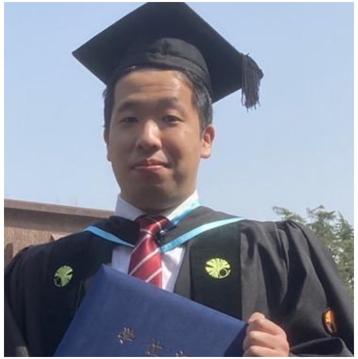
Methods In DDC

Haruki Kozuka
JSK robotics

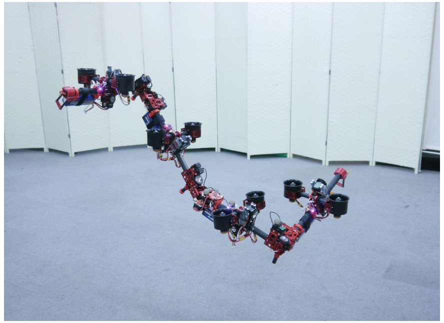
東京大学
THE UNIVERSITY OF TOKYO

Aerial Robot Team

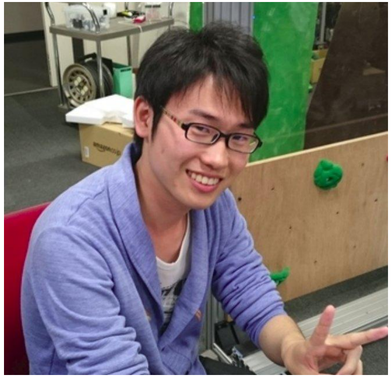
Who we are



Haruki Kozuka
Master student



Moju Zhao
Dragons' guy



Kento Kawaharazuka
Assistant Professor

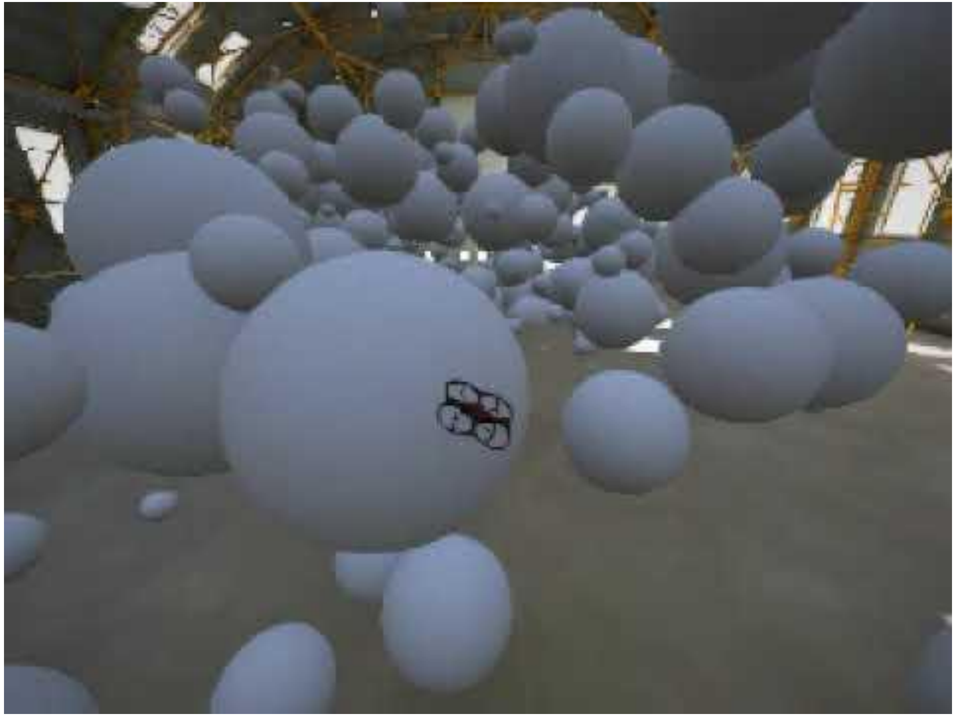
JSK
Aerial Robot Team

東京大学
THE UNIVERSITY OF TOKYO

JSK robotics Lab,
The university of Tokyo

Table Of Contents

- What is DDC?
- Initial Settings
- Submission Code
  - Main Changes in our methods
    - Polar voxel
  - Results
- Vision Based approach
- Implement to real quadrotor



Basic Information of DDC

ROBOTICS & PERCEPTION GROUP

- Drone navigation competition
- Objective: Reach Goal (x > 60) w/o clashing obstacles
    - Success rate, goal time
- Benchmark for autonomous drone navigation
- 3 learning level (easy,medium,hard)

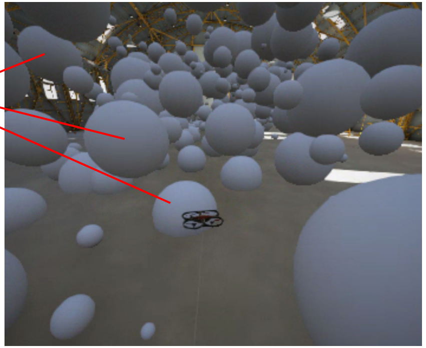
Champion at state-based category

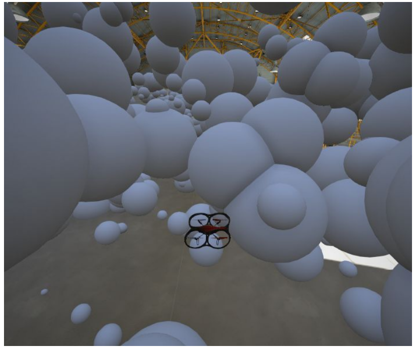

ICRA2022
@ieee_ras_icra

Congrats to the winners of the DodgeDrone Challenge - Yvo Keuter from TU Delft and Haruki Kozuka from University of Tokyo! Learn more about the competition in

Twitter: ICRA 2022

Obstacle


medium


hard

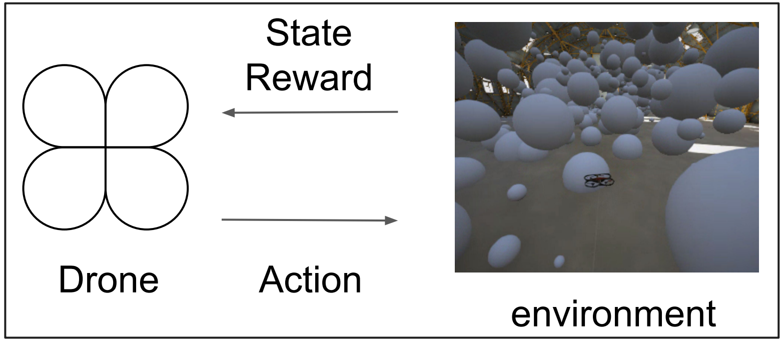
Initial RL Setting offered by the DDC organizers

Our method is Reinforcement Learning(RL) based approach

State	dimension
10 obstacles' Center & size	40 (=10*(3+1))
desired velocity	3
orientation, velocity	9+3
sum	55

Action	dimension
Bodyrate(ω)	3
full thrust	1
sum	4

Reward
survive reword
collision penalty
velocity penalty (linear&angular)



Main changes from Initial Code

State:

- Obstacle information: polar voxel

Reward:

- reward by moving x direction
- penalty when approaching boundary
- eliminate survive reward
    - disturb moving x direction (stay initial position)

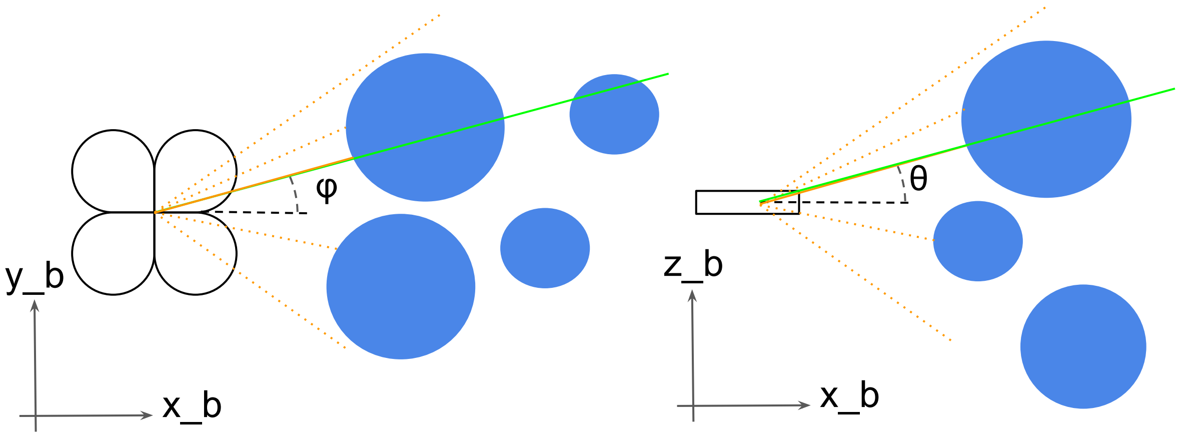
learn only in **medium** environment:

- learn in **hard** -> move slowly

State	dimension
Obstacle (Polar voxel)	64 (= 8*8)
distance from boundary(y,z)	2+2
position,attitude, (linear angular) velocity	3+3+9+3
goal velocity	3
Sum	89

Reward
x move reward
boundary penalty
collision penalty

Create Polar Voxels



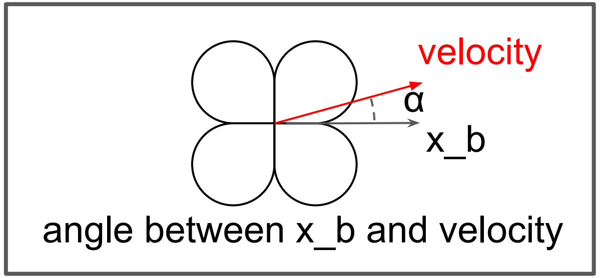
Algorithm

1. set direction (φ, θ)
2. extend straight line in 1's direction
3. calculate distance from closest obstacle
4. do 1~3 in different direction

φ,θ's characteristic

range	-45deg ~ 45 deg
Cuts	8
Output dimension	8*8 = 64

Create Polar voxels



Why θ,φ is -45deg ~ 45deg?

- In most flight, |α|< 45deg
- Quadrotor needs only obstacles' info in velocity's direction if obstacle is static

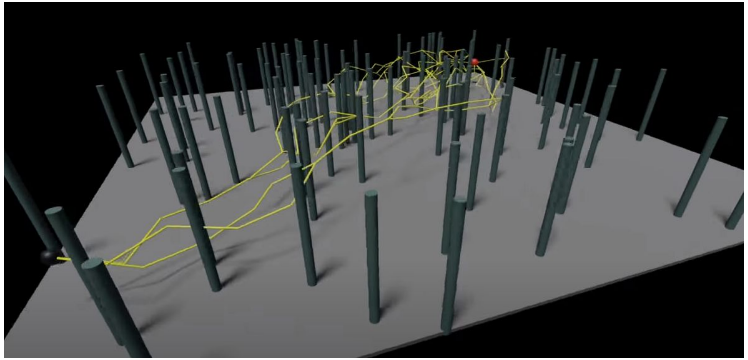
-> Not think about other obstacles' info

interpretation of our method

Check various direction's obstacle

≃ think about desired moving direction

(Cf. topological path search)



Robert Penicka et al. "Minimum-Time Quadrotor Waypoint Flight in Cluttered Environments" (2022)

Flying Result (submission policy)

Test 100 times

medium

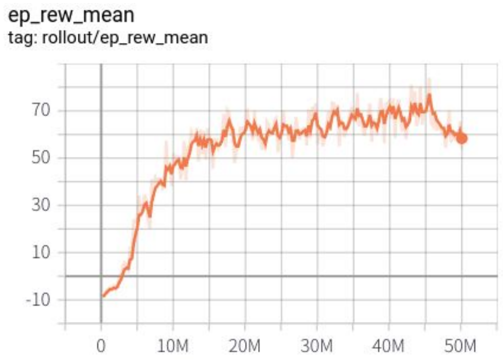
Goal rate: 77

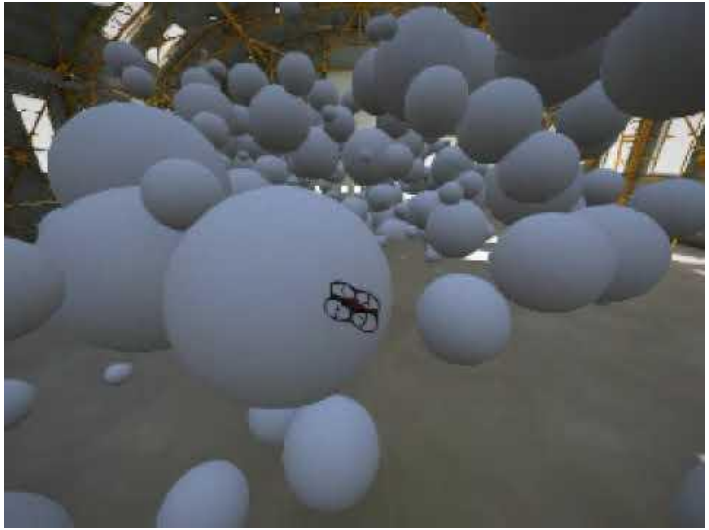
Failure: collide:16, bound: 7, time: <10s

hard

Goal rate: 26

Failure: collide:65, bound: 9

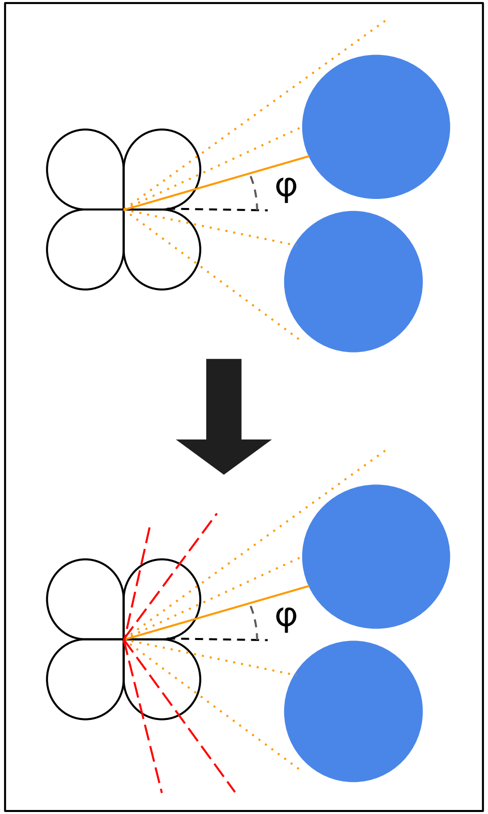




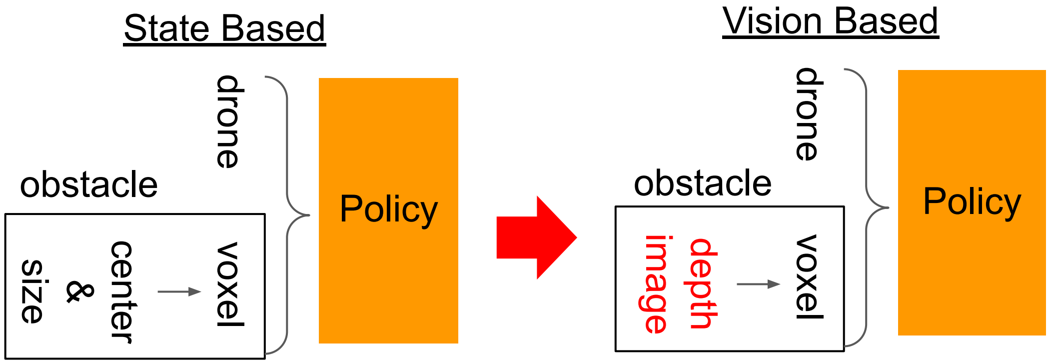
Ideas which do not work well

- Increasing viewing angle
  - because quadrotor collides on their side
  - φ, θ: -45~45 deg -> -90~90 deg
  - due to **too Large dimension of** obstacle, or **Sparse voxel**

- Curriculum Leaning (by Jeffrey Elman)
  - Firstly, learn at **medium** level (submission code)
  - Learn **hard** level based with same **NN weights**
    - pros: increase success rate **only in hard** environment
    - cons: decrease success rate in medium, and low velocity in both environment



Vision Based approach



Method

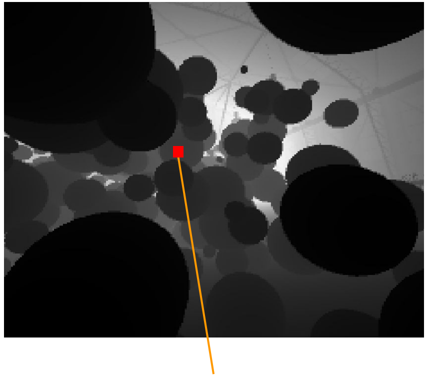
- same policy in state-based
- Make voxel by depth image

Results

goal rate: 6/10 (medium)

Problems

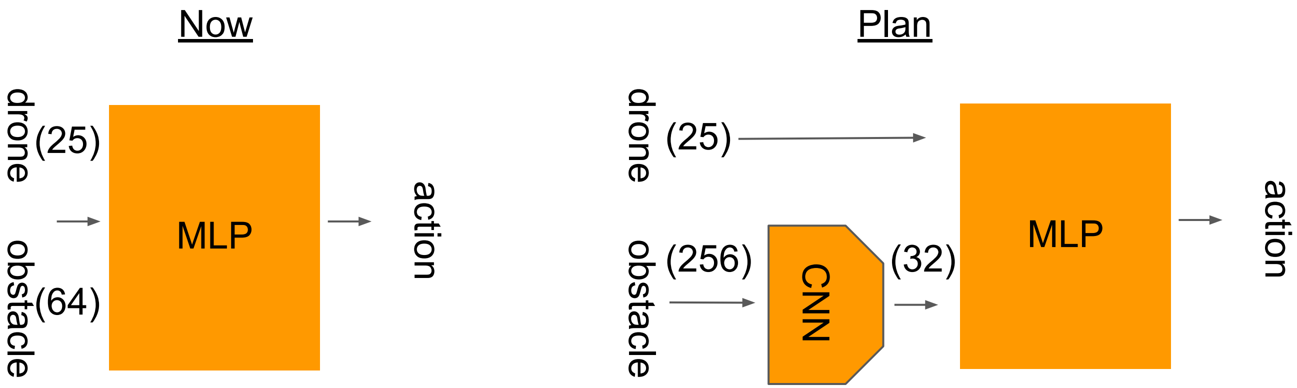
- affect ground in depth sensor
  - quadrotor assume ground as obstacle
  - drones rise
- depth image lack necessary information for voxel production
  - Increase FOV ->
    - Worse performance
    - Worse of depth data in quadrotor direction



depth pick up based on (θ,φ)

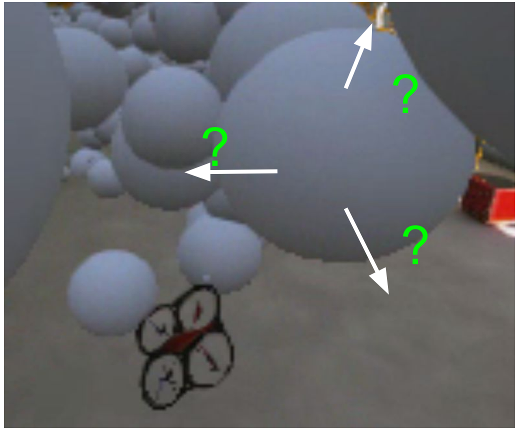
Future works

- Compress Obstacle information by CNN
    - increase input vision dimension, and FOV
        - helpful to avoid obstacle from their side
    - can use voxel position relasionship

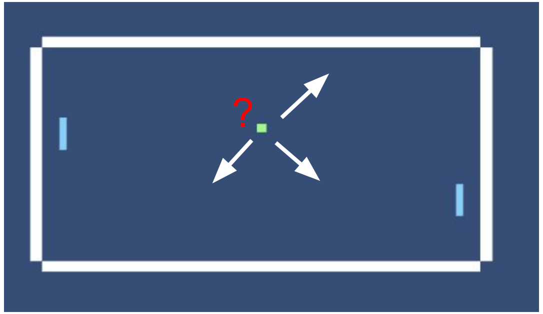


Future works

- Input time-series obstacle information
  - Now, agent don’t know obstacle velocity, and mainly hits moving obstacles
    - POMDP (cf. Pong)
      - harder than MDP
  - Implementing sliding window / RNN



DDC



Pong

Implement to real quadrotor

Implementation difficulty

- Unexplainable, unexpected flight by slight difference
  - 77/100 success on my PC ->1/3 success in the competition
  - 77/100 state-based -> 6/10 vision-based

Implement to real quadrotor

ANYmal

How to solve?

- Get NN based parameter model from real quadrotor for simulator (Jemin Hwangbo et al. Learning Agile and Dynamic Motor Skills for Legged Robots)
- Learning by cheating
  - Learn in many observation space(e.g. obstacles), Reduce observation in real quadrotor
- Dynamic balancing Model (Junhyeok Ahn et al. “Data-Efficient and Safe Learning for Humanoid Locomotion Aided by a Dynamic Balancing Model”)
  - set safety guaranteeing policy design
- Change action space to higher level
  - Eliminate unsafe, unrealistic flight
    - tradeoff of dynamic movement

- Mixed reality framework (Alessanrdo Devo, et al. “Autonomous Single-Image Drone Exploration With Deep Reinforcement Learning and Mixed Reality”)
  - convert real states to simulation engine
- Simulate with noisy observation, time delay

Table Of Contents