ISSN 0146-4116, Automatic Control and Computer Sciences, 2018, Vol. 52, No. 5, pp. 392–401. © Allerton Press, Inc., 2018.

Integration of Computervision and Artificial Intelligence Subsystems with Robot Operating System Based Motion Planning for Industrial Robots[1]

Janis Arents[a], *, Ricards Cacurs[a], **, and Modris Greitans[a], ***

[a] *Institute of Electronics and Computer Science, Riga, LV-1006 Latvia*

**e-mail: janis.arents@edi.lv*

***e-mail: ricards.cacurs@edi.lv*

****e-mail: modris.greitans@edi.lv*

Received June 4, 2018; in final form, July 13, 2018

Abstract—The paper proposes flexible system that is based on Robot Operating System framework for integration of 3D computer vision and artificial intelligence algorithms with industrial robots for automation of industrial tasks. The system provides flexibility of 3D computer vision hardware and industrial robot components, allowing to test different hardware with small software changes. The experimental system consisting of Kinect V2 RGB+Depth camera and Universal Robots UR5 robot was set up. In experimental setup the pick and place task was implemented where randomly organized two types of objects (tubes and cans) where picked from the container and sorted in two separate containers. Average full cycle time for the task was measured to be 19.675 s.

Keywords: computer vision, 3D vision, industrial robots, automation, artificial intelligence, robot operating system

DOI: 10.3103/S0146411618050024

1. INTRODUCTION

Between 2017 and 2020, it is estimated that more than 1.7 million new industrial robots will be installed in factories around the world [1]. Majority of installed industrial robots will be used to automatize processes where robot motions can be preprogrammed in order to manipulate with objects whose position and orientation are predefined in 3D space. But there are a lot of tasks where information about object position and orientation is not predefined, making it impossible to preprogram the task scenarios beforehand. This leaves an empty gap in automation to be filled, by developing solutions that allow acquiring real-time information about the region of interest and work object (shape, position, orientation).

Currently we are going through 4th industrial revolution, which is more commonly known as Industry 4.0 where the tools provided by the advancements in operational, communication, and information technology are used to increase the levels of automation and digitization of production, in manufacturing and industrial processes [2].

Recent development in artificial intelligence and computer vision algorithms enable new possibilities for automation of processes where an ability to act and make decisions regarding the corresponding situation is needed. The study performed by *Accenture* shows that artificial intelligence will boost economic growth in manufacturing by factor of 2.1 by the year 2035 [3].

The progress in 3D camera hardware area has enabled ways of acquiring increasingly precise, real-time three-dimensional information about the workspace environment. This is crucial component, that allows gathering information about the automation process and act accordingly to execute the task. Market research performed by *P&S Market Research* shows projected exponential growth in 3D camera market (Stereo cameras, time of flight cameras and structure light cameras) in upcoming years [4]. It means that new cameras with improved features are constantly being developed. This makes beneficial to create flexible system where the cameras can be easily switched, without making major software modifications to integrate new camera.

[1] The article is published in the original.

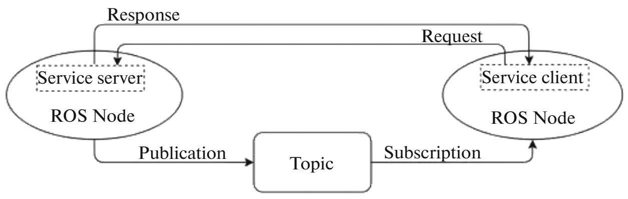

Fig. 1. Basic ROS computational graph.

In the context of a flexible system, similar to the approach of using different cameras, it is crucial to easily switch between industrial robot models supplied by different robot manufacturer's. Every industrial robot manufacturer has developed its own programming language, simulation environment and the way how computer vision software is integrated for robot control may differ for every manufacturer. Because of the fact that development process is different for every manufacturer, most of the companies that are using industrial robots are trying to hold on to one robot manufacturer. But not every industrial robot manufacturer robots are suitable for various environments or working fields. For example, *Staubli* is one of few companies who offer robots for humid environments [5]. Main profit of flexible system is opportunity to choose the most suitable robot and sensors for production environment disregarding which manufacturer supplies it.

For this paper, the selected automation task is sorting different kinds of objects, which are randomly distributed in the container. Actual challenge for developing such system is to bring together computer vision components that provide 3D information with robot manipulator. The aim of the paper is to identify way how to build such a system in flexible way, allowing to easily integrating different 3D camera manufacturers with different manipulator manufacturer systems.

The paper is organized as follows. Section Preliminaries and related work overviews existing work that are key components for such system and shows similar existing systems. Section proposed system describes our proposed system in detail. Section Experimental setup and Results describes our experimental setup and provides results for pick and place solution where two types of objects are sorted. In last section our contribution and results are summarized.

2. PRELIMINARIES AND RELATED WORK

2.1. Robot Operating System

ROS, an open-source project, provides a common framework for robotics applications. ROS is heavily utilized by the research community for service robotics applications, but its technology can be applied to other application areas, including industrial robotics. ROS capabilities, such as advanced perception and path/grasp planning, can enable manufacturing robotic applications that were previously technically infeasible or cost prohibitive [6]. ROS simplifies complex programming tasks by providing a robust and flexible framework where developers and researchers can focus on new algorithm development without coping with trivial low-level hardware communication problems. In terms of flexibility ROS offers standardized process communication and variety of drivers for interfacing different kind of hardware. In Fig. 1 three typical components of ROS are illustrated that are main building blocks for development. Nodes are basic processes that perform the computation, topics are publish/subscribe method of message exchange and services establishes request/response communication model. Communication between these three components is facilitated by messages, which are a bundle of information packaged as a strictly typed data structure. Typically, these are comprised of standard primitive types, but also may contain other messages [7].

2.2. Computer Vision

Computer vision area is constantly growing whitch can be seen from the fact, and the market research done by *Tractica* shows that computer vision market revenue will reach $48.6 billion by the year 2022 [8]. In robotic applications there is particular interest in 3D computer vision because many automation tasks require to manipulate with 3D objects in 3D space. In recent years more and more commercially available 3D camera hardware is appearing, providing increasingly higher resolution, and higher accuracy 3D information. The hardware is also getting cheaper and more accessible. The hardware is easier to use, and cameras are factory calibrated, making them usable out of the box. This reduces required time to test camera for particular application.

Currently there are three main technologies of 3D cameras: Passive 3D imaging, active infrared time-of-flight cameras and active structured light cameras. Passive 3D imaging is multiple-view 3D imaging systems that recover 3D information from scenes that are illuminated only with ambient lighting [9]. Classical 3D stereo system consists of at least two cameras and 3D information is reconstructed by triangulating points seen in both cameras. The time-of-flight (TOF) technique uses an active emitter to modulate the light in time domain, an optical sensor collects the light scattered back by the object, recovers depth information by calculating the time delay from the signal leaves the device and the signal returns to the device [10]. The structured light technique belongs to one of the active methods and utilizes a projection device to actively project structured patterns. The structured light system is similar to a stereo system with the difference of replacing one camera with a projector [11]. For all these technologies, there are low cost (int the range of $500) commercially available products. For time-of-flight cameras, popular solution is *Kinect V2* and measured accuracy by [12] of 3D reconstruction is about 1 cm depending on the calibration. Popular product for passive stereo system is *Stereo labs ZED camera* and RMS error at 1m distance measured by [13] is 9 mm. Popular products of for structured light depth sensors is *Intel RealSense* sensors. D400 is the latest family of these sensors, but currently there are no studies about the accuracy of this family. For the previous generation D200 family, the RMS noise at 1 m distance was measured about 2 mm, measured by [14]. The depth cameras vary by many factors like resolution, accuracy, sensing range, sensitivity and other factors. Each sensor manufacturer provides different Software Development kit and tools. If it is necessary to integrate multiple kinds of sensors for testing, the software must be developed using each manufacturers SDK and this is very time consuming. This can be overcome by using Robot Operating System, which provides software drivers for most popular Depth sensing devices including aforementioned – *Kinect V2*, *ZED camera* and *Intel RealSense* cameras. The interactions with cameras is then done in ROS standardized way, allowing to integrate different kinds of sensors in a much less time consuming way.

The data acquisition is only one part for the full computer vision system. To use computer vision for automation tasks, it is necessary to perform data processing to detect, classify and identify different objects from data. The research in computer vision has brought many different methods for different tasks, and there are a lot of scientific papers that explains these details. But usually it is very time consuming to implement particular methods using information from the scientific papers. For this reason, it is useful to use existing software libraries that provide implementation of these methods. The most popular such library is *OpenCV* that is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products [15].

2.3. Artificial Intelligence

In the last years the Artificial Intelligence based approaches are dominating in several tasks of computer vision solutions (object detection, classification, identification). In 2012 the Imagnet Large Scale Visual Recognition Challenge [16] was won by method that was based on Convolutional Neural Networks [17]. It was first time that this challenge was won by method where image features for classification were fully trained from the training data and not hand designed and the classification accuracy was significantly improved over previous methods. This started the widespread interest in research in Deep Artificial Neural Networks, and the field is growing rapidly allowing automation of more and more complex tasks. For AI solutions to work it is necessary to train the systems with large amount of data, which requires large processing power. Rapid development of parallel computation using graphical processing units in recent years enabled to train more and more complex AI solutions in reasonable time. The interest in the field sparked development in software frameworks for Artificial Intelligence solutions. This allows faster development, integration and validation of Artificial solutions in different applications. Most popular AI software frameworks are: Tensorflow, Theano, Caffe, Keras, Torch.

2.4. Industrial Robot Control Using ROS

The way how industrial robots are programmed strongly depends on the robot manufacturer. As we can see in Table 1, every industrial robot manufacturer has developed its own programming language, simulation software and possibilities, what their software offers differs from brand to brand. In terms of flexible system, hardware abstraction is important so industrial robots from different manufacturers could be programmed and controlled in one way. There are multiple frameworks who offer compatibility with different industrial robot brands but most popular of them is ROS-Industrial [18].

Table 1. Industrial robot manufacturers and their software

Manufacturer	Programming language	Simulation software
Universal Robots	URScript	URSim
ABB	RAPID	RobotStudio
Kuka	KRL	KUKA Sim
Kawasaki	AS	K-ROSET
Stäubli	VAL3	Stäubli Robotics Suite
Comau	PDL2	Process Simulate
Yaskawa	Inform	MotoSim
Fanuc	Karel	RoboGuide
Epson	SPEL+	Epson RC+
Nachi Fujikoshi	SLIM	FD on Desk
Adept	V+	Adept ACE

ROS-Industrial basically extends the advanced capabilities of ROS software to industrial robots working in the production process. ROS-Industrial consists of many software packages, which can be used for interfacing industrial robots. These packages are BSD (legacy)/Apache 2.0 (preferred) licensed program, which contains libraries, drivers, and tools for industrial hardware [19]. ROS-I architecture can be seen in Fig. 2, where in terms of flexible system important is ROS-I controller layer. ROS-I controller layer contains vendor-specific packages (drivers) for interfacing different manufacturer supplied industrial robots. Besides vendor specific driver, important section is URDFs (*Unified Robot Description Format*) which is standard ROS XML representation of the robot model and is unique for every robot model. Furthermore, URDF is needed by MoveIt! software for motion planning, which is described in next section.

2.5. MoveIt! Software

MoveIt! is motion planning framework, which consists of multiple different packages and tools for motion planning, 3D perception, collision detection, kinematics, robot manipulation and control [20]. As

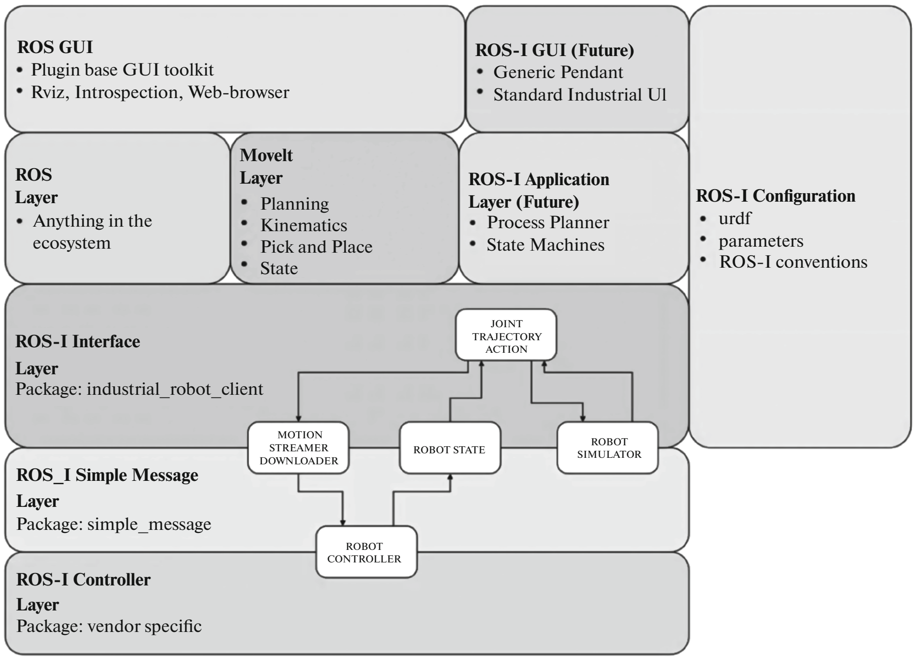

Fig. 2. ROS-Industrial high level architecture [18].

mentioned in previous section every robot model has unique URDF and basically that is the only thing that differs in MoveIt! layer for vendor specific industrial robot control. The central node in the MoveIt! software is called *move_group* and serves as an integrator: pulling all the individual components together to provide a set of ROS actions and services for users to use [20]. Moreover, to use MoveIt! capabilities and communicate with the robot interface nodes MoveIt! configuration package is needed. MoveIt! configuration package is created from URDF robot data by MoveIt! setup assistant GUI and consists of launch and configuration files. Furthermore, there is no difference what are controlled robot specifications and brand. MoveIt! continues ROS concept of flexibility in software architecture. Most of MoveIt! capabilities are plug-in based which means that it is easy to integrate and use different kind of motion planners and kinematic solvers.

2.6. Motion Planning

In applications where it is not possible to predefine how the robot must move in order to manipulate with objects, for example in pick and place task where randomly distributed objects must be picked from the container, motion planning more or less is done in real time or before the corresponding move. Motion planning can be defined as optimal collision-less trajectory finding technique how to move the robot from start state to goal state while satisfying all the constraints. In order to find collision-less trajectory-planning scene must be defined. Stationary objects in planning scene can be defined in URDF robot model so motion-planning algorithms take them into account. Another option is to build *Octomap* [21] from 3D sensors, which can be directly passed into FCL (*Flexible Collision Library*), the collision-checking library that MoveIt! uses [20].

MoveIt! works with motion planners through a plugin interface. This allows MoveIt! to communicate with and use different motion planners from multiple libraries, making MoveIt! easily extensible. The interface to the motion planners is through a ROS action or service (offered by the move_group node). The default motion planners for move_group are included in OMPL (*Open Motion Planning Library*) [22] and configured when MoveIt! configuration package is generated. Additionally STOMP (*Stochastic Trajectory Optimization for Motion Planning*) [23] and other motion planners can be easily integrated in motion planning environment.

OMPL specializes in sampling-based motion planning, the probabilistic roadmap (PRM) [24] is among the first sampling-based motion planners. This approach utilizes random sampling of the state space to build a roadmap of the free state space and finding the shortest path between start state and goal state [25]. Another sampling-based method is creating tree structures of the free state space. This method begins by rooting a tree at the starting configuration of the robot. With the first node of the tree intact, random sampling of the free space then occurs. The planner employs an expansion heuristic, which typically gives the method its name, from which the sample is connected to the tree along a collision free path [25]. STOMP approach relies on generating noisy trajectories to explore the space around an initial (possibly infeasible) trajectory, which are then combined to produce an updated trajectory with lower cost. A cost function based on a combination of obstacle and smoothness cost is optimized in each iteration [23].

3. EXISTING COMPUTER VISION SYSTEMS FOR INDUSTRIAL ROBOT AUTOMATION

There are multiple existing approaches for automated randomly distributed object bin-picking solutions that integrate 3D computer vision solutions with robotic manipulators. One of the research groups are using system that consists of KUKA KR 16 with KR C4 controller and Sick Ranger camera with two laser projectors. The system is described in papers [26], [27], [28]. The drawback of the system is that it does not provide flexibility in terms of components (video cameras and manipulators). It is designed for one type of robot manipulator and camera.

There have been works that emphasize the flexibility on the hardware components. For example [29] has developed flexible vendor independent framework using *Microsoft.NET* framework called DirectControl 3. But there is no information where to find this software and how to use it. The experimental setup consisted of an ABB IRB 4600 industrial robot and the corresponding IRC5 controller and a SICK Ruler vision system based on laser triangulation technology.

In [30] similar system to the proposed system in this paper is provided, that is built using ROS framework. But focus on flexibility is on different scenarios of object recognition and handling, but not the hardware components.

There are also publications where the main focus is on the computer vision algorithms and methodology on object detection that can be picked with robotic manipulator and object recognition. In [31] the flexible

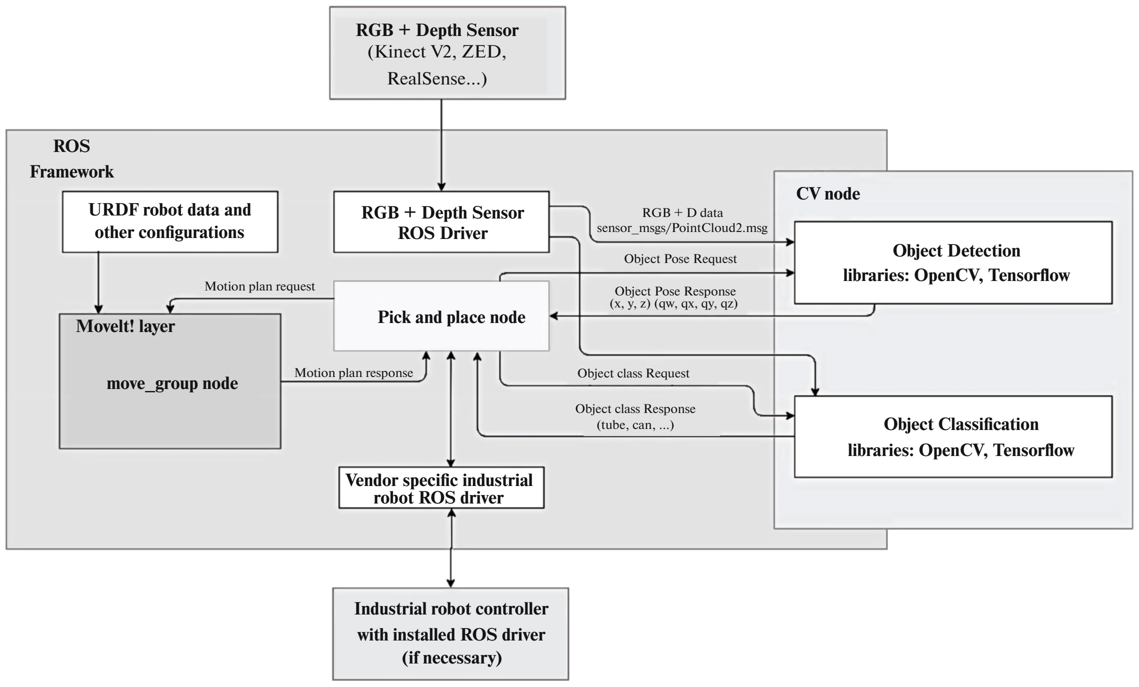
RGB + Depth Sensor
(Kinect V2, ZED, RealSense...)
ROS Framework
URDF robot data and other configurations
RGB + Depth Sensor ROS Driver
CV node
RGB + D data
sensor_msgs/PointCloud2.msg
Object Pose Request
Object Detection libraries: OpenCV, Tensorflow
Motion plan request
Object Pose Response (x, y, z) (qw, qx, qy, qz)
Pick and place node
MoveIt! layer
move_group node
Motion plan response
Object class Request
Object class Response (tube, can, ...)
Object Classification libraries: OpenCV, Tensorflow
Vendor specific industrial robot ROS driver
Industrial robot controller with installed ROS driver (if necessary)

Fig. 3. Proposed system.

bin-picking system based on RANSAM algorithm is described that can deal with any kind of object. In publication four kinds of objects where picked and the success rate depending on the object type ranged from 50–100%. In [32] dynamic object recognition system is proposed detects landmark features using neural network and provides grasping points of randomly located objects.

4. PROPOSED SYSTEM

The architecture of the proposed system is illustrated in Fig. 3 identifying all key elements of the system. The computer vision node (CV node in Fig. 3) is software component that processes the data of RGB+Depth sensor. It contains two parts: Object detection part and Object classification part. This node uses ROS framework only for data communication. The sensor data is received from the sensor driver in sensor_msgs.PointCloud2.msg format, and it contains 3D point cloud as well as the RGB color information. This provides an abstraction layer for different devices, because all these kind of sensors provide the data in the same format (with small variances), and it is easy to integrate different types of RGB+Depth sensors.

Object detection block is responsible for object detection that can be picked by industrial robot. This block takes the RGB+Depth sensor data, and produces pose of the object: position (x, y, z) and orientation in quaternion format (qw, qx, qy, qz). The pose is sent to pick and place node, when industrial robot is ready to pick an object.

Object classification block is responsible for the object classification. When industrial robot picks object, it is then classified using Convolutional Neural Network. This block allows sorting different types of objects. This block takes the RGB+Depth sensor data and sends the picked object class to the pick and place node on request.

All the software for this node is implemented using Python 2.7 programming language, which allows fast software development. For processing OpenCV library is used for basic computer vision algorithms, and TensorFlow library is used for artificial intelligence algorithms.

Pick and place node works as integrator of all subsystems and is mainly responsible for motion plan request creation depending on object position in container and object class. Pick and place node is made as flexible as possible to easily adjust configurations for different manufacturer industrial robots where vendor specific modules are easily changeable. Furthermore all the algorithms and system logic does not differ if different manufacturer robot model is used. Interface with MoveIt! layer is managed through *move_group_interface* package where access to *move_group* node is realized with C++ API. As MoveIt!

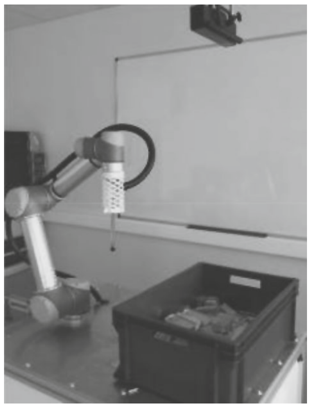

Fig. 4. Experimental setup.

works with motion planners through a plugin interface it is possible to choose easily the most appropriate motion planner for corresponding automation task and regarding workspace.

5. EXPERIMENTAL SETUP AND RESULTS

Experimental setup is illustrated in Fig. 4. The industrial robot, which is used in experimental system, is UR5 collaborative robot from Universal Robots with controller CB3. It consists of 6 rotating joints, payload is 5 kg, reach is 850 mm and repeatability is +/– 0.1 mm [33]. Universal robot ROS driver [34] is used, which can be downloaded from authors git repository [35] and robot URDF data from official ROS-Industrial git repository is used [36]. The robot URDF data was updated with stationary workspace elements and end effector, which was designed and built for this experimental setup. MoveIt! configuration package was generated with updated URDF and other experimental system configurations.

For 3D vision *Kinect V2* RGB+Depth sensor was used. The resolution for RGB sensor is 1920 × 1080 and resolution for Depth sensor is 512 × 424. The maximum frame rate for the camera is 30 Hz. Minimum depth distance 50 cm. The camera is connected to the computer through USB 3.0 interface. Software used for interaction with *Kinect V2* was Iai_Kinect2 ROS package that provides interface for data acquisition.

For the experimental system two computers were used. One for the interaction with Kinect sensor and object detection and classification with following specifications: Intel Xeon E-31245 V6 3.7 GHz, 16 GB RAM and NVidia Quadro P1000 graphics card. Second computer was used for motion planning and robot control with the following specifications: Intel Core i5-3470 CPU@ 3.20 GHz and 16 GB RAM. Computers where connected in LAN via Ethernet. The modularity of ROS made it simple to run the nodes on separate machines. Both computers were running Ubuntu 16.04 version with ROS version Kinetic Kame.

OMPL and STOMP motion planners were experimentally analyzed and compared in order to choose the most appropriate in this experimental system. Motion planners were evaluated using their default configurations in several tests were motions were planned 10 times from fixed start state to fixed goal state. Motion planners computed trajectories were compared by average planning time, trajectory length and success rate. In total 8 motion planners were experimentally tested, 7 of them were part of OMPL and 8th was STOMP. In first test motions were planned in relatively easy scenario without obstacles, with a purpose to eliminate those planners from further tests which average planning time is significantly higher than other planner average planning time. In the second test obstacles (table and box) was inserted in planning scene so motions planners had to take that into account and compute trajectories that avoid collisions with these obstacles. Only 4 motion planners were able to compute valid trajectories in this scenario. Average motion planning time for second test can be seen in Fig. 5. Furthermore, STOMP and RRTCONNECT

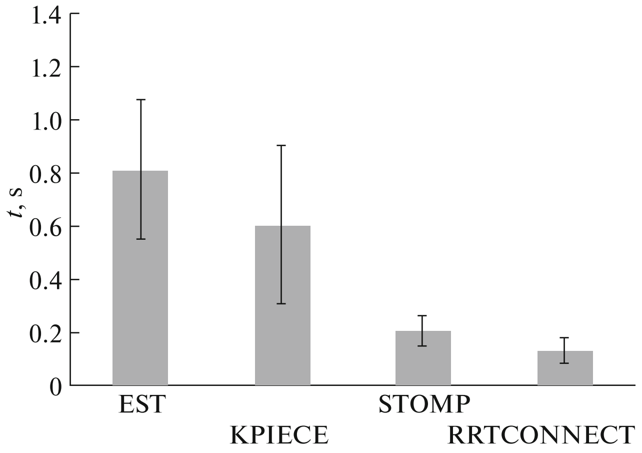

Fig. 5. Motion planner test.

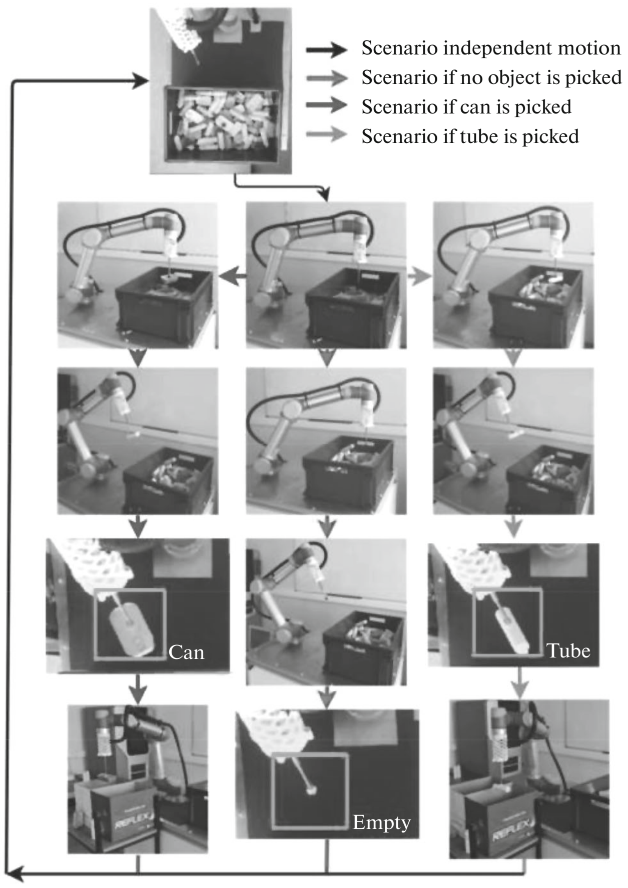
Scenario independent motion
Scenario if no object is picked
Scenario if can is picked
Scenario if tube is picked
Can
Tube
Empty

Fig. 6. The task sequence for the pick and place system.

motion planner trajectories were evaluated by trajectory length and repeatability because their average motion planning time was 3–5 times lower than other motion planner's average planning time. In result STOMP motion planner was chosen according to computed trajectories and their repeatability.

For the tests pick and place task was implemented where two kinds of objects (tubes and cans), that are randomly distributed in container in multiple layers where picked from the container and sorted in two different boxes. The full task cycle is shown in Fig. 6 for three scenarios: if the can is picked, if tube is picked and if gripping was unsuccessful and no object was picked.

Full pick and place cycle consists of 8 movements where motions are planned before every movement. Movements in full cycle are following: movement to prepick position 10 cm above detected object; object pick movement; movement back to prepick position; movement to classification position; movement to preplace position 10 cm above place position; place movement; movement back to preplace position; movement to wait position which is equal to classification position. Full cycle motion planning, movement execution, object detection and classification times can be seen in Table 2 where full cycle was exe-

Table 2. Full pick and place cycle

Action	Avg time	stdev	Max time	Min time
Motion planning	2.486	0.976	3.462	1.509
Movement execution	15.551	0.860	16.411	14.691
Object detection	1.625	0.131	1.755	1.494
Object classification	0.0132	0.0081	0.0213	0.005
Full cycle	19.675	1.975	21.651	17.699

cuted 10 times. Configurations for all the components, which are related to motion planning and movement execution, were default.

6. CONCLUSIONS AND FUTURE WORK

This paper proposes flexible system for integration of 3D computer vision and artificial intelligence solutions with industrial robots using ROS framework. ROS modularity, standardized communication interface between modules and abstraction allows development for robot control without deeper understanding of specifics of particular robot control. ROS makes it easier to integrate computer vision solutions with industrial robot components. Also, the integration is more flexible, allowing to test various manufacturer products for computer vision hardware and industrial robots.

In the section proposed system, the full system architecture is described that use ROS modularity to integrate computer vision hardware and artificial intelligence algorithms with industrial robot hardware. In this architecture all hardware components are easily replaceable, if there are software packages with drivers for these hardware components.

To test the architecture the experimental setup was created using real components: RGB+Depth sensor – *Kinect V2* and industrial robot – *Universal Robots UR5*. One of the main processes in experimental system work flow is motion planning, based on information from computer vision nodes about object position and object type. In most cases it is important to reduce the planning time, to increase work flow of the whole system. The ROS framework provides multiple existing motion planning packages, implementing different motion planning algorithms. In experiments two motion planners where compared "OMPL" and "STOMP" and it was concluded that most appropriate motion planner for UR5 robot was "STOMP" motion planner.

For the tests pick and place task was implemented where two kinds of objects (tubes and cans), that are randomly distributed in container in multiple layers where picked from the container and sorted in two different boxes. The experiments showed average full cycle time to be 19.675 seconds. The breakdown of the times for different parts of the system shows that the biggest bottle neck is the robot physical movements. All the components that are related to motion planning and trajectory execution were used with default configurations. Full cycle time can be reduced by adjusting motion planner configurations and by adjusting robots dynamical features – velocity and acceleration. Also, real time control could significantly decrease full cycle time. The future work includes the system optimization of the full cycle time of the task execution. One of the ways to increase cycle time is to reduce motion planning time. One way to reduce this time is to reuse some previous paths without motion planning, if the object coordinates doesn't change much between executions. Also, system is currently trained only on two types of objects, and addition of new objects requires acquiring datasets for machine learning algorithm training. To make this system universal for any kind of objects, the ways of how to efficiently train the system must be developed.

ACKNOWLEDGMENTS

The research leading to these results has received funding from the research project "Competency Centre of Latvian Electric and Optical Equipment Productive Industry" of EU Structural funds, contract no. 1.2.1.1/16/A/002 signed between LEO Competence Centre and Central Finance and Contracting Agency, Research no. 11 "The research on the development of computer vision techniques for the automation of industrial processes."

REFERENCES

1. IFR, Executive Summary World Robotics 2017 Industrial Robots. https://ifr.org/downloads/press/Executive_Summary_WR_2017_Industrial_Robots.pdf. Accessed July 6, 2018.
2. Gilchrist, A., *Industry 4.0. The Industrial Internet of Things,* Apress, 2016.
3. Purdy, M. and Daugherty, P., How AI boosts industry profits and innovation. https://www.accenture.com/us-en/insight-ai-industry-growth. Accessed July 6, 2018.
4. Global 3D Camera Market Size, Share, Development, Growth and Demand Forecast to 2022 – Industry Insights by Technology (Time of Flight, Stereo Vision and Structured Light Imaging), by Type (Free Camera and Target Camera), and by Application (Professional Cameras, Smartphone, Tablets, Computer and Other). https://www. psmarketresearch.com/market-analysis/3d-camera-market, 2016. Accessed July 6, 2018.
5. HE Robotic Arms (Humid Environment). https://www.staubli.com/en/robotics/product-range/6-axis-scara-picker-industrial-robots/sensitive-environments/humid-environment/. Accessed July 6, 2018.
6. About ROS. http://www.ros.org/about-ros/. Accessed July 6, 2018.

7. Aitken, J.M., Veres, S.M., and Judge, M., Adaptation of system configuration under the robot operating system, *IFAC Proc. Vol.,* 2014, vol. 47, no. 3, pp. 4484–4492.
8. Computer Vision Hardware and Software Market to Reach $48.6 Billion by 2022. https://www.tractica.com/newsroom/press-releases/computer-vision-hardware-and-software-market-to-reach-48-6-billion-by-2022/. Accessed July 6, 2018.
9. Se, S. and Pears, N., *Passive 3D Imaging,* Springer, 2012.
10. Hansard, M., Lee, S., Choi, O. and Horaud, R.P., *Time-of-Flight Cameras: Principles, Methods and Applications,* Springer Publishing Company, 2012.
11. Zhang, S., High-speed 3d shape measurement with structured light methods: A review, *Opt. Lasers Eng.,* 2018, vol. 106, pp. 119–131.
12. Lachat, E., Macher, H., Landes, T., and Grussenmeyer, P., Assessment and calibration of a RGB-D camera (Kinect v2 Sensor) towards a potential use for close-range 3D modeling, *Remote Sens.,* 2015, vol. 7, no. 10, pp. 13070–13097. https://doi.org/10.3390/rs71013070.
13. Fernandez, L., Avila, V., and Gonçalves, L., A generic approach for error estimation of depth data from (stereo and RGB-D) 3D sensors, 2017 (preprint).
14. Keselman, L., Woodfill, J.I., Grunnet-Jepse, A., and Bhowmik, A., Intel(R) RealSense(TM) stereoscopic depth cameras, *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW),* Honolulu, HI, 2017.
15. OpenCV, About OpenCV. https://opencv.org/about.html. Accessed July 6, 2018.
16. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., and Fei-Fei, L., ImageNet large scale visual recognition challenge, *Int. J. Comput. Vision,* 2015, vol. 115, no. 3, pp. 211–252.
17. Krizhevsky, A., Sutskever, I., and Hinton, G.E., ImageNet classification with deep convolutional neural networks, in *Advances in Neural Information Processing Systems,* 2012, pp. 1097–1105.
18. Ros-Industrial. https://rosindustrial.org/about/description/. Accessed July 6, 2018.
19. Lentin, J., *Mastering ROS for Robotics Programming: Design, Build and Simulate Complex Robots Using Robot Operating System and Master Its Out-of-the-Box Functionalities,* Packt Publishing, 2015.
20. Concepts Moveit! http://moveit.ros.org/documentation/concepts/. Accessed July 6, 2018.
21. Hornung, A., Wurm, K.M., Bennewitz, M., Stachnis, C., and Burgard, W., OctoMap: An efficient probabilistic 3D mapping framework based on octrees, *Auton. Rob.,* 2013, vol. 34, no. 3, pp. 189–206.
22. Sucan, I.A., Moll, M., and Kavraki, L.E., The open motion planning library, *IEEE Rob. Autom. Mag.,* 2012, vol. 19, no. 4, pp. 72–82.
23. Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., and Schaal, S., Stomp: Stochastic trajectory optimization for motion planning, *IEEE International Conference on Robotics and Automation,* 2011.
24. Kavraki, L.E., Svestka, P., Latombe, J.-C., and Overmars, M.H., Probabilistic roadmaps for path planning in high-dimensional configuration spaces, *IEEE Trans. Rob. Autom.,* 1996, vol. 12, no. 4, pp. 566–580.
25. OMPL, Open motion planning library: A primer. http://ompl.kavrakilab.org/OMPL_Primer.pdf. Accessed July 6, 2018.
26. Pochyly, A., Kubela, T., Singule, V., and Cihak, P., Robotic vision for bin-picking applications of various objects applications, *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*, 2010.
27. Pochyly, A., Kubela, T., Kozak, M., and Cihak, P., 3D vision systems for industrial bin-picking applications, *Proceedings of 15th International Conference MECHATRONIKA*, 2012.
28. Pochyly, A., Kubela, T., Singule, V., and Cihak, P., Robotic bin-picking system based on a revolving vision system, *2017 19th International Conference on Electrical Drives and Power Electronics (EDPE)*, 2017.
29. Schyja, A., Hypki, A., and Kuhlenkötter, B., A modular and extensible framework for real and virtual bin-picking environments, *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 5246–5251.
30. Tavares, P. and Sousa, A., Flexible pick and place architecture using ROS framework, *2015 10th Iberian Conference on Information Systems and Technologies (CISTI)*, 2015, pp. 1–6.
31. Buchholz, D., Winkelbach, S., and Wahl, F.M., Ransam for industrial bin-picking, *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*, 2010, pp. 1–6.
32. Kim, K., Cho, J., Pyo, J., Kang, S., and Kim, J., Dynamic object recognition using precise location detection and ANN for robot manipulator, *2017 International Conference on Control, Artificial Intelligence, Robotics Optimization (ICCAIRO)*, 2017, pp. 237–241.
33. Ur5 Technical Specifications. https://www.universal-robots.com/media/50588/ur5_en.pdf. Accessed July 6, 2018.
34. Andersen, T., Optimizing the Universal Robots ROS Driver. http://orbit.dtu.dk/files/117833332/Universal_Robot_report.pdf. Accessed July 6, 2018.
35. Andersen, T.T., The new driver for the ur3/ur5/ur10 robot arms from universal robots. https://github.com/ThomasTimm/ur_modern_driver. Accessed July 6, 2018.
36. ROS-Industrial Universal Robot Meta-Package. https://github.com/ros-industrial/universal_robot. Accessed July 6, 2018.