System simulation of a fleet of drones to probe cumulus clouds

Rafael Bailon-Ruiz\textsuperscript{1}  Christophe Reymann\textsuperscript{1}  Simon Lacroix\textsuperscript{1}  
Gautier Hattenberger\textsuperscript{2}  Hector Garcia de Marina\textsuperscript{2}  
Fayçal Lamraoui\textsuperscript{3}

SHARC 2017

June 29, 2017

\textsuperscript{1}LAAS-CNRS, France
\textsuperscript{2}ENAC, France
\textsuperscript{3}Météo-France
Context

- Characterize the boundary layer of clouds
- Follow 4D evolution of the cloud
Problem statement

Collect data with spatial resolution of 10m at 1Hz over the cloud lifespan: 1 hour over 1km$^3$

Exploring clouds is a particularly complex task:

- Follow the 4D evolution of the cloud along 1D manifolds
- Highly constrained problem: Mission duration, UAV size and wind influence

The only way is to use multiple UAVs
Problem statement

Collect data with spatial resolution of 10m at 1Hz over the cloud lifespan: 1 hour over 1km$^3$

Exploring clouds is a particularly complex task:

- Follow the 4D evolution of the cloud along 1D manifolds
- Highly constrained problem: Mission duration, UAV size and wind influence

The only way is to use multiple UAVs
Main algorithms

Planning  *Coarse level*: "map those volumes"

*Fine level*: on-line optimal path generation.
Main algorithms

Mapping Gaussian Process Regression to model the cloud
Main algorithms

Mapping  Gaussian Process Regression to model the cloud
Main algorithms

Mapping  Gaussian Process Regression to model the cloud
The need for a simulation architecture

Mapping and path planning were yet to be integrated with realistic flight simulators

No existing straightforward way to simulate this kind of system as a whole

Primary goals of this work:

- Build a software architecture integrating realistic simulators
- Integrate the mapping and exploration algorithms within this architecture
- Test and validate the whole system.
The need for a simulation architecture

Mapping and path planning were yet to be integrated with realistic flight simulators

No existing straightforward way to simulate this kind of system as a whole

Primary goals of this work:

• Build a software architecture integrating realistic simulators
• Integrate the mapping and exploration algorithms within this architecture
• Test and validate the whole system.
The need for a simulation architecture

Mapping and path planning were yet to be integrated with realistic flight simulators. No existing straightforward way to simulate this kind of system as a whole.

Primary goals of this work:

- Build a software architecture integrating realistic simulators
- Integrate the mapping and exploration algorithms within this architecture
- Test and validate the whole system.
Simulation backends

Environment:

- **MesoNH**  MeteoFrance’s realistic cloud simulator
  Offline generation of a 64 km$^3$ fair weather scenario

Flight:

- **Paparazzi**  ENAC’s open source autopilot and ground control software
- **FlightGear**  Open source flight simulator
Concept of the simulation architecture

Overall structure

- Cloud Mapping
  - Measured wind
  - Estimated state
  - Noise
  - Wind ground truth
  - State ground truth

- Cloud map

- Trajectory planning

- Trajectories

- Area to map

- Real
  - Simulated

- Simulated cloud and simulated UAVs

- MesoNH

- Paparazzi
- FlightGear
- Control
- Guidance
Requirements of the new simulation architecture

Design a new software architecture being able to:

- Be prepared to handle a fleet of aircraft
- Integrate the project’s previous work
- Seamless transfer the algorithms to the real implementation
- Add new functionality easily

We decided to use the Robot Operating System (ROS) framework
Requirements of the new simulation architecture

Design a new software architecture being able to:

- Be prepared to handle a fleet of aircraft
- Integrate the project’s previous work
  - Seamless transfer the algorithms to the real implementation
  - Add new functionality easily

We decided to use the Robot Operating System (ROS) framework
Requirements of the new simulation architecture

Design a new software architecture being able to:

- Be prepared to handle a fleet of aircraft
- Integrate the project’s previous work
- Seamless transfer the algorithms to the real implementation
- Add new functionality easily

We decided to use the Robot Operating System (ROS) framework
Requirements of the new simulation architecture

Design a new software architecture being able to:

- Be prepared to handle a fleet of aircraft
- Integrate the project’s previous work
- Seamless transfer the algorithms to the real implementation
- Add new functionality easily

We decided to use the Robot Operating System (ROS) framework
Requirements of the new simulation architecture

Design a new software architecture being able to:

- Be prepared to handle a fleet of aircraft
- Integrate the project’s previous work
- Seamless transfer the algorithms to the real implementation
- Add new functionality easily

We decided to use the Robot Operating System (ROS) framework
ROS implementation of the simulation architecture

Implemented architecture

- Cloud Mapping
- Cloud map
- Trajectory planning
- Trajectories
- Areas to map
- Real
- Simulated

- Measured wind
- Estimated state
- Flight dynamic model
- Simulated cloud and simulated UAVs
- Wind ground truth
- State ground truth
- Noise
- Noise

- Paparazzi
- Control
- FlightGear
- Guidance

- MesoNH
ROS implementation of the simulation architecture
Time management

Input (Topic)  Process (Node)  Outputs (Topics)

/\clock\_control\n
/\clock\_generator\n
/\clock\n/tick
/\fast\_tick
Mapping & path planning node

Implemented architecture

/gpr_hyper_parameters | pathplanner
---|---
/tick

/ac_1

/ac_1/expected_uav_state
/ac_1/expected_uav_state_sequence
/ac_1/trajectory_sequence
/ac_1/measured_wind
/ac_1/uav_state
/ac_1/aircraft_command
Gaussian process hyper-parameters optimization

- Improve prediction with increasing wind samples
ROS implementation of the simulation architecture
MesoNH interface with flight backends

Implemented architecture
Paparazzi vs. FlightGear control loops

Paparazzi

- Trajectory
- Navigation
  - Vz, ψd
- Guidance
  - T, θ, ψ
- Stabilization
  - T, φ, θ, ψ
- Actuation
  - M1, M2...
- Aircraft
  - FDM

FlightGear

- Trajectory
- Navigation
  - Vz, ψd
- Guidance
  - T, θ, ψ
- Stabilization
  - T, φ, θ, ψ
- Actuation
  - M1, M2...
- Aircraft
  - FDM
Interface with Paparazzi

Figure: Interface between Paparazzi and SkyScanner ROS package
Interface with FlightGear

Figure: Interface scheme between FlightGear and SkyScanner ROS package
The SkyScanner ROS package

Figure: Whole simulation and control loop
Resulting trajectories
Summary

- Deployment of a simulation architecture
  - Path planning and mapping algorithms integration
  - Interfaces with realistic simulators
  - Extensible & Reusable
- Available in: http://github.com/rafael1193/skyscanner_integration