Optimization of On-Orbit Robotic Assembly of Small Satellites

Optimization of On-Orbit Robot Assembly of Small Satellites

Ezinne E. Uzo-Okoro[1] , Prakash Manandhar[2], and Daniel Erkel[1]
Massachusetts Institute of Technology, Cambridge, MA 02139, USA

Mary Dahl[1]
Massachusetts Institute of Technology, Cambridge, MA 02139, USA

Emily Kiley[3]
Massachusetts Institute of Technology, Cambridge, MA 02139, USA

Olivier DeWeck[4]
Massachusetts Institute of Technology, Cambridge, MA 02139, USA

Kerri Cahoy[4]
Massachusetts Institute of Technology, Cambridge, MA 02139, USA

On-orbit assembly missions typically involve humans-in-the-loop and use large custom-built robotic arms designed to service existing modules. A proposed concept of on-orbit robotic assembly of modularized CubeSat components within a spacecraft locker eliminates the need for humans-in-the loop. The spacecraft locker supports use cases such as rapidly placing failed nodes within a constellation of satellites and providing sensing and propulsion capabilities in Low Earth Orbit. Despite the recent proliferation of small satellites, there are few planned demonstrations of on-orbit assembly and few demonstrations of on-orbit servicing. Key gaps challenges of in-space assembly of small satellites are (1) the lack of standardization of electromechanical CubeSat components for compatibility with commercial robotic assembly hardware, and (2) testing and modifying commercial robotic assembly hardware. In this work, we focus on testing and modifying: we develop an optimization process for a robotic assembly model to integrate small satellites in space. Our process focus is on the optimization of the on-orbit assembly time of small satellites. We use Commercial-Off-The-Shelf (COTS) robot arms to snap together components in a spacecraft, while minimizing humans-in-the-loop. Assembly time is the selected performance metric as it is critical to the assertion that building small satellites on-orbit results in reduced budget and satellite development time on Earth. We minimize on-orbit small satellite assembly time by optimizing assembly time with the Genetic Algorithm, which use dexterous robotic arms to assemble components, without any negative effects on the attitude and control system.

[1] Graduate Student, Department of Aeronautics and Astronautics
[2] System Design & Management, MIT Sloan and MIT School of Engineering
[3] Undergraduate Student, Department of Mechanical Engineering
[4] Professor, Department of Aeronautics and Astronautics, and AIAA Associate Fellow

We implement a robot arm assembly model in Python, using Inverse Kinematics. We use a Genetic Algorithm-based optimization scheme, with time as the objective function, and three constraints: robot assembly volume, power consumption, and peak power. Design variables such as joint damping, motor force (torque), position gain and velocity gain are used to model grasping a component and moving the component to the satellite assembly area of the spacecraft. The robot arms are required to be within a tolerance defined based on the 300 mm x 300 mm x 500 mm assembly area. In simulation, we observe that using a given baseline servo motor (7 V) at high proportional gains results in optimal assembly time of approximately 10-20 seconds per component assembly, compared to roughly double this time per component for a 1U CubeSat weighing 2 kg. However, we expect this improvement to result in 25% higher power consumption. Using a high gain value with a lower voltage (5 V) motor results in oscillations and additional time required to dampen out to within the given tolerance, and results in increased assembly time. The benchmarked small satellite assembly time with a human-in-the-loop requires 50 weeks to 90 months of component assembly and integration time on Earth. We anticipate that on-orbit assembly capability optimized for a 1 U functional CubeSat with 30 W of total power, would reduce the assembly time by an order of magnitude. With robotic arm models, for a 1 U CubeSat assembly, we show up to 42% saving benefit in robotic assembly time.

I. Nomenclature

g_1 = robot assembly volume
g_2 = power consumption
g_3 = peak power
J = assembly time
H = Hessian value
S = optimization vector
x = design vector

II. Introduction

The need for low-cost, low-latency and agile space infrastructure, which can reach quickly strategic orbits such as GEO and Low Earth Orbit (LEO) in addition to polar and International Space Station (ISS) orbits, could be met by using robotic assembly of modularized components into CubeSats. A standardized modular CubeSat and COTS-based robotic assembly could eliminate the reliance on long-lead high-cost legacy space hardware. Satellite cellularization [1] has made incremental advances in the modularization of small satellite subsystems. In this work, we explore a new approach to CubeSat production based on the robotic assembly of spacecraft components. We envision a new mission in which small COTS robot arms are enclosed in a free-flying small spacecraft "locker" of approximately 318 mm x 609 mm x 914 mm for the assembly of a new kind of small satellite. These mini-fridge-sized spacecraft "lockers" with propulsion capability are intended to be orbit-agnostic in order to deploy on-demand robot-assembled CubeSats where needed. The locker houses robotic arms, modular components including sensor and propulsion modules, and payloads for 1 U to 3 U-sized CubeSats. The mission is expected to deliver an improvement in first time to orbit for a satellite from >30 days to less than 10 hours for a small satellite build and deployment cycle.

On-orbit assembly missions typically involve humans-in-the-loop and use large custom-built robotic arms, which are designed to service existing modules. While robot arms are becoming ubiquitous in several industries on Earth, commercially available robotic arms suitable for small satellite space applications are not yet available. Despite the recent proliferation of small satellites, there are few planned demonstrations to assemble and service small satellites. The on-orbit robotic assembly of modularized CubeSat components within a spacecraft box offers several key benefits, including, an unparalleled number of possible CubeSat configurations for scientific and commercial use:

1. Flexibility, by allowing for selecting sensors, power and propulsion-sizing;
2. Resiliency, by deploying custom-configured CubeSats for redundancy and/or coordination;
3. Efficiency, by using dexterous robot arms for on-orbit assembly of various sensor payloads.

In-space assembly for small satellites is valuable for large constellations as it enables improved launch mass packaging, on-orbit modular customization and timely deployment to replace failed nodes or add new capabilities to existing missions. We envision a new mission in which small COTS robot arms are enclosed in spacecraft lockers. Robotic assembly based out of propulsive spacecraft "lockers" provides an on-orbit scalable deployer of small satellites faster than NASA's documented minimum launch-on-demand response time (35 days) for the International Space Station (ISS) crew rescue [28]. Figure 1 shows a comparison of how small satellites are currently built and how it will be done in the future.

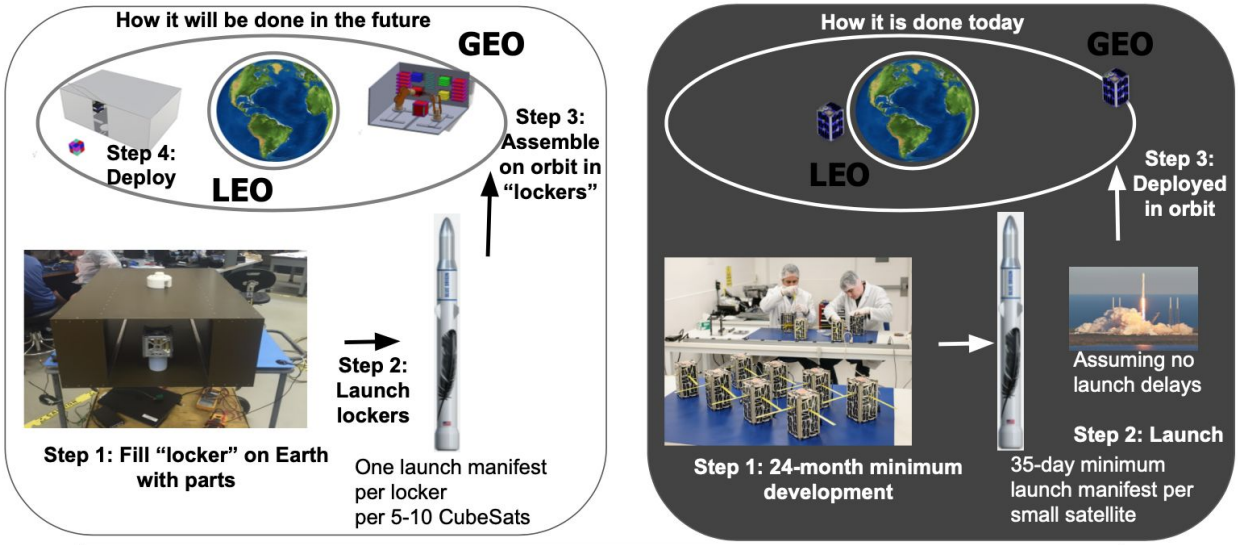

Fig 1. Concept of operations for a "locker" with robotic arm assembly, showing future rapid response time of ~hours, compared with current and traditional missions response times of several weeks

As shown in Figure 2, there are four phases necessary to successfully realize the mission concept. Phase 1 involves the ground-based robotic assembly of a CubeSat prototype using two dexterous arms and electromechanical components in a laboratory environment. Phase 1 also involves modeling and assessment of different payload and propulsion options to optimize response time and sensing, and choose the capability of the modular components for inclusion in the locker. This work addresses the feasibility of Phase 1 and characterizes the systems engineering efforts required to develop in-space robotic assembly.

The ISS Phase 2 technology demonstration is expected to prove the on-orbit assembly of modular reconfigurable CubeSats, increase Technology Readiness Level (TRL), and assess response time quantitatively. Phase 2 involves the development and launch of a flight unit locker with robot arms and CubeSat modular components, including propulsion options for the CubeSats themselves. The locker could be hosted at the ISS Japanese Experiment Module Exposed Facility (JEM-EF) [29][30][31] and would demonstrate the on-orbit assembly of five 1 U to 3 U sized CubeSats. The first prototype CubeSat will be a ground-based assembled structure deployed first in order to validate the locker deployment system, using the Nanoracks deployer. The four remaining CubeSats will be robotically assembled on-orbit.

Phase 3 develops the agile free-flyer spacecraft locker with robotic arms to assemble [32][33] and deploy rapid-response CubeSats. Response time is further reduced and options to mount the locker on existing satellites are considered. Showing response time that improves ground development time by 10x is a key objective. Phase 4 involves the development of a strategic constellation of free-flyer locker satellites with robotic arms to autonomously assemble and deploy CubeSats in select orbits. The goal is to demonstrate a response that improves ground time by 100x (from a minimum of 35 days to launch, instead to reach the desired initial orbit in less than four hours).

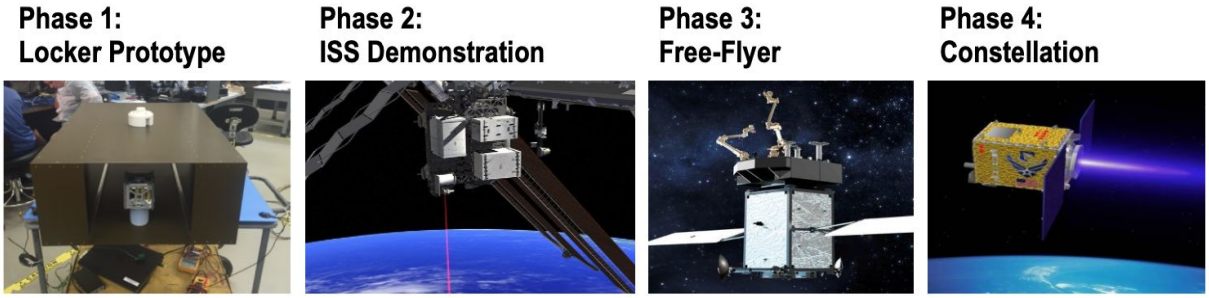

Fig. 2. The concept will progress to free-flying orbit-agnostic facilities that can rapidly deploy agile sensors. The four major steps of the project are shown: 1) Lab Prototype, 2) ISS demonstration, 3) Free-flyer with human supervision, 4) Constellation with autonomous facility.

III. Background - Selection and Timing of Robotic Arms

To date, most on-orbit assembly missions are designed to support ISS experiments, exploration, and servicing (refuel or repair existing satellites) missions [2], which are large in terms of sizing and power. Previous missions such as the U.S. Defense Advanced Research Projects Agency's (DARPA) Orbital Express program [3], the DARPA Phoenix Program [4], and the Jet Propulsion Laboratory's (JPL) Mars Insight mission [5]. Robotic manipulators, important for scientific experiments and the construction and maintenance of the ISS, have conducted on-orbit robotic assembly. Examples include the Shuttle Remote Manipulator System (SRMS) [6], also known as Canadarm, which is a 16.9-meter, seven degree of freedom (DOF) manipulator with a relocatable base; the National Space Development Agency of Japan's (NASDA) Japanese Experiment Module (JEM) Remote Manipulator System (JEMRMS), which is a 9.91-meter, six DOF manipulator; and lastly, the European Robotic Arm (ERA), which is an 11-meter, seven DOF manipulator [7]. These manipulators employed very large robotic arms to deploy, maneuver, and capture payloads. In the area of autonomy, SPHERES Universal Docking Port (UDP) demonstrates autonomous docking maneuvers using small satellites [8]. AstroBees, the free-flying robots, provide a flexible platform for research on zero-g free-flying robotics [9].

The industry's first satellite life extension vehicle, Northrop Grumman's Mission Extension Vehicle-1 (MEV-1), completed its first docking to a client satellite, Intelsat IS-901 on February 25, 2020. MEV-1 is designed to dock to geostationary satellites whose fuel is nearly depleted. MEV-1 does not make use of robot arms for its on-orbit servicing mission [10]. On-orbit robotic assembly to date is costly, as evidenced by prior and current missions [11]. For example, the Defense Advanced Research Projects Agency (DARPA) Robotic Servicing of Geosynchronous Satellites (RSGS) $400M program aims to demonstrate that a robotic servicing vehicle can perform safe, reliable, useful and efficient operations in or near the Geosynchronous Earth Orbit (GEO) environment. RSGS is using the custom-developed and large radiation-hardened Front-end Robotics Enabling Near-term Demonstration (FREND) robot arm, which is a 1.8 m arm from shoulder pitch to wrist pitch weighing 78 kg, with an additional 10 kg for electronics. The National Aeronautics and Space Administration (NASA) Goddard Space Flight Center's (GSFC) RESTORE-L servicing mission [12], is also a robotic spacecraft equipped with the tools to rendezvous with, grasp, refuel, and relocate satellites to extend their lifespan. Recently, NASA's Dragonfly has also demonstrated a ground-based test of robotic satellite assembly [13] and Made In Space (MIS) received a NASA contract to demonstrate on-orbit assembly using three robot arms to assemble 3-D printed parts in space, called the Archinaut mission [14].

Large custom-built robots need not be the only vehicles for in-space robotic assembly. The need for low-cost, low-latency and agile space infrastructure, which can reach strategic orbits such as GEO and Low Earth Orbit (LEO) in addition to polar and International Space Station (ISS) locations, could be realized using a robotic assembly of modularized components into CubeSats.

IV. Approach

We minimize the on-orbit small satellite assembly time required by dexterous robotic arms to move small satellite components within the spacecraft locker, from a shelf to the assembly workplace shown in Figure 3. This work is completed while satisfying the given power consumption and weight requirements. Secondly, we explore simultaneously optimizing assembly time of a satellite with mass, given a maximum of 2 kg for a 1 U.

Assembly time was selected as it is critical to the assertion that building small satellites on-orbit results in reduced budget and satellite development time on Earth. An example of a use case of this system is rapidly repairing damaged nodes within a constellation of satellites. We selected robotic arms, Kuka Iiwa Robot Arms and LewanSoul xArm Robots (the low-cost option), both with six Degrees of Freedom (DOF), as the basis of our modelling work. We investigated the minimization of on-orbit SmallSat assembly time by using the dexterous robot arms while satisfying the given power consumption and weight requirements at a given orbit. Given that the search parameters in the Inverse Kinematics approach - the key algorithm forming the basis of our modelling - task for a robot with many degrees of freedom is constant, a Genetic Algorithm-based approach in combination with a simulation - representing the process of the arms reaching and moving a component - was used. We also describe the technology choices and redundancy levels of the different subsystems in this optimal on-orbit assembly design.

Low-cost robotic arms such as the Lewansoul xArm shown in Figure 3 are controlled using servo motors, which are not typically used for space missions. We use the servo motors as a lab prototype test for feasibility. As we progress to space qualification, appropriate motors for space operations will be used. Robotic arms of similar size and weight come in a range of prices. A key discriminator between the robot arms available off-the-shelf is the use of powerful motors and sophisticated control systems. The more sophisticated robot arms are able to move with greater precision.

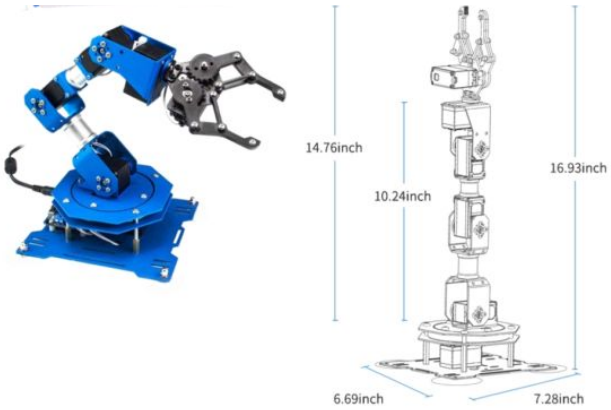

Fig. 3 (a) Lewansoul xArm Robot with 6-DOF **(b)** Robot Arm with dimensions. (Source: LewanSoul)

COTS robotic arms can be customized by adding additional sensors or swapping particular components such as a motor or link. The software used to control the arms is also usually supplied with customization for controlling system parameters; however, different control computers and real-time operating systems can be used. The interaction between the control parameters and the physical dexterity can be complex due to communication latencies and multi-tasking using the operating system. We make assumptions in the model and verify using the existing physical prototype.

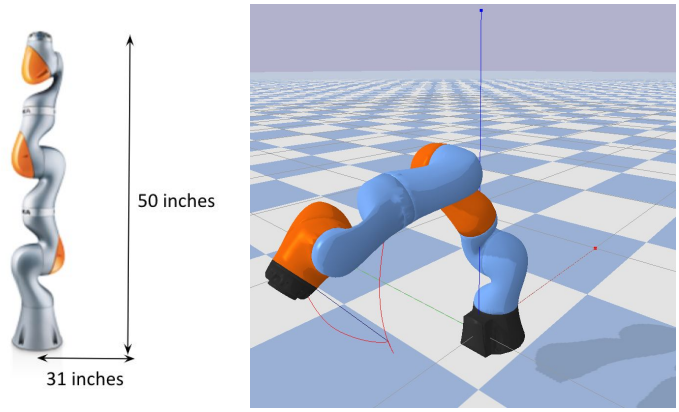

Fig. 4 Simulation of a Kuka iiwa robot arm in PyBullet with 2 DOF (Source: Kuka)

Table 1. System Design Variables and Parameters

Design Variables	Range	Comments
Joint Damping (j_d) (Nms/rad)	$0.1 < j\_d < 1$	This is the electromechanical damping present in the joint due to mechanical design.
Motor Force (mF) (Nm or kgf · cm)	$mF = 17$, $mF = 20$, or $mF = 22$	One of the off-the shelf servo motors have two torque options: 17 kgf · cm at 5 V and 22 kgf · cm at 7.4 V. Another servo motion option can provide 20 kgf · cm at 7.4 V, and we iterate with both as we optimize for assembly time.
Position Gain (pos_gain) (mm)	$0.03 < pos\_gain < 1$	This value corresponds to a PID controller proportional gain value.
Velocity Gain (vel_gain) (mm/s)	$1 < vel\_gain < 100$	This value corresponds to a PID controller derivative gain value.

This simulation describes Task 1, which is for a single step in a series of steps that are needed for the full assembly of a satellite. In order to grasp and move an object, the robot arm, which is positioned within 2 mm of the target and drop-off locations, is given a command to grasp, move and drop-off an object in Task 1. The task includes grasping a part from a shelf and bringing it to the assembly area and snapping two parts together, using some assumptions on force and alignment required for assembling LEGO-like parts together. To restrict scope of the design optimization, we test a model simulation of two degrees of freedom. Figure 4 shows a 2-DOF simulation of the xArm robot arm in PyBullet. The model simulation returns 19.09 seconds, from a starting point of 91 seconds. The output value is reasonable because the robot arm requires 5 seconds to grasp and 10 seconds to move an object to a drop-off location, and less than 5 seconds to snap-assemble the part. However, the global optimal value might be out of reach due to power constraints on the servo motors on the robot arms. During simulation with different parameters in Table 1, we see that using a powerful motor at high proportional gains results in faster (more optimal) time values but consumes more power. Conversely, using a high gain value with a weak motor results in oscillations at the take time to dampen out to within the 2 mm tolerance and hence result in higher time values. In later iterations, two robot arms and task planning to sequence the assembly steps are added and obtain similar results. We know that the simulation framework (PyBullet) can accommodate this level of fidelity in simulations because the framework has precedent [15]. Next, we modelled six degrees of freedom. Using six motors instead of two motors resulted in higher power calculations with about the same assembly time. A comparison of power and assembly time is provided in Table 2.

Table 2. Simulation results for 2-DOF and 6-DOF robot performing Task 1

Robot	Initial Assembly Time	Energy Used	Peak Power Used
2-DOF	91 seconds	220.2 Watt-seconds	13.0 Watts
6-DOF	51 seconds	451.1 Watt-seconds	33.5 Watts

We discover that the arm with 6-DOF uses more power while performing Task 1 with less time, while the 2-DOF uses a larger assembly time and less power. Given that the optimization is focused on assembly time, we select the 6-DOF robot arm for this work. After trying the AL5D 4-DOF robot arm kit, which resulted in servo burnouts after less than 100 hours of tests, we conducted a second search for available low-cost robots. We assessed a list of replacement servo motors (see Table 3) and defined low-cost for the lab prototype as under $1000 for all robots, boards and parts. We selected the HiWonder servo motors, which are used on the LewanSoul xArm robots. Table 4 lists the resulting robot arm options. The Lewansoul xArm robot arms was selected as the low-cost option and produced reliable results (more than 170 hours of tests before burnouts) and less power and thermal considerations.

Table 3. Select list of common commercial motors for robot arm use.

#	Vendor	Model	Voltage (V)	Current (A)	Condition (Current)	Max Torque	Max Torque (N.m)	Weight (Earth)	Mass (kg)	Price (USD)
1	LOBOT	LD-20MG	7.4	0.1	No Load	20 kg.cm	1.96	2.24 oz	0.06	$14.99
				1	Max					
2	Hiwonder	LX-15D	7.4	0.1	No Load	17 kg.cm	1.76	4.8 oz	0.14	$16.99
				1	Max					
3	Dynamixel	DYNAMIXEL XH430-V210-R	24	0.036	No Load	2.6 N.m	2.6	82 gm	0.08	$305.90
				0.7	Max					
4	ClearPath	CPM-SDSK-2310D-RLN	75	0.5	No Load	0.5 Nm	0.5	0.6 kg	0.6	$257.00
				1.3	Max					
5	ClearPath	CPM-SDSK-2321S-RLN	75	2.3	No Load	3.5 Nm	3.5	0.9 kg	0.9	$299.00
				2.3	Max					
6	Maxon	ECX TORQUE 22 L Ø22 mm, brushless, with Hall sensors	12	0.224	No Load	45.7 mNm	0.05	110 gm	0.11	$346.00
				3.7	Max					
7	Dynamixel	XH540-W270-T	12	0.04	No Load	9.9 Nm	9.9	0.5 lbs	0.23	$449.90
				4.9	Max					

Table 4. List of select available low-cost COTS robot arms

	Vendor	Model
A1	Franka Emika	Panda
A2	Trossen Robotics	WidowX
A3	UFactory	xArm 7
A4	UFactory	uArm
A5	LewanSoul	xArm
A6	ST Robotics	ST R17HS

Human versus Robot Assembly Time

We compare robotic assembly humans-in-the loop assembly. CubeSats are usually assembled by a team of people and not robots. There is little available information to assess how long it might take to assemble a CubeSat using robots. We begin by estimating how long it takes a human team to assemble a 1U CubeSat as a final integration step. Note that this is the final step after the common components and payload subsystems have been designed, manufactured and are ready for integration.

Table 5. Assembly time of various CubeSats by human teams

Satellite	Satellite Size	Assembly time by human teams	Source of data
NASA MarCo CubeSat	6 U	Several months by a large team	Email correspondence with JPL
Interorbital IOS CubeSat 2.0	1 U	2 days by a team of 2 people	Email correspondence with manufacturer
Planet Small Satellite	3 U	1 day for each spacecraft	Conference Paper [16]
MakerSat-1	1 U	5 minutes in International Space Station by 1 astronaut	Conference Paper [17]

From estimates obtained in Table 5, we focus on MakerSat-1, a 1 U CubeSat, which has the shortest assembly time using pre-developed subcomponents. MakerSat-1 was designed with similar intentions for rapid assembly. The first version of MakerSat-1 was released from the International Space Station and was able to collect ionizing radiation particle counts in-orbit and support an experiment on polymer degradation while operating in space for at least nine months [18]. A video demonstration of assembly under five minutes is available [19]. We use five minutes as our starting point for CubeSat robotic assembly. To achieve a simulation assembly time similar to the MakerSat-1 assembly time, we need at least 2 robot arms: one arm to hold the partly assembled satellite, and the other arm to insert and click together parts gathered from a storage location. We allow for further model refinement of robot arm functions as the grippers and different motors in the robotic arm need to be accurately modeled. Most importantly, we use five minutes as a metric for the on-orbit satellite assembly of a 1U CubeSat.

V. Design Optimization and Modeling

We optimize for two design objectives. The first objective is to minimize the assembly time of CubeSat robotic assembly, which is a surrogate for cost. The second objective is to minimize the mass of the motor assemblies. The model serving the basis of our optimization problem is implemented using the representation of a robot arm assembly in Python, relying on Inverse Kinematics [20]. The primary optimization technique used is a Genetic Algorithm-based optimization scheme [21], with time (and later time and mass simultaneously) as the objective function(s), and three constraints: robot assembly volume, power consumption, and peak power. Each potential solution of the problem represents a vector of values

$$S = \{\alpha 11, \alpha 21, \ldots, \alpha n1, \ldots, \alpha 1N, \alpha 2N, \ldots, \alpha nN\} \quad (1)$$

Since the choice of the objective function will have a decisive influence on the solution, we focus on minimizing assembly time, and later, minimizing the mass of the motor assemblies. Mass is certainly a critical parameter due to the cost of launching components into space. In addition, we can imagine that a motor able to operate at high speed, and to start and stop rapidly would need to be more powerful and be able to produce more maximum torque. Hence, there would likely be a tradeoff between the motor mass and assembly time. The design variables selected in Table 1, joint damping, motor force (torque), position gain and velocity gain, are used to model grasping a component and moving the component to the satellite assembly area of the spacecraft locker. The robot arms are required to be within a tolerance defined based on the assembly volume dimensions, 300 mm x 300 mm x

500 mm. Other constraints included in the optimization design are total power consumption, total mass (in the single-objective case), and the total budget (available to the mission). Parameters used in the model are robotic arm length, degrees of freedom of the robotic arm and the communication baud rate.

VI. Optimization Process and Modules

We identify several aspects of the system that we can simulate.

1. Articulated Rigid Body Dynamics: For each robotic arm, we create this simulation module by sourcing the parameters of COTS parts from experiments or vendor-supplied specification literature to fill in the design variables. The multi-body dynamics simulation also takes input from a layer simulation stage (feedback loop) that simulates the control system by taking forces and torques applied to the motors in the robotic arm assembly and calculating the robot state.
2. Task Sequence Planning [22]: We develop a package that lists the steps required for assembling a satellite from parts. In our design, the small satellite modular components are pre-manufactured and assembled using snap points; therefore, the robotic arm is programmed to sense, grasp and move the appropriate component to the assembly station in the right orientation for a given assembly step. As of yet, no steps require soldering, welding or gluing parts.
3. Control System: The output of the Task Sequence Planning module is routed to a control system that coordinates robots and plans paths that avoid collisions. The control system passes the output (path) of the module to individual robot arm units along with the total time required for assembly. We identify a control system simulation platform and use classical PID simulation techniques that would be modified to disallow robot collisions with the help of a multi-body dynamics simulation module. This submodule can also output the motor peak power using estimates of power usage when a motor is instructed to move at a given velocity.
4. Cost Model: This analytical module estimates the cost of a given configuration with consideration of any non-recurring-engineering costs due to uncertainties, customizations, launch to orbit, deployment and operations. We expect that some configurations can swap out components such as sensors or motors.

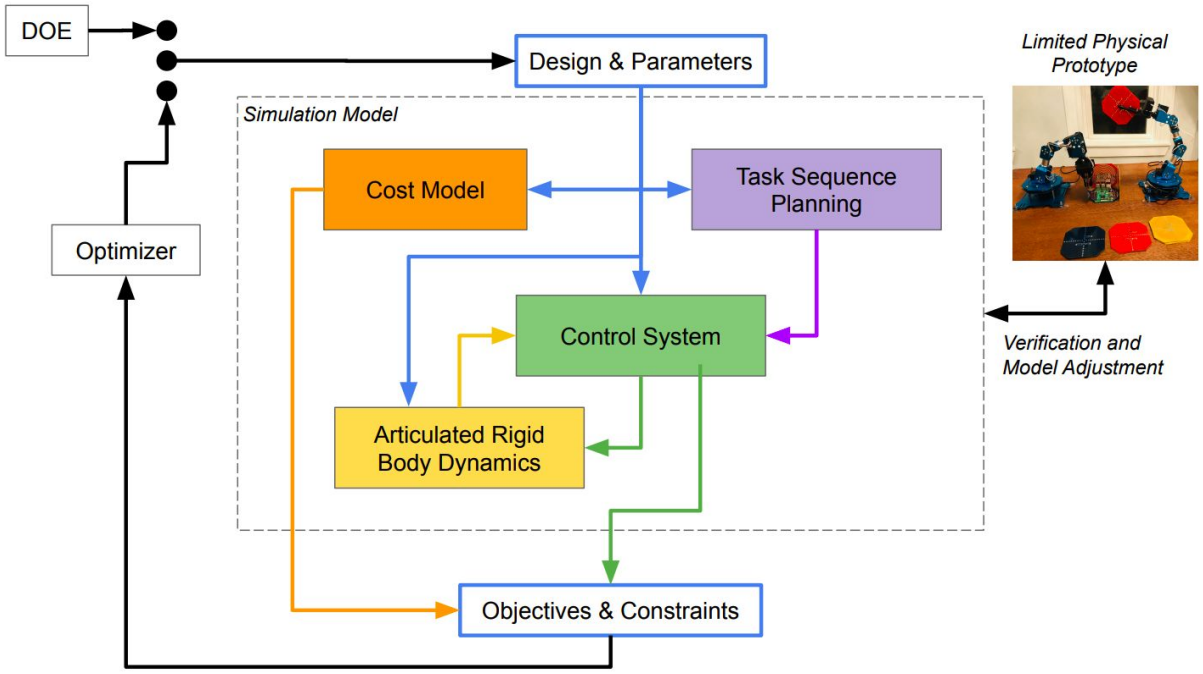

Fig. 5 Robot Arm Assembly Time Optimization Block Diagram

In this model, whole subsystems such as data communications and sensors are abstracted by the Control System and the robotic arm actuator systems are abstracted by the Articulated Rigid Body Dynamics module. We use the Design of Experiments (DOE), which is a collection of statistical techniques providing a systematic way of sampling the design space. It is useful when tackling a new problem with little available information on the design space. It is used to study the effects of multiple input variables on one or more output parameters. DOE is often a precursor to setting up a formal optimization problem. It helps to identify the key drivers among potential design variables, design variable ranges and achievable objective function values. The block diagram in Figure 5 shows feedback loops from the DOE and between all modules, particularly the Control System and Articulated Rigid Body Dynamics. The Cost Model is a separate entity while Task Sequence Planning only has inputs based on the spatial configuration.

We implemented a system model of the robot arm assembly system in Python, using an Inverse Kinematics library and a test function for the robot arm. The objective function, J, is assembly time, and the constraints are the robot assembly volume, g_1, and power (electrical energy) consumption, g_2 and peak power g_3. The system model currently Rigid Body Dynamics, Control System and Power Consumption. The Task Planning module is coded in Python. The Power Consumption module satisfies the need for control of servo motor power consumption. Servo motors are specified to have a no-load current and a maximum load. We use the inverse kinematics algorithm for the rigid body dynamics simulation. The task plan consists of two subtasks: grasping an object and moving the object to a drop-off location, where satellite assembly takes place. The robot arm end effector has to be located within 2 mm of the pickup and drop-off locations for at least 5 seconds to perform the required pickup and assembly operations. We demonstrate that the system can be executed in analysis mode through the design vector, x, which consists of the four design variables: joint damping, motor force (Torque), position gain and velocity gain.

In the model, we assume that the robot arm is constrained to move within a volume of

$$g_1 = 1.1 \text{ m}^3$$

This is automatically satisfied as the robot model has a fixed arm-length. The total power consumed is limited by the battery or other energy source available. A typical lithium ion battery used in laptops stores about 50 W hr of energy which is about 180 kJ (kilo Joules). However, given that we anticipate assembling tens of satellites, and this is a single step in assembly of a satellite, which has at least 10 steps, the constraint we chose is 180 kJ/100 = 180 J.

$$g_2 \leq 1800 \text{ J}$$

The peak power available is determined by the electrical power system within a spacecraft. For most Earth-orbiting satellites, this is generated using solar energy. We chose a peak power limit of 15 W per robot as we intend to have two robots in the system with a total power budget of 30 W. 30 W is justified based on the power availability in the ISS Nanoracks requirements [27]. We expect our assembly line to consume about the same amount of power. This is much lower than a typical industrial robotic arm. Hence, only low-cost off-the-shelf robotic arms satisfy this constraint.

$$g_3 \leq 15 \text{ W}$$

Assembly Time Feasibility

We initially address the problem of achieving a feasible solution and consider the minimization of J after the first feasible design has been achieved.

$$x_o = (0.1, 17, 0.03, 2)$$

The initial design vector x_0 is feasible because it satisfies the constraints, robot assembly volume, g_1, and power consumption, g_2 = 144.29 J and peak power g_3 = 12.96 W. The design vector returns the value, 21.27 seconds, which is a first step in minimizing J, the robot arm assembly time. Since the design is in the feasible

region, we verify that the system model has the capability to perform calculations and determine if the objective function, assembly time, can be reduced by a certain percentage and remain feasible.

VII. Design of Experiments (DOE)

Performing a DOE using an array, we efficiently explore the design space. The design variables were assigned the following notation: joint damping (A), Motor Force (B), Position Gain (C) and Velocity Gain (D). These factors were selected based on the key features of robotic arms we evaluated and design inputs for our simulation model. The levels were determined by the availability of motors. We examined the limits and best practices for our chosen system model (based on the Pybullet module). This process was partly a trial-and-error process examining the inherent limits of the variables. The design variables listed above were given levels listed in Table 6.

Table 6. List of levels for each factor

Design Factor	A1	A2	A3	B1	B2	B3	C1	C2	C3	D1	D2	D3
Level	0.1	1	10	17	20	22	0.03	0.3	3	2	1.5	1.2

Each combination of the array created was then executed using our system model, with the design variables affecting different modules. The combinations along with corresponding observations are recorded in Table 7. Based on the combinations, the main effect of each of the factor levels was then evaluated. The levels which resulted in the shortest assembly time, are highlighted in Table 8.

Table 7. Array created for the DOE with observations for each combination.

Expt no.	Joint Damping A	Motor Force B	Position Gain C	Velocity Gain D	Assembly time (s)	Total Electrical Energy (J)	Peak Electrical Power (W)
1	A1	B1	C1	D1	21.27	144.29	12.96
2	A1	B2	C2	D2	49.95	434.92	15.247
3	A1	B3	C3	D3	95.14	1565.86	16.77
4	A2	B1	C2	D3	57.43	444.06	12.96
5	A2	B2	C3	D1	89.29	1351.53	15.24
6	A2	B3	C1	D2	19.67	93.35	16.77
7	A3	B1	C3	D2	97.22	1234.42	12.96
8	A3	B2	C1	D3	20.98	110.33	15.24
9	A3	B3	C2	D1	43.04	656.79	16.77

Table 8. Main effects of each of the factors

Design Factor	A1	A2	**A3**	B1	B2	**B3**	**C1**	C2	C3	**D1**	D2	D3
Effect on Assembly time (s)	0.6	0.6	**-1.1**	3.8	-1.5	**-2.3**	**-34.2**	-4.7	39	**-3.7**	0.7	3

Based on the results in Tables 7 and 8, the optimal combination of levels for the factors is A3, B3, C1, D1. The resulting expected assembly time is 19.09 s. However, running the above combination through the model, the actual resulting assembly time is 20.01 s. This is most likely due to non-linear interaction between the different factors. Nonetheless, this is a recommended starting point (x_0) for our numerical optimization.

VIII. Genetic Algorithm

Single-Objective Genetic Algorithm

We use a Genetic Algorithm (GA) method [23][24] rather than a gradient-based method (due to the presence of discrete variables) to complete the optimization. The GA are stochastic optimization techniques introduced by Holland [25], which is inspired by genetics and natural selection. Genetic algorithms are well-suited for problems where the search space is large but has many acceptable solutions, as is the case in this work. The trajectory space is large, but there are, barring exceptional cases, numerous acceptable paths going from x_0 to x_1 without collision.

For this work, the individual is in the form of the vector S in Equation (1). Many genetic operators [26] are available; however, the more commonly used are the mutation and the cross-over operators. The mutation operator consists of randomly flipping some bits of an element of the population. The mutation rate used is 0.1. The cross-over operator consists of first randomly choosing a place where to cut the two strings of bits, and then building two new elements from this pair by simply gluing the right and the left parts of the initial pair of strings. The crossover rate used is 0.7. The fitness function evaluation contains the simulation of the robot movement and the time function evaluation. The optimizer itself was based on the Distributed Evolutionary Algorithms in Python (DEAP) package, developed by Fortin et al. at the Computer Vision and Systems Laboratory (CVSL) at Université Laval, Canada. Using the DEAP package we used binary encoding to create the chromosome for the individuals. The encoding scheme used is in Table 9.

Table 9. Encoding Used to Generate Chromosomes for GA

Design Variable:	MOTOR FORCE	JOINT DAMPING	POSITION GAIN	VELOCITY GAIN
Bits per variable	16	32	32	32
Encoding scheme	Bits *0 - 21845:* 17 Bits *21846 - 43690*: 20 Bits 43691 - 65535: 22	Next 32 bits with a continuous range between 0.1 and 10	Next 32 bits with a continuous range between 0.03 and 3	Next 32 bits with a continuous range between 1.2 and 2

Multi-Objective GA (MOGA)

A broader trade-space was obtained by extending the list of motors in Table 3 from that used in the single-objective GA. The program was adjusted to accept these data. The DEAP Python library that we used for single objective GA can do multi-objective GA without much customization. Due to the design variables and constraints, our initial attempts to obtain a Pareto front yielded sparsely populated graphs. To better explore the design-space and obtain a discernible Pareto front, we to increase the peak power constraint to 1 kW, and the energy constraint to 10 kJ. We based the new value on data from literature [24] on the power constraints for experiments flown on the International Space Station (ISS), a potential test environment for initial models of our assembly. These higher values are also feasible through an optimization of the solar array sizing of the future spacecraft equipped with the robotic arms.

VII. Results

We reran the optimization using the setup for several iterations, obtaining the results in Table 10, where J is optimal assembly time and the four design variables are identical to those introduced in the sections above. Using a constraint of $g_1 \leq 15$ W, the new optimal assembly time was close to $J_{tmin} \leq 50$s. This assembly time is

significantly higher than our previous optimum, which however, was expected with the power constraint. To fine-tune this, we varied the constraint maintaining the GA inputs and found that if we increased the constraint to 16 W, the optimum found by the GA at 100 generations and 20 individuals per generation was closer to 22 seconds.

Table 7. Array created for the DOE with observations for each combination.

Design Variable	**Value**
*x**	[7.549 (damping), 1.961 N.m (max motor force), 0.157 (position gain), 1.943 (velocity gain)]
J(x)*	21.950 s
g1(x)*	15.247 W < 16 W
g2(x)*	187.993 J < 1300 J

One of the key issues we faced in the optimization is the presence of maximum motor force as a discrete variable; hence, the use of the GA method. This is because we chose from a set of discrete options from COTS motors. We assume the secondary motor characteristics (e.g. current drawn, from which power consumption is calculated) to be proportional to the COTS robot arm motor's force. Therefore, the motor has the same torque to current ratio. Next, we calculate the diagonals of the Hessian matrix using a Finite Difference Method, beginning with a small delta in each variable, $h = 0.1$ and decreased to $h = 0.01$ and $h = 0.001$, etc. to check grid convergence, and use the central difference formula.

$$\frac{\partial^2 J}{\partial x_i^2} = \frac{J(x_1,x_2,x_i+h,...)-2J(\mathbf{x})+J(x_1,x_2,x_i-h,...)}{h^2}$$

The results for the diagonals of the Hessian are:

```
[      9.0      50.4    1079.0      78.0 ] # h = 0.1
[    200.0    1500.0   10300.0     200.0 ] # h = 0.01
[  20000.0   70000.0   80000.0   20000.0 ] # h = 0.001
[2000000.0 2000000.0 2000000.0 2000000.0 ] # h = 0.0001
```

We see that as we decrease h, the double derivative "explodes". It is suspected this could be due to the precision used in the simulation packages for kinematics. To obtain the system's physical values, we use $h = 0.1$ as the basis for normalization. Given that in the diagonals of the Hessian, only the position gain value is greater than 100, the scaling required is as follows where the position gain is multiplied by 10:

```
[ 1.0  1.0  10.0  1.0 ]
```

We rerun the code with scaling = 1.0 and the best solution that resulted was:

```
SCALING = [ 1.0  1.0  1.0  1.0 ]
x*  =  [1.516 (damping), 1.961 N.m (max motor force), 0.0524 (position
gain), 1.929 (velocity gain)]
J(x*)  =  21.01 sec
g1(x*) =  15.25 W < 16 W
```

g2(x*) = 109.60 J < 1300 J

In Figure 6, we observe that there is a small difference in the optimum assembly time achieved - with and without scaling - which could be due to the random nature of genetic algorithms. The data for the GA iterations shows that in both the scaled and non-scaled cases, there are some mutations that result in deviation from optimal, but the solution converges back to a similar optimum value. We hypothesize that our implementation of the genetic algorithm where the binary bits were spread between and minimum and maximum feasible values of the variable resulted in the inclusion of scaling. Despite the scaling that is based on the Hessian matrix being included, where most of the scales were 1.0 and only one scale was 10.0, we observed no changes.

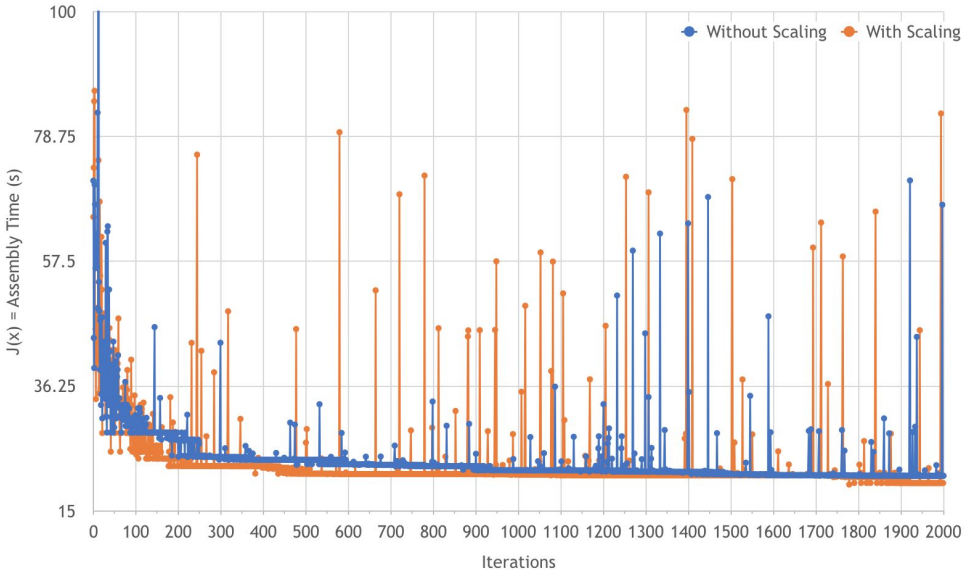

Fig. 6 Genetic Algorithm results *J(x)* with penalty for constraints and iterations, where every 20 iterations is a generation, with 2000 iterations in 100 generations

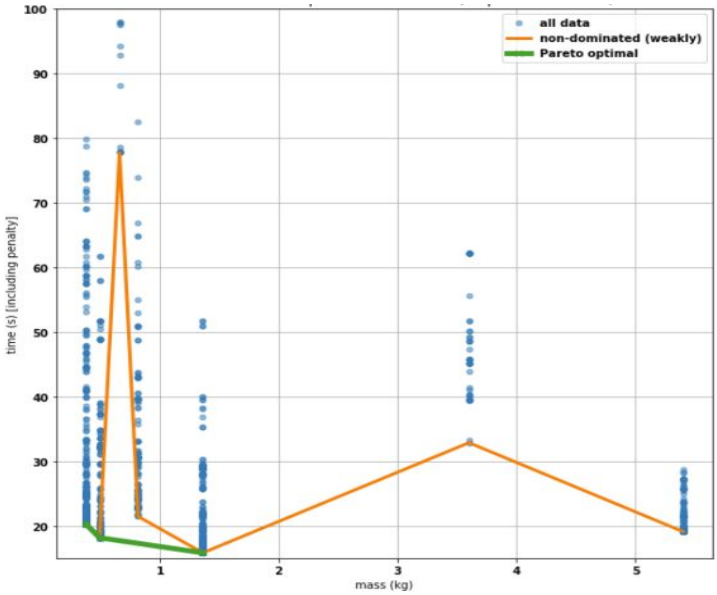

Fig. 7 Genetic Algorithm results J(x) with penalty for constraints and iterations where every 20 iterations is a generation, with 2000 iterations in 100 generations

Performing the multi-objective optimization, we obtain the results presented in Figure 7. We can see from the plot that we can optimize the time to be slightly less than 16 s by choosing a mass that is slightly higher than 1.25 kg and can optimize mass to about slightly more than 0.25 kg by staying at about 20 s in time. The trade-off is not strictly convex because not all non-dominated solutions lie on the Pareto-optimal front. If we consider only the Pareto optimal points, the front is convex. We considered all Pareto points and selected the middle point near the knee as the optimal solution. This is because at this point, we can reduce mass by a lot without compromising too much on the assembly time. Also, the middle point is likely to have more feasible solutions nearby (other COTS parts that result in a similar performance) than the more extreme points. Deviating a little from the much higher mass and the lowest mass solutions could result in a sharp rise in assembly time (e.g. due to the motor not being strong enough and needing to move very slowly) or designs that require a much higher mass (due to the use of more powerful motors than required).

This design objective reaches an optimal solution of 22 seconds in the final run, for assembly of each component part, with penalty for constraints. During the simulation, we see that using a given baseline servo motor (7 V) at high proportional gains results in optimal assembly time of approximately 20 seconds per component assembly, compared to roughly double this time per component for a 1 U CubeSat weighing 2 kg. The key parameter affecting our results proved to be the number of generations, rather than the population size. In the first run, we did not alter the parameters controlling the mating. In the first run, due to limitations in available computational time, we used a population of 20 and 100 generations. We concluded that the optimum achieved is - given the relatively small number of population size - close to the global optimum. This can be inferred from the results of the presented DOE and considerations of physical limitations in the assembly setup. However, this simulation resulted in 25% higher power consumption.

Oscillations and Lack of Damping—Using a high gain value with a lower voltage (5 V) motor results in oscillations and additional time required to dampen out to within the given tolerance, and results in increased assembly time. Since damping control is needed to prevent oscillations from becoming hazardous, the joint damping is set sufficiently large (0.1<joint_damping<1) so as to reduce the amplitude of the oscillations of the vibration modes and facilitate accurate position tracking when needed. We observe that changing the joint damping (and position gain) affect these oscillations. Our results indicate we gain high frequency performance improvement by a joint damping increase such that lateral oscillation amplitude is significantly reduced. Additionally, we observe that low-cost robots must operate in low-wind areas or in boundary-layer flow near surfaces to prevent oscillations.

VIII. Conclusion

Final recommendations on the optimization process in the three key areas (1) modelling, (2) single-objective optimization and (3) multi-objective optimization are as follows. Designers should take full advantage of the Inverse Kinematics technique for robotics programming. In our work, we obtained Inverse Kinematics by using PyBullet. Performing single-objective optimization, better results can be obtained when a greater number of options are presented in the COTS motor range. Performing GA-based single-objective optimizations, it was found that generation number had a strong effect on results. Conducting the multiobjective optimization part of our work, we found that mass was indeed a critical parameter for space operations given the deterministic nature of mass on launch costs. We found that a wider range of motors with different specs would yield denser Pareto front, which in turn yield better results, potentially also meeting n-KKT conditions (which we were unable to meet with present results).

The benchmarked small satellite assembly time with a human-in-the-loop requires weeks to months of component assembly and integration time on Earth. We see that on-orbit assembly capability, within a spacecraft - optimized to assemble a 1 U CubeSat with 30 W of total power, would reduce the assembly time by an order of magnitude. Investigating initial simulations with robotic arm models, for a 1U CubeSat assembly, we show a savings benefit in robotic assembly time from 51 s to 20 s. The best solution for this discrete problem is a robotic assembly with an optimal CubeSat development and deployment process. The solution to this problem satisfies single-route constraints, collective constraints, and the mission budget constraint. A constraint that the on-orbit assembly must be conducted by two dexterous robot arms and must take five minutes or less to assemble a 10-component CubeSat is

satisfied. We optimize the on-orbit assembly time of small satellites using dexterous COTS robot arms while satisfying the given power consumption and weight requirements and minimizing humans-in-the-loop. Assembly time is selected as it is critical to the assertion that building small satellites on-orbit results in reduced cost and satellite development time on Earth.

For future work we have identified three key areas. The first area is the simulation incorporating interaction of two arms would be necessary to create a fully representative model of the planned assembly. Further iteration on a less-explored domain of the problem: the placement of the components. The second area is the addition of the task sequencing as part of a coupled optimization. The third area is the prototyping of a robotic assembly of CubeSat components within the optimal assembly to be used to validate the optimization results.

References

[1] Barnhart, David, et al. "Changing satellite morphology through cellularization." AIAA SPACE 2012 Conference & Exposition. 2012.

[2] Weisbin, Charles R., and Guillermo Rodriguez. "NASA robotics research for planetary surface exploration." IEEE Robotics & Automation Magazine 7.4 (2000): 25-34.

[3] Whelan, David A., et al. "Darpa orbital express program: effecting a revolution in space-based systems." Small Payloads in Space. Vol. 4136. International Society for Optics and Photonics, 2000.

[4] Barnhart, David, et al. "Phoenix program status-2013." AIAA SPACE 2013 conference and exposition. 2013.

[5] Smrekar, Sue, and B. Banerdt. "The InSight mission to Mars." The 8th Mars Conference. Vol. 18. 2014.

[6] Sallaberger, Christian, Space Plan Task Force, and Canadian Space Agency. "Canadian space robotic activities." Acta astronautica 41.4-10 (1997): 239-246.

[7] Laryssa, Patten, et al. "International space station robotics: a comparative study of ERA, JEMRMS and MSS." *7th ESA Workshop on Advanced Space Technologies for Robotics and Automation*. 2002.

[8] Rodgers, Lennon, Simon Nolet, and David W. Miller. "Development of the miniature video docking sensor." *Modeling, Simulation, and Verification of Space-based Systems III*. Vol. 6221. International Society for Optics and Photonics, 2006.

[9] Bualat, Maria, et al. "Astrobee: Developing a free-flying robot for the international space station." *AIAA SPACE 2015 Conference and Exposition*. 2015.

[10] Northrop Grumman. "Companies demonstrate groundbreaking satellite life-extension service." [Online]. Available: https://news.northropgrumman.com/news/releases/northrop-grumman-successfully-completes-historic-first-docking-of-mission-extension-vehicle-with-intelsat-901-satellite

[11] Flores-Abad, Angel, et al. "A review of space robotics technologies for on-orbit servicing." Progress in Aerospace Sciences 68 (2014): 1-26.

[12] Parrish, J. "Robotic Servicing of Geosynchronous Satellites (RSGS)." Defense Advanced Research Projects Agency (DARPA).[Online]. Available: https://www. darpa. mil/program/robotic-servicing-of-geosynchronous-satellites.

[13] B.E. Kelm, et al. FREND: Pushing the Envelope of Space Robotics. Space Research and Satellite Technology. 2008 NRL Review.

[14] Reed, Benjamin B., et al. "The restore-L servicing mission." AIAA SPACE 2016. 2016. 5478.

[15] James, Stephen, et al. "Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019.

[16] Gilmore, Cheser, et al. "Flexible, High Speed, Small Satellite Production." (2019).

[17] Grim, Braden, et al. "MakerSat: A CubeSat Designed for In-Space 3D Print and Assembly." (2016). 30th Annual Conference on Small Satellites.

[18] Earth Observatory Portal Directory. "MakerSat". [Online]. Available: https://directory.eoportal.org/web/eoportal/satellite-missions/m/makersat

[19] NNU. "NNU's MakerSat-1 CubeSat Assembly." [Online]. Available: https://youtu.be/shLPETczsF4

[20] Rodgers, Lennon, Simon Nolet, and David W. Miller. "Development of the miniature video docking sensor." Modeling, Simulation, and Verification of Space-based Systems III. Vol. 6221. International Society for Optics and Photonics, 2006.

[21] Števo, Stanislav, Ivan Sekaj, and Martin Dekan. "Optimization of robotic arm trajectory using genetic algorithm." IFAC Proceedings Volumes 47.3 (2014): 1748-1753.

[22] Spensieri, Domenico, et al. "Optimal robot placement for tasks execution." Procedia CIRP 44.Supplement C (2016): 395-400.

[23] Goldberg, David E., and John Henry Holland. "Genetic algorithms and machine learning." (1988).

[24] Eiben, Agoston E., and James E. Smith. Introduction to evolutionary computing. Vol. 53. Berlin: springer, 2003.

[25] Holland, John. "Adaptation in natural and artificial systems: an introductory analysis with application to biology." Control and artificial intelligence (1975).

[26] Davidor, Yuval. "Analogous crossover." Proceedings of the Third International Conference on Genetic Algorithms. 1989.

[27] Nanoracks. “How to Build a NanoLab Payload.” [Online]. Available: https://nanoracks.com/wp-content/uploads/How-to-Build-a-NanoLab-Payload.pdf

[28] Ceccacci, Anthony, Dye, Paul. “Contingency Shuttle Crew Support (CSCS)/Rescue Flight Resource Book.” National Aeronautics and Space Administration (2005): 89.

[29] Kawasaki, Kazuyoshi. "Overview of JEM-EF on ISS." Proceedings of the RIKEN Symposium. Saitama. 2008.

[30] Steimle, Per C., et al. "Commercial Approach to Research Outside the International Space Station-A Small Size Precursor Service For Future In-Orbit Testing." AIAA SPACE 2014 Conference and Exposition. 2014.

[31] Steimle, Christian, and Uwe Pape. "ISS External Payload Platform-a new opportunity for research in the space environment." 40th COSPAR Scientific Assembly. Vol. 40.

[32] LeMaster, Edward, David Schaechter, and Connie Carrington. "Experimental demonstration of technologies for autonomous on-orbit robotic assembly." Space 2006. 2006. 7428.

[33] Sun, Yongjun, et al. "Design and optimization of a novel six-axis force/torque sensor for space robot." Measurement 65 (2015): 135-148.