Simulation: External control of Swarms of Copters

Asema Hassan and Johannes Gätjen

[1]Department of Informatics
Otto-von-Guericke Universitat Magdeburg

Swarm Lab, 2016

Outline

Outline

Background/Objective

- Projector above arena in SwarmLab
- Can be used to give inputs to copters, e.g. formations
- Implement in Simulation environment:
  - Model projector based input
  - Model color sensor
  - Let copters react to input
- Goal: Attraction and repulsion controlled by projector input

Outline

Previous work

Flying in Formation SS15

- PID Controllers
- Projector image as texture on floor
- Color sensor; 1px camera attached with copter, pointing towards floor
- Ineffective gradient following

Outline

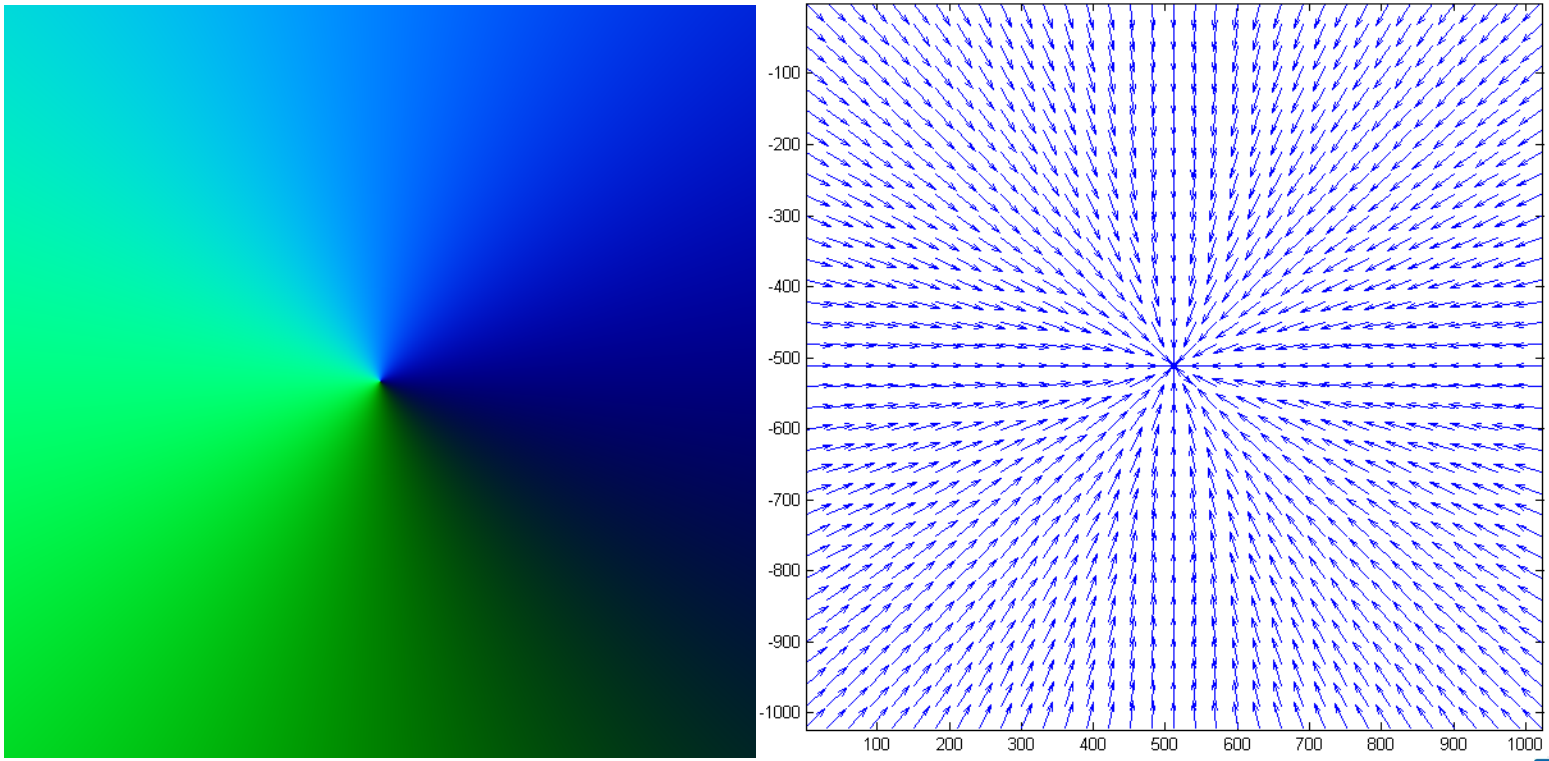
Milestones

1. Generate input images (height/gradient maps)
2. Use color sensor values to change copter behavior
3. Dynamic changing of input images
4. Position sensor (GPS or IMU)
5. Save color sensor values in local map
6. Calculate gradient from local map data
7. Use local map for gradient following
8. External control of parameters; remote API
9. Evaluate copter gradient following behavior
10. Evaluate swarm behavior

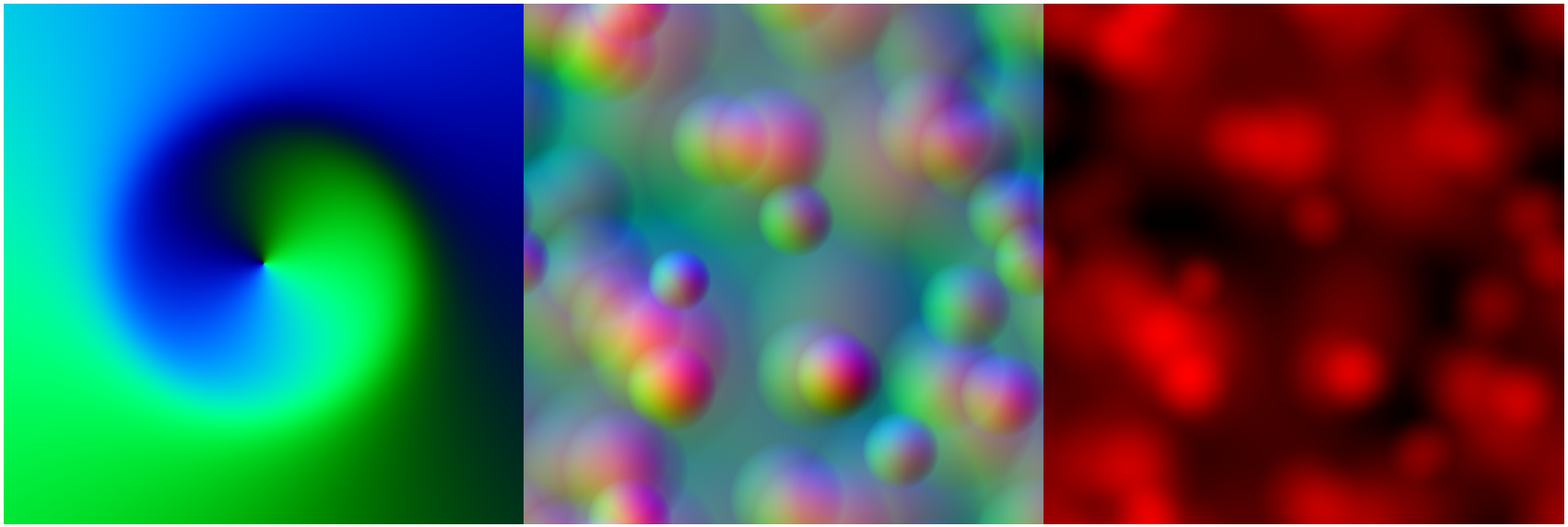
Outline

Achievements

1. *Gradient Maps:*
   - MATLAB to generate different gradient maps
   - Height in red, gradient in green and blue



Gradient Maps

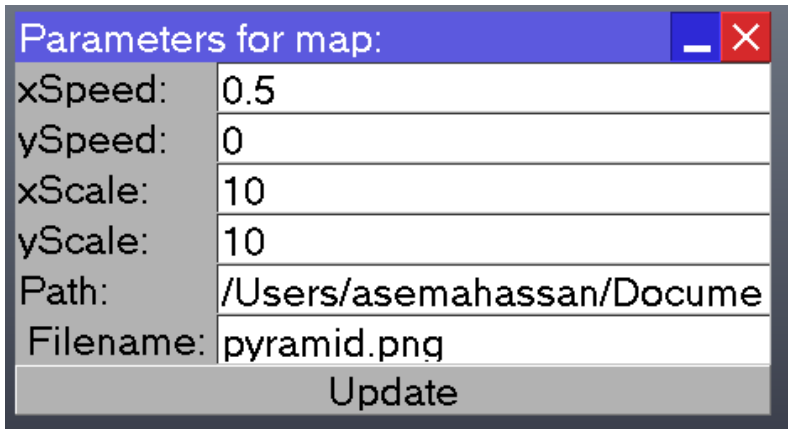


Achievements

2. *Gradient Follower Behavior:*
   - Compute gradient directly from color
   - Set target in direction of gradient
   - To reach the “highest” point
   - Tested with different gradient maps
   - [Demo simpleGradient]

Achievements

3. *Dynamic Landscapes:*
   - Dynamic loading of landscapes via external control scripts
   - Texture moves along X/Y
   - UI to set parameters
   - Can set scale, speed and file
   - Can also be set via remote API
   - Now whole arena moves with copter
   - [Demo Large_hills_smooth_gradient_map]



Achievements

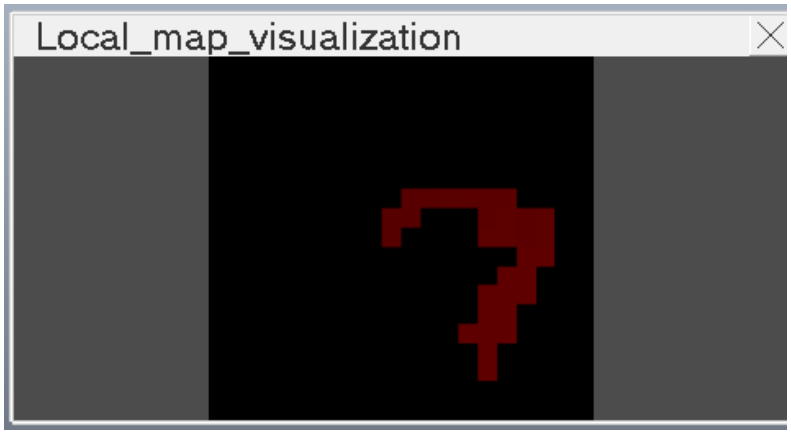
4.Implement a position sensor (GPS or IMU):

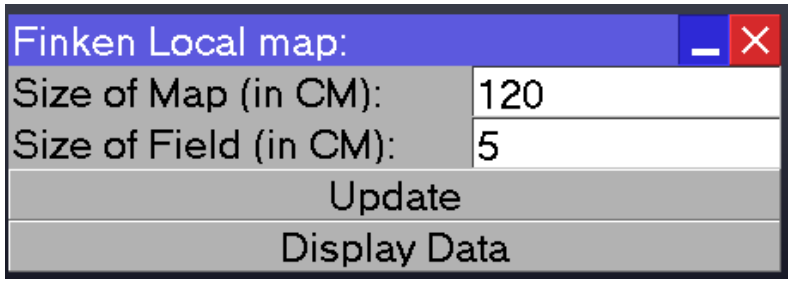
- Local map requires knowledge about position
- Implemented IMU (inertial measurement unit)
  - Absolute position not necessary
  - More general applications
- XYZ Velocity, Orientation and Angular rate
- Perfect data from simulation, added white noise
- Considered real data for noise magnitude, but sampling rate too low to get meaningful values

Achievements

5. Save sensor values in local map:

- Create virtual map grid (2D array) with a defined resolution
- User can set total size and field size (for resolution)
- Color/height values written in center
- Data shifted according to copter movement
- Incremental averaging of values: $(1 - \alpha) \cdot \text{old} + \alpha \cdot \text{new}$
- Relative position for more accuracy

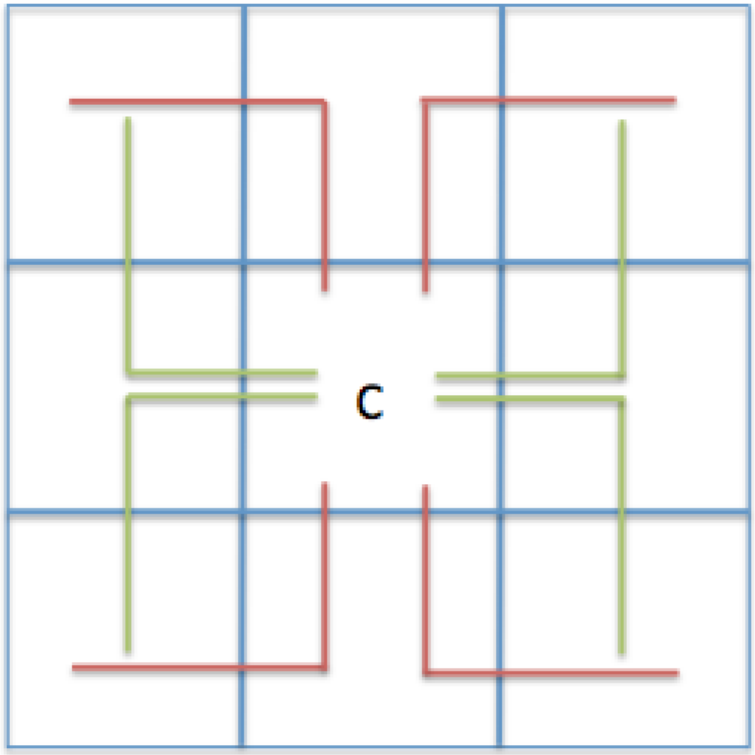


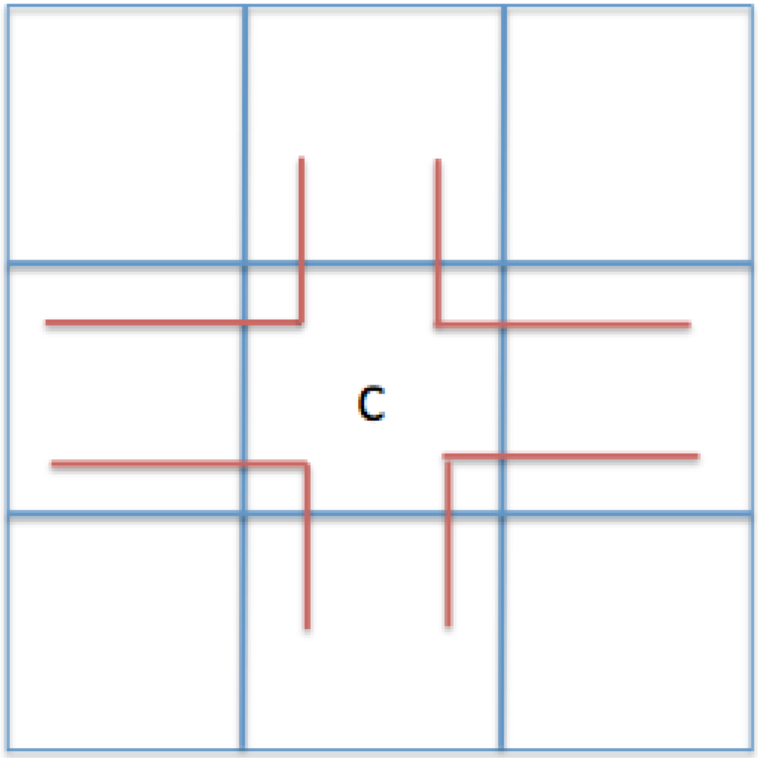


Achievements

6. Calculating gradient from local map:

- Need three values arranged in L-shape

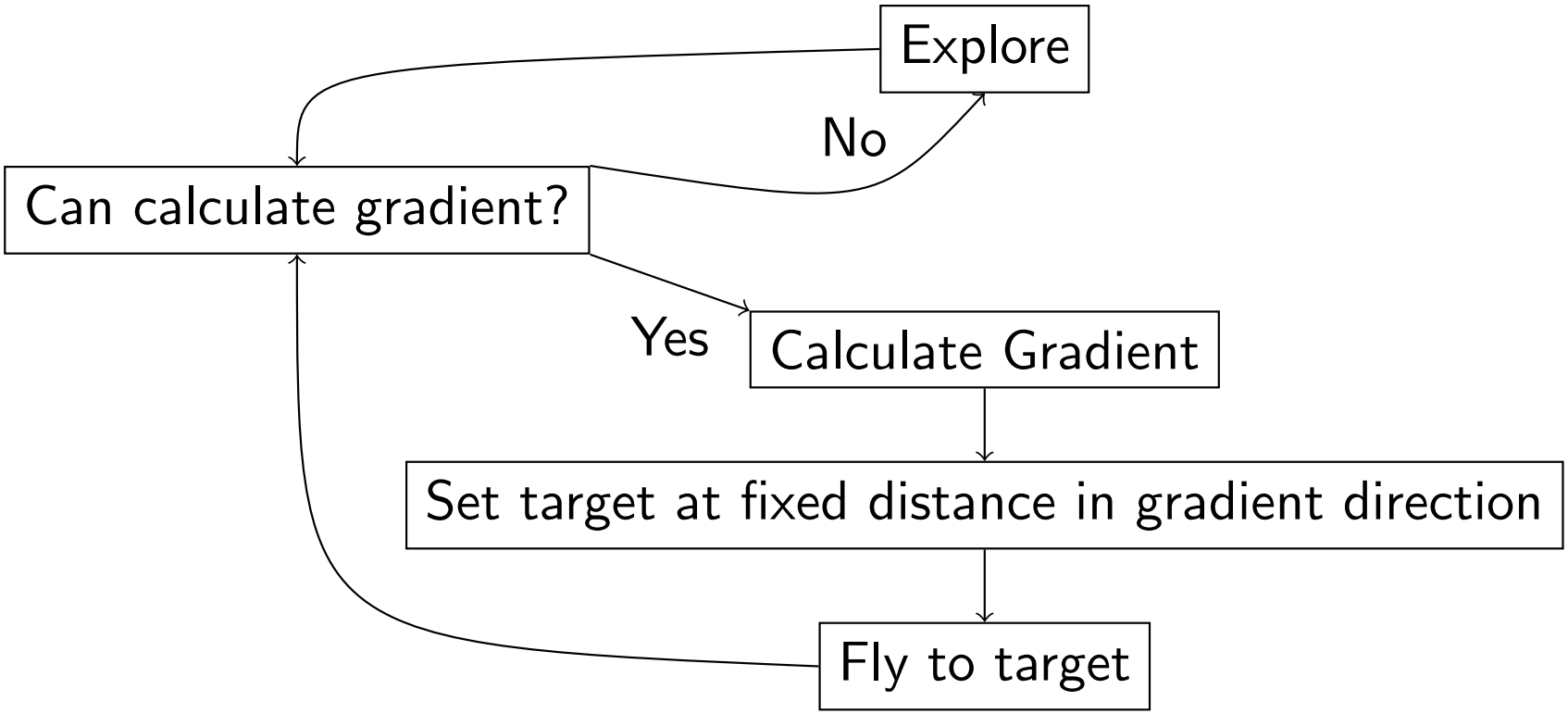




Achievements

7. Use local map for gradient following:

Baseline behavior



Achievements

7. Use local map for gradient following:

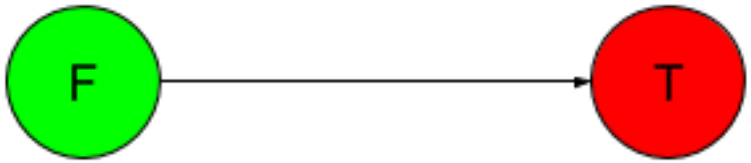


Figure: Baseline/Straight mode

Achievements

7. Use local map for gradient following:

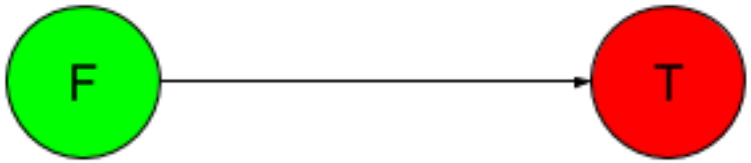


Figure: Baseline/Straight mode

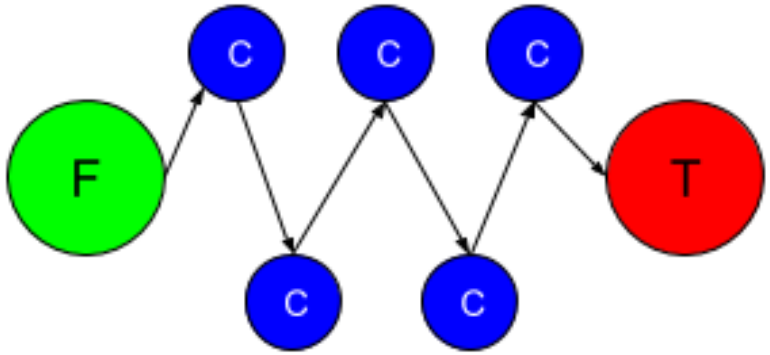


Figure: Drunk mode

Achievements

8. External Control of parameters:

- Using Remote API with Python
- To set parameters externally
- Start and stop simulation
- Repeated evaluation
- [Demo RemoteApiController]

Achievements

9. Evaluate copter gradient following behavior:

- Log recorded height values from color sensor
- Repeat simulation 25 times with random starting positions
- Average height over time
- Compare parameter settings

Evaluation Results

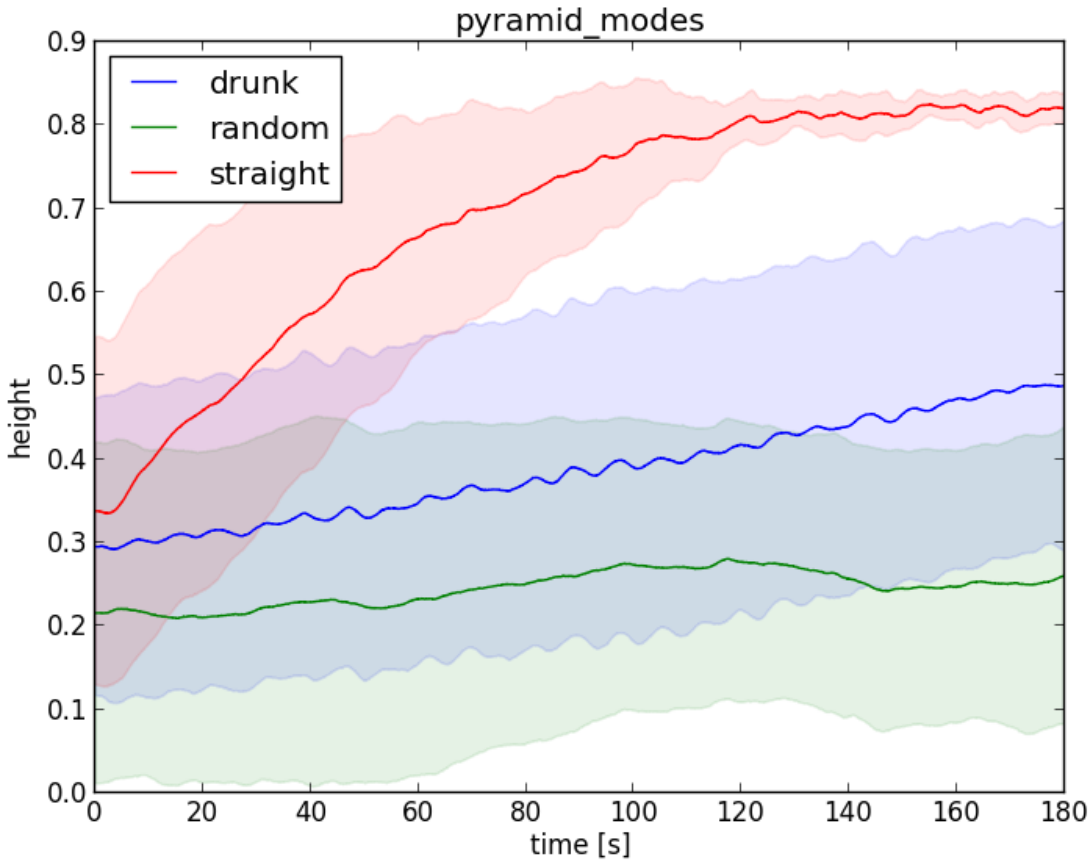


Figure: Comparison of modes on pyramid map

Evaluation Results

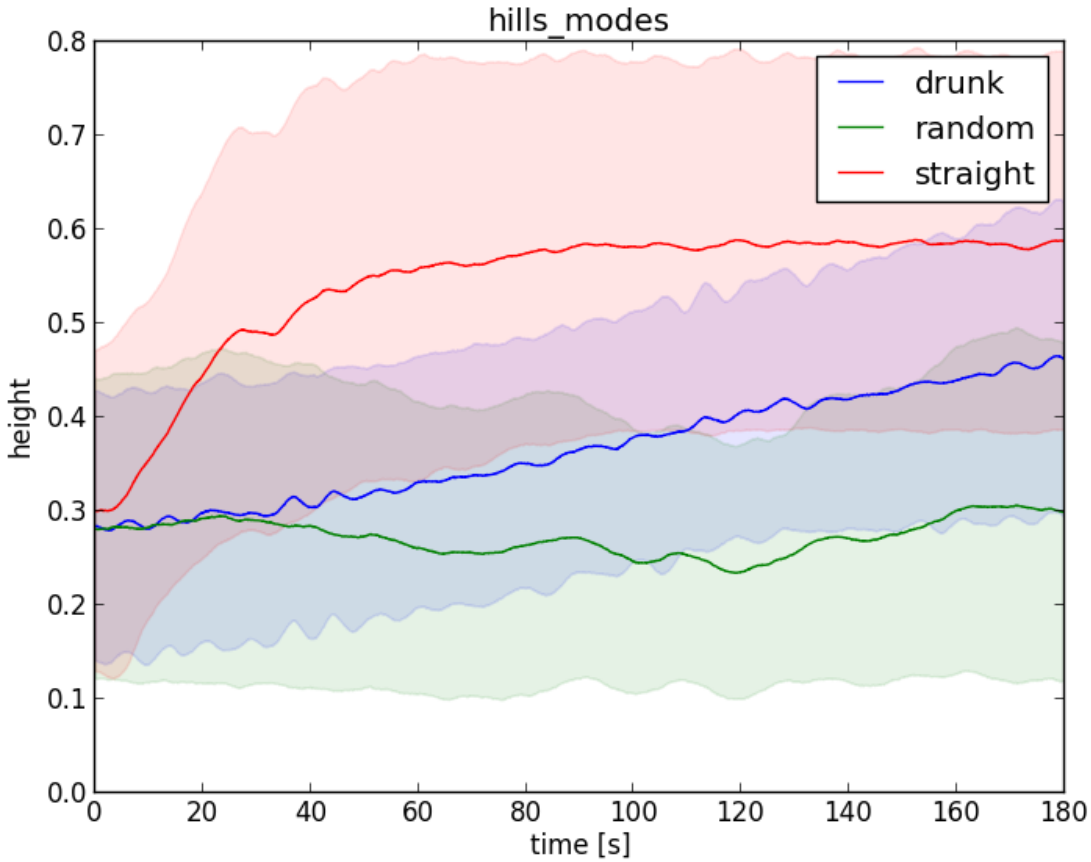


Figure: Comparison of modes on hills map

Evaluation Results

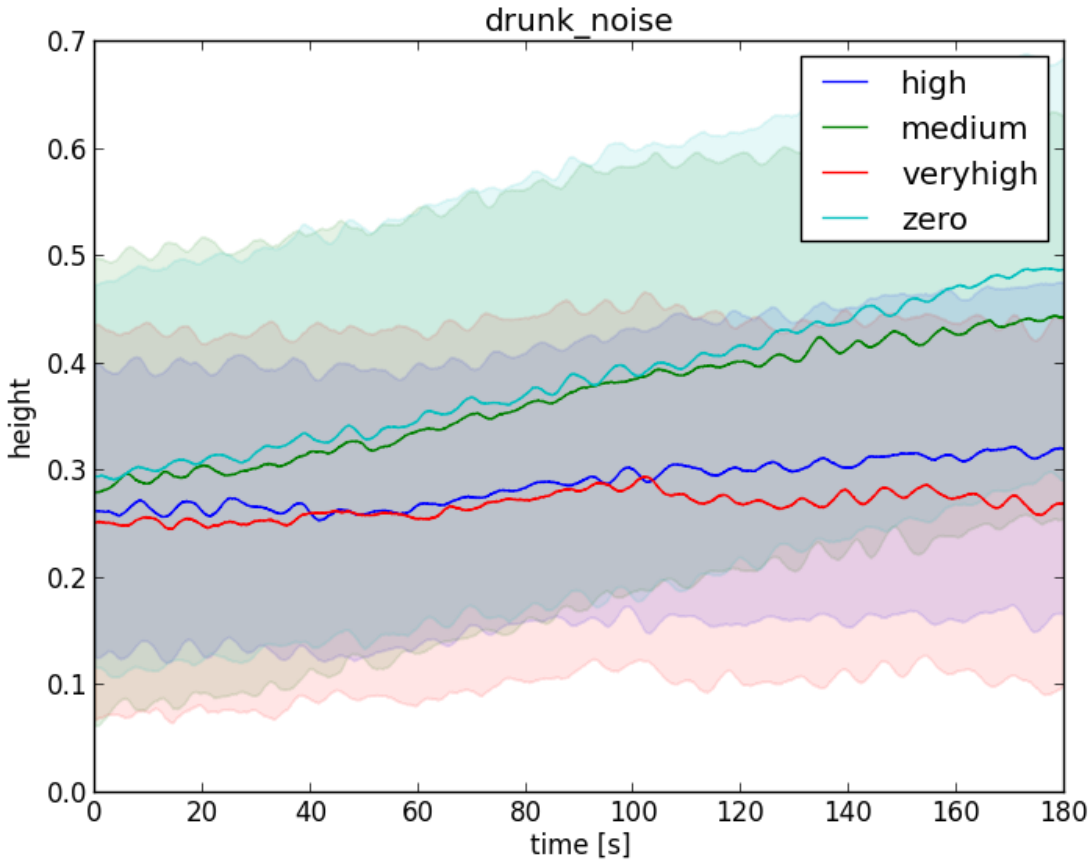


Figure: Comparison of IMU noise levels on pyramid map with **drunk** mode

Evaluation Results

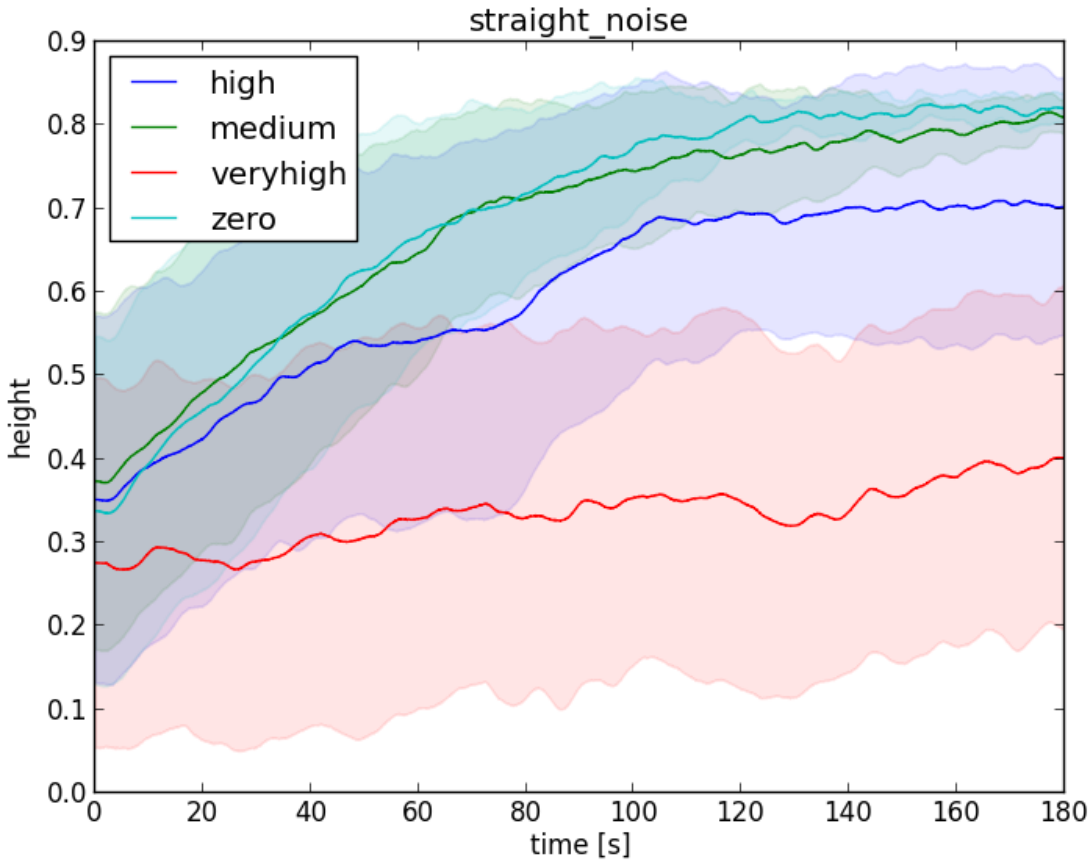


Figure: Comparison of IMU noise levels on pyramid map with **straight** mode

Summary Evaluation

- Unfortunately very many different parameters
- Copters can effectively find high points
- Straight mode faster than drunk, because no detours
  - Do not utilize additional information
- Straight mode more robust against noise
  - Local map in drunk mode more distorted

To do

10. Evaluate swarm gradient following behavior:

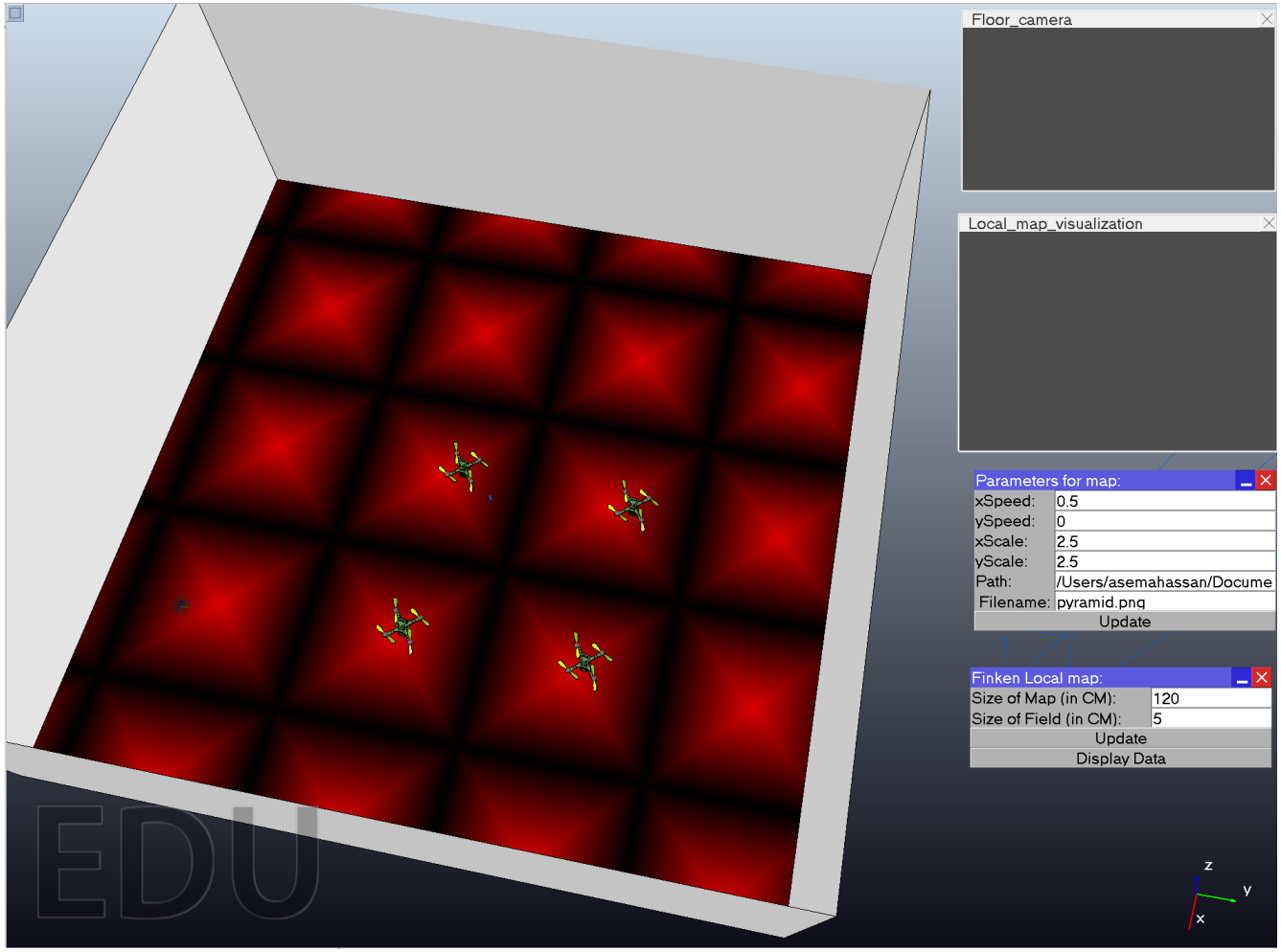


Figure: Multiple copters in Simulation not implemented yet

Conclusion

- Can generate random landscape images
- Implemented velocity sensor with white noise
- Copter builds map for surroundings
- Uses map to follow a gradient
- Simulation can be controlled via remote API

Limitations

- Color sensor unrealistically accurate
- Very many parameters to set $\rightarrow$ difficult to evaluate
- IMU noise magnitude not based on real data

Future Work

- Combine external control and swarm behavior
- Include global knowledge from local map
- Would benefit from better tuned PID controller
- Can store other information in local map
- Incorporate more long-term memory?

Discussions

Thank you for your attention!