sciendo

DOI 10.2478/ama-2021-0023

acta mechanica et automatica, vol.15 no.3 (2021)

HAND GUIDING A VIRTUAL ROBOT USING A FORCE SENSOR

Radovan GREGOR*, Andrej BABINEC*, František DUCHOŇ*, Michal DOBIŠ*

*Faculty of Electrical Engineering and Information Technology, Institute of Robotics and Cybernetics, Slovak University of Technology (STU) in Bratislava, Ilkovičova 3, SK-812 19, Bratislava, Slovakia

radkog@gmail.com; andrej.babinec@stuba.sk; frantisek.duchon@stuba.sk; michal.dobis@stuba.sk

received 14 April 2021, revised 18 July 2021, accepted 20 July 2021

Abstract: The research behind this paper arose out of a need to use an open-source system that enables hand guiding of the robot effector using a force sensor. The paper deals with some existing solutions, including the solution based on the open-source framework Robot Operating System (ROS), in which the built-in motion planner MoveIt is used. The proposed concept of a hand-guiding system utilizes the output of the force–torque sensor mounted at the robot effector to obtain the desired motion, which is thereafter used for planning consequential motion trajectories. Some advantages and disadvantages of the built-in planner are discussed, and then the custom motion planning solution is proposed to overcome the identified drawbacks. Our planning algorithm uses polynomial interpolation and is suitable for continuous replanning of the consequential motion trajectories, which is necessary because the output from the sensor changes due to the hand action during robot motion. The resulting system is verified using a virtual robot in the ROS environment, which acts on the real Optoforce force–torque sensor HEX-70-CE-2000N. Furthermore, the workspace and the motion of the robot are restricted to a greater extent to achieve more realistic simulation.

Keywords: hand guidance, force–torque sensor, motion planning, industrial robot, Robot Operating System

1. INTRODUCTION

According to the International Federation for Robotics (IFR; https://ifr.org/free-downloads/, 2020), in 2018, >400,000 new industrial robots were installed in the manufacturing sector. The annual increase in this parameter was 6%, which—at the end of 2018—increased the "population" of industrial robots to 2,439,543 pieces. The dominant segments are the automotive, electronics, metalworking, plastics, and food industries. Such growth is also due to new technologies that make it easier to work with robots for nonprofessional users. One of these technologies is power–torque control, which allows robots to provide “touch” and enables users to work with the robot at the level of manual guidance. This technology makes it possible to create robotic applications more efficiently and in a significantly accelerated manner.

The following standards apply in particular to industrial robots' work in cooperation with humans:

a) International Organization for Standardization (ISO) 10218 – 1 Robots and robotic devices - Safety requirements for industrial robots. Part 1 Robots.
b) ISO 10218 – 2 Robots and robotic devices - Safety requirements for industrial robots. Part 2 Robot systems and integration and technical standard.
c) ISO technical standard (TS) 15066 – Robots and robotic devices - Collaborative robots.

The first standard describes the requirements and guidelines for safe design, protective measures, and information for industrial robot usage. It describes the basic risks associated with robots and describes the elimination or adequate reduction of risks. The second standard deals mainly with the integration and installation of industrial robots into cells or production lines. This standard is a continuation of the first one; it is much more detailed, and it specifies the requirements for individual elements of the robotic system. The technical standard TS 15066 describes the safety requirements for the integration of collaborative robotic applications. Collaborative robotic systems differ from standard industrial ones in that the operator may work close to the robotic system. Simultaneously, the robot does not limit its activities in any way and can thus come into physical contact with the human operator.

Human–robot collaboration can have the following four modes:

- Safety-rated monitored stop;
- Hand guiding;
- Speed and separation monitoring; and
- Power and force limiting.

This paper focuses on the area of hand guiding of robots. Our research is unique in that it provides an open-source solution based on a relatively inexpensive force–torque (FT) sensor. The goal of our research is to point out the possibilities of using low-cost reusable solutions when hand guiding the robot effector, which is a better solution than using a teach pendant in the process of teaching robotic paths. An example of such use can be tracking of an uneven path with many via points when using a teach pendant, which could be more time-consuming. In this research, we have also focused on the fluency and safety of the robot’s motion during the hand-guiding process so that it is more ergonomic for an operator. The algorithms were tested on the virtual robot model KUKA KR16. It is an industrial robot that is not intended for use outside a cage, but during the teaching process, its maximum speed is limited to the safe value; thus, hand guiding is allowed.

The paper is organized as follows. Section 2 demonstrates the current state of the art and defines the uniqueness of our solution. Section 3 describes the hardware and software components used

in this work and the methodology. Section 4 shows how the data from the FT sensor were processed. Sections 5 and 6 show the visualization of the whole process and the configuration of the solution, respectively. Section 7 describes the implementation of our solution since the existing path planning system is unsuitable for our purpose. Section 8 presents the results of the experiments and the verification of the whole solution.

2. RELATED WORKS

According to Matheson et al. (2019), in the field of human–robot collaboration (HRC), robots with technologies corresponding to power- and force-limiting operating modes are primarily used, but the number of applications of standard industrial robots using hand-guiding and speed-monitoring modes is also increasing. Manual guidance is a representative functionality of cooperative robots, which allows unqualified users to interact with and program robots more intuitively than while using a teach pendant (Safeea at al., 2017). Such functionality is made possible mainly by sensing the forces and torques.

Loske and Biesenbach (2014) describe a typical implementation of an FT sensor integrated for hand guiding into an industrial robot's control. The article includes a SI-130-10 sensor integrated with a KR 16-2 robot. Functional applications, such as hand guiding, hand-guiding motion learning, and sensor-guided motion, have been demonstrated. The results show the need to configure such a sensor, especially with reference to gravity. The article also demonstrates the much higher efficiency of the robot's learning for the required movements and points in space.

Reyes-Uquillas and Hsiao (2021) analyze manual guidance control using an FT sensor. They propose an adaptive admittance law that can adjust the parameters of the FT sensor to modify robot compliance in critical areas of the workspace, such as near and on-configuration singularities, joint limits, and workspace limits, for a smooth and safe operation. This demonstrates the further usefulness of FT sensors in manual guidance and advanced control methods.

In another paper (Safeea et al., 2019), the authors deal with a robot's precise manual guidance at the level of the end effector when navigating around obstacles. An FT sensor is used to manually guide the robot, while the compliant robot's body is used to detect contact with an obstacle. Thus, the solution proves the possibility of using the FT sensor for manual guidance even with more-expensive compliant robots, such as the KUKA iiwa. Another method of precise manual guidance with an expensive compliant robot is demonstrated by Safeea et al. (2017). In this case, FT sensing is used directly from the joints of the robot itself.

Massa et al. (2015) analyze the various possibilities of manual robot guidance. For example, various admittance/impedance control methods or the use of different learning methods for this type of guidance are given. The authors show that a key aspect of robot programming is for the operator to consider the robot as a reduced-mass tool. The result is an ability to intuitively learn how to use a robot, even for an operator who has no experience with robotics. The article also excellently demonstrates the possibility of using an FT sensor for this purpose. Simultaneously, the authors focus on gravity compensation during manual guidance of the robot using an FT sensor.

Jamone et al. (2014) describe the use of a six-axis FT sensor on a humanoid robotic arm. In this case, the FT sensor was used to detect the internal and external forces acting on the robotic arm. The article also presents the control strategies for a robotic arm without knowledge of the environment. A similar system is also addressed by Jo et al. (2013), whereby a robotic arm is used to grasp objects. It is equipped with FT sensors with gravity compensation, and the control is designed based on torque-to-velocity transformation using FT sensors.

Peng et al. (2021) have presented a robotic assembly methodology for the manufacture of large, segmented composite structures. A robot wrist-mounted FT sensor enables gentle but secure panel pickup and placement. Human-assisted path planning ensures reliable collision-free motion of a robot with a large load in a tight space. This work demonstrates the versatility of sensor-guided robotic assembly operations in a complex end-to-end task using the open-source Robot Operating System (ROS) software framework.

A very interesting way of using an FT sensor is presented by Zhao et al. (2020). In their work, the peg-in-hole approach for a six-parallel-legged robot is proposed. An FT sensor is used to plan the trajectory in real time to mate the peg and the hole.

Lee et al. (2016) and Zhang et al. (2019) deal with hand-guiding learning as a more suitable alternative for the learning of industrial robots rather than learning with a teach pendant. However, these authors do not suggest manual guidance using an FT sensor but rather a sensorless guiding method based on torque control on the robotic arm's motors. This method depends on the robot's exact dynamic model, which is not always available (especially in the case of commercial manufacturers). Moreover, commercial manufacturers of robotic systems may not allow new robot control methods to be implemented. Current FT sensors can extend such functionality on a commercially available robot and quickly detect forces and torques. At the same time, the price of FT sensors has dropped dramatically recently. An example of such an innovative sensor is given in a previous work (Noh et al., 2016). An example of a similar sensor, which is used in our research, is the Optoforce sensor (https://www.coden.org/blog/finalist-optoforce-hungary-sensors-for-the-internet-of-things/, 2015). Optoforce is a young innovative spin-off company that has already been bought by OnRobot (https://www.crunchbase.com/organization/optoforce, 2020), with renaming of the original sensor as HEX FT sensor (https://onrobot.com/en/products/hex-6-axis-force-torque-sensor, 2021).

FT sensing has great applications not only in hand-guiding of robots but also in other controls, as demonstrated by Safeea et al. (2019), Jamone et al., (2014), and Jo et al. (2013). Furthermore, in the Strategic Research Agenda for Robotics in Europe 2014–2020 (https://www.eu-robotics.net/cms/upload/topic_groups/SRA2020_SPARC.pdf, 2020), emphasis is placed on sensor-based safety systems to enhance human–robot interaction, multiple degrees of freedom, tactile feedback, and physical human–robot interaction. The technology that would lead to these goals is described in this strategic document as the compliant control of a complex mechanical structure with visual and tactile perception of human interaction to produce intuitive physical interfaces. This is exactly what hand guiding of a robot using a force sensor fulfills.

FT sensors can be used effectively in various applications with industrial robots, but they are also often used as safety sensors. This is demonstrated, for example, in the article by González et al. (2021). The authors demonstrate an advanced teleoperation and control system for industrial robots in order to assist the human operator to perform tasks such as sanding, deburring, finishing, grinding, polishing, and so on. An FT sensor mounted on the robot

end effector is utilized as a safety indicator to stop the motion of the robot system when the sensor measurements are abnormally large.

As mentioned earlier, in our research, we use the Optoforce sensor, specifically, the HEX-70-CE-2000N version. The sensor is composed of a system of three three-axis hemispherical sensors. The sensors are inserted between two cylindrical aluminum plates with a diameter of 70 mm. The range of measured forces varies depending on the measurement axis. The highest measurable value of the force is in the *Fz* axis. The manufacturer specifies its value at 2000 N. At certain pressures in the *Fz* axis, the silicone forms a protective layer around the sensors to prevent damage. As a result of the flexible properties of silicone, the sensors can withstand a maximum possible overload in one axis of up to 200% of the specified nominal value without permanent damage. When obtaining data from the sensor, we used a reading frequency of 100 Hz, while the maximum possible frequency is 1 kHz. This sensor is characterized by high resolution, the possibility of use with various robots, and a relatively low price. There is also a package for this sensor in the ROS environment (https://github.com/shadow-robot/optoforce, 2020), used in the presented research.

The advantages of our solution compared to the solutions described earlier are the following:

- The solution is developed using several open-source modules in ROS. These include the following standard packages:
  - MoveIt - motion planning framework designed primarily for trajectory planning and inverse kinematics calculation based on selected algorithms;
  - Gazebo - a simulation environment that allows simulating the dynamics of a robot;
  - ros_control and ros_controllers - set of software packages for the use and development of robot controllers; and
  - RViz - a three-dimensional visualization tool.
- Considering that open-source modules are used, the solution is easily extensible and scalable, which is not the case with the closed proprietary systems described earlier.
- The solution can be easily and quickly implemented on various robots – from standard industrial robotic arms through power-sensitive robots to mobile manipulators. This advantage is also made possible by the use of ROS.
- Based on the force acting on the sensor, a planning process is periodically started, which generates the currently required trajectory for the robot effector. In contrast, rescheduling and exchanging the trajectory takes place smoothly in terms of position and speed without interruption of movement.
- Trajectory planning is constantly monitored so that a plan that would cause a fundamental change in arm configuration (e.g., from the elbow up to elbow down) is not triggered. These changes can occur either near a configuration where a joint is approaching a constraint or close to singular configurations. Such a plan is assessed as risky, and its execution is suspended.

3. COMPONENTS AND METHODOLOGY

When implementing the solution of manual robot guidance based on an FT sensor, the ROS framework was used, enabling the design and verification of robot control algorithms without the need for physical connection of the robot. The advantage of this solution is that the same implementation can be redirected to a real robot's control. The ROS acts as an interconnection element between the systems.

The control algorithms were verified on the model of the industrial manipulator KUKA KR16, which consists of models of individual parts of the robot provided in stereolithography (STL) file format, and their kinematics is defined using the unified robot description format (URDF). These models and the URDF are available in ROS-Industrial, which contains experimental packages for KUKA manipulators, including the kuka_kr16_support package (https://www.crunchbase.com/organization/optoforce, 2020). Other important components are MoveIt, ros_control, Gazebo, and RViz; the necessary configuration files for these components are described later in the “Configuration” section.

To integrate the Optoforce sensor into the system, an optoforce package was used, which contains an ROS node for operating the FT sensor (https://github.com/shadow-robot/opto-force, 2020). As shown in Section 4, the sensor measurements suffer from noise, drift, and bias; therefore, the acquired data need to be preprocessed. The force and torque vectors are then computed from the processed sensor measurements.

To perform a robot motion corresponding to the applied force, motion planners have been utilized. The input to the motion planner is usually the current and final pose of the robot effector. Therefore, we need to determine the position vector $\boldsymbol{p}$ of the desired point in the robot workspace to which the robot effector will be moved. Assuming that the position vector is expressed in the effector’s coordinate system, it can be computed by a simple proportional reduction of the force vector $\boldsymbol{F}$ using the proportional constant k_{p} according to Eq. (1).

$$\boldsymbol{p} = k_p \boldsymbol{F} \tag{1}$$

Such a point can be regarded only as an auxiliary point, and there is no need to completely achieve this position during robot motion. However, the magnitude of the position vector may be used to determine the desired speed of the robot effector during hand guidance; in this context, the safety of the operator must be ensured by limiting the speed. The value of the proportional constant $k_{\mathrm{p}} = 0.002\ \mathrm{m/N}$ was determined experimentally.

Initially, in our research, the planning process was performed using the MoveIt framework, which is described in Section 7.1; however, we find this approach to be unsuitable. Therefore, the new planning method described in Section 7.2 was implemented.

4. SENSOR DATA PREPROCESSING

To interconnect the Optoforce sensor with the ROS platform, installing the Optoforce driver was necessary (https://github.com/shadow-robot/optoforce, 2020). The driver uses the pySerial library to retrieve data from the sensor via the Universal Serial Bus. The data frequency is set to 100 Hz by default. During initialization, the configuration parameters are read, including coefficients for conversion from device-specific units to newtons and newton-meters. When started, the driver publishes messages to a topic called optoforce_0. The messages are in the ROS format called geometry_msgs/ WrenchStamped. The data message contains the force and torque vectors' coordinates acting on the sensor in the sensor coordinate system, which is aligned with the tool0 system. Tool0 is defined in the URDF description of the robot as the coordinate system of the effector.

Before the sensor data can be used, the sensor must be calibrated. A series of experiments were performed to determine the

DOI 10.2478/ama-2021-0023

calibration parameters. As an example, the calibration of the sensor in the *Z*-axis is described. The sensor was placed such that the gravitational force acts only in the opposite direction of its axis. At zero static load, a zero-output value is assumed, and at a load of 1 kg, an output value of –9.806 N is assumed. Table 1 shows the average output values for the indicated number of samples.

Tab. 1. Evaluation of the uncalibrated sensor output

Number of samples	Used weight, kg	Theoretical output, N	Maximum acquired output, N	Minimum acquired output, N	Average acquired output, N	Average deviation, N
6,578	0	0	2.1226	0.7075	1.4258	1.4258
7,898	1	–9.806	–4.0094	–5.8962	–5.2917	3.0885

Figure 1 shows that the sensor values fluctuate significantly under static load. Another measurement lasting longer at zero load revealed that the identified deviation is accompanied by drift, as evidenced in Fig. 2. Based on the measurements, the functionality was approximated by a first-order polynomial, as follows:

$$f(x) = 0.26924x + 2.923 \tag{2}$$

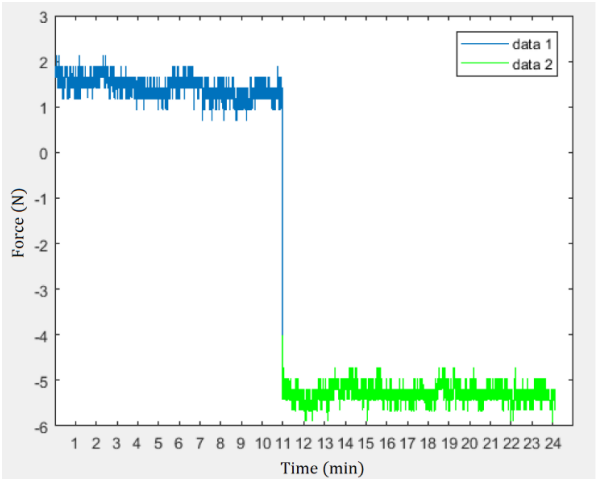

Fig. 1. Fluctuation of the sensor outputs at zero and nonzero static loads

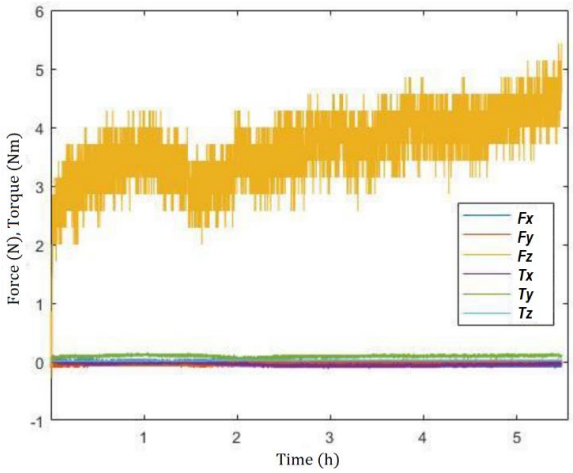

Fig. 2. Experiment revealing the drift of the sensor output value

Tab. 2. Measuring the transfer characteristic

Measurement number	Used weight, kg	Gravitational force, N	Acquired output, N	Measurement error	
				Absolute, N	Relative, %
1	1.00	9.806	28.412	18.606	189.70
2	1.97	19.317	60.51	41.193	213.20
3	4.38	42.95	123.12	80.17	186.00
4	10.09	98.942	315.6	216.658	218.9
5	20.40	200.042	638.15	438.108	219.00

Based on the other measurements listed in Table 2, the transfer characteristic between the actual and measured data was interpolated, as in Eq. (3).

$$f(x) = 3.218x - 5.633 \tag{3}$$

Using this relationship, the average relative error dropped to 5.65%. Calibration had to be done for the other axes similarly.

5. VISUALIZATION OF PROCESS

The KUKA KR16 robot model was visualized in the RViz environment. The magnitude and direction of the force acting on the Optoforce sensor, located on the sixth axis of the robot, are used to determine the new point to which the robot is moving. This point's position vector expressed in the effector coordinate system is represented by a red arrow starting from the center of the tool0 coordinate system (Fig. 3). The visualization of the vector is dynamic, depending on the actual force applied to the sensor.

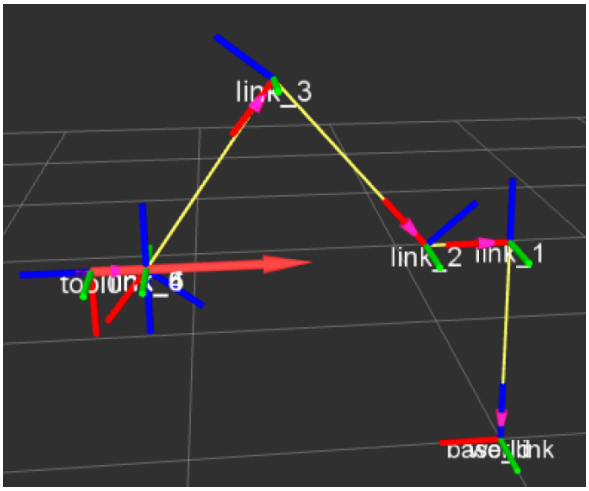

Fig. 3. Visualization of the KUKA KR16 robot model in the RViz environment with the position vector of the point to which the robot is moving

For detailed visualization of the data from Optoforce, the tool from the ROS Qt (RQt) framework that monitors the topic of optoforce_0, was used. The output is the time-dependent data shown

in Fig. 4. In the RViz environment, the results are reconstructed from the detected force and torque components and visualized with an arrow as the force and torque vectors using the WrenchStamped module (Fig. 5).

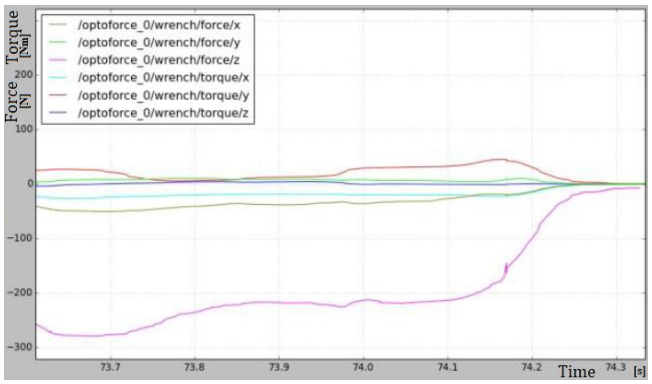

Fig. 4. Visualization of data from Optoforce by monitoring the topic of optoforce_0

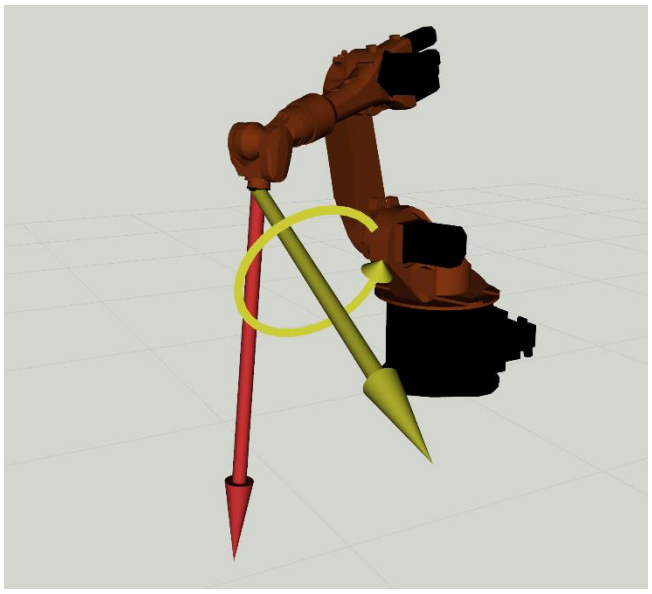

Fig. 5. Force and torque vectors reconstructed from the detected force and torque components

6. CONFIGURATION OF SOLUTION

The kinematic model of the KUKA KR16 industrial manipulator is described in the URDF format. This model needs to be extended with the manipulator's dynamic properties, such as weights, centers of gravity, and moments of inertia of all moving parts of the robot. These values are approximately calculated in the computer-aided design (CAD) software based on the geometry and material properties. They are then exported to the URDF format via the SolidWorks to URDF (SW2URDF) plugin.

An example of the modified parameters for link 1:

```
~/catkin_ws/src/kuka/robots/kuka.urdf
...
2 <inertial>
3 <origin rpy="0 0 0" xyz="0 0 0"/>
4 <mass value="21.654"/>
// default value: 2 kg
5 <inertia ixx="0.5521" ixy="0.046477"
ixz="0.28985" iyy="0.81346"
iyz="0.050687" izz="0.71343"/>
// default values: 0.01 Kg*m^-2
6 </inertial>
...
```

For the correct simulation in Gazebo, it is important to add information about the URDF structure into the <transmission> part. This part defines the relationship between drives and joints. Since the MoveIt tool is also used for trajectory planning, in addition to the URDF, it is necessary to create a Semantic Robot Definition Format (SRDF) file that describes the mutual collision positions of individual robot arms, virtual joints, planning groups, fixed joints, and other parameters related to the movement and interaction of the robot with the environment. This file is created by the MoveIt Setup Assistant, which provides a graphical interface for configuration. The default algorithm from the Kinematics and Dynamics Library (KDL) is used to calculate the inverse kinematics, and the Open Motion Planning Library (OMPL) is used for trajectory planning as it includes the RRTConnect algorithm, which is selected as a planner for MoveIt. Later, in our proposed planning method, a linear interpolation is used between the individual transition points located on the calculated planned trajectory.

To link MoveIt and Gazebo, a JointTrajectory controller is used. This controller is defined in a configuration file in Yet Another Markup Language (YAML) format, where the proportional–integral–derivative (PID) parameters of the positioner and the frequency of publishing the joint status are defined. In Fig. 6, an rqt_graph of the connection between the controller and Gazebo can be seen.

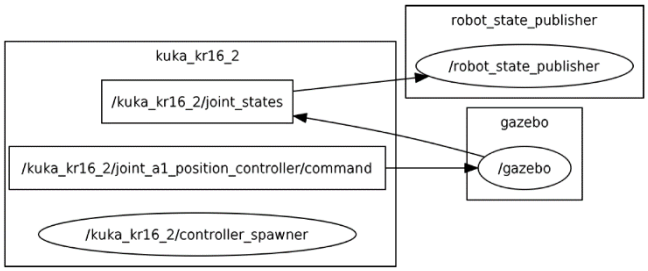

Fig. 6. Rqt_graph showing the connection between the controller and Gazebo

7. IMPLEMENTATION

The article compares two approaches. The first is the use of all available packages and the planning of robot movements using the MoveIt planner. This approach's disadvantage is that the robot always stops after the trajectory is completed to start planning a new movement. The second approach assumes the implementation of an own planning algorithm, which eliminates this problem, and the planned trajectory is modified while the robot is moving.

7.1. Planning with MoveIt

After configuring the system, a node was created with an algorithm to control the manipulator's movement. The controller accesses the data from the sensor by retrieving messages from the topic optoforce_0. Based on this, it is possible to create a result-

DOI 10.2478/ama-2021-0023

ant vector of the applied force, which defines the desired direction of movement of the robot's endpoint. Subsequently, we can determine the position vector of the point to which the robot effector is moved. Since the planner works with the coordinates relative to the robot's base coordinate system, the coordinates of the position vector of the target position are transformed into the base coordinate system. The orientation of the effector is changed similarly. Based on the resulting torque vector acting on the sensor, the effector's required orientation is also calculated as a proportional reduction of the torque vector. Using the moveit::planning_ interface::MoveGroup class and its getCurrentPose() method to get the current position of the effector, the effector's current position is acquired. The request to plan a new trajectory must contain the target position and orientation (ROS message geometry_msg::Pose), which is sent to MoveIt using the plan() method, and the output is the new trajectory. The trajectory is formed by a series of waypoints that have all the joint positions explicitly specified. The resulting trajectory is then sent to a controller that controls the position of the joints.

7.2. Planning with the proposed algorithm

The disadvantage of the move_group node is that further planning and control are possible only after the current task is completed. Therefore, the move_group node waits until it receives information from the server about reaching the goal. This results in the robot stopping at each point determined by the computed position vector. By using the FollowJointTrajectory motion service, it is possible to obtain smoother movements. This control system implements a method for replacing a part of the planned trajectory. It allows rescheduling of the trajectory and launching of a new plan at a specific time. When we send a new message of the type trajectory_msgs/JointTrajectory, the controller joins the trajectories at time t^* specified in the new message's header. At t^*, the controller switches from the previous trajectory to follow the new trajectory as depicted in Fig. 7.

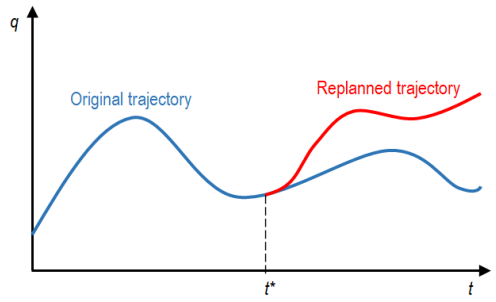

Fig. 7. Principle of the joint trajectory switching

It is important to ensure that the positions and velocities at time t^* are the same for both trajectories. For this purpose, the call to the query_state service is used, which provides the information about the current trajectory's future positions and velocities at the requested time (Joint Trajectory Action Controller, Official webpage ROS Documentation, 2020).

The planning algorithm can then use different interpolation techniques to generate a joint path from the current position and velocity at time t^* to an auxiliary position given by the position vector $\boldsymbol{p}$ calculated using Eq. (1). In our case, we used interpolation based on a cubic polynomial. It is then needed to specify the constraint conditions, which are the initial and final positions of the joints $\boldsymbol{q}(t_0)$ and $\boldsymbol{q}(t_f)$, as well as the initial and final velocities in the joints $\dot{\boldsymbol{q}}(t_0)$ and $\dot{\boldsymbol{q}}(t_f)$, to ensure that the positions and velocities at time t^* are the same for the current and the replacing trajectories. The acceleration will not be limited in any way, so we anticipate that there will be a step change in the acceleration when switching from one trajectory to another. However, the controller in the robot's joints can handle this step change without major problems. Based on the limiting conditions and the requirement of the cubic shape of the polynomial, we can compile a set of equations for one joint, as in Eq. (4), which can be rewritten into a matrix shape, as in Eq. (5) [24]. The polynomials' coefficients can then be calculated according to the equations in Eq. (6), which were derived from Eq. (5). The trajectory for each joint of the manipulator is formulated likewise.

$$\begin{aligned} q(t_0) &= a_0 + a_1 t_0 + a_2 {t_0}^2 + a_3 {t_0}^3 \\ \dot{q}(t_0) &= a_1 + 2a_2 t_0 + 3a_3 {t_0}^2 \\ q(t_f) &= a_0 + a_1 t_f + a_2 {t_f}^2 + a_3 {t_f}^3 \\ \dot{q}(t_f) &= a_1 + 2a_2 t_f + 3a_3 {t_f}^2 \end{aligned} \tag{4}$$

$$\begin{bmatrix} q(t_0) \\ \dot{q}(t_0) \\ q(t_f) \\ \dot{q}(t_f) \end{bmatrix} = \begin{bmatrix} 1 & t_0 & {t_0}^2 & {t_0}^3 \\ 0 & 1 & 2t_0 & 3{t_0}^2 \\ 1 & t_f & {t_f}^2 & {t_f}^3 \\ 0 & 1 & 2t_f & 3{t_f}^2 \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \tag{5}$$

$$a_0 = q_0 \qquad a_2 = \frac{3(q_f - q_0) - (2\dot{q}_0 + \dot{q}_f)(t_f - t_0)}{(t_f - t_0)^2}$$

$$a_1 = \dot{q}_0 \qquad a_3 = \frac{-2(q_f - q_0) + (\dot{q}_0 + \dot{q}_f)(t_f - t_0)}{(t_f - t_0)^3} \tag{6}$$

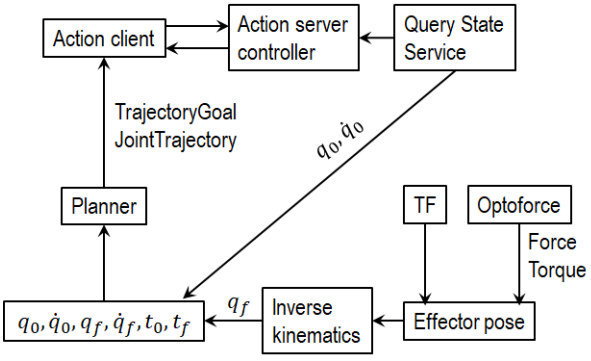

Fig. 8. Block diagram of the proposed system with continuous replanning procedure reacting on sensor readings

We obtained the initial position and velocity of the planned trajectory from the query_state service. The initial time of the planned trajectory will therefore be $t_0 = t^*$. For safety reasons, the final speed $\dot{q}(t_f)$ is set to $0\ \mathrm{ms}^{-1}$. However, during the continuous planning process, the robot will not stop because the next new planned trajectory will overwrite the current trajectory after time t^*, and the zero speed requirement is delayed. The joints' final positions must be calculated using inverse kinematics from the position vector $\boldsymbol{p}$ of the target position obtained from the data from the force sensor, as shown in Fig. 8. To calculate the inverse

kinematics in MoveIt, there is a dedicated setFromIK() method in the RobotState class.

7.3. Increasing the security of the solution

Since the planned trajectories are created from linear sections to ensure their unambiguity, there may be sudden undesirable changes in the arm's configuration (see Fig. 10) near the singularity or joint limit. Therefore, it is necessary to limit the resulting solution. If the arm configuration reaches such a position that one of the joints approaches the limit of its range (Fig. 9), then the new planned position in the original configuration would not be achievable. It would be necessary to rotate one or more joints into another overall arm configuration. From the point of view of manual guidance operation, this movement would be dangerous. Therefore, a new node has been created to monitor and limit potentially dangerous configuration changes.

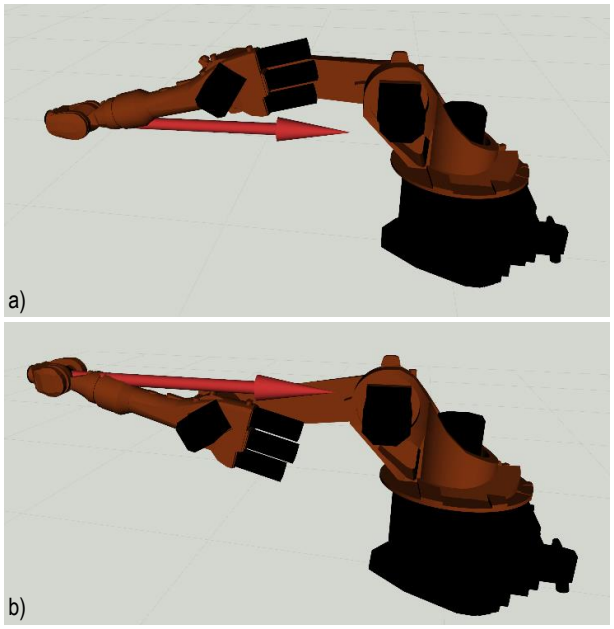

Fig. 9. Two different arm configurations for the same effector position: (a) elbow up; (b) elbow down

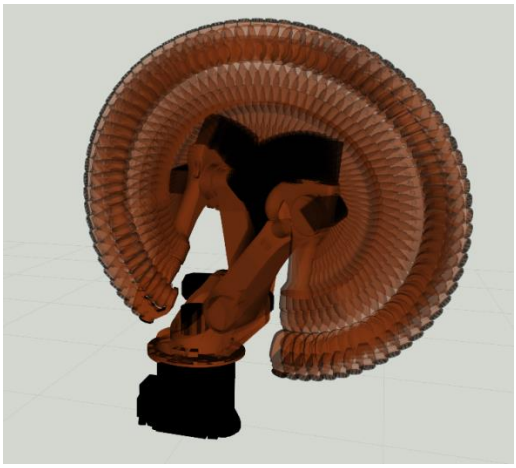

Fig. 10. Displaying a change in the configuration of the arm, which is forced by the need to rotate the first joint by 360° during the execution of MoveIt

Furthermore, it is also necessary for a more realistic simulation to limit the space in which it is possible to plan future target positions. We limited the planning space by adding collision objects, through which the planned trajectory must not pass. Thus, a floor was created on which the robot is placed, and the workspace of the robot was also partially limited, which is further shown in Fig. 17.

7.4. Teaching the path of a robot by guidance

The teaching process is simple and is performed during manual navigation. Whenever the arm stops because no force is applied to the sensor, its current position and orientation in the Cartesian system can be written to the file (these points are shown in Fig. 11 under the label data 2). When teaching is complete, all memorized positions and orientations are uploaded to the ComputeCartesianPath service. This will create additional extra positions, which can be seen as data 1 in the same image. The entire trajectory is then planned as one continuous movement.

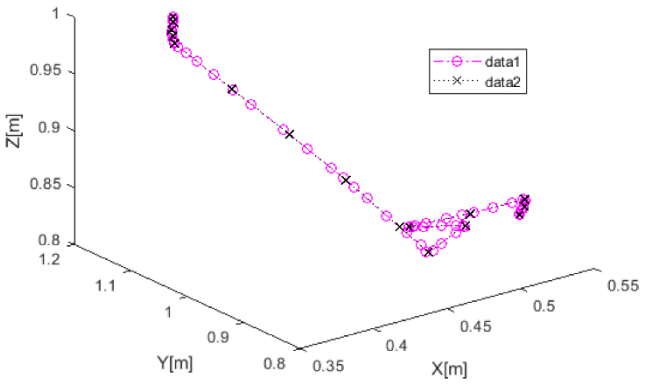

Fig. 11. Points written to a file during the manual navigation (data 2) and extra points created by ComputeCartesianPath service (data 1)

8. EXPERIMENTS

During the experimental verification of the solution, it was first necessary to verify whether it is possible for MoveIt to enter the desired target positions based on calculation of the position vectors of these points from the force vector. It was also necessary to verify what positioning error occurs with this solution. In Fig. 12, an arrow with blue label shows the position vector pointing from the tool coordinate system, obtained from the resultant force vector. This vector points to the desired target position. The cusped line with orange label represents the planned motion trajectory with the individual nodal points. The endpoint of the manipulator moves along the orange trajectory. The inaccuracy of the reached target position compared to the planned one was 0.1 mm, which is an acceptable deviation when manually guiding the robot via the FT sensor.

Subsequently, the sequential planning of trajectories was verified using MoveIt while pressure was applied to the sensor in the desired direction. The planned trajectory for the new calculated point was prepared immediately after completing the execution of the previous plan. The chart in Fig. 13 shows the shortcoming of this method, as the robot stops every time it passes the appropriate trajectory.

DOI 10.2478/ama-2021-0023

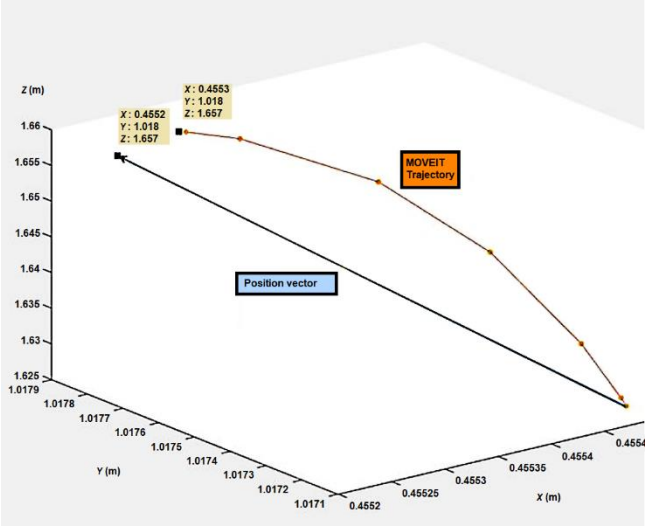

Fig. 12. Position vector pointing from the tool coordinate system to the desired position of the effector (arrow with blue label), and the planned trajectory with its via points (cusped line with orange label)

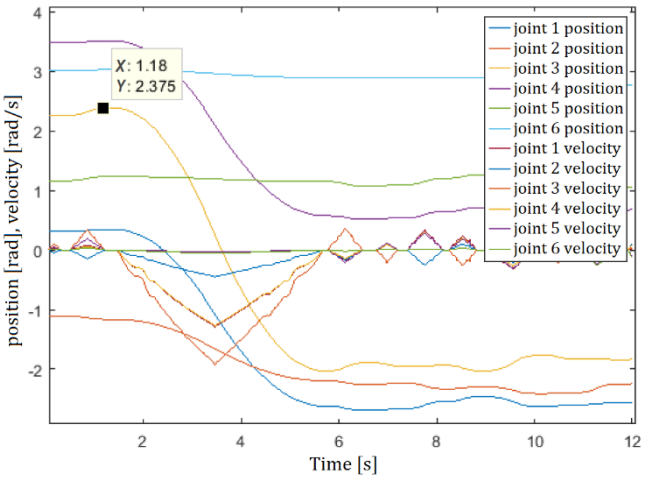

Fig. 13. Triangle-like lines represent speeds in the joints. It is obvious that the robot will stop between the individual plans

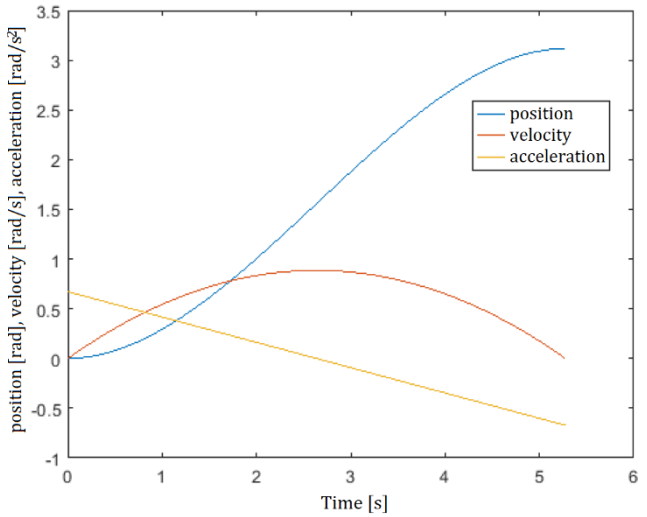

Fig. 14. Basic experiment to verify the implementation of polynomial interpolation performed on one joint. The output of the algorithm is the desired trajectory for the position, speed, and acceleration

The proposed algorithm eliminates these unwanted stops and provides smooth transitions between successive trajectories. First, verification of the algorithm was performed on one of the joints. The joint trajectory planning based on polynomial interpolation works as expected. The limiting conditions in this experiment were determined as follows: initial joint position $q(t_0) = 0$ rad, final joint position $q(t_f) = 3.11723$ rad, and the initial and final joint speeds $\dot{q}(t_0) = 0$ rad/s and $\dot{q}(t_f) = 0 \frac{\text{rad}}{\text{s}}$, respectively. In Fig.14, we see the resulting desired trajectories for the position, speed, and acceleration.

To verify the algorithm's functionality with continuous trajectory replanning, an experiment was created with smooth switching of planned trajectories. We can use the same trajectory planned in the previous experiment, and additionally, we request a switch to the new trajectory at time $t^* = 1.84$ s. We send a request to the query_state service with a timestamp equal to t^*. The response is the future value of the position $q(t^*) = 0.882$ rad and speed $\dot{q}(t^*) = 0.807$ rad/s. We use these values as the initial limiting conditions for generating a new desired trajectory. For safety reasons, we set the final limiting condition for speed to $\dot{q}(t_f) = 0$ rad/s. During normal manual guidance, the value of the final joint position in the new trajectory is given based on the inverse kinematics from the point in space to which the position vector (computed from the acting force) is directed. In other words, the trajectories are replanned periodically, and application of some amount of force causes changes in the limiting conditions used in the polynomial interpolation. For this experiment's purpose, however, we consider the joint's target position to be the value $q(t_f) = 0$ rad. The resulting desired trajectory is shown in Fig. 15. A new trajectory is created by combining the two plans, where there is no interruption of movement in position or speed, but the movements follow each other smoothly. We can see in Fig. 15 that before the moment of 1.84 s, the values with index "1" are valid, and after this moment, the values with index "2" are valid. The original values of the first trajectory after time $t^* = 1.84$ s are shown in dashed lines. With a cubic trajectory, the acceleration change is not continuous, but the controller can regulate this step change.

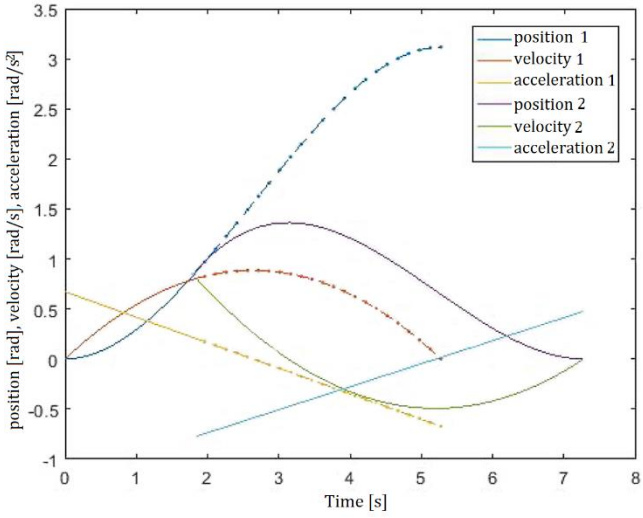

Fig. 15. Verification of the algorithm's functionality with continuous desired trajectory replanning

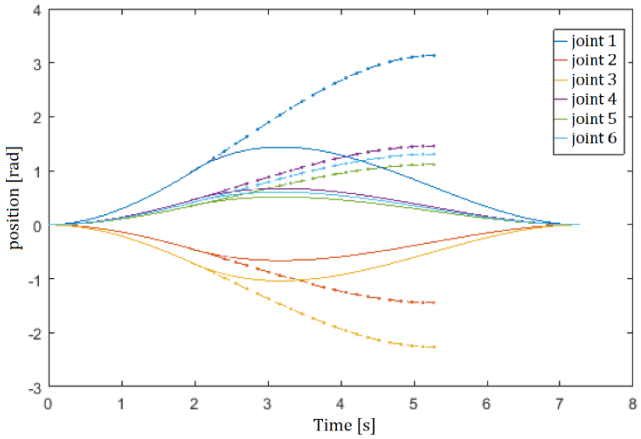

Fig. 16. Continuous trajectory replanning simultaneously in all six degrees of freedom of the robotic manipulator

A similar experiment was performed to smoothly switch to new desired trajectories simultaneously in all six degrees of freedom of the robotic manipulator. The result is shown in Fig. 16.

Finally, the additional space constraints of the robot were also verified. If the robot is manually guided by a sensor to move, such a planned trajectory that will pass through collision objects does not apply, and the robot stops. Fig. 17 shows in red the force vector that acts on the robot via the FT sensor in the effector's coordinate system and forces it to collide with the floor. However, this plan was assessed as dangerous, and therefore no collision occurred.

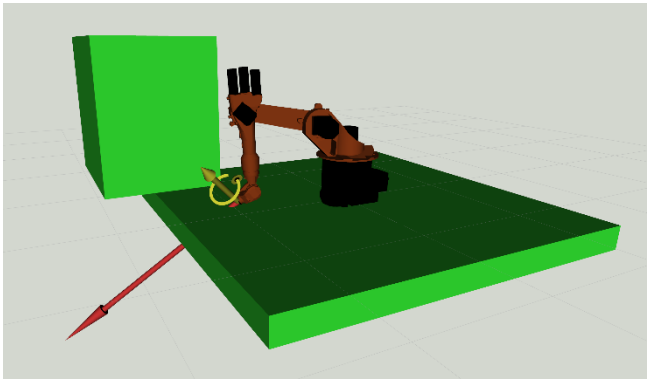

Fig. 17. The environment where the space constraints of the robot were verified

9. CONCLUSION

This article presented the implementation of the Optoforce HEX-70-CE-2000N sensor into the ROS framework for use in manual robot guidance. It was also shown that the sensor needs to be properly calibrated. In the ROS environment, the robot controllers were set so that the control of the robot endpoint also considers the influence of external forces measured on the Optoforce sensor. These are reconstructed as the force and torque vectors from the detected forces in all degrees of freedom. This work verifies the algorithm of motion planning of the virtual manipulator's effector based on the action of forces and torques on the force sensor using the basic settings of the MoveIt framework. This method of motion planning proved to be insufficient for the needs of the robotic effector's manual guidance because the robot is stopped after the execution of each scheduled partial trajectory. Therefore, our research was focused on the design and verification of a robot motion planning algorithm that uses polynomial interpolation and allows the continuous merging of subplans, which is necessary for online manual guidance. Experimental verification of the solution shows a smooth connection between the original and the new plans. The required trajectory is obtained without a step change in position and speed. To bring the simulation closer to reality, the selected implementation's basic spatial and speed limitations were added to the solution. At the same time, this increased the solution's safety, which is very important in manual guidance.

In future work, we will focus on optimizing the planning system to accelerate responses to changes in force concerning ergonomics for the operator. The system supervising the appropriate selection of the inverse kinematics solution will also be optimized to eliminate random selection and ensure the calculation of the solution taking into account the robot's current configuration. One of the options that will be verified in connection with this problem is the IKfast solver. Finally, the proposed system will be deployed and verified on a real robot.

REFERENCES

1. **Jamone L., Fumagalli M., Natale L, Nori F.** (2014), Control of physical interaction through tactile and force sensing during visually guided reaching, *2014 IEEE International Symposium on Intelligent Control (ISIC)*, 1360-1365.
2. Experimental Packages for KUKA manipulators with ROS-Industrial (2020), https://github.com/ros-industrial/kuka_experimental
3. **González C., Solanes J.E., Muñoz A., Gracia L., Girbés-Juan V., Tornero J.** (2021), Advanced teleoperation and control system for industrial robots based on augmented virtuality and haptic feedback, *Journal of Manufacturing Systems*, 59, 283-298.
4. **Jo J., other authors** (2013), Grasping force control of a robotic hand based on a torque-velocity transformation using F/T sensors with gravity compensation, *IECON 2013-39th Annual Conference of the IEEE Industrial Electronics Society*, 4150-4155.
5. Joint Trajectory Action Contoller, Official webpage ROS Documentation (year), http://wiki.ros.org/robot_mechanism_controllers/JointTrajectoryActionController
6. **Lee S.-D., Ahn K.-H., Song J.-B.** (2016), Torque control based sensorless hand guiding for direct robot teaching, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 745-750.
7. **Loske J., Biesenbach R.** (2014), Force-torque sensor integration in industrial robot control, *15th International Workshop on Research and Education in Mechatronics (REM,* 1-5.
8. **Massa D., Callegari M., Cristalli C.** (2015), Manual guidance for industrial robot programming, *Industrial Robot: An International Journal*, 42(5), 457-465.
9. **Matheson E., Minto R., Zampieri, E.G., Faccio M., Rosati G.** (2019), Human–Robot Collaboration in Manufacturing Applications: A Review, *Robotics*, *8*(4), 100.
10. **Noh Y., Bimbo J., Sareh S., Wurdemann H.** (2016), Multi-axis force/torque sensor based on simply-supported beam and optoelectronics, *Sensors*, 16(11), 1936.
11. **Peng Y.C., Chen S., Jivani D., Wason J., Lawler W., Saunders G., Wen, J.** (2021), Sensor-Guided Assembly of Segmented Structures with Industrial Robots, *Applied Sciences*, 11(6), 2669.
12. **Reyes-Uquillas D., Hsiao, T.** (2021), Safe and intuitive manual guidance of a robot manipulator using adaptive admittance control towards robot agility, *Robotics and Computer-Integrated Manufacturing*, 70, 102127.

DOI 10.2478/ama-2021-0023

13. **Safeea M., Béarée R., Neto P.** (2017), End-effector precise hand-guiding for collaborative robots, *Iberian Robotics conference*. Springer, Cham, 595-605.
14. **Safeea M., Bearee R., Neto, P**. (2017), End-effector precise hand-guiding for collaborative robots, *Iberian Robotics conference*, 595-605, Springer, Cham.
15. **Safeea M., Neto P., Béarée R.** (2019), Precise hand-guiding of redundant manipulators with null space control for in-contact obstacle navigation, *IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society*, 693-698.
16. **Spong M.,Hutchinson S.,Vidyasagar M.** (2005), Robot modeling and Control, 1st Edition. Wiley.
17. **Zhang S., Wang S., Jing F., Tan M.** (2019), A sensorless hand guiding scheme based on model identification and control for industrial robot, *IEEE Transactions on Industrial Informatics*, *15*(9), 5204-5213.
18. **Zhao Y., Gao F., Zhao Y., Chen, Z.** (2020), Peg-in-Hole Assembly Based on Six-Legged Robots with Visual Detecting and Force Sensing, *Sensors*, 20(10), 2861.
19. https://github.com/shadow-robot/optoforce, (2020)
20. https://ifr.org/free-downloads/, (2020)
21. https://onrobot.com/en/products/hex-6-axis-force-torque-sensor, (2021)
https://www.code-n.org/blog/finalist-optoforce-hungary-sensors-for-the-internet-of-things/, (2015)
22. https://www.crunchbase.com/organization/optoforce, (2020)
23. https://www.eu-robotics.net/cms/upload/topic_groups/SRA2020_SPARC.pdf, (2020)

This publication was created with support under the Operational Program Integrated Infrastructure for the project "Robotized cell for intelligent welding of small volume production (IZVAR)", code ITMS2014+: 313012P386, cofinanced by the European Regional Development Fund.

The authors also gratefully acknowledge the contribution of the Slovak Research and Development Agency under the project APVV-17-0214 and the contribution of the Scientific Grant Agency of the Slovak Republic under the grant 1/0754/19.

Radovan Gregor: https://orcid.org/0000-0002-9498-6768
Andrej Babinec: https://orcid.org/0000-0001-5550-2583
František Duchoň: https://orcid.org/0000-0003-4140-9737
Michal Dobiš: https://orcid.org/0000-0002-2453-212X