# CrazyFlie

# Content

1. The challenge
2. Our solution
    a. Interface
    b. Mapping
    c. Path Finding
    d. New Obstacles
3. The result
4. Potential Improvements

# The Challenge

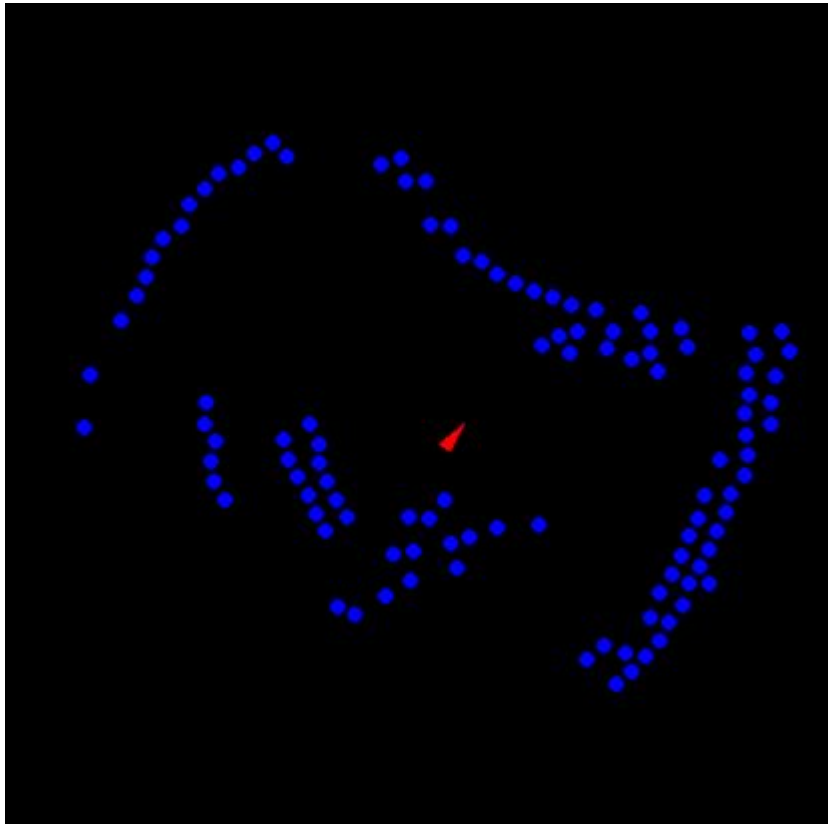
- Understanding drone
- Mapping the room
- Path Finding and executing
- Updating the room map dynamically
- Managing imperfections

# Our Solution: Interface

- PyGame
- Manual control
- Visuals
- Auto-control

# Our Solution: Mapping

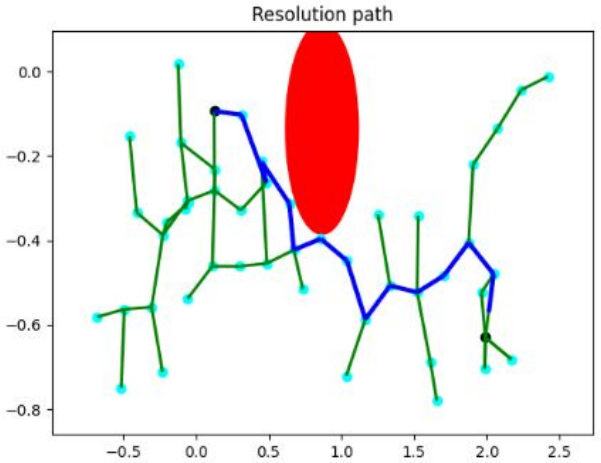
- Point Cloud
- Dynamically updated



# Our Solution: Path Finding

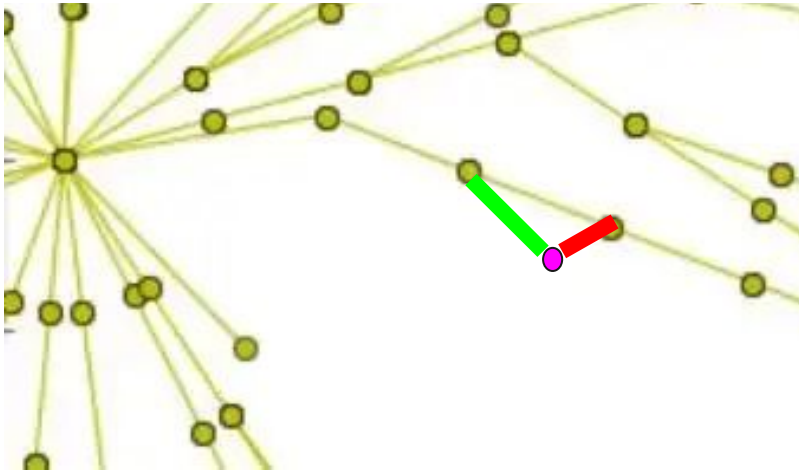
| A Star | RRT | RRT Star |
| --- | --- | --- |
| Optimal - Discrete - Fast | Smooth - Fast | Optimal -Smooth - Slower |

- RRT* + Optimize



# RRT* - Rapidly Exploring Random Trees

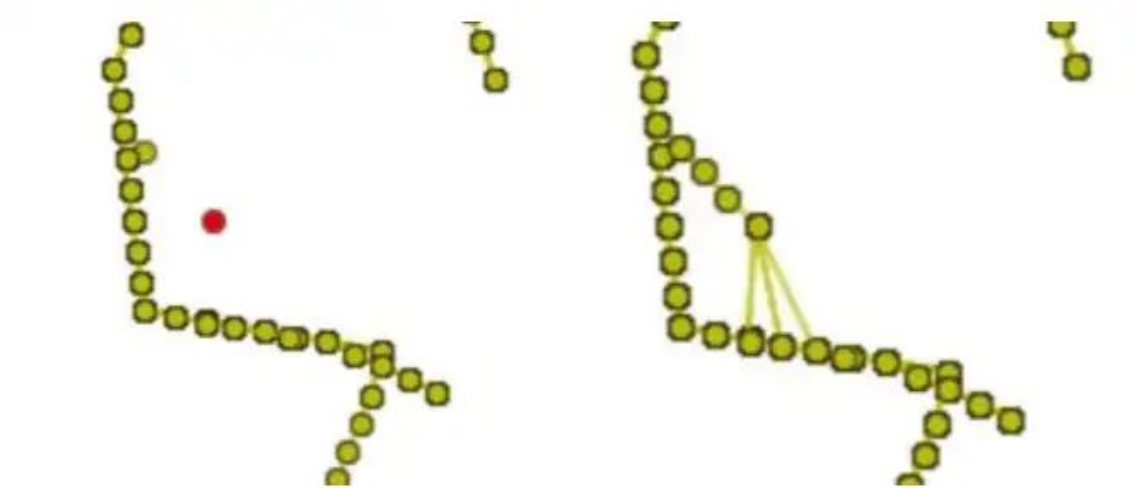
1. Random Nodes added + Not in obstacle
2. Connect to node with lowest cost + not through obstacle



Chinenov, Tim. “Robotic Path Planning: RRT and RRT*.” *Medium*, Medium, 26 Feb. 2019, https://theclassytim.medium.com/robotic-path-planning-rrt-and-rrt-212319121378.

# RRT* - Rapidly Exploring Random Trees

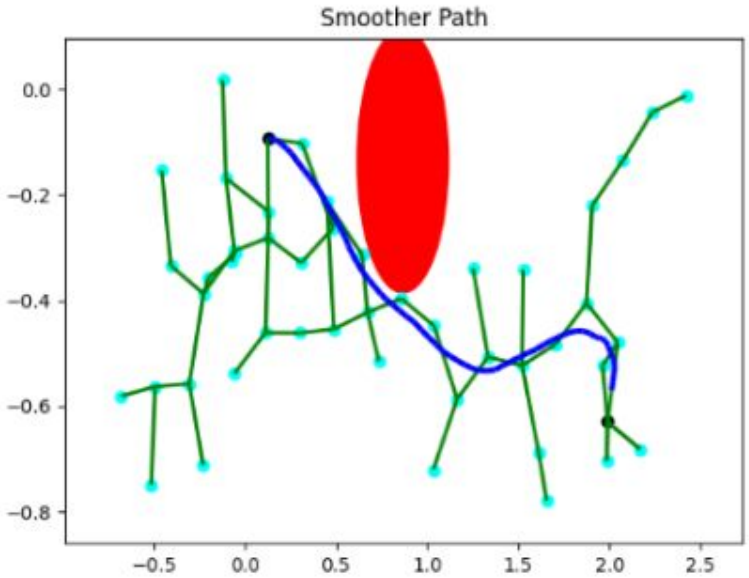
3. Check if new node caused other nodes to perhaps get better path



Chinenov, Tim. “Robotic Path Planning: RRT and RRT*.” *Medium*, Medium, 26 Feb. 2019, https://theclassytim.medium.com/robotic-path-planning-rrt-and-rrt-212319121378

# Path Following

- Path smoothing
- Constant forward velocity
- Adjust yaw based on next node
- Large error → no velocity



# Our Solution: New obstacles

1. Move specified distance away from close obstacle
2. Scan environment and update point cloud
3. Recalculate path

# The result

- Working well
- Parameters may be changed for use cases
- Demo

# Potential Improvements

- More reliable drone sensors
- SLAM
- Self exploring
- Improve efficiency of pathfinding