Technologies for Mars On-Orbit Robotic Sample Capture and Transfer Concept*

Rudranarayan Mukherjee
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
Rudranarayan.M.Mukherjee@jpl.nasa.gov
(818) 354-2677

Neil Abcouwer
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
Neil.Abcouwer@jpl.nasa.gov
(626) 755-5821

Junggon Kim
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
Junggon.Kim@jpl.nasa.gov
(818) 393-4146

Ryan Mccormick
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
Ryan.L.Mccormick@jpl.nasa.gov
(818) 354-5945

Peter Godart
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
Peter.T.Godart@jpl.nasa.gov
(818) 354-5017

Philip Bailey
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
Philip.Bailey@jpl.nasa.gov
626) 460-9653

Abstract*—Potential Mars Sample Return (MSR) would need a robotic autonomous Orbital Sample (OS) capture and manipulation toward returning the samples to Earth. The OS would be in Martian orbit where a sample capture orbiter could find it and rendezvous with it. The orbiter would capture the OS, manipulate it to a preferential orientation for the samples, transition it through steps required to break-the-chain with Mars, stowing it in a containment vessel or an Earth Entry Vehicle and providing a redundant containment to the OS (e.g., by closing and sealing the lid of the EEV). In this paper, we discuss component technologies developed for in-laboratory evaluation and maturation of concepts toward the robotic capture and manipulation of an Orbital Sample. We discuss techniques for simulating 0-g dynamics of a spherical OS, including contact, in a laboratory setting. In this, we leverage a 5dof gantry system and, alternately, a 6dof KUKA robotic arm to simulate the OS motion. Both the gantry and robotic arm are mounted with a force-torque sensor that enable detection of contact and provide measurements to simulate, in hardware, the 0-g OS dynamics. We present results that demonstrate the validity of our approach and the extent to which we are able to simulate 0-g dynamics in a laboratory setting. We also discuss techniques for detecting and tracking the OS using optical sensors and LIDAR from near-capture distances. These are discussed in the context of individual sensors as well as fusion of multiple sensor readings. Results of hardware experiments with different sensors are presented. Further, we discuss an uncertainty quantification based physics modeling capability for quantitative evaluation of different concepts for OS capture and manipulation. The computational models are based on high-fidelity multibody dynamics simulations of the OS, robotic elements and their contact mechanics. We present results that demonstrate our effective use of computational simulations in a complementary manner to hardware experiments. Finally, we present a cyber-physical approach to concurrently fusing hardware elements, computational simulation elements and autonomy software to effectively and rapidly simulate end-to-end systems concepts for end-to-end orbital sample capture and manipulation system concepts.

TABLE OF CONTENTS

1. INTRODUCTION

With the success of the Mars rover missions, interest has grown again in formulating a series of potential missions that would acquire samples from Mars and return them to Earth. These are commonly referred to as potential Mars Sample Return (MSR) [1]-[2]. Mars 2020 will take samples, seal them in tubes and leave them on the Martian surface. Concepts currently being considered involve another rover acquiring these tubes, storing them in a cache and loading the cache on a Mars Ascent Vehicle (MAV) in an Orbital Sample (OS) cache. The MAV could then launch the OS into Martian orbit where a subsequent orbiter would rendezvous with it and robotically capture it. Along with the payload for rendezvous and capture, the orbiter would also have robotic manipulation capabilities to orient the OS, within bounds, of a predetermined orientation, take the OS through various planetary protection steps (commonly referred to as Breaking the chain or BTC with Mars), stow it in an Earth Entry Vehicle

* Pre-decisional: for information and discussion purposes only

(EEV) or an alternate Secondary Containment Vessel (SCV), and finally eject the EEV or the SCV. Either the EEV would return the samples to Earth or subsequent mission(s) would return the SCV to Earth. In this paper, we discuss our ongoing efforts, and preliminary results, within a JPL internally funded research and technology development effort towards developing technologies and test-beds towards supporting the orbital near-phase rendezvous, capture and manipulation of the OS as described above. Specifically, our interests lie in the events starting from the OS being in the range of 10m or so away from the spacecraft through the robotic capture and manipulation of the OS. We do not focus on the BTC or planetary protection steps. As the orbital rendezvous and capture elements are not finalized, or committed to a mission, we use information consistent with current (and evolving) formulation efforts. Consistent with past MSR efforts and previous technology efforts, we focus on a spherical OS with a notional size of 27cm and a notional mass of 12 kg. Based on other engineering studies, we bound the relative bore-sight translational speed between the OS and the spacecraft to be no more than 10cm/s and assume that the OS may have a rotational speed of 10rpm.

2. OS Dynamics Test Beds

To simulate the OS 0g free space dynamics in a laboratory setting for the imminent capture phase, we set up two test beds. In one case, we use a robotic gantry system that provides 3 translational degrees of motion driven by DC motors with encoders. We developed a design for 2dof rotational OS as shown in figure (1). The design fundamentally consists of two rotational stages mounted at 90 degree to each other. The vertical stage is mounted on to the gantry end effector and enables a continuous rotation about the vertical axis. The vertical stage consists of a 6 axis force-torque sensor and a thin-gap motor. The design of the thin-gap motors provide very low resistance to free rotation when the motor is not turned on. This enables us to either drive the vertical stage at a desired speed or to get to a desired rotational speed and allow the OS to freely coast after that by turning off the motor. The horizontal stage is passive and rotates freely. An encoder is mounted on the horizontal stage to measure the angular rotation. Two hemispherical shells were fabricated and mounted on the horizontal stage to get the near spherical form-factor of the OS. We also instrumented the OS with a wireless accelerometer. The resulting system, of the 3 actuated gantry stages, 1 actuated rotational OS stage and 1 passive OS rotational stage were capable of simulating 5 dof OS free space dynamics within the bounds discussed above. A second test bed was developed using a 6dof KUKA industrial robot. In this case, we mounted a metal sphere, as an OS surrogate, to the end of the KUKA arm. The KUKA arm was tested and set to provide Cartesian linear motion of the OS model. The final actuator on the KUKA arm provides one rotational degree of freedom. This system was thus able to provide 4 dof OS freespace 0g dynamics. For verification, of the OS free space dynamics on the gantry system, we used a VICON camera setup to measure the OS trajectory and compare the results with the kinematics obtained from the gantry motor encoders. These were found to be consistent with the commanded trajectories and speeds, and a representative result is shown in figure (2) while the error between the encoder based and VICON based trajectories are shown in figure (**??**). We developed the capability to generate coordinated 3D motion of the gantry system such that it could move the OS along a straight line in any Cartesian direction. For the KUKA based test bed, we relied on the manufacturer's precision and repeatability measures for OS motion.

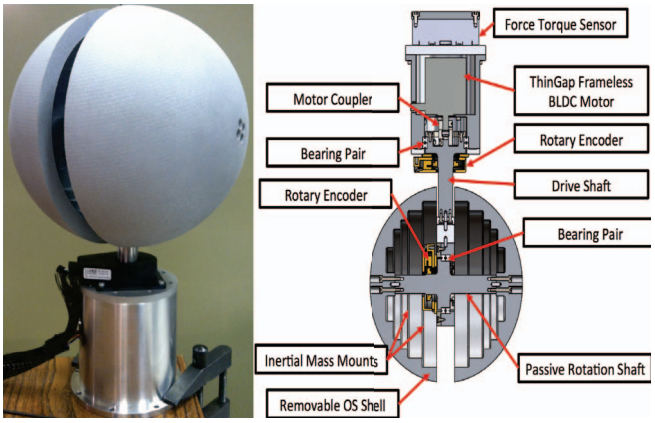


Figure 1. Image shows the OS simulator capable of two rotational degrees of freedom

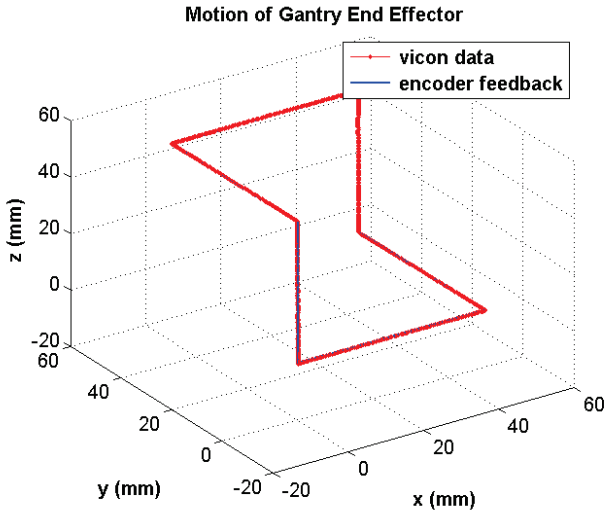


Figure 2. Trajectory comparison is shown between the data recorded using a VICON system and that obtained from the gantry encoders

3. OS Tracking

Our efforts have focussed on tracking the OS with LiDAR and cameras with a relative separation of about 10m. This is primarily aimed at algorithm development and quantifying sensor performance in a laboratory setting. As a representative LiDAR, we used a SpectroScan3D MEMS Scanning LiDAR. The raw data from the lidar is read into memory as a point cloud and three perception algorithms can be performed on the data. Once these various estimates of OS position are returned, they fused with a Kalman filter [8]. The filter provides estimates of the underlying state, including OS velocity.

Closest Point—The simplest algorithm simply finds the closest point to the LiDAR, assumes that the point is on the surface of the OS, and adds the OS radius appropriately to estimate the 3D position of the OS. This algorithm is simple, but is weak to noise on that single point and weak to clutter (necessary in lab testing for support and motion) in the field

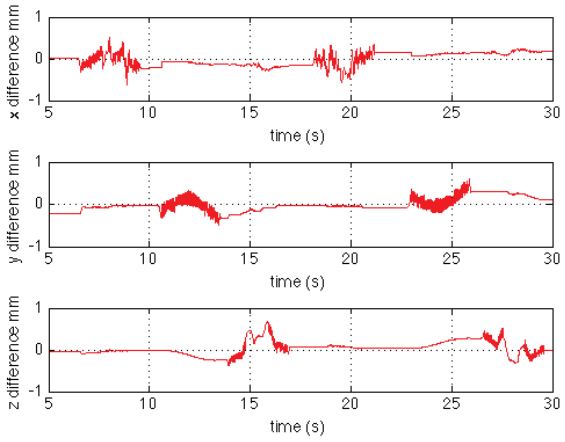


Figure 3. Error between VICON data and encoder based data

of view, but could be used in flight if any visible orbiter components are blacklisted.

RANSAC—The RANSAC algorithm[5] is applied by sampling random points in the cloud, fitting the equation of a sphere of known radius to those points, checking which points in the cloud are consistent with the model (inliers), and generating a new model from the inliers. These steps are repeated until the inlier set stops growing. This is repeated with different initial points, and the best model found is returned. Initial points can be seeded with estimates from other algorithms.

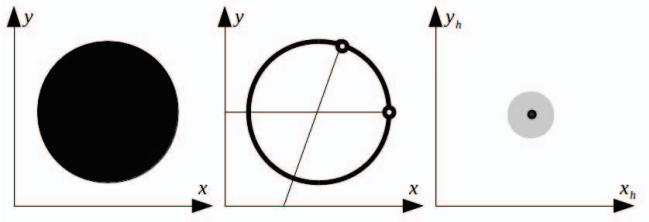


Figure 4. The Hough Algorithm applied to OS detection. OS depth image, left. The Sobel edge detector is used to find edges and the gradient is used to project lines, center, and accumulate votes in the Hough space, right. The maximum is chosen from the Hough space and converted to a 3D position estimate.

Hough Transform—The Hough transform[6] is used to extract circles or other features in the images. This can be applied to a 2.5D depth image interpretation of LiDAR data, Figure 4. First the depth image is smoothed to reduce noise, and edges are extracted using the Sobel operator[7]. Secondly, at the location of edges, the depth gradient is found, and lines are projected in the direction of negative gradient. The depth at the edge is used to estimate the radius of the OS in pixels if the OS is indeed at that location, and used to accumulate "votes" in the Hough parameter space. The Hough space maximum is converted to 3D coordinates and returned as the estimate of OS position.

We used a set-up as shown in figure (5) to conduct a set of experiments to quantify the LiDAR OS tracking performance. We painted the metal OS in a matte finish to remove reflective

Figure 5. The OS is mounted on the KUKA robotic arm and driven in a straight line trajectory towards the LiDAR

issues. Once LiDAR and KUKA data have been logged and converted to MATLAB, we apply known transformations to convert the KUKA positions to the LiDAR reference frame, correct for time differences, do a second corrective 3D transform, extract the subset of motion corresponding to the actual trials, and extract error statistics for each LiDAR algorithm. Our approach removes biases from static errors in the transforms between the LiDAR and the KUKA robotic arm to return the OS relative tracking performance.

Figure 6 shows the temporal history of comparison between the trajectories of the OS from the KUKA robotic arm encoders and the LiDAR measurements. The results of a nominal run are summarized in figure (7) while figure (8) shows the scatter plot of the error from the RANSAC algorithm as a

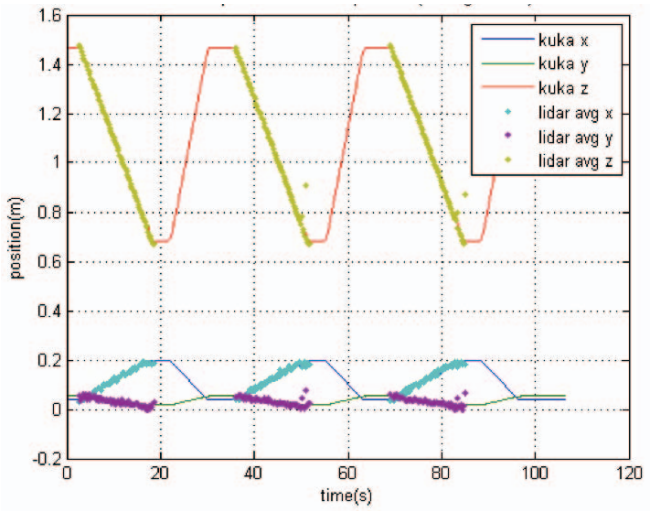


Figure 6. Three trajectories commanded to the KUKA robotic arm and those measured by the LiDAR are compared in the graph

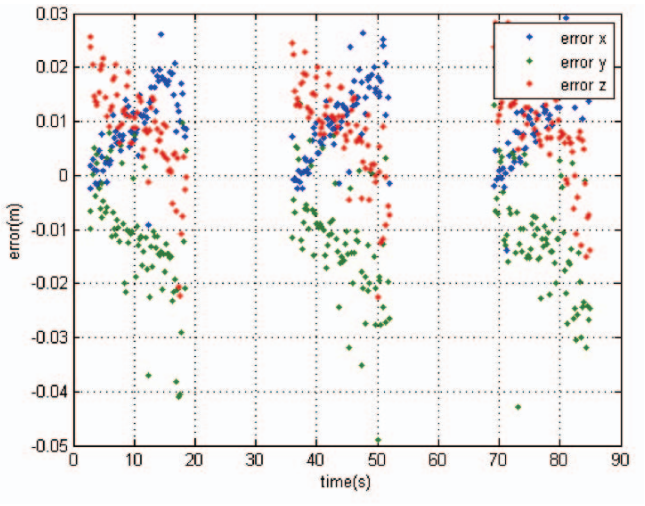


Figure 8. The scatter plot shows the error between the KUKA arm based trajectory and that measured by the LiDAR as a function of time for 3 test cases based on the RANSAC algorithm

representative data set.

	Mean Error (m)			Error Standard Dev.		
Algorithm	**X**	**Y**	**Z**	**X**	**Y**	**Z**
Closest	**0.0144**	**0.0037**	**-0.0121**	**0.0161**	**0.0148**	**0.0055**
Hough	**0.0142**	**0.0048**	**0.0015**	**0.0125**	**0.0196**	**0.0571**
RANSAC	**0.0103**	**0.0119**	**0.0089**	**0.0073**	**0.0104**	**0.0088**

Figure 7. The mean error and standard deviations in error obtained from the three different algorithms are shown in this figure

Similarly, we set up a test bed for camera based OS detection and tracking. We set up a dark room with a light source and an OS in a manner that the relative yaw and distances could be appropriately set. For detecting the OS our current approach is follows: Given that we are aware of the position of Mars and any spacecraft parts in the image, we find the approximate area of interest in the image which contains the OS by searching for the largest bright object visible. This gives a lower and upper bound on the size of the OS in the image, and an approximate position. Then, taking the field of view of the camera, the direction the camera is facing, and the relative position of the sun, we generate a patch that is approximately what the lighted region of the OS would look like in only sunlight. We do this by rotating a circle in 3D space and projecting it back onto the original viewing plane. Instead of just using a flat template, we make the template a step function, which has a value of 1 at the edge up to N at the center. Using this template, we can create a histogram of how many pixels would be in the image given an OS of a size corresponding to a number of pixels in diameter. Now taking the original image, and the region identified from the first step, there are two steps: calculating the diameter of the OS in pixels, and finding the exact center of the OS. Finding the diameter is done by counting the number of bright pixels in the test region, and comparing to the histogram created in the previous step. Then, calculating the centroid of the bright pixels in the test region, one can find the exact center of the OS by using the difference between the centroid of a template with the calculated diameter and its corresponding center. Once you have the OS diameter and position in pixels, the direction of the camera and field of view can be used to calculate the OS position in spaceship relative coordinates. Figure (9) summarizes our approach.

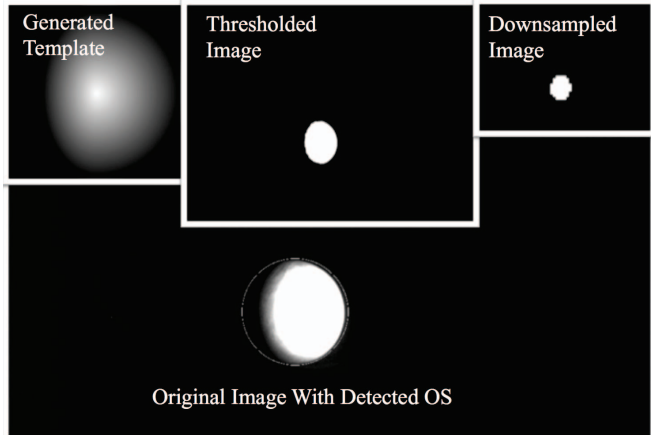


Figure 9. Summary of the overall approach for optical OS detection showing the original image, and the three stages of data processing by the algorithm

4. Capture Test Bed

We developed an approach to simulate 0g contact dynamics in the laboratory based on force-torque detection and reacting to the measured loads in a manner consistent with 0g dynamics. Towards this, we used our gantry and KUKA arm based OS hardware simulators for the pre-contact dynamics of the OS. Using the gantry, for example, we achieve the 5dof desired initial conditions in translational speed and angular velocity. As a first step, we ascertained that the gantry and the KUKA arm had the mechanical bandwidth to simulate a 0g dynamics by being able to pull a 1g bounce maneuver. As seen in figure (10), we validated the ability of the gantry to demonstrate this capability. As seen in the graph, the gantry is able to produce a 1g acceleration against gravity at the top end of the expected OS relative speed.

When the OS makes contact with the capture hardware,

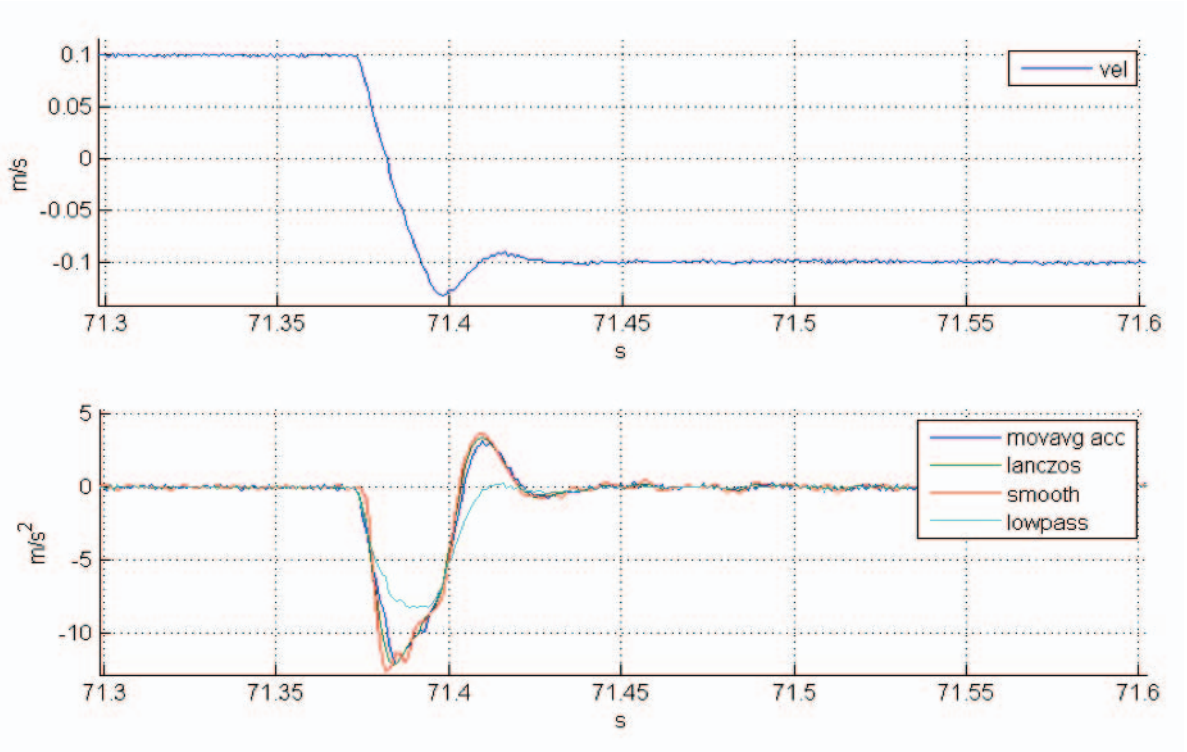


Figure 10. **The graph shows validation of the ability of the gantry system to generate 1g gravity offset acceleration. As seen, the gantry transitions from 10cm/s velocity to -10cm/s in 20ms giving a net 10m/s^2 acceleration against gravity**

discussed next, the interaction loads (forces and torques) are measured using the embedded force-torque sensor. These loads are then used in conjunction with an on-board OS 3D dynamics simulator that provides the physically correct dynamic response of the OS. The gantry is then driven in a manner consistent with the simulated OS behavior to obtain a 0g OS dynamics in the laboratory. While this is the overall architectural approach, we made two changes to meet the desired results. First, instead of reacting the measured torque and attempting to simulate the reaction, we allowed the OS to become passive during the contact phase to get the actual torque reaction. The reason is two fold: first gravity does not effect the rotational dynamics and second, the surface friction behavior is difficult to model in high fidelity. Hence, using our OS simulator, we drove the OS to its desired angular speed (less than 10rpm) and turned the thin-gap motor off just before contact. This enabled the OS to freely rotate as the thin-gap motor has very low friction. The second major implementation change was the nature of the force feedback algorithm. We found that the frequency of the force-torque sensor and the control loop was not fast enough to respond accurately in impact scenarios. This was particularly acute for the KUKA controller that has a 250Hz control loop. In such cases, we used a coefficient of restitution model to simulate the impact. The measured loads, from the sensors, were used to obtain the directions of motion involved in the contact and a prescribed coefficient of restitution was used to simulate the dynamic response. Figure (11) shows the resulting outcome where the desired and realized trajectories (superimposed) are shown for different prescribed coefficients of restitution. The results are excellent, and it is difficult to discern the error between the trajectories.

Given this contact dynamics approach, we pursued two different types of contact test beds based on (i) mechanically accommodating the uncertainty in relative OS position with respect to the spacecraft i.e. using a large mechanical capture volume or (ii) using better autonomy and sensing to precisely capture the OS. An example of a design of item (i) is a cone based capture device, and we built a test bed for that to understand the contact dynamics of the OS as it bounces around the larger mechanical capture volume as shown in figure (12). An example of item (ii) is using two small shells that are brought together autonomously by tracking the OS position and velocity accurately using sensing as shown in figure (13).

In case the resulting uncertainty from the spacecraft control and relative OS tracking estimate are such that a large mechanical capture volume is warranted, we used a mechanical cone with a lid as a capture device. This is consistent with a similar design pursued in [3]. Figure (12) shows our test bed. The 5dof OS has been previously described. The capture cone is a simple mechanical device that allows the OS to enter from the larger end. The larger end of the cone is sized to accommodate the OS diameter and the 3 sigma bounds in the uncertainty of OS relative position. The rim of the cone has a laser curtain (not shown in figure) such that the OS entering the cone breaks the laser curtain. This in turn triggers a lid that traps the OS in the cone and guides it to a restrained volume. The OS has a potential to bounce back and forth between the capture cone and the lid until the lid traps the OS in the narrow end of the cone. Each bounce causes the OS to slow down, thereby iteratively bringing it to rest. A mechanical lid is not shown in the test bed image as we used a simulation-in-the-loop hardware test bed approach where we used computer simulations of the lid. This is discussed in a later section and it allows us to try out different lid geometries and mechanisms effectively and quickly to support a trade study.

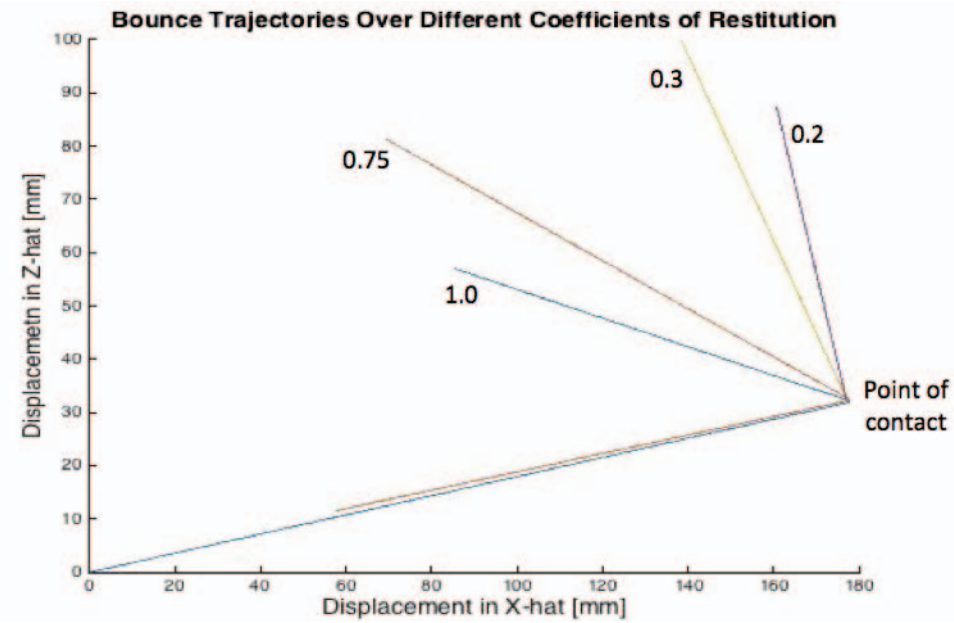


Figure 11. **Figure shows bounce trajectories from the KUKA arm for different coefficients of restitution**

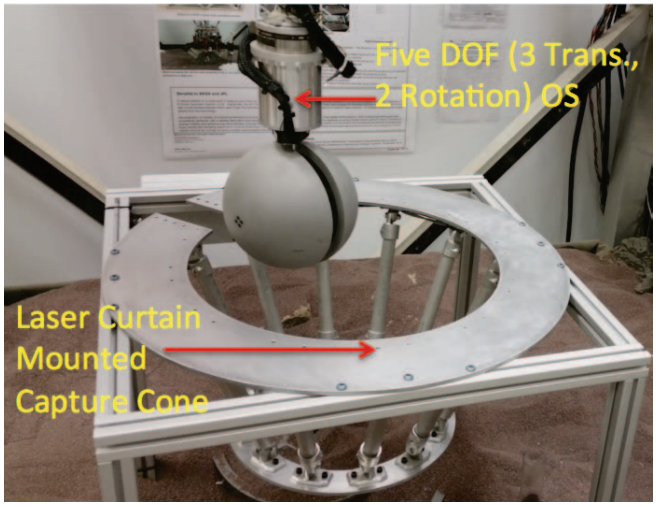


Figure 12. **The capture cone is shown here in the test bed with the OS simulator**

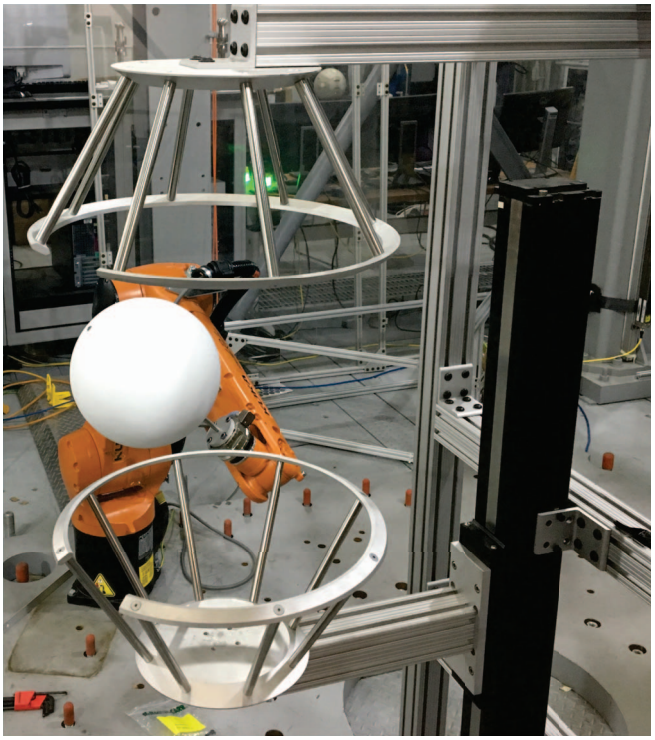


Figure 13. **The minimal capture volume test bed is shown here with the OS mounted on the KUKA arm and the two capture shells mounted on a linear slide**

In case the resulting uncertainty from the spacecraft control and relative OS tracking estimate is sufficiently small, a much smaller capture device can be used. Towards this, we built a test bed as shown in figure (13). The OS dynamics are handled using a 6dof KUKA arm. The capture device consists of two hemispherical shells, one of which is actuated using a linear-actuator while the other is fixed. The OS approaches the shells in a direction orthogonal to the linear-actuator. By precisely tracking the OS trajectory, the actuated shell moves in towards the fixed shell synchronously with the approaching OS and traps it first by *caging* it and then by closing in further until the two shells are in contact and the OS is brought to a rest. By *caging*, we mean that the shells are brought close to each other such that the OS, though still not in contact with the shells, is unable to escape due to the geometric cage formed by the fixed and approaching shells. Note that the maximum relative translational speed expected is 10 cm/s. This is sufficiently slow enough that the relative motion of the two shells is fairly slow (less than10 cm/s) and not a dynamic event like a mouse-trap.

A third test bed was developed that had a much shallower capture cone than the one discussed above, but allowed for a larger uncertainty in the OS relative position. As shown in figure (14), the capture device is a hemispherical shell with a flared opening. There are two actuated bars that close in on

the approaching OS and restrains it against the shell. Similar to the capture cone design, the rim of the shell has a laser curtain (not shown). When the OS breaks the laser curtain, the actuated bars are triggerd and then quickly close in on the OS to retain it. We tested the design successfully with misalignment of about 30% of the OS diameter between the OS approach vector and the shell center.

One of the biggest imports from these capture test bed studies has been the understanding of the nature of the contact dynamics. Unlike [3], where the relative velocities of 30cm/s were used, the contact dynamics are not dramatic and are fairly slow. This enables the accuracy of the capture devices to be relatively high and the actuation requirements are not very fast. As can be expected, the performance and success rates are much higher, almost certain, for the slower speeds of approach and lower errors in alignment of the OS with the center of the capture device.

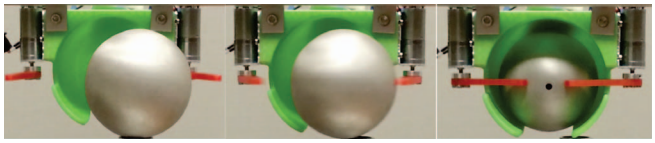


Figure 14. This is the third test bed which is based on a capture shell with flared opening and actuated fingers

5. AUTONOMY SOFTWARE

A fully distributed architecture has been created for the purposes of testing sample transfer technologies (Figure 15). Functionality is broken into stand-alone processes which communicate over TCP, UDP, or shared memory, depending on the size, frequency, and criticality of the communication. Modules are grouped into four main "layers".

Physical Layer—The physical layer consists of the M3tk [4] physics simulator, a high fidelity multibody and contact dynamics in-house simulation package, and firmware of sensor and actuator controllers.

Driver Layer—The driver layer consists of the MOT module, which brokers actuator commands from other modules, and perception modules, which push raw perception data into shared memory, estimate OS location from raw data, and perform sensor fusion. The interface between the rest of the modules and the physical layer is set up in a such a way that the other layers are agnostic to whether the data is being generated by a simulator or the physical devices. Further, the interface is developed in a manner that both hardware devices and simulator can be used concurrently. This allows a test bed to concurrently have hardware and software simulated elements.

Functional Layer— This layer represents the functionality of cohesive subsystems of the orbiter, such as the capture device, or of the test equipment, such as the gantry. These modules contain the autonomous behavior algorithms such as coordinated motion or force-feedback.

User Layer—Modules in this layer are used to launch other modules, monitor system state, and issue commands.

The distributed nature of the autonomy software enables rapid adaptation to different test beds. The *Driver Layer* software is written in a modular manner that it is agnostic to the *Functional Layer* software. It enables repeated and high reliability means of commanding device drivers across different test beds. The *Functional Layer* software allows the users to rapidly develop different test bed behaviors by implementing the autonomy algorithm and using standard API from the *Driver Layer*. This reduces development time and modularizes the custom implementations for each test bed. We are also working on developing an approach for graphically generating the autonomy behaviors and auto generating the executable software for the *Functional Layers*. This largely builds on an existing framework for specifying short-hand text representations of hierarchical state machines, as well as a method for auto-generating flight-capable real-time software. Our improvement to the state of the art in this area is the ability to auto generate the executable code rather than auto generating studs that the user had to fill in. By using the standard API provided by the *Driver Layer*, we enable the user to use SysML and graphically construct the autonomy behavior algorithm. This graphical approach uses embedded *Driver Layer* software calls and the overall graphical algorithm is translated directly and automatically into executable code that can be compiled and run without user having to write code. An accompanying paper [] discusses this work in more detail.

6. COMPUTER SIMULATIONS

As 0g hardware simulation in laboratory is a difficult prospect, we used computer simulations in conjunction with hardware elements to develop and test our concepts. We used M3tk, a high-fidelity simulation tool kit for general mechanical systems, as a simulation platform, and added a few new capabilities to it in order to meet project-specific requirements in modeling and simulation. One of the main requirements for simulating the concepts is the calculation speed. This is not only to finish a simulation within a reasonable amount of time, but also to achieve a real time simulation performance in case of running a cyber physical system where the hardware and the simulated parts must be interacting with each other in real time. Since the OS and the capturing and reorienting devices make metal-to-metal contacts with high stiffness, we use a tiny step size for time integration in order to capture the instantaneous contact phenomena as accurately as possible. Therefore we need to reduce the calculation time for the individual time step significantly. We address the issue with two methods i.e a system-of-systems approach and analytical collision detection.

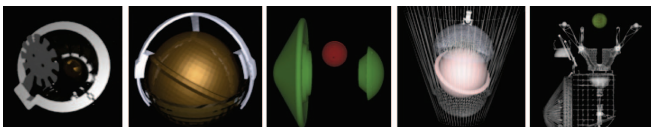


Figure 16. This shows a representative set of systems studied using our simulation capability.

Consider for example an end-to-end concept, shown rightmost in Figure (16). It consists of multiple subsystems: 1) a pair of blades for capturing and pushing the OS to the bottom shell, 2) the rollers at the end of the blades for reorienting the OS, 3) the capture cone ejection after the capture and reorientation, 4) the four-bar mechanism for covering the OS with the top shell, 5) the double pendulum mechanism for transferring to the EEV, 6) the EEV cover opening actuator, and 7) the EEV ejection system. Due to the many number of degrees of freedom, it takes a long time to solve the

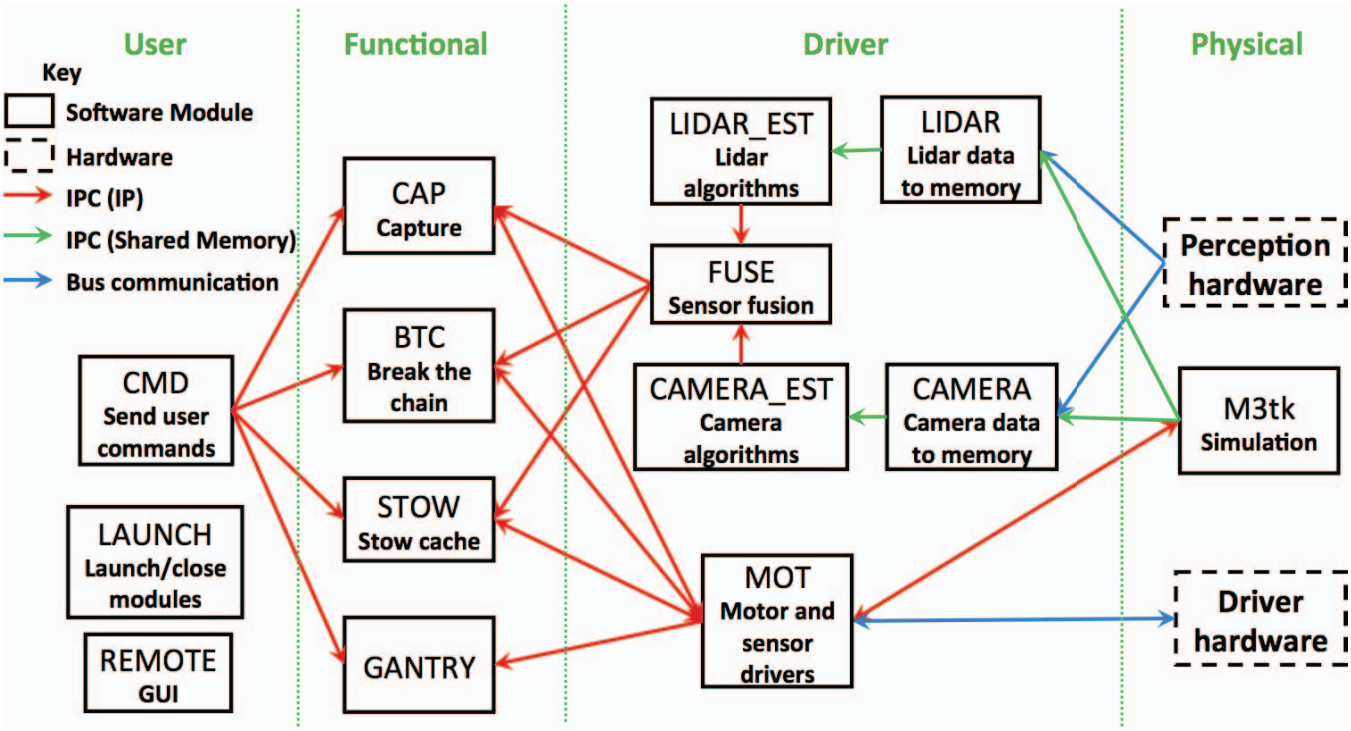


Figure 15. The MOSTT software architecture consists of distributed modules grouped into four "layers": the Physical layer, with hardware and simulations; the Driver layer, which processes raw data and brokers access; the Functional Layer, representing subsystems; and the User layer, for operation of the system.

equations of motion of the full system. Since each subsystem performs one at a time while the others are waiting until the current step is done, we have a chance to further speed up the simulation by treating each non-active subsystem as a merged body and considering only the degrees of freedom of the current active subsystem. It appears to be a simple simulation concept, but we must handle the state transition between different subsystems correctly, and also calculate the contacts in the full system and transfer the forces to the current subsystem. We were able to achieve about 5 times of speedup by introducing the subsystems in the simulation of the end-to-end system.

We developed a subsystem module for M3tk. It creates subsystems by reading a definition file written by the user and initializes their states with the full system state before starting the simulation. At every time step, only one subsystem is simulated at a time by solving the equations of motion to update its local state at the next time step. Also, at the same time, the full system is kept updated using the local state of the current subsystem because we need to use the full system to calculate external body forces such as contact forces and spring forces at every time step. The forces are then transferred back to the current subsystem in order to consider them in solving the dynamic equations. Subsystem triggering can be manually prescribed by the user or dictated by an external autonomy code, or even automatically done in M3tk, but with a limited performance, by using a heuristic method based on state monitoring. A state observer monitors the full system state, including joint velocities and their control targets, to detect a set of active joints that are expected to change continuously, and find a subsystem containing all of the active joints among the subsystem list the user defined. If there is no such subsystem in the list, the full system will be chosen. When we trigger a new subsystem, it does not know about the change in the system configuration by the previous subsystem, and thus, the inertia matrices of the bodies associated with the new subsystem are recalculated using the configuration of the full system. Note that, however, the inertia matrix update is conducted only once whenever a new subsystem is triggered, and it is not needed any more until next triggering happens.

Collision detection and identifying contact locations is a very time consuming task, especially when the model is exported from a CAD software and consists of discretized triangular meshes of complex shapes such as the concave bowl and the spherical Orbiting Sample (OS) with positive or negative features to facilitate capturing and reorienting it as appeared in Figure (1). Even with the state-of-the-art collision checking algorithms for 3D meshes, the simulation would suffer a significant slowdown when the step size for time marching is very small as in our cases. Decreasing the resolution of the meshes can give a speedup, but it sacrifices the simulation accuracy. Analytical formulation for detecting and locating contact points between geometry pairs can lead a significant speedup compared to the generic mesh-based algorithms, but is only available for a limited set of primitive pairs and not applicable to arbitrarily shaped geometries in general. M3tk originally provided analytical collision checking only for a limited set of primitive shapes such as box, sphere and capsule. We have extended the collision detection library to support modeling of more various kinds of shapes found in our concepts such as revolved surface, concave spherical shell, and cone etc. The primitives can be combined together to make a more complex shape as well. For example, cylinders can be subtracted from a sphere to model an OS with holes on the surface like a bowling ball. Figure (17) shows the collision model we used to simulate the overall concept. In the model the spherical OS has three

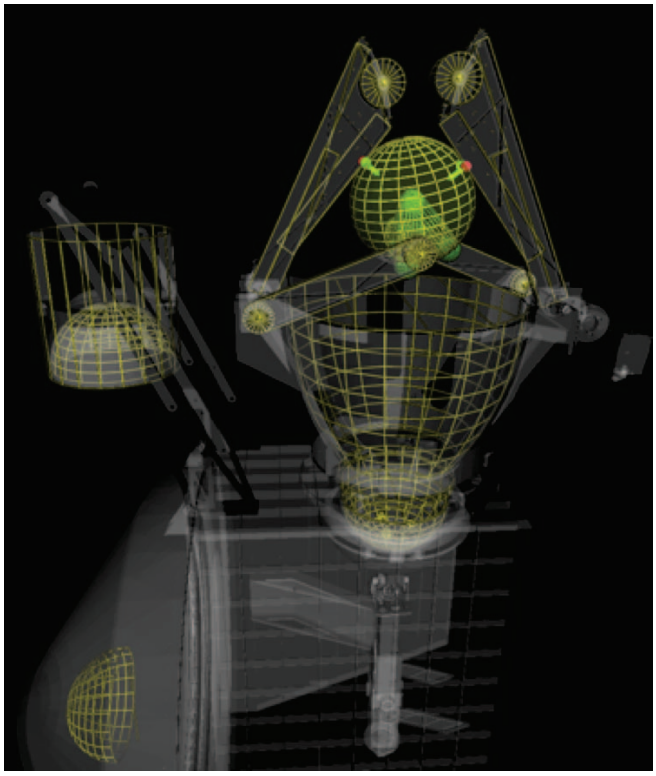


Figure 17. The primitive shapes (spheres, cylinders, cones etc.) based collision model of an end-to-end system is shown here

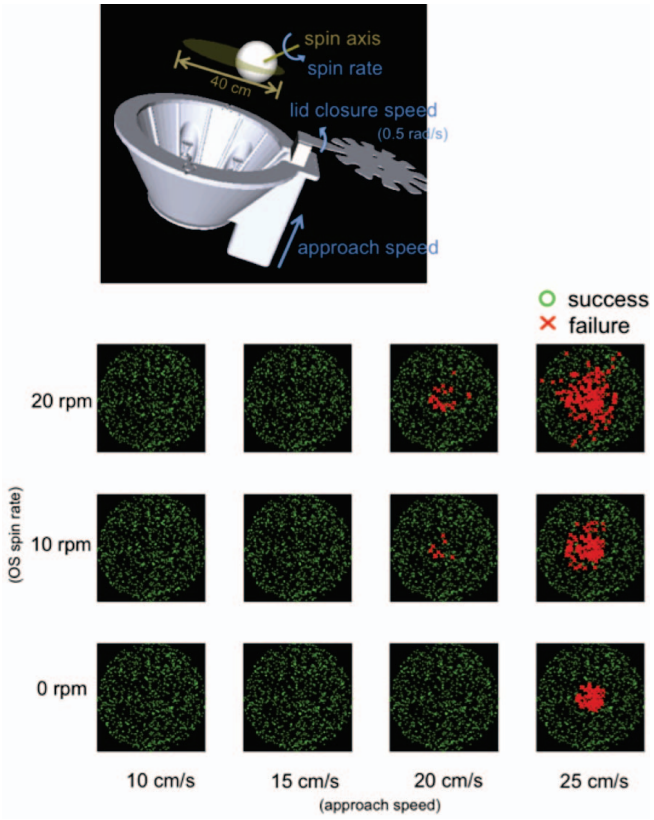


Figure 18. The results of a Monte-Carlo simulation of OS capture and the simulation parameters are shown here

cylindrical holes subtracted from the sphere so that it can be grabbed by three passive pins of the concave bottom shell, the capturing cone is modeled as a revolving surface of a curved line, and the blade is a composite of multiple boxes, cylinders, and a point cloud.

It is common practice to conduct a Monte-Carlo simulation, or running many simulations with slightly different settings or inputs, in order to account for the modeling uncertainties in the evaluation of a concept. We used an in-house python module to set up massive Monte-Carlo simulation instances and to run them in parallel on a cluster. Figure (18) shows an example of uncertainty analysis of the capture cone concept in order to study the effect of the OS spin rate and its approach speed toward the cone on the capture performance. The horizontal and vertical axes indicate the approach speeds and the OS spin rates we tested respectively, and for each velocity pair, we run a Monte-Carlo simulation to consider the uncertainty in the OS position on the plane perpendicular to the approach direction and the spin axis direction using a thousand random samples. We modeled the contact between the spherical OS and the cone and its lid assuming a Hertzian model with Coulomb friction. The material stiffness and the dynamic friction coefficient were set to 7.5E10 N/m and 0.3 respectively in the simulation. According to the simulation result, we expect that the concept would be able to capture the OS successfully if the OS speed is less than 10 cm/s for the operational scenarios chosen for simulation.

7. Cyber-Physical Simulations

To support the different electromechanical concepts that are being formulated for the orbital sample capture and ensuing manipulations, we are working towards realizing a fully functional cyber-physical simulation capability in our test beds. By cyber-physical, we mean simulation-in-the-loop hardware test beds where some elements are realized in hardware and some are simulated using detailed multibody and contact dynamics computer simulations. Toward this, we have made improvements in the capabilities of our M3tk simulation package as discussed above, particularly in the subsystem representation of the overall system and in contact detection. This has enabled a 5 times improvement in performance and ability to perform real-time simulations. We are also working on off-line, web-based visualization that may further improve the simulation performance. Similarly, we have architected the autonomy software in a manner that it can agonistically interface with hardware and software elements. From a software development perspective, the user has to set a few flags to identify what parts of the overall system are hardware and what parts are software simulated. The overall software development part, particularly in the *Functional Layer*, does not involve any code changes to accommodate or switch between hardware and software elements. As an example of this capability, figure (19) shows a snapshot of our capture cyber-physical test bed. The image on the left shows the computer simulated system. Here the OS and the capture cone are simulated with a lid. Collisions are checked between the OS, capture cone and the lid. The lid is modeled as a mechanical element that is rotated in place and then actuated downwards. This traps the OS and the OS bounces against the cone and the lid. The image to the right shows the hardware elements i.e. the physical OS and the physical capture cone. In this cyber-physical test bed, the collisions between the OS and the capture cone are obtained from the

hardware elements. The physical force-torque sensor records the interaction loads and passes it to the autonomy code. The computer simulation models the interaction between the OS and the simulated lid. The OS position is being tracked from the actual hardware element via the gantry drive motor encoders. The OS state are being fed back to the dynamics simulation. The simulation also has a simulated OS and the states from the hardware and simulation are tracked to check for error. We have recently added the ability to use the LiDAR in place of the gantry encoders to track the OS. Similarly, we also have computer simulated camera models that can also track the simulated OS and provide multiple sensor feedback for the Kalman filter based estimator.

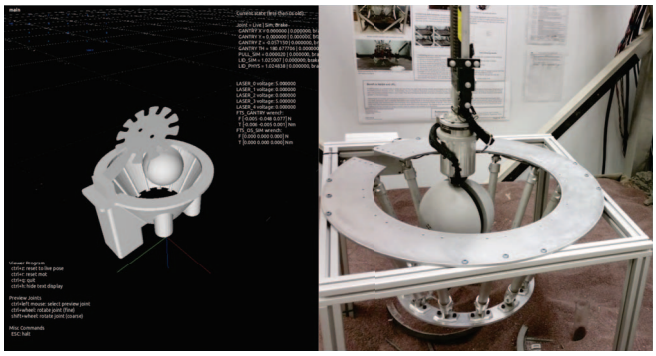


Figure 19. **This shows the juxtaposition of the computer model and the hardware element that are being controlled concurrently in the same autonomy scheme in real time. The computer model is a multibody dynamics model of the OS, capture cone and actuated lid while the hardware elements simulate the OS and its contacts with the hardware cone**

8. Future Work

As this is an ongoing work, there remain a number of items to be developed and tested as a part of this effort. We plan to conduct detailed testing, including field testing, of our perception sensors and algorithms for OS tracking. This would include sensitivity studies as well as use of stereo cameras, and LWIR imaging in conjunction with the visual tracking. Similarly, we propose to continue our hardware test element builds with a focus on the OS manipulation requirements after the capture. Particularly, there is a desire to orient the OS in a specific orientation after capture. Similarly, there are steps required to manipulate different mechanical elements for the BTC and planetary protection steps as well as transfer of the OS into an EEV or secondary containment vessel. On the autonomy side, we plan to mature our efforts in auto generating autonomy software using our graphical approach and apply this to the different test beds. Finally, we plan to continue to mature our cyber-physical test bed approach by combining different hardware and software elements and realize end-to-end systems.

9. Conclusions

In this paper we have presented a brief overview of some of the technologies and test bed elements we are developing towards supporting a potential Mars orbital sample capture and manipulation payload. This test bed is being developed with the Rendezvous and OS Capture System (ROCS) concept payload for the Next Mars Orbiter (NeMO) mission concept. We have developed detailed capabilities in mechanical capture and manipulation test articles, multibody dynamics simulations, autonomy software, perception based OS tracking capabilities, software auto generation capabilities and an interesting cyber-physical test bed capability. Our goal is to expand on these capabilities and realize a detailed test bed for an end-to-end system such as the one shown in figure (17) i.e. a capture system, subsequent manipulation stages, EEV or OS retention vessels, ejections etc. using this cyber-physical approach to enable end-to-end system testing capability.

Acknowledgments

The research described in this publication was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (NASA). Copyright 2016 California Institute of Technology. U.S. Government sponsorship acknowledged.

References

[1] Mattingly, R., Matousek, S., and Jordan, F., "Continuing Evolution of Mars Sample Return" *IEEE Aerospace Conference Proceedings*, Vol. 1, IEEE Publications, Piscataway, NJ, 2004, pp. 477-492.

[2] Mattingly, R., Matousek, S., and Jordan, F., "Mars Sample Return, Updated to a Groundbreaking Approach" *IEEE Aerospace Conference Proceedings*, Vol. 2, IEEE Publications, Piscataway, NJ, 2003, pp. 745-758.

[3] Kornfeld R. P., Parrish J. C., Sell S. "Mars Sample Return: Testing the Last Meter of Rendezvous and Sample Capture", *Journal of Spacecraft and Rockets*, Vol. 44, No. 3, May - June 2007.

[4] R. Mukherjee *et. al.*, "M3tk: A Robot Mobility and Manipulation Modeling Toolkit", *ASME 2014 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. vol. 7, 2014, DOI: 10.1115/DETC2014-34832.

[5] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[6] P. V. Hough, "Machine analysis of bubble chamber pictures," in *International conference on high energy accelerators and instrumentation*, vol. 73, 1959.

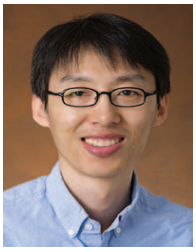
[7] I. Sobel, "An isotropic 3× 3 image gradient operator," *Machine Vision for three-demensional Sciences*, 1990.

[8] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

BIOGRAPHY

Dr. Rudranarayan Mukherjee *is a Research Technologist and Group Leader in the Robotics Modeling and Simulation group at the Robotics and Mobility Systems section at JPL. His primary role at JPL is to develop new technologies and find opportunities to apply them in flight missions.*

Neil Abcouwer *is a Robotics Electrical Engineer in the Robotic Mobility group at the Robotics and Mobility Systems section at JPL. He earned his BS and MS from Carnegie Mellon University. He is currently working on planning and mobility for the LEMUR3 robot, software and perception for the Mars Orbital Sample Transfer Technologies task, and avionics for the Athena rover upgrade.*



Dr. Junggon Kim *is a Research Technologist in the Robotics Modeling and Simulation group at the Robotics and Mobility Systems section at JPL. His research interests include motion planning and control using kinematics and dynamics. He earned his BS, MS, and PhD from Seoul National University, Korea. Before joining JPL, he was a project scientist at Carnegie Mellon University.*

Ryan Mccormick *is a Robotics Mechanical Engineer in the Robotic Vehicles and Manipulators group at the Robotics and Mobility Systems section at JPL. Before joining JPL in 2013, he was lead systems engineer to the Protector medium-sized unmanned ground vehicle and an autonomy kit for dismounted soldiers at HDT Robotics.*

Philip Bailey *is a Robotics Electrical Engineer in the Maritime & Aerial Perception Systems group at the Robotics and Mobility Systems section at JPL. He graduated from Canegie Mellon University with an integrated Masters and Bachelors focusing on robotics. Since coming to JPL, he has been working on motion planning for the Navys USV swarm and SEA Mob efforts.*

Peter Godart *began work as a Technologist in the Robotic Manipulation and Sampling Group at JPL in 2015 after graduating with SB degrees in mechanical and electrical engineering from MIT. His research interests include self-reconfigurable robotic limbs, autonomous robotic software architecture, underwater autonomy, and using aluminum as a fuel source.*