

캡스톤 디자인

- Path Planning
Final

A6 Blue

Path Planning

Search-based

- Dijkstra
- A* - Dijkstra + heuristic cost
- D* - Dynamic A*

Sampling-based

- RRT - Random Tree
- RRT* - RRT + rewire

Artificial Intelligence

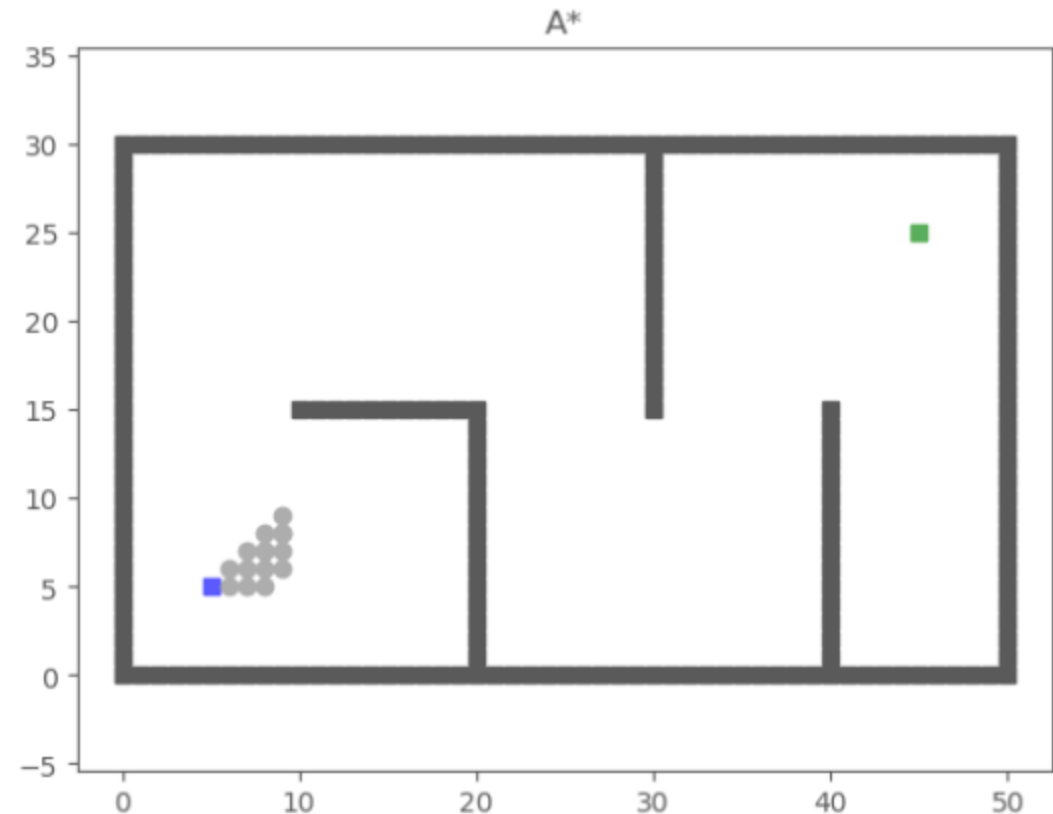
- ANN - Artificial Neural Network
- GA - Genetic Algorithm

Our Algorithm

- Design own algorithm
 - A^* , D^* 기반 : 장애물 회피
- Using python & ROS
 - Simulation
- Using drone
 - 3D path planning
 - Safety distance
 - Cost function optimization

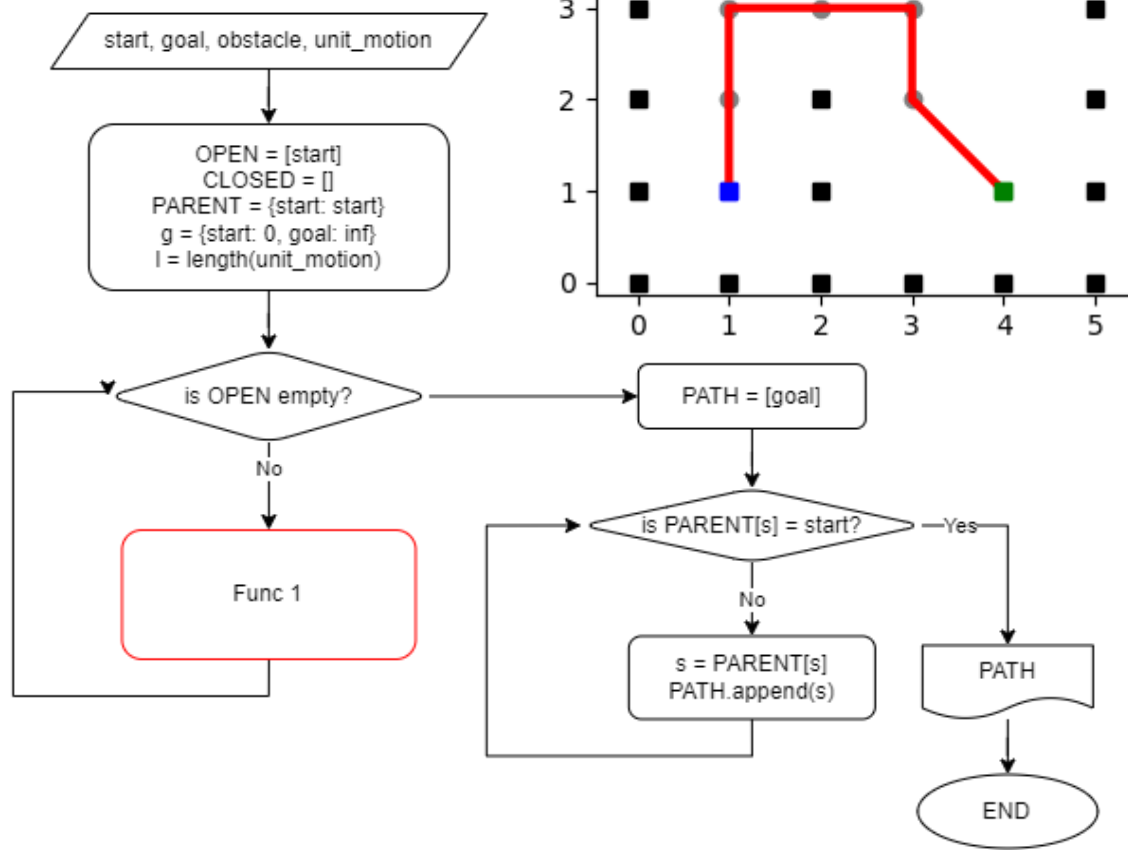
A* Algorithm

- Search-based algorithm
- Dijkstra + heuristic cost
- Cost function : $f(n) = g(n) + h(n)$
 - $g(n)$: 현재 node까지의 cost
 - $h(n)$: 현재 node부터 목표 node까지의 heuristic cost

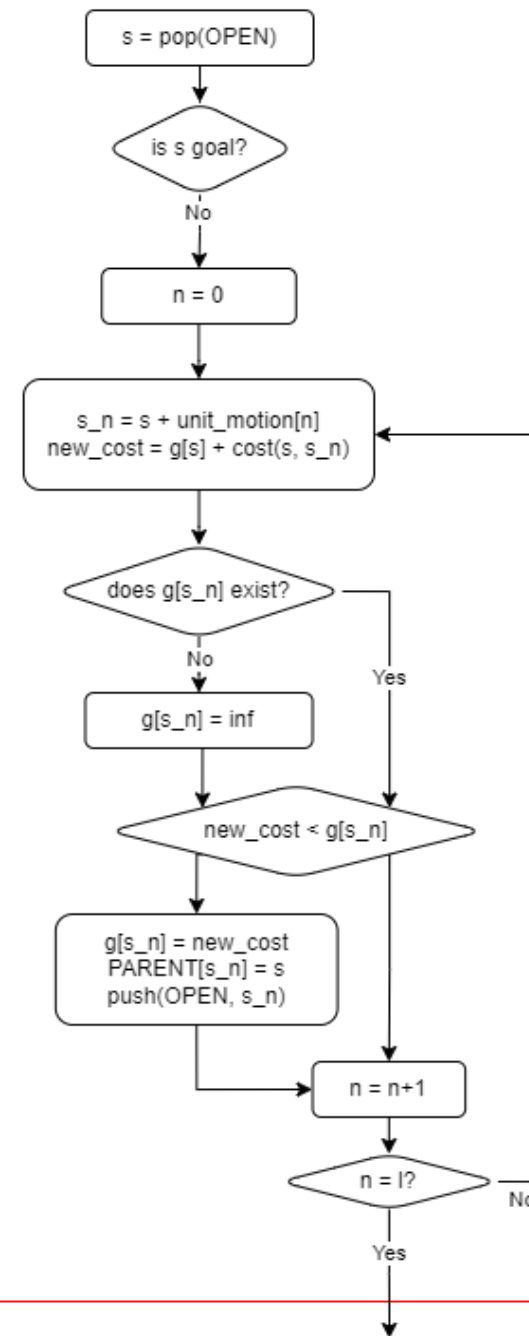


A* Algorithm

• Flowchart



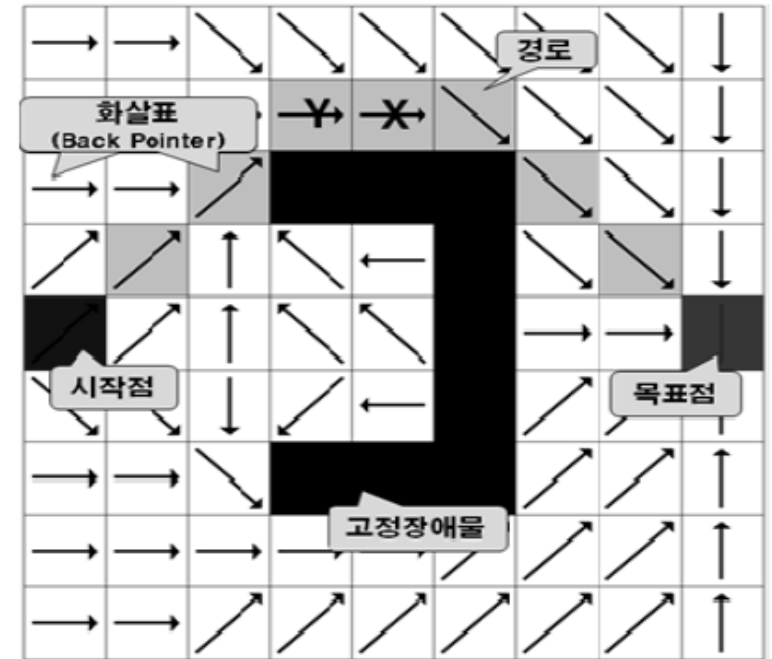
Func 1



A6 Blue

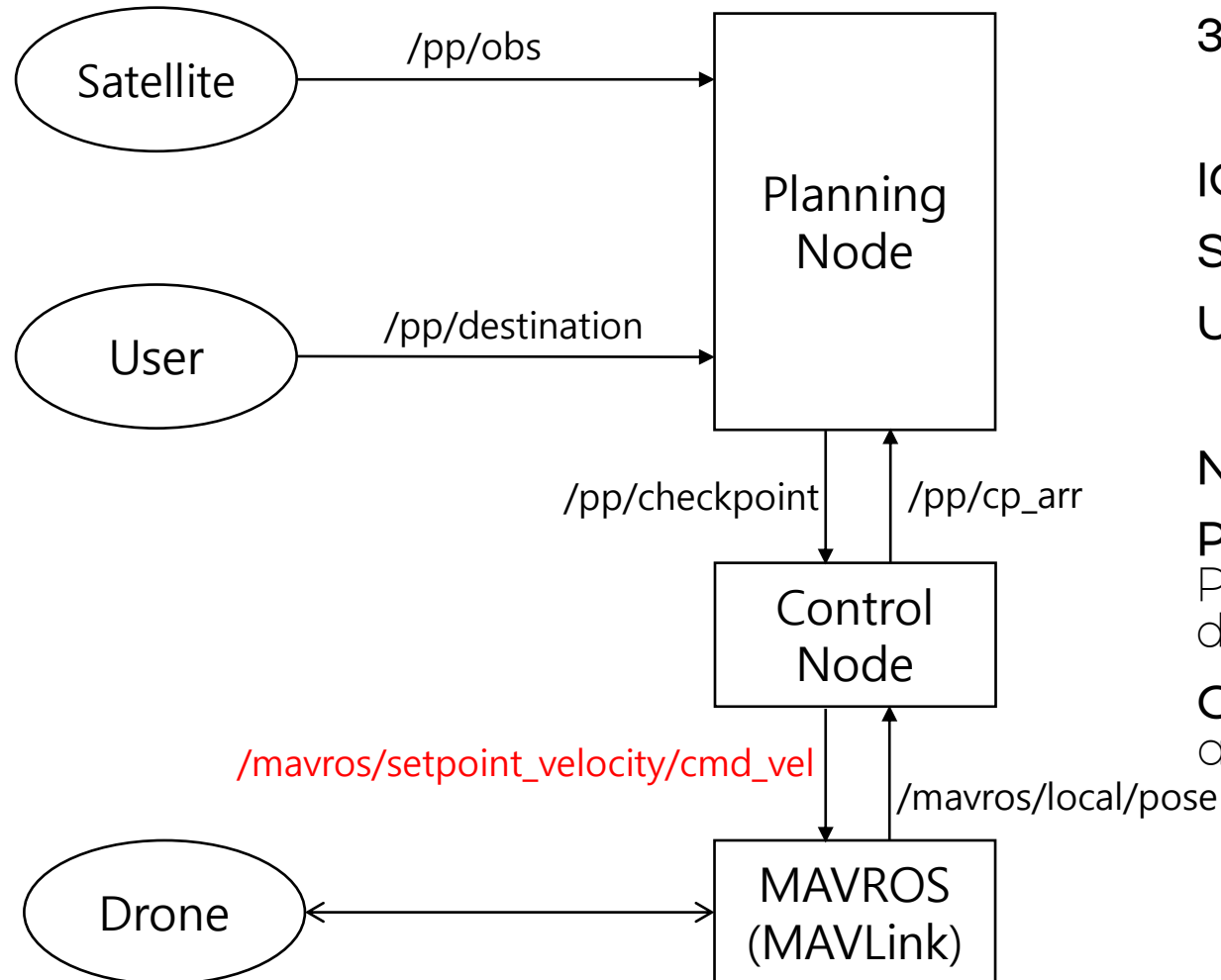
D* Algorithm

- Search-based algorithm
- Dynamic A*
- Backward searching
- Pre-mapped global map
- Bigger memory required
- Cost function : $f(n) = g(n) + h(n)$
 - $g(n)$: 현재 node까지의 cost
 - $h(n)$: 현재 node부터 목표 node
까지의 heuristic cost



Development of a Navigation Control Algorithm
for Mobile Robots Using D* Search and Fuzzy Algorithm
Yun-Ha Jung, Hyo-Woon Park, Sang-Jin Lee and Moon-Cheol Won

Schematic



3 IOs / 3 Nodes / 6 Topics

IOs:

Satellite: Detect and manage obstacles

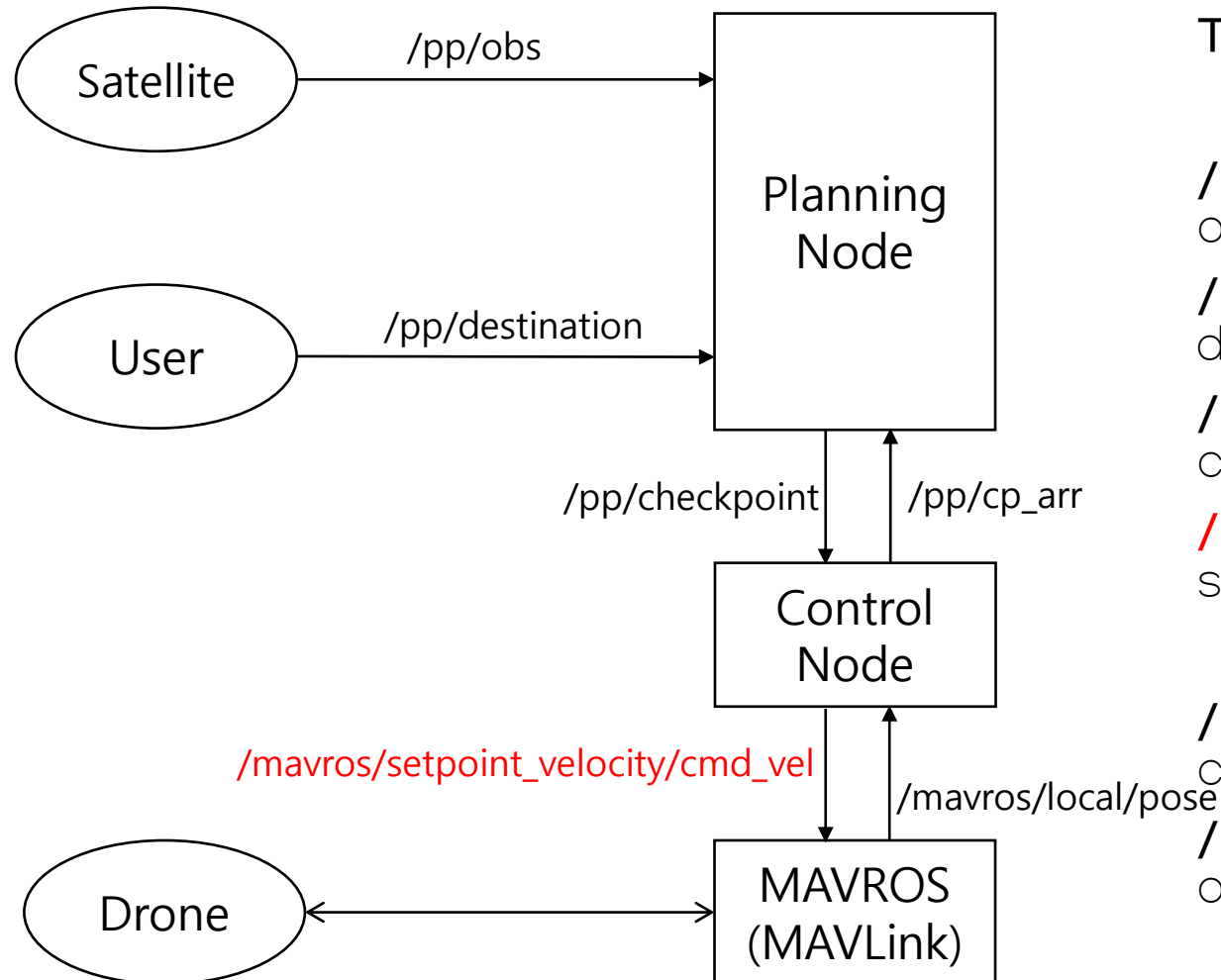
User: Set destination point

Nodes:

Planning Node: Make map with obstacle. Path planning with current position, destination and map

Control Node: Transfer control signals and arrived trigger

Schematic



Topics:

/pp/obs: contains array of xyz coordinates of obstacle

/pp/destination: contains xyz coordinates of destination

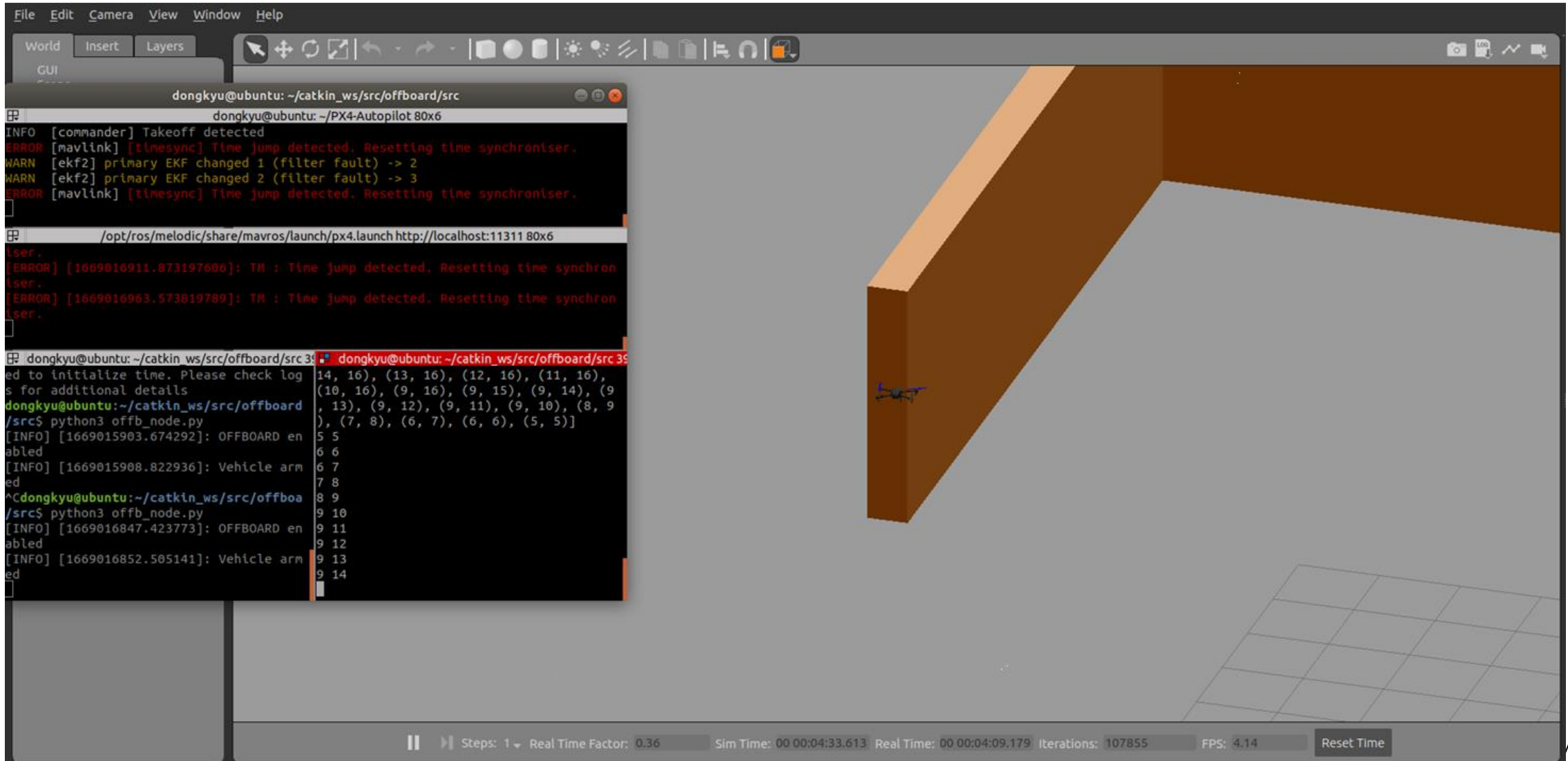
/pp/checkpoint: contains xyz coordinates of checkpoint, way to destination

/mavros/setpoint_velocity/cmd_vel: contains xyz velocity signal to checkpoint

/pp/cp_arr: contains bool if drone arrived checkpoint

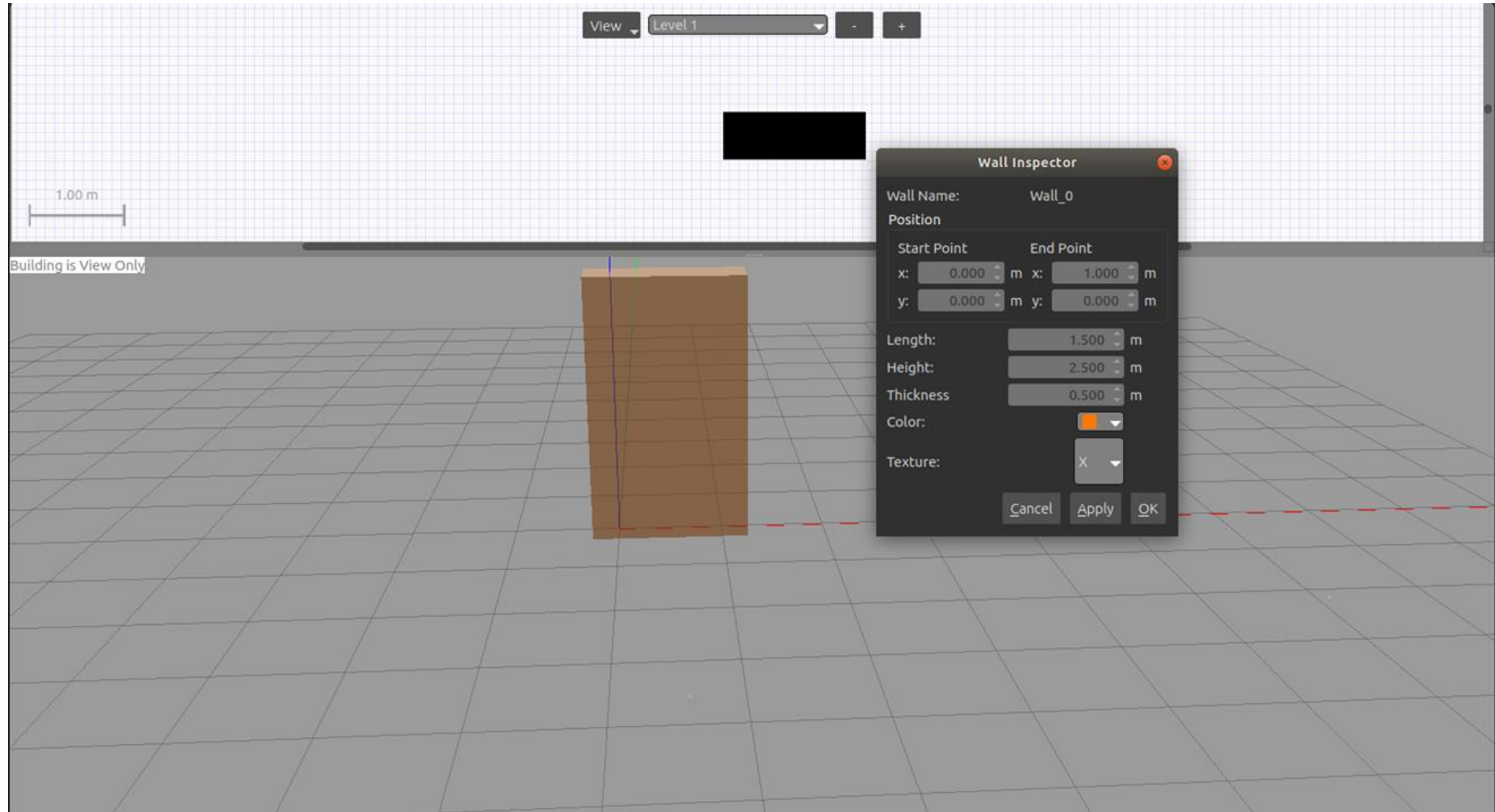
/mavros/local/pose: contains xyz coordinates of drone's local position

Simulation(problem)



Simulation

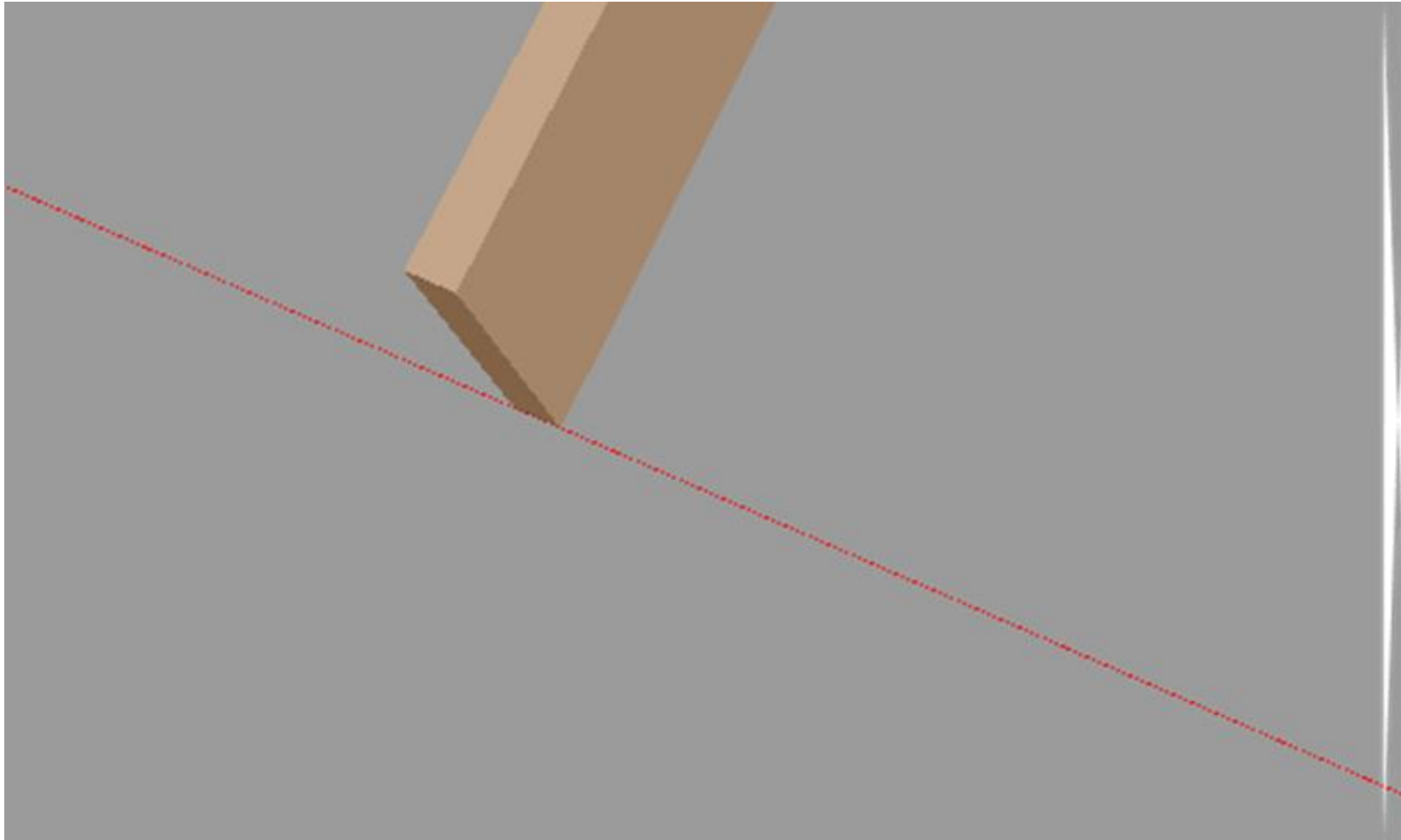
A6 Blue



2022/11/29

Simulation

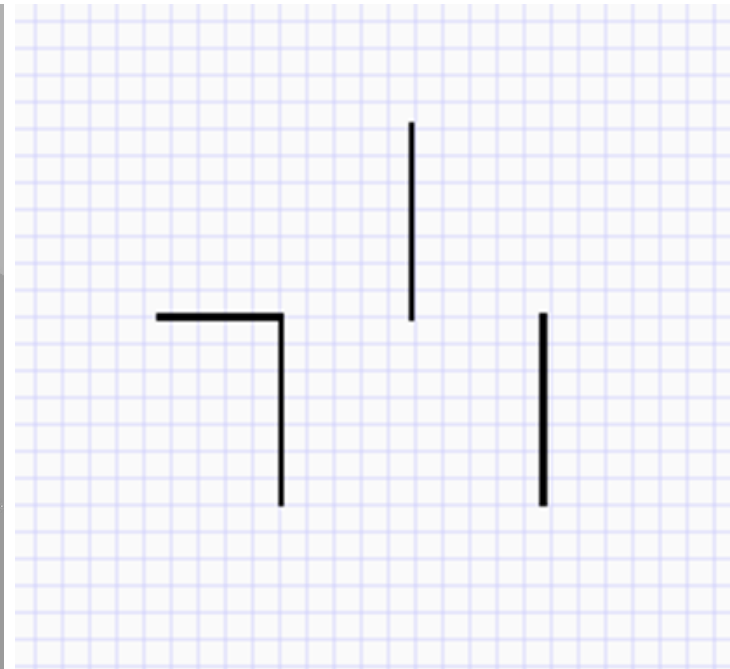
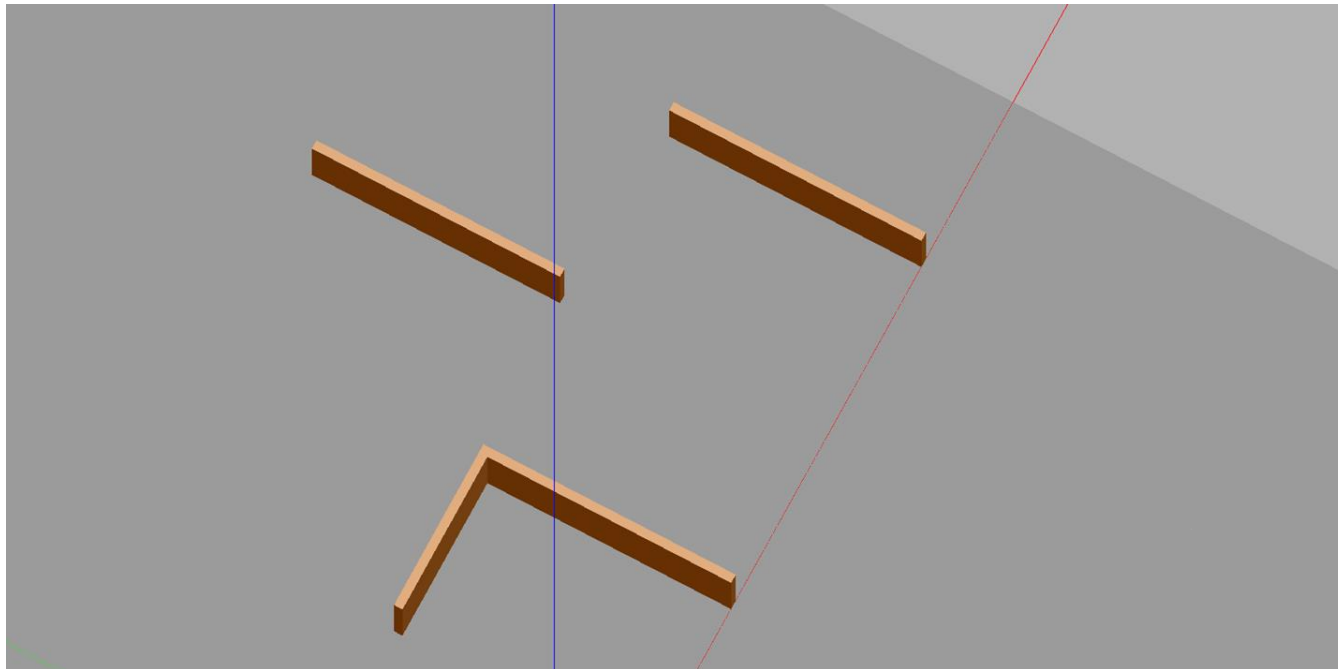
A6 Blue



2022/11/29

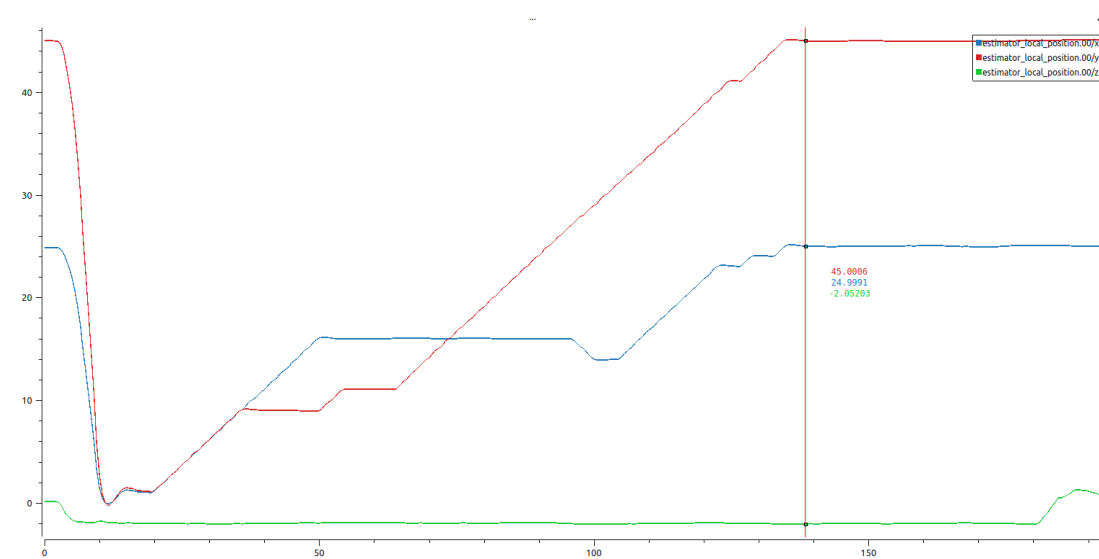
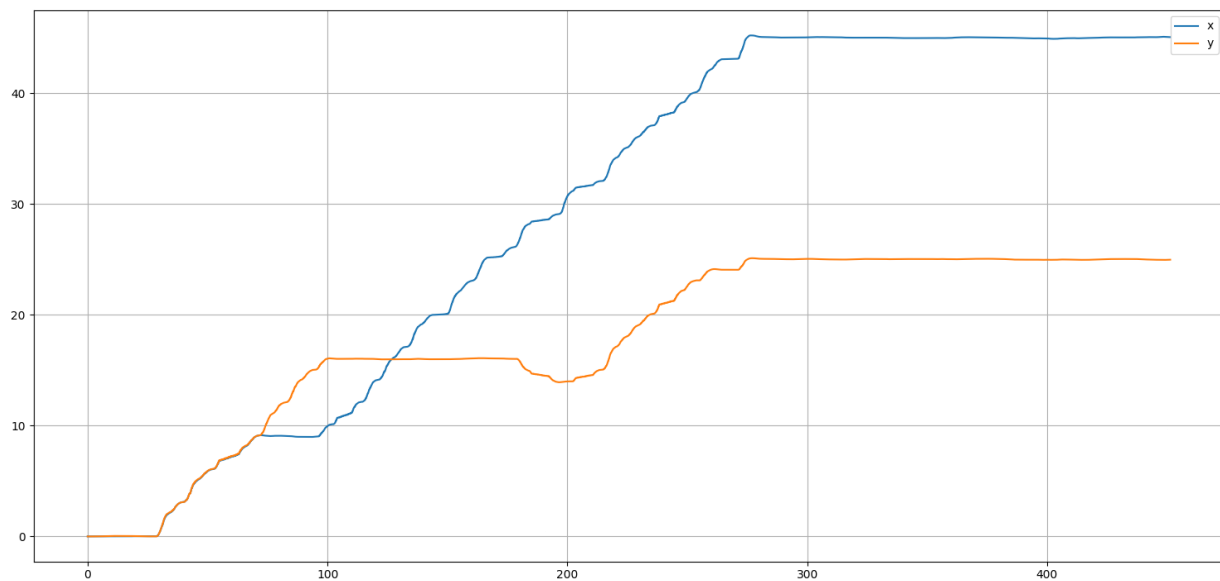
Simulation

A6 Blue

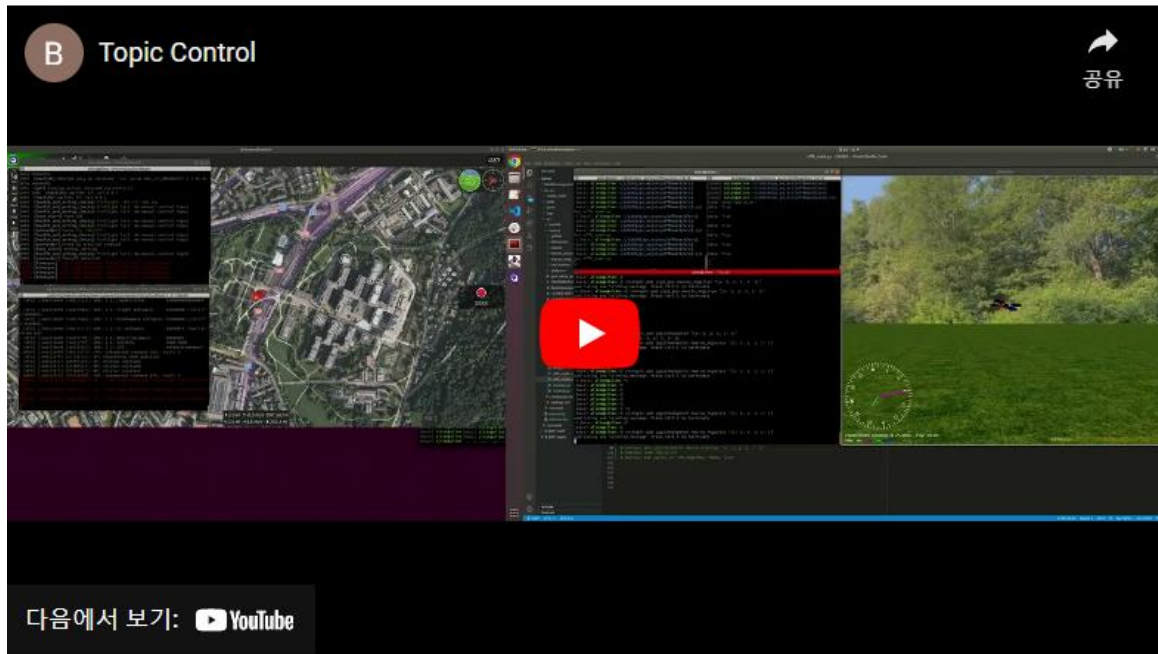


Simulation(result)

A6 Blue



Control



Achievements:

PD control implemented:

- input : checkpoint, local position
- output : arrived signal, velocity

ROS embedded:

- Works with ROS embedded planning node
- input : /mavros/local/pose, /pp/checkpoint
- output : /mavros/setpoint_velocity/cmd_vel, /pp/cp_arr

Drone controllable via planning node

Sends /pp/cp_arr

- True if distance between checkpoint and current position is under particular distance

Prevent overshooting

- PD-control with x, y velocity

Control(Overshooting)

Solutions:

1. Step position input(default)
2. Sigmoid position input
3. Velocity control, P-control
4. Velocity control, PD-control

Keypoints:

1. Faster response
2. Reduce overshoot ($\approx 0\%$)
3. Reduce stuttering
4. Minimize acceleration

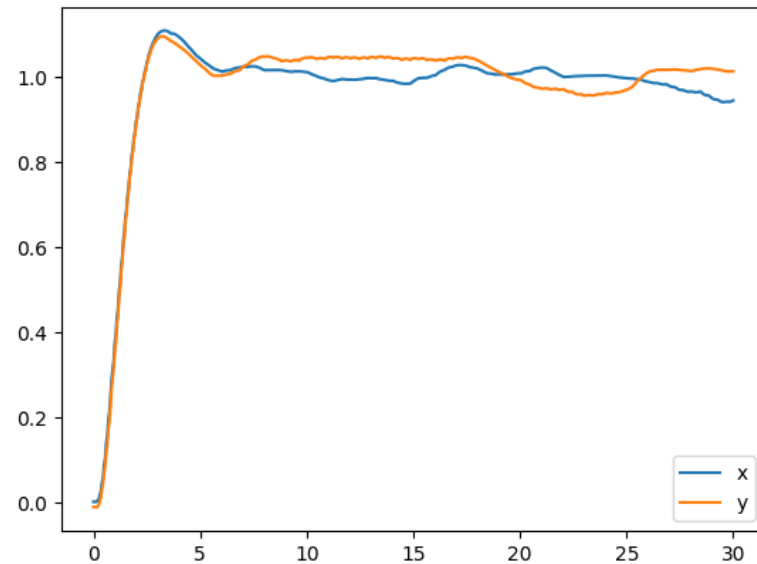
Control (Overshooting - default step position input)

1. Step position input(default)

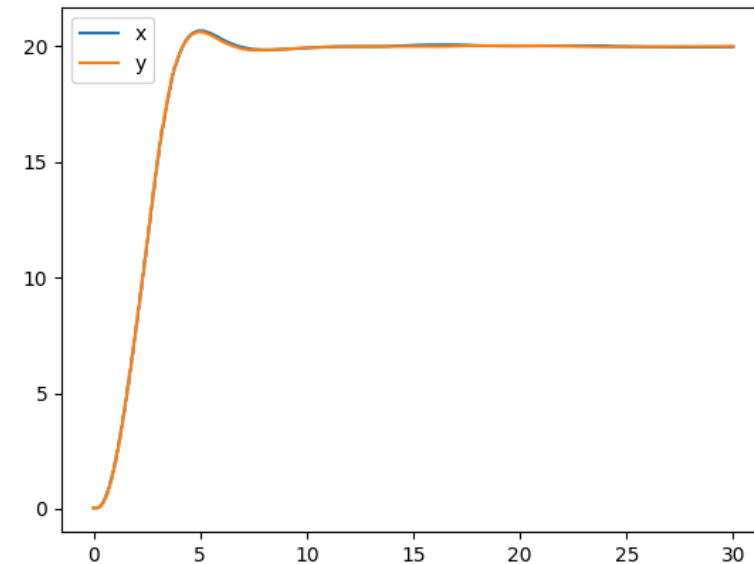
About 15% overshoot occurs in unit step input.

About 10% overshoot occurs in 20 step input.

Settling time: about 7s



Step input, x, y=1

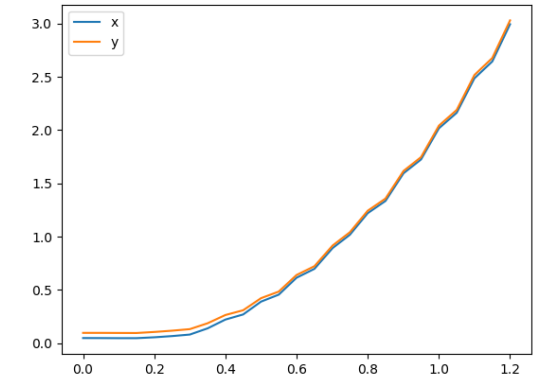
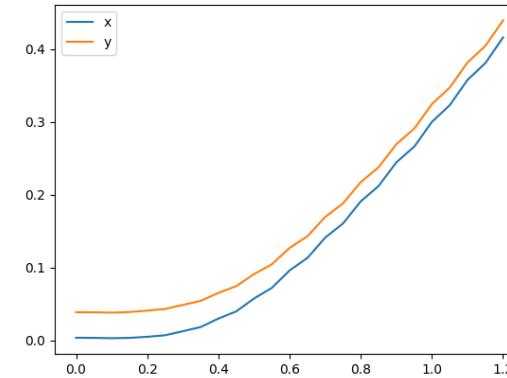
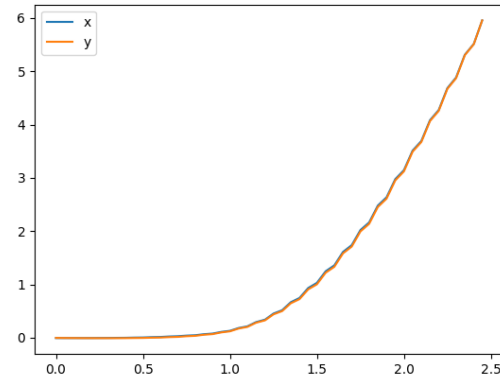
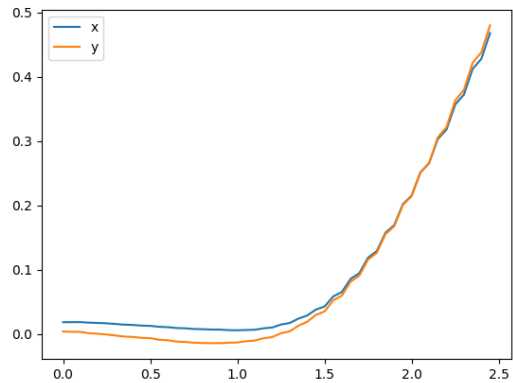


Step input, x, y=20

Control (Overshooting - sigmoid position input)

2. Sigmoid position input(logistic function)

Response time with logistic position input is too long



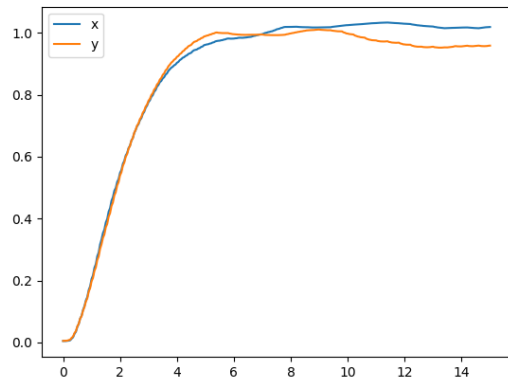
Control (Overshooting - P-controlled velocity)

3. Velocity control, P-control

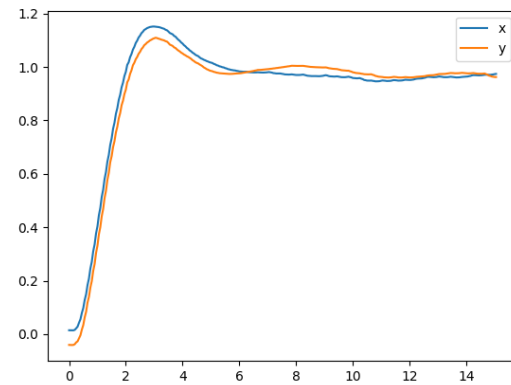
No overshoot occurs with $p=0.5$

About 15% overshoot occurs with $p=1$

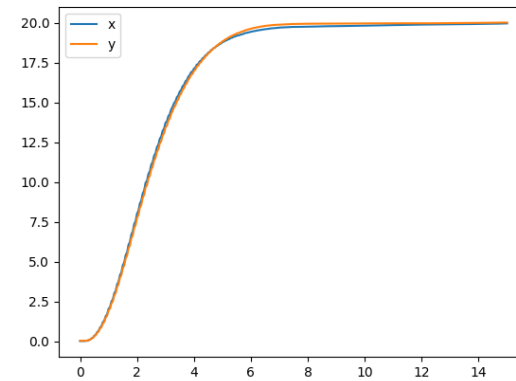
Settling time = 5~6s on both $p=0.5$ and $p=1$



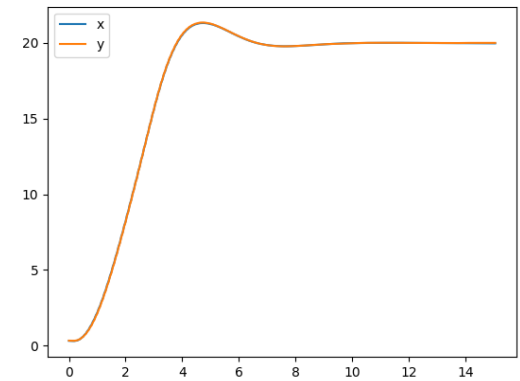
$P=0.5, x, y=1$



$P=1, x, y=1$



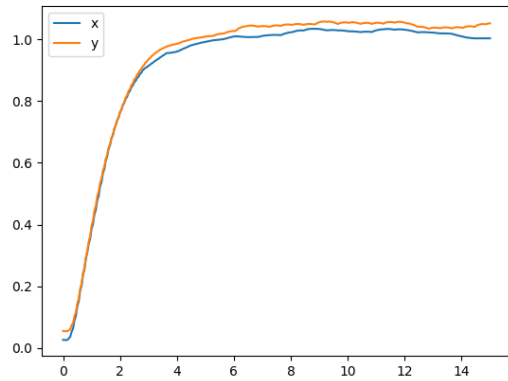
$P=0.5, x, y=20$



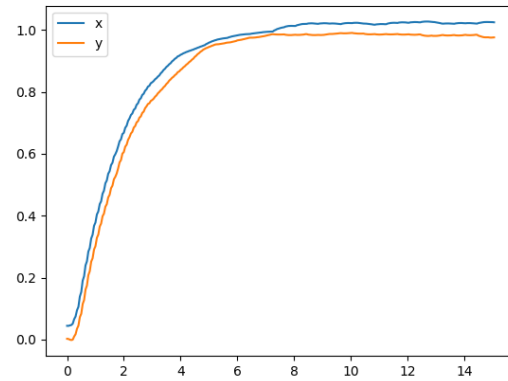
$P=1, x, y=20$

Control (Overshooting - PD-controlled velocity)

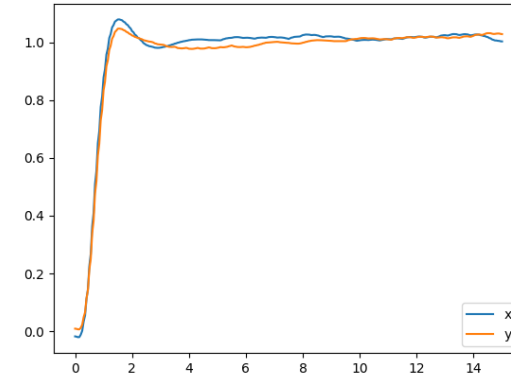
4. Velocity control, PD-control



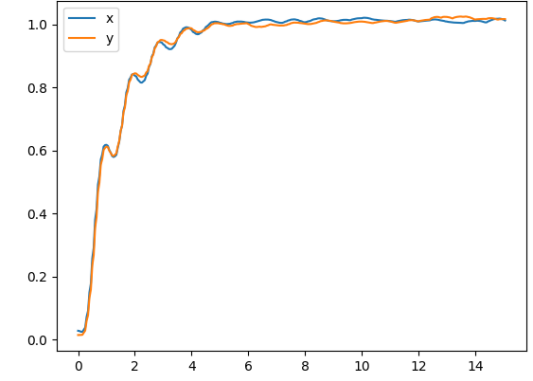
$P=1, D=200, (1)$



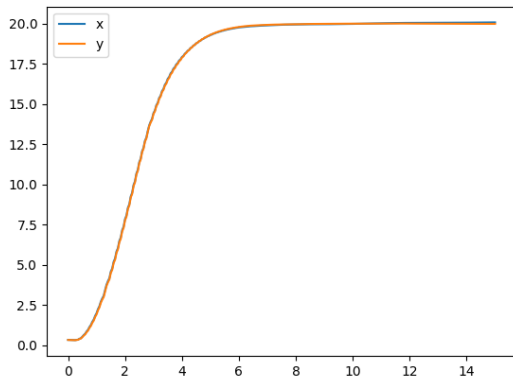
$P=1, D=400, (1)$



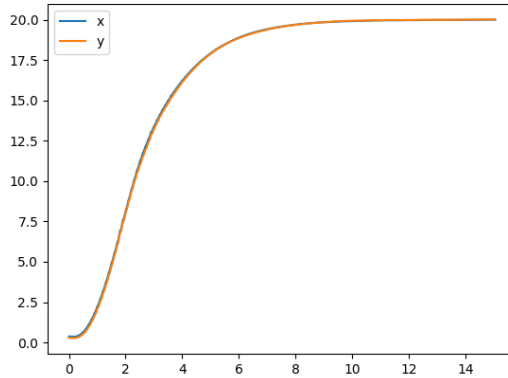
$P=3, D=400, (1)$



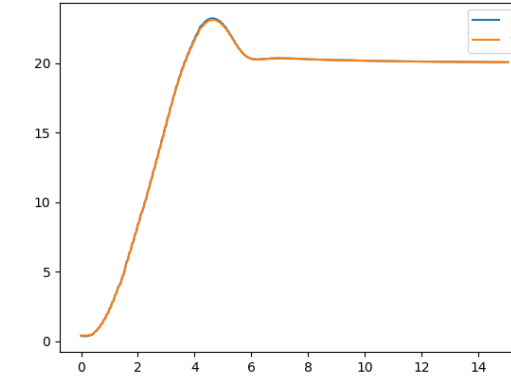
$P=3, D=1000, (1)$



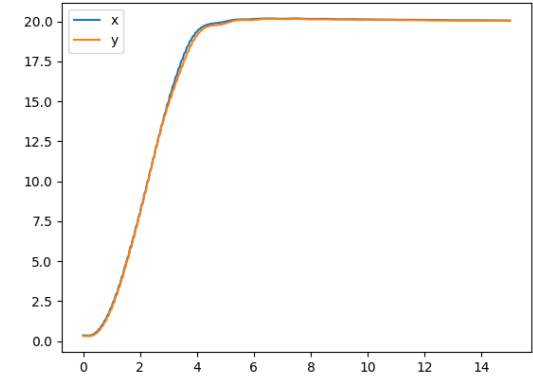
$P=1, D=200, (20)$



$P=1, D=400, (20)$



$P=3, D=400, (20)$



$P=3, D=1000, (20)$

Control (Overshooting - PD-controlled velocity)

4. Velocity control, PD-control

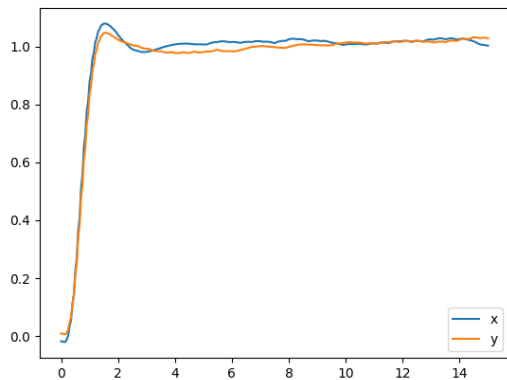
No overshoot occurs with $p=1$

Faster settling time and response time with $p=3$

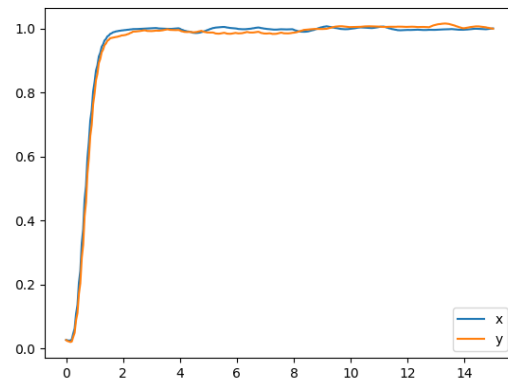
Stuttering occurs with large $d(=1000)$ for small distance

=> $p=3, d=1000$ shows best performance for large distance

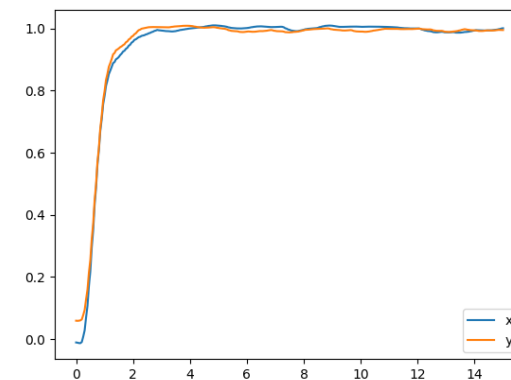
-> find optimal d parameters with $p=3$ for small distance



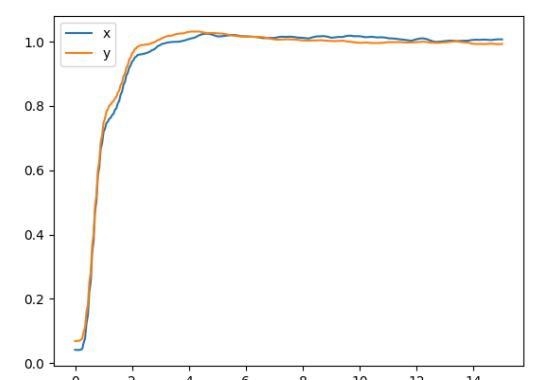
$P=3, D=400, (1)$



$P=3, D=500, (1)$



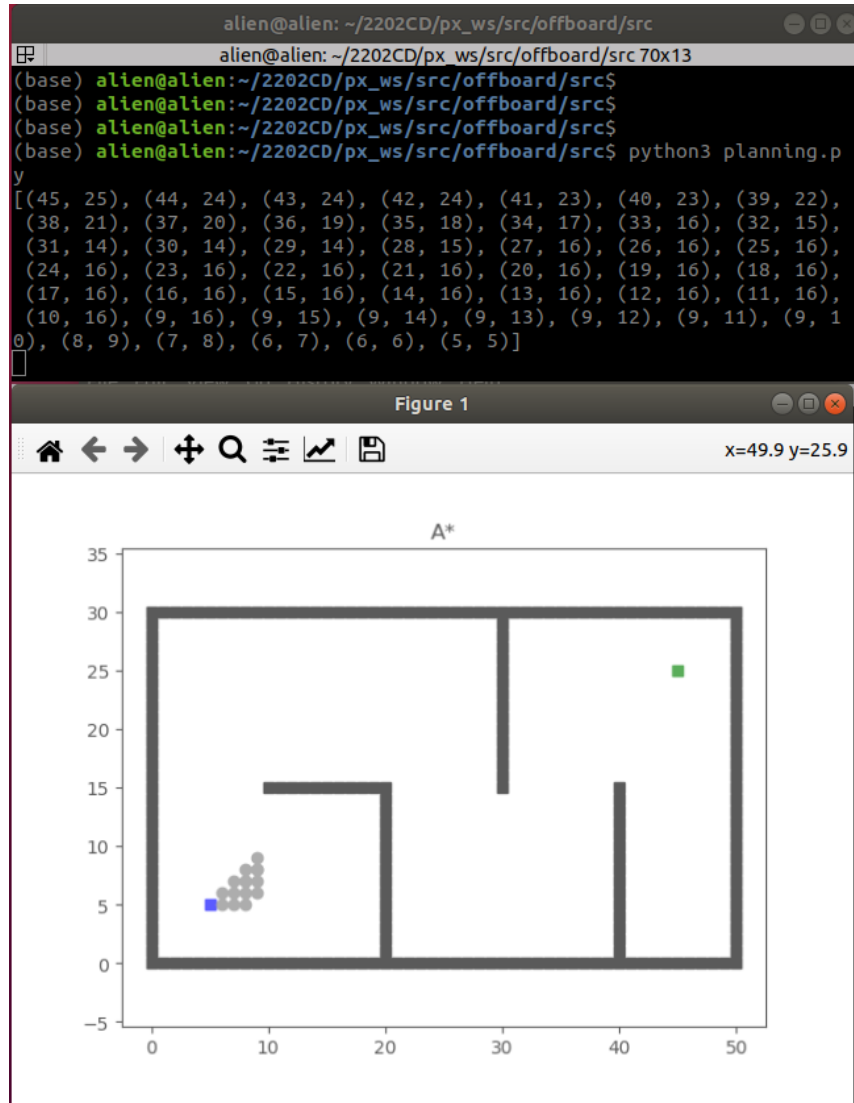
$P=3, D=600, (1)$



$P=3, D=700, (1)$

=> $p=3, d=500$ shows best performance for large distance

Planning



Achievements:

A* algorithm implemented

- input : map, start, destination
- output : route to destination

ROS embedded

- input : /pp/map, /pp/destination, /pp/cp_arr, /pp/obs
- output : /pp/checkpoint

Dynamic mapping(continuous A*)

- Update map with A* when obstacle detected

Path simplification

- Planning node simplifies the path along same direction

D* algorithm implemented

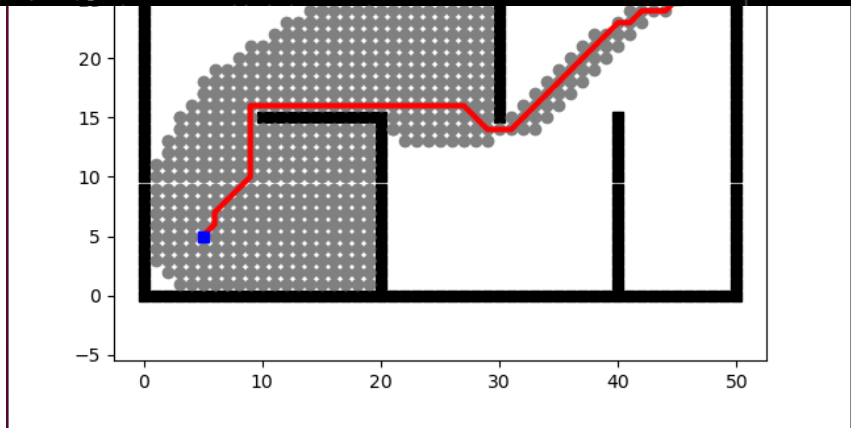
- input : map(+obstacles), start, destination
- output : route map to destination

Planning^(Dynamic)

```

blu@blu-leo: ~
blu@blu-leo: ~ 81x26
blu@blu-leo:~$ pland
[(1, 1), (1, 1), (1, 1), (1, 1), (1, 1), (1, 1), (1, 1), (1, 1), (9, 9), (9, 9),
(9, 9), (9, 9), (9, 9), (9, 9), (9, 9), (9, 16), (9, 16), (9, 16), (9, 16), (9, 1
6), (9, 16), (9, 16), (9, 16), (9, 16), (9, 16), (9, 16), (21, 16), (21,
16), (23, 14), (23, 14), (23, 14), (23, 14), (23, 14), (23, 14), (23, 14), (23,
14), (31, 14), (31, 14), (31, 14), (31, 14), (31, 14), (31, 14), (31, 14), (31, 1
4), (31, 14), (31, 14), (31, 14), (42, 25), (42, 25), (42, 25), (45, 25)]
[INFO] [1669710411.873288]: Obstacle added (9, 9)
Add obstacle at: x = 9, y = 9
new path >>
[(1, 1), (1, 1), (1, 1), (1, 1), (1, 1), (1, 1), (1, 1), (1, 1), (8, 8), (8, 8), (8, 10),
(9, 11), (9, 11), (9, 11), (9, 11), (9, 11), (9, 16), (9, 16), (9, 16), (9, 16),
(9, 16), (9, 16), (9, 16), (9, 16), (9, 16), (9, 16), (9, 16), (21, 16), (21, 16),
(23, 14), (23, 14), (23, 14), (23, 14), (23, 14), (23, 14), (23, 14), (23, 14),
(23, 14), (31, 14), (31, 14), (31, 14), (31, 14), (31, 14), (31, 14), (31, 14),
(31, 14), (31, 14), (31, 14), (31, 14), (42, 25), (42, 25), (42, 25), (45, 25)]
[INFO] [1669710435.427864]: Obstacle added (9, 16)
Add obstacle at: x = 9, y = 16
new path >>
[(1, 1), (1, 1), (1, 1), (1, 1), (1, 1), (1, 1), (1, 1), (1, 1), (8, 8), (8, 8), (8, 10),
(9, 11), (9, 11), (9, 11), (9, 14), (8, 15), (8, 15), (8, 17), (8, 17), (10, 17),
(11, 16), (11, 16), (11, 16), (11, 16), (11, 16), (11, 16), (11, 16), (11, 16),
(11, 16), (11, 16), (21, 16), (21, 16), (23, 14), (23, 14), (23, 14), (23, 14),
(23, 14), (23, 14), (23, 14), (23, 14), (31, 14), (31, 14), (31, 14), (31, 14),
(31, 14), (31, 14), (31, 14), (31, 14), (31, 14), (31, 14), (42, 25), (42, 25), (4
2, 25), (42, 25), (45, 25)]

```



events:

algorithm implemented

input : map, start, destination

output : route to destination

embedded

input : /pp/map, /pp/destination,
/pp/cp_arr, /pp/obs

output : /pp/checkpoint

dynamic mapping(continuous A*)

update map with A* when obstacle
detected

Path simplification

- Planning node simplifies the path along same direction

D* algorithm implemented

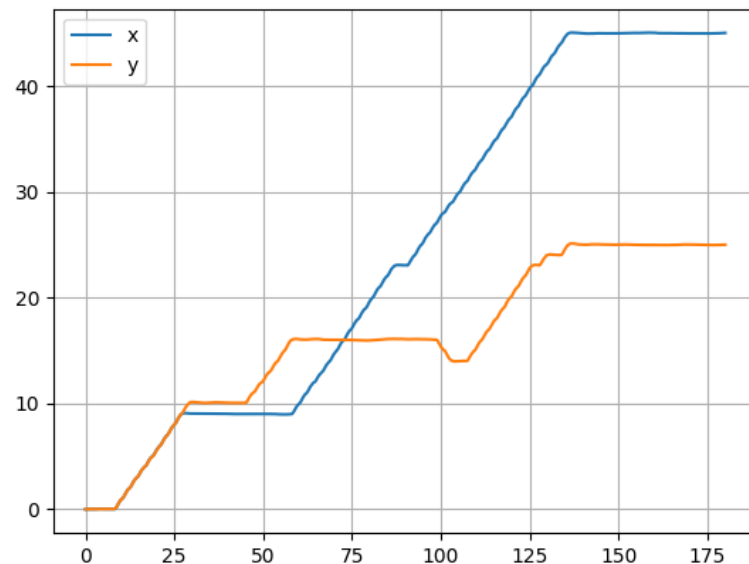
- input : map(+obstacles), start, destination
- output : route map to destination

Planning (Path simplification)

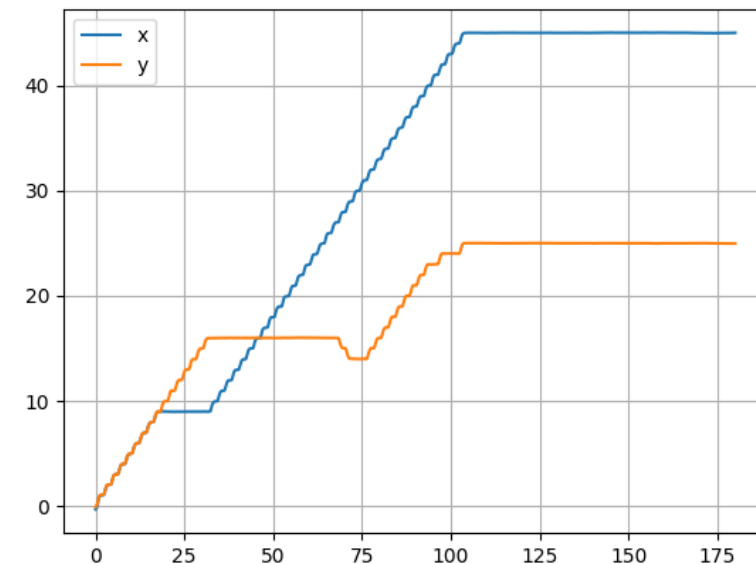
Why is path simplification required?

To prevent overshoot and to improve response time, PD-control applied.

But path with unit length(1 or 1.41) made drone continuously stops on each checkpoints.



Step input(default)



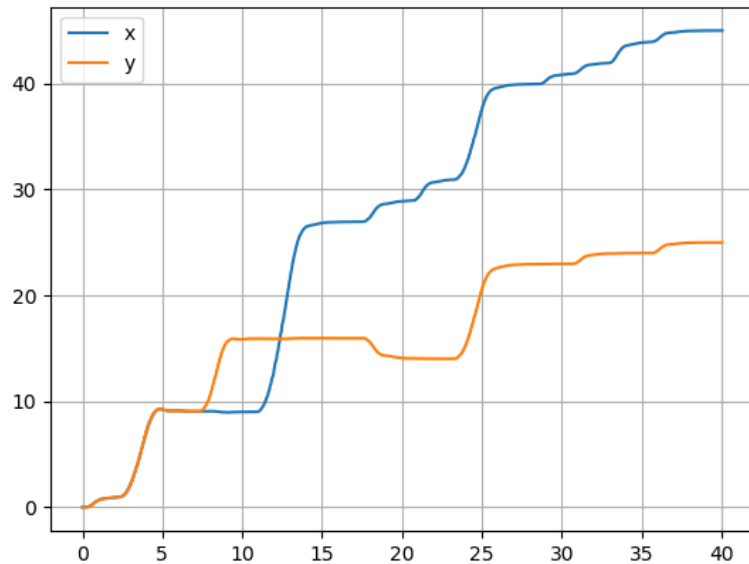
PD-control

-> **Response time shortened with PD-control:** 130s -> 110s

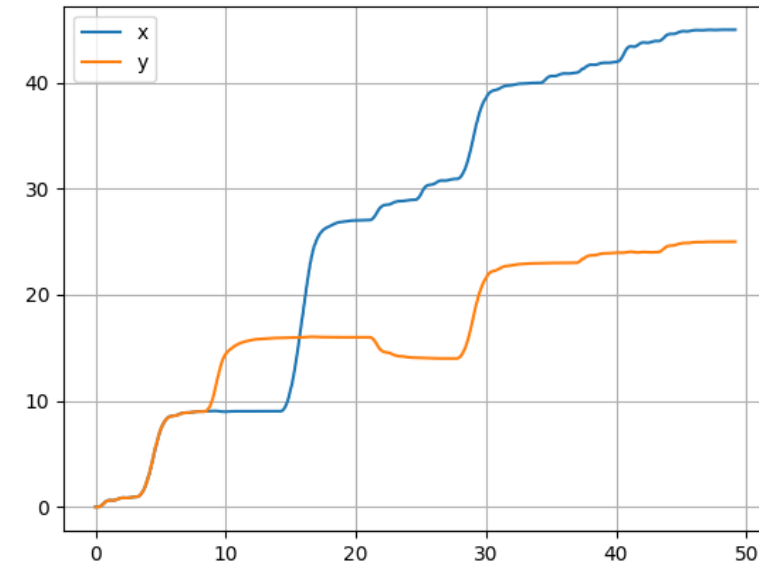
--> **Apply path simplification:** Planning node simplifies the path along same direction

Planning (Path simplification)

Path simplification made checkpoints with large distance; PD parameters had to be changed
 $P=3, D=700$ / $P=3, D=1000$ showed best performances.



$P=3, D=700$ (simple)
(video)



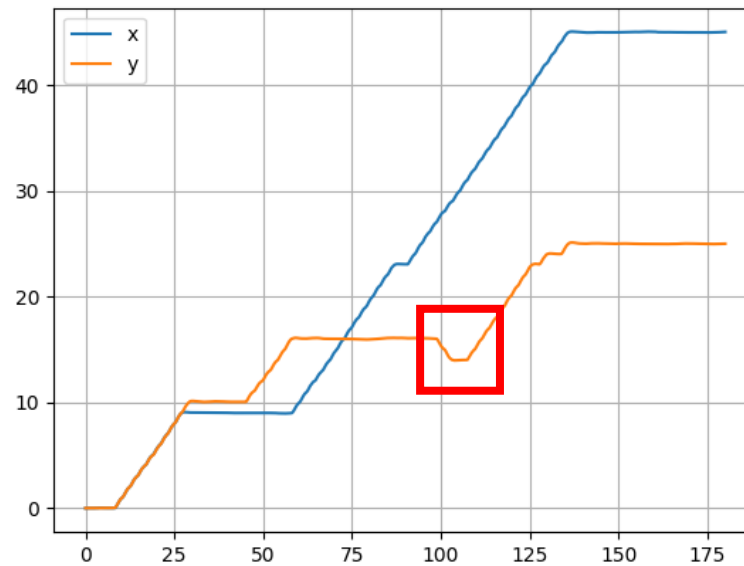
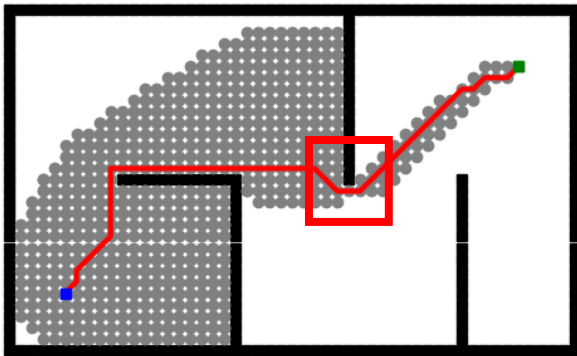
$P=3, D=1000$ (simple)
(video)

=> Response time shortened with simplification: 110s->37s, 45s

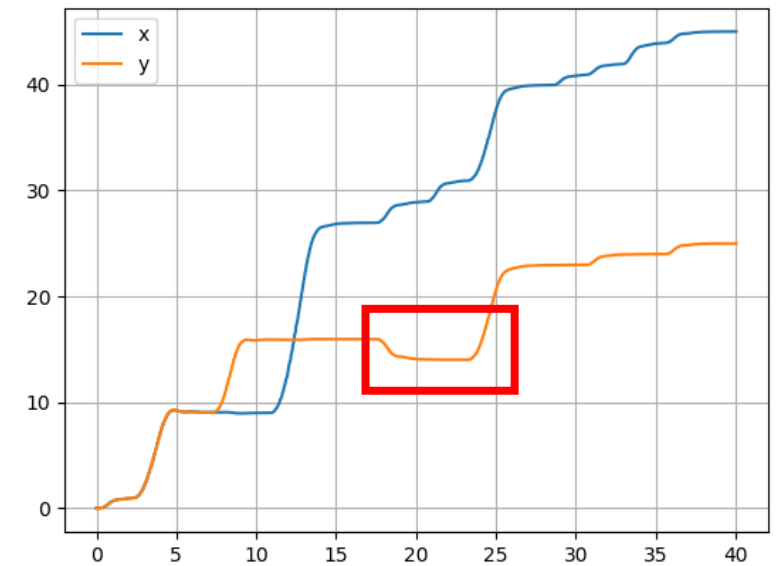
=> No stop-and-go occurred.

Planning (Path simplification - comparison)

Comparison between step and PD simplified control.



Step input(default)



P=3, D=700(simple)

Linear motion is faster with PD-simplified, zig-zag motion is faster with step position control. .

Demo

A6 Blue

2022/11/29