Robotic Demands on On-Board Data Processing

Thomas Bahls, Markus Bihler, Julian Klodmann, Freek Stulp, Gerhard Grundwald, and Alexander Beyer

***Abstract*—This paper gives an overview of the control and computing architectures of current lightweight robots and their demands on on board data processing units. On Earth, robots are an established technology in our daily life. They are getting more and more aware of their surroundings, which allows their use in many new domains. Collaborative workplaces where a robotic co-worker directly interacts with a human are already state of the art. Current research on machine learning and Artificial Intelligence (AI) in the robotic domain will introduce many new applications for robotic systems. These developments also have a great potential to be used in future space missions, for example on-orbit servicing of satellites or maintenance of space stations. However, this induces certain requirements on involved computing hardware that future on-board data processing units will also have to deal with.**

I. Introduction

The construction of the International Space Station (ISS) required a large number of Space Shuttle flights and Extravehicular Activities (EVAs) of astronauts. In contrast, it is expected that autonomous robots and robot assistants will play an essential role in making the assembly of the next generation of large orbital structures affordable [1], be it a more cost-effective, partly automatized orbiter around Earth, a cislunar station, or a manned spacecraft to Mars.

Space robotics covers a wide field of applications. It ranges from exploration tasks, such as robotic vehicle with measuring equipment [2], to servicing or maintenance tasks, such as on-orbit servicing or deorbiting of satellites [3], [4]. These tasks require different levels of autonomy, from telepresence scenarios [5] to semi-autonomous supporting task [4] and completely autonomous applications. Therefore, the system complexity of the robot varies from small modules developed for a dedicated task [6], to highly complex, versatile robotic systems, such as a rover for exploration [2], or a humanoid robot which performs tasks on a planetary surface in a supervised autonomy scenario [7].

The current Post-ISS scenarios and the continuously increasing commercial interests in GEO-servicing are strongly based on high-performance space robotic systems. The requirements for the task competences of space robots are comparable to those of advanced industrial applications on Earth. Space robots have to perform assembly, construction, maintenance, servicing, active debris removal, or other tasks that may have been unspecified at the time of the design of the robot. The robots should be able to work stand-alone, in teams, or to support astronauts during EVAs.

This latter scenario of robotic assistants has already found terrestrial applications, for instance collaborative light-weight robots in future manufacturing, where robots share a common workspace, or even physically interact to achieve a shared goal. For safety and also performance reasons, this requires high sensitivity and dexterity of the robot arms. For a large set of the above mentioned space scenarios, highly dexterous and sensitive robot arms are a missing key component, as this technology is only just becoming available for space applications (see Fig. 1).

all authors are with the Institute of Robotics and Mechatronics of the German Aerospace Center, Münchnerstr. 20, 82234 Wessling, Germany e-mail: thomas.bahls@dlr.de.

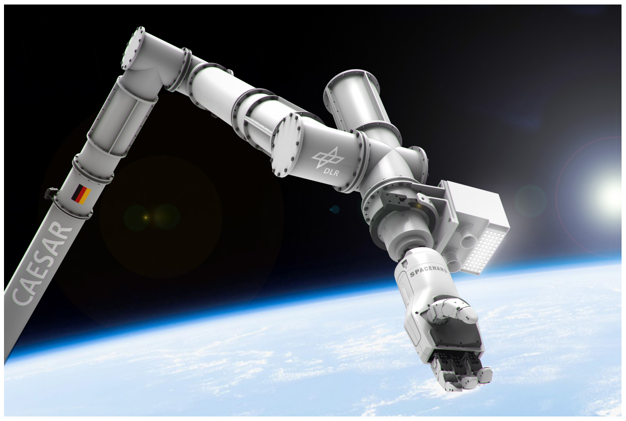

Fig. 1. Rendering of the CAESAR robot equipped with the DLR Spacehand [8].

In this paper, we consider different computational demands on the on-board data processing unit (OBDPU) that will arise from the next generation of space robots. The paper is structured along three broad classes of modules that require computational resources from the OBDPU: control demands, communication demands and application-specific demands. As an outlook, we briefly discuss the dramatic changes in computational requirements that will arise when methods from artificial intelligence – especially machine learning – will be run locally on the space robotics system.

II. Control Demands

With the development of the space-qualified robotic system CAESAR (Compliant Assistance and Exploration SpAce Robot), the Institute of Robotics and Mechatronics at DLR is continuing the work on on-orbit servicing that began with DEOS [9]. The seven degrees-of-freedom (DoF) robotic system is intended to be capable of catching satellites in Low Earth orbit (LEO) and Geostationary Earth orbit (GEO), even ones that are in tumbling, and/or non-cooperative states. The dexterity and sensitivity of CAESAR enables assembly, maintenance, and repair of satellites (see Fig. 1).

The key to CAESAR's high performance is the intelligent impedance and position control of its joints [10]. Each joint is a building block for setting up diverse robot kinematics

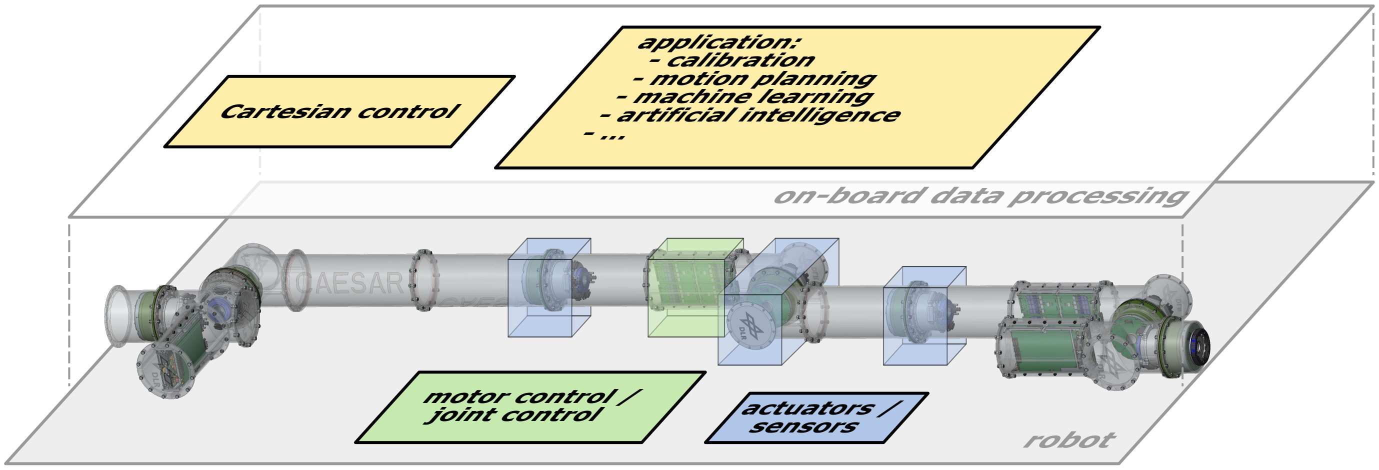

Fig. 2. Rendering of the CAESAR robot with a block diagram of modules and functionalities. The lower layer shows a transparent rendering of CAESAR with its main components visible. Joints with actuators and sensors are highlighted in blue. On the Joint Control Unit (JCU), highlighted in green, the joint and motor control tasks for two joints are computed. For a better visibility, only three joints and one JCU are highlighted. The upper layer depicts the functionalities that could be implemented on an OBDPU. Cartesian control requires information on the overall state of the robot and therefore it is typically implemented outside the robot. The same applies for applications. In other robotic systems, the joint control could also be part of the OBDPU.

depending on the different mission goals. The scalability of the robot is determined by the number of joints and the length of the links. CAESAR's seven DoF enables it to meet the dexterity and the kinematic redundancy requirements. Extending the joint control by Cartesian impedance controllers, the CAESAR arm can behave compliantly, while maintaining TCP position. The compliant behaviour is triggered if any part of the robot detects contact with the environment. With a compliant behaviour it is possible to detect unforeseen contacts with the environment.

A common approach to control such robotic systems comprises three cascaded control loops: motor control, joint control and Cartesian control [11], [12]. Motor control is typically performed in control loop frequencies of 20-100 kHz. It transforms desired motor positions or velocities into e.g. pulse width modulation (PWM) signals that are used to drive the motor. To reach the high control loop frequencies the algorithm is implemented directly on a processing unit (μC, DSP or FPGA) inside the robot. The control loop frequencies of the joint position, torque and impedance control is typically in a range of 0.5 kHz to 10 kHz. Depending on the hardware resources it is either implemented on processing units inside the robot or externally on the associated real-time host. Cartesian control with its control frequencies of 0.5 kHz to 3 kHz is typically implemented on the associated real-time host, as it requires information of the overall positions and torques of the robot. The application on top of the Cartesian control is also implemented in the real-time host. Figure 3 depicts this approach and Figure 2 shows the rough concept of CAESAR.

The relatively high control loop frequencies are the result of approximating the robot, a discrete time system, by a continuous time system. This is valid, if a sufficient sampling rate can be guaranteed [13]. The approximation is beneficial to use well known control approaches and stability conditions from the continuous time domain. However, this leads on the one hand to tough requirements for the communication systems especially regarding synchronization of all involved sensors and actuators. On the other hand it also demands for sufficient computing power as well as precise scheduling with low jitter of the real-time host. Furthermore to support scenarios with multiple robots and the integration of external sensors (e.g. cameras) a precise synchronization among each system has to be guaranteed to achieve high accuracy and low noise. This is also important with rising autonomy requirements (deep space missions) where all involved data has to be processed on an external host.

III. Communication Demands

From the communication point of view, in every control loop cycle of a robotic system the following steps have to be performed:

- A complete set of actual data representing the current state of the robot with all involved sensors is provided to the control unit.
- Out of the actual data set, the control unit computes a new set of desired data representing the desired state of the robot including all involved actuators.
- The desired data set is then provided to every actuator.

The communication infrastructure has to be designed to enable the processing of these three steps in the dedicated control frequency. In the afore presented approach, motor control frequencies from 20 - 100 kHz have to be guaranteed. The joint control is built on top. It assumes synchronized motor control units which are processed at the same point in time. Usually, a robotic system is a distributed system which does not have a common global clock. As a result, the communication must provide a common clock domain throughout the whole robotic system for synchronization.

The motor control is typically performed by processing units integrated in the robot and does not affect the determinism of the communication infrastructure. Depending on the design approach of the robot, joint control can be either performed in processing units integrated in the robot or in the real-time host.

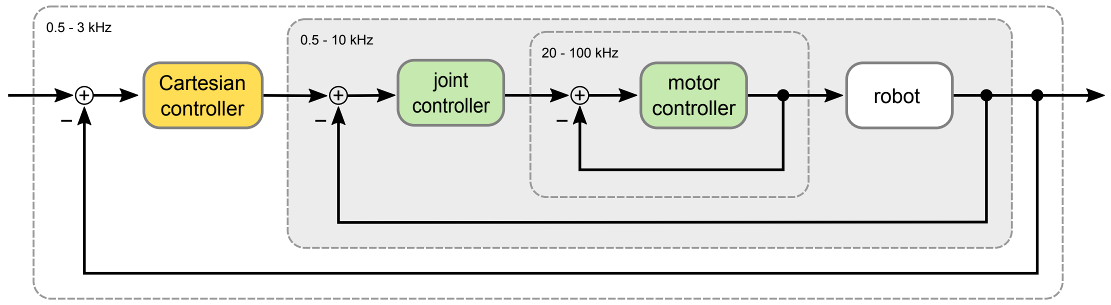

Fig. 3. Control cascades of a robot. In the case of CAESAR, the joint and motor controllers highlighted in green are implemented inside the robot while the Cartesian control highlighted in yellow is implemented outside the robot. Each joint has its own motor and joint controller. For a better visibility, exemplarily only one joint is depicted here.

Cartesian control is typically performed within the real-time host. All data, which are processed in the real-time host, leaves the highly deterministic communication infrastructure inside the robotic system and enters standard CPU architectures. This results in more jitter in the data exchange. A hardware triggered communication approach [14] and efficient packaging [15] improve this behaviour. Nevertheless, the remaining jitter has to be taken into account for the allocated computing resources of the real-time host to fulfill the control loop frequencies at any time.

Furthermore, allocation of extra computing resources for important features are necessary. Robotic systems often comprise a huge amount of sensors and actuators which have to exchange data with the real-time host (see Fig. 4). Therefore, a transport level implementation in the real-time host, which provides datagram as well as request/response communication, is strongly recommended. This enables a more comfortable usage from upper layers, a better integration into software concepts like publish-subscribe and services, and the integration of monitoring and debugging functionality. In addition, the transport level implementation in the real-time host should also consider an end-to-end data integrity check e.g. cyclic redundancy check (CRC) and encryption e.g. Advanced Encryption Standard (AES).

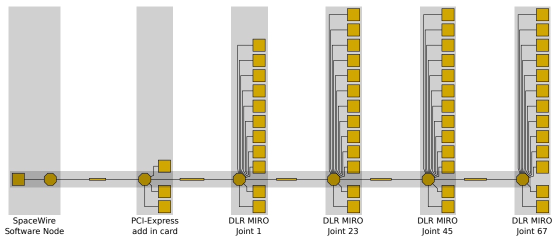

Fig. 4. Typical communication infrastructure of a 7 DOF light weight robot. Here the DLR MIRO robot which comprises 54 end point nodes and six routers distributed over six spatially separated electronics (represented as vertical bars) connected via five 1 Gb/s links (represented as horizontal bars).

IV. Application Demands

In addition to the previously introduced robotic control features, a real-time host also has to contain modules that implement applications on a higher level of abstraction. These applications provide commands to perform task specific actions. Examples are motion planing, long term calibration and grasp planing.

Motion planning is the process to determine how the robotic system moves from its current location to a new desired location. In case of a robotic arm, one example is the task to command a desired pose (position and orientation) with its end effector. For an exploration rover or a mobile robotic system in general an example would be to move to a certain location. In both cases, the motion planning is the task to calculate a trajectory to the desired pose or location without damaging the robotic system or its environment. This implies a certain level of awareness of the robots surroundings and of the robotic system itself. Depending on the application this awareness can be obtained by a combination of e.g. visual inputs and image processing, force sensors, joint position sensors, distance sensors and inertial measurement units. A detailed overview of the common motion planning approaches and the involved sensing is given by Siciliano and Kathib [16].

Modern high speed communication approaches in combination with almost unlimited storage capabilities enables the recording of huge data volumes. Hence, the system behaviour over time can be interpreted as an additional sensor source which enables a determination of the relations of different parameters among each other (see [17]) via statistical or artificial intelligence approaches. A possible application is life long calibration of a robotic system is possible. The aim is to keep an equal performance of the robot over its complete life span. The system behaviour over time is used to compensate or adjust the robot model uncertainty due to temperature drift, degrading of material, friction etc.

One of the main challenges in space robotics is to increase the autonomy of robotic systems. Here, we interpret autonomy to not only mean “independence from human intervention”, but also “the capability to decide between different courses of action based on the current goals”. This capability requires an understanding of the current state, the available actions and the current goals. When considering autonomous driving on Earth, it has become clear that achieving this capability requires a variety of different methods from the field of artificial intelligence, in particular machine learning. These methods will dramatically change the requirements for on-board data processing. For instance, novel hand-arm systems will require on-line grasp and motion planning. But state-of-the-art methods for these types of planning often have very high computational loads. And machine learning, a subfield of artificial intelligence, promises to provide solutions for the continual re-calibration of sensors, which is particularly acute in space applications due to the high wear and tear of robotic systems. But machine learning, due to its data-driven nature, will place much higher demands on the local storage and processing of data. In the future, such demands will have to be met if planning algorithms and machine learning are to run on-board, which may be essential to achieving the desired levels of autonomy.

V. Conclusion

To perform a complete robotic scenario by an OBDPU on a space craft as the equivalent to a terrestrial real-time host, the demands can be subdivided into the basic demands to run a robotic system and application specific demands. The first are closely linked to the robotic hardware. The OBDPU must be able to achieve the desired timing requirement such as control loop frequencies, synchronisation or jitter. Furthermore, it has to be able to provide standard communication concepts such as request/response and datagram to upper layers. Data integrity and data encryption have to be considered as well.

The second are dependent on the application and the operation environment. Hence, the demands are rising with the complexity through involved data sources, autonomy aspects, cooperative tasks, safety issues, etc. In addition, since robotic is closely linked to current computing trends such as machine learning and artificial intelligence, a certain amount of computing resources should be reserved to be prepared for future approaches.

References

[1] Oliver Romberg, Dominik Quantius, Claudia Philpot, Stephan Siegfried Jahnke, Wolfgang Seboldt, Hansjörg Dittus, Sven Baerwalde, Hans Schlegel, Mike Gold, George Zamka, Rodrigo da Costa, Ingo Retat, Rainer Wohlgemuth, and Max Lange. The orbital-hub: Low cost platform for human spaceflight after iss. In *67th International Astronautical Congress (IAC)*, September 2016.

[2] Armin Wedler, Martina Wilde, Josef Reill, Martin J. Schuster, Mallikarjuna Vayugundla, Sebastian G. Brunner, Kirstin Bussmann, Andreas Dömel, Martin Drauschke, Heinrich Gmeiner, Hannah Lehner, Peter Lehner, Marcus Gerhard Müller, Wolfgang Stürzl, Rudolph Triebel, Bernhard Vodermayer, Anko Börner, Rainer Krenn, Armin Dammann, Uwe-Carsten Fiebig, Emanuel Staudinger, Frank Wenzköfer, Sascha Flögel, Stefan Sommer, Tamim Asfour, Michael Flad, Sören Hohmann, Martin Brandauer, and Alin Olimpiu Albu-Schäffer. From single autonomous robots toward cooperative robotic interactions for future planetary exploration missions. In *In 69th IAC International Astronautical Congress*, Bremen, Germany, October 2018.

[3] D Reintsema, J Thaeter, A Rathke, W Naumann, P Rank, and J Sommer. Deos–the german robotics approach to secure and de-orbit malfunctioned satellites from low earth orbits. In *Proceedings of the i-SAIRAS*, pages 244–251. Japan Aerospace Exploration Agency (JAXA) Sapporo, Japan, 2010.

[4] P Rank, Q Mühlbauer, W Naumann, and K Landzettel. The deos automation and robotics payload. In *Symp. on Advanced Space Technologies in Robotics and Automation, ASTRA, the Netherlands*, 2011.

[5] C Riecke, J Artigas, R Balachandran, R Bayer, A Beyer, B Brunner, H Buchner, T Gumpert, R Gruber, F Hacker, et al. Kontur-2 mission: the dlr force feedback joystick for space telemanipulation from the iss. In *The International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*, 2016.

[6] Josef Reill, Hans-Jürgen Sedlmayr, Sebastian Kuß, Philipp Neugebauer, Maximilian Maier, Andreas Gibbesch, Bernd Schäfer, and Alin Albu-Schäffer. Development of a mobility drive unit for low gravity planetary body exploration. In *12th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA)*, ESA/ESTEC, Noordwijk, Netherlands, May 2013.

[7] Neal Y Lii, Daniel Leidner, André Schiele, Peter Birkenkampf, Ralph Bayer, Benedikt Pleintinger, Andreas Meissner, and Andreas Balzer. Simulating an extraterrestrial environment for robotic space exploration: The meteron supvis-justin telerobotic experiment and the solex proving ground. In *Proc. 13th Symp. Adv. Space Technol. Robot. Automat.*, 2015.

[8] Maxime Chalon, Maximilian Maier, Wieland Bertleff, Alexander Beyer, Ralf Bayer, Werner Friedl, Philipp Neugebauer, Thomas Obermeier, Hans-Juergen Sedlmayr, Nikolaus Seitz, et al. Spacehand: A multifingered robotic hand for space. In *Proc. 14th ESA Workshop Adv. Space Technol. Robot. Autom.(ASTRA)*, pages 1–8, 2015.

[9] D. Reintsema, J. Thaeter, A. Rahtke, W. Naumann, P. Rank, and J. Sommer. “deos the german robotics approach to secure and deorbit malfunctioned satellites from low earth orbits. In *Proc. i-SAIRAS Conf.*, 2010.

[10] Alexander Beyer, Gerhard Grunwald, Martin Heumos, Manfred Schedl, Ralph Bayer, Wieland Bertleff, Bernhard Brunner, Robert Burger, Jörg Butterfaß, Robin Gruber, Thomas Gumpert, Franz Hacker, Erich Krämer, Maximilian Maier, Sascha Moser, Josef Reill, Máximo Alejandro Roa Garzon, Hans-Jürgen Sedlmayr, Nikolaus Seitz, Martin Stelzer, Andreas Stemmer, Gabriel Tubio Manteiga, Tilman Wimmer, Markus Grebenstein, Christian Ott, and Alin Olimpiu Albu-Schäffer. Caesar: Space robotics technology for assembly, maintenance, and repair. In *69th International Astronautical Congress (IAC)*, October 2018.

[11] Gerd Hirzinger, Norbert Sporer, A Albu-Schaffer, M Hahnle, Rainer Krenn, Antonio Pascucci, and Markus Schedl. Dlr's torque-controlled light weight robot iii-are we reaching the technological limits now? In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 2, pages 1710–1716. IEEE, 2002.

[12] Ulrich Hagn, Mathias Nickl, Stefan Jörg, Georg Passig, Thomas Bahls, Alexander Nothhelfer, Franz Hacker, Luc Le-Tien, Alin Albu-Schäffer, Rainer Konietschke, et al. The dlr miro: a versatile lightweight robot for surgical applications. *Industrial Robot: An International Journal*, 35(4):324–336, 2008.

[13] J. Lunze. *Regelungstechnik 2*. Number Bd. 2 in Springer-Lehrbuch. Springer, 2008.

[14] Stefan Jörg, Mathias Nickl, Alexander Nothhelfer, Thomas Bahls, and Gerd Hirzinger. The computing and communication architecture of the dlr hand arm system. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1055–1062. IEEE, 2011.

[15] Thomas Bahls. Entwicklung einer latenz-und bandbreiten-optimierten Bridge zur transparenten Anbindung von FPGAs and Standard-CPUs, 2008. University of applied sciences Munich, master thesis.

[16] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. Springer, 2008.

[17] T. Bahls, M. Bihler, and Tarassenko S. Spacewire meets big data - realtime data mining. In *2018 International SpaceWire Conference (SpaceWire)*, pages 139–142, May 2018.