Real-time and Robust Collaborative Robot Motion Control with Microsoft Kinect® v2

Burak Teke*, Minna Lanz*, Joni-Kristian Kämäräinen†, Antti Hietanen†
* Department of Production Engineering, Tampere University of Technology
{burak.teke, minna.lanz}@tut.fi
† Department of Signal Processing, Tampere University of Technology
{joni.kamarainen, antti.hietanen}@tut.fi

***Abstract*—Recent development in depth sensing provide various opportunities for the development of new methods for Human Robot Interaction (HRI). Collaborative robots (co-bots) are redefining HRI across the manufacturing industry. However, little work has been done yet in the field of HRI with Kinect sensor in this industry. In this paper, we will present a HRI study using nearest–point approach with Microsoft Kinect v2 sensor's depth image (RGB–D). The approach is based on the Euclidean distance which has robust properties against different environments. The study aims to improve the motion performance of Universal Robot–5 (UR5) and interaction efficiency during the possible collaboration using the Robot Operating System (ROS) framework and its tools. After the depth data from the Kinect sensor has been processed, the nearest points differences are transmitted to the robot via ROS.**

***Index Terms*—Human–robot interaction, human–robot collaboration, collaborative robots, trajectory planning, Microsoft Kinect v2, ROS**

I. Introduction

In today's digital age, modern industries are expanding day by day and running with continuous production cycles. In addition to the expansion, robots have begun to appear in almost every task. An inevitable consequence of the frequent use of robots is that there will be more Human Robot Interaction (HRI) in future. Human, robot and system categories should be considered first to understand HRI clearly. Taxonomy of HRI metrics were presented in [1] which identified 42 distinct metrics with 9 branches totally. Summarized results can be seen in Fig. 1. In this study, interaction efficiency and safety issues are examined. Starting from the consideration of these metrics, smooth robot movement is considered more trustworthy than jerky movements. Besides that, if the movements are not recognized or planned movements, therefore the safety is in endangered and comfort in collaboration will be lower [2], [3].

On the other hand, intelligent machines and robots are expected to provide business-level services in the near future by collaborating autonomously in teams with humans as a critical part of a multi-company business process [4], [5]. Traditionally, fences prevent the operators from moving into robot's working area. The footprint of the safety protecting robot cells is large, and static safety measures prevent reconfiguration of the cells thus reducing the efficiency of work especially small batch size production. Due to economic

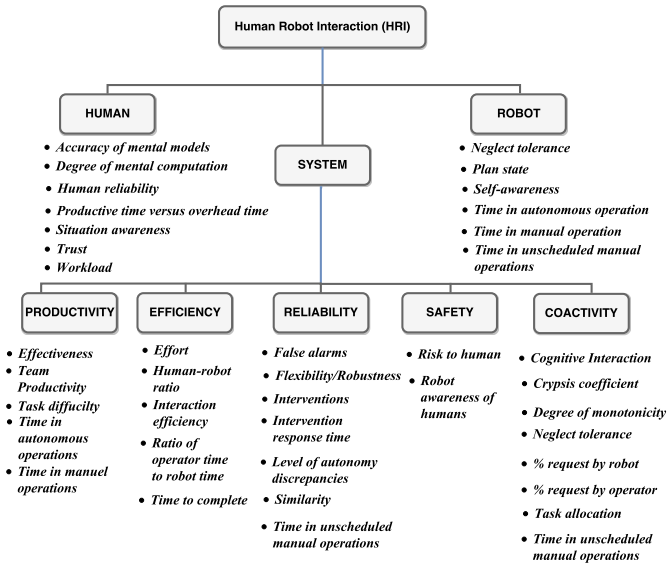


Fig. 1: Taxonomy of HRI Metrics based on [1].

reasons there is a need to decrease the factory footprint, allow existence of shared workplaces and maintain, and reconfigure the production environment while production is running.

Seamless and safe collaboration will require from the robot system transparency of operations, observation ability and intelligence. The robot system will observe and recognize the operator and his/her movements, thus providing accurate positioning and orientation of his/her physical form. With artificial intelligence, the intention can be extracted from the data and system can interfere (audio or visual feedback) in case the operator is not doing the tasks assigned to him at given point in time. In order to match the time frames of the robot and latencies originating from different sources, empirical part of this paper is explained.

II. Related Work

After the Kinect was launched in 2010, a great deal of research has been done until now. Significant number of researches is related to the gesture recognition and hand tracking application.

First of all, methods will be explained briefly; see [6] for more complete review. As one can see in [7], it proposes a

978-1-5386-4643-4/18/$31.00 ©2018 IEEE

method for tracking fingertips and palm center using depth data from Kinect. A big circle filter is applied for detecting palm center and following that fingertip detection is achieved. On the other hand, in [8] a different method is presented called Finger–Earth Mover's Distance (FEMD). It uses both color and depth data provided by Kinect to enhance robustness and efficiency. In addition to these, [9] uses K–means clustering for hand detection, eight point neighborhood for finger identification to recognize the gestures. A novel technique proposed by [10] which based on action graphs and its steps are segmentation of Kinect depth data, tracking, filtering, normalization and feature extraction to achieve the recognized gestures.

Apart from these methods, there are considerable amount of researches using both Kinect and robots. [11] worked on real time hand guiding of Universal Robot-5 (UR5). Kinect data were used to generate hand position and a smartphone was used for hand orientation. Both data are being sent to UR5 via a client. Robot navigation with the capabilities of the ROS and Kinect that relies on the fuzzy logic approach can be seen in [12]. Moreover, visual guidance systems are examined in [13], [14], [15] as good examples of the power of ROS and Kinect. Collision avoidance can be shown as a reflection of industry application in [16] and [17]. Both of them are using predictive methods to estimate collision–free UR5 robot trajectory.

In this particular HRI study, authors showed an efficient HRI with the planned trajectory based on nearest–point approach by using UR5, Microsoft Kinect and the power of ROS and its tools.

III. System Environment

A. Universal Robot 5 (UR5)

It is important that the robotic system should be reliable due to safety. The ISO specification is named ISO/TS 15066 [18] and is a supplement to ISO 10218 [19] "Safety Requirements for Industrial Robots" standards. ISO/TS 15066 describes the different collaborative concepts and details the requirements to achieve. It also presents a research study on pain thresholds, robot speed, pressure and impact for specific body parts. The Universal Robots' patented safety system features eight adjustable safety functions such as joint positions and speeds, Tool Center Point (TCP) positions, orientation, speed and force, momentum and power of the robot [20]. For these reasons, the six Degree of Freedom (DOF) UR5 from Universal Robots has been chosen to use in this study. The sketch of the coordinate frames according to the Modified Denavit-Hartenberg (MDH) convention is shown in Fig. 2. The robot is lightweight (18kg), its workspace and payload are 850mm and 5kg respectively.

B. Microsoft Kinect v2

Kinect is an RGB–D sensor that provides synchronized color and depth images. With its wide availability and lower cost than other traditional 3 Dimension (3D) cameras such as stereo cameras and Time-of-Flight (ToF) cameras [22], many researchers in computer science and robotics are leveraging the sensing technology to develop new ways in terms of interaction with machines [23], [24]. Kinect V2 has an RGB camera, infrared (IR) camera, IR emitter and multi-array microphone. In addition, right–handed coordinate system is also shown in Fig. 3.

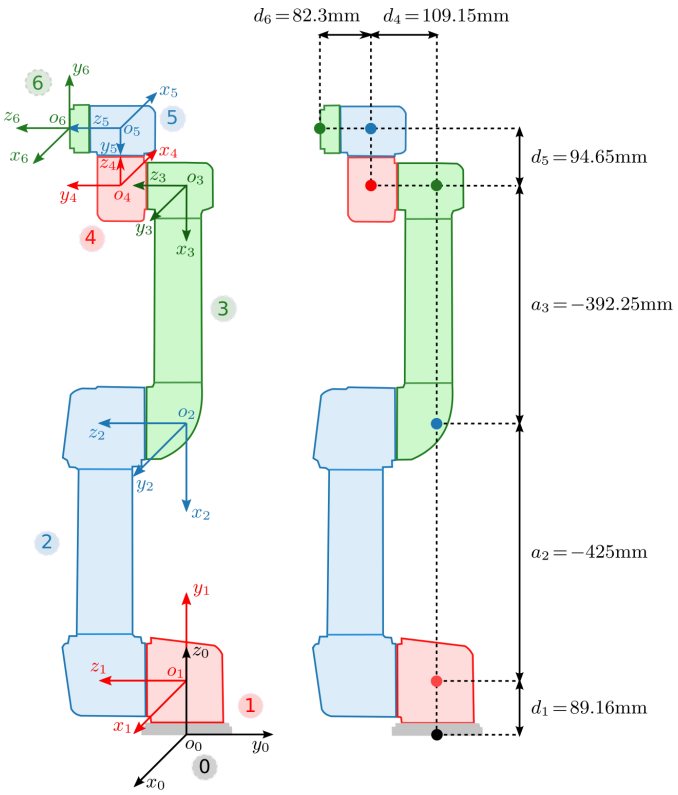


Fig. 2: Left of the figure shows sketch of the UR5 manipulator's coordinate frames according to the MDH convention and right shows the MDH parameters adapted from [21].

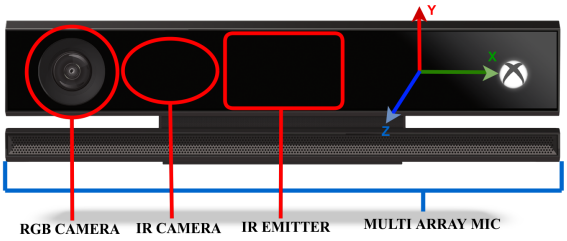


Fig. 3: Microsoft Kinect v2 right–handed coordinate system and its components.

C. Robot Operating System (ROS)

The Robot Operating System (ROS) is a set of software libraries and tools that help one build robot applications. From drivers to state–of–the–art algorithms, ROS has capabilities for different tasks with the power of peer–to–peer communication. And it's all open source [25], [26]. In this project, Kinetic

Kame version of ROS is used with Ubuntu 16.04 Long Term Support (LTS) installed computer.

D. Libraries and tools of system

In this particular research case following libraries and tools are used.

libfreenect2 is an open source driver for Kinect v2 devices created by the OpenKinect community. Its features are color image processing, IR image and depth image processing, registration of color and depth images [27], [28]. *IAI Kinect2* is a collection of tools and libraries for a ROS Interface to the Kinect v2 [29].

MoveIt! Motion Planner is state–of–the–art software for mobile manipulation and it provides the latest advances in motion planning, manipulation, 3D perception, robot kinematics and control. There are several planners such as Open Motion Planning Library (OMPL), Stochastic Trajectory Optimization for Motion Planning (STOMP), Search-Based Planning Library (SBPL) and Covariant Hamiltonian Optimization for Motion Planning (CHOMP). More details can be found in [30]. It also provides an easy–to–use platform for developing advanced robotics applications, evaluating new robot designs and building integrated robotics products for industrial, commercial, R&D and other domains [31]. Due to its strength and simplicity, the MoveIt! motion planner is used in this project. Its overall structure can be seen in Fig. 4.

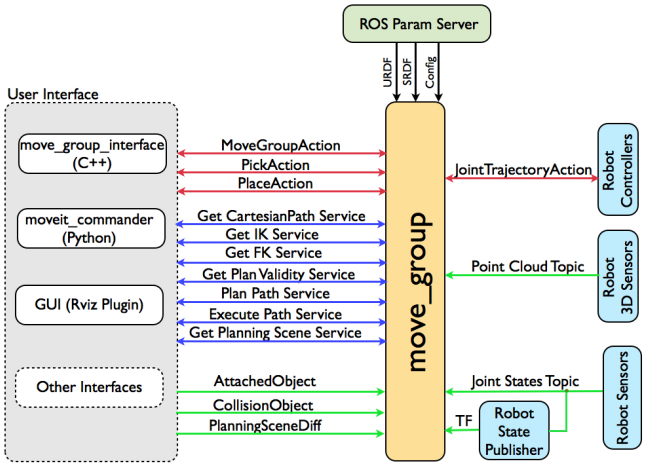


Fig. 4: MoveIt! Structure adapted from [30].

Gazebo Simulation Tool (GST) is a well-designed simulator that makes it possible to test algorithms rapidly, design robots, perform regression testing, and train Artificial Intelligence (AI) system using realistic scenarios [32]. It also offers to simulate populations of robots in indoor and outdoor environments accurately and efficiently. It has a robust engine, high-quality graphics and easy to use graphical interfaces.

Point Cloud Library (PCL): A point cloud is a data structure used to represent a collection of multi-dimensional points. It is commonly used to represent three-dimensional data. In a 3D point cloud, the points usually represent the X, Y, and Z geometric coordinates. When color information is present, the point cloud becomes 4D [33]. Point clouds can be acquired from hardware sensors such as stereo cameras, 3D scanners, or Time-of-Flight (TOF) cameras [34]. PCL also supports the 3D interfaces, therefore, it can acquire and process data from devices such as the PrimeSensor 3D cameras, the Microsoft Kinect, or the Asus XTionPRO.

IV. Methods

In this study, following methods and tools are used.

A. Data Acquisition

Data are acquired from Kinect using PCL, *libfreenect2* and *IAI Kinect2* tools. Data are points representation of our environment. Thus, this point representation enables efficient algorithms and filters to be applied to our system.

B. Euclidean Distance

The Euclidean distance or Euclidean metric is the "ordinary" straight–line distance between two points in cartesian space. With this distance, Euclidean space becomes a metric space. The associated norm is called the Euclidean norm. A generalized term for the Euclidean norm is the L^2 norm or L^2 distance. In general, for an n–dimensional space, the distance can be generalized in Eq.(1). p and q represent two points in cartesian space.

$$d(p,q) = \sqrt{\begin{array}{l}(p_1-q_2)^2+(p_1-q_2)^2+\ldots \\ +(p_i-q_i)^2+\ldots+(p_n-q_n)^2\end{array}} \quad (1)$$

Euclidean distance is used in the PCL point representation of our environment. Points has X,Y and Z dimensions. In order to calculate the Euclidean distance, Eigen tool is used for matrix calculation [35].

C. Voxel Grid Filter

A voxel grid is as a set of tiny 3D boxes in space. Voxel Grid (VG) filter aims to downsample the point cloud. In each voxel (3D box), all the points downsampled with their centroid [33]. C++ implementation is shown in Algorithm 1.

Algorithm 1 Voxel Grid Filter

```
1: pcl::VoxelGrid <pcl::PointXYZRGB>vg;
2: vg.setInputCloud (input_cloud);
3: vg.setLeafSize(0.1f, 0.1f, 0.1f);
4: vg.filter (filtered_cloud);
```

D. Radius Outlier Removal Filter

Radius Outlier Removal (ROR) filter removes all indices in its input cloud that do not have at least some number of neighbors within a certain range [33]. C++ implementation also be shown in Algorithm 2.

Algorithm 2 Radius Outlier Removal Filter

```
pcl::RadiusOutlierRemoval <pcl::PointXYZRGB>ror;
ror.setInputCloud (input_cloud);
ror.setRadiusSearch (0.5);
ror.setMinNeighborsInRadius (80);
ror.filter (filtered_cloud);
```

E. Statistical Outlier Removal

Statistical Outlier Removal (SOR) filter is based on the computation of the distribution of point to neighbors distances in the input cloud. For each point, the mean distance from all its neighbors is being computed. By assuming that the resulted distribution is Gaussian with a mean and a standard deviation, all points whose mean distances are outside an interval defined by the global distances mean and standard deviation can be considered as outliers and trimmed from the dataset [33]. Algorithm 3 represents C++ implementation of the filter.

Algorithm 3 Statistical Outlier Removal

```
pcl::StatisticalOutlierRemoval <pcl::PointXYZRGB>sor;
sor.setInputCloud (input_cloud);
sor.setMeanK (25);
sor.setStddevMulThresh (1.0);
sor.filter (filtered_cloud);
```

V. Experimental Results

A. Mapping

Cartesian coordinates need to be mapped as shown in Fig. 5. According to the setting of our system, no change is made in the X–axis. The Kinect's Y–axis is mapped to the Z–axis of UR5 and finally the Z–axis of Kinect is mapped inversely to the Y–axis of UR5.

B. Pre-Processing of Point Cloud

The point cloud is processed using VG, ROR and SOR filters. Performance of filters based on processing time can be shown in Fig. 6. According to the results, although VG has best execution time, it is not enough to use only it due to the outliers. ROR or SOR filters have to be used to estimate the nearest point accurately. Our system has got 200 thousand points approximately. Fig. 6 also gave information about the latency.

C. Results and Discussion

In order to test our system under difficult conditions, a test trajectory is created with the help of teach pendant of UR5 that is shown in Fig. 7. Kinect turned to the robot and the test trajectory is run. Following that the depth data are saved during the first cycle of the trajectory. The robot is brought to the starting position. This time, the system is run with the recorded data and the movement of the robot is observed. The main idea of the test is that how our algorithm follows the exact trajectory accurately. Firstly, classical cartesian method which only uses

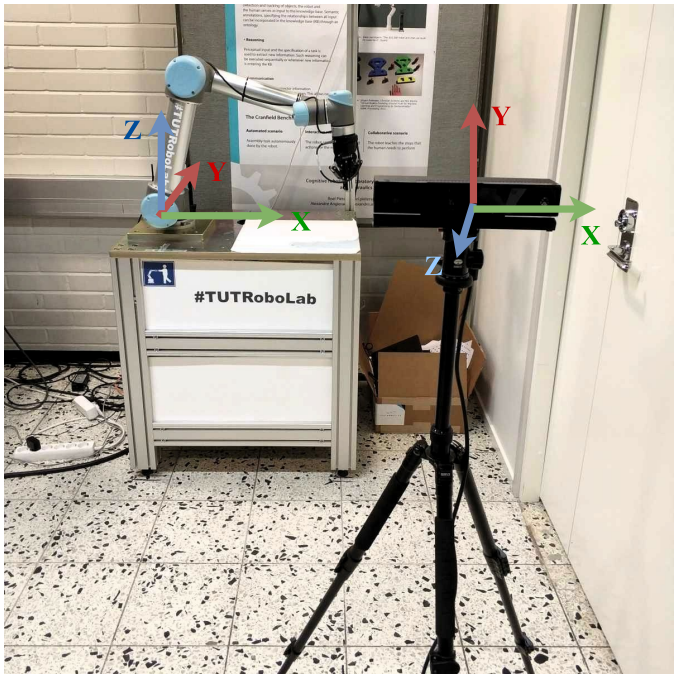


Fig. 5: Environment of system.

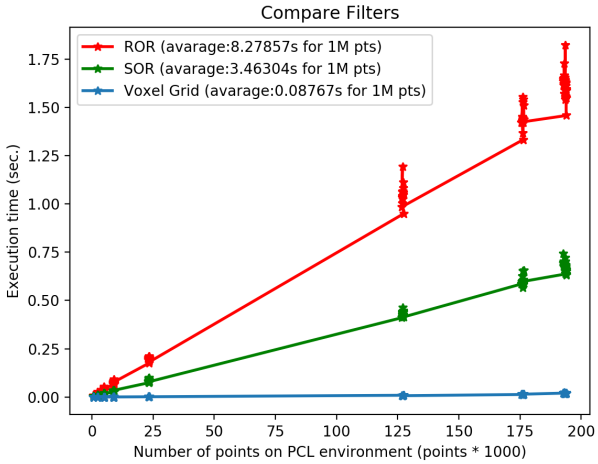


Fig. 6: Compare of three of filters such as Voxel Grid, Radius Outlier Removal and Statistical Outlier Removal filters based on execution time in second. And average execution time for a million points can be seen, too.

position is tested. Next, time parameterization method which uses position, velocity and acceleration is tested, too. The difference between trajectories for cartesian method is shown in Fig. 8 and for time parameterization is shown in Fig. 9. Accuracy according to the cartesian method can be seen in Tab. I and accuracy according to the time parameterization can be seen in Tab. II.

In order to reduce pre–processing time, the number of points are reduced. The system has 200 thousand points and only ROR filter is applied to it. Pre-processing time is 1.75 seconds

TABLE I: ACCURACY OF UR5 USING CARTESIAN METHOD

Coordinates	Accuracy for Cartesian Method
X	85.79%
Y	87.14%
Z	79.63%

TABLE II: ACCURACY OF UR5 USING TIME PARAMETERIZATION METHOD

Coordinates	Accuracy for Time Parameterization Method
X	97.32%
Y	93.34%
Z	91.77%

now. As one can see in Fig. 6, VG filter is quite fast. The points are downsampled to 55–60 thousand with VG filter and now SOR filter is applied following that. The pre–processing time is reduced to 0.14 seconds. As one can see, the VG filter has had no effect on time.

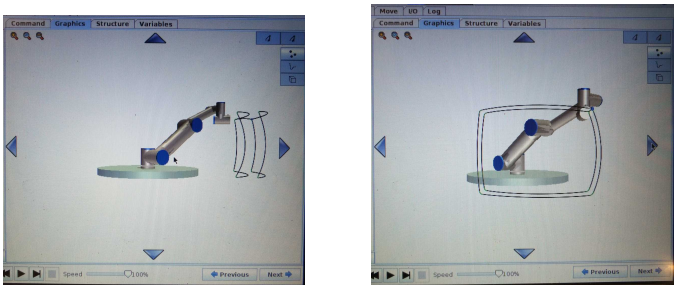
Using these methods, faster response is received. Researchers who want to increase the HRI and trajectory efficiency can develop their systems by using the PCL library and various filters as it is done in this paper.

VI. Conclusion and Future Work

This thesis presented an analysis of possible error sources originating from used technologies, algorithms and deployment methods. The identification error sources and deploying appropriate mitigation procedures are much needed in ensuring safe and seamless human–robot collaboration. Especially, the reduction of latency becomes critical if vision–based safety systems are used for ensuring safety in heavy duty HRI applications in manufacturing industry.

This work demonstrates that latency affects efficiency and seamless cooperation directly in real–time HRI experimental work. In addition to this, it is essential to ensure safe interaction through planned trajectory. From these two main HRI metrics, an interaction study is conducted with existing softwares and libraries. As an use case scenario, a pick and place task that includes smooth trajectory can be easily programmed by an inexperienced operator with further improvements.

Finally, the limitations can be listed as follows. ROS only works on Linux based operating systems, so the built-in functions of Kinect can not be used and the external library must be used to get the data (e.g., *libfreenect2* and *IAI Kinect2* for this project). Next, when the downsampling is done with the VG filter, the resolution is lost in point cloud and the point is calculated incorrectly if the points go below 60K. On the other hand, in order to use built–in filters in PCL, only C++ development and programming can be used at this time. Moreover, in MoveIt!, since the search–based calculation method is used, milliseconds can not be reached for the trajectory planning.



(a) View of the test script which used for implementation

(b) View of the test script from a difference angle

Fig. 7: The test trajectory of UR5

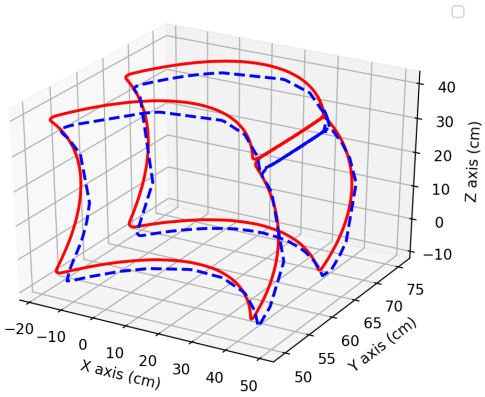


Fig. 8: Cartesian method. Red straight line represents the trajectory which robot should follow and blue dash line represents the trajectory which robot actual follow.

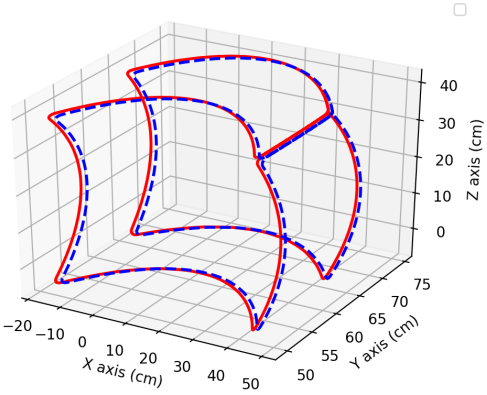


Fig. 9: Time parameterization method. Red straight line represents the trajectory which robot should follow and blue dash line represents the trajectory which robot actual follow.

As a future work, one can focus on closed form trajectory planning method to reduce the trajectory calculation time in milliseconds order. Besides, the efficiency can be enhanced by using different kind of filters in the PCL library and testing on various complex robot trajectories.

ACKNOWLEDGMENTS

Authors would like to thank Jane & Aatos Erkko Foundation and Technology Industries of Finland Centennial Foundation for the support of UNITY (2016-2019) project.

REFERENCES

[1] R. R. Murphy and D. Schreckenghost, "Survey of metrics for human-robot interaction," in *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, March 2013, pp. 197–198.

[2] A. De Luca and F. Flacco, "Integrated control for phri: Collision avoidance, detection, reaction and collaboration," in *Biomedical Robotics and Biomechatronics (BioRob), 2012 4th IEEE RAS & EMBS International Conference on*. IEEE, 2012, pp. 288–295.

[3] A. De Santis, B. Siciliano, A. De Luca, and A. Bicchi, "An atlas of physical human–robot interaction," *Mechanism and Machine Theory*, vol. 43, no. 3, pp. 253–270, 2008.

[4] J. Saarinen, J. Suomela, and A. Halme, "The concept of future worksite–towards teamwork-centered field robotic systems," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 14 952–14 957, 2011.

[5] M. R. Walter, M. Antone, E. Chuangsuwanich, A. Correa, R. Davis, L. Fletcher, E. Frazzoli, Y. Friedman, J. Glass, J. P. How *et al.*, "A situationally aware voice-commandable robotic forklift working alongside people in unstructured outdoor environments," *Journal of Field Robotics*, vol. 32, no. 4, pp. 590–628, 2015.

[6] J. Suarez and R. R. Murphy, "Hand gesture recognition with depth images: A review," in *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*, Sept 2012, pp. 411–417.

[7] J. L. Raheja, A. Chaudhary, and K. Singal, "Tracking of fingertips and centers of palm using kinect," in *Computational intelligence, modelling and simulation (CIMSiM), 2011 third international conference on*. IEEE, 2011, pp. 248–252.

[8] Z. Ren, J. Meng, and J. Yuan, "Depth camera based hand gesture recognition and its applications in human-computer-interaction," in *2011 8th International Conference on Information, Communications Signal Processing*, Dec 2011, pp. 1–5.

[9] Y. Li, "Hand gesture recognition using kinect," in *2012 IEEE International Conference on Computer Science and Automation Engineering*, June 2012, pp. 196–199.

[10] A. Kurakin, Z. Zhang, and Z. Liu, "A real time system for dynamic hand gesture recognition with a depth sensor," in *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, Aug 2012, pp. 1975–1979.

[11] S. Moe and I. Schjlberg, "Real-time hand guiding of industrial manipulator in 5 dof using microsoft kinect and accelerometer," in *2013 IEEE RO-MAN*, Aug 2013, pp. 644–649.

[12] D. G. Schneider, L. L. d. Silva, P. Diehl, A. H. R. Leite, and G. S. Bastos, "Robot navigation by gesture recognition with ros and kinect," in *2015 12th Latin American Robotics Symposium and 2015 3rd Brazilian Symposium on Robotics (LARS-SBR)*, Oct 2015, pp. 145–150.

[13] N. B. Kumbla, J. A. Marvel, and S. K. Gupta, "Using sensor feedback to accurately estimate part pose in a gripper."

[14] D. Shin, S. Shin, D. Kim, S. Kim, J. Hwang, and Y. Kim, "Visual guidance system for remote-operation," in *2016 13th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, Aug 2016, pp. 657–661.

[15] R. Raja and S. Kumar, "A hybrid image based visual servoing for a manipulator using kinect," in *Proceedings of the Advances in Robotics*. ACM, 2017, p. 52.

[16] J. Mieikis, K. Glette, O. J. Elle, and J. Torresen, "Multi 3d camera mapping for predictive and reflexive robot manipulator trajectory estimation," in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, Dec 2016, pp. 1–8.

[17] Y. Wang, Y. Sheng, J. Wang, and W. Zhang, "Optimal collision-free robot trajectory generation based on time series prediction of human motion," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 226–233, Jan 2018.

[18] *Robots and robotic devices – Collaborative robots*, ISO/TS Std. 15 006, 2016.

[19] *Robots and robotic devices – Safety requirements for industrial robots*, ISO Std. 10 218, 2011.

[20] K. Mathiassen *et al.*, "An ultrasound robotic system using the commercial robot ur5," *Frontiers in Robotics and AI*, vol. 3, p. 1, 2016. [Online]. Available: https://www.frontiersin.org/article/10.3389/frobt.2016.00001

[21] K. Kufieta, "Force estimation in robotic manipulators: Modeling, simulation and experiments," M. Eng. thesis, Norwegian University of Science and Technology, Trondheim, Norway, Jan. 2014.

[22] S. B. Gokturk, H. Yalcin, and C. Bamji, "A time-of-flight depth sensor - system description, issues and solutions," in *2004 Conference on Computer Vision and Pattern Recognition Workshop*, Jun. 2004, pp. 35–35.

[23] Z. Zhang, "Microsoft kinect sensor and its effect," *IEEE MultiMedia*, vol. 19, no. 2, pp. 4–10, Feb. 2012.

[24] J. Han, L. Shao, D. Xu, and J. Shotton, "Enhanced computer vision with microsoft kinect sensor: A review," *IEEE Transactions on Cybernetics*, vol. 43, no. 5, pp. 1318–1334, Oct. 2013.

[25] M. Quigley, J. Faust, T. Foote, and J. Leibs, "Ros: an open-source robot operating system."

[26] "Robot operating system ROS," http://www.ros.org/, Open Source Robotics Foundation OSRF, accessed: 29-01-2018.

[27] F. J. Lawin, P.-E. Forssén, and H. Ovrén, "Efficient multi-frequency phase unwrapping using kernel density estimation," in *European Conference on Computer Vision (ECCV)*. Amsterdam: Springer International Publishing AG, October 2016, vR Projects: Learnable Camera Motion Models, 2014-5928, Energy Models for Computational Cameras, 2014-6227.

[28] L. Xiang, F. Echtler, C. Kerl, T. Wiedemeyer, Lars, hanyazou, R. Gordon, F. Facioni, laborer2008, R. Wareham, M. Goldhoorn, alberth, gaborpapp, S. Fuchs, jmtatsch, J. Blake, Federico, H. Jungkurth, Y. Mingze, vinouz, D. Coleman, B. Burns, R. Rawat, S. Mokhov, P. Reynolds, P. Viau, M. Fraissinet-Tachet, Ludique, J. Billingham, and Alistair, "libfreenect2: Release 0.2," Apr. 2016. [Online]. Available: https://doi.org/10.5281/zenodo.50641

[29] T. Wiedemeyer, "IAI Kinect2," https://github.com/code-iai/iai_kinect2, Institute for Artificial Intelligence, University Bremen, 2014 – 2015, accessed June 12, 2015.

[30] "Moveit! motion planning framework," http://moveit.ros.org/, Open Source Robotics Foundation OSRF, accessed: 29-01-2018.

[31] S. Chitta, I. Sucan, and S. Cousins, "Moveit![ros topics]," *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.

[32] "Gazebo robot simulation tool," http://gazebosim.org/, Open Source Robotics Foundation OSRF, accessed: 29-01-2018.

[33] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[34] A. Aldoma, Z. C. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. B. Rusu, S. Gedikli, and M. Vincze, "Tutorial: Point cloud library: Three-dimensional object recognition and 6 dof pose estimation," *IEEE Robotics Automation Magazine*, vol. 19, no. 3, pp. 80–91, Sept 2012.

[35] G. Guennebaud, B. Jacob *et al.*, "Eigen v3," http://eigen.tuxfamily.org, 2010.