

# **ME400: 창의적 시스템 구현 - 최종 보고서**

2018-spring

**Team: 걷지 말고 기어**

20150760 최 원

## **§목차**

### **1. Hardware Design**

#### **1.1 Pick-up module**

#### **1.2 Fin & Fan**

#### **1.3 Gear-Wheel System**

#### **1.4 차체**

### **2. Software design**

#### **2.1 Overall algorithm**

#### **2.2 Ball detection**

#### **2.3 Pick the ball**

#### **2.4 go to the basket**

### **3. Process**

#### **3.1 Monthly plan**

#### **3.2 SolidWorks**

#### **3.3 Labview**

#### **3.4 OpenCV**

#### **3.5 ROS**

### **4. Additional Materials**

# 1. Hardware Design

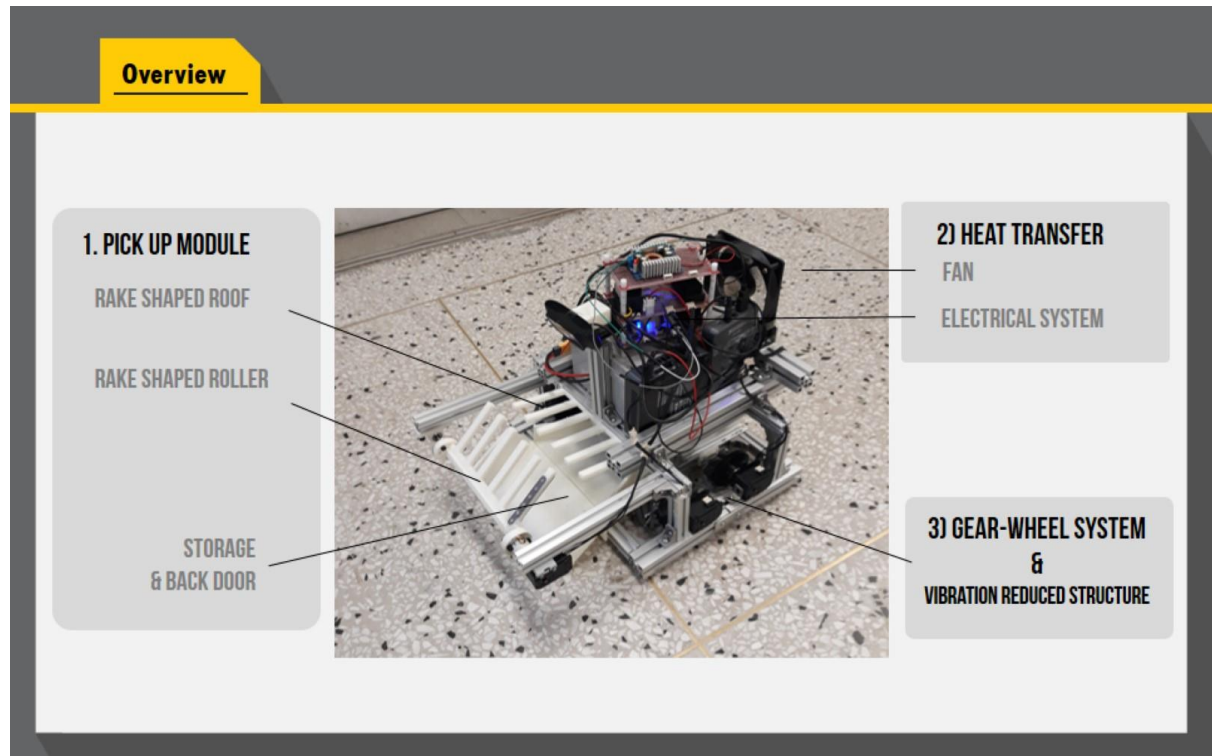


Figure 1 Overview of hardware

Mobile platform의 구조는 위와 같다. 특징적인 구조는 pick-up module, 열 전달 설계(fin & fan), 그리고 기어 시스템이 있다. 카메라 위치, 차체의 크기 또한 최적으로 설계하였다.

## 1.1 Pick up module

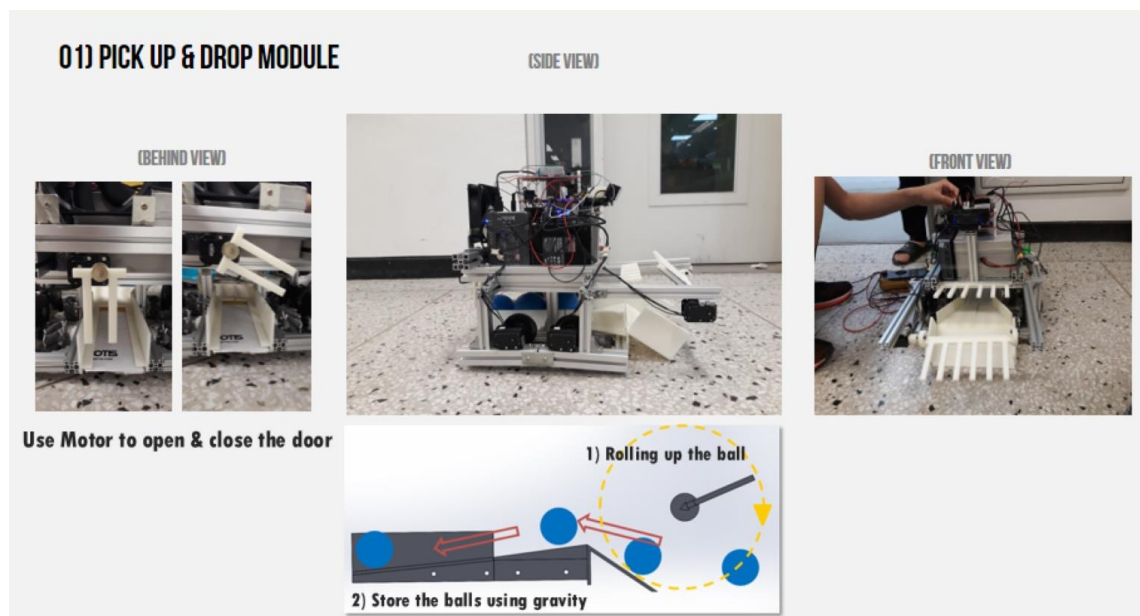


Figure 2 pick up & drop module

다음은 공을 주워 담고, 내려놓는 모듈의 앞, 옆, 뒤에서 본 모습이다. 공은 차가 앞으로 이동하면서 앞 쪽에 있는 롤러에 의해서 앞쪽 경사면을 타고 들어올려진다. 경사면을 지나고 난 이후에는 중력에 의해 뒤쪽으로 굴러갈 수 있도록 경사로 형태로 저장소를 설계하였다.

### 1) Roller design

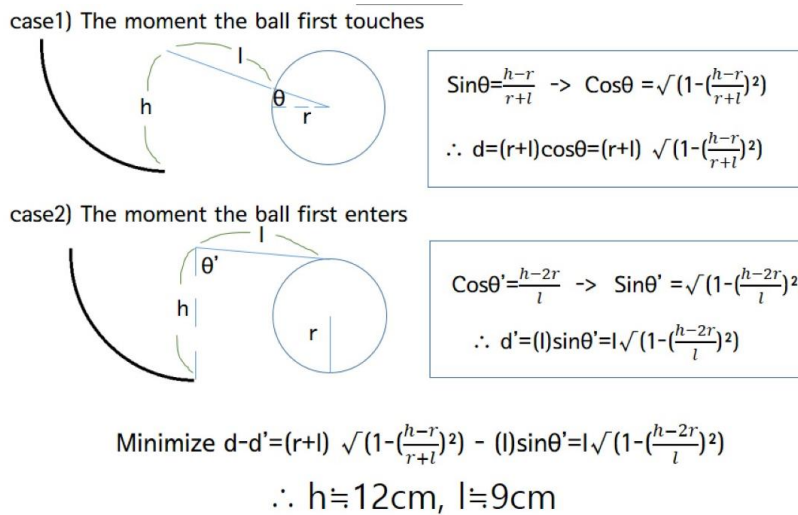


Figure 3 optimal dimension for roller

Figure 3 은 공을 잡을 확률이 가장 높은 roller 의 길이와 roller 중심 높이의 위치를 계산한 것이다. 그러기 위해서는 공을 놓치는 공의 위치 범위를 최소로 만들어야 한다. 그 결과 roller 의 길이는 9cm, 중심의 높이는 12cm 로 설정되었다.

또한 roller 에 의해 공이 다시 빠져나갈 수 없도록 위쪽에 가림 막을 설치하였다.

### 2) drop module

저장고가 뒤쪽으로 경사져 있기 때문에 뒤에 아무런 장애물이 없다면 공이 자연스럽게 떨어지도록 되어있다. 따라서 공을 주울 때는 저장소의 뒤쪽을 막아 두었다가 공을 떨어뜨릴 때에만 막아둔 것을 치우도록 했다. 다른 전원을 쓰지 않는 방법도 존재했지만 설계의 복잡도가 올라간다. 따라서 모터를 이용해 뒤의 장애물을 회전시켜 떨어뜨리도록 했다.

### 3) rake shape

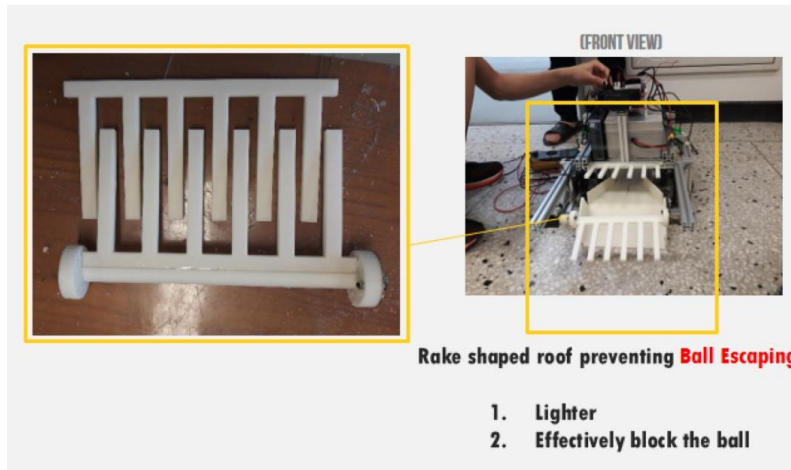


Figure 4 rake shaped

가림 막의 설치가 용이하고, 무게를 줄여 에너지 효율을 높이기 위해서 TRIZ 공간의 분리를 이용해 위와 같은 빗 모양의 구조를 사용하였다.

## 1.2 Fan & Fin

차체의 최고 온도가 기준 온도인  $70^{\circ}\text{C}$  를 넘지 않았지만 가장 많은 에너지가 사용되기에 온도가 가장 높아졌던 모터 컨버터 위주로 냉각을 위한 설계를 진행하였다. Conduction은 냉매를 이용해야해서 설계의 복잡성이 커지고, radiation은 건물 내벽의 온도가 차체의 절대온도와의 차이가 크지 않기 때문에 영향이 적을 것으로 판단되어 fin 과 fan 을 이용한 forced convection 을 사용하였다.

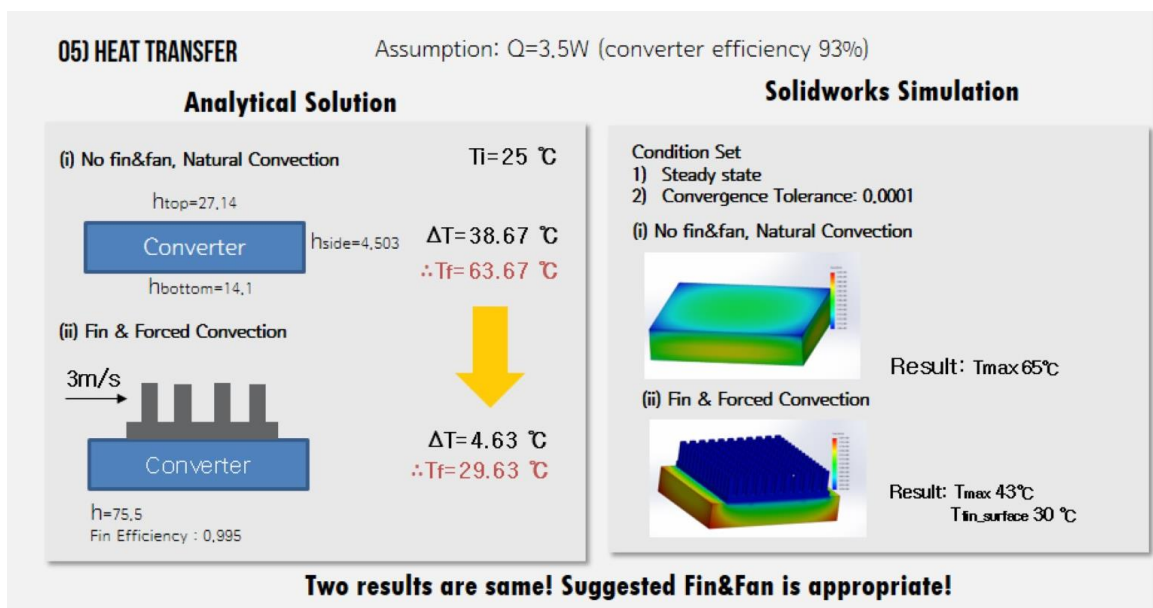
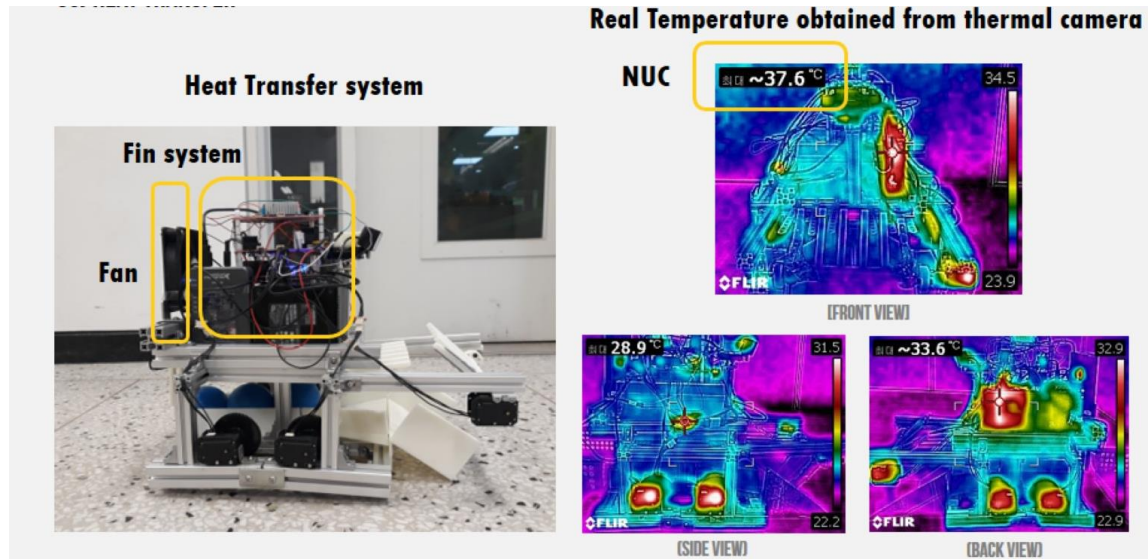


Figure 5 Heat analysis

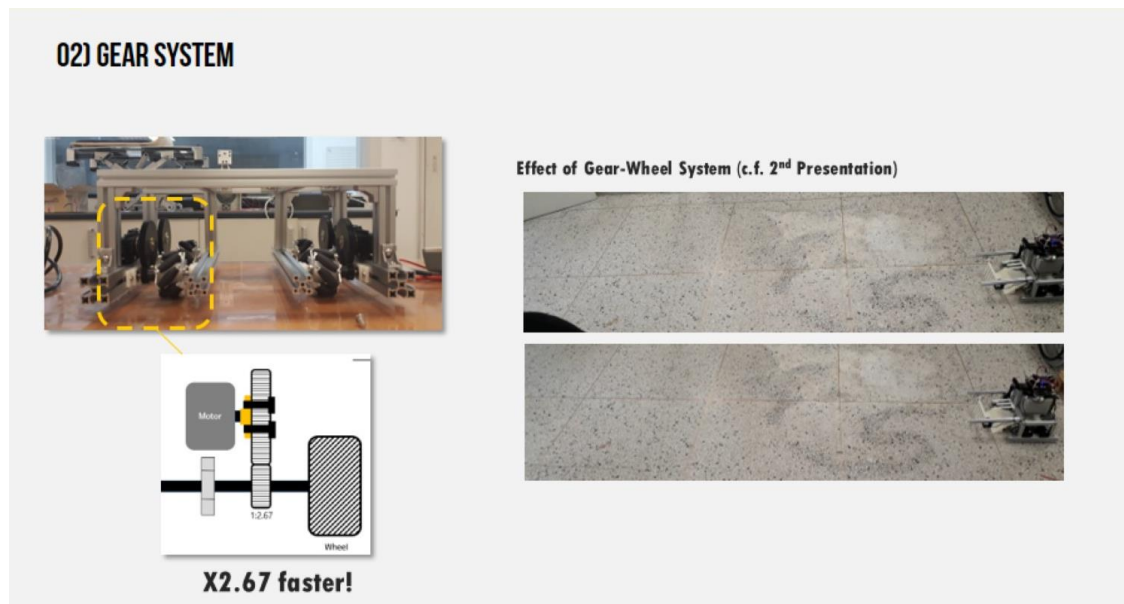
온도를 20℃ 이상 냉각시킬 수 있다는 것을 확인하였다.



**Figure 6 Temperature measured**

또한 실제 온도 측정 결과 또한 이론적인 분석을 뒷받침했다.

### 1.3 Gear-Wheel system



**Figure 7 Gear-Wheel system**

기준에 주어진 모터 MX-28AT 는 최대 55rpm 으로 매우 느렸다. 따라서 기어를 사용하여 속도를 높였다. 기어는 바퀴의 크기 등을 고려하여 20:80 즉 기어 비 약 1:2.67 로 설정했다. 결과적으로 약 2.6 배 정도 속도가 빨라진 것을 확인할 수 있었다.



또한 기어를 설계하면서 바퀴를 두 축으로 고정하면서 바퀴의 진동으로 인해 생기는 문제를 줄였다.

## 1.4 차체

### 1) rigid body problem

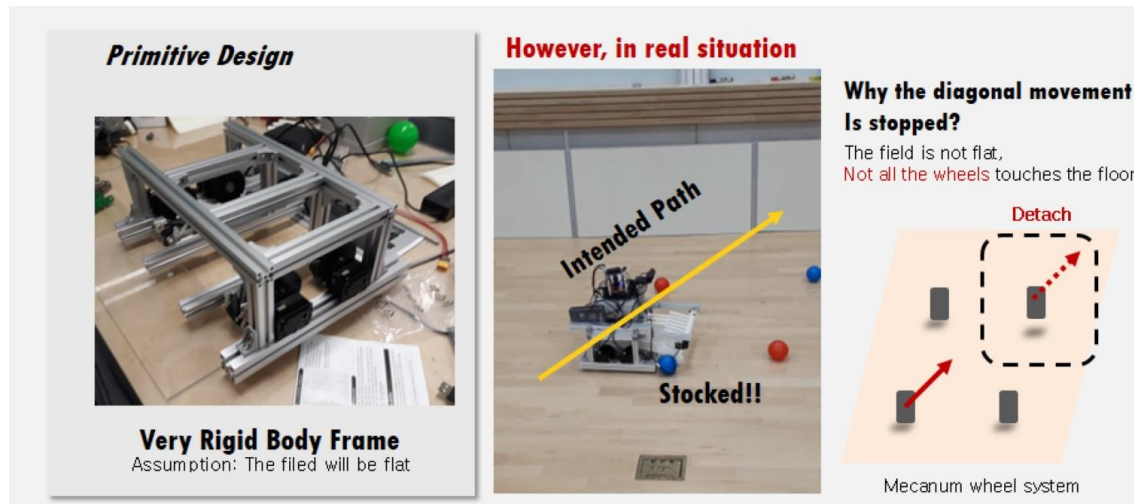


Figure 8 rigid body problem

차체를 알루미늄 프로파일로 제작하고 바퀴를 두 축 고정하는 설계는 차체를 rigid 하게 만들었다. 따라서 지면의 형태라는 noise 로 인해 모든 바퀴가 땅에 고정되어 있지 않는 상황이 발생하였고, 이는 대각선 방향 진행에 큰 문제점으로 작용되었다. 따라서 차체에 자유도를 줄 수 있는 방법을 생각했었고, TRIZ 기법을 이용하여 차체를 분리하여 회전 자유도를 부여할 수 있었다.

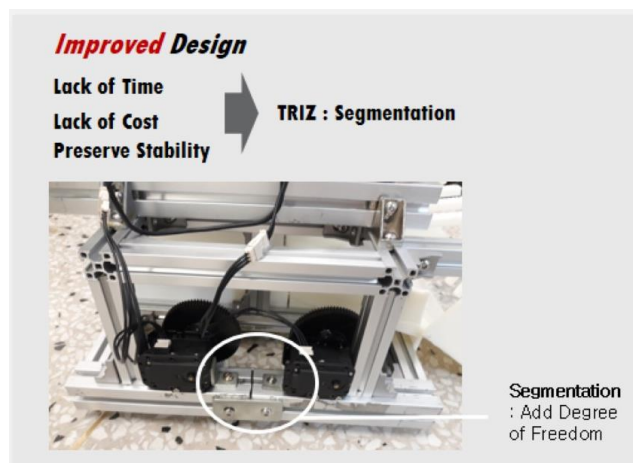
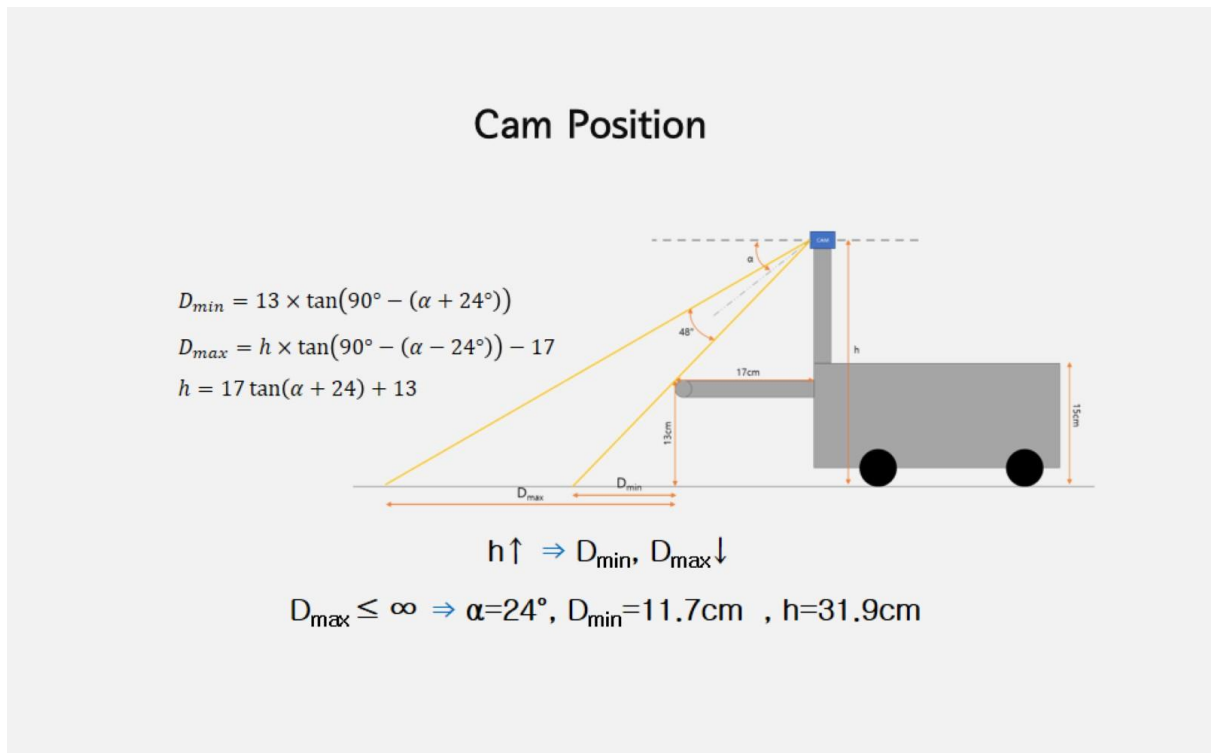


Figure 9 Frame segmentation

### 2)webcam 위치 정하기

최대한 가까운 곳까지 보면서 멀리 있는 공 또한 인식할 수 있도록 웹캠의 위치를

계산하였다. 대각선 시야각(FOV)이 76°라는 사실을 이용해 가로(66°), 세로(48°)의 시야 각을 추정해 계산했다.



**Figure 10 Webcam position**



## 2. Software Design

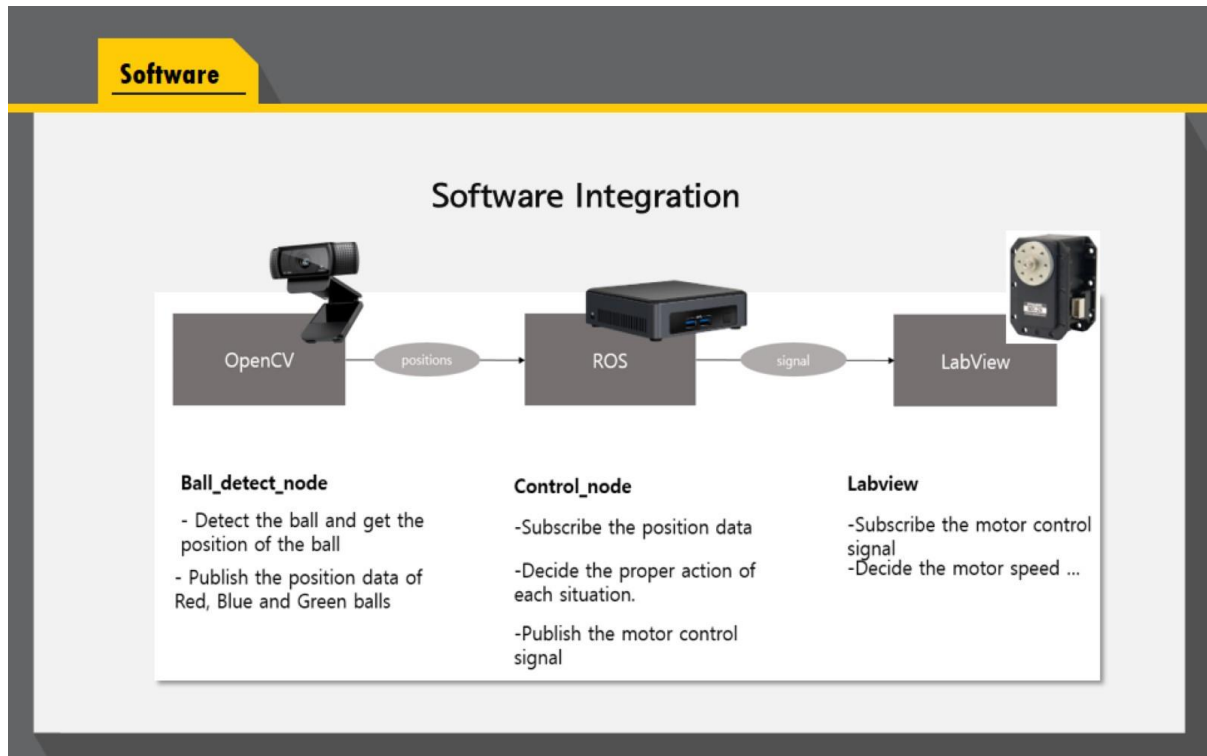


Figure 11 Software Integration

### 2.1 Overall Algorithm

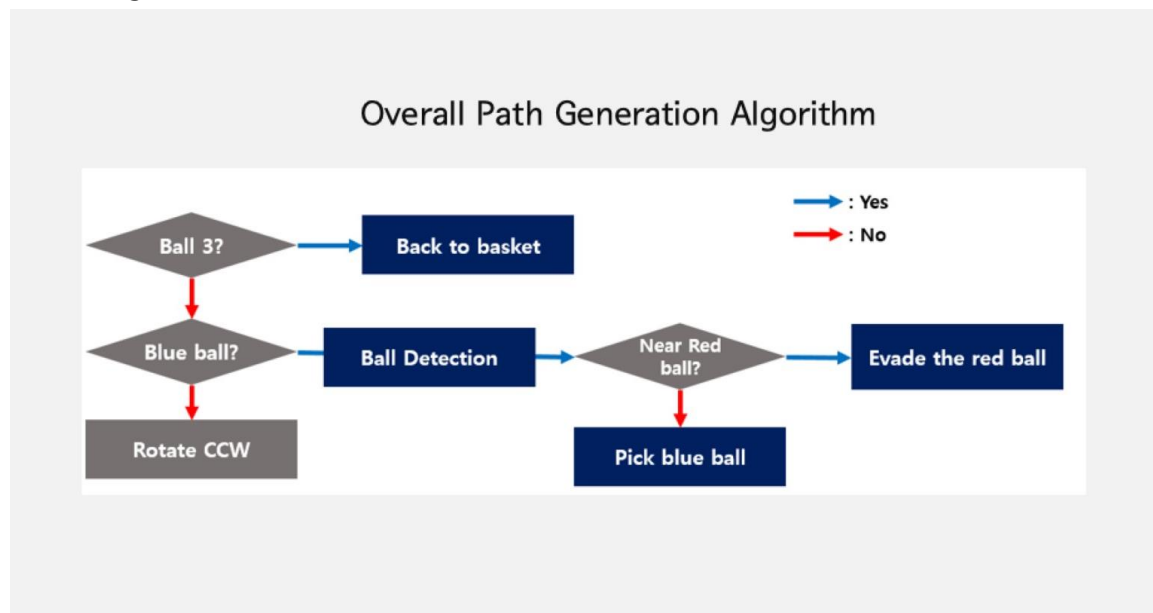
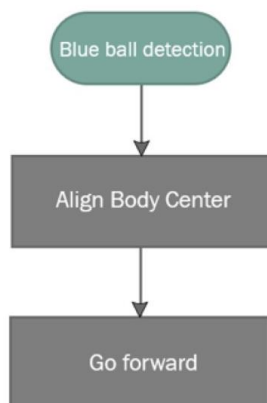


Figure 12 Overall Algorithm

전체 알고리즘은 다음과 같다. 공을 인식하고, 공에 접근하고, 파란 공이 가까워지면 차체와 파란 공 사이에 있는 빨간 공을 피하고, 공을 잡고, 공을 세번 다 주우면 바구니로 돌아간다.

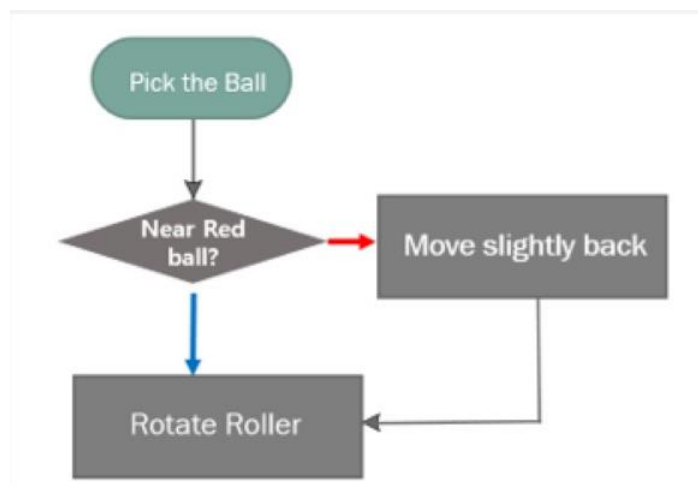
## 2.2 Ball detection



**Figure 13 Blue ball detection**

파란 공을 인식하는 과정은 위와 같다. 먼저 파란 공이 있으면 가장 가까운 파란 공을 가운데에 위치시키고 공을 향해 직진한다. 파란 공보다 거리가 가깝고, 차의 폭에 해당하는 범위 안에 빨간 공이 있을 때에는 대각선 방향으로 이동하면서 빨간 공을 피한다. 그리고 파란 공을 차체 중앙에 위치시키고 공을 향해 나아가는 것을 반복한다.

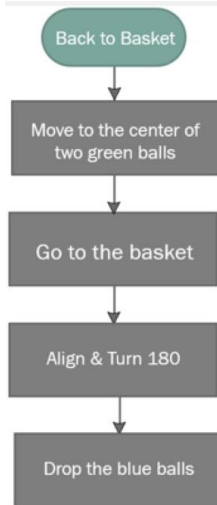
## 2.3 Pick the ball



**Figure 14 Pick the ball**

2.2 의 과정을 통해 파란 공에 접근하여 특정 거리보다 파란 공이 가까워지면 파란 공을 줍는다. 이 때 파란 공 뒤에 빨간 공이 너무 가까이 있으면 차를 조금 후진 시킨다.

## 2.4 Back to the basket



**Figure 15 Back to the Basket**

2.3 의 과정이 세 번 진행되면(파란 공을 세번 주우면) 바구니로 돌아간다. 바구니로 돌아갈 때는 바구니에 부착된 두 초록색 공을 이용한다. 먼저 두 초록 공의 수직 이등분 선으로 이동한다. 이때 두 초록색 공의 거리가 비슷해질 때를 조건으로 한다. 수직 이등분 선에 진입하기 위해서는 더 먼 쪽 공 방향 대각선으로 이동한다. 수직 이등분 선에 진입하면 차량의 진행방향 거리(카메라 좌표 계에서  $z$  방향)가 같아지게 차량을 회전시킨다. 그 후에 차량을 180 도 회전시킨다. 그리고 후진을 한다. 바구니에 차량 뒤쪽이 닿으면 차량의 뒤쪽 문을 열어 공을 방출한다.

### 3. Process

#### 3.1 Monthly Plan

3 월은 각자 파트를 공부하고 익히는데 중점을 두고 dynamixel 이 도착한 4 월, 5 월에 본격적으로 프로젝트를 계획하고 진행하였다.

#### Plan in April



Figure 16 Plan in April

## Plan in May



Figure 17 Plan in May

### 3.2 Whole process

#### 1-5 주차)

모든 파트가 각 파트 공부.

#### 6 주차)

4/2

1st prototype 제작에 대한 계획을 수립.

4/3

우선적으로 picking module, 바퀴에 부착되는 기어 시스템, path, 컨버터에 대한 설계 진행

공을 충분히 끌어들이 수 있는 조건을 만족하는 롤러 부분의 회전 팔의 길이와 회전 중심의 높이를 결정함(높이=12cm, 길이:10cm), 롤러 구조를 회전시키는데 필요한 토크 계산을 통해 필요한 모터 스펙 결정(0.2Nm 정도를 넘으면 충분히 구동 가능).

현재 mecanum wheel 에 dynamixel MX-28AT 를 이용해 구동시키면 왕복 시간만 150 초 정도 소요됨. 따라서 이동 속력을 늘리고자 기어 시스템을 설계. 기어비는 1:2 와 1:3 을 채택.

Path 는 기존에 다양한 네비게이션 알고리즘이 존재 진행상황을 지켜본 뒤(mobile platform 의 크기 등) 가장 적합한 알고리즘을 채택할 계획.

NUC, Myrio, 바퀴, picking module 용 모터에 필요한 컨버터를 찾음.

4/5

과학 상자로 대략적인 picking module 의 공이 쓸어 올려지는 부분부터 저장되는 부분까지 제작. 이를 바탕으로 picking module 의 dimension 을 정하고 solidworks 로 그림.

## 7 주차)

4/11

대화동에 가서 베어링, 전원 커넥터, 연마 봉 구매. 다이소에서 Prototype 의 몸체가 될 플라스틱 바구니 3 개 구매. 대화동에서 파는 기어는 크고 무거워서 기어는 추후에 더 알아보고 구매하기로 계획.

4/13

KHK 사의 카탈로그를 보고 내경 6 파이의 기어 결정. 기어비를 1:2 와 1:3 으로 하기 위해 잇수가 30, 60, 80 인 기어로 결정.

모터와 컨버터 주문 완료.

## 8 주차)

4/19

주문한 컨버터를 갖고 남땀을 완료하였다.

4/20

KHK 사의 내경 6 파이의 잇수가 30, 80 인 기어로 결정하여 주문하였다. (기어비 1:2.67)

4/22

기계동 로비에서 4/27 에 자체에 바퀴를 달고 움직이는 것, 시각 인식, 공을 쥌는 기능을 수행하기로 계획했다.

## 9 주차)

4/23

주문한 모터에 문제가 있어서 Robotis 의 'XL430-W250-T' 모터를 추가로 주문함.

System 이 integrate 될 수 있는 상황이 될 때까지 각 파트에서 수행해야 할 목표를 달성하기로 함.

4/25

주문한 기어가 도착했다. 생각보다 기어가 무거움. 기어와 모터를 연결하는 방법과 또다른 기어와 mecanum wheel 을 연결할 방법을 고민. 기존에 사용하기로 했던 플라스틱 상자는 하중에 의한 변형이 심하다는 한계가 존재하여 아크릴로 차체를 제작하기로 하였다.

4/26

기어가 포함된 차체의 dimension 을 정하고 아크릴로 차체 제작을 하였다.

기어와 모터는 직접 볼트로 체결하기로 하였다.

4/27

차체에 dynamixel 을 부착하고 기어 없이 mecanum wheel 을 체결하여 앞, 뒤, 양 옆, 네 가지 대각선(45 도) 방향으로 mobile platform 을 움직여 보았다. 간단히 웹캠을 차체에 없어서 ball detection 을 시행했을 때는 약간의 진동이 있었지만 ball detection 에 영향을 줄 수 있는 수준은 아니었다.

## 10 주차)

4/30

기어를 통해 이동 속도를 2.67 배 증가시킬 수 있는 차체를 제작하였다.

5/1

기어를 부착한 차체를 8 가지 방향에 대해서 구동해보았다. 초기에는 정상적으로 구동하다가 기어의 alignment 가 틀어지기 쉽다는 설계의 오점을 발견하여 5/2 일에 차체를 보완하여 재조립하기로 함.

5/2



차체를 보완하여 재조립함, Picking module 제작을 완료함

Picking module 까지 완성된 차체를 이용하여 빨간 공을 피하면서 세 개의 파란 공을 잡는 시행을 성공함.

진동 분석을 위해 카메라를 통해 받은 공의 위치 정보를 수집하여 분석함

5/3

발표 내용에 대해 교수님께 피드백을 듣고 야간에 조모임에서 열, 진동, pickup, gear 에 대한 자료를 수집하고 작성함.(진동은 1Hz 이하에서 특성이 나타나 무시할 수 있다 판단.)

## 11 주차)

5/8

기어 alignment 문제로 차체를 완성도 높여서 제작하고, 열, 진동, ball detection, 들어온 공을 check 하는 것, lidar 이용 등의 작업들이 필요하다고 판단, 업무 분담을 통해 각 부분을 완성해 나가기로 결정하였다.

5/10

차체 수정을 완료함

5/11

Picking 과 차체 움직임을 test 하고 picking module 과 차체에 대해 수정할 사항을 발견함

5/13

구매할 물품과 backdoor 에 대한 아이디어를 수렴함.

진동 분석을 위해 시각 인식을 통해 직선과 대각선 방향 회전에 대한 데이터를 수집함.

열과 진동에 관해서는 각각 김형수 교수님과 박용화교수님께 자문을 구하기로 함

## 12 주차)

5/17

공동강의실에서 첫 데모 진행

공동강의실 환경에 맞춰 공을 인식하기 위한 H,S,V 값의 조정을 함

5/18

뒷문 제작하여 MX-12W 모터와 결합하여 연결하고 구동 확인

storage 에서 공이 바구니로 잘 내려가지 않음-> storage 를 두개의 파트로 나누어서 뽑기로 결정

롤러 수정

Fin, fan, cam 등 주문

### 13 주차)

5/22

공동강의실에서 2 차 데모 진행

차체의 대각선 motion 이 원활하지 않음, 차체를 변경하거나 코드 상으로 해결하기로 결정

5/23

2 차 데모에서 대각선 motion 이 잘 되지 않는 것을 차체가 rigid 하기 때문인 것으로 판단. 각 방향 바퀴가 따로 움직일 수 있도록 자유도 하나를 추가하기로 함.

5/25

3 차데모 진행->볼트로 고정된 부분을 조금 풀어서 자유도를 추가하려고 시도 하였으나 차체가 약해질 수 있어서 다른 방법을 찾기로 함. 프로파일을 두개로 나누어 두 바퀴가 모두 바닥에 닿을 수 있도록 회전에 대한 자유도 추가.

5/26

4 차 데모 진행-> 대각선 방향은 대각 방향 속도를 높이고 자유도를 추가함으로써 해결.

롤러 모터에 이상이 있는듯 함

### 14 주차)

롤러와 뒷문 모터에 이상이 있어서 여상은 선생님께 MX-28AT 여분 2 개 받음. 토크가 달라서 공이 주워지는 것이 달라짐.

## 3.3 SolidWorks

### 1-5 주차)

수요일, 목요일 솔리드웍스 오피스 아워에 참여하면서 고무줄과 같은 재질의 힘 해석에 대해 조언을 구하였습니다. 다만 정확한 해석을 위해서는 공의 무게를 직접 재서 시뮬레이션 할 것을 조언해 주셨습니다.

금요일 11 시 반에 솔리드웍스 담당 조교님들과 발표 전 필수 사전 검사를 약 15 분간 받았습니다. 그동안 아이디어 회의를 통해 나왔던 아이디어들에 대해 조교님들께 피드백을

받았습니다. 차체 프레임 두가지 아이디어를 컨펌받았으며, 진동 방식에 대해서 고려한 부분은 좋지만, 집기 방식이 더 구체화될 필요가 있음을 조언해 주셨습니다.

토요일부터는 롤러 방식으로 새롭게 바꾼 아이디어에 대해 솔리드웍스 설계를 시작하였습니다.

#### **6 주차)**

지난주 메카넘휠과 기어, 다이نام셀, 차체의 어셈블리를 제작하였습니다. 기타 부품들을 쌓는 몸통과 공을 쏘는 부분에 대해서는 추가적으로 제작했습니다.

#### **7 주차)**

팀미팅에서 정한 dimension 을 바탕으로 롤러 부분과 공이 넘어와 저장소로 넘어가는 받침대 부분을 제작하였습니다. 여상은 선생님께 stl 파일을 전달하였습니다.

#### **8 주차)**

1 차적으로 제작한 받침대 부분을 접착하여 프로토타입을 제작하였습니다.

팀 내 의논을 바탕으로 수정된 받침대와 롤러부분을 다시 제작하여 stl 파일을 여상은 선생님께 전달하였습니다. 시험이 끝나는 대로 롤러, 받침대, 과학상자를 이용해 만든 storage 부분을 연결, 고정할 계획입니다.

#### **9 주차)**

슬라이딩 부분과 롤러 부분에 대한 3d 프린팅 결과물을 전달받았으며, 이를 과학상자로 만든 부품들과 결합하였습니다.

금요일(20 일)에 솔리드웍스 조교님들과 미팅을 가졌으며, 소프트웨어의 빠른 인테그레이션을 위해 하드웨어 부분을 빠르게 구축할 것이라 피드백을 전달받았습니다.

#### **10 주차)**

플라스틱 상자를 차체로 하여 1 차적으로 프로토타입을 제작하였습니다. 하지만 플라스틱 상자가 하중에 대한 변형이 생각보다 크다는 점, 납땀한 부분에 문제가 발생한 점 등으로 인해 설계와 납땀을 재진행 하였습니다.

기어 dimension 을 고려한 차체 설계를 진행하였고, 2 차적으로는 단단한 아크릴 판을 이용하여 밑판을 제작하였습니다. 하지만 현재 차체는 크기가 커진다는 단점이 있어서, 크기를 줄일 방안을 모색하고 있습니다

#### **11 주차)**

화요일에 기어 진동에 문제가 생겨 수요일에 재가공 및 재조립을 진행하였습니다.

롤러 모터와 롤러 축을 연결할 커플링을 추가적으로 3d 프린팅하여 롤러를 연결하였습니다.

#### **12 주차)**

목요일(10 일)에는 차체를 다시 분해하여 다음과 같은 부분을 보완하였습니다.

- 1) 기어 축에 슬립을 방지하기 위한 탭을 내어 슬립링을 끼우고 축의 이동으로 인한 뒤틀림을 보완하였습니다.
- 2) 바닥면의 네개의 프로파일 사이에도 추가로 축길이 만큼의 프로파일을 끼워 축에 외력이 작용하여 기어가 뒤틀리는 것을 보완하였습니다.

3) 기어에 의한 모터의 회전을 방지하기 위해 외부에 L자형 고정단을 추가로 부착하였습니다.

#### 13 주차)

뒷문 파트를 제작하고 MX-12W 모터에 결합하여 납땜에 연결, 구동 확인

Storage 에서 기존의 경사가 완만하여 공이 바구니로 잘 내려가지 않는다는 문제가 발생. 3d 프린터에서 큰 파트를 한번에 뽑아 발생하는 문제로, 좌우는 나누어 다시 제작.

롤러 부분 모터 커플링 다시 제작하여 부착함

#### 14 주차)

롤러 부분 길이 재조정

대각선 진행 문제에 대해 조교님들과 논의한 것을 바탕으로 차체 재조립

방열판 및 팬 모터 부착 & 열해석 진행

공이 튀어 나가는 것을 방지하기 위한 가림막 제작

### 3.4 LabView

#### 1-5 주차)

3/19 ~ 3/25 - HW1 을 통해 LabVIEW 및 MyRIO 등을 익혔으며 이를 바탕으로 HW2 를 진행함. 4 개의 LED 를 각각 1, 0.5, 0.25, 0.125 (s) 간격으로 반짝이는 코드, 순서대로 반짝이는 코드, MyRIO 의 3 축 가속도 측정 및 일정 가속도 초과시 LED 를 켜는 코딩을 하였음.

3/26~4/1 - 이후 HW3 를 통해 Dynamixel 에 메카닉휠을 장착한 후 PC 와 직접 연결하여 모터의 position 및 rotation speed 를 바꾸며 돌리는 것을 진행함. 교재를 바탕으로 간단한 TCP/IP 컨트롤 예제를 통해 server 및 client 간의 통신을 주고받는 과정과 TCP 함수들을 이해하였음. 또한 4 개의 바퀴를 제어하는 코딩도 해보았으며, xbox 를 아직 구현하지 못했지만 예제를 보며 그 코딩을 익히기 위해 노력하였음. 이를 바탕으로 PC 와 myRIO 의 TCP/IP 통신 및 myRIO 와 Dynamixel 의 serial 통신을 하기 위해 다양한 시도들을 해보았고 어려운 점은 담당 조교님께 도움을 요청하여 궁금증을 해결함.

#### 6 주차)

전주에 하지 못했던 PC와 myRIO 간의 TCP/IP 통신을 하기 위한 코드를 짤. 모터를 돌아가게 짚음에도 불구하고 모터가 돌아가지 않아 조교님으로부터 도움을 받음. 예제를 이용하는 것을 모르고 처음부터 client 와 server 를 짜 큰 곤란을 겪음. 다시 한 번 TCP/IP 코드를 유심히 분석하여 문제점을 발견하였고 그를 바탕으로 모터제어를 어떻게 해야 하는 건지 알게 됨. Xbox 에 짜여진 신호를 액추에이터에 주면 모터가 돌도록 코드를 짜서 실행하여 모터를 pc command 로부터 돌림. 이를 바탕으로 바퀴 네 개를 서로 전선으로 이은 후 바퀴 네 개를 동시에 돌리는데 성고하여 10 가지 모드에 대한 코드를 짤. pc 에서 신호를 주면 네 개의 바퀴가 총 10 가지 모드로 돌도록 코드를 완성함.

#### 7 주차)

저번 주에 pc 커맨드를 이용해 10 가지 모드의 모터를 제어하는 코드를 짤지만 몇 가지 문제점이 있던 점을 해결하려 했다. 첫 번째는 가끔 초기 시작을 해도 모터가 돌아가지

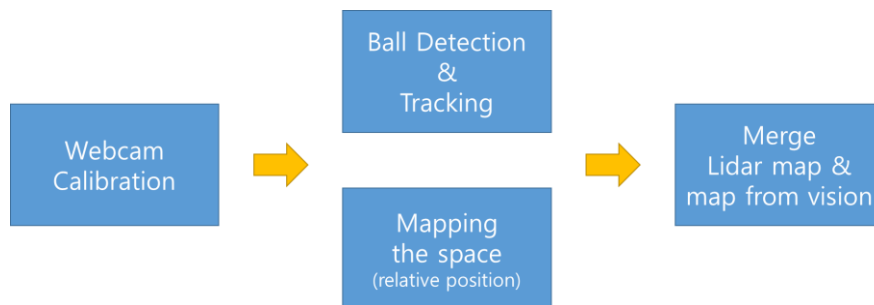
않으며, 모드 변경을 했을 때 반응속도가 매우 느리며, 모드가 명령 값에 비해 한 칸씩 밀린다는 점이다. 문제를 해결하지 못해 김범준 조교님께 도움을 요청해 함께 논의해 보았다. 우선 반응속도가 느린 점은 코드 자체가 양방향으로 짜여있어서 그렇다는 말을 들었다. 더 빠르게 하고 싶다면 단방향 코드를 짜야 한다고 하셨다. 그리고 초기 시작을 했을 때 모터가 작동하지 않는 것은 어떨 때는 작동하고 어떨 때는 작동하지 않아서 코드가 문제인지, 컴퓨터가 문제인지 뭐가 문제인지는 아직 모르겠다. 모드 변경을 해도 한 단계 전 모드가 작동하는 문제점은 while loop 가 문제 인 것으로 보이는데 정확하게 우리 코드의 어떤 점이 문제인지 정확하게 모르겠다. Local variable 이 한 타임 늦게 반응 하는 것 같기도 하다. 다음 주에 조교님을 한 번 더 불러 디버깅을 한 번 해봐야 할 것 같다. 이 작업만 마무리 된다면 ROS 팀과 함께 xbox 를 이용해 모터를 제어하는 단계로 넘어가야 할 것 같다. ROS 로부터 전송된 데이터를 읽는 코드를 짜야 하는데 이 부분은 공부가 많이 필요할 것 같다.

#### 8-14 주차)

코드 수정 및 모터 인식 문제 해결

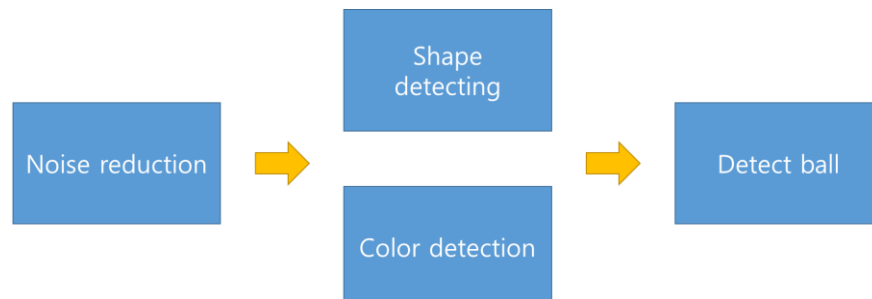
### 3.5 OpenCV

#### 1-5 주차)



**Figure 18 overview of OpenCV session**

Opencv session 의 mission 은 **Figure.1** 과 같은 순서로 시행되어야 할 것이다. 먼저 camera calibration 코드를 이용해 webcam 의 calibration 을 진행하였다.



**Figure 19 Ball detecting Process**

먼저 Ball detecting 과정이다. 첫째로 이미지에 존재하는 noise 를 제거해야한다. 이는 blur 효과나 contrast 를 높이는 방법을 이용하여 noise 자체를 제거하거나 물체의 경계

면을 정확하게 하는 것이다. 그 이후에는 공을 인식해야 하는데 이는 공 모양을 인식하는 방법과 공의 색을 인식하는 방법으로 나뉜다. 먼저 공의 색을 인식하는 방법은 Thresholding operation 을 이용하였고, 공의 모양을 인식하기 위해서는 Canny edge detector 를 사용하였다.

#### 6 주차)

Ball detection 예제 코드를 이용해 공을 인식하고, 그 위치를 detect 함. HSV 색상과 canny edge detector 를 이용.

#### 7 주차)

기존의 다양한 Ball tracking 방식과 mapping 방식을 고민 중에 있음.

ROS 에서 정보처리하는 방식을 공부하고 그에 맞게 mobile platform 의 위치, lid 와 시각 정보를 어떻게 합치고 처리하는지 스터디. 물체의 위치를 파악하는데, stereo camera 가 더 유리할 수 있어서 추가적인 카메라 사용을 고민 중에 있음.

#### 8 주차)

Logitech C920 카메라가 갖는 최대 시야를 확보하기 위해서 가로:세로 16:9 비율로 영상을 받는다. 1920x1080 로 설정하였다.

기존에 주어진 정보를 토대로 공이 위치할 군집의 위치를 지정한다. 인식하는 정확도를 높이기 위해서 카메라의 위치 정보를 받아서 카메라와 군집과의 최소 거리와 최대 거리를 계산한다. 그 후 주어진 공의 크기(지름:7.5cm)와 카메라의 시야 각(가로 시야 각:68°)을 이용해서 공이 지름 길이가 차지할 픽셀 최대와 최소 개수를 계산한다. 이를 바탕으로 공을 크기 범주를 지정해서 공의 인식 정확도를 높인다.

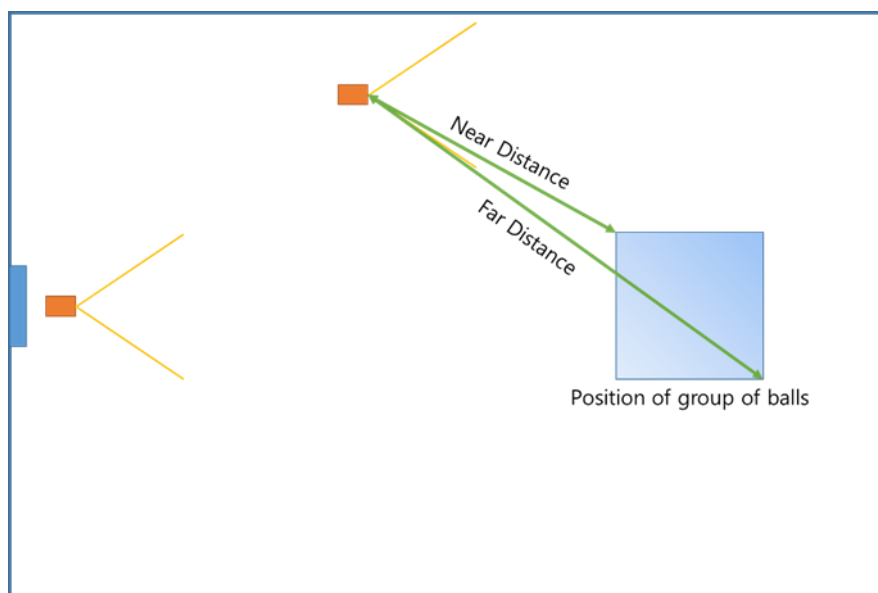


Figure 20 guess ball size

### 9 주차)

Ros 의 ball detection node 에 기존에 작성했던 코드를 합쳤다. Ball detection node 에서 data integration node 에 빨간 공들과 파란 공들의 위치 정보를 넘겨주었다. 그 결과 기존에는 확인하지 못했던 비슷한 위치에 공이 두개 존재하기도 하는 문제점을 발견하였으나 detection 에 영향을 미치는 요소가 아니었으므로 추후에 수정할지 여부를 정하기로 하였다. Ball detection node 로는 카메라의 위치 정보를 받을 예정이지만 차체의 정확한 위치(카메라와 고정된 상대 위치를 갓음)를 알 수 없을 가능성이 존재한다. 따라서 차체의 정확한 위치를 알 수 있도록 센서를 사용하거나 myrio 의 accelerometer 를 사용하는 방법을 고안하기로 했다.

### 10 주차)

웹캠의 FOV 를 토대로 캠의 최적 위치와 시야를 계산하였다. 그 결과는 대략 32cm 높이에 23.5 도지면쪽으로 틸트하는 것이다.

### 11 주차)

초록색 공을 detect 하는 코드를 추가하고, 기존의 코드 에러를 수정하였다. 초록공의 detect 가 어려울 경우 바구니를 detect 하는 옵션도 추가할 필요성이 있다. 실험적으로 카메라의 tilt angle 을 결정함.(23.5 도)

### 12 주차)

진동 분석을 위해 영상을 촬영하여 데이터를 수집하는 코드 작성 및 작동

### 13 주차)

진동 분석 중. 실제로 진동이 구동에 큰 영향을 미치지 않는 않지만 분석적으로 그 이유를 타당히 밝히려고 노력.

초록색 공을 두개로 보는 현상 코드 수정

## 3.6 ROS

### 1-5 주차)

월요일 ROS session 에서 기본적인 우분투 환경에 대한 강의를 받아서 매뉴얼 pdf 에 적혀있는 명령어, 코드들을 단순 복사 붙여 넣기 하는 것이 아니라 해당 명령어, 코드들이 어떻게 작동하는지에 대한 이해를 하게 되었다. 따라서 적절한 수정을 통하여 내 플랫폼에 맞게 적용하고 오류가 났을 때 무엇이 문제인지 조교님께 묻지 않고도 스스로 해결 할 수 있게 되었다.

다음으로 과제를 해결하였는데, 문제를 해결 하면서 topic, service 가 어떻게 작동하는지 실제로 볼 수 있었으며, 여러 개의 node 의 묶음을 편리하게 한번에 실행시켜주는 roslaunch 를 어떻게 사용하는지 배울 수 있었다. 또한 추가 과제로는 하나의 노드가 topic 에서 데이터를 송신하는 publisher 와 데이터를 수신하는 subscriber 로서의 역할을 동시에 하도록 하는 문제가 나왔는데, 단순히 예제 코드의 publisher, subscriber 의 내용을 합치고,



메시지이름을 바꾸어 주는 것으로는 송신은 정상적으로 진행되지만, 수신은 `rqt_graph` 를 통해 node 들 간의 연결관계를 보아서는 연결이 되어 있지만, 실제로 터미널 창에서는 표현이 안되는 문제가 있었고, 이를 해결하기 위해 ROS 관련 사이트들을 검색해본 결과, 문제는 `spin()` 함수를 통해 `callback` 을 하는 방식으로 예제 코드가 구현이 되어있었는데, 이 내용이 `while` 문과 겹쳐, 계속 반복이 되는 문제였다. 따라서 `spin()` 대신 `spinOnce()` 함수를 활용하여 해결 할 수 있었다.

금요일 session 에서는 조교님들이 구현해 놓으신 기본 소스코드를 받고, 그 코드들이 어떻게 작동하는지를 배우는 과정을 거쳤다. 그 과정에서 `webcam`, `lidar` 가 어떻게 사용되는지 실제로 볼 수 있었고, `xbox controller` 를 통해 조종 데이터를 수신하고, `tcp/ip` 서버를 구성하여 송신하는 방법에 대해 배울 수 있었다.

#### 6 주차)

LIDAR 와 WEBCAM 을 사용하는 코드들을 읽어보면서 어떻게 작동하는지 알아보고 해당 코드를 다른 파트와 어떻게 통합시켜 발전시켜야 할 지에 대해서 생각 해보는 시간을 가졌다.

#### 7 주차)

별다른 진행 없음

#### 8 주차)

Ros 파트에서 `transfer function` 에 대한 강의가 있었는데 시간상 참여하지 못했기 때문에 시험이 끝난후 해당 강의 자료를 보며 복습 하는 과정을 거쳤고, `transfer function` 을 어떻게 사용할 지에 대한 정보를 알 수 있게 되었다.

다만, 현재 우리의 시스템에서 `rplidar` 가 하는 역할이 매우 적고, 실질적으로 맵핑이 쉽지 않은 만큼 `transfer function` 이 과연 정말로 필요할지, 실제로는 어떻게 사용할지에 대한 생각이 필요할 것으로 예상된다. 만약 `rplidar` 가 필요 없다면, `transfer function` 은 사용할 필요가 없거나, 사용하더라도 웹캠과 플랫폼의 중심 위치를 보정해 주는 역할로 바뀌게 될것이기 때문에 우선 빠르게 웹캠을 이용한 `ball detection` 을 하고, 이 데이터를 활용하여 공들을 주워 오는 것을 해 보아야 전체적인 시스템 구현의 그림이 바뀌기 때문에 최우선적으로 해야 할 목표라고 할 수 있다.

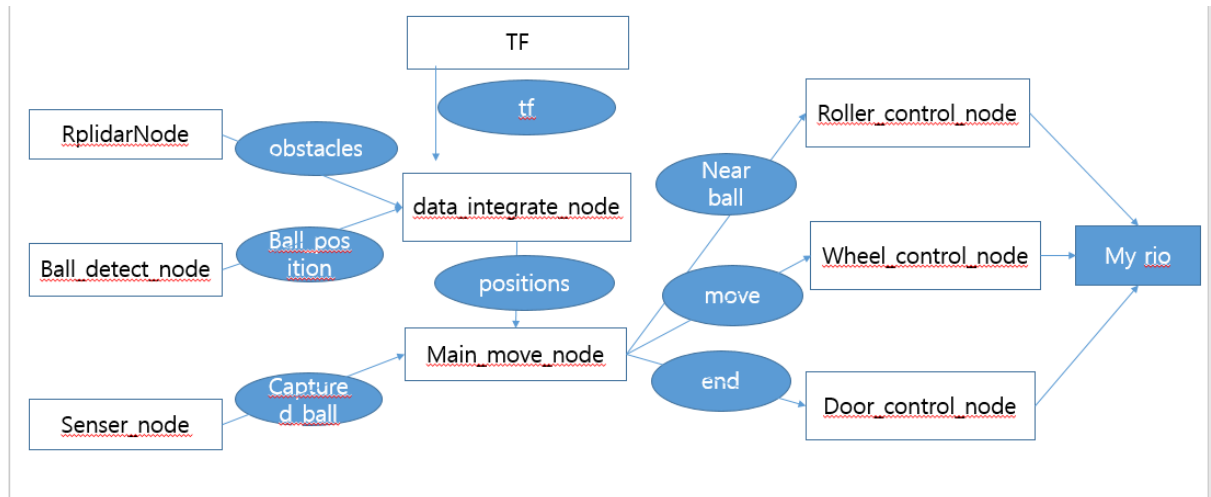
따라서, 주어진 `ball detection node` 가 실제로 어떻게 작동하는지 확인 하고, 코드가 어떻게 쓰여 있는지를 공부하여 `opencv` 와의 통합을 쉽게 하도록 하였다.

#### 9 주차)

ros 파트에서는 첫째로 `tcp ip` 통신을 `my rio` 와 하는 것을 랩뷰파트와 함께 하여, 모터를 실제로 `xbox` 컨트롤러를 통해 조종할 수 있었다.

또한, `opencv` 파트의 `ball detection` 에 관련된 코드를 `ball detection node` 에 통합시키고 `data show node` 에 결과를 보이도록 하는 과정을 거쳤다. `Opencv` 에서 구현한 코드와 주어진 `ball detection node` 의 코드가 상당히 달라서 그 차이를 이해하고 통합하는데 많은 노력을 해야 했다. 이 과정에서 `webcam node` 를 사용하는 것 대신 `ball detection node` 에 통합시켰고, 메시지에 푸른공, 붉은공에 대한 데이터를 모두 보내주도록 수정했다.

결과적으로 ball detection node 를 열면 웹캠으로부터 이미지 데이터를 받아 data show 와 data integration 쪽으로 인식된 공들의 데이터를 publish 해주는 역할을 하도록 할 수 있었다. 또, 전체적 process 를 노드/토픽 단위로 다음과 같은 그림으로 만들어 보는 과정을 거치며 실제로 각각의 노드들이 어떤 일을 할 것이며 해당 노드들을 어떻게 구현해야 할지를 더 구체적으로 알아 볼 수 있었다.



#### 10 주차)

해당 주차에는 control node 를 생성하여, 간단한 수준의 자율 주행을 하는것에 중심을 뒀다. 주어진 data integration node 를 기본으로 하여, 웹캠에서 온 공의 좌표값에 따라 플랫폼이 취해야 할 행동을 결정하고, my rio 에 xbox control node 에서 수행했던 것 처럼 값을 넣어주는 역할을 하도록 했다.

구체적으로 알고리즘을 설명하자면, 파란 공을 발견하지 못했을 땐, 파란 공을 찾을 때 까지 제자리에서 회전하고, 파란 공을 발견하면, 빨간 공을 피해야 하는지 확인을 하고 피해야 한다면, 빨간공이 있는 반대쪽 대각선으로 이동한다. 빨간 공을 피하지 않아도 된다면, 파란 공이 플랫폼의 정면에 있도록 회전을 한 후, 파란 공 쪽으로 직진한다. 마지막으로, 파란공을 향해 직진 하다가 가까워지면 일정시간동안 계속 직진하면서, 롤러의 모터를 돌려 공을 잡는다.

실제 작동시킨 결과, 전반적으로 생각했던 것 보다도 잘 작동했지만 몇가지 문제점이 있었다. 첫째로 너무 빠르게 회전해서 공이 정면에 오기 힘든 문제가 있었는데, 간단한 P 컨트롤을 사용하고 기존에는 정면에 오는 기준을 Y 좌표를 통해 계산했는데, 이렇게 계산하면, 너무 먼거리에 있는 공은 약간의 회전만으로 많은 Y 값이 변하므로, 각도 조건을 추가하여 둘중 하나만 만족하면 되도록 하였다. 다만 웹캠으로 공을 인식하고 가는 방식의 특성상 공이 너무 가까워져 시야시 벗어났을 때 생기는 문제는 해결하지 못했다.

#### 11 주차)

이번 주차에는 롤러를 통해 공을 잡고 마지막으로 공을 바구니에 놓는 과정을 발전시키기 위한 노력을 하였다. 일단 센서를 사용하여 잡힌 공의 개수를 파악하려고 했는데, 공의 무게가 너무 가볍기 때문에 압력센서를 사용하여 인식하기에는 힘이 부족하다는 결론을

내렸고, lego 의 색 인식 센서가 공을 인식하기 좋고, ros 에서의 node 를 지원하는 장점을 가지기 때문에 잠정적 후보로 삼기로 했고 다른 웹캠을 다는 방식도 고려해보기로 했다.

Rplidar 사용에 있어서는 일단 SLAM 을 전반적으로 구현하기 힘들고, 공 이외의 다른 장애물이 없는 상황에서 특별히 중요한 역할을 하긴 힘들 것으로 보인다. 다만 벽을 인식하여, 벽을 피하는 용도, 마지막에 공을 모두 잡고 돌아온 후 공을 바구니에 넣을 때 공이 뒤로 나오게 되므로 공을 놓기 위해 180 도 회전할 때 초록공이 보이지 않기 때문에 벽을 통해 기준 축을 정하는 용도, 공을 잡는 과정에 문제가 있을 때, 다시 대략적으로 맵 가운데로 돌아와 다시 행동을 수행하도록 하는 용도 등 부가적인 요소로는 사용하면 좋을 것 같다는 생각을 했다.

롤러의 경우엔 기존 디자인 그대로 갈 경우, 롤러 모터가 너무 빠르면 공을 쳐내고, 너무 느리면 힘이 부족해 경사로를 오르지 못해 플랫폼 내부에 들어가지 않는 문제점이 있었다. 이 문제를 해결하기 위해서 일단 경사로를 다양한 모양으로 실험해본 결과 현재의 곡선형 보다는 직선을 사용할 때 더 잘 올라간다는 판단을 내릴 수 있었고, 이렇게 하드웨어적인 부분을 수정 한 후에 롤러 모터의 속도를 조절하여 보도록 하였다.

## 12 주차)

전반적인 설계가 끝나고 rplidar 사용을 위해 rplidar 를 활용하는 방안에 대하여 생각해 보았다. Hector mapping , gmapping 등의 package 를 활용하여 rough 한 mapping 을 활용할 순 있지만, 가속도계나 encoder 가 없는 상황에서 이 mapping 이 큰 의미를 갖기 힘들다는 생각을 했고, rplidar 의 raw data 인 각도와 거리를 활용하는 방식에 대해서 생각을 해 보았고, 바구니나 공을 인식하여 결과를 얻거나, 벽을 인식하여 align 하는 방안이 있었으나, 결국 웹캠을 활용하여 조종하는것에 비해 큰 의미를 갖지 못한다고 판단이 되었으며, 결국 rplidar 를 설계에서 제외하는 방안으로 가기로 했다.

그 외의 ros 파트에서는 일단 데모가 진행되는 과정을 지켜본 결과 실제로 공 3 개를 안정적으로 잡기 위해 여러가지 값들 예를 들어 빨강색 공을 장애물로 인식하는 범위나 파란 공 앞에서 롤러를 돌리기 시작하는 거리 등 구체적인 값들을 튜닝하여 안정적으로 공을 잡을 수 있도록 했고, 초록색 공을 인식하여 picking 이 끝나고 난 뒤 다시 바구니로 돌아와 공을 놓는 방법에 대해 생각을 해 보았는데, 일단 2 개의 초록색 공 사이를 목표로 두고 이동한 후 두개의 공의 Z 좌표가 같아지도록 회전을 하여 바구니와 수평이 되도록 align 을 한 후, 180 도 회전이 되는 시간을 측정하여 그 시간동안 회전하도록 명령을 주어 완벽하게 바구니 앞에 멈추어 서는 것을 목표로 코딩을 하고 있다.

## 13 주차)

이번주는 데모 전 주인 만큼 최대한 모든 할 일들을 미리 끝마치려 했으나 시스템적인 문제가 너무 많아 어려움이 있었다. 일단 모터가 계속 인식이 안되어서 롤러 부분이나 뒷문 쪽이 제대로 돌아가지 않는 문제가 있었고, 초록 공을 활용하여 align 을 하려고 원래 방식보다 이상적인 위치로 이동하게끔 두 공의 수직 이등분선 위로 플랫폼을 이동 시키고, 거기서 두 공 사이를 보고 이동하도록 하는 방식을 고안하여 보았다.

그러나 문제는 초록색 공의 인식이 튀어서 공 하나가 같은 위치에 2 번 찍히는 문제점이 있었는데 그 문제가 화면상에서는 잘 보이지 않았고, 그래서 그 문제를 잘 모른채 코드 문제로 알고 수정하느라 시간을 낭비하였다. 이 문제를 찾아낸 후 open cv 파트에서 같은 위치에 있는 공의 인식을 없애는 코드를 넣었고 ros 파트에서 혹시 여러개의 공이 인식이 되어도 상관 없게끔 2 개의 공을 사용할 때, 원래는 0 번 1 번공을 활용하여 놓던 것을 가장 왼쪽 오른쪽의 공을 활용하도록 코드를 바꾸어서 문제를 해결하였다.

대각선 움직임 자체가 전체 움직임 중 일부분에 불과 하다는 것을 고려하였을 때, 완벽하진 않지만 충분히 괜찮다고 볼 수 있고, 최대한 코드를 짜는 과정에 있어서 대각선 이동을 배제하고 대각선 이동을 한다면, 오래 가만히 있으면 탈출하는 (전후좌우로 움직이는) 방식을 사용한다면 문제를 해결 할 수 있다는 결론에 이르렀다.

## 4. Additional Materials

### -OpenCV code

```
#include <opencv2/imgproc.hpp> //include opencv library
#include <opencv2/highgui.hpp> //include opencv library
#include <iostream> //include standard library input/output stream
#include <string> //include library to use string type
#include <stdlib.h> //include standard library
#include <math.h> //include math library which supports some mathematical functions ex)sqrt ...
using namespace std; //makes to omit 'std::' when using standard function
using namespace cv; //makes to omit 'cv::' when using opencv function
// Declaration of trackbar functions to set HSV colorspace's parameters
//for red detection
void on_low_h_thresh_trackbar_red(int, void *); //declare trackbar for low h
void on_high_h_thresh_trackbar_red(int, void *); //declare trackbar for high h
void on_low_h2_thresh_trackbar_red(int, void *); //declare trackbar for low h2 color: red needs two range of h because red has two region on hsv plane
void on_high_h2_thresh_trackbar_red(int, void *); //declare trackbar for high h2
void on_low_s_thresh_trackbar_red(int, void *); //declare trackbar for low s
void on_high_s_thresh_trackbar_red(int, void *); //declare trackbar for high s
void on_low_v_thresh_trackbar_red(int, void *); //declare trackbar for low v
void on_high_v_thresh_trackbar_red(int, void *); //declare trackbar for high v
//initiating values to set initial settings
int low_h2_r=160, high_h2_r=180; //initiate high_h2_r, low_h2_r value
int low_h_r=0, low_s_r=100, low_v_r=136; //initiate low_h_r,low_s_r, low_v_r value
int high_h_r=10, high_s_r=255, high_v_r=255; //initiate high_h_r,high_s_r,high_v_r value
//for blue detection
void on_low_h_thresh_trackbar_blue(int, void *); //declare trackbar for low h
void on_high_h_thresh_trackbar_blue(int, void *); //declare trackbar for high h
void on_low_s_thresh_trackbar_blue(int, void *); //declare trackbar for low s
void on_high_s_thresh_trackbar_blue(int, void *); //declare trackbar for high s
void on_low_v_thresh_trackbar_blue(int, void *); //declare trackbar for low v
void on_high_v_thresh_trackbar_blue(int, void *); //declare trackbar for high v
//initiating values to set initial settings
int low_h_b=110, low_s_b=100, low_v_b=100; //initiate low_h_b,low_s_b, low_v_b value
int high_h_b=130, high_s_b=255, high_v_b=255; //initiate high_h_b,high_s_b,high_v_b value
// Declaration of functions that changes data types
string intToString(int n); //declare function which change the type of value: int to string
string floatToString(float f); //declare function which change the type of value: float to string
// Declaration of functions that changes int data to String
void morphOps(Mat &thresh);
// Declaration of functions that calculates the ball position from pixel position
vector<float> pixel2point(Point center, int radius);
// Declaration of trackbars function that set canny edge's parameters
void on_canny_edge_trackbar_red(int, void *); //declare trackbar for canny edge detection of red
//initiating values to set initial settings
int lowThreshold_r = 100; //declare lowThreshold_r as integer and initialise to 100
int ratio_r = 3; //declare ratio_r as integer and initialise to 3
int kernel_size_r = 3; //declare kernel_size_r as integer and initialise to 3
void on_canny_edge_trackbar_blue(int, void *); //declare trackbar for canny edge detection of blue
//initiating values to set initial settings
int lowThreshold_b = 100; //declare lowThreshold_b as integer and initialise to 100
int ratio_b = 3; //declare ratio_b as integer and initialise to 3
int kernel_size_b = 3; //declare kernel_size_b as integer and initialise to 3
// Initialization of variable for dimension of the target
float fball_radius = 0.075 ; //set the reference length: in this case ball is reference
// Initialization of variable for camera calibration parameters >>>change to our own value!!!!
Mat distCoeffs;
float intrinsic_data[9] = {648.494831, 0, 330.912390, 0, 649.675640,246.894210, 0, 0, 1}; //set intrinsic parameter of camera calibration which is K
float distortion_data[5] = {0.042623, -0.140928, -0.001380, -0.005298,0}; //set distortion vector D
// Initialization of variable for text drawing
double fontScale = 2; //set fontscale as 2
int thickness = 3; //set thickness as 3
String text; //declare test as String type
```

```

int iMin_tracking_ball_size = 20; //declare Minimum tracking size of the ball as 20 in int type
int main()
{
    Mat frame, bgr_frame, hsv_frame, hsv_frame_red, hsv_frame_red1, hsv_frame_red2, hsv_frame_blue,
        hsv_frame_red_blur, hsv_frame_blue_blur, hsv_frame_red_canny, hsv_frame_blue_canny, result; //declare matrix for frames and result
    Mat calibrated_frame; //declare matrix for calibrated frame
    Mat intrinsic = Mat(3,3, CV_32FC1); //set intrinsic matrix as 3x3 matrix with 32bit float 1 channel
    Mat distCoeffs; //declare matrix distCoeffs
    intrinsic = Mat(3, 3, CV_32F, intrinsic_data); //put intrinsic_data to intrinsic matrix
    distCoeffs = Mat(1, 5, CV_32F, distortion_data); //put distortion_data to distCoeffs matrix
    vector<Vec4i> hierarchy_r; //declare hierachy_r as 4 element vector(line)
    vector<Vec4i> hierarchy_b; //declare hierachy_b as 4 element vector(line)
    vector<vector<Point> > contours_r; //declare contours_r as point vector
    vector<vector<Point> > contours_b; //declare contours_b as point vector
    VideoCapture cap(1); //get image from video, May be changed to 0 for NUC
    namedWindow("Video Capture", WINDOW_NORMAL); //create a window and name it as "Video Capture"
    namedWindow("Object Detection_HSV_Red", WINDOW_NORMAL); //create a window and name it as "Object Detection_HSV_Red"
    namedWindow("Object Detection_HSV_Blue", WINDOW_NORMAL); //create a window and name it as "Object Detection_HSV_Blue"
    namedWindow("Canny Edge for Red Ball", WINDOW_NORMAL); //create a window and name it as "Canny Edge for Red Ball"
    namedWindow("Canny Edge for Blue Ball", WINDOW_NORMAL); //create a window and name it as "Canny Edge for Blue Ball"
    namedWindow("Result", WINDOW_NORMAL); //create a window and name it as "Result"
    moveWindow("Video Capture", 50, 0); //Moves "Video Capture" window to the specified position (50,0)
    moveWindow("Object Detection_HSV_Red", 50, 370); //Moves "Object Detection_HSV_Red" window to the specified position (50,370)
    moveWindow("Object Detection_HSV_Blue", 470, 370); //Moves "Object Detection_HSV_Blue" window to the specified position (470,370)
    moveWindow("Canny Edge for Red Ball", 50, 730); //Moves "Canny Edge for Red Ball" window to the specified position (50,730)
    moveWindow("Canny Edge for Blue Ball", 470, 730); //Moves "Canny Edge for Blue Ball" window to the specified position (470,730)
    moveWindow("Result", 470, 0); //Moves "Result" window to the specified position (470,0)
    // Trackbars to set thresholds for HSV values : Red ball
    createTrackbar("Low H", "Object Detection_HSV_Red", &low_h_r, 180, on_low_h_thresh_trackbar_red); //create trackbar and name it "Low H", on window "Object
Detection_HSV_Red", position of slider is low_h_r, set maximum position of slider as 180, get vlaues from function on_low_h_thresh_trackbar_red
    createTrackbar("High H", "Object Detection_HSV_Red", &high_h_r, 180, on_high_h_thresh_trackbar_red); //create trackbar and name it "High H", on window
"Object Detection_HSV_Red", position of slider is high_h_r, set maximum position of slider as 180, get vlaues from function on_high_h_thresh_trackbar_red
    createTrackbar("Low H2", "Object Detection_HSV_Red", &low_h2_r, 180, on_low_h2_thresh_trackbar_red); //create trackbar and name it "Low H2", on window
"Object Detection_HSV_Red", position of slider is low_h2_r, set maximum position of slider as 180, get vlaues from function on_low_h2_thresh_trackbar_red
    createTrackbar("High H2", "Object Detection_HSV_Red", &high_h2_r, 180, on_high_h2_thresh_trackbar_red); //create trackbar and name it "High H2", on
window "Object Detection_HSV_Red", position of slider is high_h2_r, set maximum position of slider as 180, get vlaues from function on_high_h2_thresh_trackbar_red
    createTrackbar("Low S", "Object Detection_HSV_Red", &low_s_r, 255, on_low_s_thresh_trackbar_red); //create trackbar and name it "Low S", on window "Object
Detection_HSV_Red", position of slider is low_s_r, set maximum position of slider as 255, get vlaues from function on_low_s_thresh_trackbar_red
    createTrackbar("High S", "Object Detection_HSV_Red", &high_s_r, 255, on_high_s_thresh_trackbar_red); //create trackbar and name it "High S", on window
"Object Detection_HSV_Red", position of slider is high_s_r, set maximum position of slider as 255, get vlaues from function on_high_s_thresh_trackbar_red
    createTrackbar("Low V", "Object Detection_HSV_Red", &low_v_r, 255, on_low_v_thresh_trackbar_red); //create trackbar and name it "Low V", on window "Object
Detection_HSV_Red", position of slider is low_v_r, set maximum position of slider as 255, get vlaues from function on_low_v_thresh_trackbar_red
    createTrackbar("High V", "Object Detection_HSV_Red", &high_v_r, 255, on_high_v_thresh_trackbar_red); //create trackbar and name it "High V", on window
"Object Detection_HSV_Red", position of slider is high_v_r, set maximum position of slider as 255, get vlaues from function on_high_v_thresh_trackbar_red
    // Trackbars to set thresholds for HSV values : Blue ball
    createTrackbar("Low H", "Object Detection_HSV_Blue", &low_h_b, 180, on_low_h_thresh_trackbar_blue); //create trackbar and name it "Low H", on window
"Object Detection_HSV_Blue", position of slider is low_h_b, set maximum position of slider as 180, get vlaues from function on_low_h_thresh_trackbar_blue
    createTrackbar("High H", "Object Detection_HSV_Blue", &high_h_b, 180, on_high_h_thresh_trackbar_blue); //create trackbar and name it "High H", on window
"Object Detection_HSV_Blue", position of slider is high_h_b, set maximum position of slider as 180, get vlaues from function on_high_h_thresh_trackbar_blue
    createTrackbar("Low S", "Object Detection_HSV_Blue", &low_s_b, 255, on_low_s_thresh_trackbar_blue); //create trackbar and name it "Low S", on window
"Object Detection_HSV_Blue", position of slider is low_s_b, set maximum position of slider as 255, get vlaues from function on_low_s_thresh_trackbar_blue
    createTrackbar("High S", "Object Detection_HSV_Blue", &high_s_b, 255, on_high_s_thresh_trackbar_blue); //create trackbar and name it "High S", on window
"Object Detection_HSV_Blue", position of slider is high_s_b, set maximum position of slider as 255, get vlaues from function on_high_s_thresh_trackbar_blue
    createTrackbar("Low V", "Object Detection_HSV_Blue", &low_v_b, 255, on_low_v_thresh_trackbar_blue); //create trackbar and name it "Low V", on window
"Object Detection_HSV_Blue", position of slider is low_v_b, set maximum position of slider as 255, get vlaues from function on_low_v_thresh_trackbar_blue
    createTrackbar("High V", "Object Detection_HSV_Blue", &high_v_b, 255, on_high_v_thresh_trackbar_blue); //create trackbar and name it "High V", on window
"Object Detection_HSV_Blue", position of slider is high_v_b, set maximum position of slider as 255, get vlaues from function on_high_v_thresh_trackbar_blue
    // Trackbar to set parameter for Canny Edge
    createTrackbar("Min Threshold:", "Canny Edge for Red Ball", &lowThreshold_r, 100, on_canny_edge_trackbar_red); //create trackbar and name it "Min Treshold:",
on window "Canny Edge for Red Ball", position of slider is lowThreshold_r, set maximum position of slider as 100, get vlaues from function
on_canny_edge_trackbar_red
    createTrackbar("Min Threshold:", "Canny Edge for Blue Ball", &lowThreshold_b, 100, on_canny_edge_trackbar_blue); //create trackbar and name it "Min
Treshold:", on window "Canny Edge for Blue Ball", position of slider is lowThreshold_b, set maximum position of slider as 100, get vlaues from function
on_canny_edge_trackbar_blue
    while((char)waitKey(1))!= 'q'){ //run until 'q' is pressed

```

```

cap>>frame; //get new frame from camera
if(frame.empty()) //if there is no data read from camera(frame is empty)
    break; //break loop
undistort(frame, calibrated_frame, intrinsic, distCoeffs); //Transforms an image to compensate for lens distortion~ input: frame, output: calibrated_frame,
by using intrinsic
result = calibrated_frame.clone(); //deep copy calibrated_frame to result
medianBlur(calibrated_frame, calibrated_frame, 3); //input image:calibrated_frame, output to calibrated_frame, with aperture linear size of 3(should be odd
num)
cvtColor(calibrated_frame, hsv_frame, cv::COLOR_BGR2HSV); //input image:calibrated_frame, output to hsv_frame, convert color space from rgb to hsv
// Detect the object based on RGB and HSV Range Values
inRange(hsv_frame,Scalar(low_h_r,low_s_r,low_v_r),Scalar(high_h_r,high_s_r,high_v_r),hsv_frame_red1); //if each pixel from input(hsv_frame) satisfies the
condition lowerbound & upperbound, output is hsv_frame_red1
inRange(hsv_frame,Scalar(low_h2_r,low_s_r,low_v_r),Scalar(high_h2_r,high_s_r,high_v_r),hsv_frame_red2); //if each pixel from input(hsv_frame) satisfies the
condition lowerbound & upperbound, output is hsv_frame_red2
inRange(hsv_frame,Scalar(low_h_b,low_s_b,low_v_b),Scalar(high_h_b,high_s_b,high_v_b),hsv_frame_blue); //if each pixel from input(hsv_frame) satisfies
the condition lowerbound & upperbound, output is hsv_frame_blue
addWeighted(hsv_frame_red1, 1.0, hsv_frame_red2, 1.0, 0.0,hsv_frame_red); //merge two frames(hsv_frame_red1, hsv_frame_red2) ratio of 1:1 add scalar
0, output to hsv_frame_red
morphOps(hsv_frame_red); //apply function morphOps to hsv_frame_red
morphOps(hsv_frame_blue); //apply function morphOps to hsv_frame_blue
GaussianBlur(hsv_frame_red, hsv_frame_red_blur, cv::Size(9, 9),2, 2); //gaussian blur; input: hsv_frame_red, output: hsv_frame_red_blur, gaussian kernel
size 9x9, gaussian kernel standardeviation of x:2, y:2
GaussianBlur(hsv_frame_blue, hsv_frame_blue_blur, cv::Size(9,9), 2, 2); //gaussian blur; input: hsv_blue_red, output: hsv_frame_blue_blur, gaussian kernel
size 9x9, gaussian kernel standardeviation of x:2, y:2
Canny(hsv_frame_red_blur, hsv_frame_red_canny, lowThreshold_r,lowThreshold_r*ratio_r, kernel_size_r); //canny edge detection; input:
hsv_frame_red_blur, output: hsv_frame_red_canny, first threshold for the hysteresis procedure as lowThreshold_r, second threshold for the hysteresis procedure as
ratio_r*lowThreshold_r, with kernel size kernel_size_r
Canny(hsv_frame_blue_blur, hsv_frame_blue_canny, lowThreshold_b,lowThreshold_b*ratio_b, kernel_size_b); //canny edge detection; input:
hsv_frame_blue_blur, output: hsv_frame_blue_canny, first threshold for the hysteresis procedure as lowThreshold_b, second threshold for the hysteresis
procedure as ratio_b*lowThreshold_b, with kernel size kernel_size_b
findContours(hsv_frame_red_canny, contours_r, hierarchy_r,RETR_CCOMP, CHAIN_APPROX_SIMPLE, Point(0, 0)); //find contour from
hsv_frame_red_canny to contours_r, optional output vector(containing image topology) hierarchy_r, contour retrieval mode: RETR_CCOMP, contour approximation
method: CHAIN_APPROX_SIMPLE, shift point (0,0)(don't shift the point)
findContours(hsv_frame_blue_canny, contours_b, hierarchy_b,RETR_CCOMP, CHAIN_APPROX_SIMPLE, Point(0, 0)); //find contour from
hsv_frame_blue_canny to contours_b, optional output vector(containing image topology) hierarchy_b, contour retrieval mode: RETR_CCOMP, contour
approximation method: CHAIN_APPROX_SIMPLE, shift point (0,0)(don't shift the point)
vector<vector<Point> > contours_r_poly( contours_r.size() ); //set contours_r_poly as a vector of points size of contours_r
vector<vector<Point> > contours_b_poly( contours_b.size() ); //set contours_b_poly as a vector of points size of contours_b
vector<Point2f>center_r( contours_r.size() ); //set center_r as point2f vector size of contours_r
vector<Point2f>center_b( contours_b.size() ); //set center_b as point2f vector size of contours_b
vector<float>radius_r( contours_r.size() ); //set radius_r as float type vector size of contours_r
vector<float>radius_b( contours_b.size() ); //set radius_b as float type vector size of contours_b
for( size_t i = 0; i < contours_r.size(); i++ ){ //run the loop while size_t type i from 0 to size of contours_r-1 size by increasing i 1
    approxPolyDP( contours_r[i], contours_r_poly[i], 3, true ); //approximate contours_r[i] to polygon and put output in contours_r_poly[i] with
approximation accuracy 3
    minEnclosingCircle( contours_r_poly[i], center_r[i], radius_r[i] ); //get position of center and radius from minimum enclosing circle from contours_r_poly[i]
}
for( size_t i = 0; i < contours_b.size(); i++ ){ //run the loop while size_t type i from 0 to size of contours_b-1 size by increasing i 1
    approxPolyDP( contours_b[i], contours_b_poly[i], 3, true ); //approximate contours_b[i] to polygon and put output in contours_b_poly[i] with
approximation accuracy 3
    minEnclosingCircle( contours_b_poly[i], center_b[i], radius_b[i] ); //get position of center and radius from minimum enclosing circle from
contours_b_poly[i]
}
for( size_t i = 0; i < contours_r.size(); i++ ){ //run the loop while size_t type i from 0 to size of contours_r-1 size by increasing i 1
    if( radius_r[i] > iMin_tracking_ball_size){ //if radius_r[i] is larger than iMin_tracking_ball_size
        Scalar color = Scalar( 0, 0, 255); //set scalar color as 255 red
        drawContours( hsv_frame_red_canny, contours_r_poly, (int)i, color, 1, 8, vector<Vec4i>(), 0, Point() ); //draw contour
output&inputhsv_frame_red_canny with contours_r_poly, contour indexi, color:color, thickness:!, linetype:8, hierarchy ,offset:0
        vector<float> ball_position_r; //declare float vector named ball_position_r
        ball_position_r = pixel2point(center_r[i], radius_r[i]); //put position info to ball_position_r processed by function; pixel2point
        float isx = ball_position_r[0]; //declare float isx and put value; ball_position_r[0]
        float isy = ball_position_r[1]; //declare float isy and put value; ball_position_r[1]
        float isz = ball_position_r[2]; //declare float isz and put value; ball_position_r[2]
        string sx = floatToString(isx); //declare string sx and put value isx processed(change data type from float to string) from floatToSting
        string sy = floatToString(isy); //declare string sy and put value isy processed(change data type from float to string) from floatToSting
    }
}

```



```

        string sz = floatToString(isz); //declare string sz and put value isz processed(change data type from float to string) from floatToString
        text = "Red ball:" + sx + "," + sy + "," + sz; //put message(Red ball: sx,sy,sz) to text
        putText(result, text, center_r[i], 2, 1, Scalar(0, 255, 0), 2); //indicate text to result matrix, content: text, position: center_r[i], font type: 2, font scale: 1,
color: 255 green, line thickness: 2
        circle(result, center_r[i], (int)radius_r[i], color, 2, 8, 0); //indicate circle to result, position: center_r[i], radius: radius_r[i], font type: 2, font scale: 1, color:
255 green, line thickness: 2
    }
}
for( size_t i = 0; i < contours_b.size(); i++){ //run the loop while size_t type i from 0 to size of contours_b-1 size by increasing i 1
    if(radius_b[i] > iMin_tracking_ball_size){ //if radius_b[i] is larger than iMin_tracking_ball_size
        Scalar color = Scalar( 255, 0, 0); //set scalar color as 255 blue
        drawContours( hsv_frame_blue_canny, contours_b_poly, (int)i, color, 1, 8, vector<Vec4i>(), 0, Point() ); //draw contour
output&inputhsv_frame_blue_canny with contours_b_poly, contour index i, color: color, thickness: 1, linetype: 8, hierarchy, offset: 0
        vector<float> ball_position_b; //declare float vector named ball_position_b
        ball_position_b = pixel2point(center_b[i], radius_b[i]); //put position info to ball_position_b processed by function; pixel2point
        float isx = ball_position_b[0]; //declare float isx and put value; ball_position_b[0]
        float isy = ball_position_b[1]; //declare float isy and put value; ball_position_b[1]
        float isz = ball_position_b[2]; //declare float isz and put value; ball_position_b[2]
        string sx = floatToString(isx); //declare string sx and put value isx processed(change data type from float to string) from floatToString
        string sy = floatToString(isy); //declare string sy and put value isy processed(change data type from float to string) from floatToString
        string sz = floatToString(isz); //declare string sz and put value isz processed(change data type from float to string) from floatToString
        text = "Blue ball:" + sx + "," + sy + "," + sz; //put message(Blue ball: sx,sy,sz) to text
        putText(result, text, center_b[i], 2, 1, Scalar(0, 255, 0), 2); //indicate text to result matrix, content: text, position: center_b[i], font type: 2, font scale: 1,
color: 255 green, line thickness: 2
        circle(result, center_b[i], (int)radius_b[i], color, 2, 8, 0); //indicate circle to result, position: center_r[i], radius: radius_r[i], font type: 2, font scale: 1,
color: 255 green, line thickness: 2
    }
}
// Show the frames
imshow("Video Capture", calibrated_frame); //show image calibrated_frame on "VideoCapture"
imshow("Object Detection_HSV_Red", hsv_frame_red); //show image hsv_frame_red on "Object Detection_HSV_Red"
imshow("Object Detection_HSV_Blue", hsv_frame_blue); //show image hsv_frame_blue on "Object Detection_HSV_Blue"
imshow("Canny Edge for Red Ball", hsv_frame_red_canny); //show image hsv_frame_red_canny on "Canny Edge for Red Ball"
imshow("Canny Edge for Blue Ball", hsv_frame_blue_canny); //show image hsv_frame_blue_canny on "Canny Edge for Blue Ball"
imshow("Result", result); //show image result on "Result"
}
return 0;
}
string intToString(int n){ //change int to string
    stringstream s; //declare stringstream type value s
    s << n; //put n to s
    return s.str(); //return string type value s
}
string floatToString(float f){ //change float to string
    ostringstream buffer; //declare ostringstream type value buffer
    buffer << f; //put f to buffer
    return buffer.str(); //return string type value buffer
}
void morphOps(Mat &thresh){ //dilate and erode image
    //create structuring element that will be used to "dilate" and "erode" image.
    //the element chosen here is a 3px by 3px rectangle
    Mat erodeElement = getStructuringElement( MORPH_RECT, Size(3,3));
    //dilate with larger element so make sure object is nicely visible
    Mat dilateElement = getStructuringElement( MORPH_RECT, Size(8,8));
    erode(thresh, thresh, erodeElement);
    erode(thresh, thresh, erodeElement);
    dilate(thresh, thresh, dilateElement);
    dilate(thresh, thresh, dilateElement);
}
vector<float> pixel2point(Point center, int radius){ //get center of ball and transform it into ball position
    vector<float> position; //declare float type vector named position
    float x, y, u, v, Xc, Yc, Zc; //declare float type values
    x = center.x; //x; // .at(0);
    y = center.y; //y; //
    u = (x - intrinsic_data[2]) / intrinsic_data[0]; //calculate x position of image plane

```

```

v = (y-intrinsic_data[5])/intrinsic_data[4]; //calculate y position of image plane
Zc=(intrinsic_data[0]*fball_radius)/(2*(float)radius); //compute Z value of the ball center
Xc=u*Zc; //calculate real x position from camera coordinate
Yc=v*Zc; //calculate real y position from camera coordinate
Xc=roundf(Xc * 1000) / 1000; //make Xc to 4digit
Yc=roundf(Yc * 1000) / 1000; //make Yc to 4digit
Zc=roundf(Zc * 1000) / 1000; //make Zc to 4digit
position.push_back(Xc); //insert Xc to vector position
position.push_back(Yc); //insert Yc to vector position
position.push_back(Zc); //insert Zc to vector position
return position; //return vector position
}

// Trackbar for image threshodling in HSV colorspace : Red
void on_low_h_thresh_trackbar_red(int, void *){
    low_h_r = min(high_h_r-1, low_h_r); //set low_h_r as minimum value of high_h_r-1 and low_h_r
    setTrackbarPos("Low H", "Object Detection_HSV_Red", low_h_r); //set trackbar position as low_h_r on trackbar "Low H" on window "Object
Detection_HSV_Red"
}
void on_high_h_thresh_trackbar_red(int, void *){
    high_h_r = max(high_h_r, low_h_r+1); //set high_h_r as minimum value of high_h_r and low_h_r+1
    setTrackbarPos("High H", "Object Detection_HSV_Red", high_h_r); //set trackbar position as high_h_r on trackbar "High H" on window "Object
Detection_HSV_Red"
}
void on_low_h2_thresh_trackbar_red(int, void *){
    low_h2_r = min(high_h2_r-1, low_h2_r); //set low_h2_r as minimum value of high_h2_r-1 and low_h2_r
    setTrackbarPos("Low H2", "Object Detection_HSV_Red", low_h2_r); //set trackbar position as low_h2_r on trackbar "Low H2" on window "Object
Detection_HSV_Red"
}
void on_high_h2_thresh_trackbar_red(int, void *){
    high_h2_r = max(high_h2_r, low_h2_r+1); //set high_h_r as minimum value of high_h_r and low_h_r+1
    setTrackbarPos("High H2", "Object Detection_HSV_Red", high_h2_r); //set trackbar position as high_h2_r on trackbar "High H2" on window "Object
Detection_HSV_Red"
}
void on_low_s_thresh_trackbar_red(int, void *){
    low_s_r = min(high_s_r-1, low_s_r); //set low_s_r as minimum value of high_s_r-1 and low_s_r
    setTrackbarPos("Low S", "Object Detection_HSV_Red", low_s_r); //set trackbar position as low_s_r on trackbar "Low S" on window "Object Detection_HSV_Red"
}
void on_high_s_thresh_trackbar_red(int, void *){
    high_s_r = max(high_s_r, low_s_r+1); //set high_s_r as minimum value of high_s_r and low_s_r+1
    setTrackbarPos("High S", "Object Detection_HSV_Red", high_s_r); //set trackbar position as high_s_r on trackbar "High S" on window "Object Detection_HSV_Red"
}
void on_low_v_thresh_trackbar_red(int, void *){
    low_v_r = min(high_v_r-1, low_v_r); //set low_v_r as minimum value of high_v_r-1 and low_v_r
    setTrackbarPos("Low V", "Object Detection_HSV_Red", low_v_r); //set trackbar position as low_v_r on trackbar "Low V" on window "Object
Detection_HSV_Red"
}
void on_high_v_thresh_trackbar_red(int, void *){
    high_v_r = max(high_v_r, low_v_r+1); //set high_v_r as minimum value of high_v_r and low_v_r+1
    setTrackbarPos("High V", "Object Detection_HSV_Red", high_v_r); //set trackbar position as high_v_r on trackbar "High V" on window "Object
Detection_HSV_Red"
}
// Trackbar for image threshodling in HSV colorspace : Blue
void on_low_h_thresh_trackbar_blue(int, void *){
    low_h_b = min(high_h_b-1, low_h_b); //set low_h_b as minimum value of high_h_b-1 and low_h_b
    setTrackbarPos("Low H", "Object Detection_HSV_Blue", low_h_b); //set trackbar position as low_h_b on trackbar "Low H" on window "Object
Detection_HSV_Blue"
}
void on_high_h_thresh_trackbar_blue(int, void *){
    high_h_b = max(high_h_b, low_h_b+1); //set high_h_b as minimum value of high_h_b and low_h_b+1
    setTrackbarPos("High H", "Object Detection_HSV_Blue", high_h_b); //set trackbar position as high_h_b on trackbar "High H" on window "Object
Detection_HSV_Blue"
}
void on_low_s_thresh_trackbar_blue(int, void *){
    low_s_b = min(high_s_b-1, low_s_b); //set low_s_b as minimum value of high_s_b-1 and low_s_b
    setTrackbarPos("Low S", "Object Detection_HSV_Blue", low_s_b); //set trackbar position as low_s_b on trackbar "Low S" on window "Object

```

```

Detection_HSV_Blue"
}
void on_high_s_thresh_trackbar_blue(int, void *){
    high_s_b = max(high_s_b, low_s_b+1); //set high_s_b as minimum value of high_s_b and low_s_b+1
    setTrackbarPos("High S", "Object Detection_HSV_Blue", high_s_b); //set trackbar position as high_s_b on trackbar "High S" on window "Object
Detection_HSV_Blue"
}
void on_low_v_thresh_trackbar_blue(int, void *){
    low_v_b= min(high_v_b-1, low_v_b); //set low_v_b as minimum value of high_v_b-1 and low_v_b
    setTrackbarPos("Low V","Object Detection_HSV_Blue", low_v_b); //set trackbar position as low_v_b on trackbar "Low V" on window "Object
Detection_HSV_Blue"
}
void on_high_v_thresh_trackbar_blue(int, void *){
    high_v_b = max(high_v_b, low_v_b+1); //set high_v_b as minimum value of high_v_b and low_v_b+1
    setTrackbarPos("High V", "Object Detection_HSV_Blue", high_v_b); //set trackbar position as high_v_b on trackbar "High V" on window "Object
Detection_HSV_Blue"
}
// Trackbar for Canny edge algorithm
void on_canny_edge_trackbar_red(int, void *){
    setTrackbarPos("Min Threshold", "Canny Edge for Red Ball",lowThreshold_r); //set trackbar position as lowThreshold_r on trackbar "Min Threshold" on window
"Canny Edge for Red Ball"
}
void on_canny_edge_trackbar_blue(int, void *){
    setTrackbarPos("Min Threshold", "Canny Edge for Blue Ball",lowThreshold_b); //set trackbar position as lowThreshold_b on trackbar "Min Threshold" on window
"Canny Edge for Blue Ball"
}
}

```

-presentation materials

>1<sup>st</sup> PT

## ROLLING BALL COLLECTOR

CAPSTONE DESIGN 1<sup>ST</sup> PRESENTATION

Advisor: Prof. Yanghee Park;  
TA: Seoyoung Kim;  
Team: Shwan, Sangwon Hyeon, Daesung Kim, Minyoung Kim, Won Chul, Yeohan Lee, Sangmin Hong, Chul Seung

### CONTENTS



PROBLEM DEFINITION

- Common mission
- Function Analysis
- Define Unknowns



CONCEPT

- Concept Overview
- Pick up
- Heat Transfer & Vibration



OTHER WORKS

- Ros
- Open CV
- Labview



FUTURE WORKS

- Unsolved Problems
- Next Goals

## PROBLEM DEFINITION

Specify given problems and set the plan for the approach

### COMMON MISSION



DISTINGUISH

- Algorithm for exact mapping



< 5MIN

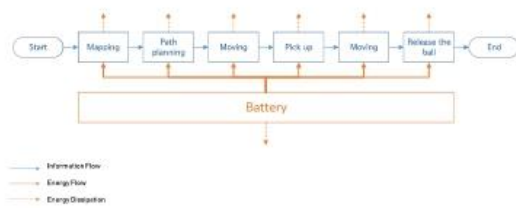
- Plan for the shortest route



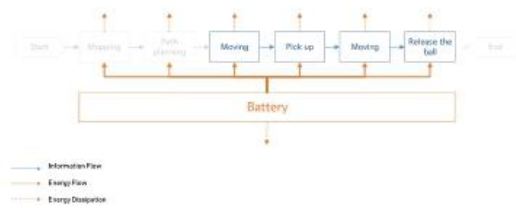
COOLING

- Effective Heat transfer
- Minimize battery consumption

### FUNCTION ANALYSIS



### FUNCTION ANALYSIS



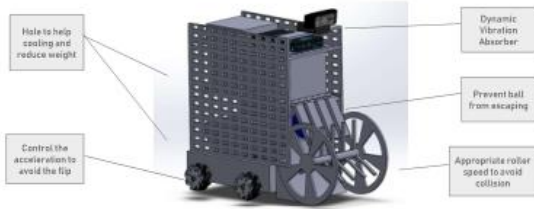
## CONCEPT

Ideation result for the pickup, heat transfer, and vibration

### CONCEPT OVERVIEW



## SUMMARY



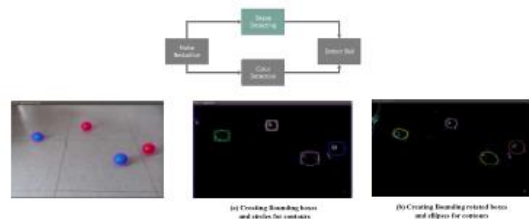
## OTHER WORKS

Brief summary for the progress of RGS, Open CV, and Labview

## ROS

TOPIC	SERVICE	Rviz
<ul style="list-style-type: none"> <li>Continuous communication which is used to send and receive the data</li> <li>Get the distance data and send it to myRIO to control the velocity</li> </ul>	<ul style="list-style-type: none"> <li>Sporadic communication ordering for the specific event</li> <li>Used for changing the direction and avoiding obstacles</li> </ul>	<ul style="list-style-type: none"> <li>Visualize of LIDAR /cm data</li> </ul> 

### CURRENT PROGRESS (2) OPEN CV



### CURRENT PROGRESS (3) LABVIEW

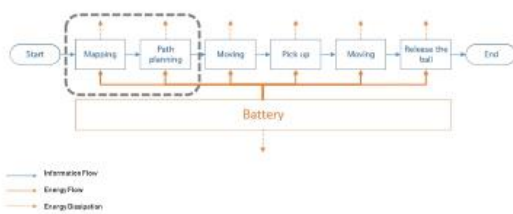
### Wheel & Position mode control



## FUTURE WORKS

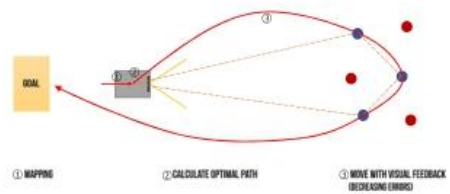
Overall summary for the system and next goal

## FUNCTION ANALYSIS

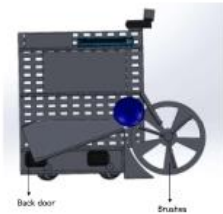


## PATH GENERATION

We can find an optimal path because our robot collect balls while moving



## PICK UP



## GOODS & BADS

### GOODS

- Time effective
  - Catch the ball during the movement
- Ease of control
  - One additional motor required
- Ease of release

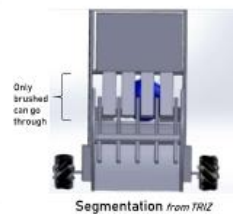
### BADS

- Go in and Get out
- Possibilities of hit the ball
- Instability
  - Additional mass of roller, slide and a storage
  - Center of mass goes higher

## SOLUTION

### BADS

- Go in and Get out
- Possibilities of hit the ball
- Instability
  - Additional mass of roller, slide and a storage
  - Center of mass goes higher

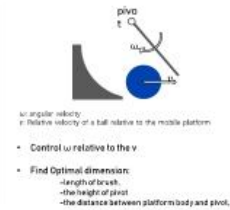


Segmentation from TRIZ

## SOLUTION

### BADS

- Go in and Get out
- Possibilities of hit the ball
- Instability
  - Additional mass of roller, slide and a storage
  - Center of mass goes higher



## SOLUTION

### BADS

- Go in and Get out
- Possibilities of hit the ball
- Instability
  - Additional mass of roller, slide and a storage
  - Center of mass goes higher

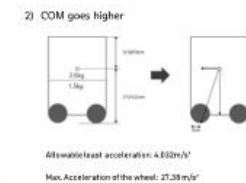


Taking out from TRIZ

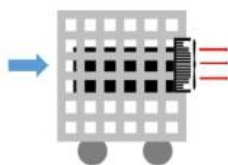
## SOLUTION

### BADS

- Go in and Get out
- Possibilities of hit the ball
- Instability
  - Additional mass of roller, slide and a storage
  - Center of mass goes higher



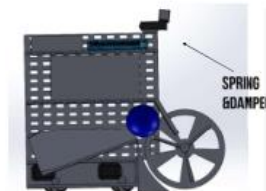
## HEAT TRANSFER



### STRATEGY

- Minimize Energy usage
  - Optimal path planning
  - Light weight
  - Rotating rotar only for the pick up process
- Increase Surface area
  - Perforated Frame with high conductivity
- Heat Sink and Cooling Fan
  - Maximize air flow

## VIBRATION



### STRATEGY

- Absorb vibration only for the cam, rather than the whole system
- Make high frequency ratio  $(\omega/\omega_0 > 2)$  & low damping ratio

## NEXT GOALS

ROS	OPEN CV	LABVIEW
<ul style="list-style-type: none"> <li>Get the information from webcam and transfer to myRIO</li> <li>Motor control using Xbox</li> <li>Based on the webcam, find the optimal route</li> </ul>	<ul style="list-style-type: none"> <li>Find proper ball tracking method at various condition</li> <li>3D mapping</li> <li>Combine mapping data with Lidar</li> </ul>	<ul style="list-style-type: none"> <li>Mecanum wheels control <ul style="list-style-type: none"> <li>Straigh movement</li> <li>Rotation</li> </ul> </li> <li>Roller motor control</li> <li>Backdoor open system</li> </ul>

## Q&A

## REFERENCE

## BRAINSTORMING

1. 격자형 설계
2. W자형 설계
3. 끈끈이
4. 피제자
4. 우선 다 덮고 골라내기 (갈매기 형)
6. 발간을 채우기
7. 바구니를 깔고가기
8. 벽으로 밀어서 일렬로 만들기
9. 바구니까지 이어서는 통로

## BRAINSTORMING

1. 드라이아이스 (혹은 다른 양모)
2. Pms 위아래로 공간을 두어 이동을 이 용하여 convection
3. 열유량이 큰 오일 에 담그기
3. 발열판과 물러


## DECISION MATRIX (1) PICK UP

Issue: Choose the most efficient pick up method		Relative 가중치를 곱하기	가중치	가중치	가중치	가중치	가중치	가중치
		정확도	집는 시간	복구 가능성	크기/다용도성	제작원리성	에너지 소모	
정확도	26	0	1	1	0	0	1	1
집는 시간	18	0	1	1	1	0	1	1
복구 가능성	8	0	0	0	0	0	0	0
크기/다용도성	14	0	0	0	1	0	1	1
제작원리성	26	0	1	1	1	0	1	1
에너지 소모	8	0	1	1	1	1	1	0
		0	26	26	34	4	64	56

## DECISION MATRIX (1) PICK UP

Issue: Choose the most efficient pick up method		정확도	집는 시간	복구 가능성	크기/다용도성	제작원리성	에너지 소모
정확도	26	0.74	0.68	0.62	0.56		
집는 시간	18	0.62	0.6	0.5	0.8		
복구 가능성	8	0	0	0.32	0.5		
크기/다용도성	14	0.7	0.7	0.44	0.56		
제작원리성	26	0.66	0.66	0.67	0.7		
에너지 소모	8	0.74	0.8	0.56	0.48		
		68.88	64.92	65.14	62.68		

## DYNAMIC INTERPRETATION

1. 


$$Ma = 4F_t$$

$$T - \mu Fr = \frac{1}{2} mra$$

$$T = \frac{3}{2} mra + \frac{1}{2} Mra$$

$$a = \frac{3mr/2 + Mr/4}{M}$$

max T is given (motor spec, 2.38N)  
Find the maximum acceleration with respect to given T by this equation.  
Maximum a: 27.38m/s<sup>2</sup>

2. 

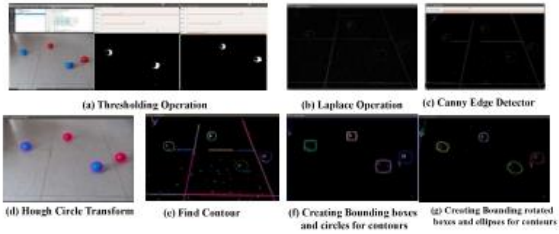
$$a: g = 7.17.0122$$

$$\therefore a = 4.032m/s^2$$

In our model, mass and geometry is shown in figure 2.  
In geometry, vector sum of g and platform's acceleration must do not exceed the contact point of the wheel and ground



## BALL DETECTING DETAIL



## GANTT CHART

구분	작업명	시작	종료	진행	진행률 (%)	진행률 (%)
1	System Overview	2018-03-19	2018-03-31	8.4%		
2	System Overview	2018-03-19	2018-03-31	2%		
3	System Overview	2018-03-19	2018-03-31	2%		
4	System Overview	2018-03-19	2018-03-31	2%		
5	System Overview	2018-03-19	2018-03-31	2%		
6	System Overview	2018-03-19	2018-03-31	2%		
7	System Overview	2018-03-19	2018-03-31	2%		
8	System Overview	2018-03-19	2018-03-31	2%		
9	System Overview	2018-03-19	2018-03-31	2%		
10	System Overview	2018-03-19	2018-03-31	2%		

THANK YOU

> 2<sup>nd</sup> PT



Wolfram-Str. 1, Langhanspark  
14, Sanyang-Kim  
Daeu-Heindeu / Sanyang-huon, Sanyang-Kim, Malyang-Kim, Wan-Chu, Yeh-Hui, Sanyang-Hong, Chou Sanyang

## CONTENTS

1<sup>st</sup> PROTOTYPE

## HARDWARE

SOFTWARE

#### FUTURE WORKS

- Summary of concept
- Feedback
- Prototype overview

- Gear
- Vibration
- Pickup
- Hapt

- System Integration
- OpenCV
- Labview
- ROS

- Unsolved Problems
- Final Goals

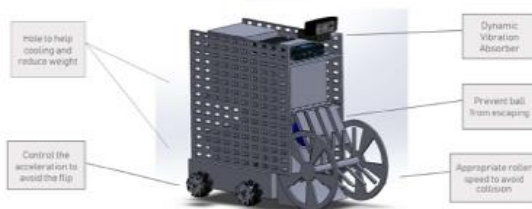
## 1<sup>ST</sup> PROTOTYPE

How we developed the very first concept?

### SUMMARY OF PREVIOUS CONCEPT



### SUMMARY OF PREVIOUS CONCEPT



## PROBLEMS



HEAVY

- ineffective management of the power



### WEAK SPEED STRATEGY

- No Competitiveness
- Long heat setting time

## HARDWARE

### Detailed Analysis and Design

## MATERIAL



PLASTIC BOX

### Large Transformation



ACRYLIC PANEL

Heavy Weight



ALUMINUM

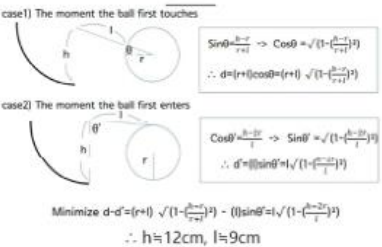
PROTOTYPE OVERVIEW



PICK UP



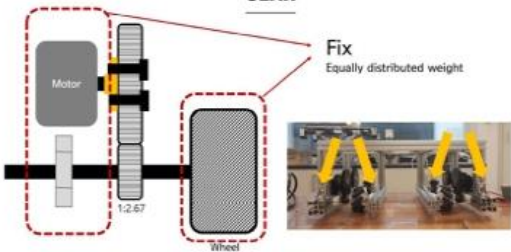
PICK UP



GEAR



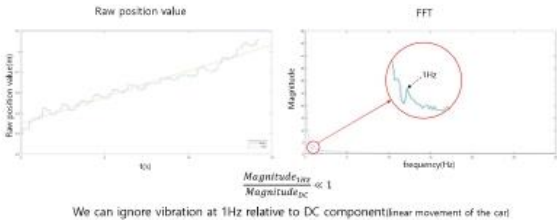
GEAR



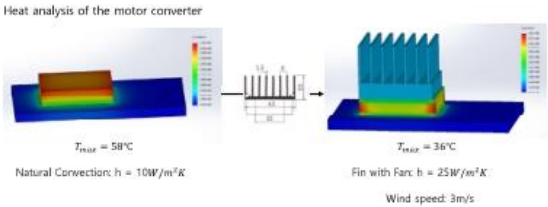
VIBRATION



VIBRATION



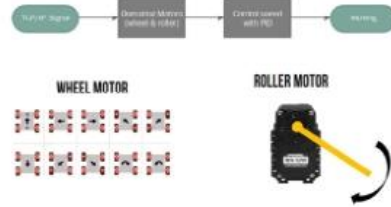
HEAT



## SUMMARY



## LABVIEW

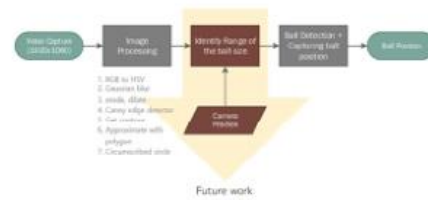


## SOFTWARE

Current Progress: Ball detecting strategy

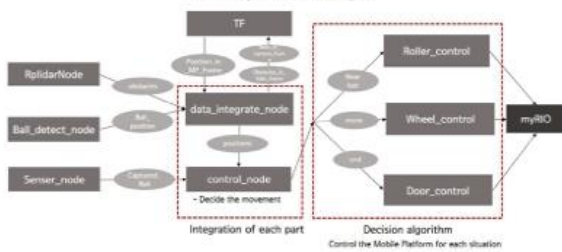
## OPEN CV

Ball detection



## ROS

Whole Progress of the Collecting Ball



## INTEGRATION OF EACH PART

Integration with OpenCV

- Camera recognizes the balls.
- Find the position of the balls in camera frame.
- Topic communication with another node to decide the Mobile Platform's action.



Integration with LabView

- TCP/IP communication with NUC and myRIO.
- Mobile Platform operates in 10 modes.
- Xbox controller or finding blue ball algorithm can control the 10 modes.



## DESIGN ALGORITHM

Ball Detecting



## DESIGN ALGORITHM

Avoid the red ball



## REAL PROJECT CONDITION



## FUTURE WORKS

What we have to do for the final project

## COMPENSATE DESIGN

### VIBRATION

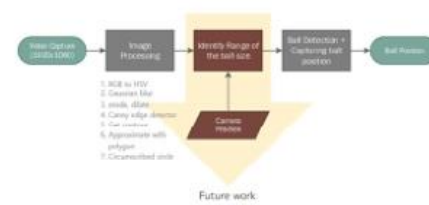
- Robust design
  - Vibration absorber system at the gear-wheel
- More accurate detection
  - M.C.K analysis of the cam and design vibration absorber

### HEAT TRANSFER

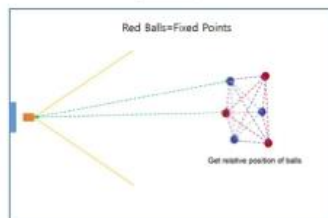
- Identify the heat dissipating part
  - Check the actual temperature variance using thermo-graphic camera
- Design fan & heat sink

## OPEN CV

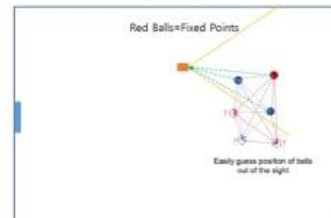
Ball detection



## MAPPING IDEA

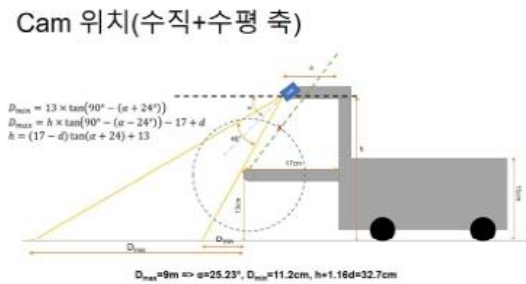
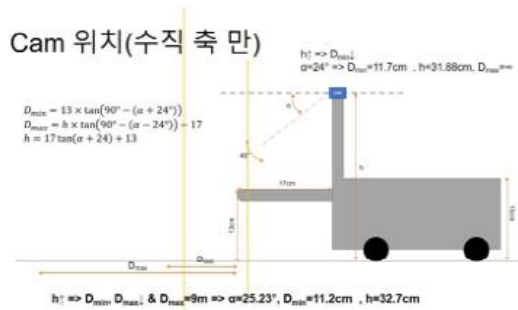
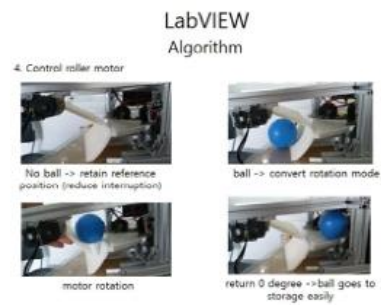
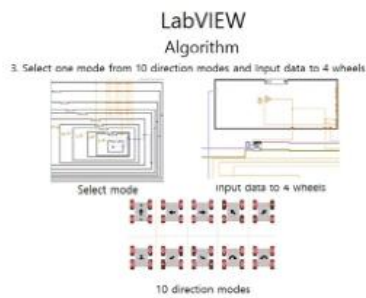
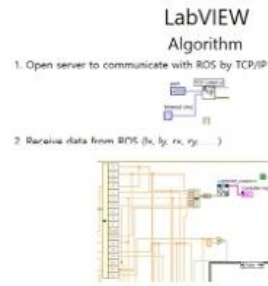
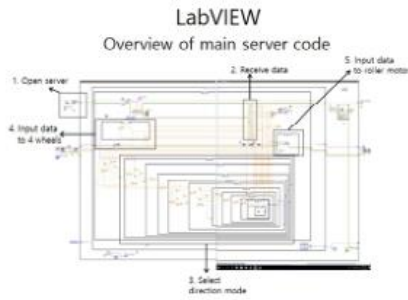


## MAPPING IDEA



## Q&A

## REFERENCE



### Cam 위치 요약

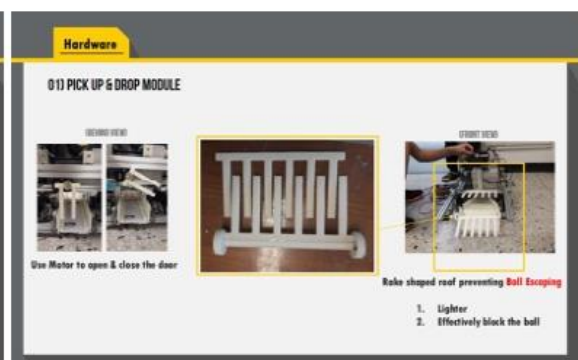
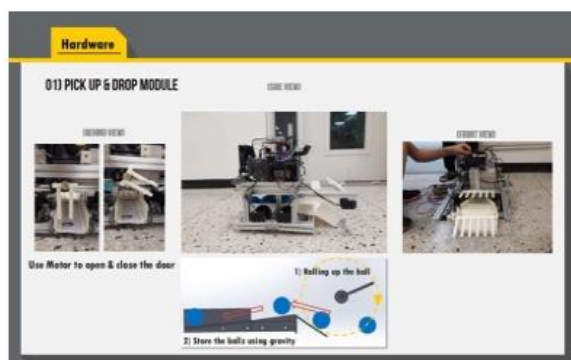
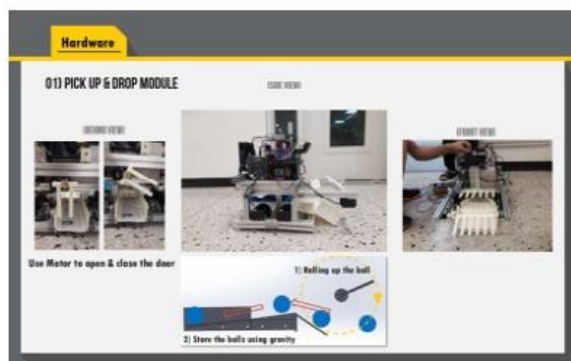
- 시야에 차체는 안 들어온다고 가정
- $D_{min}$ 의 최댓값=11.7cm
- $D_{max}$ 가 9m이기 위해서  $h=32.7\text{cm}$ ,  $\alpha=25.23^\circ$ 이면 되고, 이때 인식 가능한 최소 거리( $D_{min}$ )는 11.2cm가 된다.

$h \Rightarrow D_{min}$   
 $D_{max} = 10\text{cm} \Rightarrow \alpha = 28.43^\circ, h = 35.1\text{cm}, D_{min} = 4.36\text{m}$

THANK YOU



> Final PT



**Hardware**

### 02) GEAR SYSTEM



Effect of Gear-Wheel System (i.e. 1<sup>st</sup> Presentation)




X2.67 faster!

**Hardware**

### 03) FRAME ISSUE

**Primitive Design**



**Very Rigid Body frame**  
Assumption: The floor will be flat

**However, in real situation**



**Why the diagonal movement is stopped?**  
The field is not flat.  
Not all the wheels touches the floor.



Detach  
Mecanum wheel system


**Hardware**

### 03) FRAME ISSUE


**Improved Design**

Lack of Time  
Lack of Cost  
Preserve Stability

TRIZ : Segmentation




Segmentation : Add Degree of Freedom



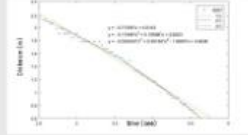
**Hardware**

### 04) VIBRATION ANALYSIS

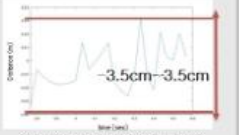
**Ball Depth Detection**



**Raw Distance Data (AC+DC)**



**Vibration (AC) Distance Data**




Eliminate 2nd order polynomial & Phil errors

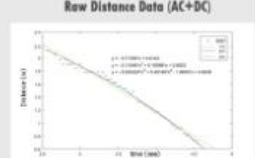
**Hardware**

### 04) VIBRATION ANALYSIS


**Horizontal position Detection**



**Raw Distance Data (AC+DC)**



**Vibration (AC) Distance Data**




Eliminate 2nd order polynomial & Phil errors

**Hardware**


### 04) VIBRATION ANALYSIS

**Acceptable Range of Picking up**


**Ball Pick up Algorithm :**  
The roller starts to rotate when the blue ball is close away



**Side view**



**Front view**



Depth error  
Horizontal Error

60cm

Detection error from vibration is in affordable range !

**Hardware**

### 05) HEAT TRANSFER

**Before Heat Transfer Design**

Roller Converter : 40<sup>o</sup>C  
Wheel Motor Converter : 67<sup>o</sup>C  
Heat Transfer Options is required  
NUC: 45-50<sup>o</sup>C



Analyze required spec of fin & fan!



**Hardware**

### 05) HEAT TRANSFER

Assumption: Q=3.5W (converter efficiency 93%)

**Analytical Solution**

(1) No Fin&fan, Natural Convection  
T<sub>amb</sub>=27.14  
P<sub>conv</sub>=1.551  
P<sub>rad</sub>=14.1  
T<sub>roller</sub>=25<sup>o</sup>C  
ΔT=38.67<sup>o</sup>C  
ΔT<sub>roller</sub>=63.67<sup>o</sup>C

(2) Fin & Forced Convection  
3m/s  
P<sub>conv</sub>=14.1  
P<sub>rad</sub>=14.1  
Fin Efficiency = 0.895  
ΔT=4.63<sup>o</sup>C  
ΔT<sub>roller</sub>=29.63<sup>o</sup>C

**Solidworks Simulation**

Condition Set  
1) Steady state  
2) Convergence Tolerance: 0.0001  
(1) No Fin&fan, Natural Convection  
Result: T<sub>roller</sub>=65<sup>o</sup>C  
(2) Fin & Forced Convection  
Result: T<sub>roller</sub>=43<sup>o</sup>C  
T<sub>ambient</sub>=30<sup>o</sup>C

Two results are same! Suggested Fin&Fan is appropriate!



**Hardware**

053 HEAT TRANSFER

Heat Transfer system

Fin system

Fan

Real Temperature obtained from thermal camera

NUC

~37.6

28.9

33.6

FRONT VIEW

SIDE VIEW

BACK VIEW

**Software**

**Software**

Software Integration

OpenCV

ROS

LabView

Ball\_detect\_node

- Detect the ball and get the position of the ball
- Publish the position data of Red, Blue and Green balls

Control\_node

- Subscribe the position data
- Decide the proper action of each situation
- Publish the motor control signal

Labview

- Subscribe the motor control signal
- Decide the motor speed ...

**Software**

Overall Path Generation Algorithm

```

graph TD
    Start([Start]) --> Ball37{Ball 37?}
    Ball37 -- Yes --> BackToBasket[Back to basket]
    BackToBasket --> Start
    Ball37 -- No --> BlueBall{Blue ball?}
    BlueBall -- Yes --> RotateCCW[Rotate CCW]
    RotateCCW --> Ball37
    BlueBall -- No --> BallDetection[Ball Detection]
    BallDetection --> HearRedBall{Hear Red ball?}
    HearRedBall -- Yes --> EvadeRedBall[Evade the red ball]
    EvadeRedBall --> PickBlueBall[Pick blue ball]
    PickBlueBall --> Ball37
    HearRedBall -- No --> PickBlueBall
  
```

**Software**

Ball Detection

```

graph TD
    Start([Start left ultrasonic]) --> AlignBldyCenter[Align Bldy Center]
    AlignBldyCenter --> GoForward[Go forward]
  
```

**Software**

Pick the ball

```

graph TD
    Start([Pick the Ball]) --> HearRedBall{Hear Red ball?}
    HearRedBall -- Yes --> MoveSlightlyBack[Move slightly back]
    MoveSlightlyBack --> RotateRobot[Rotate Robot]
    RotateRobot --> HearRedBall
    HearRedBall -- No --> PickBlueBall[Pick blue ball]
    PickBlueBall --> Ball37{Ball 37?}
  
```

**Software**

Back to the basket

```

graph TD
    Start([Back to Basket]) --> MoveToCenter[Move to the center of two green balls]
    MoveToCenter --> GoToBasket[Go to the basket]
    GoToBasket --> AlignTurn180[Align & Turn 180]
    AlignTurn180 --> DropBlueBall[Drop the blue ball]
  
```

**Software**

Check the final system on the DEMO!!

Conclusion:  
Creative  
solutions

#### Creativity

##### Light & Simple structure

Reduce roller weight



Frame Segmentation



##### Fast Speed



Gear system!  
X2.67 faster

##### Efficient Algorithm

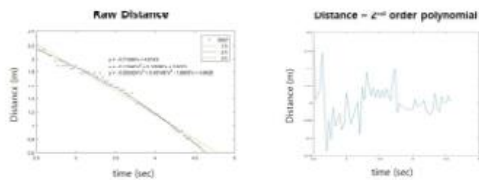
Picking while moving!  
Feedback control!



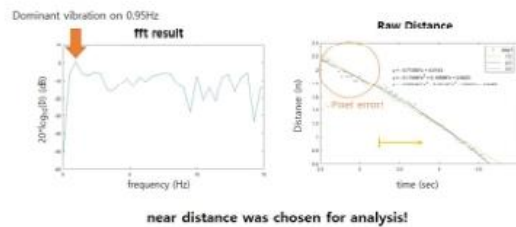
Q&A

Appendix

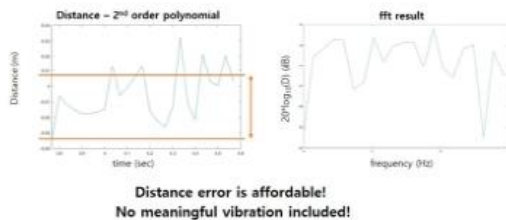
#### Vibration analysis



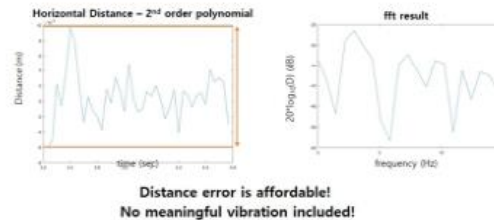
#### Vibration analysis- fft result



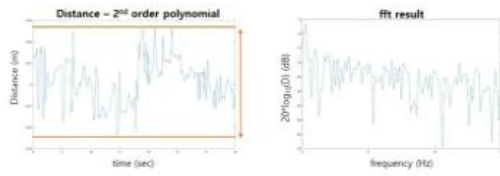
#### Vibration analysis- fft result (direct motion)



#### Vibration analysis- fft result (direct motion)



## Vibration analysis- fft result (diagonal motion)



Distance error is affordable!



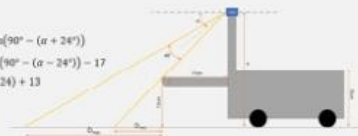
∴ 0,7cm vibration amplitude is acceptable within the roller boundary!

## Cam Position

$$D_{\min} = 13 \times \tan(90^\circ - (\alpha + 24^\circ))$$

$$D_{\max} = h \times \tan(90^\circ - (\alpha - 24^\circ)) - 17$$

$$h = 17 \tan(\alpha + 24) + 13$$



$$h \uparrow \Rightarrow D_{\min} \uparrow, D_{\max} \downarrow$$

$$D_{\max} \leq \infty \Rightarrow \alpha = 24^\circ, D_{\min} = 11.7 \text{ cm}, h = 31.9 \text{ cm}$$

Thank You

