DOI: 10.3217/978-3-85125-663-5-06

An Autonomous Mobile Handling Robot Using Object Recognition

Johannes N. Rauer[1]; Wilfried Wöber[1] and Mohamed Aburaia[1]

Abstract— **Due to the trend away from mass production to highly customized goods, there is a great demand for versatile robots in the manufacturing industry. Classic fixed-programmed industrial robots and rail-bound transport vehicles, which are restricted to transporting standardized boxes, do not offer enough flexibility for modern factories. Machine learning methods and 3D vision can give manipulators the ability to perceive and understand the environment and therefore enable them to perform object manipulation tasks. State of the art grasp-detection methods rely on data with cumbersome annotated grasp-poses, while labelled data for object recognition only is easier to gather.**

This work describes the development of an automatic transport robot using a sensitive manipulator and 3D vision for autonomous transport of objects. This mobile manipulator is able to drive flexible paths, localize predefined objects and grasp them using an out-of-the-box neural network for object detection and hand-crafted methods for extracting grasp-points from depth images to avoid cumbersome grasp-point-annotated training data. Furthermore, this paper discusses problems occurring when a neural network trained on human-captured photos is applied to robot-view images.

I. Introduction

Since the beginning of the 1960s, traditional fixed-base robots have been used in factories [40]. Classic industrial robots are unable to deal with the uncertainties in the real world, due to a lack of sensing capabilities. Therefore, stationary manipulators have mainly been deployed in mass production, where a constant environment can be assumed and thus highly engineered programs can work efficiently [33], [35]. Giving manipulators the ability to perceive the environment and using these perceptions for machine learning, flexible part recognition in unstructured environments is possible [8]. Due to the trend away from mass production to highly customized goods, there is a great demand for versatile robots in the manufacturing industry [17], [16]. Over the last few years e.g. the production of cars has been highly individualized. As Pavlichenko et al. [27] describe, this has made "kitting" necessary, where all parts of a car are collected in a warehouse and brought to the assembly line just in time as a "kit". This task is frequently performed by warehouseman due to the high flexibility needed to find, collect and transport specific components [27]. Therefore, part handling during assembly is the only task in the automotive industry with an automation level below 30% [17].

For such intra-logistical transport tasks, mobile manipulators have been developed. They consist of sensitive manipulators which are associated with mobile transport vehicles to combine the advantages of both types of robots: Working together with humans and grasping individual objects, as well as changing the location autonomously to extend the workspace of the manipulator [16], [7]. A mobile manipulator should be able to drive autonomously through a shop floor, detect parts with sensors, grasp them with its manipulator and transport them to production facilities [27].

While there already exist well established methods for path planning and safe navigation of mobile robots, grasping objects flexibly remains a challenging task [25], [4]. Since a large amount of grasp-pose-annotated training data is needed for automatic grasp-detection in machine learning, training new objects is complex and expensive [28].

Therefore, this paper describes the development of a mobile manipulator using an out-of-the-box object recognition framework and handcrafted grasp-point calculation algorithms to allow grasping of standard objects with pre-trained algorithms and not requiring cumbersome annotated grasp-data. It describes a software concept for a mobile platform extended with a sensitive manipulator, a gripper and a 3D sensor. This system enables path planning, searching for predefined objects using an image stream and machine learning, as well as grasping and transporting them. Since the mobile robot's integrated path planner is being used, this part is not considered in more detail. Furthermore, this paper discusses problems occurring when a neural network for object detection trained on human-captured photos is used with robot-view images.

The following chapters are structured as follows: Section II presents the state of the art of 3D-vision-systems, grasp detection and object detection in machine learning with a focus on deep convolutional neural networks. In section III, the software representation of the robot and the implemented software are described, as well as the hardware and structure of the mobile manipulator. Section IV explains the abilities of the robot and verifies and discusses them using experiments, followed by summary and future work in section V.

II. State of the Art

In the following section, the state of the art of 3D-vision-systems, grasp detection and machine learning is discussed.

A. 3D Vision Systems for Robotic Manipulation

Visual sensors play an important role in object detection and manipulation. 3D vision systems, which map the 2D image pixels to 3D world coordinates, are most suitable for this task, if they are mounted to the arm of a robot [42], [2].

According to Giancola et al. [13], there are three main 3D vision technologies on the market which are suitable for such

[1]Department of industrial engineering, University of Applied Sciences Technikum Wien, 1200 Wien, Austria <mr17m019,woeber,aburaia>@technikum-wien.at

an application: Time-of-Flight (TOF), structured-light and active stereoscopy, whereby the last two use a triangulation process to estimate the depth. TOF-systems measure depth-distance directly from the time a light beam needs until it is reflected back [13]. Active stereoscopy devices have the advantage over classical stereoscopic depth systems, that they project an infrared pattern and are therefore able to find matching points for the triangulation also on textureless surfaces [19]. 3D cameras often include an RGB-sensor and therefore provide RGB-D images [13].

B. *Machine Learning for Object Detection*

To extract information such as object types and positions from image data, neural networks are used [6], [1]. Deep neural networks [15] consist of numerous layers with artificial neurons, which allow them to handle big data effectively. A convolutional neural network (CNN) [20] is a special type of deep network, which is designed to work with data with a grid-like topology, such as images [1], [15]. An example CNN is presented in Fig. 1. It consists of a series of convolutional and pooling layers which act as feature extractors or filters and fully connected layers for classification or regression [1], [15].

To predict besides class probabilities also bounding boxes for objects in a single-shot, Redmon et al. have presented the CNN YOLOnet [31]. Classic approaches [1], [15] work with a sliding window where a classifier is run over the complete image. More recent R-CNN methods first propose regions where objects may be and then run a classifier on these bounding boxes [14], [9], but due to the high number of individual components, these networks are slower and hard to optimize [31]. YOLOv3 [32] can process images at up to 45 fps to allow fast reaction to changes in the environment which is particularly important for object manipulation [30].

C. *Grasp Detection*

The goal of robotic grasp detection is to detect and calculate graspable regions in images of objects and compute a trajectory to them [21], [41], [5]. Due to challenging variations of the objects and light conditions, as well as occlusions and clutter, grasp-detection systems lag far behind human performance levels [4].

Some grasp-detection methods focus on the object pose estimation and lookup of the corresponding grasp points or gripper pose in a database [41], [24], [5]. At other approaches, neural networks are trained to provide the full grasp configuration [30], [39] or the success-probability for a grasp in a given gripper pose [28], [22]. Due to the creation of an appropriate training dataset being currently a big obstacle, numerous approaches generate data automatically using robots or simulations [28], [24], [39]. Labeling data manually is a challenging and time-consuming task due to objects being graspable in multiple ways and the necessary label being of a high dimension and gripper-dependent [28].

Therefore, in this work grasp points are estimated using an out-of-the-box object detection network and hand-crafted, object dependent methods which extract features from depth-data. This approach does only require training data for object detection, which is available in large number [10].

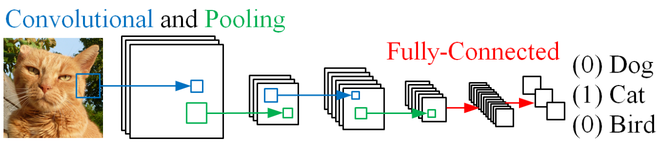

Fig. 1. Typical structure of a CNN: Blue - Convolutional Layer, Green - Pooling Layer, Red - Fully-Connected Layer

III. Methods and Implementation

In this section, the representation of the mobile manipulator and its environment with coordinate frames is discussed and the structure of the robot as well as all components are introduced. Furthermore, the implementation of the robot's software is explained.

A. *Robot Representation*

The presented mobile manipulator consist of a MiR100 mobile platform [26] and an UR5 articulated robot [37], as well as a gripper and a depth camera (see III-C). All components are represented using coordinate frames which are connected with each other and with the world-frame via geometrical relations (see Fig. 2).

The *map*-frame represents the robot's world coordinate system and functions thereby as fixed origin. As the robot moves, the *odometry*-frame is moved according to the wheel-odometry and signals from the inertial measurement unit continuously in the *map*-frame. The *base*-frame is rigidly attached to the mobile platform and moved in the *map*-frame according to sensor observations, such as the localization in the map using laser-scanner signals [23]. The *base-UR*-frame is attached to the *base*-frame statically and serves as reference for the frames in the joints of the manipulator. At the end of the kinematic chain, a frame for the tool center point of the gripper – in which planning of arm trajectories takes place – as well as a frame for the camera – in which the pose of the object is calculated – are connected statically to the *EEF*-frame (end-effector-frame).

If an object is detected by the robot, its xyz-position relatively to the camera is calculated. With the knowledge of the transformations between all the other frames, the pose of the object in the robot's world is computable. The transformations between the *object*-frame and the *EEF*-frame are determined by calibrating the camera.

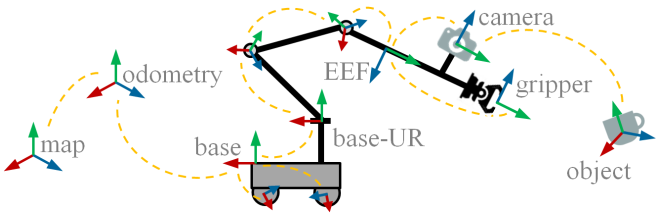

Fig. 2. Positions and connections of the robot's frames. Orientations are exemplary. To reduce complexity, unimportant frames are not shown or labeled. The dashed lines illustrate the mutual dependencies of the frames.

B. *Implementation of the Software*

The software of the mobile manipulator is based on ROS (Robot operating system) [29]. The structure of the basic ROS-nodes, which provide communication to the hardware and the ROS-Core, is presented in Fig. 3. When the task-execution is started, the connections to the basic nodes are established as illustrated. The program flow of the task-execution is presented in Fig. 4 and explained in III-B.2. The basic nodes are described in the following section:

1) Basic Nodes: The basic nodes are responsible for providing communication with the hardware and elementary information of the robot, such as its kinematic structure. Therefore, a robot model including its structure and the coordinate frames is sent to the core by the *Model-Uploader*. The *ROS-Core* provides it to all nodes to enable visualizations and calculation of trajectories.

Using this robot-description, tf [12] builds a tree as illustrated in Fig. 2. The module *TF-Transformer* serves as interface between the *Task-Execution-Software* and tf-tree, providing necessary transformations for robot movements. A TCP-IP connection to the UR5 is established using *UR-Driver* [3]. This node provides the actual robot pose and is able to send trajectories to the robot. The ROS-package *MoveIt!* [36] communicates with the *UR-Driver* and serves as motion planner and kinematics solver for the UR-robot. It offers a programming interface to get the robot's joint states and arm pose. Given a target pose it solves the inverse kinematics and calculates a trajectory.

The communication between the industrial PC and the MiR's controller is established using *MiR-Driver* [11]. Since the MiR launches its own ROS-Core, this bridge establishes a connection to it, reads all topics and provides the same topics in the local ROS-environment on the industrial computer. It reads all messages of the MiR and publishes them to the local ROS-Core – e.g. the actual pose. If e.g. a goal pose is sent to a MiR-topic, it is forwarded to the MiR and published to its ROS-Core. Path planning and safe movement is implemented by the manufacturer and therefore not discussed further.

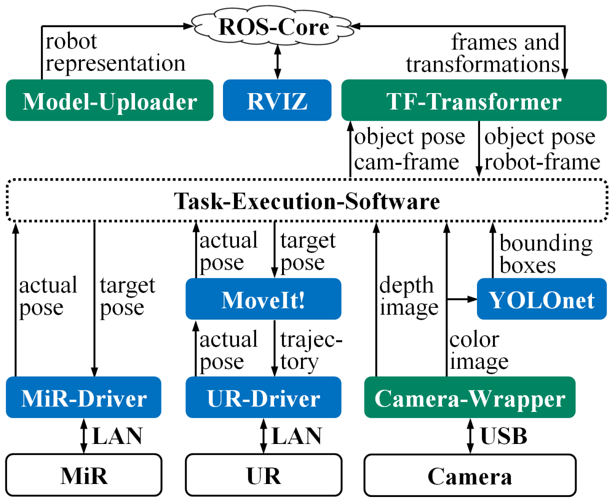

Fig. 3. Structure of the basic nodes of the software, which provide topics to communicate with the hardware. Green nodes have been developed. Blue nodes symbolize provided ROS-Packages.

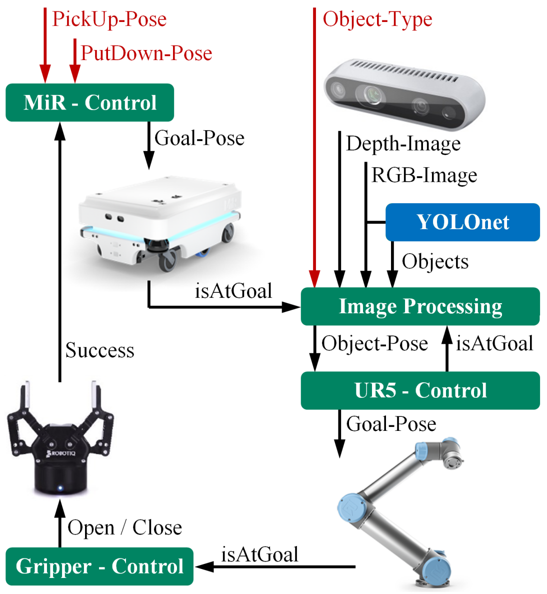

Fig. 4. Program flow of the robot's task-execution-software. Controls all components to grasp an object of given type at given pick-up-position and to transport it to given put-down-position. User inputs are marked red. Green nodes have been developed. Blue nodes symbolize provided ROS-Packages.

In addition to driving and moving the robotic arm, perceiving the environment is necessary to enable the mobile manipulator to grasp objects. Therefore, *Camera-Wrapper* connects to the RGB-D-camera and publishes color and depth images. The CNN *YOLOnet* subscribes to the color-image-stream and publishes the object-types and -positions in the image frames as bounding boxes [32].

With the discussed basic services and communications, the actual task execution of driving, searching, locating and grasping the object is possible, which is explained in the following section:

2) Task execution: The program which is necessary for executing the task itself is started by user-input and calls different modules with special purposes. This enables to change modules and implement e.g. a different grasping algorithm with no effect to other program parts. In Fig. 4, the general procedure for grasping an object is presented abstractly and following the parts are described in detail.

A complete task-execution consist of driving to a goal, searching for the object and calculating the grasp point, as well as moving the robotic arm and grasping the object at the calculated point. Then the robot can move to a goal position and put the object down. For sending the robot to a target pose, the module *MiR-Control* builds a ROS-Message from a given PickUp-Pose, that is forwarded to the robot. The MiR-robot calculates its path and drives to the goal without further intervention, as it is able to perform localization, path planning and obstacle detection autonomously [25].

When the MiR is at the goal, the pose of the object has to be determined. The camera's RGB-image is analyzed by *YOLOnet*, which offers the object-types and positions of the bounding boxes for each recognized object in the image [32]. Together with the depth-image and the camera intrinsic parameters, the image-processing-module calculates the coordinates of the demanded object in 3D-coordinates relatively to the camera. As the pose estimation is not precisely when analyzing a single shot, the UR is moved multiple times and different image processing steps are performed: First the UR is moved to search positions where the camera has a good overview of the scene to search for the object. If it is detected, the robot moves about 35 cm over the item with the camera parallel to the plane under the object for grasp detection. It is not possible to only scan the table from this planar position, since object detection does not work for certain poses, as described in IV-B.

Grasp detection is performed for each object individually and the processes are designed manually. To detect grasping points, objects are separated from background using the depth-image and circle detection is performed (see Fig. 5). Bottles can be grasped using the estimated circle's center. The grasp point of cups is calculated using the center of the circle and the end of the handle, which is the furthest point from the center. Using the positions of these points with regard to the camera, the necessary grasp angle is calculable. These grasp points are provided to the tf-tree to calculate the grasp point relatively to the robot's frame. Using this transformation, *UR5-Control* calculates the goal pose of the UR5 to grasp the object.

After the UR has reached the aimed grasping pose, *Gripper-Control* sends a command to the robot to close the gripper. The gripper is connected to the UR to make it controllable with the robot's teach pendant. Therefore, commands have to be sent to the gripper via the UR's LAN-connection. If the object has successfully been grasped by the robot, the MiR is sent to the PutDown-Pose where the UR5 can lie down the object.

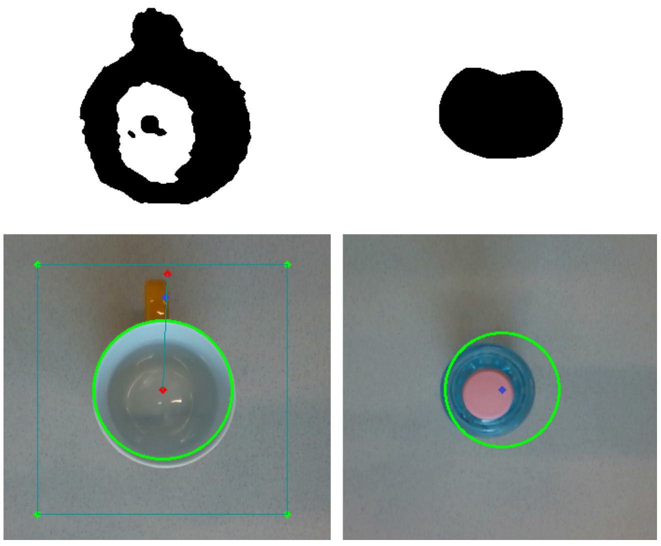

Fig. 5. Detection of grasp point (blue). Top: Depth-images separated from background. Left: Grasp-Point located at the handle of a cup. Right: Grasp-Point at the center of a bottle.

C. *Components & Structure of the Robot*

The mobile platform which serves as base for all components is a MiR100 [26]. It is able to calculate and travel the path to a given goal autonomously and is thereby aware of safety issues [25]. On top of an aluminum structure, a collaborative articulated UR5 robot [37] is installed. This manipulator weights about 20 kg at a payload of 5 kg and a maximum grasp-distance of 850 mm. It is connected to the MiR's ethernet-switch via LAN.

A 3D-printed fixture with an Intel Realsense D435 RGB-D-camera [18] is mounted onto the flange of the manipulator. This active stereoscopy based camera provides RGB-D-images between 0.2 and 4.5 meters in depth at up to 90 frames per second. It is connected via USB-C to the industrial computer. The parallel-jaw gripper 2F-85 [34] is mounted onto the camera fixture at the end of the kinematic chain of the robot. With its payload of 5 kg at an opening width of 85 mm it is suitable to grasp small objects. It is connected to the control unit of the UR5 via USB.

To control all subcomponents of the mobile manipulator, the industrial computer ECS-9100-GTX1050T [38] is installed on the mobile robot. With its NVIDIA GeForce GTX 1050 Ti graphics processor it is suitable for image processing tasks. The industrial computer runs on Ubuntu 16.04 and hosts a ROS-Kinetic-Core. It is connected to the ethernet-switch of the MiR and thereby able to access and control all components.

IV. Results & Discussion

In the following section, the abilities of the robot are described and verified using experiments, focusing on the quality of object detection. Furthermore, reasons for problems of the object recognition are presented and discussed.

A. *Abilities of the Robot & Success Rate*

The presented mobile manipulator is able to localize and classify predefined objects, to perform a path and movement planning, to drive to the object, and to grasp and transport it as well as put it down. The current supported objects are cups and bowls, whereby a software platform has been created which can be extended to manipulate other objects.

A qualitative experiment has been performed to evaluate the robot's success rate: the mobile manipulator is sent to the same goal multiple times, where one of the objects is randomly placed. The robot is able to grasp the object successfully in each of 30 repetitions, although the robot's pose varies up to approximately 25 cm and 20°.

Separation of object detection and grasp-point calculation enables simple and fast implementation of grasp-control for standard objects using a pre-trained network without the need of cumbersome training-data generation and annotation. It has to be noted that new methods must be hand-crafted for additional objects and grasp-point estimation from depth-data as presented is error-prone if multiple objects are visible. Therefore, if to some extend constant ambient conditions can not be ensured, switching to an integrated solution with more

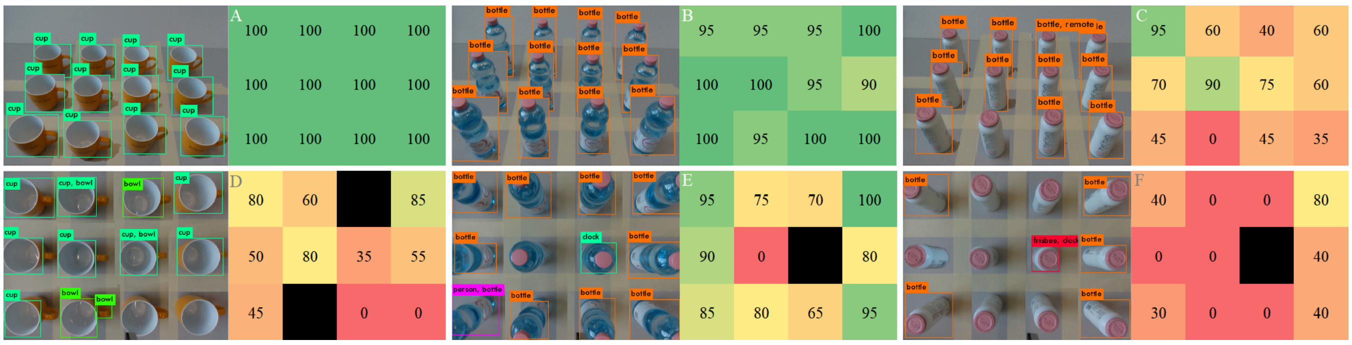

Fig. 6. Quality of object detection: Cup (left), Standard water bottle (center), Unconventional smoothie bottle (right). Camera has been oriented 40° (top) and 90° (bottom). Images show the merged pictures of all positions. Colored fields illustrate the calculated probability of the object being of the correct class. Black fields indicate wrong detections.

complex training data such as PoseCNN [41] or Silhonet [5] might be preferable.

B. Object Recognition

An object can only be grasped if it is recognized correctly first. Since the hand-crafted methods for grasp-point calculations need a planar view onto the object, scanning a surface for objects with the camera parallel to it would be most appropriate. This could save time since only one pose before grasping the object would be possible. The quality of the object detection, depending on the angle of the camera and the position of the object in the captured image has therefore been evaluated practically using experiments. For this, the robot has been located in front of a table showing a 4x3 grid under constant lighting conditions. The camera has been oriented by manipulating the robot's arm to show the whole grid at a specific angle. Afterwards, an object has been presented to the camera at the different positions of the grid under constant orientation. It has been evaluated if the object is recognized in the captured image and the quality of this detection has been analyzed using the mean class-probabilities provided by the neural network. This procedure has been performed at camera angles of 40° and 90° at a distance of 750 and 600 mm to the grid's center respectively, using three different objects: A cup with great contrast, a standard water bottle and a particular smoothie bottle. The objects have been oriented with the cup's handle pointing 90° to the right and the bottle's labels pointing to the robot.

The results of these experiments are presented in Fig. 6. The color of the fields indicate the probability of an object being of the correct class, whereby red marks positions where no class could be determined and black illustrates detections of a wrong class. The detection rate of cups and standard water bottles from a side view are very high, as Fig. 6 (A, B) shows. Unusual bottles as captured in (C) are less likely to be recognized correctly. The experiments show significant lower success rates of the object recognition for top view images. Cups are detected less accurate at all locations, whereby especially the lowest positions cause problems. Detecting cups incorrectly as bowls is also a common mistake (D). The detection rate of water bottles from the top view (E) is also smaller than from the side, whereby no correct detection of bottles shown from exactly above at the center positions are observable. This is visible for the smoothie bottles too (F). It is also interesting to note that bottles in the top view images (E, F) at a specific location have always been identified as clocks with a probability of 50%.

These experiments show clear weaknesses of the used training data for a robotic application. The ImageNet dataset [10] which has been used to train YOLOnet is using images found on the Internet. Therefore most of the objects are captured in their natural position as humans see them – e.g. most of the images of cups show medium-sized ceramic mugs with the handle clearly visible from an angle of about 45°. There are only few images showing cups in aerial perspective, from a flat angle or upside-down. The neural network learns from this insufficient dataset and thereby struggles to generalize to objects in new or unusual poses or configurations.

The results demonstrate the importance of an accurate and application specific training-dataset for neural networks. To enable high detection rates, images of concrete objects to be recognized from real-life perspectives have to be included in sufficient number.

V. Summary and Future Work

In this paper, an automatic transport robot using a sensitive manipulator and 3D vision sensors for autonomous object-transport has been presented. This mobile manipulator is able to drive autonomously, localize predefined objects and grasp them using an out-of-the-box neural network for object detection and hand-crafted methods for extracting grasp-points from depth images. A modular software based on ROS has been developed, which has the advantage of the basic nodes and hardware communication being decoupled from the actual task execution software.

Experiments have shown that despite inaccuracies of the mobile robot's goal, objects can be grasped successfully if the scene is captured from different perspectives. The main weakness is detecting objects reliable from exactly above, which is the necessary pose for the hand-crafted grasp-point detection to work, due to insufficient training of the neural network.

Therefore, in future work the neural network should be trained using a different dataset which shows objects as the robot's camera captures them. Using a neural network which performs object pose estimation would make detection of grasp points independent of image processing and therefore raise robustness. To enable an easy extendability to detect and grasp new objects, automatic generation of training data using a robot or rendering of objects should be considered.

References

[1] Q. Abbas, M. E. Ibrahim, and M. A. Jaffar, "A comprehensive review of recent advances on deep vision systems," May 2018.

[2] M. M. Ali, H. Liu, R. Stoll, and K. Thurow, "Arm grasping for mobile robot transportation using kinect sensor and kinematic analysis," in *2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings*, May 2015, pp. 516–521.

[3] T. T. Andersen, *Optimizing the Universal Robots ROS driver*. Technical University of Denmark, 2015.

[4] U. Asif, M. Bennamoun, and F. Sohel, "Real-time pose estimation of rigid objects using rgb-d imagery," in *Proceedings of the 2013 IEEE 8th Conference on Industrial Electronics and Applications, ICIEA 2013*. IEEE, June 2013, pp. 1692–1699.

[5] G. Billings and M. Johnson-Roberson, "Silhonet: An rgb method for 3d object pose estimation and grasp planning," *CoRR*, vol. abs/1809.06893, Sept. 2018.

[6] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[7] Z.-E. Chebab, J.-C. Fauroux, N. Bouton, Y. Mezouar, and L. Sabourin, "Autonomous collaborative mobile manipulators : State of the art," in *TrC-IFToMM Symposium on Theory of Machines and Mechanisms*, Izmir, Turkey, 2015.

[8] C. H. Corbato, M. Bharatheesha, J. Van Egmond, J. Ju, and M. Wisse, "Integrating different levels of automation: Lessons from winning the amazon robotics challenge 2016," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 11, pp. 4916–4926, Nov. 2018.

[9] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," in *NIPS*, 2016.

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.

[11] DFKI Robotics Innovation Center. mir_robot. [Online]. Available: http://wiki.ros.org/mir_driver

[12] T. Foote, "tf: The transform library," in *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference*, ser. Open-Source Software workshop, April 2013, pp. 1–6.

[13] S. Giancola, M. Valenti, and R. Sala, "A survey on 3d cameras: Metrological comparison of time-of-flight, structured-light and active stereoscopy technologies," in *SpringerBriefs in Computer Science*, ser. SpringerBriefs in Computer Science. Cham: Springer International Publishing, 2018.

[14] R. Girshick, "Fast r-cnn," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015, pp. 1440–1448.

[15] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[16] H. Hirsch-Kreinsen, "Digitization of industrial work: development paths and prospects," *Journal for Labour Market Research*, vol. 49, no. 1, pp. 1–14, July 2016.

[17] D. Holz and S. Behnke, "Fast edge-based detection and localization of transport boxes and pallets in rgb-d images for mobile robot bin picking," in *Proceedings of ISR 2016: 47st International Symposium on Robotics*, June 2016, pp. 1–8.

[18] Intel Corporation. Intel realsense depth camera d435. [Online]. Available: https://click.intel.com/intelr-realsensetm-depth-camera-d435.html

[19] L. Keselman, J. I. Woodfill, A. Grunnet-Jepsen, and A. Bhowmik, "Intel realsense stereoscopic depth cameras," *Computer Vision and Pattern Recognition*, 2017.

[20] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, Dec. 1989.

[21] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, Jan. 2015.

[22] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," in *Robotics: Science and Systems*, July 2017.

[23] W. Meeussen, "Coordinate frames for mobile platforms," 2010. [Online]. Available: http://www.ros.org/reps/rep-0105.html

[24] C. Mitash, K. E. Bekris, and A. Boularias, "A self-supervised learning system for object detection using physics simulation and multi-view pose estimation," in *IEEE International Conference on Intelligent Robots and Systems*, Vancouver, Canada, Sept. 2017, pp. 545–551.

[25] Mobile Industrial Robots, *MiR100 User Guide*, Mobile Industrial Robots, 2019.

[26] Mobile Industrial Robots ApS. Mir100 - mobile industrial robots. [Online]. Available: http://www.mobile-industrial-robots.com/en/products/mir100

[27] D. Pavlichenko, G. M. García, S. Koo, and S. Behnke, "Kittingbot: A mobile manipulation robot for collaborative kitting in automotive logistics," in *Intelligent Autonomous Systems 15 - Proceedings of the 15th International Conference IAS-15, Baden-Baden, Germany, June 11-15, 2018*, 2018, pp. 849–864.

[28] L. Pinto and A. Gupta, "Supersizing self-supervision : Learning to grasp from 50k tries and 700 robot hours," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2016, pp. 3406–3413.

[29] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *International Conference on Robotics and Automation (ICRA), workshop on open source software*, 2009.

[30] J. Redmon and A. Angelova, "Real-time grasp detection using convolutional neural networks," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 1316–1322.

[31] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2016, pp. 779–788.

[32] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv*, 2018.

[33] M. A. Roa, D. Berenson, and W. Huang, "Mobile manipulation: Toward smart manufacturing [tc spotlight]," *IEEE Robotics Automation Magazine*, vol. 22, no. 4, pp. 14–15, Dec. 2015.

[34] Robotiq. 2f-85 gripper. [Online]. Available: https://robotiq.com/products/2f85-140-adaptive-robot-gripper

[35] J. Sturm, *Approaches to Probabilistic Model Learning for Mobile Manipulation Robots*, ser. Springer Tracts in Advanced Robotics (STAR), B. Siciliano and O. Khatib, Eds. Springer, 2013.

[36] I. A. Sucan and S. Chitta. Moveit! [Online]. Available: http://moveit.ros.org

[37] Universal Robots A/S. Ur5 collaborative robot arm. [Online]. Available: https://www.universal-robots.com/products/ur5-robot

[38] Vecow. Ecs-9200/9100 gtx1050. [Online]. Available: http://www.vecow.com/dispUploadBox/PJ-VECOW/Files/3238.pdf

[39] U. Viereck, A. ten Pas, K. Saenko, and R. Platt, "Learning a visuomotor controller for real world robotic grasping using simulated depth images," *CoRR*, vol. abs/1706.04652, 2017.

[40] J. Wallén, "The history of the industrial robot," *Technical report from Automatic Control at Linköpings universitet*, 2008.

[41] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A convolutional neural network for 6d object pose estimation in cluttered scenes," *Robotics: Science and Systems (RSS)*, 2018.

[42] S. Xie, E. Haemmerle, Y. Cheng, and P. Gamage, "Vision-guided robot control for 3d object recognition and manipulation," in *Robot Manipulators*. InTech, Sept. 2008, pp. 521–546.