Third Presentation

Robot Design Analysis & Demo

Group D

Team KIRBY

담당교수 최세범

we own no copyright to this image

Part 1

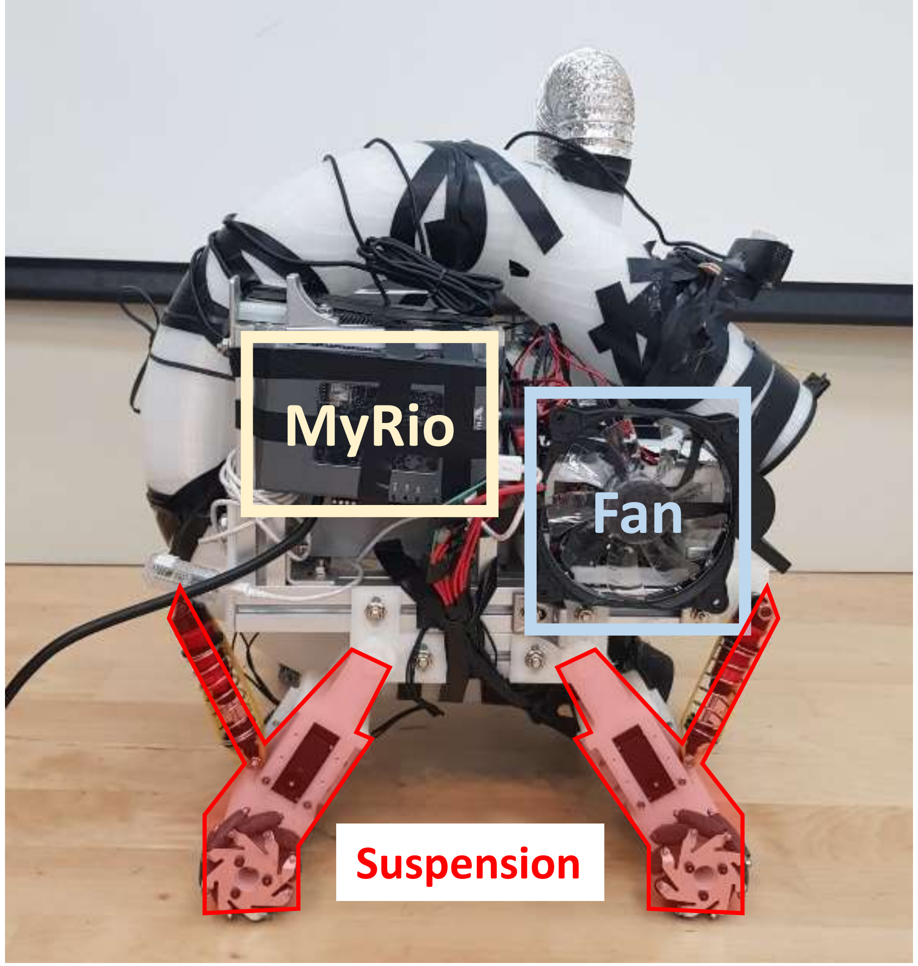
Design and Analysis Results

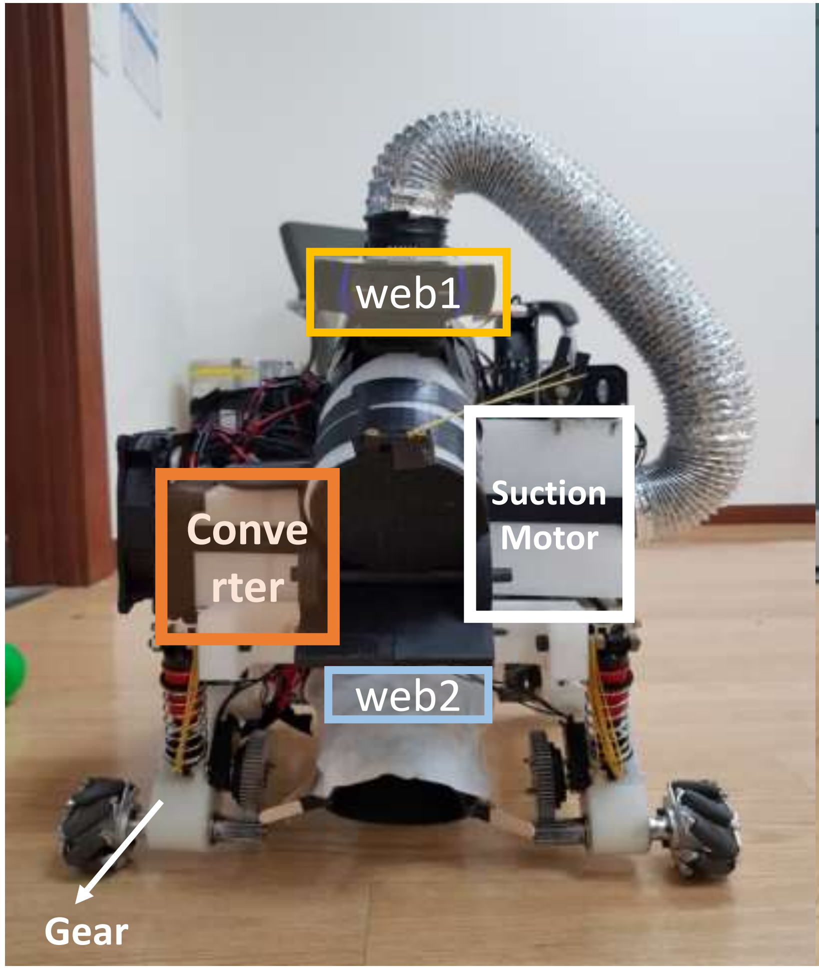
- System design
- Motor control
- Vision processing
- System integration

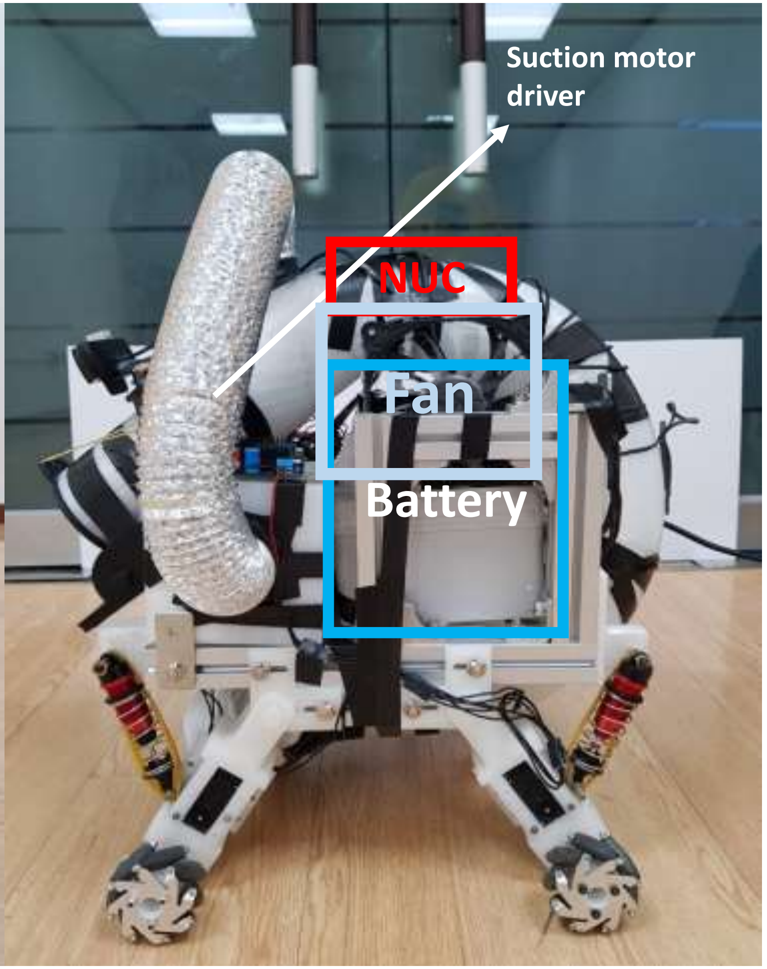
System Design

"Maximizing the advantages and minimizing the disadvantages of suction system"

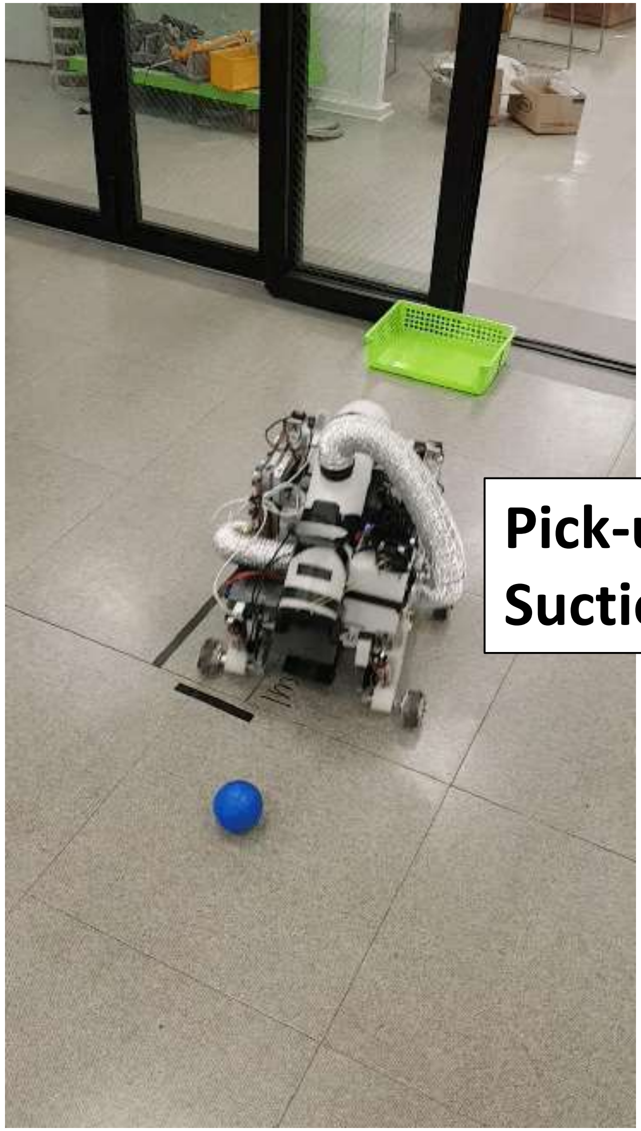
System Design Overview







System Design Why do we use SUCTION?



Pick-up module
Suction

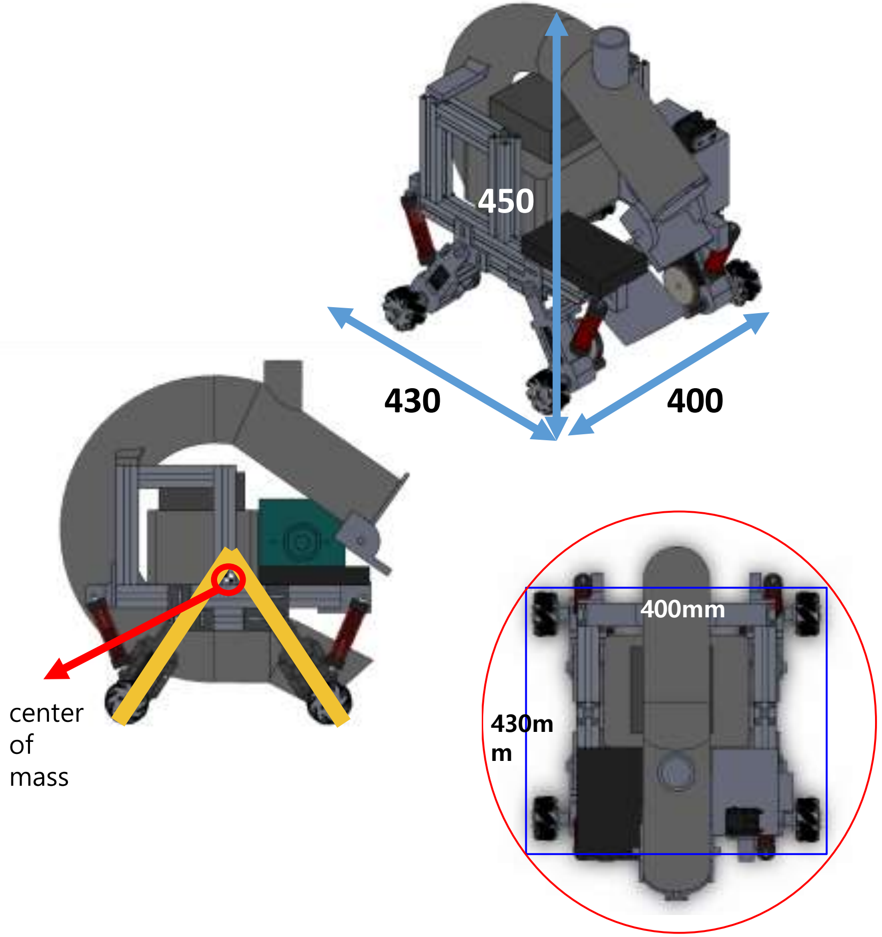
- Less moving parts → simple mechanism
- We can collect balls without pausing
- No accurate adjustment required
- Using suction air outflow for cooling

System Design Suction module: system development

Problem

But, using suction can produce **very Bulky system**

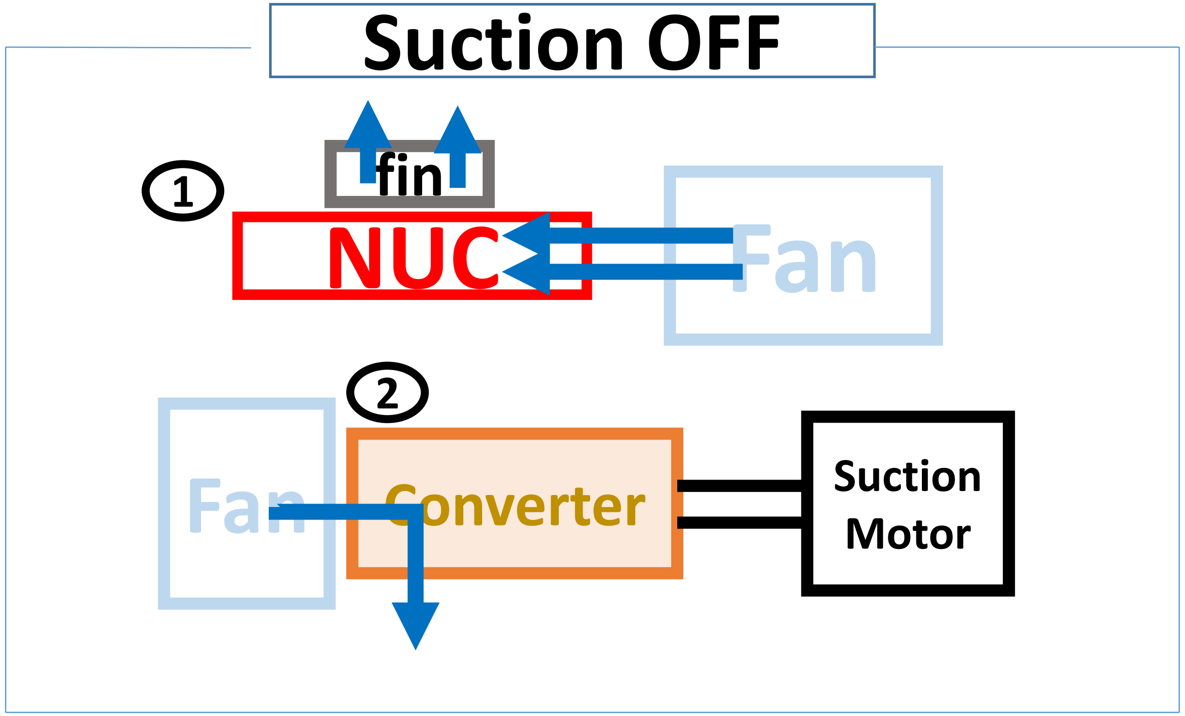
Solution	Result
Designed the **circular pipe (largest part) first** that maximized vacant space for other parts	**Compact design** (Cube-shaped)
Fit the heavier parts first, utilizing every vacant space	**Better vehicle control** with **Low COM!!!**
Added suspension to **eliminate pitch motion**	
Square-shaped base platform design	**Minimized radius of rotation**

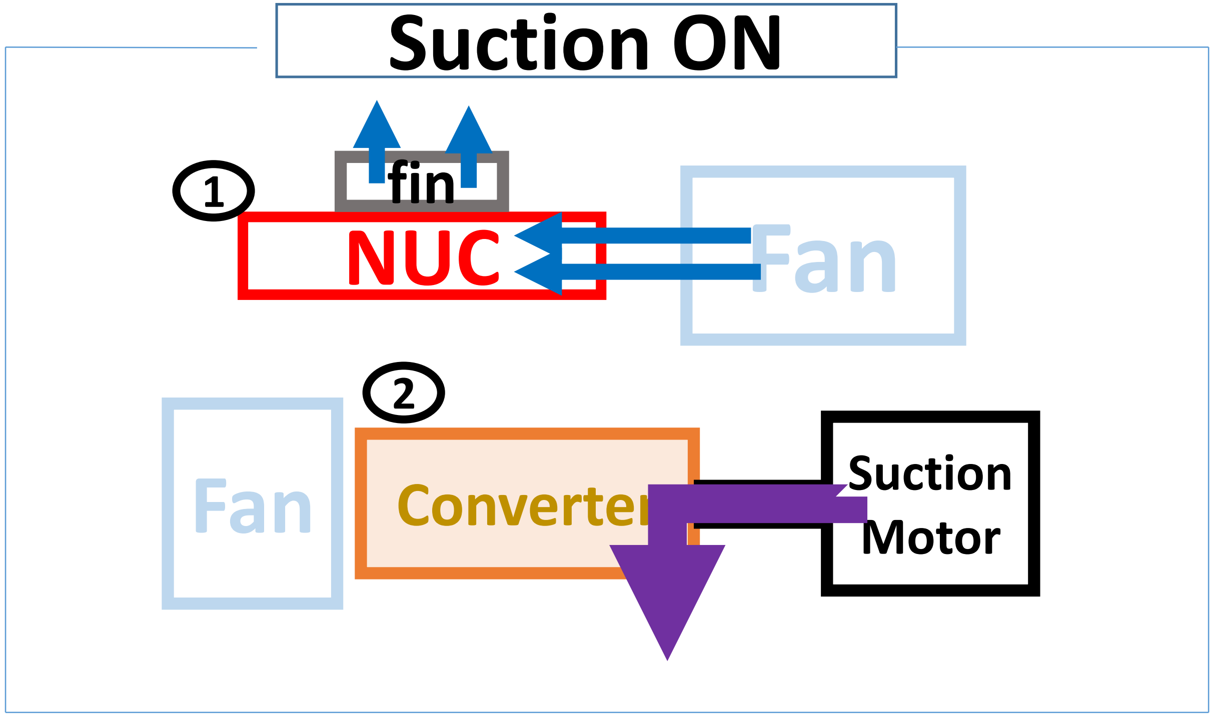


System Design Cooling module

Suction converter creates a lot of heat!

Therefore, we redesign not to use only suction out flow for cooling.





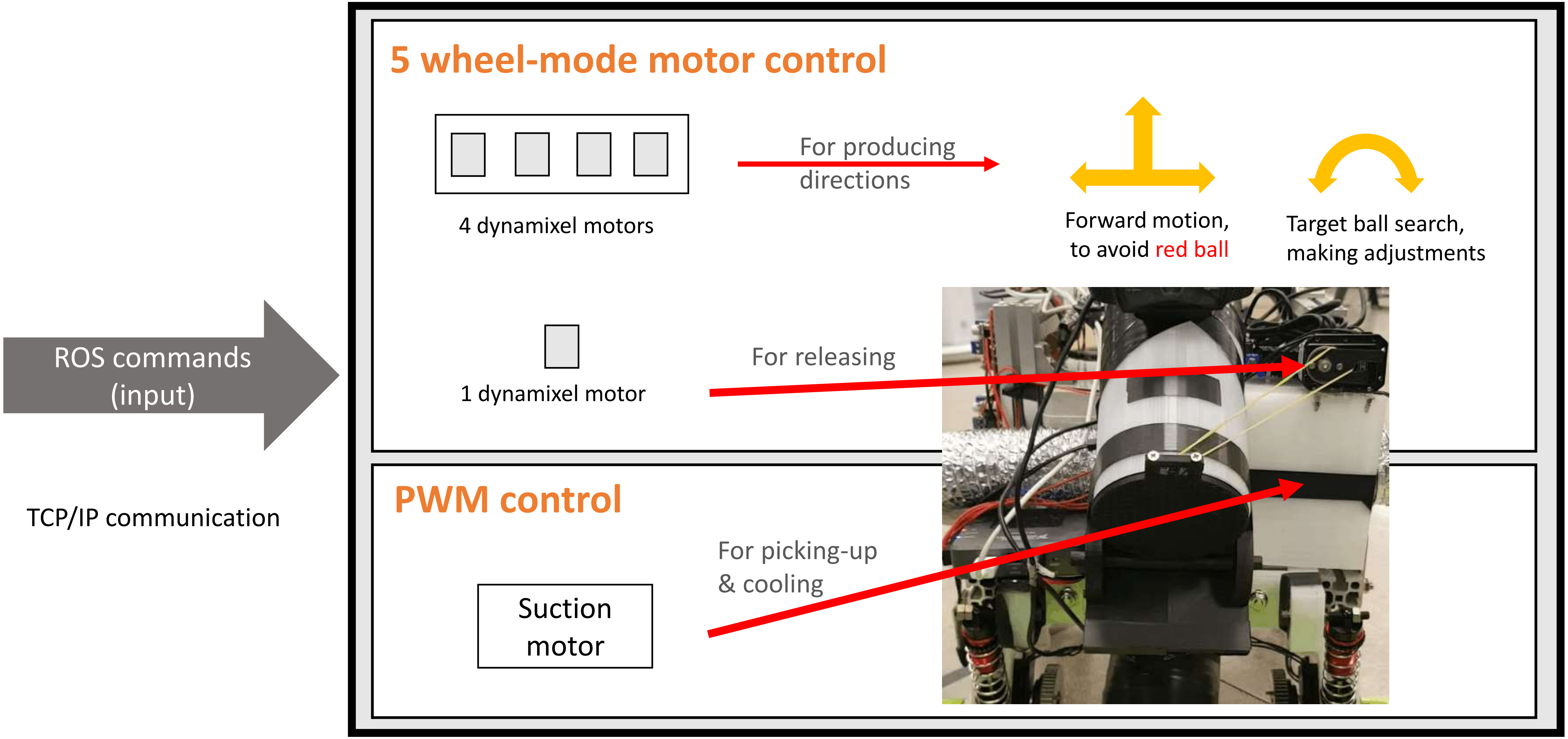
During suction off, fan is used to cool down both converter and NUC.

During suction on, powerful suction out flow is used in order to prevent converter from increasing its temperature rapidly.

Motor Control

“Motor rpm control achieving trapezoidal velocity profile for no slipping”

Motor Control Structure and function



ROS commands (input)

TCP/IP communication

5 wheel-mode motor control

4 dynamixel motors

For producing directions

Forward motion, to avoid red ball

Target ball search, making adjustments

1 dynamixel motor

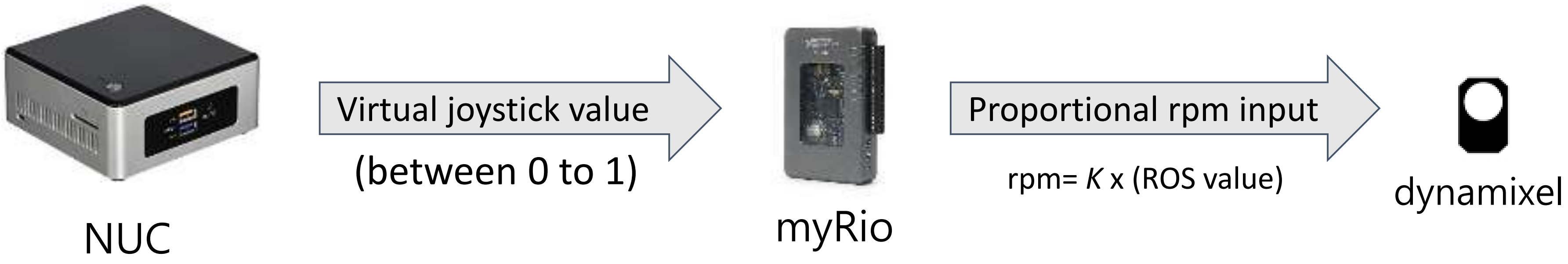
For releasing

PWM control

Suction motor

For picking-up & cooling

Motor Control Proportional rpm input



NUC

Virtual joystick value

(between 0 to 1)

myRio

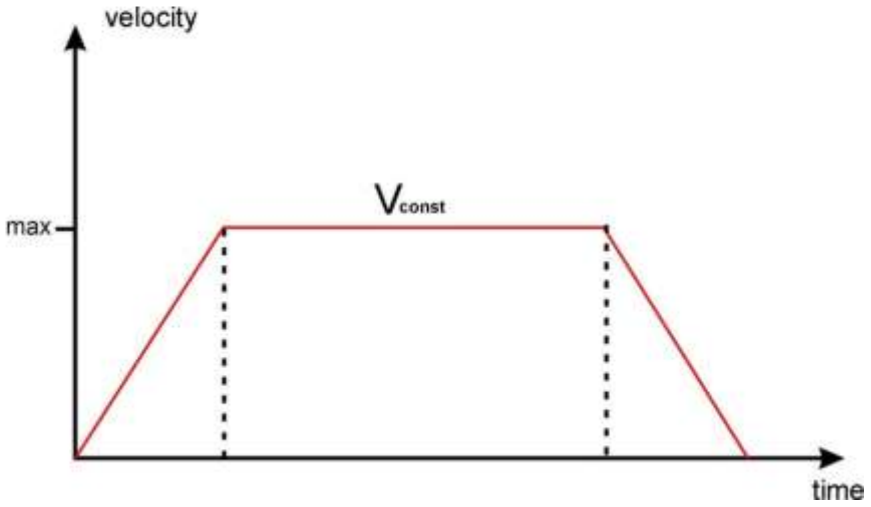
Proportional rpm input

rpm= *K* x (ROS value)

dynamixel

We give proportional rpm input to dynamixel to optimize motor control by achieving

trapezoidal velocity profile



We can obtain:

1. More stabilized motion
2. No slipping
3. Less pitching

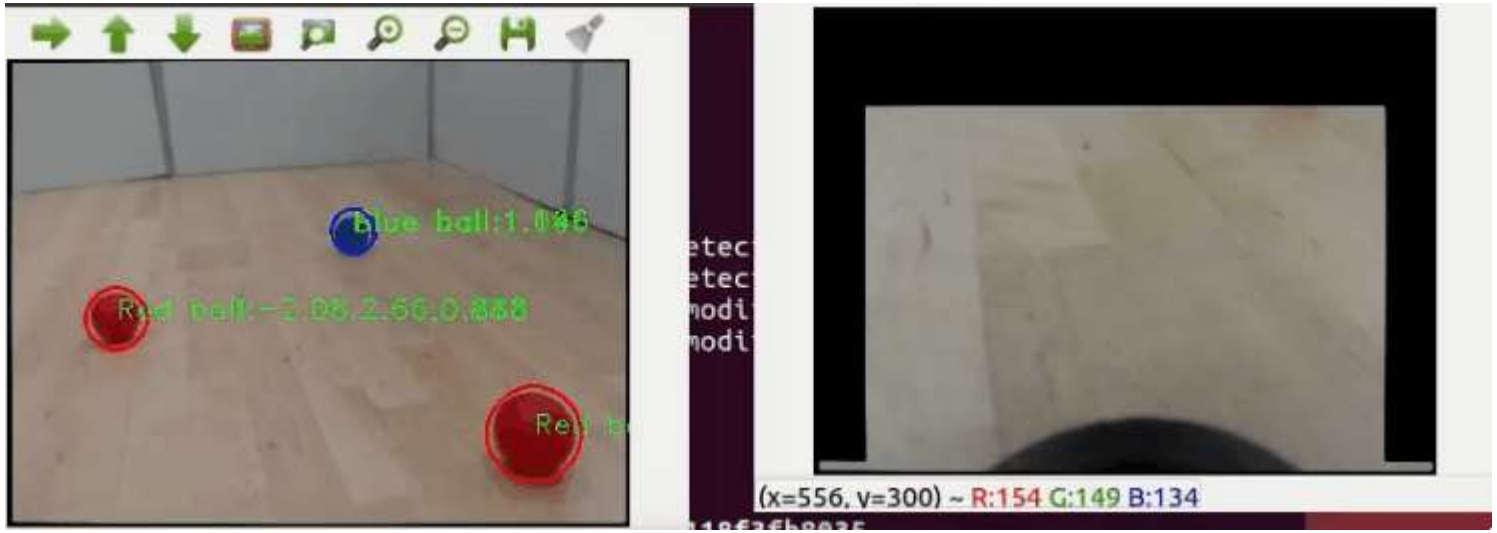
Optimized motor control

Vision Processing

“Filter node for noise handling to give better control”

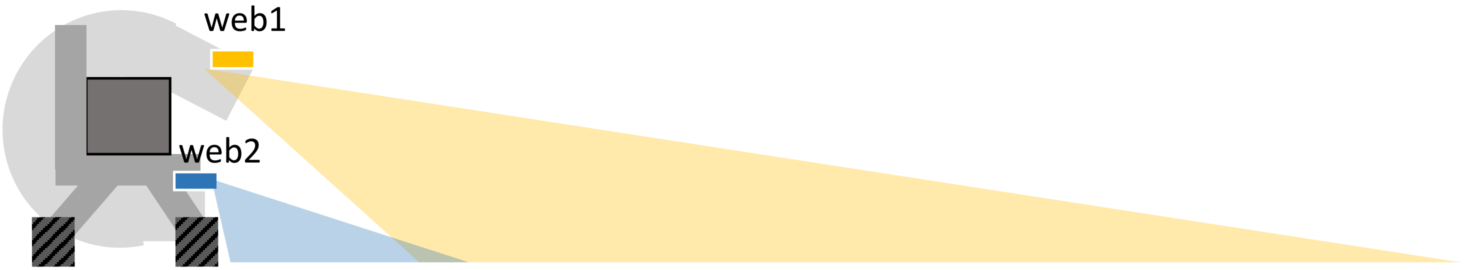
Vision Processing Broad vision with Dual camera

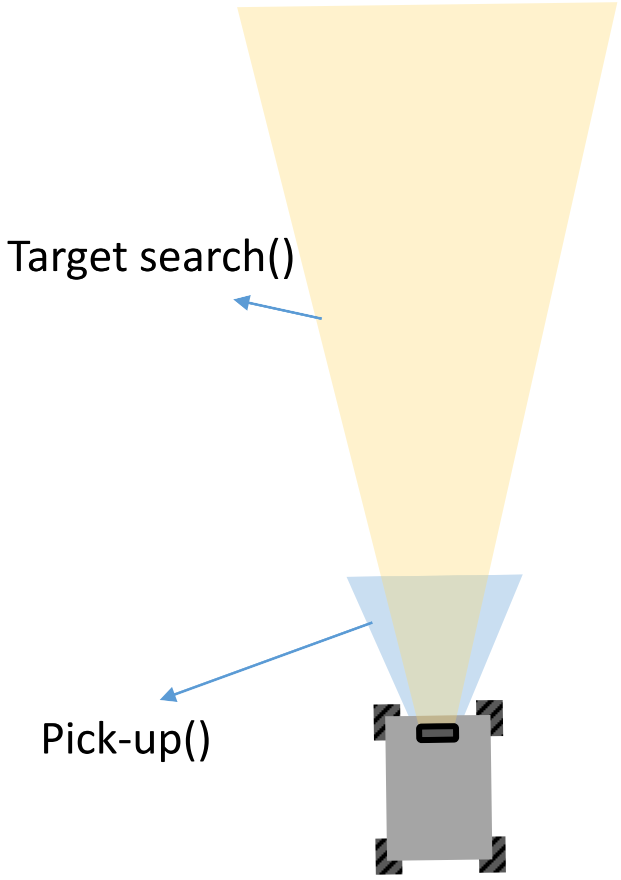
Dual cameras to **broaden vision range** for optimized ball pick-up



Webcam 1

Webcam 2



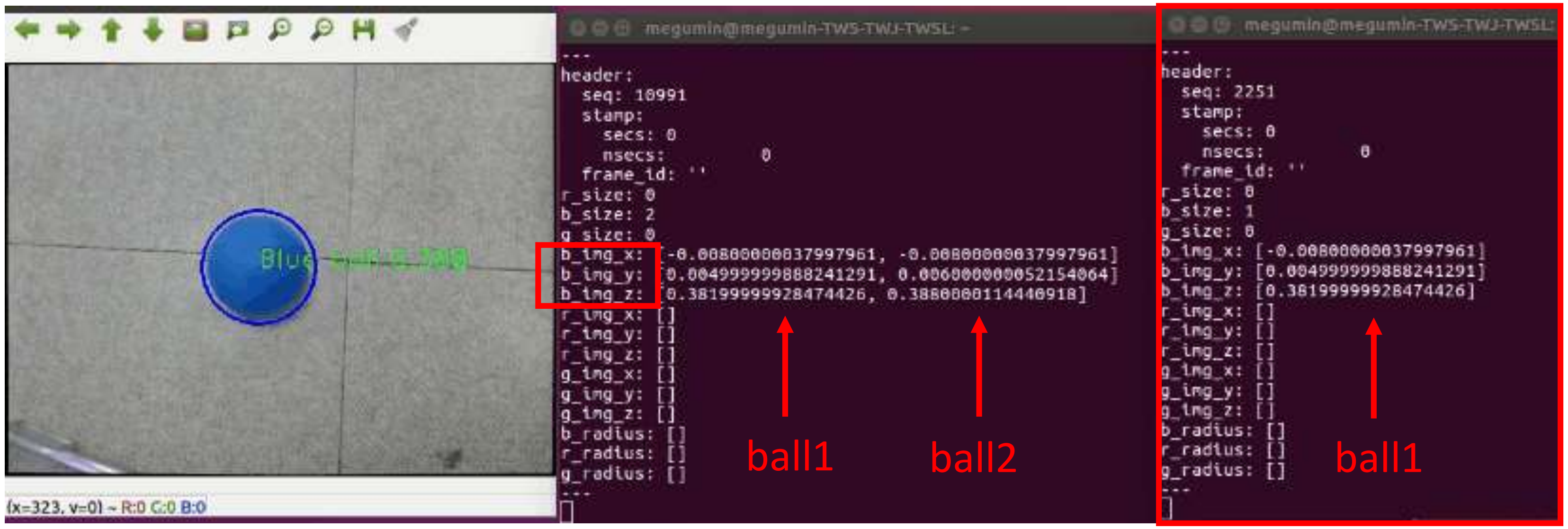


Vision Processing Noise Handling- Multiple ball counting

Filter_1

Problem: Multiple ball count data from the target single ball

Solution: Delete other data with **similar x,y coordinates**


Blue ball
(x=323, y=0) ~ R:0 G:0 B:0
megumin@megumin-TWS-TWJ-TWSL: ~
ball1
ball2
megumin@megumin-TWS-TWJ-TWSL:
ball1

```
---
header:
  seq: 10991
  stamp:
    secs: 0
    nsecs:          0
  frame_id: ''
r_size: 0
b_size: 2
g_size: 0
b_img_x: [-0.00800000037997961, -0.00800000037997961]
b_img_y: [0.004999999888241291, 0.006000000052154064]
b_img_z: [0.38199999928474426, 0.38800001144409180]
r_img_x: []
r_img_y: []
r_img_z: []
g_img_x: []
g_img_y: []
g_img_z: []
b_radius: []
r_radius: []
g_radius: []
---
```

```
---
header:
  seq: 2251
  stamp:
    secs: 0
    nsecs:          0
  frame_id: ''
r_size: 0
b_size: 1
g_size: 0
b_img_x: [-0.00800000037997961]
b_img_y: [0.004999999888241291]
b_img_z: [0.38199999928474426]
r_img_x: []
r_img_y: []
r_img_z: []
g_img_x: []
g_img_y: []
g_img_z: []
b_radius: []
r_radius: []
g_radius: []
---
```

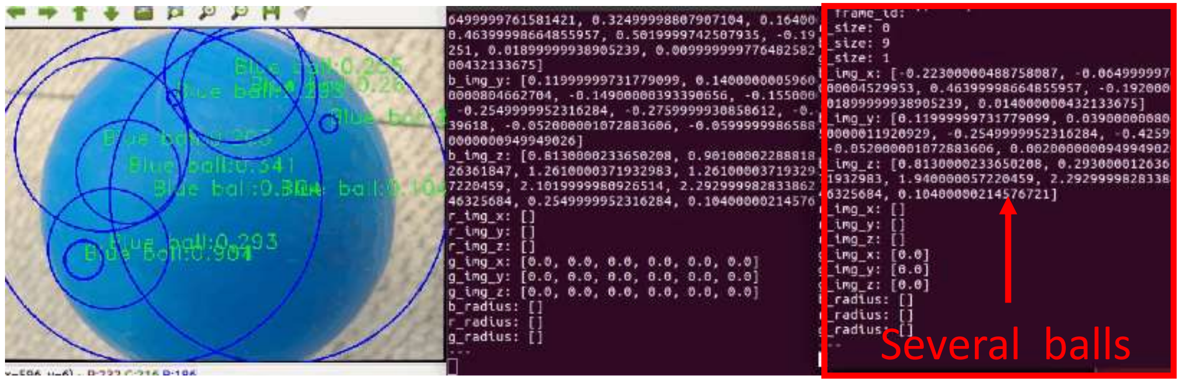
Filtered blue ball data using **Filter_1**

Vision Processing Noise Handling- Ball overlapping

Filter_2

Problem: Detect a smaller ball on the closest (largest) ball

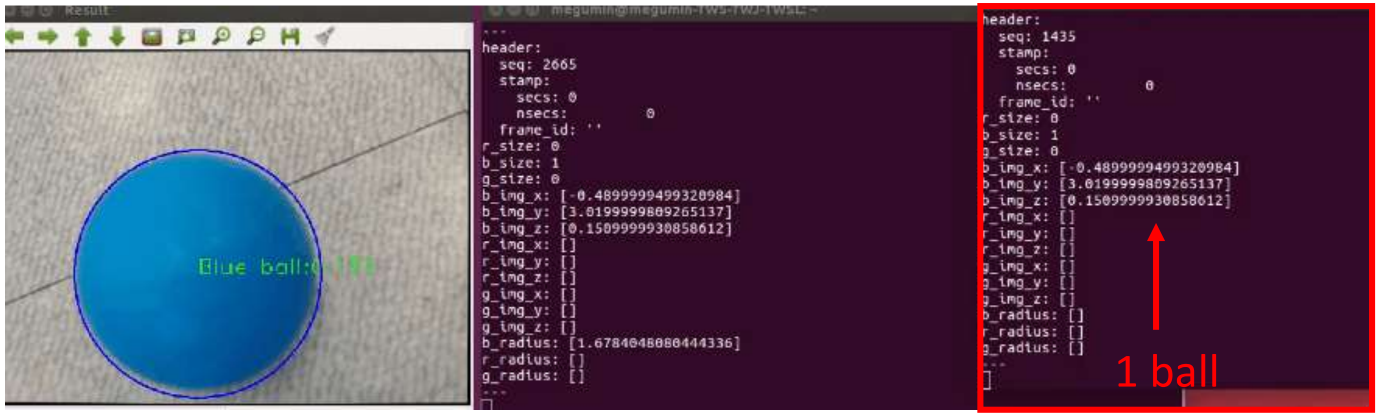
Solution: Delete the ball data detected **within the largest ball radius**



```
6499999761581421, 0.32499998807907104, 0.16400
0.46399998664855957, 0.5019999742507935, -0.19
251, 0.01899999938905239, 0.009999999776482582
00432133675]
b_img_y: [0.11999999731779099, 0.1400000005960
0000804662704, -0.14900000393390656, -0.155000
 -0.2549999952316284, -0.2759999930858612, -0.
39618, -0.05200000107288360б, -0.0599999986588
0000000949949026]
b_img_z: [0.8130000233650208, 0.9010000228881B
26361847, 1.2610000371932983, 1.26100003719329
7220459, 2.1019999980926514, 2.292999982833862
46325684, 0.2549999952316284, 0.10400000214576
r_img_x: []
r_img_y: []
r_img_z: []
g_img_x: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
g_img_y: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
g_img_z: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
b_radius: []
r_radius: []
g_radius: []
---
```

```
 frame_id: ''
_size: 0
_size: 9
_size: 1
_img_x: [-0.22300000488758087, -0.064999997
00004529953, 0.46399998664855957, -0.192000
01899999938905239, 0.014000000432133675]
_img_y: [0.11999999731779099, 0.03900000080
0000011920929, -0.2549999952316284, -0.4259
-0.05200000107288360б, 0.0020000000949949026
_img_z: [0.8130000233650208, 0.293000012636
1932983, 1.940000057220459, 2.2929999828338
6325684, 0.10400000214576721]
_img_x: []
_img_y: []
_img_z: []
_img_x: [0.0]
_img_y: [0.0]
_img_z: [0.0]
_radius: []
_radius: []
_radius: []
---
```

Filtered Blue ball data using **Filter_1**



```
---
header:
  seq: 2665
  stamp:
    secs: 0
    nsecs:         0
  frame_id: ''
r_size: 0
b_size: 1
g_size: 0
b_img_x: [-0.4899999499320984]
b_img_y: [3.0199999809265137]
b_img_z: [0.1509999930858612]
r_img_x: []
r_img_y: []
r_img_z: []
g_img_x: []
g_img_y: []
g_img_z: []
b_radius: [1.6784048080444336]
r_radius: []
g_radius: []
---
```

```
header:
  seq: 1435
  stamp:
    secs: 0
    nsecs:         0
  frame_id: ''
r_size: 0
b_size: 1
g_size: 0
b_img_x: [-0.4899999499320984]
b_img_y: [3.0199999809265137]
b_img_z: [0.1509999930858612]
r_img_x: []
r_img_y: []
r_img_z: []
g_img_x: []
g_img_y: []
g_img_z: []
b_radius: []
r_radius: []
g_radius: []
---
```

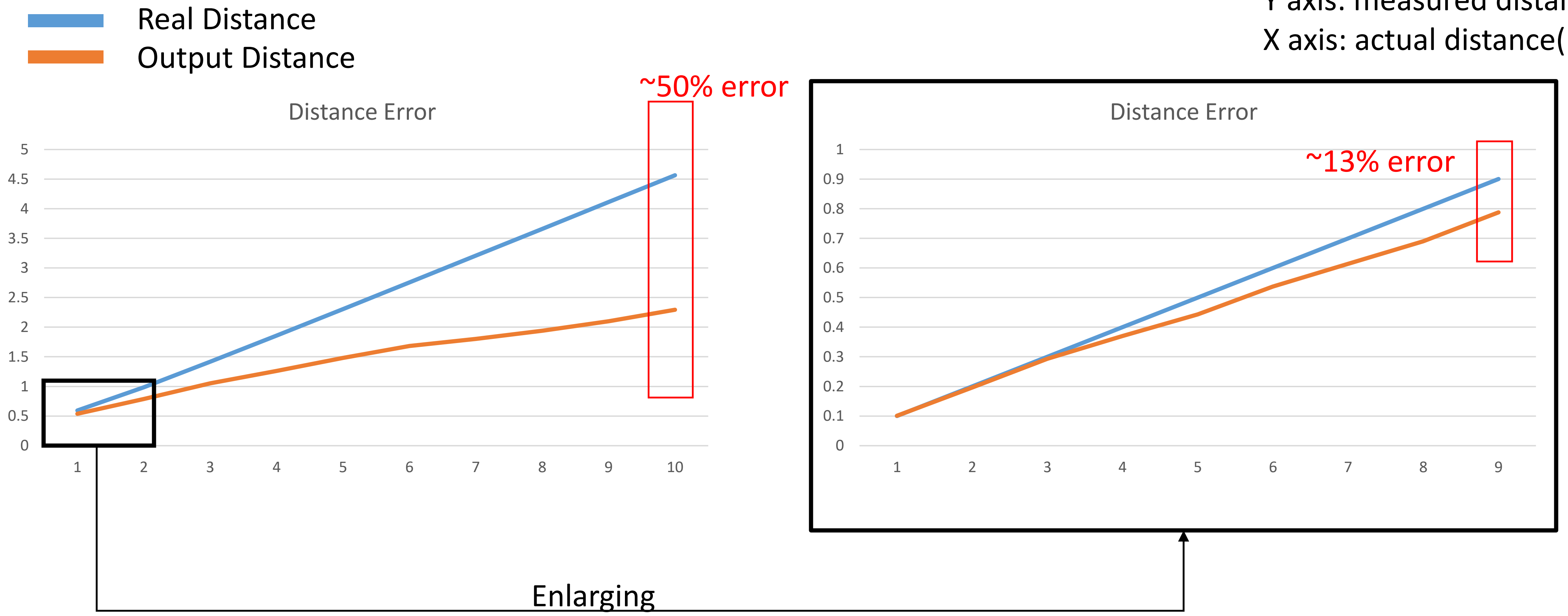
Filtered Blue ball data using **Filter_1** and **Filter_2**

Vision Processing Final result

Camera shaking does not matter!

Vision Processing Distance Calibration?

Distance Error within our range for computation can be **ignored**

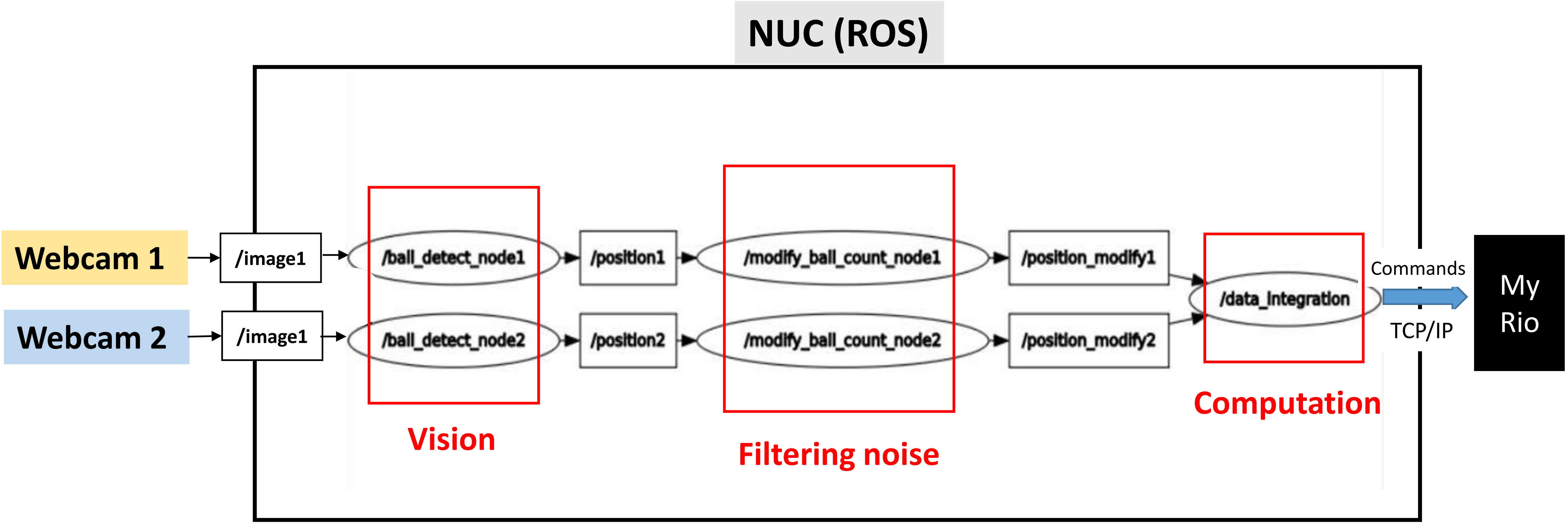


Y axis: measured distance(m)
X axis: actual distance(m)

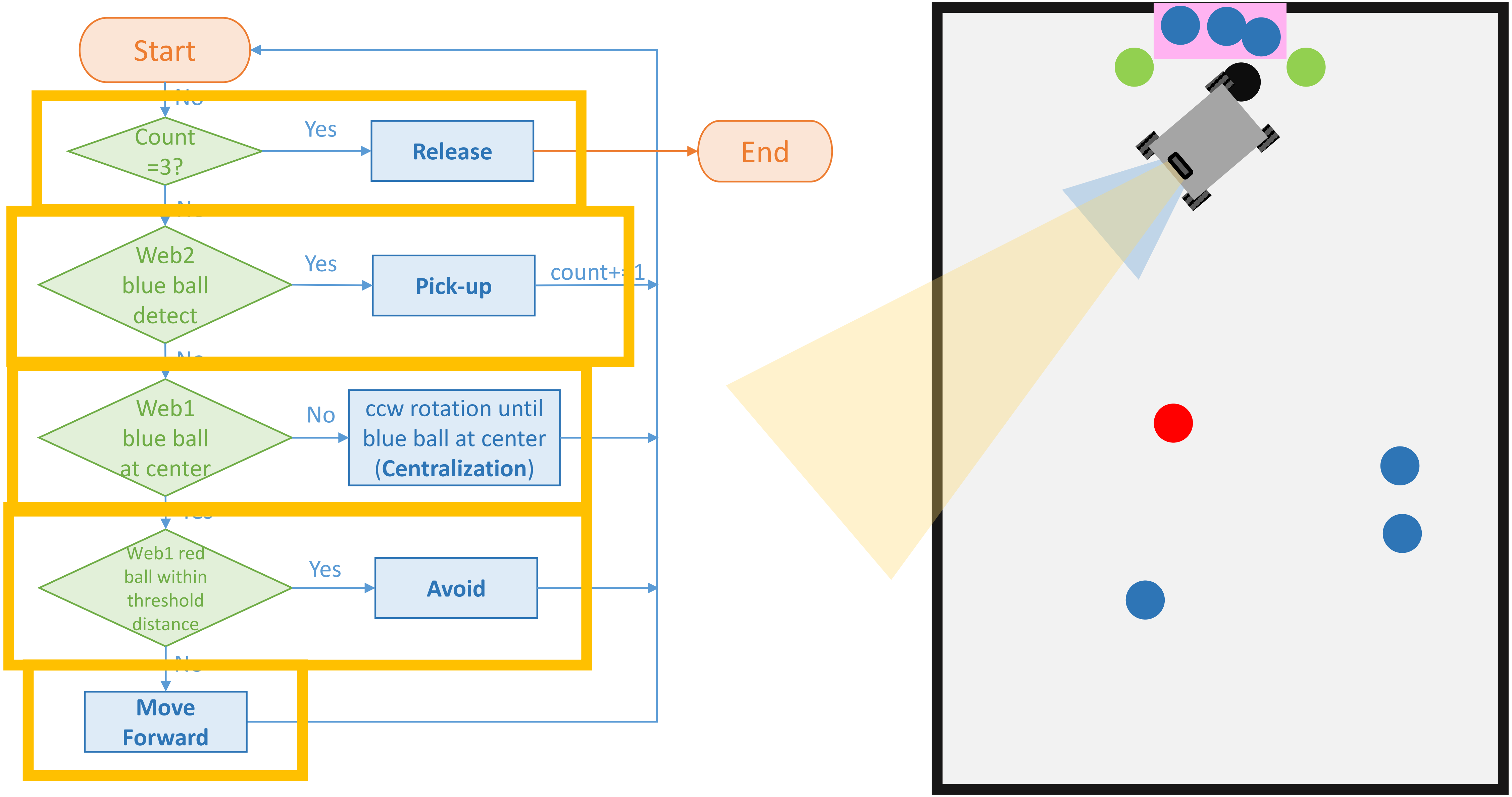
System Integration

“Case segmentation to obtain right decision in general.”

System Integration Overview



System Integration Algorithm



Start

No

Count =3?

Yes

Release

End

No

Web2 blue ball detect

Yes

Pick-up

count+=1

No

Web1 blue ball at center

No

ccw rotation until blue ball at center **(Centralization)**

Yes

Web1 red ball within threshold distance

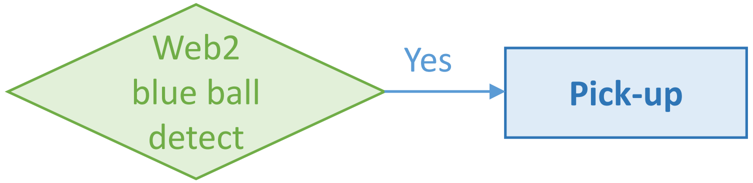
Yes

Avoid

No

Move Forward

System Integration Algorithm_Pickup



Web2 blue ball detect

Yes

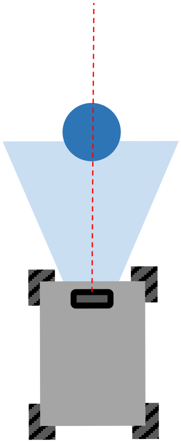
Pick-up

Pick-up

- **Web2** blue ball detection → Suction ON
- Adjust to centralize blue ball and then move forward
- **No** Blue ball detection by **web2** → Suction OFF after **1 seconds**

  Count+=1

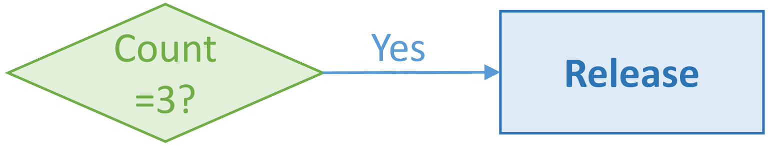
  (to ensure complete ball collection)



Debouncing ball-count algorithm

- After ball collection (count+=1), due to **signal bouncing**, it detects the blue ball **virtually** which disappears right after → **Two balls counted!!!** (count+=2)
- This is solved by **verifying the no blue ball detection after 0.7 seconds** → **Debouncing**

System Integration Algorithm_Release

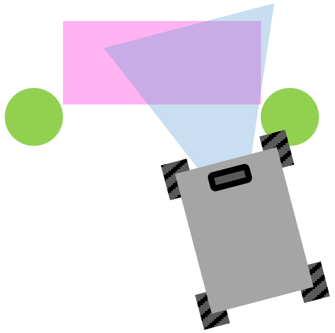


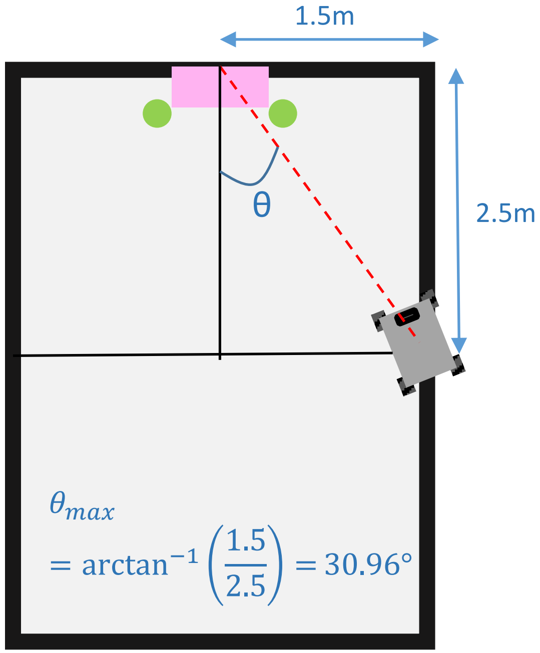
Release

- Rotation until **web1** detects green ball at center
- Move forward until green ball is just in front
- Move left(or right) until green ball is out of **web2** sight
- Open the lid

Possible releasing mechanism due to:

1. Our **path-providing-lid design**
2. Worst-case angle difference is not too much





$$\theta_{max} = \arctan^{-1}\left(\frac{1.5}{2.5}\right) = 30.96°$$

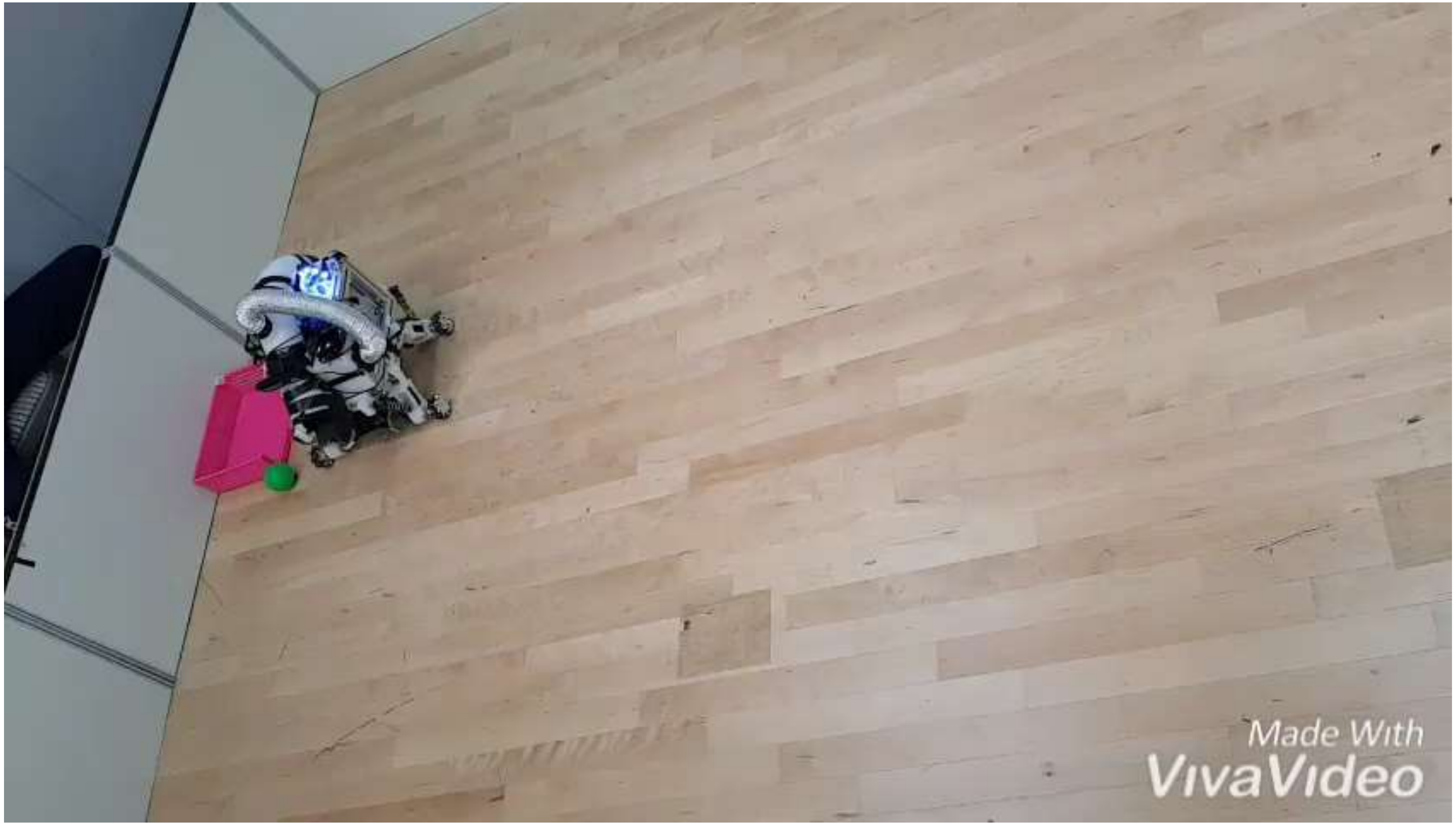
Kirby- “I am ROBUST”

Parts	Features
Hardware Design	Compact design
	Suspension added
	Pick-up module: Suction, less moving parts
Motor Control	Trapezoidal velocity profile
Vision Processing	Dual Camera
	Noise handled vision data
Software algorithm	Debouncing algorithm

Part 2

Prototype demo video

Demo Video



End

Thanks for watching!

Any questions?