Reachability and Dexterity: Analysis and Applications for Space Robotics

Oliver Porges, Roberto Lampariello, Jordi Artigas, Armin Wedler, Christoph Borst, Máximo A. Roa
German Aerospace Center (DLR), 82234 Wessling, Germany, Email:firstname.lastname@dlr.de

***Abstract*—The utility of a mobile manipulator largely depends on its kinematic structure and mounting point on the robot body. The reachable workspace of the robot can be obtained offline and modeled as a discretized map called Reachability map. A Capability map is obtained by including some quality measure for the local dexterity of the manipulator, which helps to identify good and bad regions for manipulation. Once the maps are obtained based on forward or inverse kinematic methods, they can be used for numerous analysis tasks such as robot kinematics and workspace quality assessment, robot mounting point analysis or redundancy and failure analysis. This paper covers basic aspects of the Reachability and Capability map generation and storage, and shows particular applications of the maps for space robotics.**

I. Introduction

The ability of a robotic manipulator to grasp and manipulate objects depends on the kinematic structure of the arm, on its location in the physical space, on the relative location of the objects with respect to the arm, and on environmental restrictions. The reachable and dexterous workspace of the robot can be computed offline, which saves time for online queries of the map like grasp selection or path planning, or helps in the design process to obtain a correct performance for predefined manipulation tasks.

The manipulator workspace is defined as the set of all of the robot tool frame (TCP) poses that can be reached with some choice of joint angles. An efficient representation of the workspace can be generated by discretizing the task space $SE(3)$ in a hierarchical way: the reachable Cartesian workspace in $\mathbb{R}^3$ is divided in voxels with a predefined desired resolution, and each Cartesian voxel has an associated rotational grid that discretizes $SO(3)$. Using forward or inverse kinematics, each cell in the grid receives a binary value that indicates if it is reachable or not; this process generates the Reachability map [1], [2]. The inverse Reachability map can be created to look for the proper placement of a mobile robot according to a desired task [3]. The cells can also have an associated quality index that measures the dexterity of the robot when located in this position, thus creating a Capability map [1]. In the general case of a mobile manipulator, the robotic arm is mounted on a mobile platform. This base can also be included in the computation of the map in order to identify regions that are reachable and avoid self-collisions with the robot.

This paper summarizes the methods to generate the reachability map based on forward and inverse kinematics, and a hybrid method that combines advantages from both worlds, allowing the creation of such maps in a reasonable time while still guaranteeing complete coverage of the workspace. The efficient allocation of the map in memory is presented, which allows fast online queries and path planning. The prediction accuracy of the maps is discussed to show the influence of the map resolution on the quality of the obtained representations. The maps generated in this way represent ideal robot workspaces that only consider the kinematic design quality. In a real life scenario, the robotic manipulator is always mounted on a platform and embedded in an environment that imposes more restrictions on its movement. We show how to include these factors in the generation process to obtain the effective workspace for a particular application scenario. The potential of the Reachability and Capability maps is illustrated with different applications in the space domain, in particular: determining the optimal mounting point for a manipulator on a Lightweight Rover Unit (LRU), online grasp feasibility evaluation for a DEOS-TerraSAR-X deorbit mission, and a workspace analysis for the mission simulator DEOS-SIM.

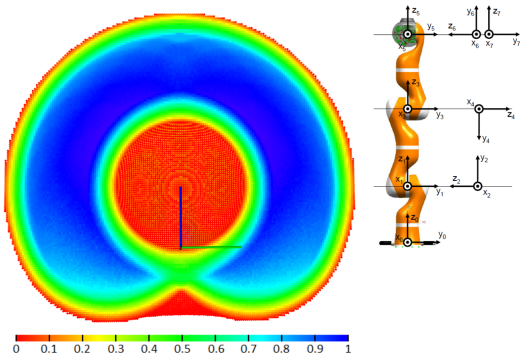


Fig. 1: Cross-section of the workspace of a KUKA-DLR LBR arm in form of a Capability map. The HSV color scale encodes the dexterity of each region in the space; blue indicates areas with higher dexterity of the manipulator.

II. Generation of Reachability Maps

The Reachability maps represent all possible positions and orientations that are reachable by the robot tool frame (TCP). The representation is obtained by hierarchically discretizing the robot workspace in all 6 degrees of freedom of the TCP (position and orientation): the reachable workspace in $\mathbb{R}^3$ is divided into equally sized cubes called voxels, with a predefined desired resolution, and each Cartesian voxel has associated rotational grids that discretize $SO(3)$, as illustrated in Fig. 2. The map is stored as a binary array where each bin represents the reachability of a small range of translations and rotations of the TCP.

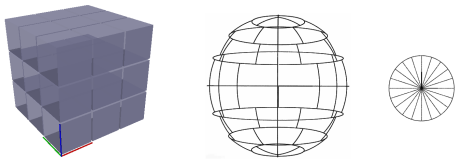


Fig. 2: Hierarchical discretization of $SE(3)$: Cartesian voxels define spatial locations (cube), each voxel has an associated grid for the approach directions (sphere), and each spherical region has an associated discretization for the roll directions (circle).

The bins are set to 1 if the range is considered reachable, and 0 if the robot cannot position and orient the end-effector within the bin range.

Different quality measures can be employed to further interpret the reachability data. Most commonly, a reachability index that indicates the percentage of bins that are reachable within a given voxel is used [1], which provides a single measure reflecting local dexterity. The set of reachability indices of the whole map creates a Capability map, as shown in Fig. 1. Further discussion on the reachability index and visualization of the data will be presented in Section III-C.

A. Generation Strategies

A straightforward method for the map generation uses a combination of forward kinematics (FK) and random joint sampling. Due to the low computational requirements of this approach, it is able to generate a large number of TCP samples in a short time. However, the random sampling does not guarantee that the whole workspace of the robot is covered in a finite time. Also, the larger the number of samples the most likely that the TCP samples start to fall in previously visited bins, which slows down the whole map generation process. The progress of the generation process can be assessed based on the ratio of new reachable bins within a number of generated samples. The map is ready to be used when this ratio falls below a predefined threshold.

Inverse kinematics (IK) can also be used for the map generation, as it guarantees a complete workspace exploration. Each bin in the map is represented by one TCP sample (commonly chosen at the center of the bin). If an IK solution (i.e. a combination of joint values) exists to reach this TCP, the bin is marked as reachable. However, while an FK check takes only some microseconds to obtain the TCP pose, an IK solver may take a longer time (several milliseconds) to determine whether a desired TCP pose is reachable or not. Due to the large number of samples that need to be generated, this method is rather slow.

A third generation method, the hybrid (HYB) method, combines the two previously mentioned strategies [4]. The process starts by using an FK approach based on random sampling of the joint positions. At the beginning, a large number of bins is correctly set to 1, employing only a small computation time per bin. The time required to determine new bins (or the ratio of newly discovered bins within a number of generated samples) is monitored, as it will increase (decrease, respectively) over time, and once it reaches a predefined threshold an IK-based generation is triggered to complete the map by verifying if there exists an IK solution for the bins that have not yet been set. The threshold depends on the particular IK solver, and it is related to how much time does the solver need to find a solution or determine that a pose is not reachable. In this way, the number of IK queries required to guarantee a complete exploration of the workspace is reduced, thus reducing the computational effort required for the map generation.

B. Prediction accuracy

The elementary evaluation for the quality of the map is its prediction accuracy. To test the accuracy, an arbitrary TCP frame is chosen and the map is queried for its reachability. Afterward, an IK solution is attempted, and there should be a match between the information coming from the map and the solver. A practical range of accuracies for this type of maps should be above 90-95% [4]. The remaining percentage (error) corresponds to a wrong identification of TCP reachability, which can be either false positives (FP) or false negatives (FN). The ratio of FP/FN queries changes according to the generation method. While the IK generation process produces roughly equal FP/FN queries, the FK generated maps significantly shift the ratio towards FP errors. This is due to the fact that with increasing numbers of random samples there is a higher chance that a TCP falls within a bin whose range of poses is mostly unreachable, yet still marked as fully reachable. For the hybrid method, the FP/FN ratio can be steered based on the switching criterion, and it will always be somewhere between the FP/FN levels of the IK and FK generation processes.

Naturally, the accuracy of the map depends also on the chosen discretization level. Let r be the voxel resolution (typically in cm), d_a the number of discretized approach directions and d_r the number of roll directions (Fig. 2). Table I presents prediction accuracies obtained for the case of the LBR (Fig. 1). Further details on performance and prediction accuracy for the three generation methods can be found in [4].

TABLE I: Prediction accuracy of IK, FK and Hybrid generation methods for the Reachability map of the LBR

r - d_a - d_r	IK	FK	HYB
5 - 50 - 10	95.8%	90.6%	93.7%
5 - 100 - 20	96.4%	92.7%	94.7%
5 - 200 - 30	97.5%	94.5%	95.2%

C. Environment model in generation process

The previously described generation processes only considers the kinematic structure of the robot itself, which is good enough to analyze kinematic limitations or redundancies in the actuation, or to assess the ability of a robot to perform some specified tasks. However, in reality the

robot is mounted on a base and interacts with a semi-structured environment. The generation above described is then enriched by including a model of the environment and using collision detections within the generation loop to obtain a workspace representation valid for particular use cases (i.e. valid TCP samples for which the robot is in collision are not considered valid and are not included in the map). Furthermore, the same approach should be applied during on-line queries of the maps.

A self-made collision checker was developed to be able to work directly with sensor data in a highly parallelizable way. The robot is modeled as a set of sphere and cylinder primitives. The robot parts are inscribed into the primitives, which can be enlarged if more conservative collision avoidance is required. The environment is modeled as a point set. These point sets can be obtained from CAD models by a user-defined discretization of the surfaces, which results in a controlled precision in the generation process. More importantly, as RGB-D sensors are a very common choice for robotic perception nowadays, with this approach the sensor data can be directly used later on to perform online collision checks.

At the beginning, a three dimensional binary tree is built for the point cloud of the model or scene [5]. The tree is built only once for a model, and on per frame basis for point clouds directly coming from the sensor. The binary tree helps to speed up radius queries (i.e. obtaining all points within a given position and radius without the necessity to check every point in the cloud). For a given robot configuration we perform a radius check for each primitive of the robotic chain to obtain the set of points possibly in collision with the primitive. The remaining points do not need to be checked at all. A point to primitive check is performed on each point and its corresponding primitive in vicinity. The squared distance from a sphere center or from the central axis of a cylinder is compared to the model parameters to detect possible collisions.

III. Implementation details

This section provides some details on the computational implementation of the Reachability maps, and presents the generation of a Capability map without requiring to explicitly store it.

Let $W_c \subset \mathbb{R}^6$ be the set of all reachable poses by an end-effector. Each pose within W_c can be described by a homogeneous transformation matrix as follows

$$TCP(t,R) = \begin{bmatrix} R_{1,1} & R_{1,2} & R_{1,3} & t_x \\ R_{2,1} & R_{2,2} & R_{2,3} & t_y \\ R_{3,1} & R_{3,2} & R_{3,3} & t_x \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

By using the decoupled approach for discretizing the space (Fig. 2), a set of unique mapping functions for each bin can be defined. The first mapping function assigns an index $i_c \in \mathbb{N}$ for the ordering of the space of translations.

$$f_c(t_x, t_y, t_z) \rightarrow i_c \tag{2}$$

Control parameters for the function $f_c(t_x, t_y, t_z)$ are the voxel resolution r, the coordinates of the origin (the lowest point the grid can contain), m_x, m_y, m_z, and the grid dimensions D_x, D_y, D_z. The orientation of the grid is assumed to be aligned with the robot base frame, and we further assume that the point in question lies inside the bounding box of the grid.

$$\begin{aligned} f_c(t_x, t_y, t_z) = \left\lfloor \frac{t_x - m_x}{r} \right\rfloor + \left\lfloor \frac{t_y - m_y}{r} \right\rfloor \times d_z + \\ \left\lfloor \frac{t_z - m_z}{r} \right\rfloor \times d_y \times d_z \end{aligned} \tag{3}$$

where $\lfloor x \rfloor$ denotes the floor operator that rounds to the highest integer value not larger than x. The mapping function in Eq. (3) discretizes the robot workspace into voxels. Each voxel has an associated discretization for the set of orientations. The pitch and yaw angles are coupled as an approach direction. The set of approach directions can be pictured as a spherical surface being inscribed into each voxel, where the spherical surface is divided by a grid into segments of equal areas. Each combination of pitch and yaw angle uniquely identifies a point on the spherical surface, which is used to assign an index with the following mapping function.

$$f_a(\alpha, \beta) \rightarrow i_a \tag{4}$$

Further details on the discretization of the spherical surface, including the explicit expression for $f_a(\alpha, \beta)$, can be found in [6]. Note that the spherical segments should ideally have equal areas, otherwise a non-desired weighting would be imposed on the discretization. Each cell of approach directions carries an additional grid that discretizes the remaining roll angle and completes the mapping of $SE(3)$.

$$\begin{aligned} f_r(\gamma) &\rightarrow i_r \\ f_r(\gamma) &= \left\lfloor \frac{\gamma}{r_r} \right\rfloor \end{aligned} \tag{5}$$

where r_r is the resolution of roll angle discretization. Indices i_c, i_a, i_r are directly used to address the memory structure of Reachability maps.

A. Memory structure

The map is stored in the memory as blocks of binary strings. We start by grouping all roll bins within one approach direction. These groups are stored one after another based on their index i_a. A complete set of approach directions corresponds to a voxel. Voxels are again stored based on their index i_c. The size of each voxel in the memory is determined by $d_r \times d_a$ bits, with d_r be the number of discretized roll directions and d_a the number of approach directions. Figure 3 illustrates the hierarchy and outline. The offset of n-th voxel in the memory is $v_o = n \times d_r \times d_a$ bits.

To guarantee higher accuracy, higher resolutions of the maps are desired, which have a large memory footprint.

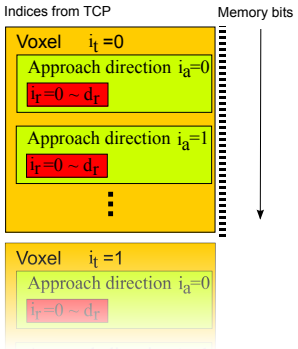


Fig. 3: Dense memory organization based on mapping indices i_c, i_a, i_r.

The operating system might not allow the allocation of one large continuous block of memory for the map (especially on 32-bit systems). For this reason, in the current implementation the memory is fragmented into fixed size blocks. These blocks are of arbitrary but constant size. They are allocated separately, which decreases the chance of allocation refusal by the operating system. To access a particular voxel, first the voxel offset v_o must be computed, and then it is mapped into the two dimensional array indices i_1 and i_2 as follows

$$\begin{aligned} i_1 &= \frac{v_o}{b_s} \\ i_2 &= v_o \mod b_s \end{aligned} \tag{6}$$

where b_s is the size of the blocks in bits.

B. Memory requirements

It is intuitive that a higher discretization resolution leads to higher prediction accuracy. The undesired effect of higher resolution is higher memory footprint. In particular, the memory requirement grows with the third power while increasing the voxel resolution. Table II shows the memory requirements for the KUKA-DLR LBR map at different resolution parameters.

TABLE II: Map memory requirements in MB

	r=7.5cm	r=5cm	r=2.5cm	r=1cm
d_a=50, d_r=10	0.69	2.3	18	258
d_a=100, d_r=20	2.9	9.7	73	1100
d_a=200, d_r=30	8.6	30	219	3301

C. Capability maps and visualization

Simple visualization of the 6D information in the Reachability map is not too meaningful. The Capability map, on the other hand, provides richer information. Each voxel is associated with its reachability index, described by the following equation

$$R_i = \frac{\sum_{a=1}^{d_a \times d_r} V_i(a)}{d_a \times d_r} \times 100 \tag{7}$$

where i is the index of i-th voxel and $V_i(a)$ is a particular binary reachability value stored in the map. Reachability index essentially describes how many of the discretized directions are reachable within each voxel, and quantifies its dexterity. The index is later used to encode the color for visualization. Using the voxel coordinates in space and the associated color, the robot workspace with the dexterity information can be visualized. For static applications, we commonly use a cross section of the workspace, as shown in Fig. 1. Thanks to the storage method of Reachability map, there is no need for storing an extra capability information. Using the common in-built CPU instruction for population count, each voxel's data is passed through and thus the reachability index is obtained on the fly.

IV. Applications

This section presents different applications of Reachability and Capability maps in several applications within the space domain, for ongoing projects developed at the DLR.

A. Deutsche Orbitale Servicing Mission (DEOS) Simulator

The DEOS Simulator, or DEOS-SIM, is a ground-based facility that serves as a test bed and experimental validation tool for on-orbit servicing scenarios. The system uses two KUKA KR-120 arms with force-torque sensors mounted at the end-effector. The readings of the sensors are used to generate appropriate motions for both KR-120 robots to simulate on-orbit conditions. In our case, two satellite mock ups are mounted at the end of each robot. One model represents a client (target) satellite, the other one represents the servicer, which is equipped with a smaller manipulator, a KUKA LBR. The goal of the servicer is to approach the tumbling target and grasp it with the LBR. Further details on the system can be found in [7].

The workspace of both kinematic chains, represented with a Capability map, is shown in Fig. 4. For this particular application, special interest is posed on the common workspace between the robots, as it is the useful area where mission simulations can be performed. One challenge for this implementation is the large workspace volume to be stored in memory. Thanks to the memory structure above described, we are able to store the workspace on a commodity workstation. The second challenge is the kinematic complexity, as the simulator employs two systems - a 6 DOF chain for the target and a 13 DOF chain for the servicer. We are able to obtain the reachability map within 12 hours of calculations using the FK based method at 2 cm voxel resolution, discretizing 500 orientations (50 approach directions and 10 roll angle ranges).

Let W_1 be the workspace of the KR-120 and W_2 be the workspace of the second KR-120 with LBR. We want to determine the common workspace volume W_c, with $W_c = W_1 \bigcap W_2$. The cross-sections of the common workspace W_c are shown in Fig. 5; the volume of W_c is approximately $23.28m^3$. This number is a parameter of the simulator and can change with a given mission scenario. For example, if a precise grasping point on the target is known, the transformation from the KR-120 TCP would

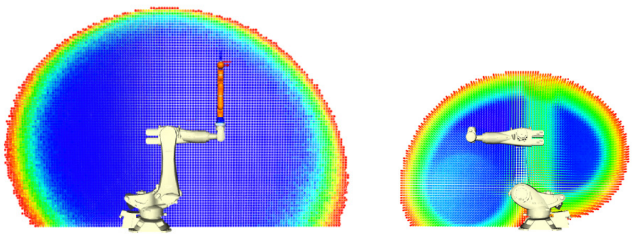


Fig. 4: Cross-sections of the capability maps for the servicer robot (left) and the client robot (right).

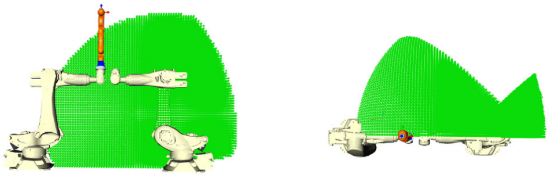


Fig. 5: Common workspace volume (cross-section) of the DEOS-SIM, side-view (left) and top-view (right).

increase its workspace volume. If a CAD model of the target is known, the collision detection would decrease the common volume. It is also possible to restrict the orientations of the servicer's TCP and obtain a mission specific common volume, along with the ideal volume in the workspace where the mission could be executed.

B. DEOS mission with TerraSAR-X

The DEOS mission is designed to perform an autonomous on-orbit robotic manipulation. The Capability map is specially useful to address the grasping and manipulation part of the mission. In the planned scenario, the servicer approaches the target on orbit. Once the relative distance and poses of the two satellites allow a feasible capture, the robot arm should reach out to the target, grasp it and perform a docking maneuver. A predefined structure dedicated for grasping is mounted on the target. This capture process cannot be planned ahead, but rather robust perception and planning algorithms have to run onboard of the servicer.

The fact that there is no possibility for precise preplanning implies that the servicer has to be able to robustly asses the feasibility of the planned motions. Due to the constant relative motion of the two satellite bodies, the planned motions are only valid within a certain small time frame. The feasibility assessment has to be carried out in a meaningful time frame for not loosing the window of opportunity. In this application the maps are used to obtain a set of feasible grasps on the target body, using either the CAD model or sensor depth data. Note that not every grasp is acceptable to perform the subsequent docking. There can be kinematic restrictions of the robot or collisions that would prevent the completion of the maneuver. After selecting a grasp, we can evaluate whether it is possible to perform the docking maneuver.

The effective Reachability map for the servicer system, including collision detection with the satellite's body, was

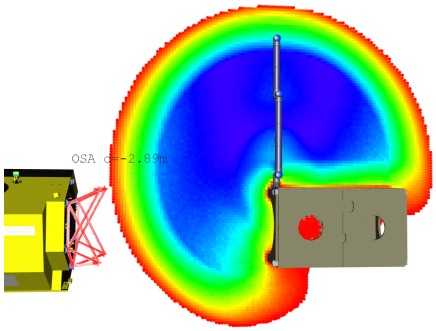


Fig. 6: DEOS workspace with TerraSAR-X in view, including the grasping structure (in red).

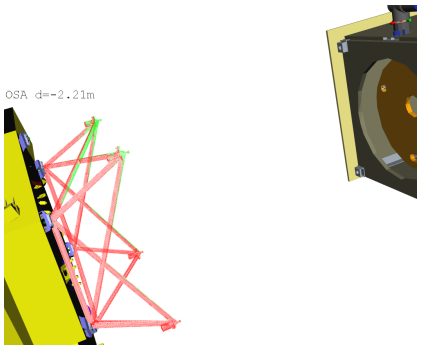


Fig. 7: TerraSAR-X approaching DEOS. Surfaces highlighted in green are reachable in this part with the current relative pose.

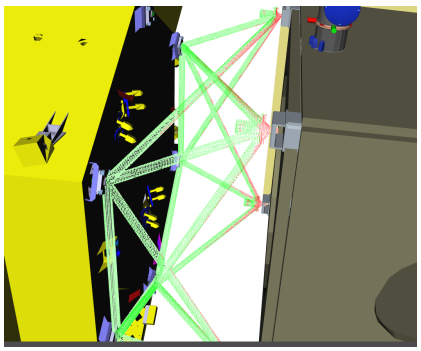


Fig. 8: Docking position of the two satellites. Surfaces highlighted in red are not reachable by the robot.

generated off-line. The CAD model of the target is preprocessed to discretize the model's surface into point-normal pairs. A point-normal pair is essentially a 5 DOF pose (position and approach direction), while the sixth parameter remains undefined. We formulate a query for the map for each point-normal in the model. The 5 parameters of position and approach direction are set, the remaining degree of freedom is the roll of the TCP. We check all the discretized roll bins for the given pair. The point is reachable if at least one discretized roll bin is set as reachable within the map. Such point-normal could be used as a grasping frame if it is possible to maintain the

grasp during the complete path execution. This is achieved by transforming the target model along a desired docking trajectory. Those points that are reachable at all times throughout the complete path can be considered as good grasping points.

Figure 6 shows the DEOS satellite with a cross-section of its workspace. The structure to be grasped is all highlighted in red because no points on the surface are currently reachable. As the two satellites get closer, as shown in Fig. 7, some parts of the surfaces are reachable by the robot (highlighted in green in the figure). The final docking situation is presented in Fig. 8. Note that not all the points on the surface are graspable in this situation, therefore it is crucial to choose the right grasping point at the starting configuration to avoid the need for re-grasping during the maneuver. Given a model of an end-effector, valid grasping points can be easily obtained on-line. A similar evaluation of path and grasp validity was presented in [8] and [9].

C. Lightweight Rover Unit (LRU)

Fig. 9: DLR's Lightweight Rover Unit - LRU.

The LRU[1], shown in Fig. 9, is primarily designed to explore Mars and the Earth's Moon in a semi autonomous way. To deal with the communication delays, LRU operates in a semi-autonomous mode where the operator only provides way points or goals, and the system follows them without the need of real-time teleoperation. This is possible thanks to several state of the art technologies developed at DLR, such as the space qualified drives and the stereo matching algorithm SGM [10].

LRU's design is currenlty being enhanced with a robotic manipulator to increase its usefulness. Different arm systems were considered to be mounted on the rover: The LBR[2] by KUKA, P-Rob 1R[3] arm by F&P Personal-Robotics, and Jaco[4] by Kinova robotics. Table III presents the workspace volumes for different dexterity levels without taking the rover into consideration. Full workspace is the complete volume that the end-effector can cover. We also compare the dexterous volumes which have at least 50% and 75% dexterity.

After choosing a sample mounting point, we recalculated the workspace volumes considering the effects of collisions with the rover body. Table IV presents the volumes of all reachable space and volumes of at least 50% and at least 75% dexterities.

[1]LRU - Lightweight Rover Unit, presented at ASTRA 2015.
[2]http://www.kuka-labs.com/en/service_robotics/lightweight_robotics/
[3]http://www.fp-robotics.com/en/prastandard/
[4]http://kinovarobotics.com/products/jaco-rehabilitation/

TABLE III: Workspace volumes [m^3] at different dexterities - without taking the LRU body into account

Arm / Dexterity	Full workspace	$\geq$ 50 %	$\geq$ 75 %
LBR	2.6673	1.8545	1.4518
F&P	1.5715	0.75254	0.2389
JACO	1.9505	1.8713	1.2746

TABLE IV: Workspace volumes [m^3] at different dexterities with robots mounted on the front plate of LRU

Arm / Dexterity	full workspace	$\geq$ 50 %	$\geq$ 75 %
LBR	2.4985	1.4979	0.93034
F&P	1.4047	0.5361	0.17357
JACO	1.752	1.5367	0.61783

Considering all design parameters (including the added weight on the rover), the Jaco arm was chosen. The next step involves the definition of a proper mounting point. The goal is to maximize the frontal reach of the robot, maximize the effective workspace volume, and allow for all predefined paths to be executed. As the rover can have storage areas on top or on the sides of its body, those areas need to be reachable when starting the motion at the frontal part of the rover. We also want to maximize the volume of the workspace where high dexterity is possible. The robot can be mounted in the front or in the back of the rover body. On both sides, an arbitrary tilt angle and a range of height from the ground can be chosen. The allowed degrees of freedom of the mounting point are discretized to calculate the effective workspace for each option. These different mounting angles are depicted in Fig. 10. Parameters such as workspace volume, high dexterity volume (above 75% reachability), and path feasibility around the storage areas are extracted from each map and compared.

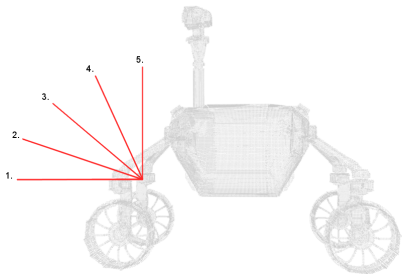
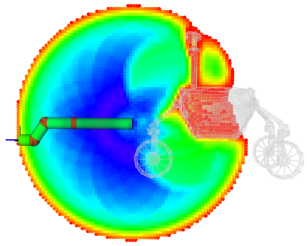
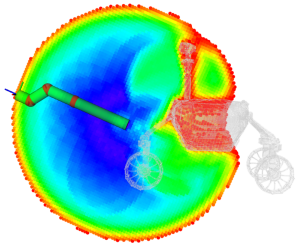


Fig. 10: Different mounting angles considered for a manipulator mounted on LRU.

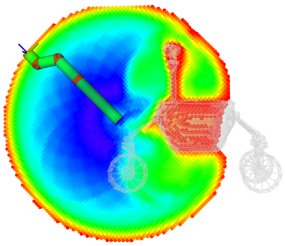
Cross-sections of the workspace maps for different mounting points are presented in Fig. 11. The collision model for Jaco arm is visualized in its default position. We can see how the dexterity changes on top of the robot body. After close examination, we determined for instance that the collisions with the pan-tilt head unit prevent the robot



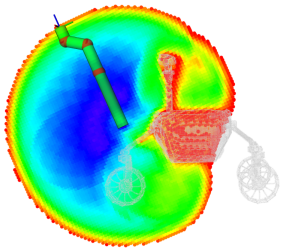
(a) Robot mounted at 0° angle



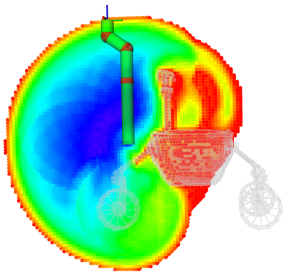
(b) Robot mounted at 22.5° angle



(c) Robot mounted at 45° angle



(d) Robot mounted at 67.5° angle



(e) Robot mounted at 90° angle

Fig. 11: Cross-sections of workspaces corresponding to the 5 mounting angles with the robot arm in its default position.

to reach the top of the body under a perpendicular TCP orientation. The maximum reach on a flat surface decreases with higher tilt angles. An ideal compromise is to mount the robot under approximately 50°.

V. Conclusion

This paper presented the Reachability and Capability maps as tools to describe the workspace of a robot and to understand and quantify different aspects of the inner structure of the workspaces. The algorithm consists of two stages, an offline-generation and online-application. The generation strategies and their tradeoffs were summarized. We described a data structure for the second stage to perform fast querying, thus allowing for application scenarios beyond pure workspace analysis. Three real life use cases were presented to demonstrate the applicability of the algorithm to analyze and compare the kinematic structures of robots, to make informed design decisions about robot mountings, and for online identification of good grasping locations with respect to the predefined task. Reachability and Capability maps are well suited for design, analysis and online tasks in both Earth and Space robotics. Current efforts are focused on increasing the prediction accuracy and minimizing the memory footprint to be able to run the maps with limited hardware resources.

References

[1] F. Zacharias and C. Borst and G. Hirzinger, *Capturing Robot Workspace Structure: Representing Robot Capabilities*, IEEE-RAS Int. Conf. Intelligent Robots and Systems - IROS 2007, pp.3229-3236.

[2] R. Diankov, *Automated Construction of Robotic Manipulation Programs*, CMU-RI-TR-10-29, Robotics Institute - Carnegie Mellon University, 2010.

[3] N. Vahrenkamp, T. Asfour and R. Dillmann, *Robot Placement based on Reachability Inversion* In IEEE Int. Conf. on Robotics and Automation - ICRA 2013, pp.1970-1975.

[4] O. Porges, T. Stouraitis, Ch. Borst and M.A. Roa, *Reachability and Capability Analysis for Manipulation Tasks*, ROBOT2013: First Iberian Robotics Conference, Series: Advances in Intelligent Systems and Computing, Vol.253, Ed. Springer, 2014, pp.703-718.

[5] M. Muja and D. Lowe, *Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration*, Int. Conf. on Computer Vision Theory and Application - VISSAPP 2009, pp.331-340.

[6] P. Leopardi, *A partition of the unit sphere into regions of equal area and small diameter*, Electronic Transactions on numerical analysis, 2006, Vol. 25, pp.309-327.

[7] J. Artigas, M. de Stefano, W. Rackl, B. Brunner, R. Lampariello, W. Bertleff, R. Burger, O. Porges, C. Borst, A. Albu-Schaeffer, *The DEOS-SIM: An On-ground Simulation Facility For On-Orbit Servicing Robotic Operations*, IEEE Int. Conf. on Robotics and Automation - ICRA 2015.

[8] F. Zacharias, *Knowledge Representations for Planning Manipulation Tasks*, Springer, 2012.

[9] F. Zacharias and C. Borst and G. Hirzinger, *Online Generation of Reachable Grasps for Dexterous Manipulation Using a Representation of the Reachable Workspace*, IEEE Int. Conf. Advanced Robotics 2009, pp.3229-3236.

[10] H. Hirschmueller, *Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information*, IEEE Conf. on Computer Vision and Pattern Recognition - CVPR 2005, pp.807-8014.