

Autonomous Navigation Behaviors for an Aerial Robotics Software Framework

Guillermo Echeゴen Blanco
Technical University of Madrid
2018



2018-07-19

Autonomous Navigation Behaviors for an Aerial Robotics Software Framework

Guillermo Echeゴen Blanco
Technical University of Madrid
2018



Autonomous Navigation Behaviors
for an
Aerial Robotics Software Framework

1. Introduction

2. Problem Description

3. State of the Art

4. Implementation

5. Experiments

6. Conclusions

7. References

└ Overview

2018-07-19

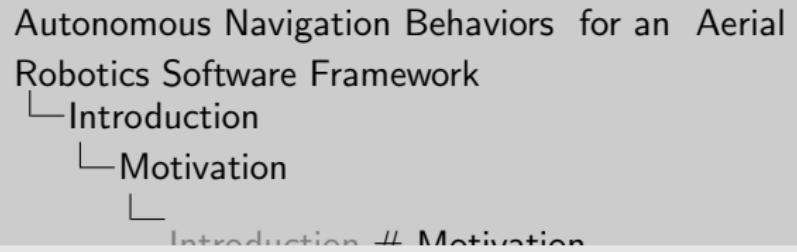
- 4. Implementation
- 1. Introduction
- 5. Experiments
- 2. Problem Description
- 6. Conclusions
- 3. State of the Art
- 7. References

Introduction

Autonomous Navigation

Move a drone through an automatically-generated, obstacle-free trajectory to a given point.

- Planning
- Localization
- Mapping



2018-07-19

Introduction # Objectives

As a **general goal** we want to:

- Construct an indoor *Autonomous Navigation* interface inside Aerostack using a **lidar** sensor.

We propose the use of different state of the art algorithms following the Aerostack framework. The **specific goals** are:

- Contribute to the research community
- Test implementation suitable for real time aerial robotics applications.

Autonomous Navigation Behaviors for an Aerial Robotics Software Framework

└ Introduction └ Objectives

└ Introduction # Objectives

2018-07-19

Introduction # Objectives

As a **general goal** we want to:

- Construct an indoor *Autonomous Navigation* interface inside Aerostack using a **lidar** sensor.

We propose the use of different state of the art algorithms following the Aerostack framework. The **specific goals** are:

- Contribute to the research community
- Test implementation suitable for real time aerial robotics applications.

Problem Description

Problem Description # Aerostack

General purpose, modular software framework for aerial robotics.

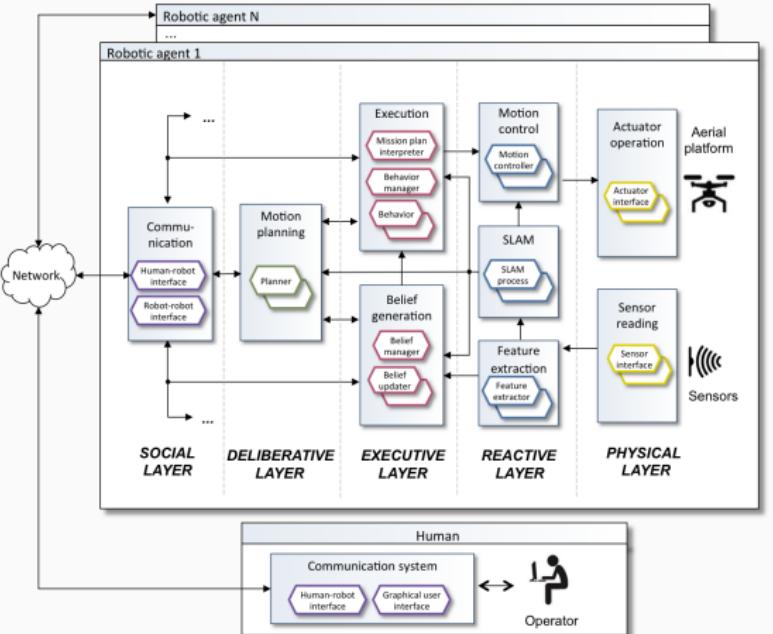


Figure 1: The Aerostack architecture

Autonomous Navigation Behaviors for an Aerial Robotics Software Framework

— Problem Description

—Aerostack

L

Problem Description # Aerostack

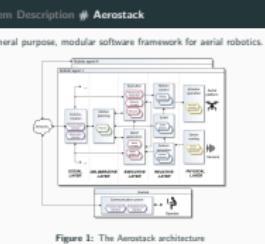


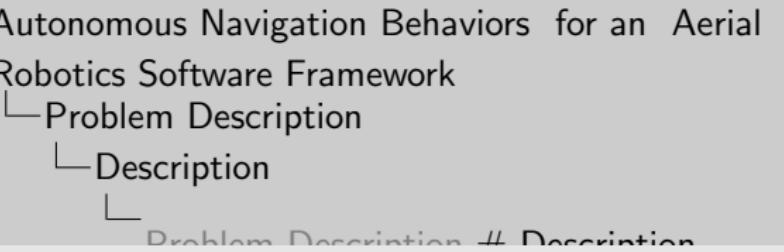
Figure 1: The Aerostack architecture

Problem Description # Description

Aerostack

Current version features:

- Planner: 2D geometric.
- Localization: Odometry, Aruco (*requires preparation*)
- Mapping: Hardcoded in the robot.



| Problem Description | # Description |
|---------------------|--|
| Aerostack | <p>Current version features:</p> <ul style="list-style-type: none">● Planner: 2D geometric.● Localization: Odometry, Aruco (<i>requires preparation</i>)● Mapping: Hardcoded in the robot. |

Problem Description # Improvements

To provide an **autonomous, lidar** based navigation interface we propose:

- A module to localize and map using **lidar** sensor.
- A module to generate obstacle-free trajectories using that map.
- A module to follow a given trajectory.

Autonomous Navigation Behaviors for an Aerial Robotics Software Framework

Problem Description

Improvements

2018-07-19

Problem Description # Improvements

To provide an autonomous, lidar based navigation interface we propose:

- A module to localize and map using **lidar** sensor.
- A module to generate obstacle-free trajectories using that map.
- A module to follow a given trajectory.

State of the Art

Autonomous Navigation Behaviors for an Aerial
Robotics Software Framework
└ State of the Art

2018-07-19

Usually based on trilateration.

Outdoor Localization

External sources:

Usually based on trilateration.

Outdoor Localization

External sources:

- GSM Antennas
- Satellites

Indoor Localization

Landmarks, beaconing (can provide more information)

- Bluetooth
- WiFi

In general, preparation of the environment is required.

Autonomous Navigation Behaviors for an Aerial Robotics Software Framework

State of the Art

Localization

State of the Art # Localization

Bluetooth beacons are used in conferences and hotels to provide useful information.

2018-07-19

State of the Art # SLAM

With **Simultaneous Localization and Mapping**, the robot constructs a map and localizes inside it at the same time.

Does not require environment preparation.

- EKF Slam
- Particle Filters
- Graph Optimization
- Lidar Slam
- Visual Slam

Autonomous Navigation Behaviors for an Aerial Robotics Software Framework

State of the Art

SLAM

State of the Art # SLAM

2018-07-19

State of the Art # SLAM

With **Simultaneous Localization and Mapping**, the robot constructs a map and localizes inside it at the same time. Does not require environment preparation.

- EKF Slam
- Particle Filters
- Graph Optimization
- Lidar Slam
- Visual Slam

- EKF: classic, intractable
- Particle Filters: Currently in use, fast, does not scale well for big maps
- Graph Optimization: Currently in use, very fast
- Lidar: the one we will use, fast and scalable
- Visual: SOTA orb slam, very interesting, requires large dictionary of words

State of the Art # Lidar Slam

Use a **lidar** sensor to do SLAM (Hector Slam [3]).

- Ray traces alignment as a Gauss-Newton minimization problem.
- Multi resolution occupancy grid map.
- Already available as ROS node.
- Suitable for real time
- Reliable enough

Autonomous Navigation Behaviors for an Aerial Robotics Software Framework

State of the Art

Lidar Slam

State of the Art # Lidar Slam

2018-07-19

State of the Art # Lidar Slam

Use a lidar sensor to do SLAM (Hector Slam [3]).

- Ray traces alignment as a Gauss-Newton minimization problem.
- Multi resolution occupancy grid map.
- Already available as ROS node.
- Suitable for real time
- Reliable enough

- Navigation system: interpolates IMU with SLAM, giving 3D, 6DOF
- 2D SLAM

It handles 3D with 6DOF by using a navigator that interpolates the IMU with the SLAM data. As a hill climbing algorithm (gradient ascent), it can get stuck in local minima, therefore, multiresolution occupancy grid map is used. Each coarser map have half the resolution of the preceding one, like in CV image pyramid problem. Instead of downsampling, the maps are kept in memory and simultaneously updated with the pose estimates of the alignment process.

State of the Art # Hector Slam

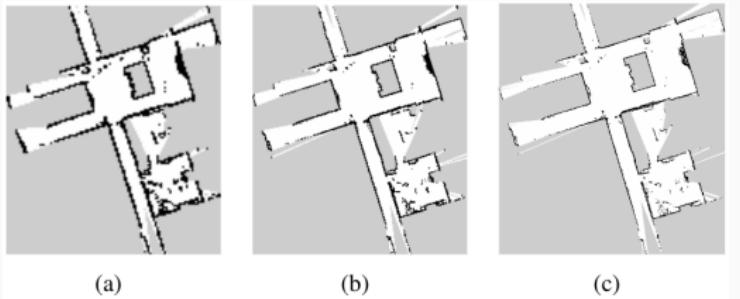


Figure 2: Multiresolution representation of the map



Figure 3: Vehicle experiments: Learned map during a competition

Both images taken from [3]

Autonomous Navigation Behaviors for an Aerial Robotics Software Framework
└ State of the Art
 └ Hector Slam
 └ State of the Art # Hector Slam

2018-07-19



Figure 2: Multiresolution representation of the map

Figure 3: Vehicle experiments: Learned map during a competition

State of the Art # Planning

Lots of planning algorithms:

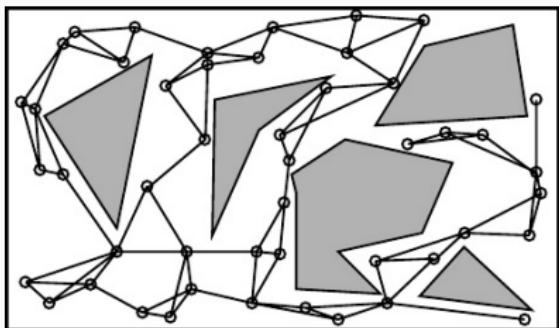


Figure 4: Probabilistic roadmaps
(taken from [2])

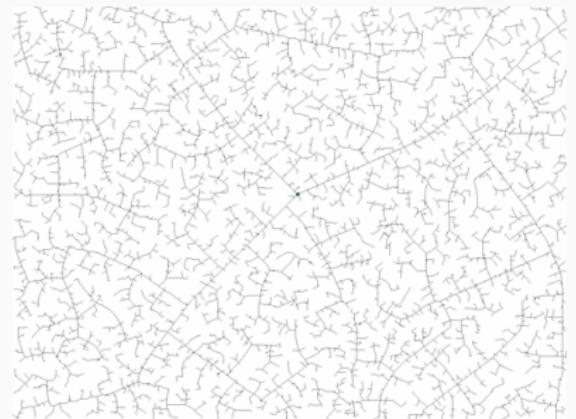


Figure 5: Rapidly exploring
Random Trees (taken from [1])

Autonomous Navigation Behaviors for an Aerial Robotics Software Framework

- └ State of the Art
 - └ Planning
 - └ State of the Art

2018-07-19

State of the Art # Planning

State of the Art # Planning

Lots of planning algorithms:



Figure 4: Probabilistic roadmaps
(taken from [2])

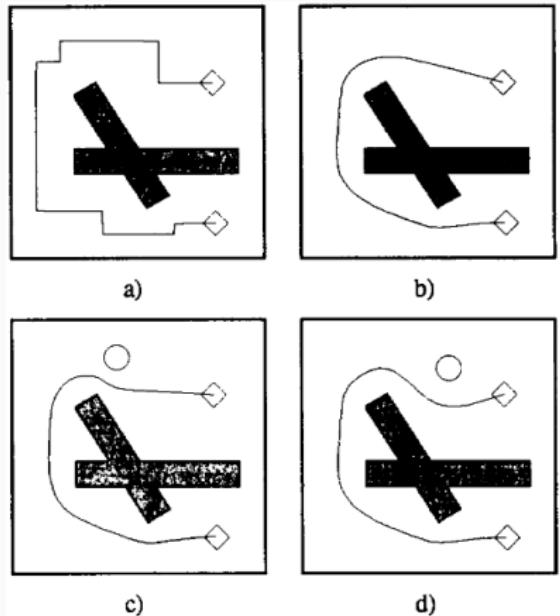


Figure 5: Rapidly exploring
Random Trees (taken from [1])

To Do := Comment algorithms and ?redo slide to include planners as text and figures as rows? Say something about both algorithms

State of the Art # Elastic Band Planner

Already available as a ROS module ([move base](#))



Pre planned path

- Forces applied
- Handle moving obstacles

In c) and d) an obstacle moves and the path is deformed.

Figure 6: Elastic Band with a moving obstacle (taken from [4])

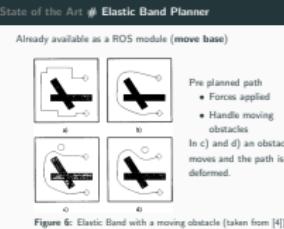
Autonomous Navigation Behaviors for an Aerial Robotics Software Framework

└ State of the Art

└ Elastic Band Planner

└ State of the Art # Elastic Band Planner

2018-07-19

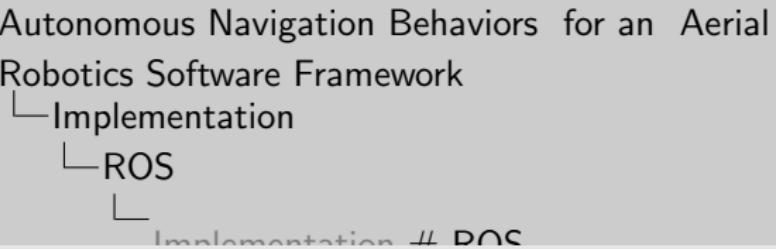


Implementation

Implementation # ROS

Robot Operating System. Widely used in robotics research.

- Pub-Sub architecture (topics & services).
- Multiple programming languages (C, C++, Python...)
- Logic encapsulation



2018-07-19

Implementation # ROS

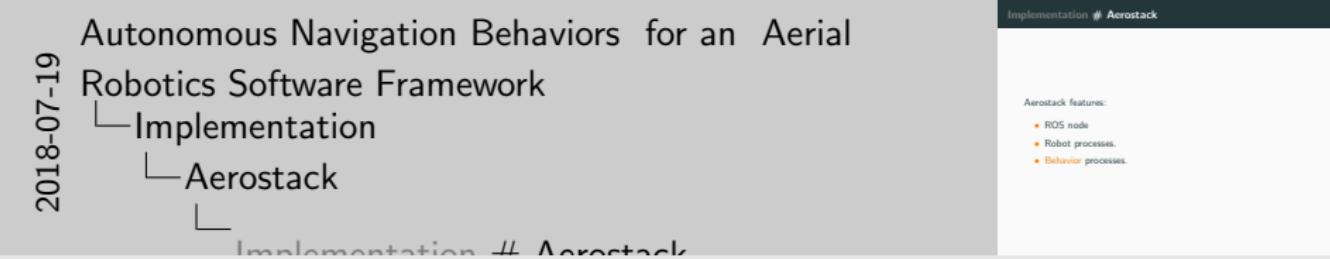
Robot Operating System. Widely used in robotics research.

- Pub-Sub architecture (topics & services).
- Multiple programming languages (C, C++, Python...)
- Logic encapsulation

Implementation # Aerostack

Aerostack features:

- ROS node
- Robot processes.
- Behavior processes.



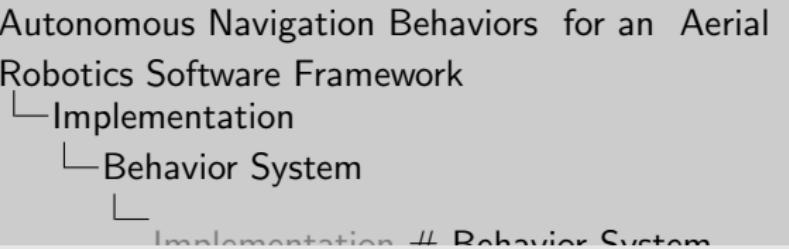
- ROS: Lowest level processes. Drivers & hardware.
- Robot process: Mid level process, general algorithms (like aruco recognition)
- Behavior: Highest level, algorithm monitor, useful to be coordinated.

Implementation # Behavior System

The Behavior Manager coordinates the execution of all the behaviors. A behavior:

- Is the highest level abstraction component.
- Encapsulates an algorithm.
- Listens to *start* and *stop* events and emits *error* events.
- Can have *exclusive constraints* and dependencies on other processes.

15



2018-07-19

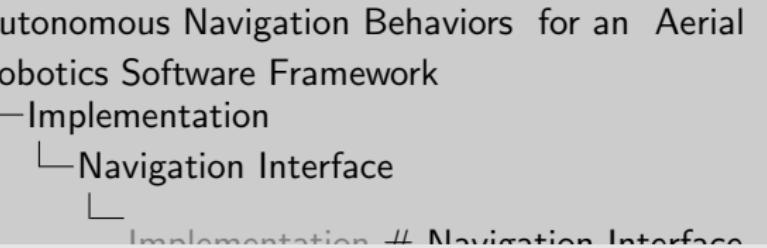
Implementation # Behavior System

- The Behavior Manager coordinates the execution of all the behaviors. A behavior:
- Is the highest level abstraction component.
 - Encapsulates an algorithm.
 - Listens to start and stop events and emits error events.
 - Can have exclusive constraints and dependencies on other processes.

Implementation # Navigation Interface

Therefore, the following components should be provided as behaviors or robots processes:

- **Behavior** to localize using lidar.
 - Behavior *Self Localize And Map by Lidar*
 - Hector Slam ROS module (monitored by ↑)
- **Behavior** to navigate the drone to a certain pose.
 - Behavior *Go To Point*
 - Behavior *Follow Path*
 - Behavior *Generate Path*
- **Robot process** to plan obstacle-free trajectories using lidar.
 - *Path planner robot process*
 - Move base ROS module (monitored by ↑)



2018-07-19

Implementation # Navigation Interface

Therefore, the following components should be provided as behaviors or robots processes:

- **Behavior** to localize using lidar.
 - Behavior *Self Localize And Map by Lidar*
 - Hector Slam ROS module (monitored by ↑)
- **Behavior** to navigate the drone to a certain pose.
 - Behavior *Go To Point*
 - Behavior *Follow Path*
 - Behavior *Generate Path*
- **Robot process** to plan obstacle-free trajectories using lidar.
 - *Path planner robot process*
 - Move base ROS module (monitored by ↑)

Implementation # Implemented Behaviors

Implementing each behavior separately enables:

- Different, granular control over functionality
- Testability of components
- Separation of functionality.

Autonomous Navigation Behaviors for an Aerial Robotics Software Framework

Implementation

Implemented Behaviors

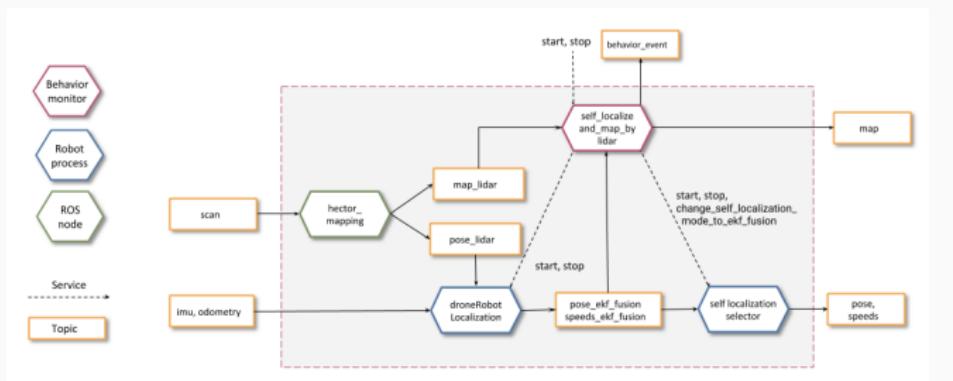
Implementation # Implemented Behaviors

- Implementing each behavior separately enables:
- Different, granular control over functionality
 - Testability of components
 - Separation of functionality.

Implementation # Behavior Self Localize And Map by Lidar

- Monitors the *Hector Slam* ROS module
- Applies EKF to: IMU, Odometry, SLAM
- Instructs the Aerostack for localization and mapping.

```
#-----  
# SELF_LOCALIZE_AND_MAP_BY_LIDAR  
#-----  
- behavior: SELF_LOCALIZE_AND_MAP_BY_LIDAR  
timeout: 120  
processes:  
- hector_mapping  
- droneRobotLocalization  
- self_localization_selector  
#-----  
# Self-localization behaviors are mutually exclusive  
#-----  
- mutually_exclusive:  
- SELF_LOCALIZE_BY_ODOMETRY  
- SELF_LOCALIZE_BY_VISUAL_MARKERS  
- SELF_LOCALIZE_AND_MAP_BY_LIDAR
```



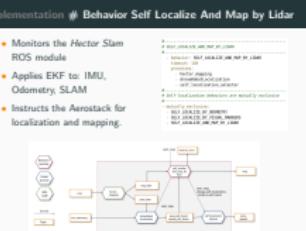
18

Autonomous Navigation Behaviors for an Aerial Robotics Software Framework

Implementation

Behavior Self Localize And Map by Lidar

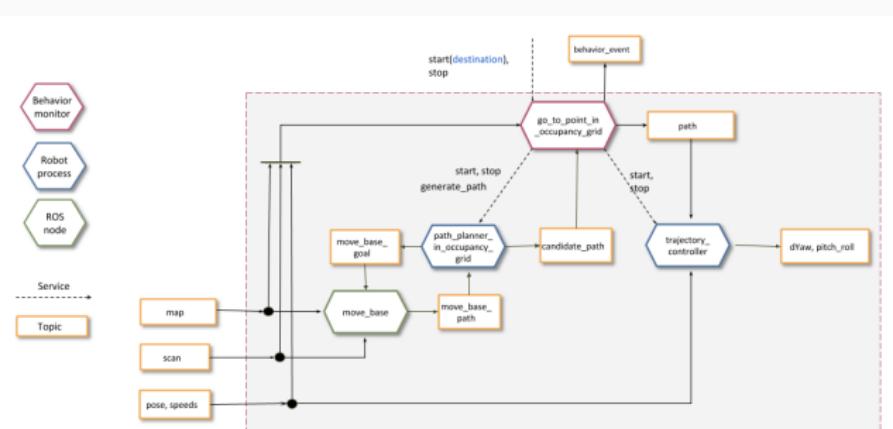
Implementation # Behavior Self Localize And



Implementation # Behavior Go To Point in Occupancy Grid

Given a point:

- Gets an obstacle-free trajectory.
- Instructs the trajectory controller to move the drone.
- Replans if an obstacle arises in the way.



```
#-----#
# GO_TO_POINT_IN_OCCUPANCY_GRID
#-----
- behavior: GO_TO_POINT_IN_OCCUPANCY_GRID
  timeout: 240
  processes:
    - move_base
    - path_planning_in_occupancy_grid
    - droneTrajectoryController
  arguments:
    - argument: coordinates
      allowed values: [-100,100]
      dimensions: 4

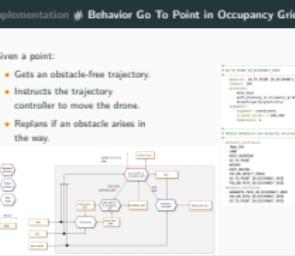
#-----#
# Motion behaviors are mutually exclusive
#-----
- mutually_exclusive:
  - TAKE_OFF
  - LAND
  - KEEP_HOVERING
  - GO_TO_POINT
  - ROTATE
  - KEEP_MOVING
  - FOLLOW_OBJECT_IMAGE
  - GO_TO_POINT_IN_OCCUPANCY_GRID
  - FOLLOW_PATH_IN_OCCUPANCY_GRID
  - mutually_exclusive:
    - GENERATE_PATH_IN_OCCUPANCY_GRID
    - FOLLOW_PATH_IN_OCCUPANCY_GRID
    - GO_TO_POINT_IN_OCCUPANCY_GRID
```

Autonomous Navigation Behaviors for an Aerial Robotics Software Framework
└ Implementation

└ Behavior Go To Point in Occupancy Grid

└ Implementation # Behavior Go To Point in

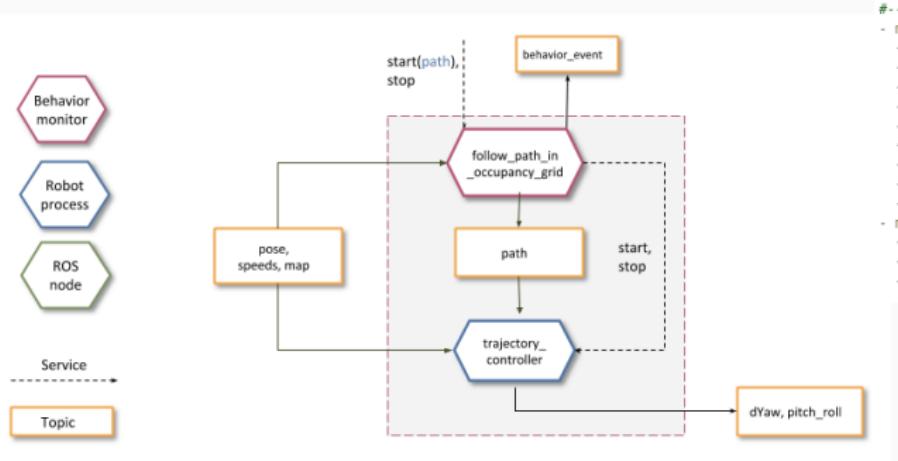
2018-07-19



Implementation # Behavior Follow Path in Occupancy Grid

Given a path:

- Instructs the trajectory controller to move the drone.
- No obstacle avoidance is provided.



```
#-----  
# FOLLOW_PATH_IN_OCCUPANCY_GRID  
#-----  
- behavior: FOLLOW_PATH_IN_OCCUPANCY_GRID  
  timeout: 240  
  processes:  
    - path_planning_in_occupancy_grid  
    - move_base  
    - droneTrajectoryController  
  arguments:  
    - argument: path  
      dimensions: 1  
  
#-----  
# Motion behaviors are mutually exclusive  
#-----  
- mutually_exclusive:  
  - TAKE_OFF  
  - LAND  
  - KEEP_HOVERING  
  - GO_TO_POINT  
  - ROTATE  
  - KEEP_MOVING  
  - FOLLOW_OBJECT_IMAGE  
  - GO_TO_POINT_IN_OCCUPANCY_GRID  
  - FOLLOW_PATH_IN_OCCUPANCY_GRID  
- mutually_exclusive:  
  - GENERATE_PATH_IN_OCCUPANCY_GRID  
  - FOLLOW_PATH_IN_OCCUPANCY_GRID  
  - GO_TO_POINT_IN_OCCUPANCY_GRID
```

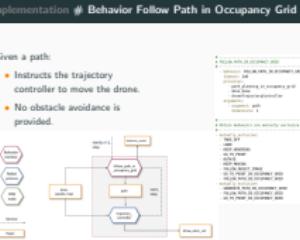
20

Autonomous Navigation Behaviors for an Aerial Robotics Software Framework

Implementation

Behavior Follow Path in Occupancy Grid

Implementation # Behavior Follow Path in

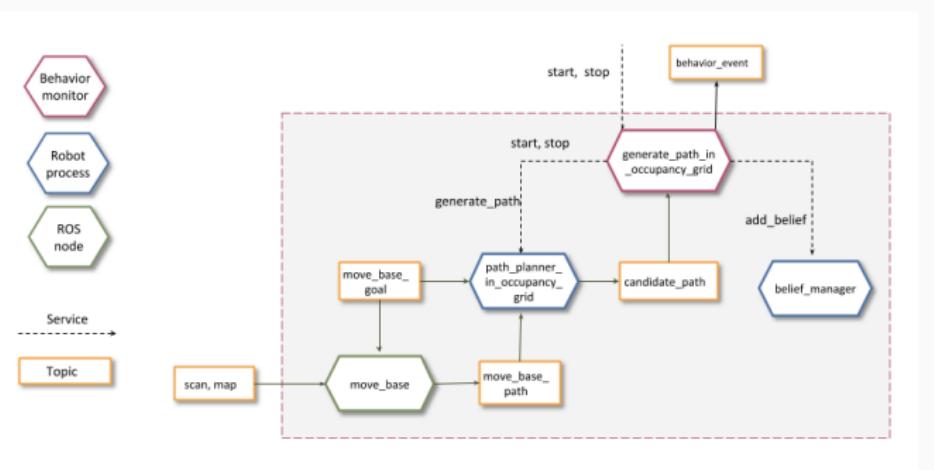


Implementation # Behavior Generate Path in Occupancy Grid

Given a point:

- Gets an obstacle-free trajectory.
- Stores the trajectory in the Belief Memory

```
#-----  
# GENERATE_PATH_IN_OCCUPANCY_GRID  
#-----  
- behavior: GENERATE_PATH_IN_OCCUPANCY_GRID  
timeout: 120  
processes:  
- move_base  
- path_planning_in_occupancy_grid  
arguments:  
- argument: coordinates  
allowed_values: [-100,100]  
dimensions: 3
```



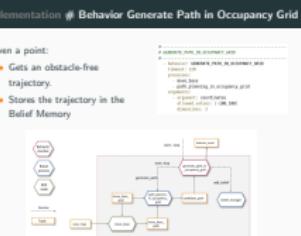
Autonomous Navigation Behaviors for an Aerial Robotics Software Framework

- Implementation
 - Behavior Generate Path in Occupancy Grid

2018-07-19

Behavior Generate Path in Occupancy Grid

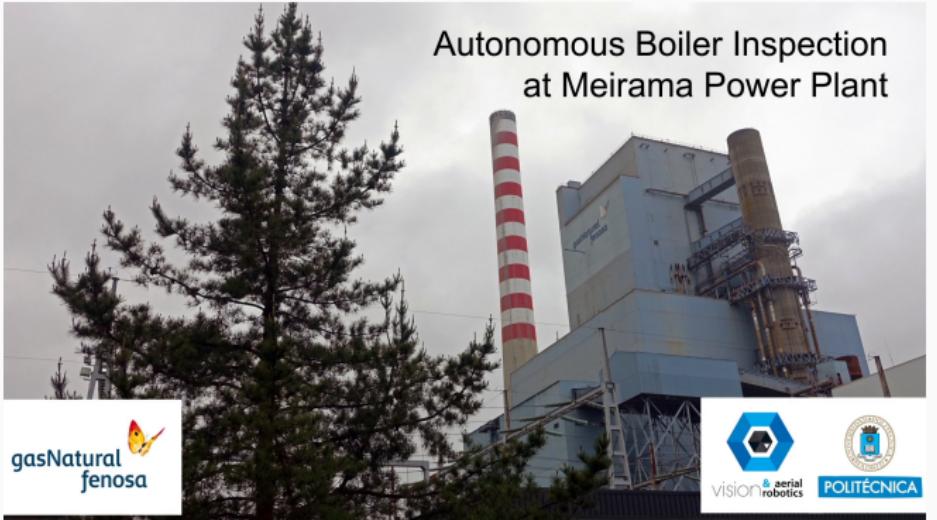
Implementation # Behavior Generate Path in



Experiments

Experiments

Boiler inspection mission



Autonomous Navigation Behaviors for an Aerial Robotics Software Framework
└ Experiments

2018-07-19

└ Experiments



Experiments # Experiment Details

Boiler mission both in simulation and real flight:

Simulation

- Gazebo + RotorS simulator
- Hummingbird Drone
- 16x16x57 - WxDxH mts

Real Flight

- Sports Center
- DJI 100 Matrice Drone
- 10x25x14 - WxDxH mts

Autonomous Navigation Behaviors for an Aerial Robotics Software Framework

Experiments

Experiment Details

Experiments # Experiment Details

2018-07-19

Experiments # Experiment Details

Boiler mission both in simulation and real flight:

Simulation

- Gazebo + RotorS simulator
- Hummingbird Drone
- 16x16x57 - WxDxH mts

Real Flight

- Sports Center
- DJI 100 Matrice Drone
- 10x25x14 - WxDxH mts

Experiments # Simulated Experiment

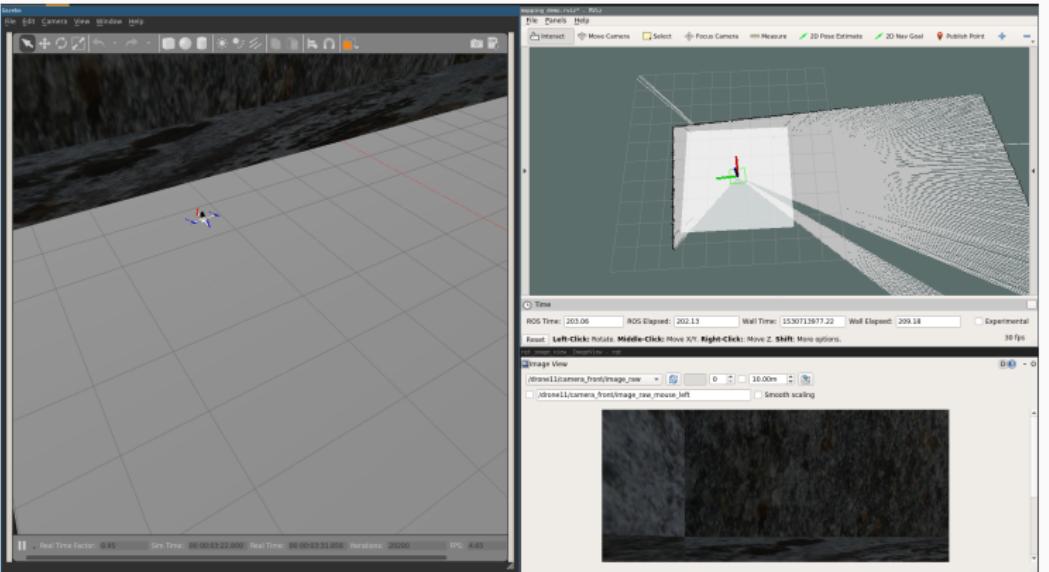


Figure 7: Simulated boiler environment

Autonomous Navigation Behaviors for an Aerial Robotics Software Framework
└ Experiments
 └ Simulated Experiment
 └ Experiments # Simulated Experiment

2018-07-19

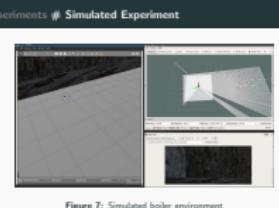


Figure 7: Simulated boiler environment

Experiments # Real Experiment

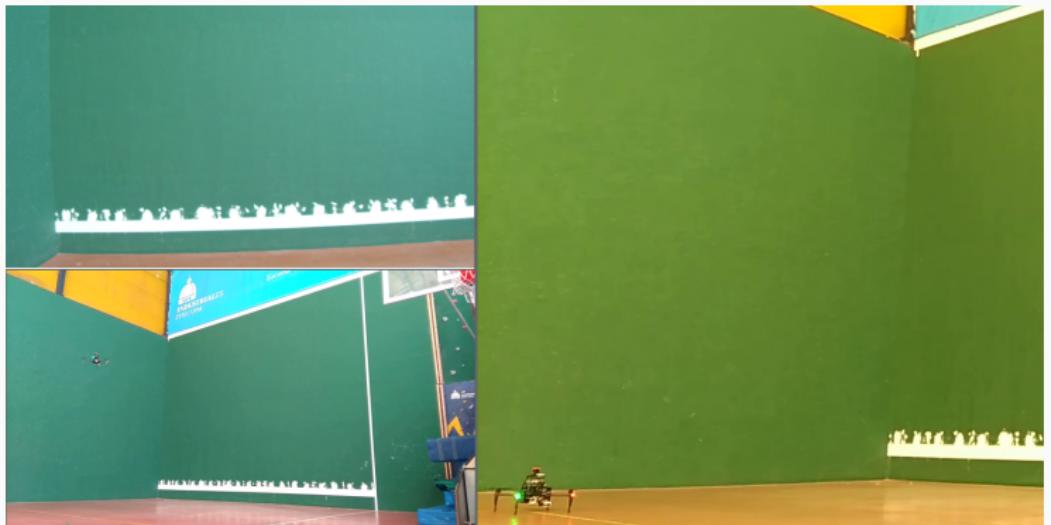


Figure 8: Real experiment on the Sports center ([video](#))

Autonomous Navigation Behaviors for an Aerial Robotics Software Framework

└ Experiments

 └ Real Experiment

 └ Experiments // Real Experiment

2018-07-19

Experiments # Real Experiment



Figure 8: Real experiment on the Sports center ([video](#))

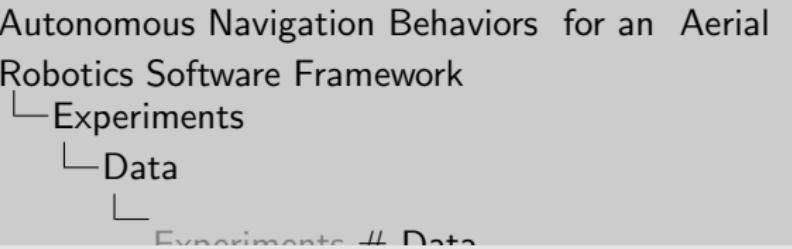
Experiments # Data

Simulation (10 tests, 6 points each)

| # | Correct | Avg. Total Point | Avg. Time |
|---------------|--------------|--------------------|---------------------|
| Go to Point | 41/60 (68%) | 1.82 (\pm 0.52) | 11.43 (\pm 3.12) |
| Follow Path | 39/60 (65%) | 1.97 (\pm 0.59) | 12.21 (\pm 3.58) |
| Generate Path | 60/60 (100%) | 0.20 (\pm 0.00) | 1.20 (\pm 0.01) |

Real Flight (3 tests, 6 points each)

| | | | |
|-------------|-------------|--------------------|--------------------|
| Go to Point | 16/18 (88%) | 0.94 (\pm 0.23) | 5.97 (\pm 1.38) |
|-------------|-------------|--------------------|--------------------|



| Experiments # Data | | | |
|--------------------------------------|--------------|--------------------|---------------------|
| Simulation (10 tests, 6 points each) | | | |
| # | Correct | Avg. Total Point | Avg. Time |
| Go to Point | 41/60 (68%) | 1.82 (\pm 0.52) | 11.43 (\pm 3.12) |
| Follow Path | 39/60 (65%) | 1.97 (\pm 0.59) | 12.21 (\pm 3.58) |
| Generate Path | 60/60 (100%) | 0.20 (\pm 0.00) | 1.20 (\pm 0.01) |

| Real Flight (3 tests, 6 points each) | | | |
|--------------------------------------|-------------|--------------------|--------------------|
| # | Correct | Avg. Total Point | Avg. Time |
| Go to Point | 16/18 (88%) | 0.94 (\pm 0.23) | 5.97 (\pm 1.38) |

- Simulation has more tests
- EKF Localization worse in simulation
- Time adds up because of timeouts (bad loc)

- Much room for improvement
- Localization is not accurate enough (lots of timeouts)
- Real flight is more stable & fast
- Simulation is more tested

Autonomous Navigation Behaviors for an Aerial Robotics Software Framework

└ Experiments

└ Discussion

└ Experiments # Discussion

2018-07-19

- Much room for improvement
- Localization is not accurate enough (lots of timeouts)
- Real flight is more stable & fast
- Simulation is more tested

Conclusions

Conclusions

With ...

- Improve Localization
- Lower timeout
- Extend testing

... an **autonomous navigation can be achieved.**

Autonomous Navigation Behaviors for an Aerial Robotics Software Framework

└ Conclusions

2018-07-19

└ Conclusions

Conclusions

With ...

- Improve Localization
- Lower timeout
- Extend testing

... an autonomous navigation can be achieved.

References

References

-  By Javed Hossain - Own work, CC BY-SA 3.0. *Rapidly Exploring Random Trees image Wikimedia Commons.* [Online; accessed 10-July-2018]. 2018. URL: https://upload.wikimedia.org/wikipedia/commons/6/62/Rapidly-exploring_Random_Tree_%28RRT%29_500x373.gif.
-  Howie M Choset. *Principles of Robot Motion: Theory, Algorithms, and Implementation.* Cambridge, Mass: MIT Press.

 S Kohlbrecher et al. "A Flexible and Scalable SLAM System with Full 3D Motion Estimation". In: *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE. Nov. 2011.

 S Quinlan and O Khatib. "Elastic bands: connecting path planning and control". In: *[1993] Proceedings IEEE International Conference on Robotics and Automation*. May 1993, 802–807 vol.2. DOI: 10.1109/ROBOT.1993.291936.

Autonomous Navigation Behaviors for an Aerial Robotics Software Framework

References

2018-07-19

References

-  S Kohlbrecher et al. "A Flexible and Scalable SLAM System with Full 3D Motion Estimation". In: *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE. Nov. 2011.
-  S Quinlan and O Khatib. "Elastic bands: connecting path planning and control". In: *[1993] Proceedings IEEE International Conference on Robotics and Automation*. May 1993, 802–807 vol.2. DOI: 10.1109/ROBOT.1993.291936.