 조금 더 명확하고 부드러운 mask로 만들어준다. 그 후 openCV의 findContours 함수에 매개변수 'RETR_EXTERNAL'와 'CHAIN_APPROX_SIMPLE'로 객체의 외부 쪽만 검출하고 가장자리 좌표들을 단순화하여 끝점만 남긴 윤곽선을 그린다. 위의 사진은 사과의 빨간색 공간과 나뭇잎의 초록색 공간을 찾아 각각의 mask를 추출해 빨간색 공간은 초록색 외곽선을, 초록색 공간은 빨간색 외곽선을 그린 모습이다.

2.1.3.3 허프 변환

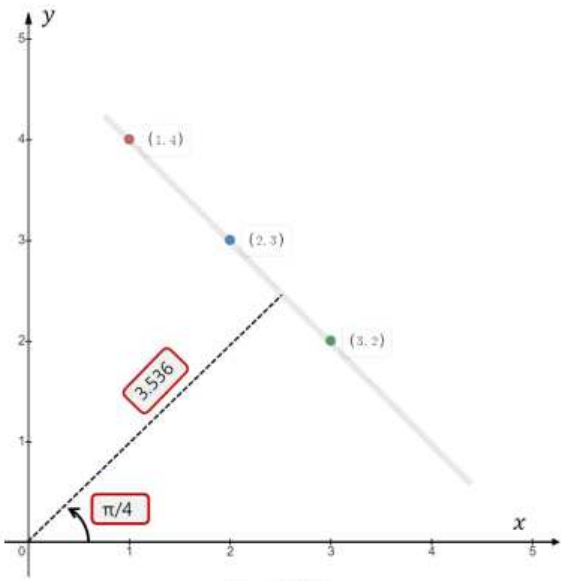




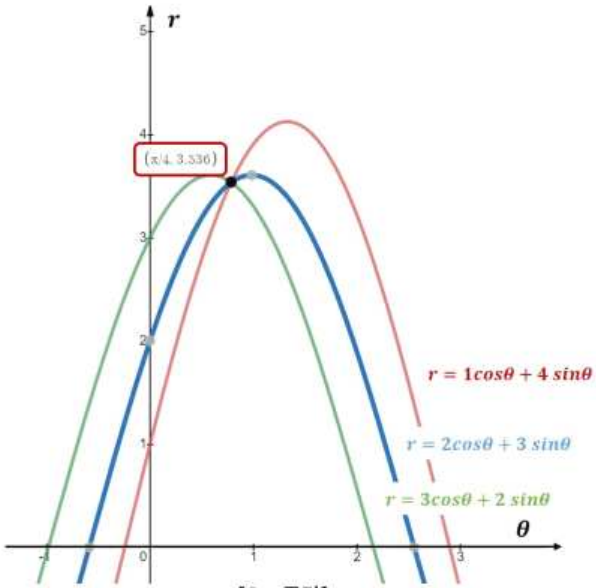
[x-y 공간]



[x-y 공간]



[x-y 공간]

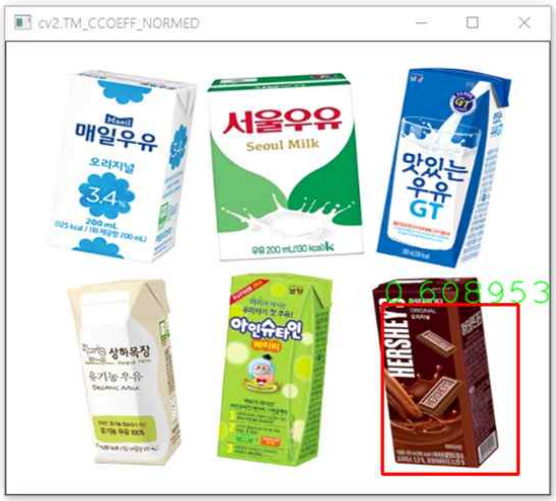


[θ-r 공간]

허프 변환은 이미지에서 직선을 검출하기 위한 알고리즘이다. 허프 변환은 직교 좌표계에서 극 좌표계로 바꿔주는 역할을 하는데, 직교 좌표계의 기울기 값의 범위가 무한으로 발산하기 때문에 데이터로 처리하기 어렵기 때문이다. 직교 좌표계에서 같은 직선상에 위치하는 점이 많을수록 이미지에 직선이 존재할 확률이 높아진다. 직교 좌표계에 두 점이 한 직선 위에 있으면 극 좌표계에는 교점이 존재하게 된다. 이러한 원리를 이용해서 이미지의 한 픽셀을 한 점으로 인식하여 직선을 찾을 수 있다.

2.1.3.4 템플릿 매칭

<template 이미지>



<reference 이미지>

템플릿 매칭은 미리 찾고자 하는 객체가 그려진 영상(template)을 준비해 두고 입력받은 영상(reference)과 비교해서 입력 영상에 객체가 있는지 판단하는 기술이다. reference 영상에서 객체를 찾아야 하므로 template 영상은 reference 영상보다 작아야 한다. template 영상과 reference 영상의 객체 위치는 달라도 매칭이 되지만, 회전된 객체는 찾기 힘들다는 단점이 있다. 하지만 휴머노이드 미션에서는 글자가 회전되어 있지 않으므로 템플릿 매칭을 활용해도 문제가 없을 것으로 판단하였다.

위의 사진은 초콜릿 우유의 객체를 찾기 위한 template 이미지를 미리 준비해 놓은 후 reference 이미지에서 같은 객체를 찾은 모습이다.

2.1.4 로봇의 동작 구동원리

2.1.4.1 기구학과 역기구학

기구학이란, 기계에서 어떠한 기준점을 시작으로 끝점까지의 순차적 위치를 결정하는 것이다. 로봇의 개념에서 몸체의 위치를 결정하고 손이나 발끝의 위치를 순차적으로 결정하는 것이다. 보행과 모션에 필요한 여러 가지 관절들은 독립적으로 동작할 수 있겠지만, 발끝의 위치는 모든 관절들의 움직임에 영향을 받아 종속된다. 이렇듯 여러 관절의 동작에 의해 발끝의 COM(Center Of Mass)의 좌표를 구하는 것에 기구학 개념이 이용된다. 역기구학이란, 기구학과 반대되는 개념이라고 할 수 있다. 발끝 혹은 손의 위치를 먼저 결정한 후 순차적으로 중심 위치까지 결정해나가는 것이다. 기구학과 역기구학의 개념을 파악하고 로봇의 가장 안정적인 무게중심이라 할 수 있는 ZMP(Zero Moment Point)를 찾는 것이 안정적인 보행을 위한 필수적인 사항이다.

2.1.4.2 모션 캡처

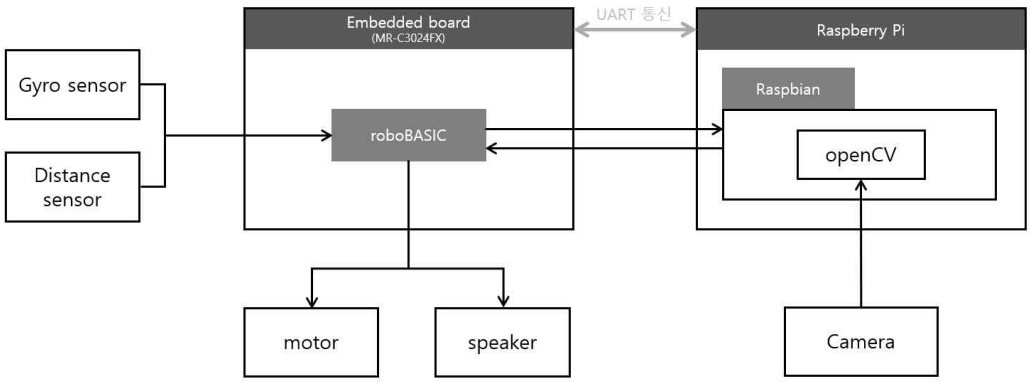
모션 캡처는 주로 인간의 동작을 캡처하여 값을 측정하고 측정된 데이터를 휴머노이드 같은 인간형 로봇이나 컴퓨터 애니메이션에 적용하는 것이다. 본 팀이 사용하는 모션 캡처는 로봇에서 원하는 관절 그룹 혹은 전체 관절 그룹의 모터를 off 상태에서 관절을 제어하여 필요한 동작을 만든 후, 해당 관절 그룹 혹은 전체 관절 그룹의 모터를 on 하여 원하는 로봇의 자세를 추출한다.

2.1.4.3 최적의 영점 찾기

로봇마다 각기 다른 영점이 존재한다. 따라서 같은 코드를 작성하더라도 로봇의 영점에 따라 다른 동작을 수행하기도 하며, 안정적인 보행 코드일지라도 불안정한 보행을 할 수 있다. 따라서 휴머노이드의 최적 영점을 찾아서 안정적인 보행과 미션을 수행할 수 있도록 하는 것이 중요하다.

2.2 개발 방법

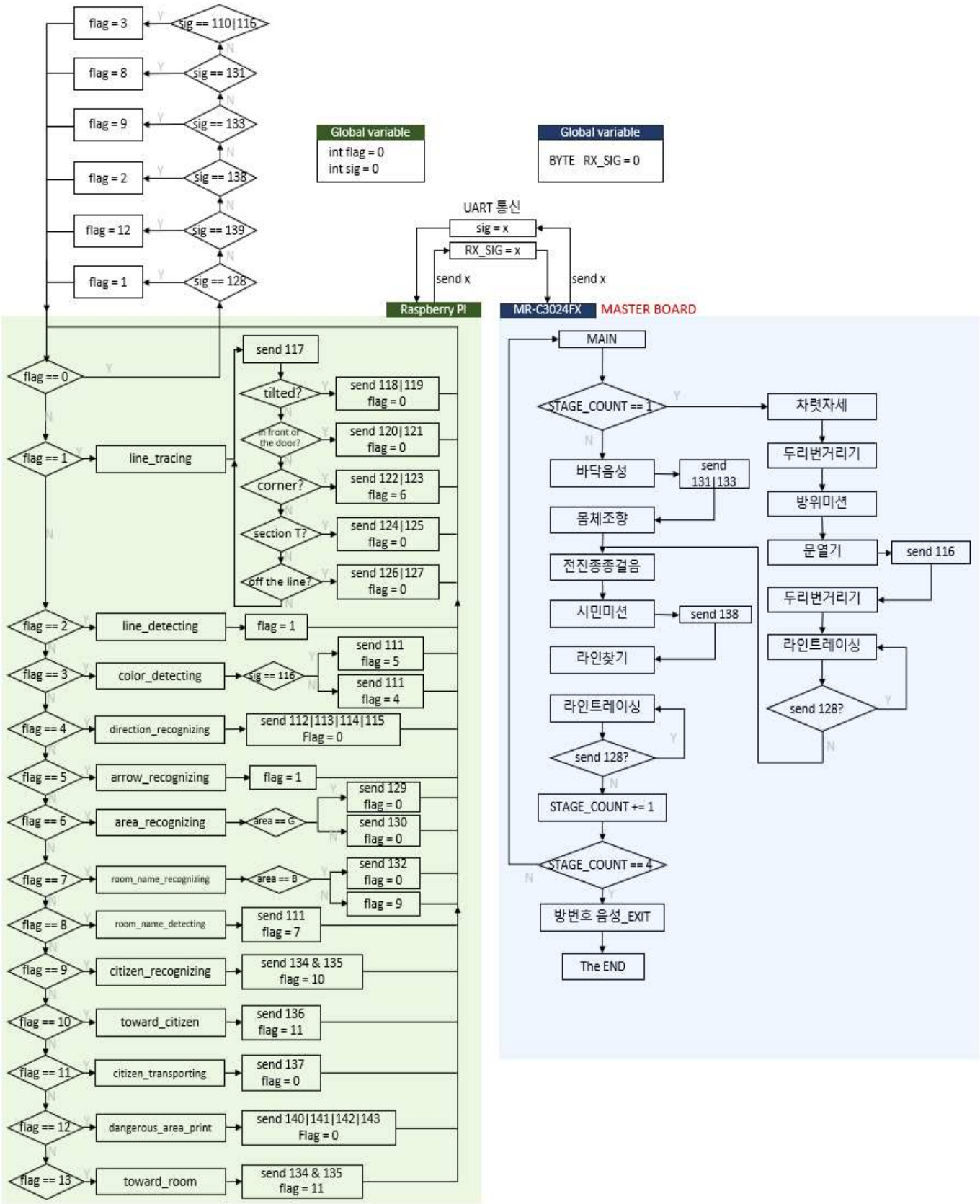
2.2.1 소프트웨어와 하드웨어 관계



<sw와 hw의 구조도>

소프트웨어와 하드웨어 간의 관계는 위의 그림과 같다. 경기에 사용되는 MF-RAPI4 로봇은 두뇌보드인 라즈베리파이와 제어 보드인 MR-C3024FX, 두 개의 보드로 구성되어 있다. 라즈베리파이에서는 카메라를 이용해 영상을 읽어 처리한다. MR-C3024FX에서는 자이로 센서와 거리센서를 통해 기울기나 사물과의 거리를 측정할 수 있고, 로보 베이직을 통해 모터와 스피커를 제어할 수 있다. 즉, 라즈베리파이의 입력장치로는 카메라가 있고, MR-C3024FX의 입출력장치로는 자이로 센서, 거리센서, 모터, 스피커가 있다.

2.2.2 소프트웨어 구조



<sw흐름도>

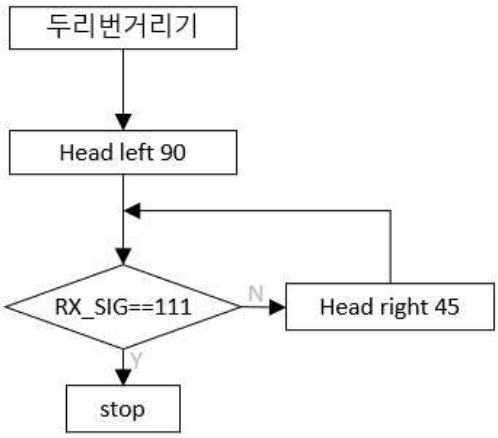
전체 SW 프로그램은 두뇌보드인 라즈베리파이와 제어 보드(임베디드 보드)인 MR-C3024FX 보드에서 실행되는 두 개의 프로그램으로 구성되어 있다. 라즈베리파이에서 실행되는 프로그램은 카메라를 통해 입력받는 영상에서 정보를 추출해서 제어 보드로 전달하는 보조적인 역

할을 한다. 라즈비안 OS에서 파이썬을 사용해 영상처리를 한다. 위의 sw흐름도에서 좌측에 위치한 흐름도가 전체 영상처리 프로그램의 흐름도이다. 제어 보드에서 실행되는 프로그램을 마스터 프로그램으로 지정을 하였다. 따라서 제어 보드에서는 센서들과 라즈베리파이에서 송신하는 값들을 통해 로봇의 모터와 스피커를 제어한다. 위의 흐름도에서 우측에 위치한 것이 제어 프로그램의 전체 흐름도이다. 위의 흐름도에선 전체적인 프로그램 흐름과 두 보드 간에 송수신하는 값들만 나타내었다. 제어 프로그램에서의 구체적인 행동 블럭들은 다음 절에서 자세히 설명한다.

송수신값	raspberryPI(**R**) ↔ MR-C3024FX(**M**)	의미
110	M → R	영상에서 방위 찾기
111	R → M	방위(또는 화살표나 방 이름) 찾았으니 두리번 멈추기
112~115	R → M	방위 인식했으니 방위에 맞는 음성출력, 모션
116	M → R	화살표 찾기
117	R → M	직진
118, 119	R → M	라인에서 좌(또는 우)로 틀어졌으니 바로잡기
120, 121	R → M	좌(또는 우)로 회전해서 문 열고 나가기
122, 123	R → M	좌(또는 우)로 45도 회전
124, 125	R → M	좌(또는 우)로 90도 회전
126, 127	R → M	라인이 중심에서 벗어났으니 좌(또는 우)로 한발짝 가기
128	M → R	118~127에 해당하는 행동 다했으니 다시 라인 영상처리
129, 130	R → M	미션지역에서 지역 종류 인식했으니 그에 맞는 음성출력
131	M → R	안전지역(혹은 확진지역) 음성 출력하고 모션 완료
132	R → M	확진 지역으로 전진해서 들어감
133	M → R	확진 지역으로 들어가기 완료
134	R → M	시민 찾기 위해 몸체 조향
135	R → M	시민 인지 되었으니 정지하고 시민 쪽으로 이동
136	R → M	시민 가까워졌으니 정지, 시민 파지하고 미션지역으로 이동
137	R → M	미션지역 도착했으니 정지
138	M → R	시민 두기 완료, 라인 탐색
139	M → R	미션 3개 다 완료하고 문 열고 나가기 완료, 음성 출력 준비
140~143	R → M	확진 지역만 방 이름 음성출력
144	M → R	방 이름 하나 출력 완료, 출력할 방 이름 남았는지 확인
145	R → M	경기 종료, 종료 세레머니

<송수신값 정리>

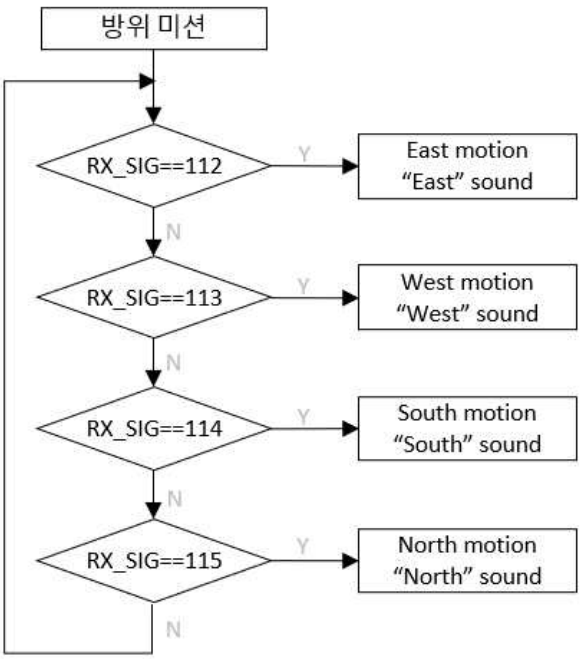
2.2.2.1 두리번거리기



<두리번 거리기 흐름도>

'두리번거리기' 라벨은 방위, 화살표 또는 방 이름을 인식하기 전에, 인식할 수 있는 최적의 상태가 되도록 영상 프레임 속에 대상을 가운데에 위치하도록 한다. 제어 프로그램에서는 로봇의 머리를 움직여 영상의 프레임이 계속 변하게 하고, 동시에 영상처리 프로그램에서는 입력 영상에서 대상의 색상 픽셀 수가 일정 값 이상이면 해당 물체가 프레임 속에 존재한다고 판단하고 제어 보드에게 정지하라는 뜻의 111 값을 송신한다. 로봇의 모션 관점에서 설명하면 맨 처음 좌측으로 고개를 조향한 다음 대상이 프레임에 들어왔다는 신호를 받을 때까지 우측으로 45도씩 고개를 조향한다.

2.2.2.2 방위미션

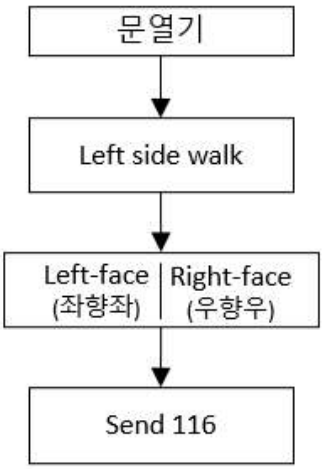


<방위미션 흐름도>

'방위미션' 라벨은 로봇이 문 앞에서 방위 표식을 인식한 후 해당 방위에 맞는 모션을 취하고 음성을 출력하는 미션을 수행하기 위한 라벨이다. 제어 프로그램에서는 영상처리 프로그램에서 각각의 방위에 해당하는 신호를 보낼 때까지 대기하다가 신호를 수신하면 정해진 모

션을 하고 해당 방위의 음성을 출력한다. 영상처리 프로그램에서는 영상에서 '문자'에 해당하는 부분을 추출하고 해당 부분의 테두리를 미리 저장해놓은 템플릿 이미지와 매칭하여 E, W, S, N 중 어떤 문자와 특징점이 가장 비슷한지 판단하여 문자를 인식한다. 그런 후 미리 정해둔 송수신 값을 제어 프로그램에게 전달한다.

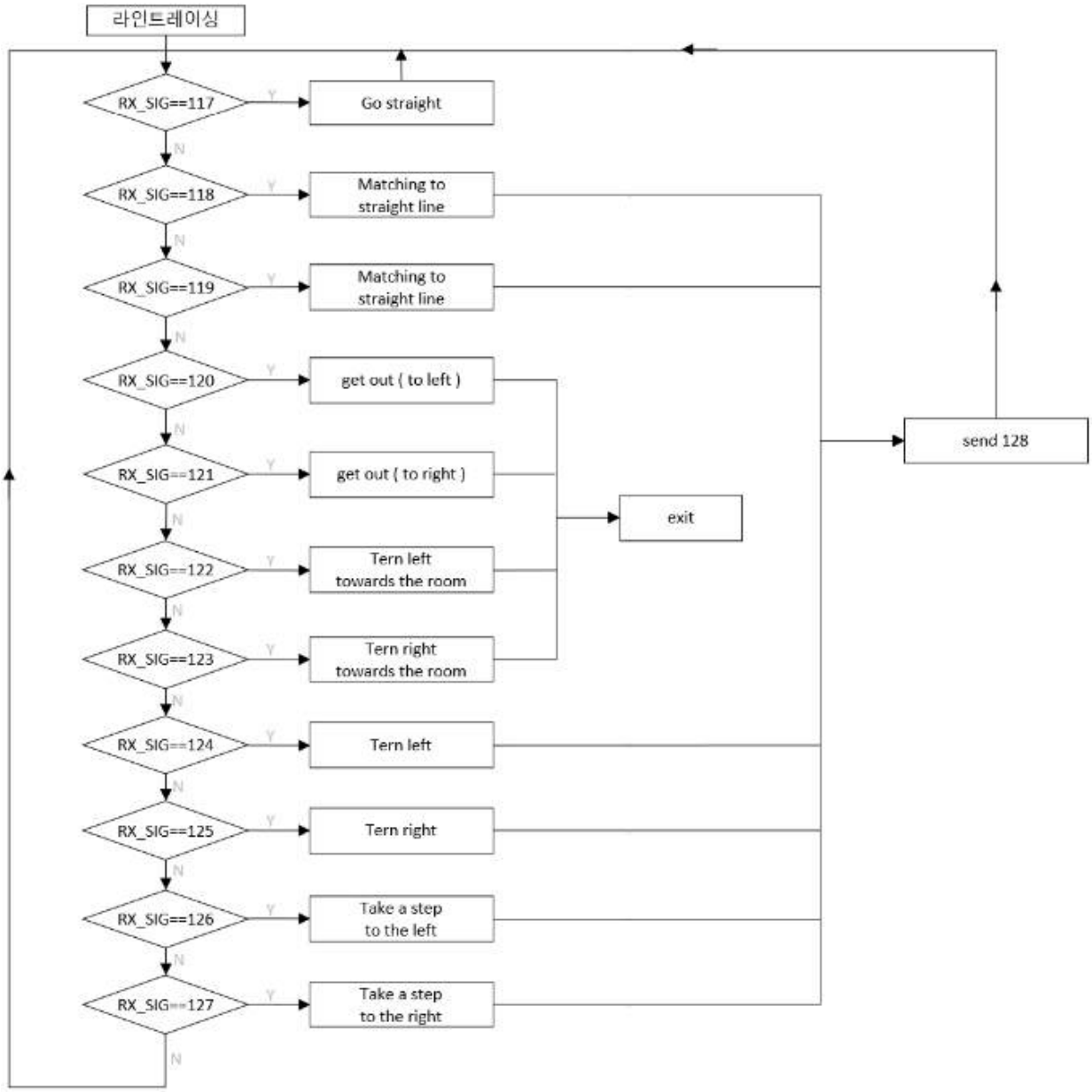
2.2.2.3 문열기



<문열기 흐름도>

'문열기' 라벨은 경기장 안으로 로봇이 들어가거나 미션을 모두 완료한 후 경기장에서 나오기 위해 문을 열 때 사용하기 위한 라벨이다. 문을 열 때 가장 중요한 것은 안정적인 자세로 넘어지지 않는 것인데, 그렇게 하기 위해 로봇의 자세를 낮춰 무게 중심을 아래로 내리고 팔을 벌려 옆으로 문을 미는 행위를 한다. 문을 열고난 후 다음 미션을 하기 위해 116의 송수신 값을 라즈베리파이로 송신한다.

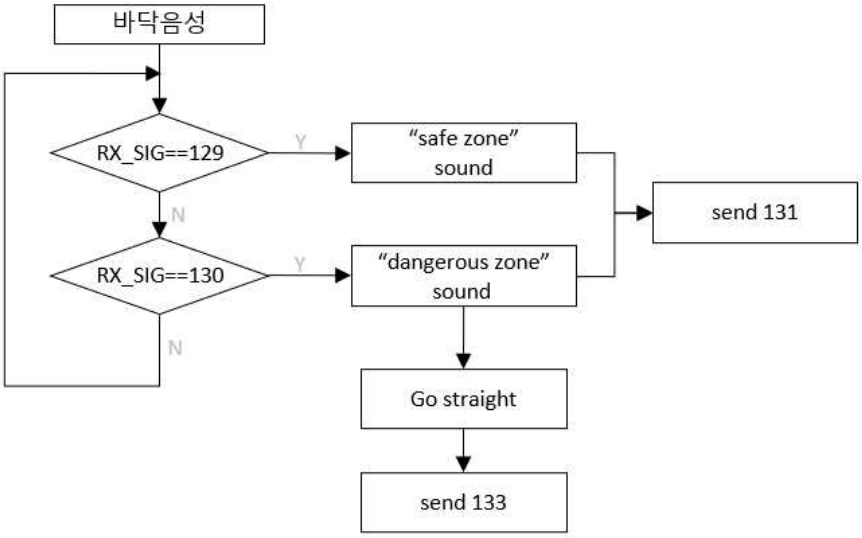
2.2.2.4 라인 트레이싱(모션제어)



<라인트레이싱 흐름도>

'라인트레이싱' 라벨은 영상처리 프로그램에서 영상 속 라인을 인식하고 현재 상태에 대한 정보를 송신하면, 제어 프로그램에서 해당 정보를 통해 로봇이 안정적으로 라인을 따라 걷게 하는 라벨이다. 영상처리 프로그램에서 제어 보드로 송신하는 정보는 6가지로 분류할 수 있다. 라인이 중앙에 바르게 있는가, 라인이 기울어졌는가, 문 앞인가, 코너 구간인가, 화살표 앞 구간인가, 라인이 중앙에서 벗어났는가에 대한 정보이다. 이러한 정보를 수신하면 제어 프로그램에서는 로봇이 각각 다음과 같은 행동을 취한다. 직진, 기울어진 방향 반대 방향으로 조향, 경기장 밖으로 나가야 하는지 판단, 미션 지역을 향해 조향, 좌회전 또는 우회전, 라인이 중앙에서 벗어난 반대 방향으로 한 걸음 가기. 이 행동들 중에 바로 다음 미션을 진행해야 하는 행동이 아닌 라인을 정확히 따라가기 위한 행동들은 행동을 취한 후에도 계속 라인을 따라가야 하기 때문에 128의 값을 송신해 영상처리 프로그램에서 계속 라인을 추적하도록 하고 제어 프로그램에서도 수신할 값을 계속 기다린다.

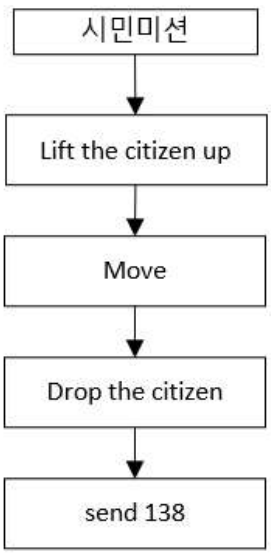
2.2.2.5 바닥음성



<바닥음성 흐름도>

'바닥음성' 라벨은 영상처리 프로그램에서 미션 지역의 바닥 색을 인식하여 안전지역인지, 확진 지역인지 판단하여 지정된 값을 송신하면 제어 프로그램에서 각 지역에 따라 정해진 미션을 수행하는 라벨이다. 만약 영상처리 프로그램에서 안전지역이라고 판단하여 129의 값을 송신했다면 제어 프로그램에서는 '안전지역'이라고 말하는 음성파일을 출력한다. 반대로 확진 지역이라고 판단하여 130의 값을 송신했다면 '확진 지역'이라고 말하는 음성파일을 출력하고 직진보행을 하여 확진 지역 안으로 들어간다. '라인트레이싱' 라벨에서 코너 구간에 미션 지역 쪽으로 조향하도록 했기 때문에 직진 보행을 하는 것만으로 확진 구역에 진입할 수 있다. 확진 지역일 때만 진입하는 이유는 시민을 파지하여 확진 지역 밖으로 이동해야 하기 때문에 미리 진입하여 다음에 시민을 인식할 때 확진 지역 안에 있는 시민을 인식하기 위함이다.

2.2.2.6 시민미션

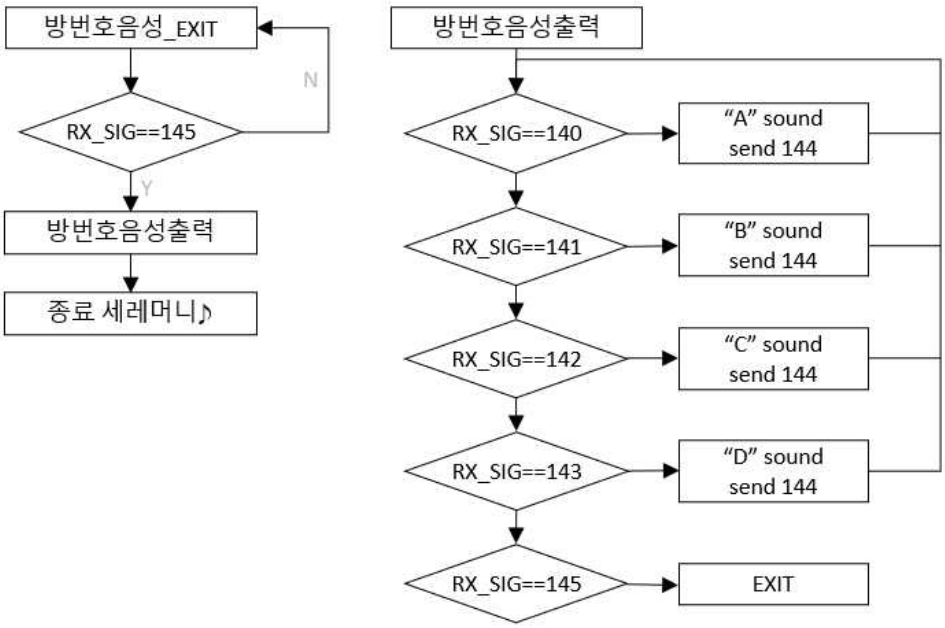


<시민미션 흐름도>

'시민미션' 라벨은 영상처리 프로그램에서 시민을 인식한 후 제어 프로그램에서 로봇이 시

민을 들어올려 정해진 구역으로 시민을 이동시키기 위한 미션을 진행하기 위한 라벨이다. 시민 파지 모션 시 가장 중요한 점은 무게 중심을 잃지 않고 안정적인 자세로 시민을 들어 올리는 것이다. 따라서 전체적으로 로봇의 자세를 낮춰 안정적이게 하고 각 팔의 움직임의 시점이 차이가 나지 않게 하여 동시에 움직이는 것과 같은 효과를 내 정확하게 파지할 수 있도록 한다. 시민을 들어 올린 후 각 미션 지역 별로 상이한 구역으로 시민을 들고 이동해 시민을 떨어뜨린다. 이때, 이동하는 동안 시민을 놓치지 않고 안정적으로 파지한 채로 이동한다.

2.2.2.7 방번호음성



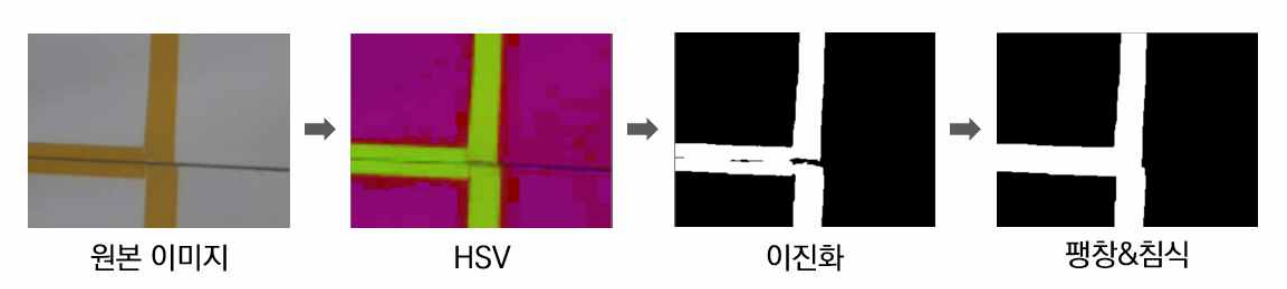
<방번호음성_EXIT과 방번호음성출력 흐름도>

'방번호음성_EXIT' 라벨은 경기장 내부에서 3번의 미션을 완수한 후 경기장 밖으로 나와서 확진 지역에 해당하는 방 번호를 순서대로 출력하는 미션을 수행하기 위한 라벨이다. 로봇이 경기장 밖으로 나와서 경기를 종료한다는 의미의 값 145를 제어 프로그램이 수신하면 로봇은 '방번호음성출력' 라벨을 수행한다. 영상처리 프로그램에서는 확진 지역의 이름을 순서대로 기억했다가 각 방 이름에 해당하는 신호를 하나씩 보내주는데 140~143 사이의 값이다. 제어 프로그램은 이 범위 안에 있는 값을 수신하면 해당 방 이름을 말하는 음성파일을 출력하고 다시 영상처리 프로그램에 144의 값을 보내 출력할 확진 지역이 더 남았는지 확인한다. 더 남아있으면 140~143 범위의 값을 또 보내주고 없으면 145를 보내 해당 라벨에서 벗어나게 한다. '방번호음성출력' 라벨이 종료되면 로봇은 종료 세레머니 모션을 수행한 후 경기를 마무리한다.

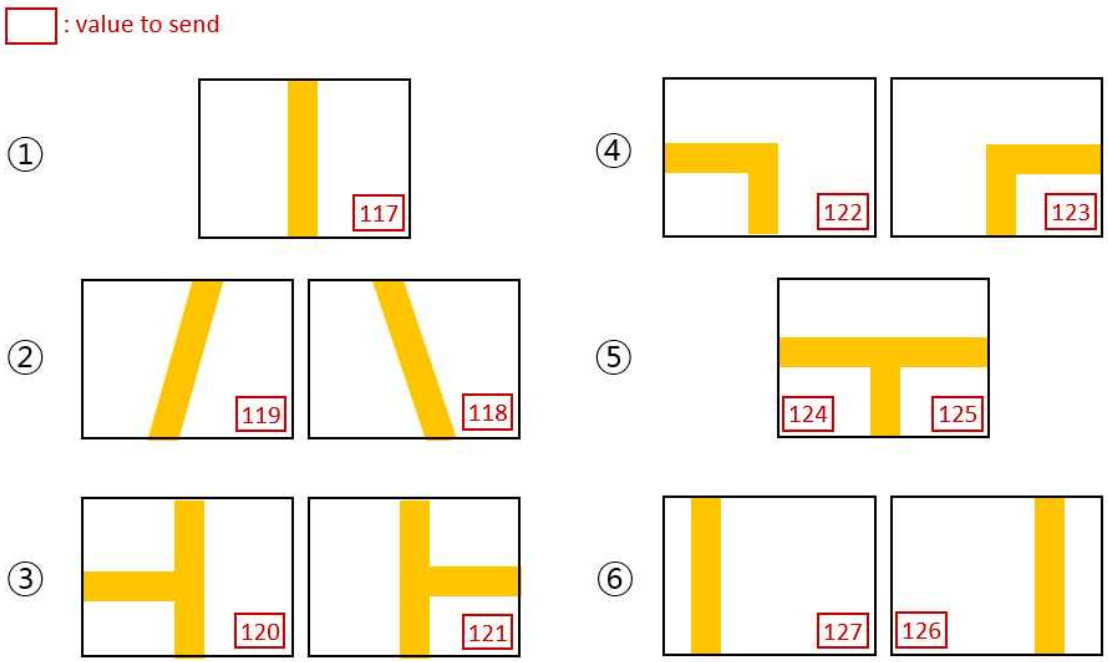
2.2.3 라인 트레이싱(영상처리)

이 절에서는 영상처리 프로그램에서의 라인 트레이싱 알고리즘에 대해 설명한다. 라인 트레이싱 알고리즘은 입력받은 영상에서 라인을 인식하고 해당 라인의 기울기나 모양 정보를 통해 라인을 따라 걷고 있는 로봇의 현재 상태를 파악하는 것이다. 로봇의 보행 상태를 파악하면 로봇이 안정적으로 라인을 따라 걸을 수 있도록 제어 프로그램에 UART 통신을 통해 미리 지정된 값을 송신하여 그에 따른 행동을 할 수 있게 한다. 라인 트레이싱 알고리즘에서 가장 선행되어야 하는 것은 라인을 인식하는 것이다. 따라서 몇 가지의 영상처리 알고리

즘을 통해 영상에서 배경과 라인을 구분한다. 라인을 추출하기 위한 알고리즘은 아래와 같다.

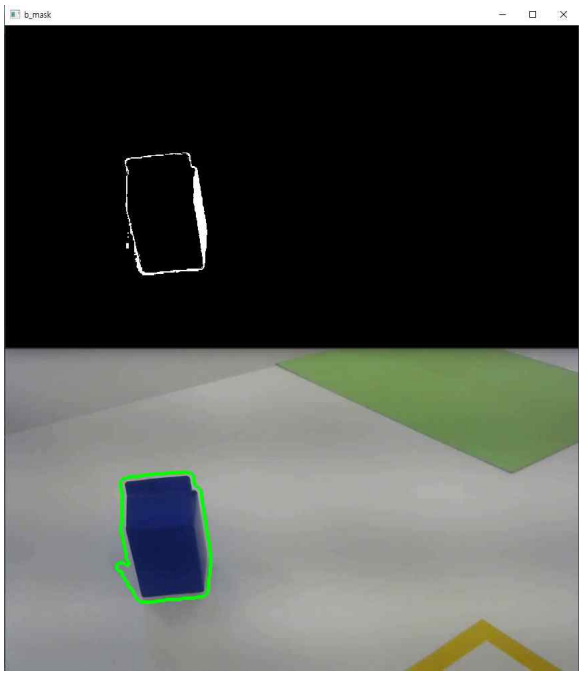


우선 입력 영상의 원본은 R, G, B 3개의 채널로 구성된 컬러 이미지이다. 하지만 RGB 색 공간의 단점은 빛과 같은 노이즈에 취약하다는 것이다. 조명이나 역광 그늘의 유무 등 다양한 환경적 변화 요인은 RGB 색 공간을 통해 동일한 색상을 추출하기 어렵게 한다. 따라서 빛의 노이즈에 보다 강한 HSV 색 공간으로 변환한다. 그런 다음, 라인에 해당하는 부분의 H(색상), S(채도), V(명도) 범위 안에 있는 픽셀은 1로 그 밖의 범위에 있는 픽셀은 0으로 바꾼 이진화 영상을 만든다. 그리고 마지막으로 이진화 영상을 팽창, 침식의 모폴로지 연산을 거쳐 노이즈를 없애준다. 이 과정을 거치면 영상에서 라인에 해당하는 부분만 추출할 수 있다.

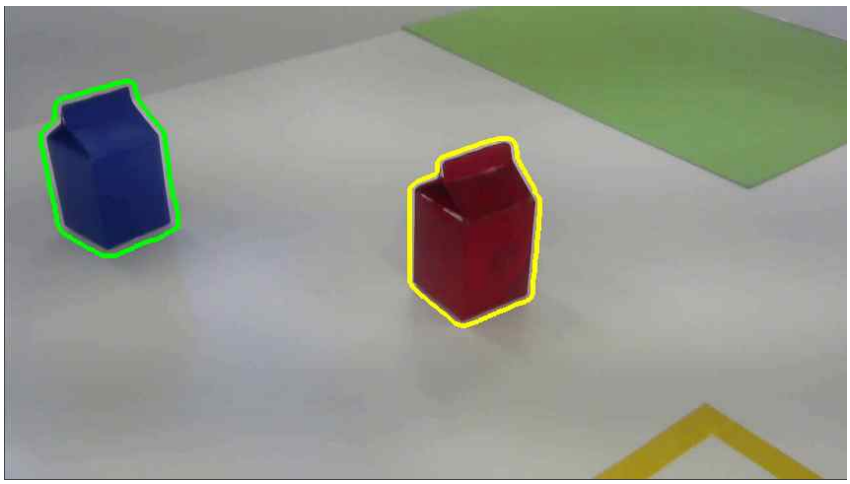


경기장에 있는 라인들을 촬영한 입력 영상은 위의 그림처럼 6가지로 분류할 수 있다. 위에서 설명한 영상처리 과정을 거쳐 추출한 라인을 기반으로 라인의 기울기와 라인 중점의 위치를 통해 6가지의 경우의 수를 구분한다. 라인의 기울기를 구하는 것을 허프변 환 알고리즘을 이용해 구한다. 라인의 중점의 좌표를 구하는 것은 이진화로 된 라인 이미지에서 1의 값을 가지고 있는 픽셀(즉, 라인에 해당하는 부분)의 좌표를 구하고 그 좌표의 중앙값을 구한다.

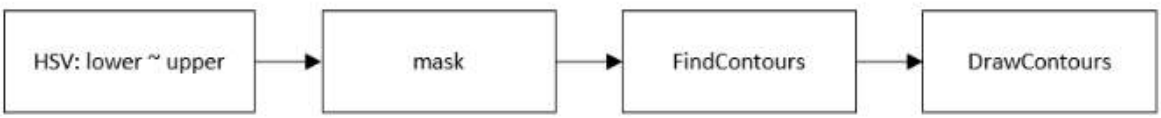
2.2.4 색 인식



<파란색 공간 mask 이미지와 원본 영상에 drawContours 처리한 이미지>



<색 인식을 이용해 찾은 파란색과 빨간색 객체>



HSV 영상에서 인식하고자 하는 객체의 색 범위를 찾아 저장한다. 이때, 영상을 RGB가 아닌 HSV로 변환한 영상을 사용하여 색상, 채도, 명도에 따라 더 정확하게 객체를 인식하도록 하였다. 지정한 HSV 범위 안쪽에 있는 픽셀들은 하얀색, 그 외는 검은색으로 처리한 mask 영상을 만든다. findContours를 활용해 mask 영상에서 흰 픽셀들을 따라 객체의 외곽선을 찾았으면 drawContours 함수로 원본 영상에 그려준다. 색 인식을 통해 확진 지역, 안전지역, 시민 구별 등에 활용할 수 있다.

2.2.5 글자 인식

<reference binary 이미지>

<reference dilation-erosion 이미지>

<template contour 이미지>

<글자 인식 이미지>

가장 먼저, 로봇이 촬영하고 있는 영상에서 글자가 위치한 곳을 알아야 한다. 영상의 edge를 사용하여 글자가 있는지 판단할 수 있다. 본 팀은 findContour 기술을 사용하여 글자의 edge를 따오는 작업을 하였다. 조금 더 선명하고 명확한 edge를 찾기 위해 gray scale로 변환 후 인식한 글자의 픽셀만 검은색으로 바꾸는 binary 작업을 하였다. 그 후 침식, 팽창의 모폴로지 연산을 하여 문자의 가장자리만 따온 뒤 findContours 함수로 가장자리 좌푯값을 받아왔다. drawContours 함수로 원본을 복사해 둔 영상에 좌푯값을 찍으면 글자의 모양대로 경계선을 그려줄 수 있다. 그다음 글자가 위치한 영역만 정사각형으로 잘라 템플릿으로 설정하고 레퍼런스 영상과 비교하여 어떤 글자인지 찾을 수 있다. 즉 레퍼런스 영상은 미리 준비해 둔 알파벳 사진이 포함된 영상이고, 카메라로 받아오는 영상의 글자 영역이 템플릿이 된다.

3. 개발 중 장애요인과 해결방안

3.1 프로그램 구조화

개발하면서 가장 어려움을 느꼈던 것은 영상처리를 위한 파이썬 코드와 모션 제어를 위한 로보 베이직 코드가 시간이 지나면서 복잡해질수록 프로그램의 흐름을 파악하기가 어려워지고 송수신 값이 중첩되어 오류를 많이 범하게 되었다. 이런 상태로 진행하기 어렵다고 판단하여 도중에 개발을 중단하고 파이썬과 로보 베이직 프로그램을 구조화하고 도식화시키는 과정을 먼저 진행하였다. 영상처리 담당 팀원들과 모션제어 담당 팀원들이 지속적으로 각자의 프로그램의 흐름을 공유하고 송수신에 필요한 값들도 개발 이전에 미리 정해놓았다. 그 결과 개발을 직접 시작한 시점은 늦어졌지만, 개발을 시작하고 나서부터는 흐름의 꼬임 때문에 생기는 오류는 발생하지 않았고 충분한 커뮤니케이션 과정을 거쳤기 때문에 팀원 모두가 프로그램의 흐름을 파악하고 있어 진행이 용이해졌다.

3.2 딥러닝 기반 글자 인식

글자 인식 프로그램을 할 때 pytesseract 라이브러리를 사용한 OCR 알고리즘을 계획하고 시도하였지만 실패하였다. tensorflow를 라즈베리파이에서 실행하기에 용량이 비교적 큰 것도 문제였고 사이즈가 작은 문자들은 잘 인식하였지만 큰 문자들은 인식을 잘 하지 못하는 문제도 발생하였다. 또한 딥러닝을 사용한 글자 인식은 템플릿 매칭 기법에 비하여 시간도 오래 걸렸고 용량과 비용적인 측면에서도 템플릿 매칭이 훨씬 뛰어났다. 템플릿 매칭은 문자 인식 시간이 짧고 비용적인 측면에서 좋은 모습을 보여준다는 장점에 더해 정확도 또한 높았다. 알고리즘과 코드도 어렵지 않게 제작할 수 있어서 템플릿 매칭을 사용하여 글자 인식을 하게 되었다.

4. 개발의 차별성

4.1 영상의 글자 인식

로봇이 미션을 수행하기 위해선 영상을 읽어와서 그 데이터를 통해 어떤 행동을 취할지 명령을 주어야 한다. 미션 중 영상의 글자를 인식하여 그 방향으로 움직이는 모션이 필요한데, 이때 영상의 글자 인식을 어떻게 해야 하는지 고민하였다. 딥러닝을 통하여 모델을 학습시킨 후 입력을 주어 출력값을 내보내 결과를 얻어내는 방법이 가장 일반적인 방법이었다. 하지만 이 방법은 많은 학습 데이터가 필요할 뿐만 아니라 영상의 결과를 얻어내는 데에도 시간이 오래 걸린다는 결론이 나왔다.

따라서 본 팀은 템플릿 매칭이라는 방법을 사용하여 글자 인식을 하게 되었다. 기본적인 글자 인식 알고리즘을 설명하면, 먼저 이미지를 모노 영상으로 변환한 후 이진화 하여 문자의 픽셀과 배경의 차이를 둔다. 이후 침식과 팽창을 통해 문자의 경곗값만 따온 후 함수처리하여 외곽선 좌표값을 따온다. 원본을 복사해둔 영상에 경계선을 그려주면 이제 글자의 모양대로 경계선이 나오게 된다. 후에 외곽선 좌표값으로 글자가 위치한 영역만 정사각형으로 잘라 인식한 글자와 방위 사이의 모양을 비교하게 되는데 이때 템플릿 매칭 기법이 사용된다. 방위가 적혀있는 그림을 미리 준비한 후 정사각형으로 가져온 글자 외곽선 영상과 비교하여 두 영상 사이의 매칭 값을 구하게 된다. 매칭 값이 기준치 이상인 글자를 인식한 글자로 판단하여 로봇에게 인식한 방위 값을 넘겨주는 것이다. 이 방법을 사용함으로써 본 팀은 더 빠르고 효율적인 글자 인식 알고리즘을 구현하였다.

4.2 두뇌 보드와 제어 보드의 송수신

본 대회의 미션을 수행하기 위해선 영상처리를 하는 두뇌 보드와 모션을 제어하는 제어 보드가 서로 상호작용해야한다. 이를 위해 송신 값과 수신 값이 필요한데, 미션의 개수도 작지 않으며 영상처리를 통해 나온 결과에 따라 해야 하는 모션들이 상이하다. 따라서 통신 값들의 중첩으로 인한 에러를 방지하기 위하여 체계적으로 송신 값과 수신 값을 미리 정립해야 한다고 생각했다. 본 팀은 미션의 수행 순서에 따라서 제어 보드 알고리즘을 작성하였고 그 알고리즘의 흐름을 기반으로 송신 값과 수신 값을 설정하였다. 미리 송수신 값을 정립해 놓았기 때문에 송수신 값의 문제로 로봇이 원하는 출력을 하지 못하는 오류를 잡았고, 두뇌 보드의 정확한 명령과 제어 보드의 확실한 출력을 제어할 수 있게 되었다.

5. 개발 일정

No	내용	2021年				
		6月	7月	8月	9月	10月
1	영상처리 이론 학습					
2	관절 제어를 위한 기구학, 역기구학 학습					
3	개발 환경 구축					
4	ORB알고리즘을 통한 객체 인식					
5	Template Matching을 이용한 문자 인식					
6	라인트레이싱 기반 걸음새 생성					
7	보행 궤적 안정화					
8	미션 수행을 위한 모션 구현					
9	두뇌보드와 제어보드의 송수신 제어					
10	미션 흐름에 따른 전체적인 코드 생성					
11	테스트 및 디버깅					
12	오류 검출 및 해결					
13	최종 보고서 작성					

6. 팀 업무 분장

No	구분	성명	참여인원의 업무 분장
1	팀장	정다은	전체 프로그램 구조 구축 라인 관련 영상처리 담당 영상처리 알고리즘 구상 및 구현
2	팀원	김나영	객체 인식 관련 영상처리 담당 문자 인식 관련 영상처리 담당 영상처리 알고리즘 구상 및 구현
3	팀원	김선진	전체 프로그램 구조 구축 로봇 제어 및 모션 프로그램 구조 구축 모션 관련 알고리즘 구상 및 구현
4	팀원	이슬기	전체 프로그램 구조 구축 모션 관련 알고리즘 구상 및 구현 미션 알고리즘 개발