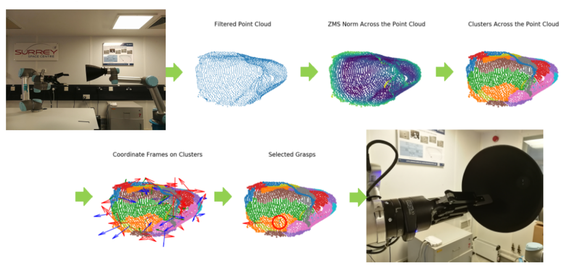
Formal Modelling and Runtime Verification of Autonomous Grasping for Active Debris Removal

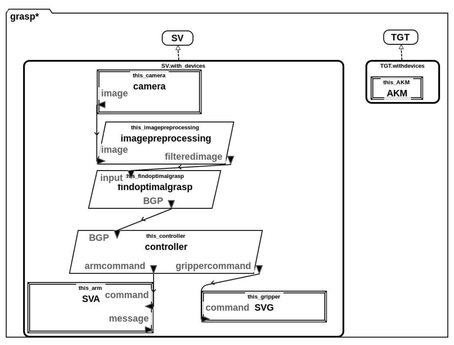
Marie Farrell **Nikos Mavrakis** Angelo Ferrando Clare Dixon Yang Gao

Introduction

- **Active debris removal** in space is a necessary activity to maintain and facilitate orbital operations. Current approaches adopt a**utonomous robotic systems which are furnished with a robotic arm** to safely capture debris by **identifying a suitable grasping point**.
- **Formal verification** methods enable us to analyse the software that is controlling these systems and to provide a **proof of correctness** that the software obeys its requirements.
- We describe the process that we used to **verify a pre-existing system for autonomous grasping** which is to be used for active debris removal in space.

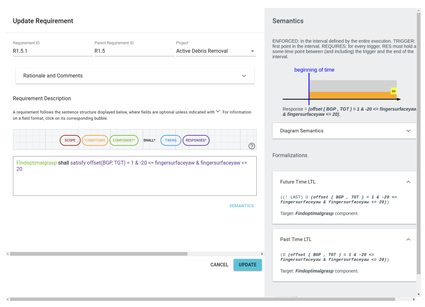


AADL



- Both the **hardware and software components** of the service vehicle (`SV`) and the target (`TGT`).
- `SV` contains a `camera`, arm (`SVA`) and gripper (`SVG`).
- Software components preprocess the input image (`imagepreprocessing`), calculate the optimal grasp (`findoptimalgrasp`) and control the arm and gripper (`controller`).

Formal Requirements Elicitation Tool (FRET)



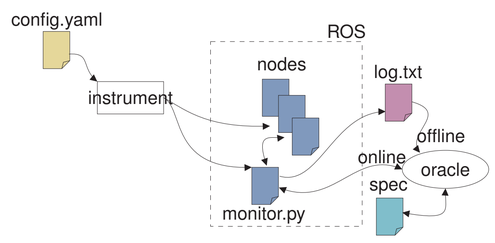
- Supports the **formalisation**, **understanding** and **analysis of requirements** through a user-friendly interface with intuitive diagrammatic explanations of requirement semantics.
- Users specify their requirements in **restricted natural language**, called FRETISH, which embodies a **temporal logic semantics**.

`SCOPE CONDITION COMPONENT SHALL TIMING RESPONSE`

Verification: Dafny and ROSMonitoring

```
datatype Point = Point(x:real, y:real, z:real)
datatype Grasp = Grasp(x:real, y:real, z:real, qx:real, qy:real, qz:real, qw:real)
datatype Score = Score(areaScore:real, angleScore:real, index:int)

method imagepreprocessing(t:real, p:array<Point>, v:real, nb:int, rf:real)
  returns (filteredimage:array<Point>)
requires 0 < p.Length; // R1.3.2
requires v > 0.0;
ensures filteredimage.Length ≤ p.Length;//R1.4
ensures filteredimage.Length > 0 ; // R1.4
{
 filteredimage := removeDepth(p,t); //remove distant points from p.
 filteredimage := downSample(filteredimage,v); //make voxel representation of p.
 filteredimage := filter(filteredimage,nb,rf); //removes noise and speckles from p.
}
```

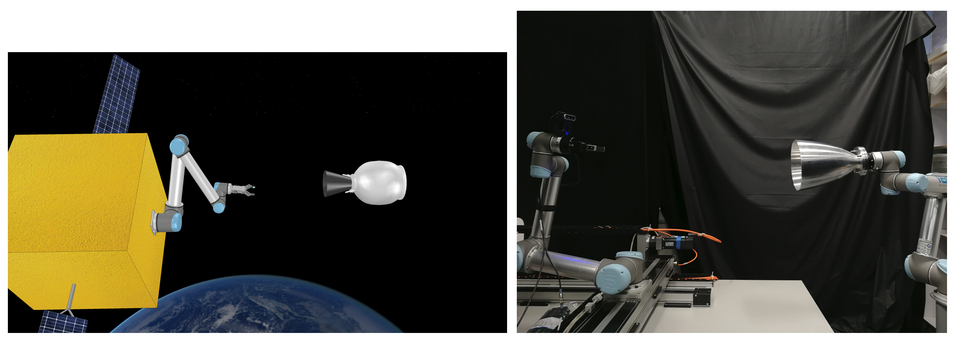


- **Static verification** with Dafny.
- **Runtime verification** with ROSMonitoring.

Requirements Elicitation: 20 Requirements

ID	English-Language Description	FRET Formalisation
R1	The SV shall grasp the TGT at the BGP and draw it closer.	SV shall satisfy (grasp(TGT, BGP) & closer(SV, TGT))
R1.1	The Camera of the SV shall be positioned at least 0.5m from the TGT.	Camera shall satisfy distance(Camera, TGT) $\geq$ 0.5
R1.2	The TGT shall be motionless before contact with the SVA.	TGT shall satisfy if !contact(SVA, TGT) then motionless(TGT)
R1.3	The Camera shall return a valid point cloud.	Camera shall satisfy valid(p)
R1.3.1	The point cloud shall be structured with maximum resolution of 1280 $\times$ 720.	Camera shall satisfy maxRes(p) = 1280*720
R1.3.2	The point cloud shall not be empty.	Camera shall satisfy length(p) $>$ 0
R1.4	The imagepreprocessing shall return a filtered point cloud.	Imagepreprocessing shall satisfy length(filteredimage) $\leq$ length(p) & length(filteredimage) $>$ 0
R1.5	findoptimalgrasp shall return the optimal grasp point (BGP) if one exists.	Findoptimalgrasp shall satisfy if exists(BGP) then return(BGP)
R.1.5.1	The BGP shall be optimal according to the criteria: minimum offset from the TGT nozzle edge of 1cm and finger-surface yaw angle between -20 and 20 degrees.	Findoptimalgrasp shall satisfy offset(BGP, TGT) = 1 & -20 $\leq$ fingersurfaceyaw & fingersurfaceyaw $\leq$ 20
R1.5.2	findoptimalgrasp shall generate several candidate grasping points.	findoptimallgrasp shall satisfy length(grasps) $\geq$ 0
R1.6	If no BGP exists then findoptimalgrasp shall output an error message.	Findoptimalgrasp shall satisfy if !(exists(BGP)) then printerror
R1.7	Controller shall execute a joint trajectory to reach the BGP.	Controller shall satisfy executeJointTrajectory(SVA, BGP)
R1.8	The SVA shall capture the TGT at the BGP.	SVA shall satisfy captured(TGT) $\Rightarrow$ contactpoint(SVA, TGT) = BGP
R1.9	The total pulling distance shall be between 0.3 and 0.5m.	SV shall satisfy totalpullingdistance $\geq$ 0.3 & totalpullingdistance $\leq$ 0.5
R2	The SV shall not collide with the TGT.	SV shall always satisfy !collide(SV, TGT)
R2.1	The position of the SV shall not be equal to the position of the TGT.	SV shall always satisfy !(position(SV) = position(TGT))
R2.2	The SV shall only make contact with the TGT at the BGP using the SVG.	SV shall always satisfy contactpoint(SVG, TGT) = BGP.
R2.2.1	No part of the SV, other than the SVG shall make contact with the TGT.	SV shall satisfy if !grasped then contactpoint(SV, TGT) = null
R2.2.2	The SVG shall only make contact with the TGT at the BGP (within some margin of error).	SV shall satisfy if grasped then contactpoint(SVG,TGT) = BGP + errormargin
R2.3	The SVG shall apply a force of 180N once contact has been made with the TGT.	SVG shall satisfy captured(TGT) $\Rightarrow$ force = 180

Experimental Results: Simulation and Physical Testbed



- Intentionally **injected a fault** into the system.
- We reduced the grasping force used by the gripper to grasp the target, substantially less than the lower limit of R2.3.
- Applied force was not able to hold the target because it slipped through the gripper fingers, and the SV lost contact with the TGT.
- **Fault correctly identified** by the monitors for R1.9 and R2.3.

Gaps in the Requirements

- Monitors helped to identify **gaps** in the requirements.

R1.9: *The total pulling distance shall be between 0.3 and 0.5 m.*
R2.3: *The SVG shall apply a force of 180N once contact has been made with the TGT.*

- Satisfied in simulation but not on the physical testbed.
- Cause: **hardware limitations**.
- We **could not formally verify** three of the requirements

R1.1: *The Camera of the SV shall be positioned at least 0.5m from the TGT.*
R1.2: *The TGT shall be motionless before contact with the SVA.*
R1.7: *Controller shall execute a joint trajectory to reach the BGP.*

Post-Implementation Verification

- System was **nearly complete** when we were asked to verify it.
- We were able to **reverse engineer** our verification method.
- Made **minor adjustments to the software** to expedite verification.
- Having an **implementation** to evaluate against was beneficial.
- Implementation and verification artefacts **informed each other**.
- **Modularity** was key.

Our Paper

SCAN ME

Farrell, M., Mavrakis, N., Ferrando, A., Dixon, C., & Gao, Y. (2021). Formal Modelling and Runtime Verification of Autonomous Grasping for Active Debris Removal. *Frontiers in Robotics and AI*, 8.

Work supported by grants EP/R026092 (FAIR-SPACE) and EP/R026084 (RAIN) through UKRI under the Industry Strategy Challenge Fund (ISCF) for Robotics and AI Hubs in Extreme and Hazardous Environments.