Hybrid Planning to Minimize Platform Disturbances during In-orbit Assembly Tasks

Ismael Rodríguez*
German Aerospace Center (DLR)
Institute of Robotics and Mechatronics
Münchner Str. 20, 82234 Weßling, Germany

Technische Universität München
Faculty of Informatics
Garching, Germany
ismael.rodriguezbrena@dlr.de

Jean-Pascal Lutze*, Hrishik Mishra,
Peter Lehner, Máximo A. Roa
German Aerospace Center (DLR)
Institute of Robotics and Mechatronics
Münchner Str. 20, 82234 Weßling, Germany

{Jean-Pascal.Lutze, Hrishik.Mishra,
Peter.Lehner, Maximo.Roa}@dlr.de

Abstract— **In-orbit space assembly has been proposed as a method to overcome the obstacles for deployment of large spatial structures. To make such assemblies economically feasible, they must rely on robotic arms to perform the required manipulation actions. The operations with the robotic arm inevitably affect the attitude and orientation of the spacecraft. This influence is well understood for simple trajectories; however, assembly sequences for full structures require multiple repetitive motions, with and without load, which significantly affect the attitude and orbital control of the satellite. This paper analyzes such perturbations for a complex assembly task, the construction of the primary mirror for a space telescope, using a hybrid planner with two levels: a low level that considers individual motions of the robotic arm, and a high level that generates the overall assembly sequence while minimizing the perturbations created on the attitude control system. The method effectively minimizes perturbations during orbital assembly tasks, therefore minimizing fuel or energy consumption in the spacecraft.**

TABLE OF CONTENTS

1. INTRODUCTION

The deployment of large structures in space is limited by the size of the cargo areas in existing launchers. In-orbit assembly has emerged as a possibility to allow construction of such large structures, by launching the required materials or modules with standard carriers, and performing the assembly directly in space [1]. Although such assembly operations could be performed by astronaut workforce, as in the case of the International Space Station (ISS), this requires highly demanding and complex operations from the astronauts, and results in very expensive missions for completing such constructions. Autonomous robotic assembly holds the promise of decreasing the need of astronaut workforce for such tasks.

The operations with a robotic arm mounted on a free floating base in space significantly affect the attitude and orientation of the satellite. This influence is well understood for simple point to point motions, where the arm path can be computed in such a way that it minimizes the perturbations on the base [2], [3]. However, for assembling more complex structures, such as a space telescope, the manipulator must execute multiple motions, with and without additional loads. In addition, the inertial properties of the satellite are modified as the assembly process progresses, i.e., the center of mass and inertial properties of the satellite and assembled structure change over time [4]. This requires the consideration of the perturbations to the floating base within the in-space assembly planning problem.

In this work we propose a two-level planner to solve the problem of minimizing the perturbation on the spacecraft during a space-based assembly task. As a particular example we consider the assembly of the primary mirror for a space telescope using segmented mirror tiles [5]. The proposed planner ensures that the generated assembly sequence fulfills constraints such as mechanical stability, power transmission and data interconnectivity during the assembly process, which can be considered through a semantic constraint representation. The planner also computes the manipulator joint motions in order to minimize the perturbations on the base or to guarantee that such disturbances are within prescribed values. The planner is composed by two levels: a high-level planner that creates the overall assembly plan, and a low-level planner that computes the manipulator joint motions to be executed. The high-level planner constructs a graph representation on the assembly process and assigns costs to the transitions between different sub-assemblies (sub-states). The costs are later used to find the cost-optimal assembly plan. In addition, this representation allows the identification of repetitive sequences using pattern matching algorithms, which leads to a reduction of the overall planning time. The low-level planner implements a path planner based on Stochastic Trajectory Optimization for Motion Planning (STOMP, [6]), to provide the individual manipulator joint motions during the full assembly process. This algorithm uses a cost function based on a dynamic model of the system, which computes the disturbances created on the free floating system by the robotic arm displacements.

The low-level planning tool is tested on hardware in the OOS-SIM (On-Orbit Servicing Simulator, [7]) at the Ger-

* Equal contribution to this work

 DOI: 10.1109/AERO53065.2022.9843530

man Aerospace Center (DLR). The tests verify the dynamic model used during the computation of the cost function for the STOMP algorithm. A dynamic simulation verifies the correct behavior of the high-level assembly planner. The simulated experiments show the flexibility of our system by providing feasible plans for different assemblies. The created plans minimize the base perturbations, and thus consume less spacecraft fuel, when compared to an assembly plan focused only on the robotic operations.

2. RELATED WORK

We identify three main categories of related work: Systems targeting autonomous assembly in space, algorithms that employ hybrid planning, and algorithms for optimizing manipulator trajectories.

Autonomous assembly in space

On-orbit (free-flying) robotic systems include a satellite, equipped with thrusters and/or reaction wheels, that carries (at least) one robotic manipulator. These space manipulator systems have been mainly constructed for on-orbit servicing missions, such as ETS-VII [8] and Orbital Express [9]. A well-known example is the Special Purpose Dexterous Manipulator (SPDM), or Dextre [10], a dual arm robot mounted on the ISS that has also been used for servicing and maintenance tasks on the ISS and for on-orbit servicing as in the Robotic Refueling Mission (RRM) [11]. Although mainly used for servicing so far, these space-based robotic manipulators have the potential to carry out more complex tasks, including autonomous assembly. Initial ideas about autonomous assembly in space were proposed in the early 80s [12], [13]. Several concepts followed this direction. For instance, a proposal for hybrid in-space fabrication and assembly using a multi-arm robot was developed by Tethers Unlimited with the SpiderFab [14]. NASA is expanding the work in this area with the OSAM-1 and OSAM-2 missions. OSAM-1 (formerly known as Restore-L) is focused on rendezvous, grasp and refueling of a satellite using the SPIDER (Space Infrastructure Dexterous Robot) robotic arm (developed in the former project Dragonfly). OSAM-2 (formerly known as Archinaut) combines additive manufacturing with robotic assembly for creating large space structures. In Europe, several projects took initial steps in the direction of On-Orbit Assembly, such as PULSAR (Prototype of an Ultra Large Structure Assembly Robot, [15]) and MOSAR (MOdular Spacecraft Assembly and Reconfiguration, [16]). Currently, two parallel projects are developing phases A/B1 for on-orbit servicing missions: PERIOD (PerAspera In-Orbit Demonstration), for developing a satellite factory in space, and EROSS+ (European Robotic Orbital Support Services), for exploring on-orbit services using the DLR CAESAR robotic arm [17].

Hybrid Planners

Traditional task planning is focused on symbolic representations, mainly based on transforming geometric constraints into a semantic representation. This representation, typically implemented using the Planning Domain Definition Language (PDDL, [18]), allows a fast solution of different planning problems. However, not all the problem constraints can be represented symbolically, which led to the creation of hybrid planners. These planners have been widely used in the last years to solve Task And Motion Planning (TAMP) problems.

Hybrid planners combine two domains, a purely symbolic representation, and a geometric representation of the world. One of the first hybrid planners proposed for simple robotic manipulation tasks was the Asymov system [19]. MIT (Massachusetts Institute of Technology) has done great progress in this area in the last years by showing that symbols can also be used to show the effect of actions [20]. These effects can also be learned autonomously to solve more complex tasks [21]. Thus, a hybrid planner could perform preparation tasks based on geometric constraints by understanding how such tasks will impact the robot capabilities to interact with the environment.

For solving TAMP tasks, we have taken the approach of executing the robot tasks in simulation, and learning from the constraints that appear while specifying the problem [22], [23]. These constraints are not only related to the domain itself, but also to the own capabilities or skill sets of the robot. Following this approach, we are able to test how different failures may affect the robot task execution and how they impact the overall plan [24]. In this work we follow the same ideas, keeping the robot within the planning loop, and via simulation we measure the limitations of the system and try to optimize an action plan based on such measures.

Trajectory Optimization

Robot motion planning can be solved using two different approaches: sampling- or optimization-based motion planning. Sampling-based motion planners, such as the Probabilistic Roadmap Method (PRM) [25] and the Rapidly-Exploring Random Trees (RRT) [26], construct a search tree via random sampling. These methods are particularly efficient for solving complex, high dimensional motion planning problems with narrow passages, but generate sub-optimal trajectories and require post-processing steps to generate sensible motions. Optimization-based methods excel in motion planning problems with low obstacle density. These methods start with an initial trajectory and iteratively optimize it based on external constraints, for example joint and torque limits, and collision with obstacles. Examples of these methods include the use of sequential convex optimization (TrajOpt) [27], covariant gradient-descent (CHOMP) [28], and stochastic gradient-descent (STOMP) [6]. While these methods cannot solve any arbitrarily complex motion planning problem and are susceptible to local minima, experiments have shown that they usually converge quickly to directly executable trajectories, with low search complexity in most of the cases. Due to the large portion of free space around the robotic manipulator during in-orbit assembly problems, our method employs trajectory optimization (in the physical layer of the planner) for obtaining robot joint trajectories for the required motions. We chose STOMP for our implementation due to its ability to overcome local minima during the search for a feasible trajectory, and to handle general cost functions without the need for an explicit gradient computation.

3. ARCHITECTURE OF THE HYBRID PLANNER

The problem of planning an assembly with a space robotic system while considering the perturbations on the attitude of the satellite can be efficiently solved using a hybrid planner. To illustrate our method, we will employ as use case the assembly of the primary mirror for a space telescope, using segmented mirror tiles [4]. As an input, the planner requires only the final desired configuration of the mirror. It is assumed that there is no other restriction in the order that the mirror can be built, except for a maximum number of

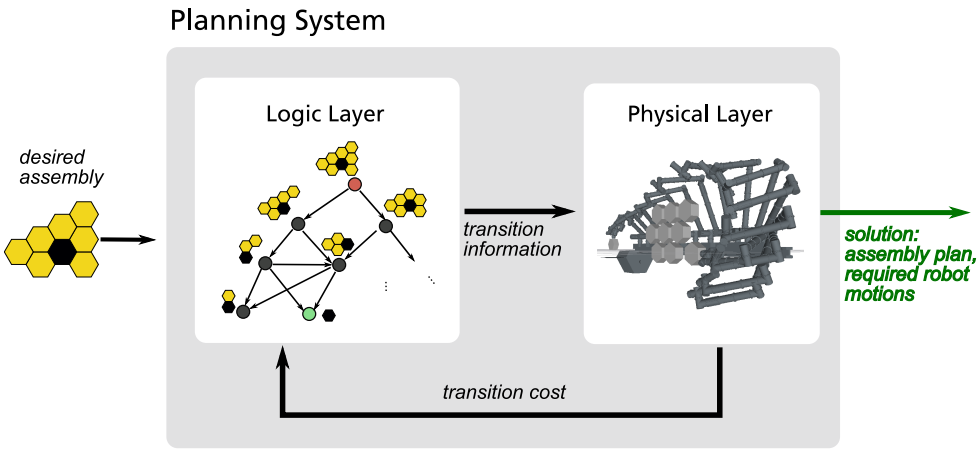

Figure 1: Overall architecture of the hybrid planner. Both layers are connected via the cost of the transitions between states.

segmented mirror tiles that can be moved at the same time. It is also assumed that all the mirror tiles are identical, and that the robotic system has full access to all of them. Finally, only collision with the satellite body, other SMTs, and the robotic system itself are considered. The outcome of the whole planning process is a sequence of steps that allows the robot to incrementally construct the desired structure while minimizing the disturbances created in the base of the system.

A hybrid planning approach is commonly used for scenarios where a combination of discrete and continuous conditions needs to be fulfilled. For this case, the discrete conditions correspond to the semantic constrains that guarantee mechanical, power and data connectivity of the mirror tiles, while the continuous conditions correspond to the motion of the robotic arm for changing its configuration during the assembly execution. A hybrid planner is composed by two layers, a logic one and a physical one. By separating the planning process in these two layers, the solutions can be sequentially checked. First, a verification of the structure of the solution in the logic layer is performed, considering only symbolic constraints. Second, a more time-consuming check is performed in the physical layer to analyze the dynamics of the solution while computing the disturbances suffered in the base. The overall architecture of the proposed planner is shown in Fig. 1. This core approach is similar to the one we used in [4], but with several extensions in the implementation of both layers to cope with the increased complexity of the planning problem, which now considers the effects on the satellite base (not considered in the planner proposed in [4]).

The logic layer or high-level planner deals with the step sequence required to assemble the telescope in the correct way. In this layer the problem is formalized as a graph search, where the nodes represent sub-assemblies, and the transitions between nodes represent the robot actions required to convert the structure from one sub-assembly to another one. In our previous work in [4] only single tiles could be added at a given step. Now, to better consider even larger structures, we include the possibility to assemble up to 3 mirror tiles at the same time in a sub-module, which is later added to the overall structure. The search of a feasible sequence in the graph representation is performed using a Breadth-First search algorithm together with a heuristic to prioritize the best transitions.

The physical layer takes care of the dynamical effects of the assembly process. Indeed, up to this point the sequence plan is merely symbolic; this layer incorporates the robot dynamics. This is performed by adapting the STOMP planning algorithm, which provides the joint motions for the robot arm, with a suitable cost function that considers the attitude disturbances created in the base of the satellite. The trajectories provided by this layer correspond to the transitions between sub-states (nodes) in the graph representation of the logic layer. The values provided by the cost function are tested and verified with the OOS-SIM simulator system to experimentally show the reduction of disturbances on the satellite base when a robot arm follows the optimized trajectories instead of the non-optimized ones.

The logic and physical layer are connected via the cost assigned to the transitions between nodes. The physical layer computes these values and the logic layer uses them to prioritize some sub-assemblies during the search process. In this way, the disturbances are not only minimized by the motion planner in a point to point motion, but also during the overall assembly sequence.

4. Physical (low-level) layer

This section presents the formulation required for obtaining the cost function that computes the attitude disturbances on the satellite base according to the movement of the robot arm. We integrate this cost function in the STOMP algorithm to improve the trajectory and minimize the perturbations on the satellite base.

Dynamics of disturbances in the base

An illustration of an Orbital Robotic Assembly Vehicle (ORAV) is shown in Fig. 2. Its complete configuration is given by $(g_b, q) \in \mathrm{SE}(3) \times \mathbb{R}^n$, where $g_b \equiv (R_b, p_b) \in \mathrm{SE}(3)$ denotes the pose of the spacecraft base frame $\{\mathcal{B}\}$ relative to the inertial frame $\{\mathcal{I}\}$, and $q \in \mathbb{R}^n$ is the vector of n holonomic joint positions. The pose g_b consists of the orientation $R_b \in \mathrm{SO}(3)$ and the position vector $p_b \in \mathbb{R}^3$. The differential kinematics for the pose, g_b, is given by $\dot{g}_b = g_b V_b^\wedge$, where $V_b = [V_T \quad V_\omega] \in \mathbb{R}^6$ is the spacecraft body velocity, and V_T and V_ω are the translational and rotational components, respectively. The operation $V_b^\wedge = \begin{bmatrix} V_{\omega\times} & V_T \\ 0 & 0 \end{bmatrix}$ is used, where $(\bullet)_\times$ is the skew-symmetric form of the argument, and $(V_b^\wedge)^\vee = V_b$.

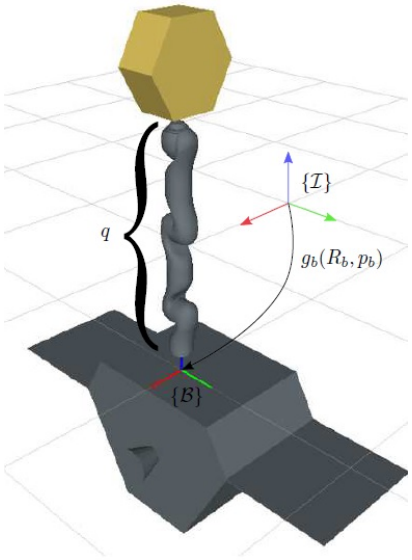

Figure 2: Frames assigned in the ORAV for the dynamic formulation.

Considering the full configuration velocity $V = \begin{bmatrix} V_b^\top & \dot{q}^\top \end{bmatrix}^\top$, the motion equations for the ORAV are described using a floating-base dynamics formulation [29], [30]. The motion equations are

$$M(q) \begin{bmatrix} \dot{V}_b \\ \ddot{q} \end{bmatrix} + C(V) \begin{bmatrix} V_b \\ \dot{q} \end{bmatrix} = \begin{bmatrix} \mathcal{F}_b \\ \tau \end{bmatrix} \tag{1}$$

where $\mathcal{F}_b \in \mathbb{R}^6 \cong \mathfrak{se}(3)^*$ is the spacecraft actuation wrench and $\tau \in \mathbb{R}^n$ is the actuation of the robotic assembly system, respectively. Also, $C \in \mathbb{R}^{(6+n)\times(6+n)}$ is the non-unique Coriolis and Centrifugal (CC) dynamic matrix, and $M = \begin{bmatrix} M_b & M_{bq} \\ M_{bq}^\top & M_q \end{bmatrix}$ is the coupled inertia, where M_b, M_{bq}, M_q are the locked, coupling and manipulator inertias, respectively [7]. Further details about these equations are provided in [31].

In contrast to (1), an alternative way of describing the motion equations of the ORAV is through the total dynamics of its shape (joints) and momentum. These equations reveal a block-diagonalized inertia, which enables writing the shape dynamics as an Euler-Lagrange equation. In particular, the full system velocity is written alternatively as $\xi = \begin{bmatrix} \mu^\top & \dot{q}^\top \end{bmatrix}^\top \in \mathbb{R}^{6+n}$, with locked velocity μ obtained as $\mu = V_b + \mathcal{A}_l(q)\dot{q}$, where $\mathcal{A}_l = M_b^{-1} M_{bq}$ [32]. The locked velocity μ has a physical interpretation of being the velocity of the instantaneous equivalent rigid body system by locking the joints of the orbital robot, and $\mathcal{A}_l$ is known as the *dynamic-coupling* factor [33] in orbital robotics. The commonly-known generalized momentum, $\mathcal{J}$, is related[1] to μ as $\mathcal{J} = \mathrm{Ad}_{g_b}^{-\top} M_b \mu$. Firstly, note that a linear transformation of V leads to ξ, as $\xi = T(q)V$, where $T = \begin{bmatrix} \mathbb{I}_{6,6} & \mathcal{A}_l \\ 0_{n,6} & \mathbb{I}_{n,n} \end{bmatrix}$. Secondly, using T, a transformation of (1) is obtained [34, eq. 15-18], to get the new motion equations as

$$\begin{bmatrix} M_b(q) & 0_{6,n} \\ 0_{n,6} & \Lambda_q(q) \end{bmatrix} \begin{bmatrix} \dot{\mu} \\ \ddot{q} \end{bmatrix} + \begin{bmatrix} \Gamma_b(V) & \Gamma_{bq}(V) \\ -\Gamma_{bq}(V)^\top & \Gamma_{qq}(V) \end{bmatrix} \begin{bmatrix} \mu \\ \dot{q} \end{bmatrix} = \begin{bmatrix} \mathcal{F}_b \\ \tau - \mathcal{A}_l^\top \mathcal{F}_b \end{bmatrix} \tag{2}$$

where $\Gamma_{(\bullet)}$ are the transformed CC terms, and Λ_q is the reduced inertia matrix [33]. The top and bottom of (2) are the momentum and reduced joint dynamics [33], respectively. For the sake of illustration of the key concept, the following assumption is used.

Assumption 1: The ORAV is operating in a zero momentum state, i.e., $\mathcal{J} = \mu = 0_6$ after all residual momentum has been dumped out, and no external actuation is used, i.e., $\mathcal{F}_b = 0_6$.

Under Assumption 1, the second row in (2) can be compactly written as,

$$\Lambda_q(q)\ddot{q} + \Gamma_{qq}(q, \dot{q})\dot{q} = \tau \tag{3}$$

The differential kinematics for the spacecraft pose g_b is $\dot{g}_b = g_b\big(-\mathcal{A}_l(q)\dot{q}\big)^\wedge$, where $-\mathcal{A}_l$ acts as an inertia-dependent Jacobian that maps joint velocities $\dot{q}$ to the spacecraft body velocity, V_b. Considering the components $\mathcal{A}_l = \begin{bmatrix} \mathcal{A}_{lT} \\ \mathcal{A}_{l\omega} \end{bmatrix}$, we obtain,

$$\dot{R}_b = R_b\big(-\mathcal{A}_{l\omega}(q)\dot{q}\big)^\wedge \tag{4}$$

To solve (4), the following assumption is required.

Assumption 2: The trajectory in joint-space is such that $\mathcal{A}_{l\omega}$ remains nearly constant. This condition is fulfilled by a typical ORAV because, in a practical mission scenario, the spacecraft is likely much larger in inertia than the robotic mechanism.

Using Assumption 2, (4) is solved as,

$$R_b(t_f) = \exp\big(-\int_0^1 (\mathcal{A}_{l\omega}\dot{q})^\wedge dt\big) = \exp\big(-\int_{\partial C} (\mathcal{A}_{l\omega} dq)^\wedge\big) \tag{5}$$

where ∂C is the path in joint-space. Note that in (5) the second equality results simply from converting the time-integral to a path integral. Using the inverse of the exp map (see [35]) on (5), we obtain,

$$\omega = \log(R_b)^\vee = -\int_{\partial C} \mathcal{A}_{l\omega} dq \approx \sum_{k=1}^{N} \underbrace{\mathcal{A}_{l\omega}^k \Delta q_k}_{\omega_k} \in \mathbb{R}^3 \tag{6}$$

where N is the number of steps in the trajectory, and $\Delta q(t) = q(t) - q(t-1)$ is a trajectory slice. Note that the last approximation follows after applying Assumption 2 and assuming piecewise constant Δq, which can be guaranteed by the planner. The total platform disturbance for the whole trajectory, ω, is composed of individual trajectory slide disturbances, ω_k in (6). This mathematical machinery is exploited here for trajectory planning to minimize the platform disturbances over the whole trajectory.

STOMP application

To minimize the base disturbance we used STOMP (Stochastic Trajectory Optimization for Motion Planning), as it allows

[1] $\mathcal{J}$ has been employed in other works [34] instead of μ to define the new system velocity coordinates.

the usage of non-linear and non-differentiable cost functions [6]. Thus, it is easy to add new cost functions to the optimizer, including for instance joint limits or other binary checks.

We use for our optimization three cost functions together: collision cf_{coll}, joint limits cf_{joint}, and attitude movement of the base $cf_{\omega base}$, all depending on $\boldsymbol{\theta}_t$, the joint configuration of the robot at time t.

$$cf(\boldsymbol{\theta}_t) = cf_{coll}(\boldsymbol{\theta}_t) + cf_{joint}(\boldsymbol{\theta}_t) + cf_{\omega base}(\boldsymbol{\theta}_t) \quad (7)$$

The first two functions cf_{coll} and cf_{joint} are smoothed Heaviside step functions. When a collision occurs or a joint limit is exceeded, these functions return a high cost, otherwise they return zero. In this way, the solutions will converge to be collision free and within the joint limits. The third cost function, $cf_{\omega base}$, is associated to the attitude change of the base, which we want to minimize. We are using the L_1 norm to measure the change of rotation of the base $\boldsymbol{\omega}_t = A_{l\omega t}\Delta q(t)$ around α, β, γ at the time-step t. Here, $\lambda_{\omega base}$ is the weight associated to this cost.

$$cf_{\omega base}(\boldsymbol{\theta}_t) = \lambda_{\omega base}\|\boldsymbol{\omega}_t\|_1 \quad (8)$$

The algorithm computes the cost of a trajectory by summing up all costs for the N time steps, plus one smoothing term.

$$Q(\boldsymbol{\theta}) = \lambda_s \boldsymbol{\theta}^T \boldsymbol{R}\boldsymbol{\theta} + \sum_{t=1}^{N} cf(\boldsymbol{\theta}_t) \quad (9)$$

Here, $\lambda_s \boldsymbol{\theta}^T \boldsymbol{R}\boldsymbol{\theta}$ denotes the squared acceleration of the trajectory, with a weight of λ_s [6]. The factor penalizes differences between two time steps, ensuring smoothness of the trajectory.

To find a valid trajectory we initialize the STOMP algorithm with a trajectory that connects the start and end pose in the Cartesian space. The algorithm computes for the trajectory an associated cost at each time step. Based on this cost, and following the equations in [6], a family of trajectories is generated. Among this family, the trajectory with the minimal cost is chosen to start a new iteration. During this process the cost monotonically decreases. The algorithm has no particular convergence criteria, the process is stopped after a predefined number of iterations. The final value of cost strongly depends on the start and goal configuration of the trajectory, since these may alter the local minimum found by STOMP. Fig. 3 shows an example of the evolution of the trajectory cost after several iterations of STOMP.

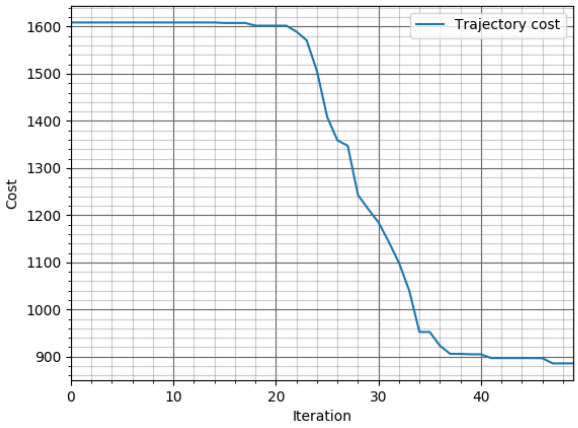

Figure 3: Example of cost optimization in STOMP for a particular trajectory.

5. Logic (High-level) Layer

This section describes the logic layer, which is in charge of generating the sequence of steps required to assemble the segmented mirror. First, the formal representation of the problem as a graph search is introduced. Then, an explanation on how the search is performed over this graph and how the performance is enhanced is given. Finally, an extension of the planning formulation to reuse previously planned configurations for new assemblies is presented.

Formal representation

The assembly planning problem is represented using a graph, where the nodes represent the sub assemblies of the final telescope and the edges of the graph represent the transitions between sub-assemblies, as presented in Fig. 4. Such transitions require motions of the robotic arm that have certain influence on the satellite base orientation, i.e. they create the disturbances we want to minimize. To find feasible assembly sequences, we employ a disassembly-for-assembly strategy, starting with the final assembly, and removing parts consecutively until obtaining the initial mirror tile.

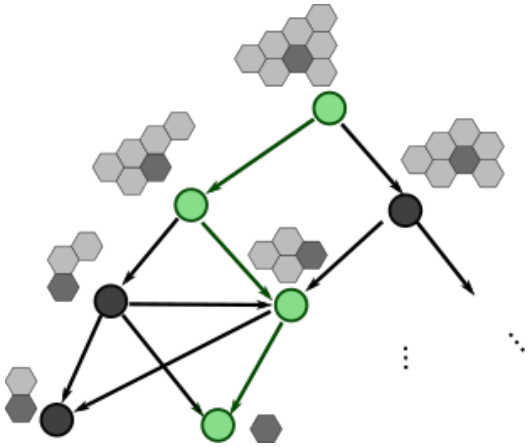

Figure 4: Graph representation of the assembly planning problem. Each node represents a sub-assembly, starting from the final configuration and transforming it until only the initial (central) tile is left. The lowest cost path is the solution for the assembly planning problem in the logic layer.

Formally, we define the graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. Calling SMT to the set of all the mirror tiles in the final telescope, then a node $n \in V$ contains a set of tiles $smt_n \subseteq SMT$. An edge e is defined between the nodes $start$ and end, which involves the movement of a set of mirrors $smt_e = smt_{end} - smt_{start}$. In the real system the set smt_e represents the tiles that are pre-assembled before being set in their final position. For simplicity of the analysis in this work, we assume $\forall e \in V$ that $|smt_e| \leq 3$, which means that only up to 3 mirror tiles can be assembled in one transition (see Fig. 5). Each of the transitions has an assigned cost indicating the disturbance

generated in the base by the movement required for the robot to perform this assembly step. These costs are minimized in the physical layer via the STOMP algorithm.

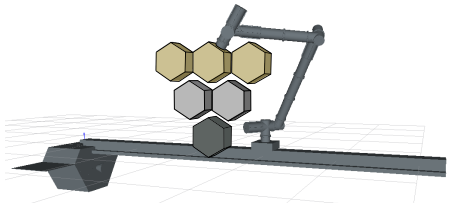

Figure 5: Example of the robotic system manipulating three SMTs at the same time (golden ones). In our definition of the problem, we assume that the robot is only able to move up to 3 tiles to simplify the computation of the motions.

The logic layer is also in charge of verifying certain constraints that the transitions must follow. First, it verifies that all the mirrors in smt_e are connected between them. Such condition ensures that the full set of mirrors can be moved in one transition. The other condition checked in this layer is the feasibility to connect the mirrors in smt_e to the already placed mirrors. This is illustrated in Fig. 6, where valid states are identified when none of the mirror tiles in smt_e has 3 or more neighbors already placed, and the tiles in the remaining assembly are all connected. These constraints are problem-specific, i.e. they apply for our particular case of hexagonal mirror tiles, but may vary for other shapes. Nevertheless, the representation in the logic layer using nodes for subassemblies and edges for transitions can be generalized to other shapes or assembly problems.

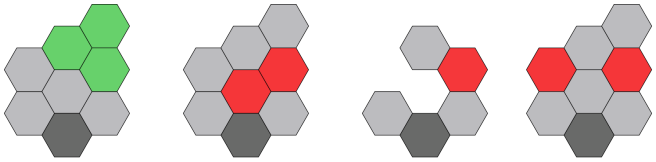

Figure 6: Examples of valid and invalid transitions. a) is a valid transition, as all moving tiles (green) are connected and none has 3 or more neighbors. b) is invalid since one tile in the set smt_e (in red) has 3 or more neighbors. c) is invalid since the already placed tiles do not form a unique island of mirrors. d) is invalid because the tiles moved in one transition are not connected between them.

The problem of minimizing the disturbances in the base during the whole process is translated into a graph search. The optimal path is the one minimizing the sum of the cost of the edges, which is a well known problem in graph search. Formally, we look for the sequence of edges $\{\hat{e}^1, \hat{e}^2, \hat{e}^3, ..., \hat{e}^t\}$ that minimizes $\sum_{i=1}^{t} cost(\hat{e}^i)$ subject to $\hat{e}^t.end = SMT$, $\hat{e}^1.start = smt_0$ and $\hat{e}^i.end = \hat{e}^{i+1}.start \ \forall i \in= \{1, 2, 3, ..., t-1\}$. This search is performed using an A* algorithm, as explained in more detail below.

A search*

For the graph representation used in this problem, the most common approach to find the cheapest path employs the Dijkstra algorithm. This algorithm is useful when the cost of all the edges is known, but in our case all these costs must be first computed by the physical layer. Such computation is possible but very time consuming, which is not desirable for very large configurations due to the large number of possible transitions. Therefore, the nodes and edges to be computed have to be chosen in an intelligent manner, as provided by the A* search.

The heuristic chosen for the A* search is the accumulated disturbance in the base until reaching the given configuration, divided by the amount of placed mirrors. Despite not being a heuristic that guarantees finding the optimal solution, it provides good empirical guidance for finding an adequate sequence fast. Another advantage of introducing such heuristic is limiting the branching factor. Since the amount of possible transitions in a given state is too high (in the order of hundreds for a 20-tile mirror), it is good practice to prune already some of the states that are not promising. This translates in reducing the amount of nodes that are selected to be analyzed in the future, which speeds up the planning process. The pruning process is represented in Fig. 7.

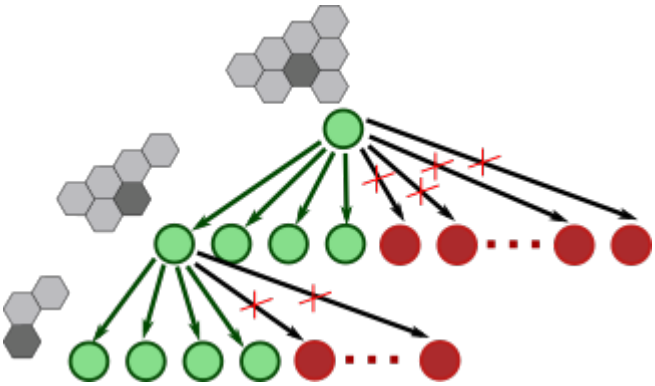

Figure 7: Graph exploration during planning time, using a branching factor of four as example. Only the best four transitions according to the heuristic are kept to be analyzed in the future, if required.

Re-use of previous planning

Given the modular nature of the segmented mirror tiles, it is normal that previously planned structures appear as sub-problems for new configurations. As the system evolves over time, more information is gathered for each new configuration planned. When a solution is found, it was already optimized in both layers in a time-consuming process. Also, all the sub-steps done in the solution are also optimal according to the used heuristic. This can be reused recursively, in case that the sub configuration is repeated again in future steps. Because of this, not only the initial configuration solution is stored, but also all the intermediate steps. We can extend the algorithm as shown in Algorithm 1 to profit from previously explored sub-steps in the assembly process.

6. EXPERIMENTS

In order to verify the correct behavior of our planner, we exposed it to different scenarios and ran experiments to verify each layer. For the physical layer we show two main points. First, we verify the dynamic equations in a real robotic system by executing trajectories and comparing our simulation results with the measurements of the system, in order to show that the whole model preserves zero momentum in the system. Second, we verify the decrease of disturbances in the base when the optimized trajectory is used, compared to a non-optimized one.

Algorithm 1: Search for solutions in the logic layer.

$PriorityQ_{TELESCOPE} \leftarrow \emptyset$;
SEARCH LOGIC LAYER $(SMT, known_{SOL}, b_{MAX})$
  $t_{actual} \leftarrow SMT$;
  while $t_{actual} \notin known_{SOL}$ **do**
    $PriorityQ_{NEW} \leftarrow$ **GENTRANSITIONS**(t_{actual});
    for b_{MAX} **do**
      $t_{new} \leftarrow PriorityQ_{NEW}$.pop(0);
      t_{new}.parent $\leftarrow t_{actual}$;
      t_{new}.accCost $\leftarrow t_{new}$.accCost + t_{actual}.cost;
      $PriorityQ_{SAT}$.add(t_{new});
    end
    $t_{actual} \leftarrow PriorityQ_{TELESCOPE}$.pop(0);
  end
  while $t_{actual}! = SMT$ **do**
    $know_{SOL}.add(t_{actual})$;
    $t_{actual} \leftarrow t_{actual}$.parent
  end
  $know_{SOL}.add(t_{actual})$

For the logic layer we show three different points. First, we show how reusing previous plans can enhance the planning times by requesting less calls to the physical layer. For this, we run our planner for different telescope designs, with an increasing number of mirror tiles (from 4 to 10). In all the cases the central tile of the mirror is already placed, and the robot places the remaining tiles. Second, we compare the total disturbance perceived in the base for different cases in two scenarios, one with our complete planning system and another one without optimizing the assembly order nor the trajectories. Again, the experiments are held for configurations between 4 and 10 segments. Third, we show the capabilities of our system for planning the assembly of a telescope with a large number of mirrors (20). For all these experiments, collisions between the mirror segments, the robot arm and the satellite are taken into account.

Physical Layer

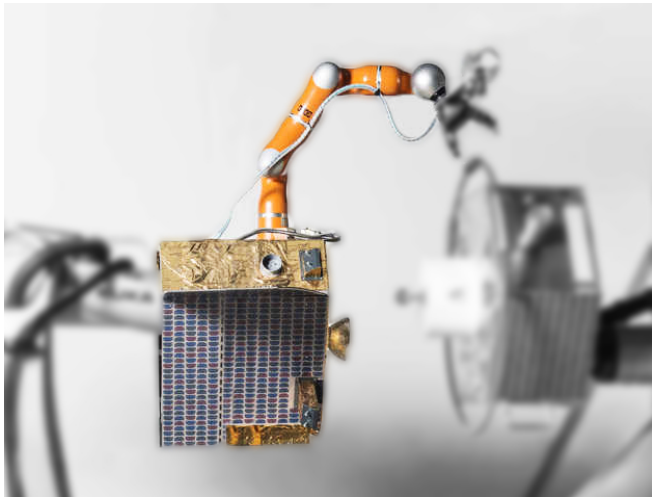

Figure 8: OOS-SIM setup, including a servicer satellite carrying a robotic arm.

The verification of the physical model in the simulation and the validation of the proposed optimization approach was performed using the On-Orbit Servicing Simulator (OOS-SIM) [7]. In the OOS-SIM (Fig. 8), a KUKA KR120 robot generates the motion of the floating vehicle (satellite) using a model-based multibody hardware-in-the-loop approach with consistency of momentum [32]. A 7-DoF KUKA LWR-4+ robot is mounted on the satellite mock-up to provide the actual robotic manipulator. The manipulator is equipped with a gripper, which has a mass of $4.02Kg$ and inertia tensor of $I = diag\,(0.05, 0.05, 0.04)$, with $diag$ indicating a diagonal matrix. All the experiments were performed with this load.

The first objective of the physical experiments was to demonstrate the compliance between the satellite motion in a relevant environment (OOS-SIM) and the simulations employed in this paper. Fig. 9 compares the disturbance of the satellite base over a complete trajectory. Note that there is a minor difference between the physical robot behavior and the simulation, which is mainly attributable to unmodeled effects that include joint encoder noise and admittance dynamics of the KR120 robot.

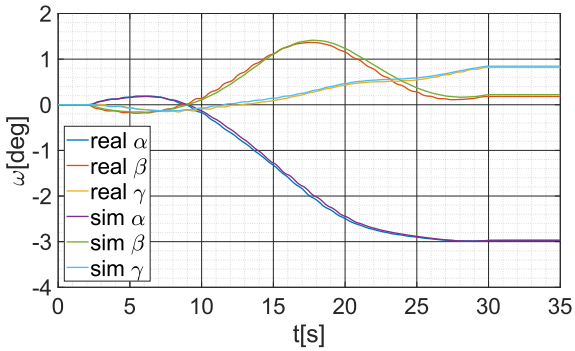

Figure 9: Attitude of the OOS-SIM satellite base over time for simulated ("sim") data vs OOS-SIM measured data ("real").

We also tested the performance of our optimization of the base disturbance. For our experiments we first selected a goal pose of the end-effector in Cartesian space. Then, we use the STOMP algorithm to find a path using 30 samples along the trajectory. These 30 sample points are then interpolated to get a smooth trajectory, which is sent to the robot controller with a sampling rate of $1ms$. Fig. 10 and Fig. 11 show the non-optimized and optimized trajectory, respectively. Note that for the non-optimized trajectory the start and goal configurations are connected quite directly with a cubic interpolation. However, the optimized trajectory performs a more complex motion in joint space in order to minimize base disturbances.

The generated trajectories can be reliably followed by the robotic system. Thus, the next step is to measure the changes in the attitude in the base, taking a closer look to the attitude around the α, β, γ axis for the non-optimized (Fig. 12) and optimized (Fig. 13) joint paths. The final displacement around the α axis is 6 and 3 degrees for the non-optimized and optimized trajectories respectively, already showing a significant improvement. For the β axis in both cases the displacement ends up in zero, but it is important to point out that we want to minimize the disturbance over the complete trajectory, not only in the final configuration. The peak displacement in the β axis is 2.5 and 1.5 degrees for the non-optimized and optimized trajectory respectively, again showing an improvement.

We tested the approach with 11 different end effector poses. In all the trajectories we were able to optimize the cost. How big this improvement was correlates with the change of the

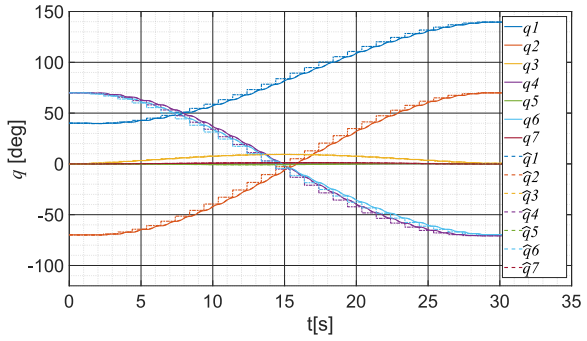

Figure 10: Joint values of the LWR manipulator over time, showing the incoming values $\hat{q}n$ from STOMP (using 30 timesteps) and the interpolated values $\hat{q}n$, with n the joint number, for the *non-optimized* trajectory.

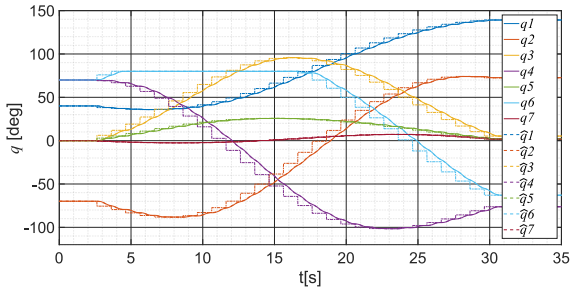

Figure 11: Joint values of the LWR manipulator over time, showing the incoming values $\hat{q}n$ from STOMP (using 30 timesteps) and the interpolated values $\hat{q}n$, with n the joint number, for the *optimized* trajectory.

base attitude, in the best case an improvement of around 45%, and an average of 18% for single trajectories, are obtained. Note that the optimizer tries to move from the start to the goal pose staying as close as possible to the center of mass of the whole system[2].

Finally, to prove the physical consistency of the experiments, the locked velocity μ, which is proportional to the total momentum generated on the OOS-SIM facility, is shown in Fig. 14 for the optimized case. Note that $\mu \approx 0_6$ with peaks within the bounds $[-0.004, 0.004]$, which agrees well with the experimental requirement of the zero-momentum case. The variations in Fig. 14 arise due to the admittance dynamics of the KR120. This is because it is required to track the model based on the interpolated joint trajectory of only 30 sample points, which leads to fast accelerations in the robot joints.

Logic Layer

The first aspect to verify with the logic layer enhancements is how they speed up the search for a solution. As the calls to the physical layer are the most time-consuming part of our planner, minimizing them has direct impact on the total time the system takes to solve a configuration. For the analysis, several configurations were given to the planner, in an increasing degree of difficulty (larger number of mirrors). On one hand, the experiment is done without storing any information, and on the other hand the information about

[2]A video of this experiment can be found in `https://youtu.be/r-Htx_MBGRc`

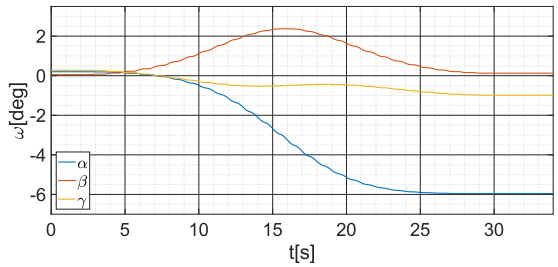

Figure 12: Attitude of the base around α, β, γ for the *non-optimized* trajectory, with data measured directly from the OOS-SIM system.

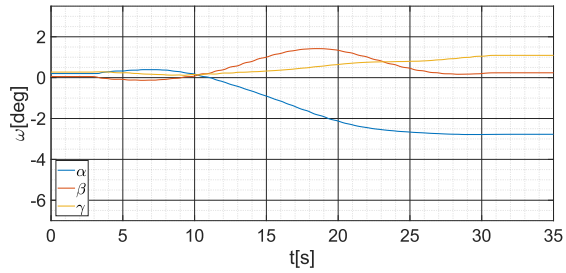

Figure 13: Attitude of the base around α, β, γ for the *optimized* trajectory, with data measured directly from the OOS-SIM system.

previous solutions is used. Table 1 compares the number of calls to physical layer when the system re-uses information and when it does not. The results show a clear advantage when information is re-used, particularly for complex cases, where the amount of states to be visited grows exponentially.

The second analysis performed is the comparison of the proposed planner using both layers, against non-optimized trajectories nor sequences. In this case, for the collision and dynamic model we used a satellite endowed with a robotic arm (CAESAR, [17]) mounted on a linear rail, to enhance its reachability (Fig. 15). The model includes the collision models and masses and inertia tensors for all parts. In this experiment we compare our method against a naive approach were we simply compute collision-free trajectories, and the mirror tiles are added one by one starting from the center and going outwards. The metric for comparison is the sum of the displacement α, β, γ added over the 30 time steps of the trajectories, and also added over all the transitions required for the assembly. The results are shown in Table 2. Even for the small cases, an improvement of more than 50% is detected. This happens for all the telescope sizes, reaching improvements of almost 80% in two cases. This shows the importance of an assembly planner that can consider the dynamical requirements into the sequence plan. Such results can have a big impact on the real execution, as this reduction of changes in the base attitude is translated into less energy required for the thrusters or reaction wheels to keep a fixed position and orientation of the satellite.

As a final experiment, we show the capability of our system to plan assembly sequences for big telescopes. For this, a 20 mirror tile telescope is created and given as an input. Our system was fed with the previous examples, and has stored the experience of those planned solutions. The solution sequence to our problem is shown in Fig. 16. Note that the system

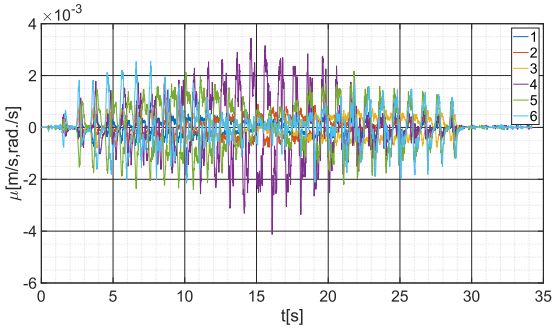

Figure 14: Measured locked velocity (μ), which is proportional to the total momentum generated on the OOS-SIM facility during the experiment with optimized trajectory.

Table 1: EVALUATION OF NUMBER OF CALLS TO PHYSICAL LAYER BY REUSING INFORMATION

# of mirrors	calls to physical layer *without* reusing information	calls to physical layer *with* reuse of information	% reduction
4	6	6	0.0%
5	12	12	0.0%
6	20	11	45.0%
7	38	34	10.5%
8	125	42	66.4%
9	58	36	89.9%
10	149	15	91.6 %

prefers to do transitions with larger sub-assemblies, and tries to keep the sub-assembly symmetry around the axis defined by the rail.

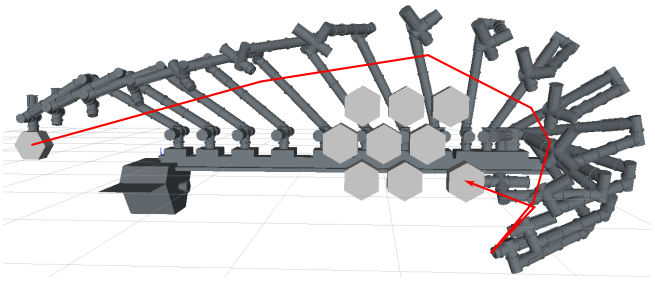

Figure 15: Satellite setup for experiment done with a telescope of 12 mirror tiles. The satellite consist of a rail where the DLR CAESAR robot arm can translate and perform the assembly operations. The image displays optimized motions to grasp and place the last mirror tile at the desired position. The optimized trajectory reduced 34% of the cost with respect to the original trajectory. The red line shows the movement of the mirror tile for its placement. Note that the optimizer tries to keep the tile as close as possible to the center of mass of the structure.

Table 2: EVALUATION OF THE TOTAL DISTURBANCE EXPERIENCED BY THE SATELLITE

# of mirrors	Disturbances non-optimized plan (rad)	Disturbances optimized plan (rad)	% reduction
4	1.08	0.50	53.3%
5	1.28	0.42	67.7%
6	1.62	0.57	65.1%
7	1.92	0.62	67.9%
8	2.10	0.44	79.0%
9	2.20	0.76	65.7%
10	2.37	0.52	78.1 %

7. CONCLUSIONS AND FUTURE WORK

We have developed a planning system that generates the assembly sequence and manipulator motions required to construct the primary mirror for a telescope using segmented mirror tiles. The approach uses a hybrid planner, with a physic and a logic layer, which is able to diminish the disturbances (changes in orientation) in the base significantly: By as much as 65 % in some cases. The planner is also capable of generating plans for large structures of up to 20 mirror tiles, and given more time to keep learning new partial solutions, it can scale up to many more.

We have verified that the dynamic models on which our simulations in the physical layer are based are correct, by comparing the predictions of our simulations with measurements taken in a real robotic system. In this scenario, we were also able to verify how STOMP helps to significantly diminish the disturbances in the base. Regarding the logic layer, we have shown the importance of reusing information of previously planned sequences. The number of calls to the physical layer planner considerably decreased when the number of mirror tiles increased. Lowering these calls has a direct impact on the total planning time, since this computation is by far the most time-consuming step.

We have demonstrated how the hybrid planner can solve an assembly planning problem using as cost function the minimization of perturbations on the satellite base. However, any other cost function could be designed to adapt the planner to a different domain. By separating the problem in two layers we were able to perform a complex high level plan while taking into consideration the dynamic constraints of the system. This would not have been possible if we had seen the whole process in only one dimension.

Another potential use of this planner is in risk assessment for assembly operations. Thanks to the two-layer approach, it is simple to keep a same plan in the logic layer but giving more limitations in the physical layer. By doing so we could test for instance the same sequence of tasks but reducing or enhancing the capabilities of the robot (to analyze the influence of other factors such as joint failures, limited skill sets or modified lengths of the robot links), which would provide a better idea of the robustness to failures of the overall assembly plan.

Finally, there are some aspects that could be improved. For instance, STOMP cannot ensure an optimal solution in a finite amount of time, so we can only approach the overall optimum with certain probability. Also, better feedback might be used between both layers to extend the knowledge gathered in one execution, so that the acquired knowledge can be applied not only for the assemblies that repeat exactly the same pattern, but also for other assemblies that include similar (not identical) sub-assembly patterns.

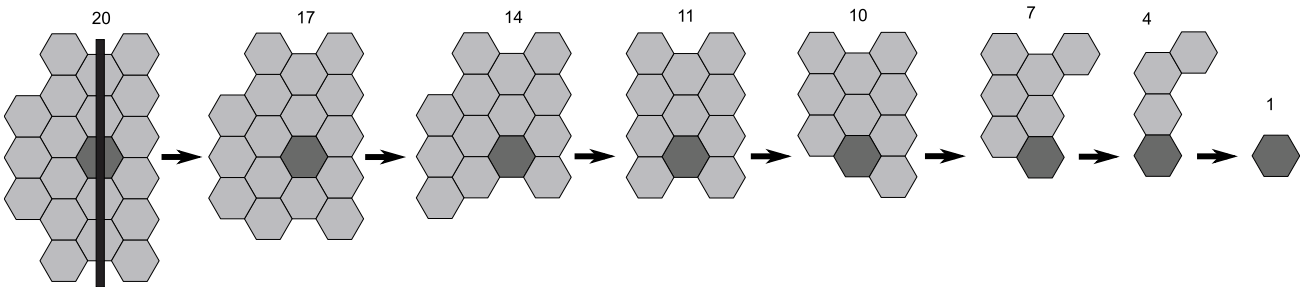

Figure 16: Solution given by our planner for assembling a large telescope with 20 mirror tiles. The sequence of steps is provided for the disassembly, the assembly process would follow the inverse order. The black rectangle in the first configuration shows the position of the linear rail (Fig. 15).

REFERENCES

[1] M. Rognant, C. Cumer, J. Biannic, M. A. Roa, A. Verhaeghe, and V. Bissonnette, "Autonomous assembly of large structures in space: a technology review," in *Proc. Europ. Conf. for Aeronautics and Aerospace Sciences (EUCASS)*, 2019.

[2] E. Papadopoulos, I. Tortopidis, and K. Nanos, "Smooth planning for free-floating space robots using polynomials," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2005, pp. 4272–4277.

[3] K. Nanos and E. Papadopoulos, "On cartesian motions with singularities avoidance for free-floating space robots," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2012, pp. 5398–5403.

[4] J. Martínez-Moritz, I. Rodríguez, K. Nottensteiner, J.-P. Lutze, P. Lehner, and M. A. Roa, "Hybrid planning system for in-space robotic assembly of telescopes using segmented mirror tiles," in *Proc. IEEE Aerospace Conf.*, 2021, pp. 1–16.

[5] M. A. Roa, K. Nottensteiner, G. Grunwald, P. Lopez, A. Cuffolo, S. Andiappane, M. Rognant, A. Verhaeghe, and V. Bisonnette, "In-space robotic assembly of large telescopes," in *Proc. Symp. on Advanced Space Technologies in Robotics and Automation (ASTRA)*, 2019.

[6] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2011, pp. 4569–4574.

[7] J. Artigas, M. De Stefano, W. Rackl, R. Lampariello, B. Brunner, W. Bertleff, R. Burger, O. Porges, A. Giordano, C. Borst, *et al.*, "The OOS-SIM: An on-ground simulation facility for on-orbit servicing robotic operations," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2015, pp. 2854–2860.

[8] K. Yoshida, K. Hashizume, and S. Abiko, "Zero reaction maneuver: flight validation with ETS-VII space robot and extension to kinematically redundant arm," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2001, pp. 441–446.

[9] A. Ogilvie, J. Allport, M. Hannah, and J. Lymer, "Autonomous satellite servicing using the orbital express demonstration manipulator system," in *Proc. Int. Symp. on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*, 2008.

[10] E. Coleshill, L. Oshinowo, R. Rembala, B. Bina, D. Rey, and S. Sindelar, "Dextre: Improving maintenance operations on the International Space Station," *Acta Astronautica*, vol. 64, pp. 869–874, 2009.

[11] L. Metcalfe and T. Hillebrandt, "Robotic refuelling mission - demonstrating satellite refuelling technology on board the ISS," in *Proc. Int. Symp. on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*, 2014.

[12] D. Akin, M. Minsky, E. Thiel, and C. Kurtzman, "Space applications of automation, robotics and machine intelligence systems (ARAMIS) - phase II," in *NASA contractor report 3734*, 1983.

[13] K. Yoshida, "Space Robot Dynamics and Control: To Orbit, From Orbit, and Future," in *Robotics Research*, J. Hollerbach and D. Koditschek, Eds. Springer, 2000, pp. 449–456.

[14] R. Hoyt, J. Cushing, J. Slostad, G. Jimmerson, T. Moser, G. Kirkos, M. Jaster, and N. Voronka, "SpiderFab: An architecture for selffabricating space systems," in *Proc. AIAA SPACE Conf.*, 2013.

[15] V. Bissonnette, C. Bazerque, S. Trinh, C. Porte, G.Arcin, M. Rognant, J.-M. Biannic, C. Cumer, T. Loquen, and X. Pucel, "A simulation tool for in-orbit assembly of large structures," in *Proc. Int. Symp. on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*, 2020.

[16] P. Letier, X. Yan, M. Deremetz, A. Bianco, G. Grunwald, M. Roa, R. Krenn, M. Munoz, P. Dissaux, J. Garcia, R. Ruiz, L. Filippis, G. Porceluzzi, M. Post, M. Walshe, and P. Perryman, "MOSAR: MOdular Spacecraft Assembly and Reconfiguration demonstrator," in *Proc. Symp. Advanced Space Technologies Robotics and Automation (ASTRA)*, 2019.

[17] A. Beyer, G. Grunwald, M. Heumos, M. Schedl, R. Bayer, W. Bertleff, B. Brunner, R. Burger, J. Butterfaß, R. Gruber, *et al.*, "CAESAR: Space robotics technology for assembly, maintenance, and repair," in *Proc. Int. Astronautical Congress (IAC)*, 2018.

[18] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins, "PDDL-the planning domain definition language," *AIPS98 Planning Competition Committee*, 1998.

[19] F. Gravot, S. Cambon, and R. Alami, "aSyMov: a planner that deals with intricate symbolic and geometric problems," in *Int. Symp. on Robotics Research*. Springer, 2005, pp. 100–110.

[20] T. Lozano-Pérez and L. P. Kaelbling, "A constraint-based method for solving sequential manipulation planning problems," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2014, pp. 3684–3691.

[21] L. P. Kaelbling and T. Lozano-Pérez, "Learning composable models of parameterized skills," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2017, pp. 886–893.

[22] I. Rodríguez, K. Nottensteiner, D. Leidner, M. Durner, F. Stulp, and A. Albu-Schäffer, "Pattern recognition for knowledge transfer in robotic assembly sequence planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3666–3673, 2020.

[23] I. Rodríguez, K. Nottensteiner, D. Leidner, M. Kaßecker, F. Stulp, and A. Albu-Schäffer, "Iteratively refined feasibility checks in robotic assembly sequence planning," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1416–1423, 2019.

[24] I. Rodríguez, A. Bauer, K. Nottensteiner, D. Leidner, G. Grunwald, and M. A. Roa, "Autonomous robot planning system for in-space assembly of reconfigurable structures," in *Proc. IEEE Aerospace Conf.*, 2021, pp. 1–17.

[25] L. Kavraki, P. Svestka, J. C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[26] S. M. LaValle, "Rapidly exploring dense trees," *Planning Algorithms*, pp. 228–237, 2004.

[27] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization," in *Robotics: Science and systems*, 2013, pp. 1–10.

[28] M. Zucker, N. Ratliff, A. Dragan, M. Pivtoraiko, M. Klingensmith, C. Dellin, A. Bagnell, and S. Srinivasa, "CHOMP: Covariant hamiltonian optimization for motion planning," *Int. J. Robotics Research*, vol. 32, no. 9–10, pp. 1164–1193, 2013.

[29] P. From, I. Schjølberg, J. Gravdahl, K. Pettersen, and T. Fossen, "On the boundedness and skew-symmetric properties of the inertia and coriolis matrices for vehicle-manipulator systems," *Proc. IFAC Symp. on Intelligent Autonomous Vehicles*, vol. 43, no. 16, pp. 193–198, 2010.

[30] G. Garofalo, C. Ott, and A. Albu-Schäffer, "On the closed form computation of the dynamic matrices and their differentiations," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013, pp. 2364–2359.

[31] H. Mishra, M. De Stefano, A. M. Giordano, R. Lampariello, and C. Ott, "A geometric controller for fully-actuated robotic capture of a tumbling target," in *Proc. American Control Conf. (ACC)*, 2020, pp. 2150–2157.

[32] H. Mishra, A. M. Giordano, M. De Stefano, R. Lampariello, and C. Ott, "Inertia-decoupled equations for hardware-in-the-loop simulation of an orbital robot with external forces," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2020, pp. 1879–1886.

[33] K. Yoshida and D. N. Nenchev, "A general formulation of under-actuated manipulator systems," in *Robotics Research*, Y. Shirai and S. Hirose, Eds. Springer, 1998, pp. 33–44.

[34] G. Garofalo, B. Henze, J. Englsberger, and C. Ott, "On the inertially decoupled structure of the floating base robot dynamics," *IFAC-PapersOnLine*, vol. 48, no. 1, pp. 322–327, 2015.

[35] H. Mishra, M. De Stefano, A. M. Giordano, and C. Ott, "A nonlinear observer for free-floating target motion using only pose measurements," in *Proc. American Control Conf. (ACC)*, 2019, pp. 1114–1121.

BIOGRAPHY

Ismael Rodríguez *received his degree of Ingeniero en Electrónica from Universidad ORT Uruguay in 2015. Since 2016 he is pursuing a Ph.D. in Robotics at the German Aerospace Center (DLR) in Weßling. His research is focused on the development of an assembly planner that is aligned with the requirements of the mass customization phenomenon.*

Jean-Pascal Lutze *received his B.Eng degree in Robotic and Automation from Heilbronn University of Applied Sciences. Since 2017, he is part of the Institute of Robotics and Mechatronics at the German Aerospace Center (DLR). Since 2019 he is pursuing his Master degree in Robotics, Cognition, Intelligence at the Technical University of Munich (TUM). His main focus is space robotics.*

Hrishik Mishra *is a research associate at the German Aerospace Center (DLR), focused on orbital robotic systems. He received his M.Sc. in Satellite Application Engineering from the Technical University of Munich (TUM), Germany, in 2017. In 2010 he received his B.Tech. in Electrical and Electronics Engineering from the Biju Patniak Unversity of Technology (BPUT), India. At DLR, his research is directed towards whole-body control, shared control, and hardware-in-the-loop simulation of orbital robots.*

Peter Lehner *is a researcher at the Institute of Robotics and Mechatronics, German Aerospace Center (DLR), since 2014. He received his master's degree in Computer Engineering from the Technical University of Berlin in 2014. His current research activities include developing methods for autonomous motion generation for mobile manipulation systems to empower mobile robots to autonomously interact with their environment.*

Máximo A. Roa-Garzón *received his doctoral degree in 2009 from Universitat Politecnica de Catalunya, and the Project Management Professional (PMP) Certification in 2016. He worked for Hewlett Packard R&D before joining the Institute of Robotics and Mechatronics of the German Aerospace Center (DLR) in 2010 as Senior Research Scientist, leading the group on Robotic Planning and Manipulation.*