UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

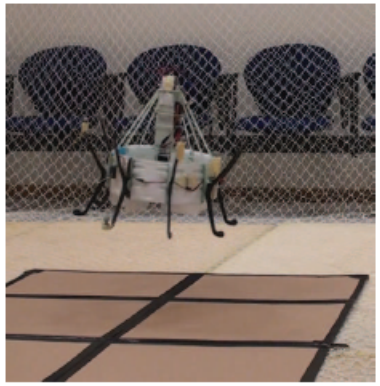
Integrated Planning and Execution for an Aerial Service Vehicle

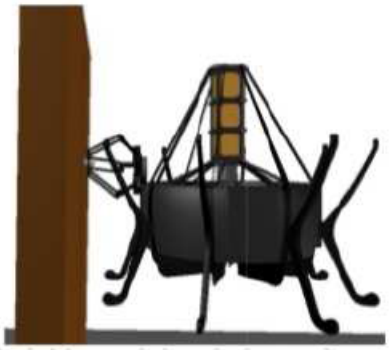
J. Cacace, A. Finzi, V. Lippiello, G. Loianno, D. Sanzone
DIETI, Università degli Studi di Napoli Federico II,
via Claudio 21, 80125, Naples, Italy

Introduction

- We present the case study of a high-level control system designed for an Aerial Service Vehicle (ASV)

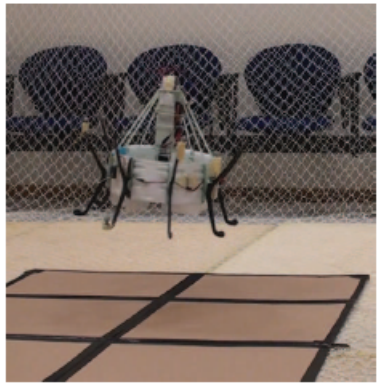
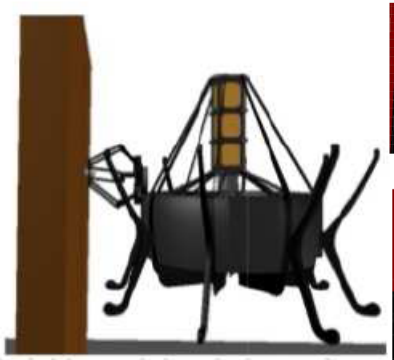
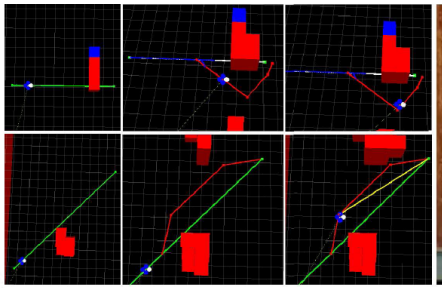
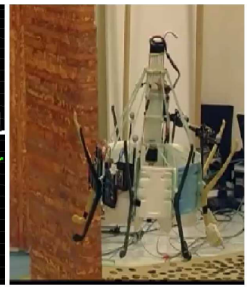
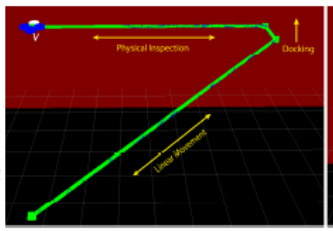
- This work is framed within the the AIRobots project (FP7 ICT2486669, Marconi et al. 2012a):
  - "A new generation of unmanned service helicopters, equipped with sensors and end-effectors, and capable not only to fly, but also to achieve robotic tasks in proximity and in contact with the surface".

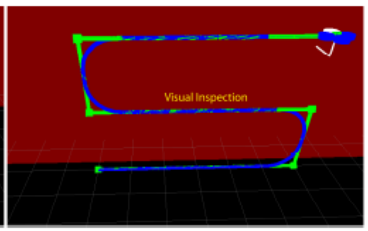




AIRobots Domain

- Remote inspection of industrial plants with ASV
- The ASV operates in proximity and in contact with the surfaces:
  - visual inspection, contact, manipulation, sample picking
- The autonomous control system should supervise and orchestrate a new set of operations:
  - Not only free-flight navigation, but also wall approach, docking, undocking, wall scanning, wall sliding, simple manipulation, etc..



Requirements

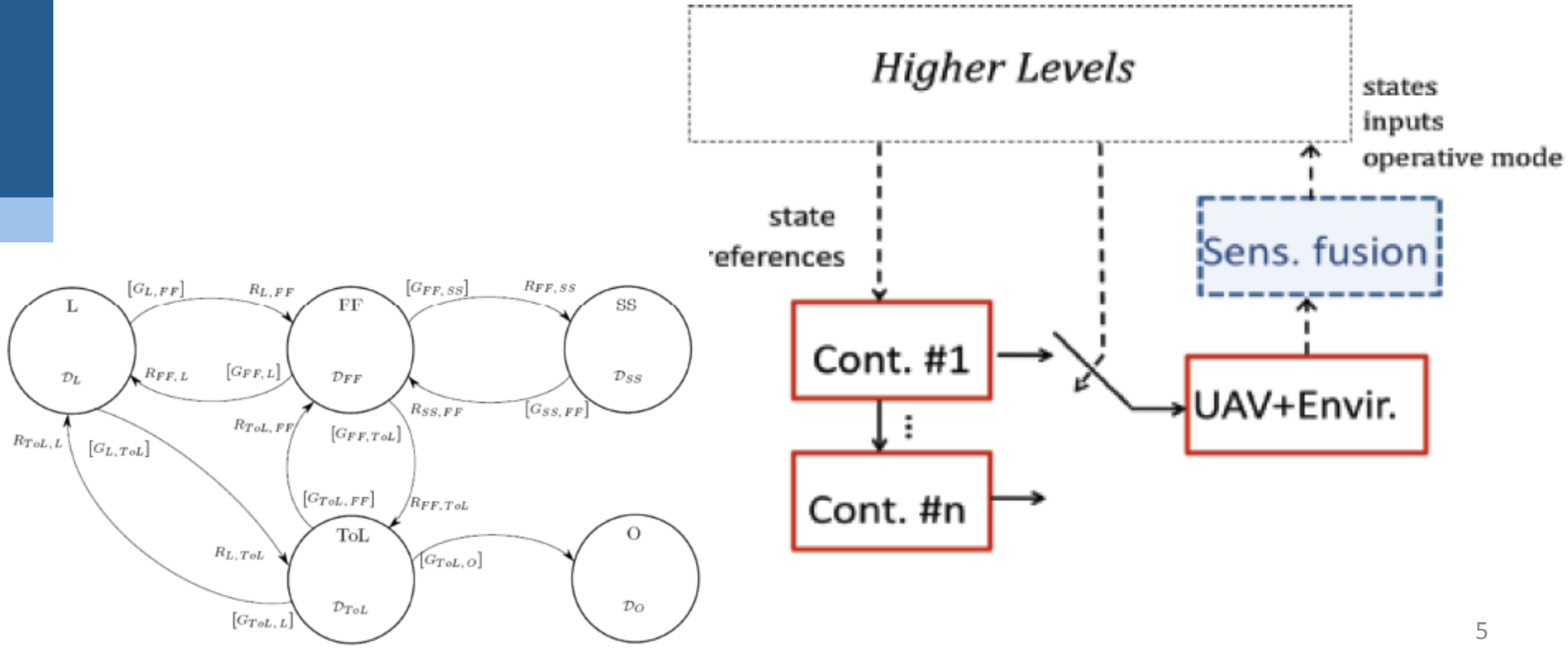
- **Reaction time and replanning:**
  - close interaction with the environment, hence reactive, adaptive, and flexible planning/ replanning capabilities are needed (task/ path/motion).

- **Sliding Autonomy and Mixed Initiative Control:**
  - Both autonomous and human-in-the-loop control modalities should be supported to allow human interventions and teleoperation.

- **High-level/Low-level integration:**
  - High-level control strategies should be defined taking into account the low-level operative modes and constraints;
  - Different control modes, smooth task switching, adjustments on-the-fly.

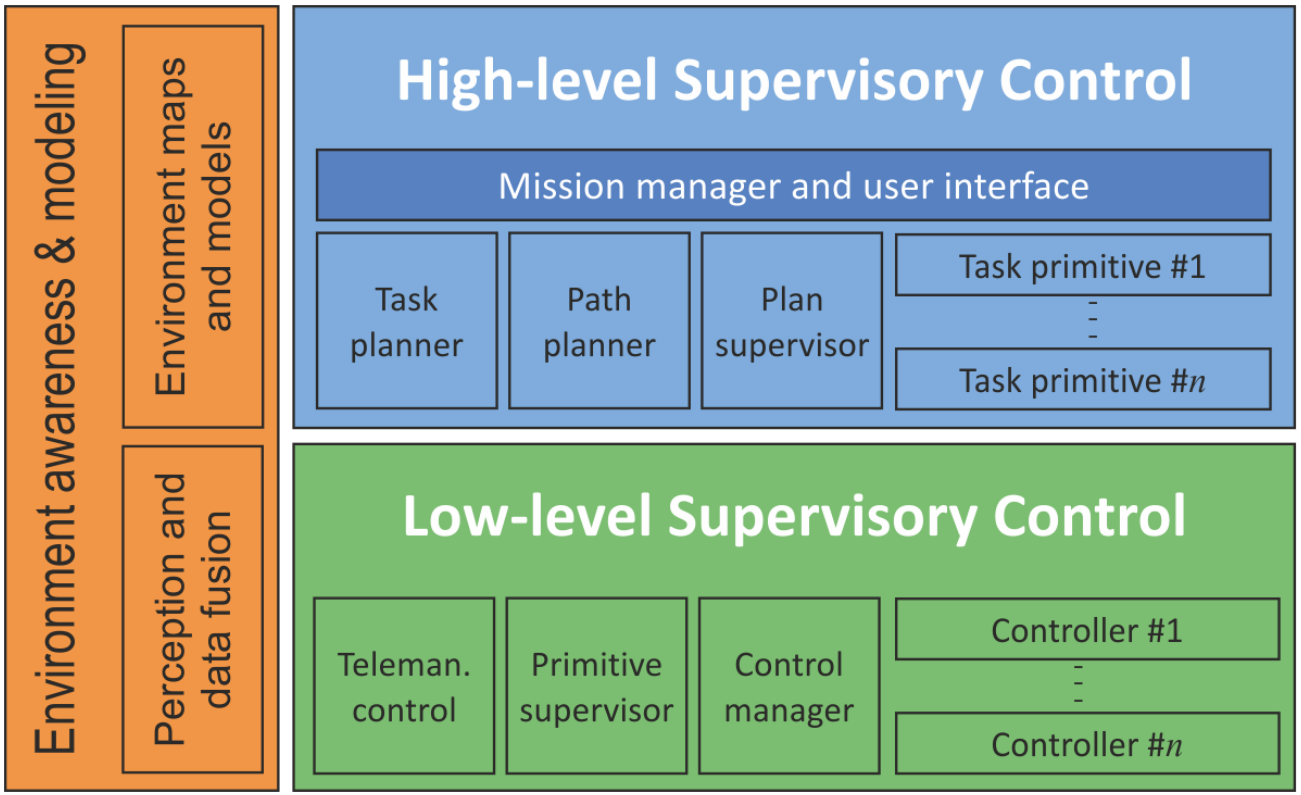
High-level/Low-level integration

- The LL control system can be modeled as a hybrid automaton [Naldi, Marconi, Gentili 2011].
- Each motion primitive has at least one controller for each operative mode.
- More controllers could be designed for a certain motion primitive depending on dynamic parameters (max velocity, maximum acceleration, etc).

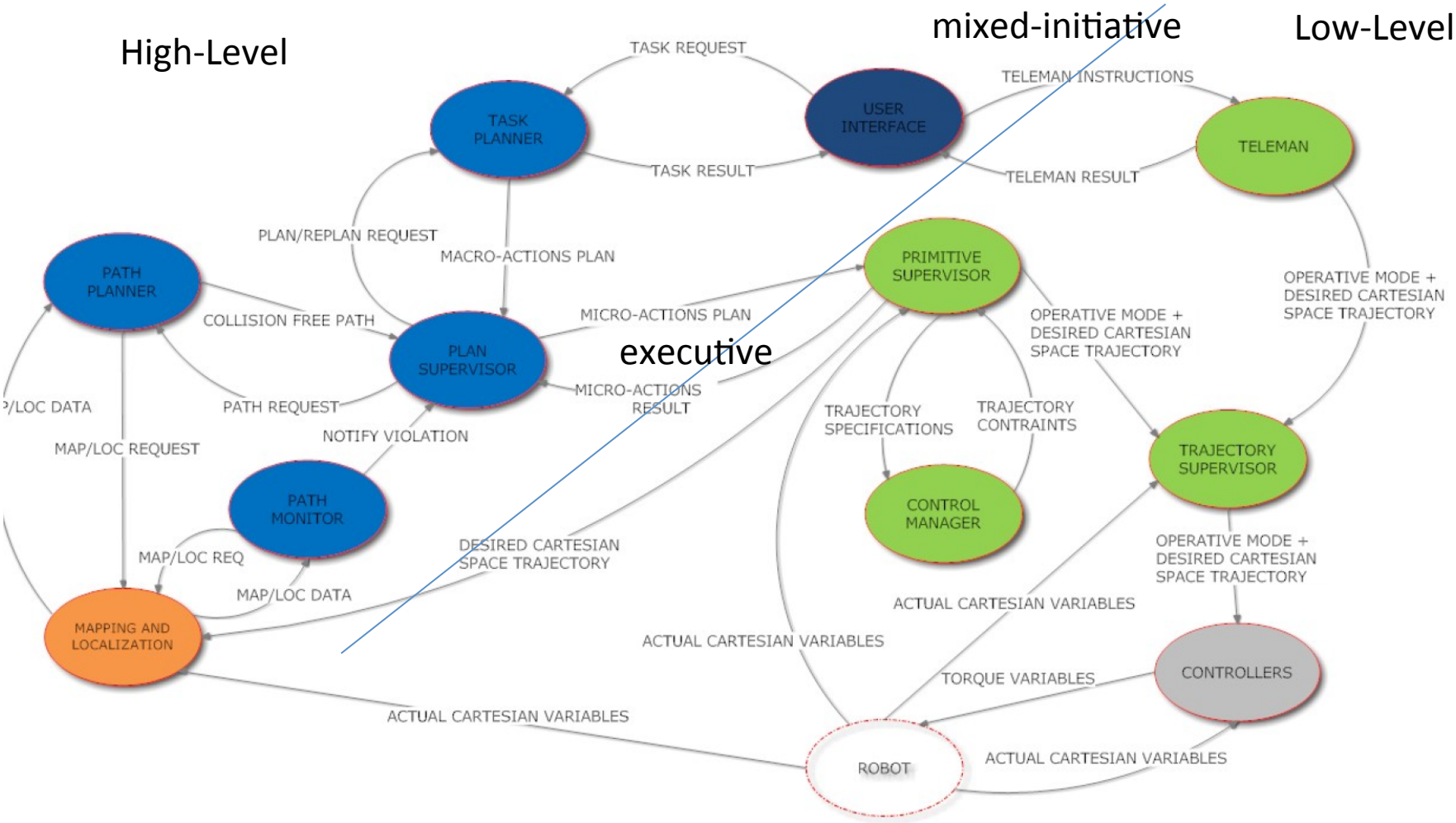


System Architecture

- Path Planner, Task Planner, Plan Supervisor (High Level), Primitive Supervisor manages trajectory planning and execution (Low Level);
- Switches between operative modes depends on HL system decisions;
- Choice of controllers in the current operative mode assigned to the Control Manager of the Low Level Supervisory Control



System Architecture



Task Decomposition

- Task Primitives (out: Task Planner, in: Path Planner):

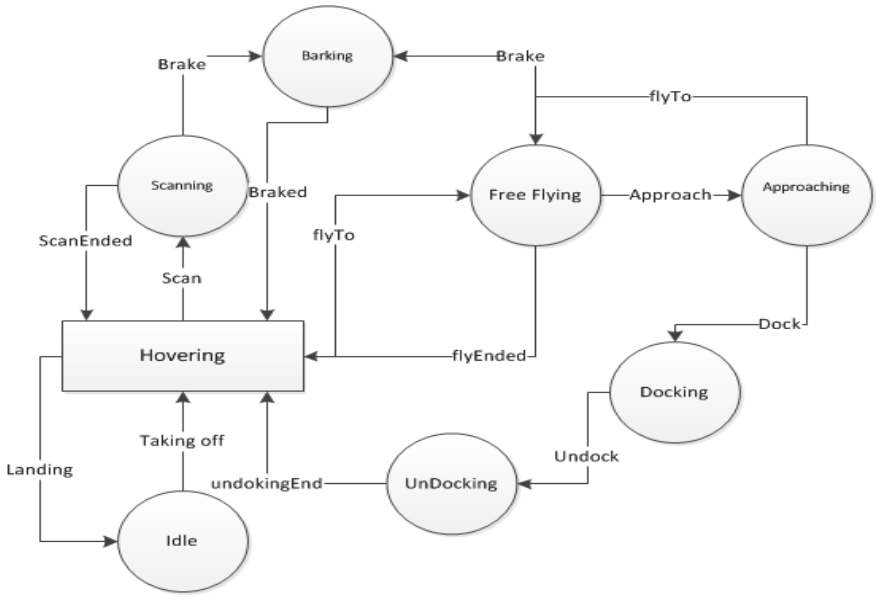
$TakeOff(C, Pos)$	Take off from the current pose and hover in the pose Pos;
$Land(C, Pos)$	Land from the current position to Pos;
$Hover(C, Pos)$	Keep the pose Pos;
$MoveTo(C, Pos)$	Move from the current pose to Pos;
$MoveCircular(C, Pos, I)$	Execute a circular movement around the center P with radius in the interval I ;
$Scan(C, Srf)$	Scan the surface Srf;
$Inspect(C, Obj, P)$	Observe the object Obj in position P;
$Brake(C)$	Execute a hard brake from the current position;
$Approach(C, P)$	Approach the target position P;
$Dock(C, P)$	Dock to a target position P;
$UnDock(C)$	Undock from the current position;
$Manipulate(C, Obj, P)$	Manipulate an object Obj in position P.

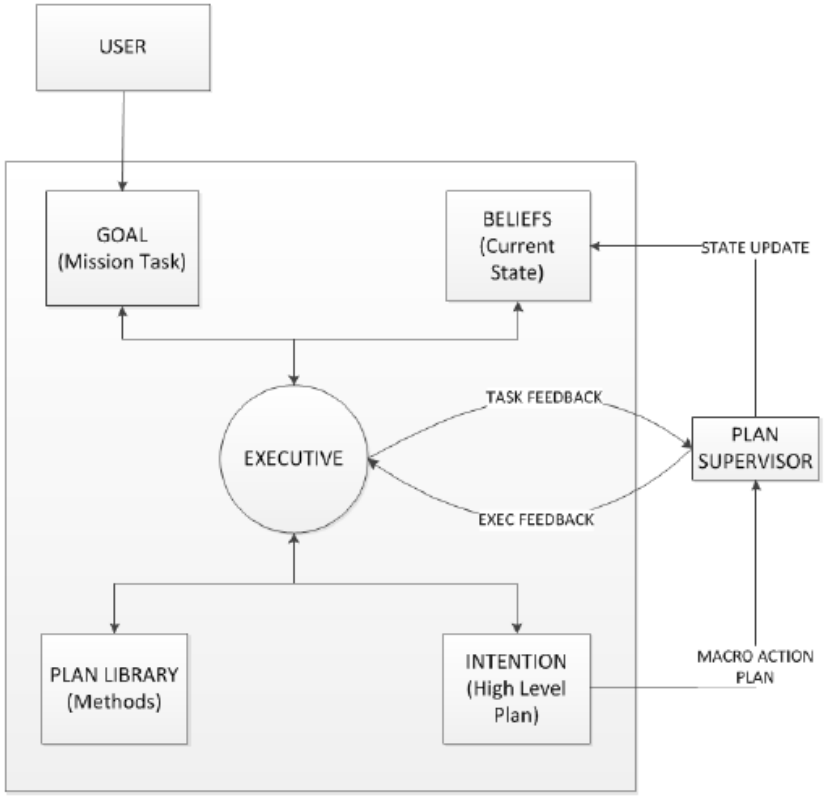
- Motion Primitives (out: Path Planner, in: Motion Planner):

$TakeOff(p)$	take off and hover at the altitude p
$Land$	land from the current position
$FlyLinearTo(\vec{ps})$	free flight from an initial pose till the final one through a set of poses $\vec{ps}$.
$FlyCircular(p_1, p_2, p_3)$	circular flight along the circumference passing through the points p_1, p_2, and p_3.
$FlyArc(p_1, p_2, p_3)$	circular flight along the circumference arc from p_1 to p_2, through p_3.
$Brake$	hard brake.
$Escape(\vec{ps})$	escape from the current position following the waypoints in the set of poses $\vec{ps}$.
$ReplanFlyLinearTo(\vec{ps})$	free flight replan following the set of poses $\vec{ps}$.
$Approach(p, d)$	approach the wall towards the point p and hover at distance of d (before docking).
$Docking$	dock to the wall.
$UnDocking$	undock from the wall.

High-Level Execution Cycle

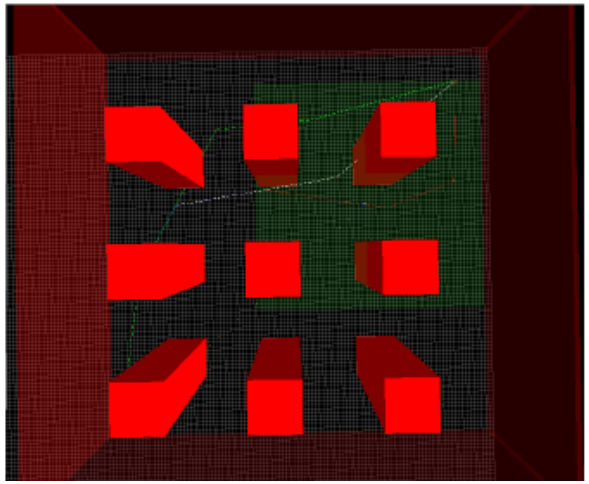
- The high-level executive system exploits a Belief-Desire-Intention (BDI) architecture to coordinate task decomposition, task replanning, path planning, and plan monitoring:
  - Belief base as the current abstract state, plan library, goals are mission tasks, and the intention base rap. the plan to be executed;
  - BDI engine based on PRS.

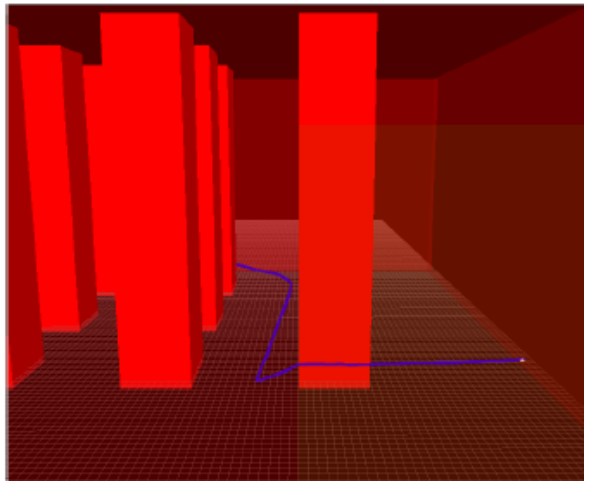




Path Planner

- The Path Planner expands each macro-action into micro-actions;
- Rapidly-exploring Random Tree (RRT) [Lavalle 98]
  - non-convex, high-dimensional search space, efficient, sub-optimal, plan refinement, replanning;
- Discretized 4D search space (N x M x Q grid map + yow)
- Constraints:
  - Min distance from obstacles;
  - Max angles (yow, pitch);
  - Max time to compute (timeout);
  - Cost threshold.



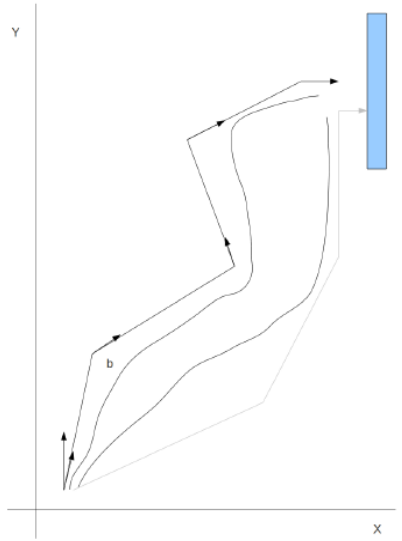


Path Planner

- The RRT planning process generates several solutions to refine the current one (until *cost threshold* or *timeout*)

Algorithm 1 Refine_RRT(q_{init},q_{goal},$threshold$,$timeout$)

initialize($path$,$time$);
while $((time < timeout) \wedge (preempted = false) \wedge (pathCost \geq threshold))$ **do**
  $newPath \leftarrow$ solveRRT(q_{init},q_{goal},$timeout$);
  if $C(newPath) < path$ **then**
    $path \leftarrow newPath$;
    $pathCost \leftarrow C(newPath)$;
  end if
end while
return $path$



- Optimization function minimizes:

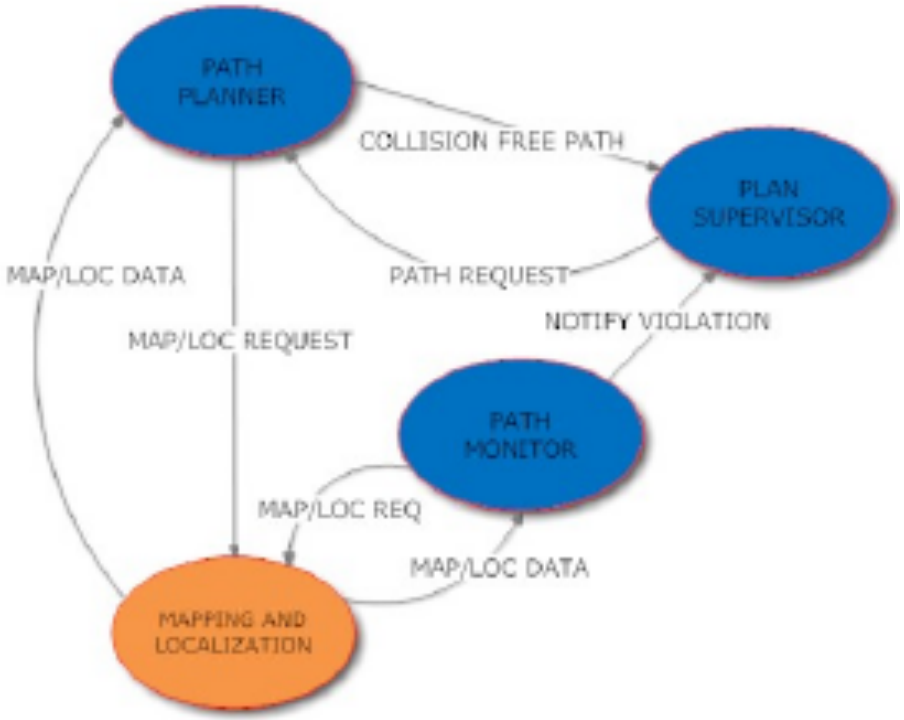
$$\mathbf{c}(path) = \mathbf{c}_{lng}(path) \cdot p_{lng} + \mathbf{c}_{ang}(path) \cdot p_{ang} + \mathbf{c}_{way}(path) \cdot p_{way} + \mathbf{c}_{obs}(path) \cdot p_{obs} + \mathbf{c}_{unk}(path) \cdot p_{unk}$$

i.e. path length ; angular variations; number of waypoints; obstacle distance; unknown space

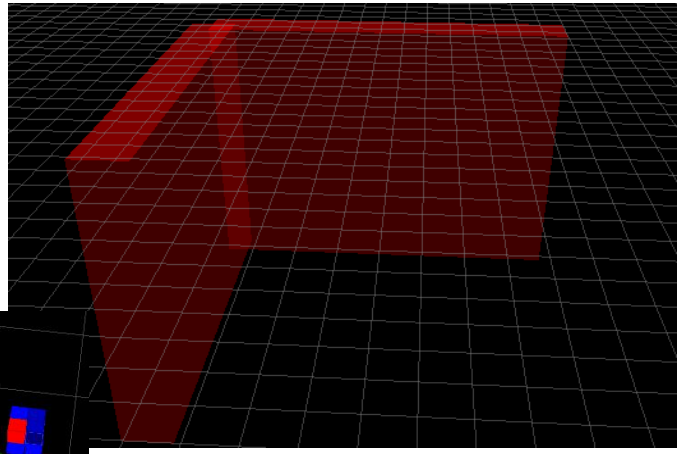
- Constraints for each segment (max speed, min distance, error tolerance).

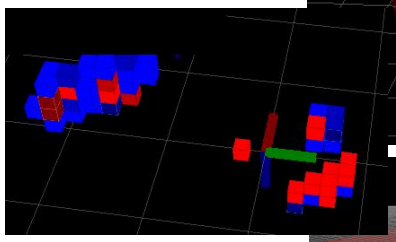
Mapping and Execution

- The 3D grid-map is continuously updated (stereoscan [Geiger et al. 2011])
- The path monitor checks for deviations or collisions
- If collision/deviation detected then replanning

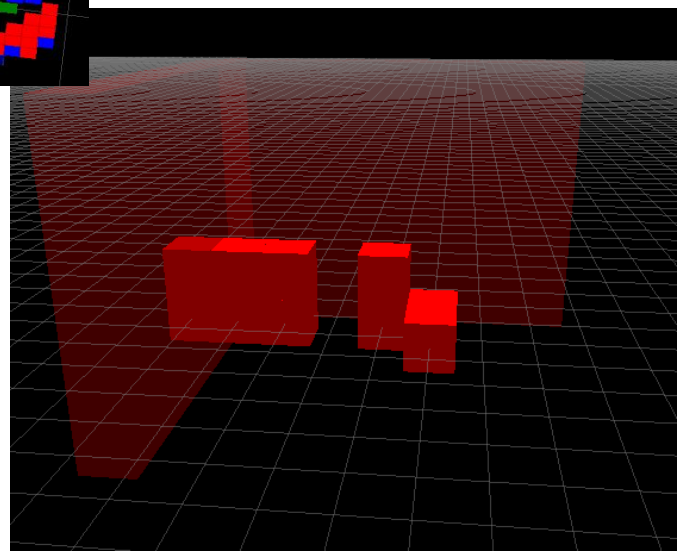


Old map





New data



New map

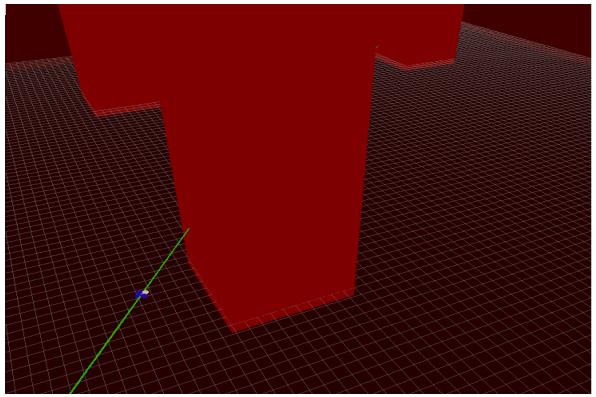
Path Replanning

- Path replanning is managed with different strategies depending on the time available for path generation;
- Given the time to collision t_c we distinguish the following cases:
  - if $(t_c < t_b)$ Brake
  - if $(t_c < t_e)$ Escape
  - Otherwise Replan

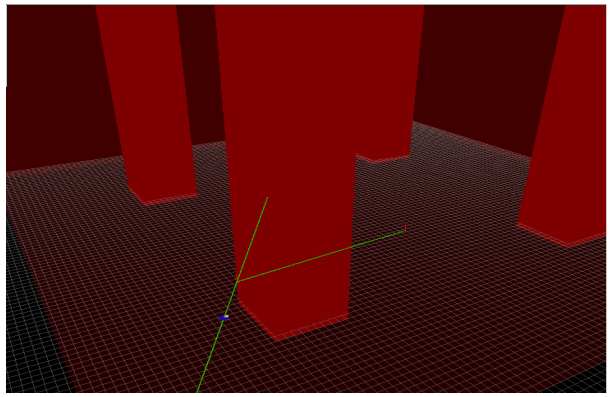
t_c = collision time

t_b = time to brake

t_e = time to escape

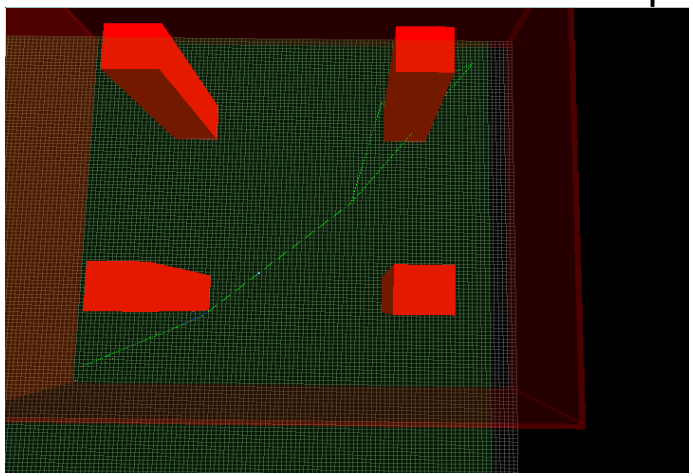


Brake



Escape

Replanning



Path Replanning

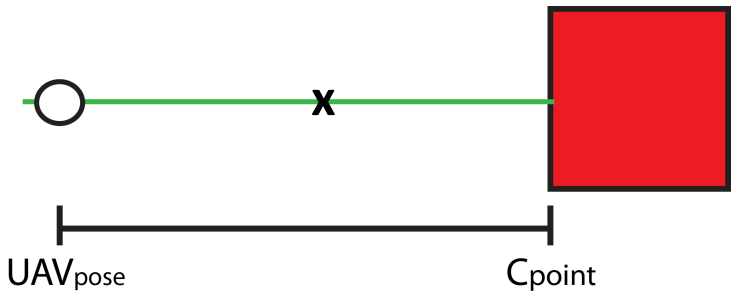
- Path replanning should find an alternative path that connects the old trajectory with a new one_while the robot is flying;

Algorithm 3 Replan(q_{goal}, $path_{old}$, q_{obs}, t_{ttc})

$q_c \leftarrow$ getPosition();
$t_{rp} \leftarrow$ estimatedRepTime(q_c, q_{goal}, $path_{old}$, q_{obs});
$wp_{rp} \leftarrow$ selectDeviationWP(q_c, q_{obs}, $path_{old}$, t_{rp});
$threshold \leftarrow$ setThreshold(wp_{rp}, q_{goal}, t_{rp}, t_{ttc});
$path_{new} \leftarrow$ Refine_RRT(wp_{rp} , q_{goal}, $threshold$, t_{rp});
return $path_{new}$

- Deviation wp:
  - Given the estimated time t_{rp} for replanning, a deviation wp_{rp} is selected: far enough from q_c to replan from wp_{rp}, but not too close to the obstacle

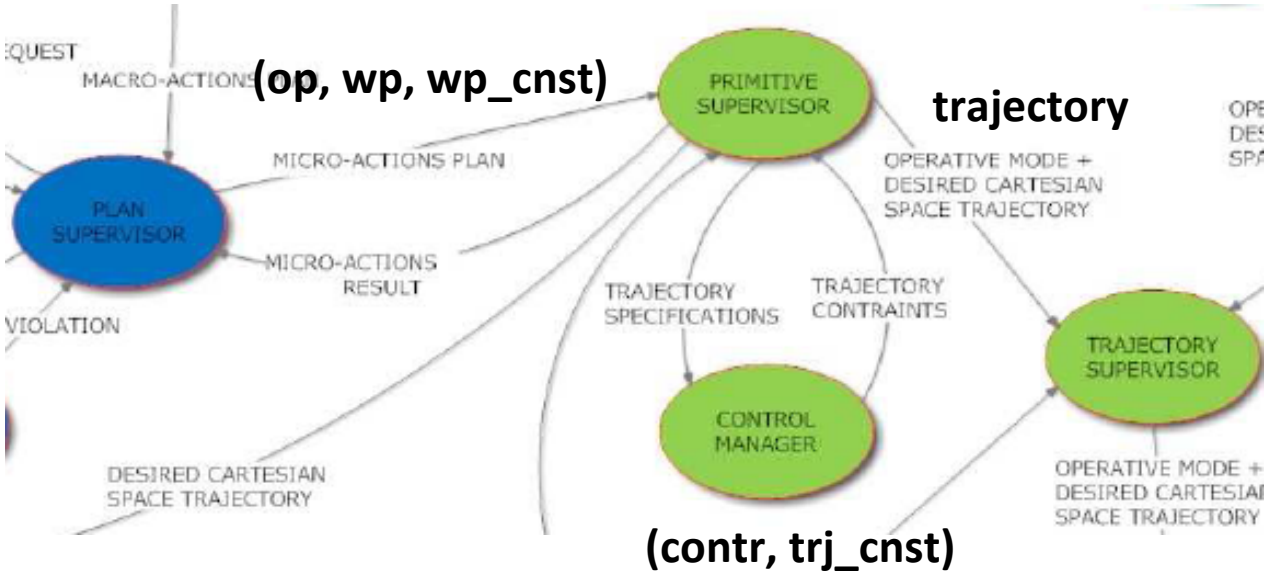
- Video



Low Level Supervisory Control

The **Primitive Supervisor** receives a sequence of micro-actions and constraints from the **Plan Supervisor** and selects the controller, generates and executes the trajectory.

The **Control Manager** selects the right controller depending on the waypoint constraints and the operative mode



Low Level Supervisory Control

The **Control Manager** selects the right controller depending on the constraints and the operative mode:

- Each controller has trajectory constraints (maximum velocity, acceleration, etc.) to guarantee the maximum position error and cruise velocity

- The Control Manager sends to the Primitive Supervisor the controller and the associated trajectory constraints so that the trajectory can be generated.

	e_{max} $[m]$	$\dot{x}_{max}$ $[m/s]$	$\dot{y}_{max}$ $[m/s]$	$\dot{z}_{max}$ $[m/s]$	$\ddot{x}_{max}$ $[m/s^2]$	$\ddot{y}_{max}$ $[m/s^2]$	$\ddot{z}_{max}$ $[m/s^2]$	$\dddot{x}_{max}$ $[m/s^3]$	$\dddot{y}_{max}$ $[m/s^3]$	$\dddot{z}_{max}$ $[m/s^3]$
Cont.1	0.25	0.5	0.5	0.3	0.2g	0.2g	0.2g	2000	2000	2000
Cont.2	0.15	0.25	0.25	0.2	0.1g	0.1g	0.1g	1000	1000	1000
Cont.3	0.1	0.1	0.1	0.1	0.05g	0.05g	0.05g	500	500	500

	$\Delta\theta_{max}$ $[rad]$	$\dot{\theta}_{max}$ $[rad/s]$	$\ddot{\theta}_{max}$ $[rad/s^2]$	$\dddot{\theta}_{max}$ $[rad/s^3]$
All	0.1	0.2	0.2g	2000

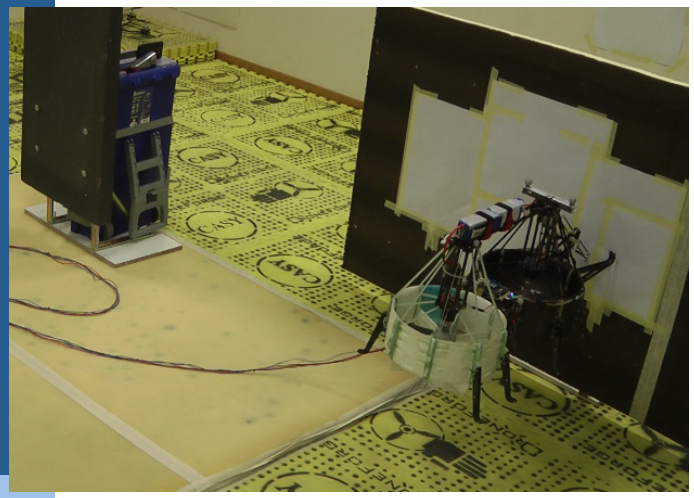
Motion Planning

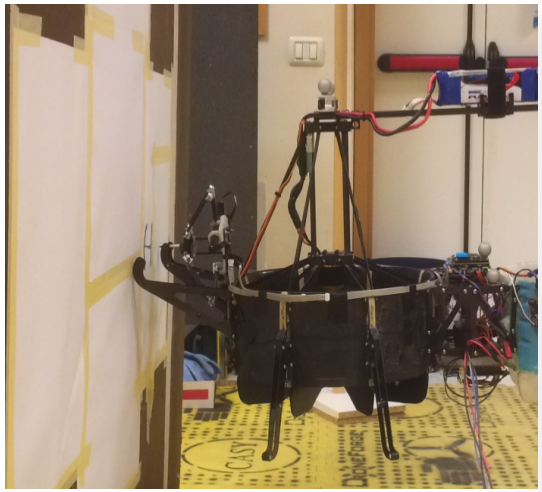
- **Constraints:**
  - Smoothness: The trajectory needs to be continuous up to the acceleration;
  - Limits: Limits on the absolute values of velocities, accelerations and jerk;
  - On-line and incremental: Both a point-to-point and fly motions; addition of new points for a multi-fly movement without modifying the already generated trajectory.

- **Approach:**
  - The motion trajectory is generated as fifth-order polynomials concatenations [Macfalane Croft 2003] (smoothness, jerk-bounded, on-line)
  - Number of waypoints can be incrementally added (dynamic and incremental). This horizon is regulated by the Plan Supervisor.

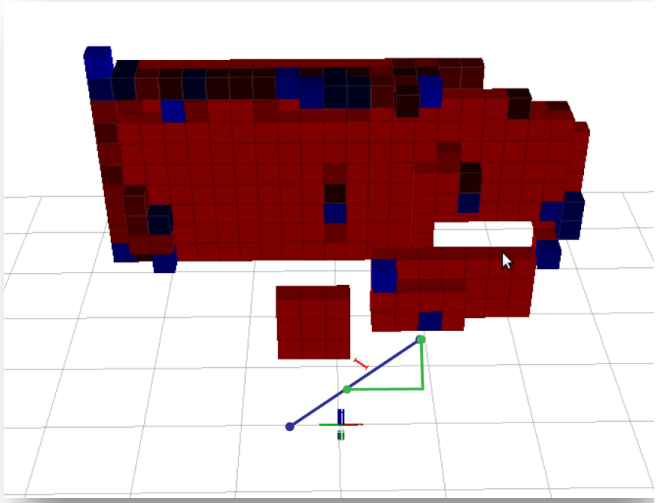
Test: Inspection Tasks

- Inspection tasks (physical contact, visual inspection)



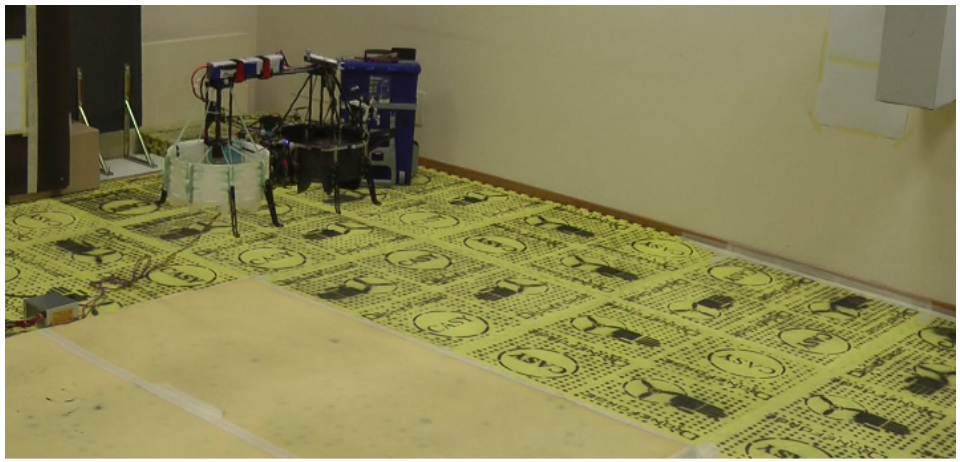


Docking and Manipulation in the real scenario



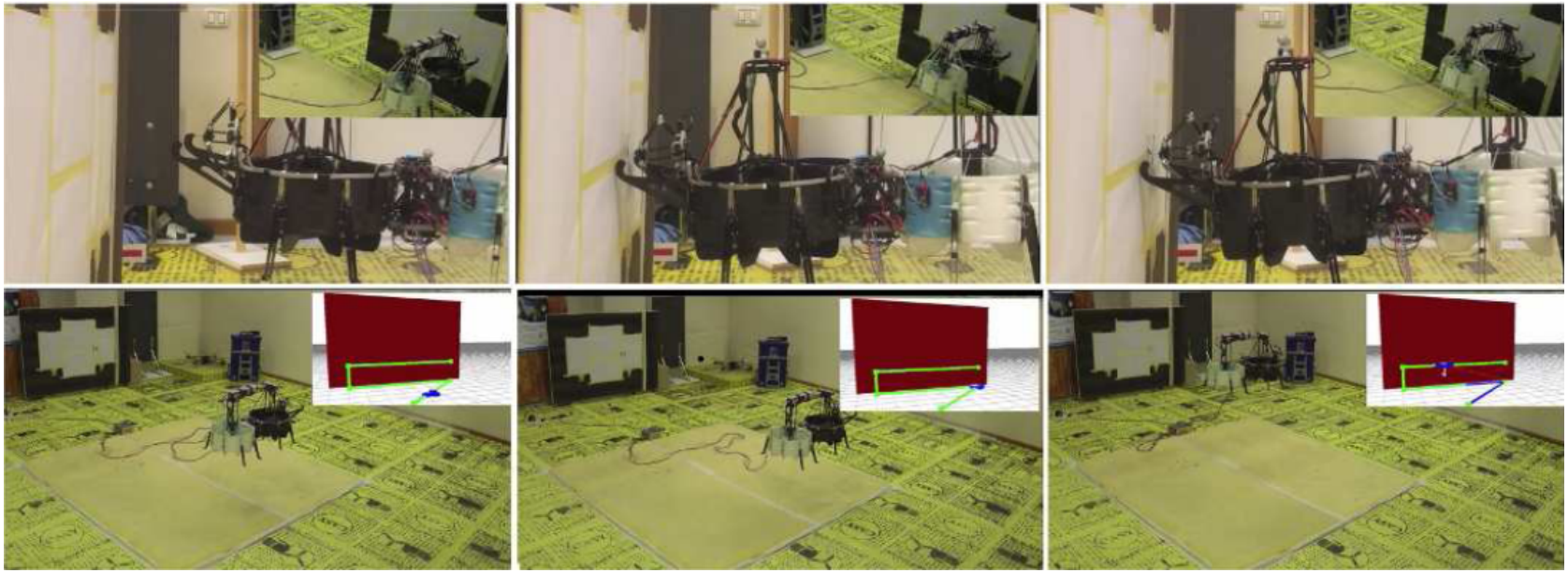
Replan, Execute, and Docking

Wall detection and Visual Inspection in the real scenario



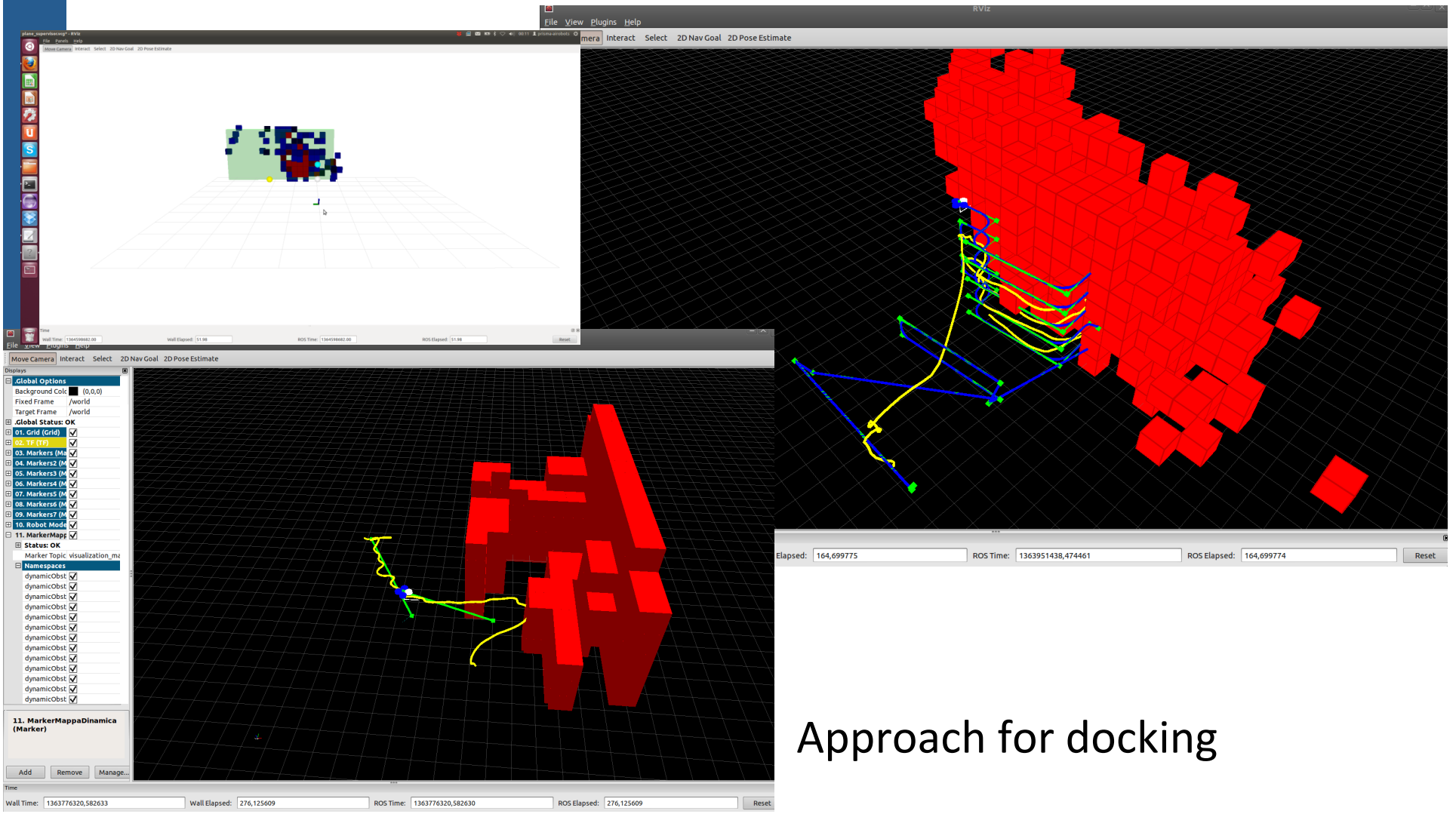
Test: Inspection Tasks

- Docking and physical inspection
- Wall detection and visual inspection



Test: Inspection Task

Strategy for wall detection and visual inspection



Approach for docking

Conclusion

- Aerial Service Robotics is a novel application for plan-based autonomy;
- We presented the challenges of the ASV domain along with the solutions provided within the AIRobot project;
- The proposed high-level system combines hierarchical task decomposition, mixed-initiative control, BDI execution, RRT path planning/replanning to allow reactivity, flexibility, and sliding autonomy;
- Future work:
  - Deeper integration between high-level and low-level system;
  - Learning methods for parameters setting;
  - More complex mixed initiative and sliding autonomy.

SHERPA

Smart collaboration between **H**umans and ground-a**E**rial **R**obots for im**P**roving rescuing activities in **A**lpine environments