A reactive system for autonomous MAV navigation in unknown environments

Luca Di Stefano
Relatore: Prof. Eliseo Clementini
Correlatore: Dott. Enrico Stagnini

Università degli Studi dell'Aquila

Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica

UAV (Unmanned Aerial Vehicles)

Civilian use cases:

- Monitoring, data gathering
- Support to industry (e.g. precision farming)
- Delivery services

UAS (Unmanned Aerial System)

- UAV
- GCS (*Ground Control Station*)
- Communication

Multicopter UAVs

- Unexpensive
- Sufficient payload for most use cases
- High maneuverability

Autonomous navigation and guidance

Mission: sequence of operations the UAS must perform, e.g.:

1. Takeoff; reach 50m above ground
2. Reach *waypoint* W_1
3. Take a photograph
4. Go back to takeoff site
5. Land

Autonomous navigation and guidance

Mission: sequence of operations the UAS must perform, e.g.:

1. Takeoff; reach 50m above ground
2. Reach *waypoint* W_1
3. Take a photograph
4. Go back to takeoff site
5. Land

Navigation: following a flight plan, usually based on *waypoints* (and usually GPS ones, if outdoor)

Guidance: high-level mission-related decision: monitoring progress, altering, canceling

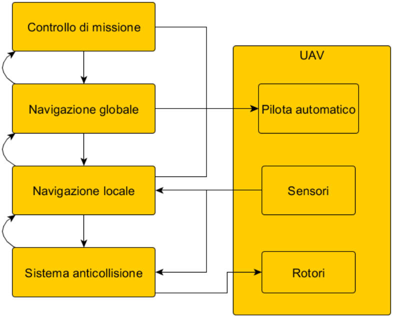
Autonomy of an UAS: ability to accomplish a mission without interaction with a human operator or other external systems.

State of the art

Hyerarchy of subsystems with different goals (Parentheses = main input)

- Collision avoidance (onboard sensors)
  Avoid crashing into objects (*obstacles*)
- *Local* navigation (onboard sensors)
  Reach the next *waypoint*
- *Global* navigation (maps of mission area)
  Flight plan management
- Mission control (Semantic/decision systems)
  Monitor/Change/Cancel the mission

Layered architecture example

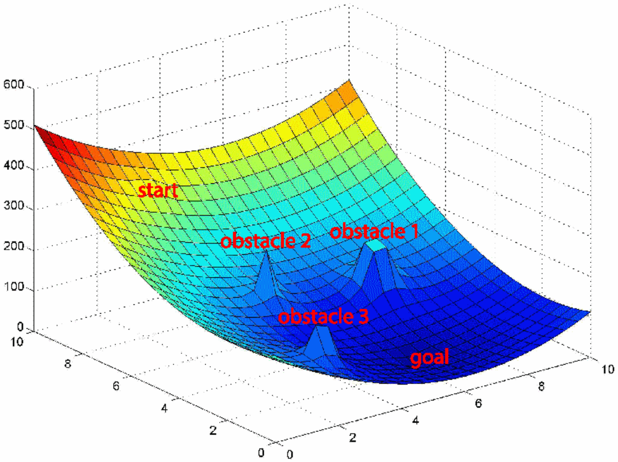


Layers architectural pattern (increasing abstraction)
Distributed system (UAV, GCS)

Local navigation

1. Map building from sensor data (graph or voxel DB)
2. Decision, based on one of the following (just to name a few):
   - Shortest path
   - Numerical optimization
   - Potential fields

Potential fields



Goals

An autonoumous local navigation system

Requirements

Function with limited environment data
High compatibility wrt. UAVs/sensors
Low computational cost

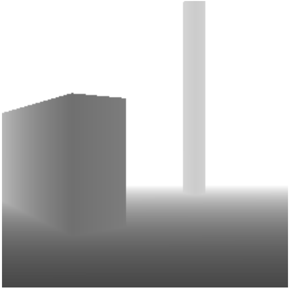
Architecture

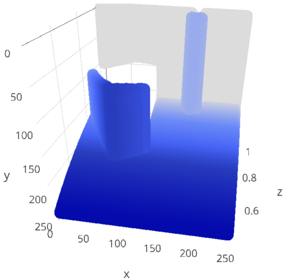
Onboard collision avoidance (safety measure wrt. gushes of wind, malfunctioning etc.)
Local navigation system deployed on the GCS

- Mapless: no model/map of the environment is stored
- Reactive: decisions only depend on current “stimuli”

Depth map

- Acquired by the UAV (LIDAR, TOF, etc.)
- Floating point raster, value is in range 0 to 1 for all pixels
- Value = normalized distance wrt. sensor range
- Horizontal/vertical angles of vision are known: we can reconstruct obstacle position in 3D space wrt. the UAV





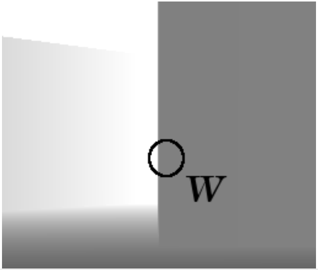
Visibility

Assume we know the GPS coordinates of the UAV and the goal W.

1. Get spherical coordinates of W wrt. the UAV;
2. Project W on the image plane, resulting in a pixel W';
3. Evaluate depth map in a neighbourhood of W'.
4. Recuperare i valori della depth map in un intorno di W'

By comparing the values with the UAV-W distance we can find out wheter an obstacle obstruct the visibility of W from the UAV position.
The radius of the neighbourhood is chosen according to:

- Drone radius
- Drone-waypoint distance



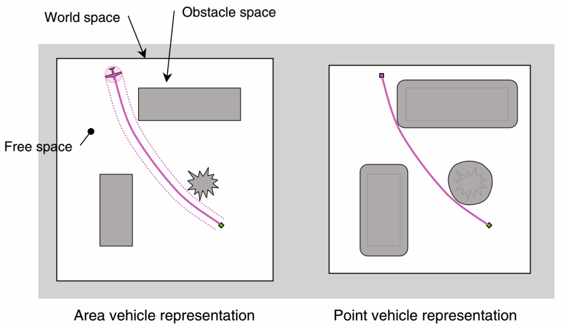
General description of the algorithm

Input: target waypoint T

```
1. W = T
2. While UAV is not in T:
    2.0 If UAV is in W and W != T:
        • W = T
    2.1. Point UAV towards W
    2.2. Acquire depth map
    2.3. Check visibility of W:
        • If visible, reach it
        • Otherwise:
            • If W == T:   W = (intermediate waypoint)
            • Otherwise    W = T
```

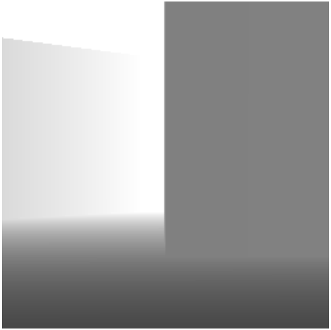
Replanning (1)

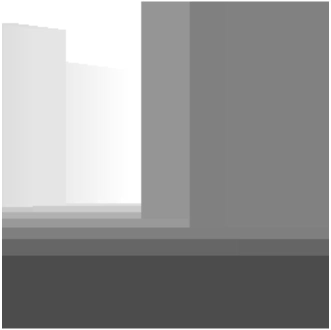
If goal is not visible, a new waypoint W_{new} has to be placed in the visible space. $DW_{new} \leq DW$
But the UAV volume might be a problem!
Solution (in 2D): *dilation* of the obstacle map:



Replanning (2)

Divide the depth map in n layers, according to their values
Separately dilate each layer: bigger radius for closer layers

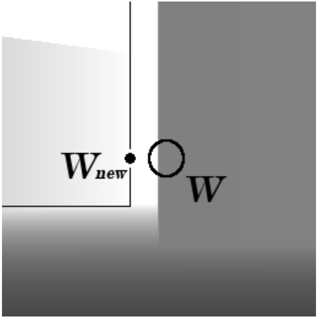




Replanning (3)

Set to 1 all pixels above a treshold (foreground/background segregation)
Choose the best pixel among those adjacent to the dilated obstacle

We now have ϑ, φ for the new waypoint.
ρ = value of first pixel < 1 on the segment goind from W_{new} to W (obstacle edge)

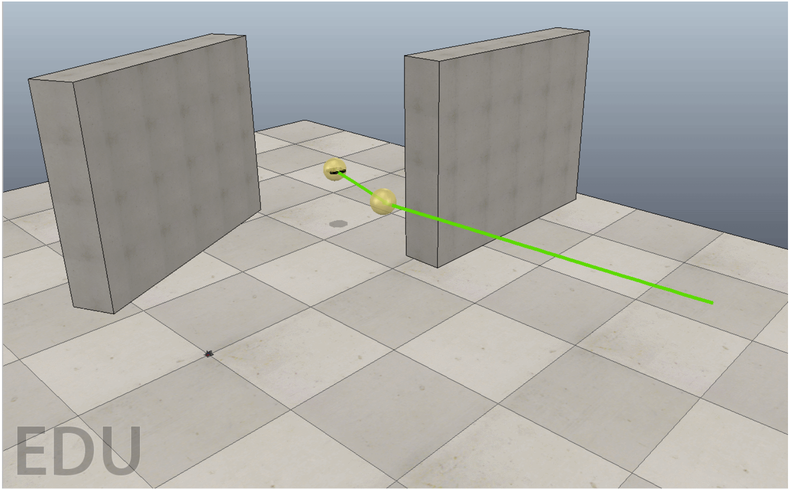


Simulation platform

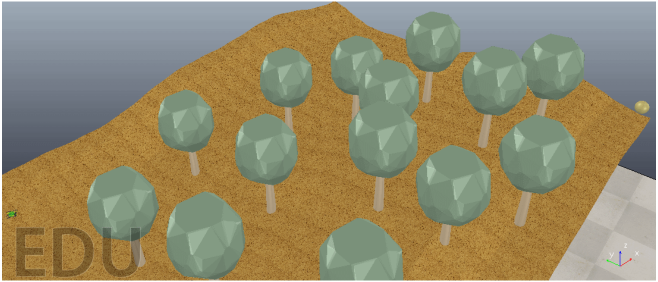
- V-REP simulation platform
- Algorithm written in Python 3.5
  - API V-REP
  - Numpy
  - OpenCV

Adapter classes encapsulates communication between the algorithm and V-REP: this should speed up porting to a real platform.

1st Scenario



2nd Scenario - Description

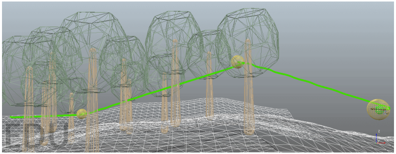


12 simulations
LIDAR resolutions: 256x256 to 32x32
Fixed AOV, 90° (horizontal/vertical)
Low UAV speed (0.6 m/s)

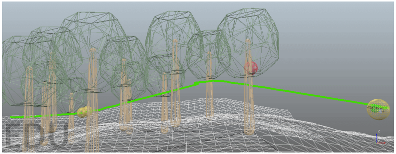
2nd Scenario - Results (1)

#	Risoluzione	Waypoint		Distanza (m)		Tempo
		Raggiunti	Scartati	Assoluta (m)	Normalizzata	
1	256	2	0	24.16	1.04	**00:58.6**
2	256	2	1	23.83	1.03	01:12.0
3	256	3	0	**23.43**	**1.01**	01:22.5
4	128	2	1	23.62	1.02	01:13.0
5	128	4	1	24.08	1.03	01:32.0
6	128	3	1	23.54	1.01	01:12.1
7	64	4	5	**24.99**	**1.07**	**02:09.6**
8	64	3	2	24.19	1.04	01:20.7
9	64	3	1	23.68	1.02	01:20.3
10	32	4	1	23.73	1.02	01:25.8
11	32	3	2	24.05	1.03	01:46.2
12	32	3	2	23.79	1.02	01:18.3

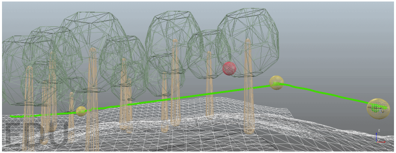
Scenario 2 - Results (2)



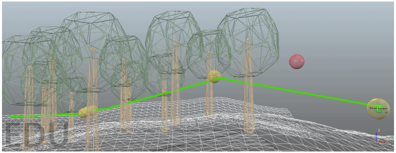
(a) Simulazione 1.



(b) Simulazione 4.



(c) Simulazione 9.



(d) Simulazione 10.

Conclusions

- Robust wrt depth map resolution
- High quality of results wrt ptimal distance

Possibili sviluppi futuri

- Real implementation
- Dynamic safety margin, taking quality of GPS fix into account
- Depth map processing to build obstacle maps (supporting future missions in the same area)