A Flexible Robotic Framework for Autonomous Manufacturing Processes: Report from the European Robotics Challenge Stage 1

Arne-Christoph Hildebrandt, Christoph Schuetz, Daniel Wahrmann, Robert Wittmann and Daniel Rixen

***Abstract*— Common industrial automation approaches consist on heavy and fixed robotic manipulators working in separated and closed production lines. Recent advances in sensing and control are leading to flexible and versatile robotic manipulator platforms, which could work in barrier-free production areas and might be the next advance in industrial production. In our contribution, we present a sophisticated and completely autonomous software framework for handling complex pick-and-place tasks using state-of-the-art tools and algorithms. It covers applicable solutions for 3D image processing, motion planning, grasping as well as error handling and task scheduling strategies. Its competitiveness in terms of robustness and performance has been proven by earning the first place among 39 teams from all over Europe in the simulation stage of the *EuRoC*. At the challenge 2 of this EU-funded project, a mobile robotic platform shall be applied for intelligent, flexible and highly automated production systems. We discuss our results as well as the applicability of our framework to real industrial applications.**

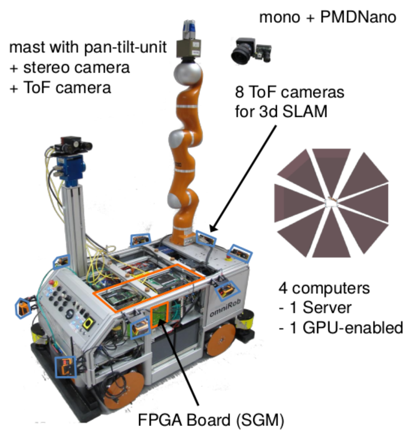

Fig. 1. Challenge platform for challenge 2 of the *EuRoC* challenge [3].

I. Introduction

Paradigms of the industrial production are shifting: to this date, automation found his way into large-volume manufacturing lines while many tasks are still done by human workers for medium-sized companies with a smaller number of units [1]. Industrial robots work mostly separated from humans in secured areas performing repetetive tasks in well-known and clearly defined environments. Handling operations (e.g. for plastic moulding, packaging, pick-and-place) and welding represent more than 75% of the use-cases in Europe of 2011 while half of all robots are concentrated in the automotive industry [2]. Regarding recent developments and future perspectives, industrial production has to become more and more flexible: in order to achieve shorter series with customized products, there is a need for reduction of change over times, reusable processes, algorithms and set-ups as well as safe human-robot collaboration.

The use of additional sensor feedback offers many options for the automation of more complex tasks, including detection of various objects, obstacles, safe navigation in human workspaces or precise positioning with uncertainties. A promising and innovative approach is the use of autonomous mobile robot units, equipped with a manipulator arm and additional sensors (e.g. cameras and force/torque sensors). Its key advantage is expected to be the ability to share a common working area with human workers, facilitating a flexible application for varying and complex tasks. Several commercial mobile platforms of this kind have been presented for research and academical use (e.g. *PR2* by *Willow Garage* [4], *youbot* by *KUKA* [5]). In robotics research, algorithms and software frameworks are being developed since years aiming at achieving high levels of autonomy. However, they are not common in industrial production since they have not yet shown the necessary level of robustness. An overview of related research projects and conceivable applications is given in [6]. Recently, the Autonomous Robotic Manipulation (ARM) competition was launched by DARPA [7]. The competition's software track focused on solving specific manipulation problems using tools such as drilling or unlocking and opening of a door. Each task was tested separately in static environments with only minor focus on obstacle avoidance but on grasping, sensor calibration and force feedback. Several teams have published their approaches [8], [9], [10]. In 2014, the EU-funded project *EuRoC* [1] [3] has been started. In course 2 of the *EuRoC*, competitor teams are developing autonomous robot systems for logistics and robotic co-workers. The challenge platform consists of a KUKA omniRob mobile platform equipped with a torque-controlled light-weight KUKA iiwa 7 DOF manipulator and several vision sensors (see Fig. 1). In contrast to the *DARPA ARM* [8] the *EuRoC* focuses on automatic high level sequencing of actions including advanced error handling and task planning for pick-and-place tasks in complex environments. Our team *AM-Robotics* ranked

Institute of Applied Mechanics, Technische Universität München, 85748 Garching, Germany. Email: {arne.hildebrandt, christoph.schuetz, daniel.wahrmann, robert.wittmann, rixen}@tum.de

[1]European Robotics Challenge, 2014–2017, Grant Agreement No.608849.

first at stage 1[2] of the challenge amongst 39 teams. Various pick-and-place tasks had to be performed without any user interaction while dealing with uncertain object localization, calibration errors, obstructed environments and multi-level decision processes. We have developed a powerful and robust software framework with the main design objectives:

- Ready for reuse: easy adaptability for different tasks in changing environments.
- Full autonomy: robust error-handling without any user interaction.
- State-of-the-art technology: use of latest image processing and motion planning frameworks.

II. Software Framework

Solving complex tasks and considering a variety of unknown events in the overall procedure requires a sophisticated software framework. This section gives an overview of our framework, starting with the architecture and continuing with a closer look on the main modules.

A. Architecture

The overall platform, provided by the host of challenge 2 is shown in Fig. 2. The simulator and its communication interface for interacting with the challenger application are predetermined and can not be changed. In the following an overview of our challenger application control system architecture is described.

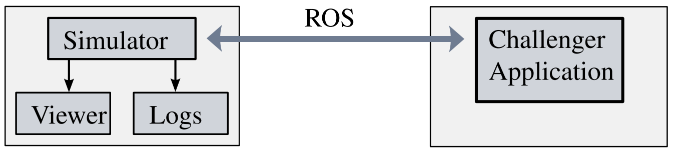

Fig. 2. Platform provided by challenge 2 host including the (Gazebo-based) simulator, the ROS simulator-interface and the challenger application.

A common way to solve complex tasks is to divide them into smaller subtasks. Brooks et al. [11] present a flexible control system for a mobile robot and decomposes it in modules based on task achieving behaviors. Each module is running asynchronously as part of a finite state machine. The authors of [12] present a scheduler based framework which is used in flexible assembly cells. This method allows to adapt the overall sequence for each individual part.
Another important aspect that has to be considered in the design of a control architecture is reliability and safety. [13] gives an overview of existing work concerning those topics. Our framework consists of independent modules, each of which is responsible for a certain subtask of the overall pick-and-place process. They are running as independent ROS-nodes and use ROS-provided tools (messages, services, actions) for communication[3]. The following modules are realized: state machine, state observer, vision system, grasping and motion planning. This structure is intrinsically flexible. As long as communication interfaces are maintained, each module can be changed or adapted according to the application needed. An overview of the overall system architecture is shown in Fig. 3. The interface to the simulator is as follows:

1) the vision system obtains current camera data,
2) the state observer and transformation broadcaster receive measurements for joint angles and torques and
3) the motion planning commands desired configurations.

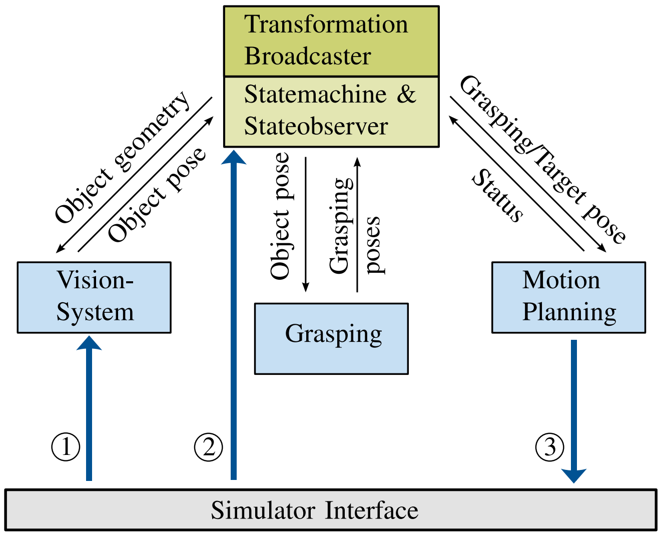

Fig. 3. Control system (challenger application) overview including the main modules and communication data.

The state machine node acts as central communication and coordination module, including the overall task sequence. The sequence to solve a basic task is depicted in Alg. 1. Together with the state observer, it is responsible for the error handling whereat an error causes a modification of the task sequence. An exemplary sequence and adaption due to an error is shown for the gripping sub-task in Alg. 2. Similar to [12], we are using a scheduling system to adapt the state sequence which is processed in the state machine. Due to the usage of threaded service calls and ROS *actionlib*, there are no blocking function-calls in the state machine which enables the reaction at any time to errors. There is an additional node, using ROS tf-package, that publishes all available transformations of the robot's current configuration and the transformation from the camera frame into inertial frame of reference (FoR). This is used by several modules e.g. to transform the vision data from the camera FoR into world coordinates.

B. Modules

1) Vision System: The vision module handles and processes the environment data from the simulated RGBD sensors to obtain the location of objects and obstacles. Each sensor consists of an RGB image and a 2.5D depth stream, which are overlayed by varying levels of gaussian noise according to the task (refer to Fig. 5). Based on this data,

[2]The challenge is divided into 3 stages: Stage 1, simulation contest, 07/2014 - 11/2014; Stage 2, realistic labs; Stage 3, field tests.
[3]ROS: Robot Operating System, `http://www.ros.org` [14]

Algorithm 1 Pick & Place control sequence

```
1:  Request task & start simulation
2:  Get task-description
3:  Scan environment
4:  Set n = 1
5:  repeat
6:      Locate Object n
7:      Calculate possible grasping poses
8:      Move to object
9:      Grip object
10:     Move to target zone
11:     Place object n
12:     if Object in zone then
13:         Set object finished
14:         n → n+1
15:     end if
16: until All objects finished
17: save log & stop simulator
```

Algorithm 2 Grip sequence with error handling

```
1:  object gripped = False
2:  while object gripped == False do
3:      Perform close range object pose estimation
4:      Move to object
5:      Close gripper
6:      Lift gripper and remain in static configuration
7:      if object is gripped == True then
8:          object gripped = True
9:      end if
10: end while
```

the vision module needs to determine the pose of several objects -which differ in shape and color- and generate an environment map for safe navigation.

An overview of traditional and modern approaches for object recognition can be found at [15]. In recent years, small and inexpensive 3D sensors became available and motivated new research and developments in 3D data processing (see [16]). In this context the open-source *Point Cloud Library* (*PCL*) [17] has been developed and integrated into ROS. For further information about its basic principles and algorithms, see [18]. Our vision module is based on the *PCL* since it provides tools to efficiently handle and process 3D point clouds (PCs). Most 2D (and some 3D) pose estimation algorithms rely on previous feature learning from a training set of images of the object. We consider these methods as not applicable for our task since object characteristics (shape, color) are not known in advance. Instead, we use a 3D PC-data descriptor for pose estimation (*Fast Point Feature Histogram*) as described in [19].

Our general strategy for image processing is shown in Fig. 4 and consists of the following steps:

a) Scan Environment: As the environment is not dynamic, multiple images are taken from different points of view (using both the scene and the tool center point (TCP) camera and filtering known parts of the environment). In the challenge tasks, objects can have only one out of six different colors. Thus, the PC is separated into six colored sub-PCs via HSV-filtering. An example of this procedure is shown in Figs. 5, 6 and 7.

b) Environment Mapping and Voxelizing: Once the PCs are created and combined, they are voxelized both for computational efficiency and to have a homogeneous distribution of points. The total (not color-filtered) PC is passed to the motion planning module via *Octomaps* [20] (see Fig. 6).

c) Pose Estimation: For object pose estimation, an ideal PC is created based on the geometric description of the object. The corresponding colored PC is clustered and obstacles or other objects are filtered out based on size. Finally, using the *Fast Point Feature Histogram* both PCs are compared iteratively in order to find the correct transformation (see Fig. 7).

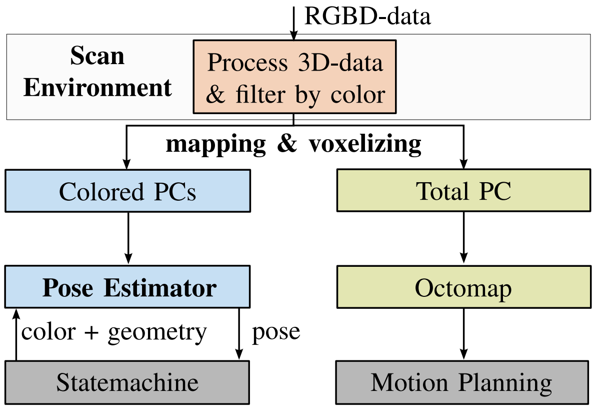

Fig. 4. Image processing strategy and integration in the overall framework.

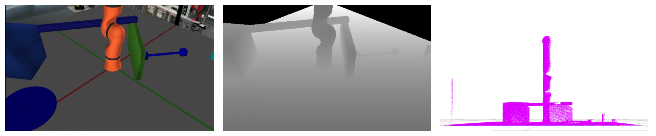

Fig. 5. RGB image (left) and 2.5D depth stream (middle) sent by the sensor, and the resulting 3D PC (right).

2) Motion Planning: The motion planning module is responsible for generating feasible joint paths in order to reach a given target pose of the end-effector. Furthermore, we use the motion planning node for feasibility checking of the grasping poses (refer to II-B.3). Collisions have to be avoided and kinematic constraints have to be met. Depending on the task, the joint space $\mathcal{C}$ is defined up to 9–dimensional

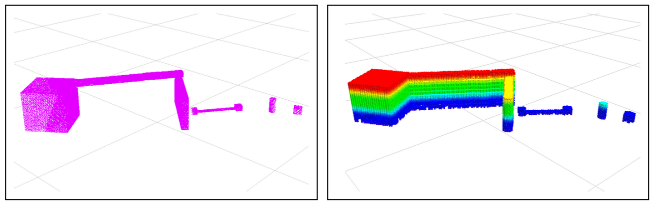

Fig. 6. Filtered total PC (left) and *Octomap* sent to the motion planning module for colision avoidance (right).

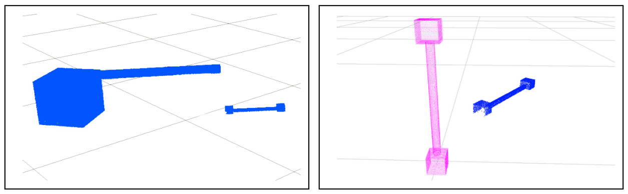

Fig. 7. Color “blue” PC (left) and the feature comparison (right) of the ideal PC (purple) and the clustered color PC (blue) for pose estimation.

with a planar joint at the base and 7 rotational joints of the manipulator While joint limits are known, information about obstacles is passed by the vision module via *Octomaps*.
Driven by the task and the geometry of the end-effector, the workspace $\mathcal{W}$ is defined by the position and orientation of the gripper TCP. Thus, the boundary constraints are given by the initial configuration of the robot in $\mathcal{C}$-space and the target pose of the end-effector in $\mathcal{W}$-space. For high-dimensional planning in obstructed environments, sampling based methods such as *Rapidly Exploring Random Trees* (RRT) or its derivatives [21] exploit the space of feasible paths very efficient and complete due to their probabilistic character. Besides, several optimization based planning frameworks have been presented [22], [23].
Recently, the comprehensive planning framework *Moveit!* [24] has been presented for ROS. It includes the *Open Motion Planning Library* [25] which offers implementations of various sampling based algorithms as well as collision detection algorithms as part of the *Flexible Collision Library* [26]. Our group takes advantage of this framework since it is well suitable for the given tasks. The planning algorithm *LBKPIECE* gave the best results in our experience.

3) Grasping: The aim of the grasping module is to compute possible grasping poses for a given object geometry and object pose. Possible geometries are limited to the three primitives — cube, cylinder, handle and puzzle piece — with varying dimensions. The robot is equipped with a Schunk PG70 Gripper, that has one linear axis to modify the distance between its two parallel fingers (see Fig. 8). The authors of [27] introduce two performance measures for grasping poses which are used in their automatic grasp pose generation algorithm. Some criteria presented in that work, like the distance from the contact between gripper and object to the object’s center of mass, are also used in our approach. Vahrenkamp et al. [28] present a RRT-based method to determine collision-free grasping motions.
Since the objects geometries are limited to three primitivies no sampling based algorithm is neccessary. Instead, grasping poses can be calculated fast with predefined strategies depending on the object type. First, these strategies search for two parallel planes where the gripper is able to grab. Thereupon, the exact gripping position for each combination of parallel planes/lines is computed as close as possible to the object’s center of mass while taking into account possible collisions of the gripper with the object and the environment. The output is a set of grasping poses shown in Fig. 8 that are checked for inverse kinematics solutions. Kinematically feasible poses are then returned to the state machine.

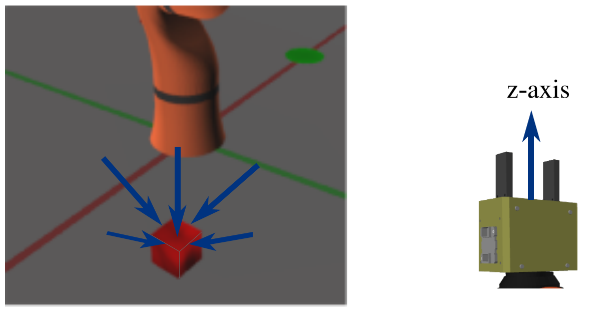

Fig. 8. Right: gripper and corresponding z-axis of gripper TCP frame of reference. Left: Possible gripper poses for red cube object. Each arrow indicates two poses that are rotated by $180\,^{\circ}$.

4) State observer: The measured force of the gripper and the measured joint torques are observed by the state observer. If the joint torques exceed a predefined limit the stateobserver causes an immediate stop of the overall motion. After such an event the limits are relaxed and the state machine commands the same goal pose with a decreased speed limit. In case the manipulator is in contact with an obstacle and moving the manipulator is unsuccessful for several times, an abortion of current state is caused and the manipulator tries to return to its homing pose.
The state observer is also responsible for interpreting the effective external load at the TCP in order to check whether an object was gripped successfully or not.

III. Simulation Results

In the following section, we present different tasks that have to be solved at stage 1 of the *EuRoC*. Each task has been tested by the challenge hosts for different setups to examine the robustness of our framework[4].

A. Pick-and-Place in Obstructed Environments

The basic tasks are *pick-and-place* scenarios with increasing difficulties. The robot has to locate objects, to pick them and to place them in the corresponding target zones while avoiding obstacles and maintaining its dynamic and kinematic limits. The obstacles as well as the dimensions and locations of the objects are not known in advance. The target zones are known from the task description but vary in each setup. Fig. 9 shows a top-view of the setup.
Alg. 1 describes the overall strategy: First, the environment is scanned applying the procedure of Alg. 3. The *Octomap*-based representation of the environment is generated for collision checking based on PC-data from the vision module. Since some parts of the environment are not visible for the scene camera, the environment representation has to be created successively using views of the TCP camera as well. Several pose sets are defined to fully explore the environment.

[4]A video showing the different simulation tasks can be found online at `https://youtu.be/OTWEZd6BMk8`.

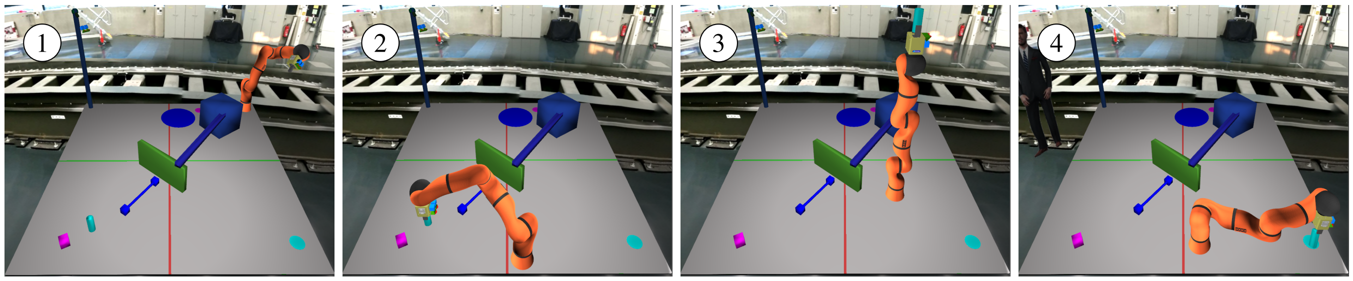

Fig. 10. *Pick-and-Place* task in obstructed environment: 1) *Scan Environment*. 2) *Grip object* (cyan cylinder) 3) *Move to target zone* in 2 DOF *transport*-configuration. 4) *Place Object* into target zone.

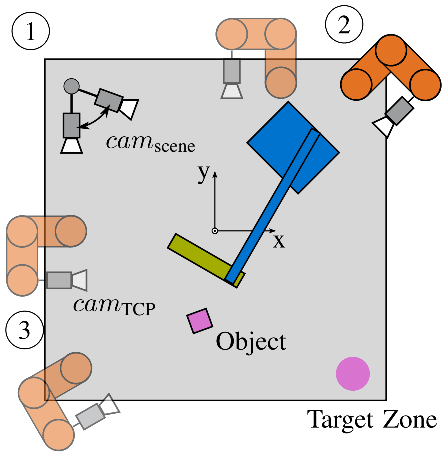

Fig. 9. Top-View of the scene *Pick-and-place in obstructed environments*. The object (magenta) has to be located and placed into the target zone while avoiding obstacles (blue, green).

Algorithm 3 Scan Environment

```
1: Primary and secondary explore pose sets $\mathbb{A}, \mathbb{B}$
2: Record scene with $cam_{\text{scene}}$ (1)
3: Set $n_{\text{success}} = 0$, increase counter with each taken image.
4: Choose random pose set $i$ of $\mathbb{A}$
5: repeat
6:     Move to first feasible configuration of $i$ and take image with
       $cam_{\text{TCP}}$ (2)
7:     if no feasible pose found then
8:         Switch pose set $i$
9:     end if
10: until first feasible pose found or no remaining pose in $\mathbb{A}$
11: Record scene with $cam_{\text{scene}}$
12: Move to next feasible configuration of current pose set $i$
13: repeat
14:     Choose random pose set $i$ of $\mathbb{B}$
15:     Move to 2 feasible configurations of $i$ and take images with
        $cam_{\text{TCP}}$ (3)
16: until $n_{\text{success}} = n_{\text{success,max}}$
```

In the following, objects are located sequentially. For a located object the grasping module calculates feasible grasping and placing poses (refer to II-B.3).
Based on these poses the *pick-and-place* procedure is performed as follows: The robot moves to a predefined *transport*-configuration and approaches the located object using only the linear axes of the base in order to reduce the complexity of the planning problem. Once the distance between the robot and the object is less than a predefined threshold the gripping sequence as described in Alg. 2 is performed using 9DOF. After picking up the object, the robot moves to the corresponding target zone switching back to the *transport*-configuration and performs a placing procedure similar to the grip sequence (see Fig. 10).
The whole *pick-and-place* sequence is performed until all objects are placed in their target zones. Our approach of a $\mathcal{C}$-space adaption has proven to be robust as well as time-efficient.

B. Puzzle Assembly

In the puzzle assembly scenario puzzle pieces have to be assembled into a fixture. Fig. 11 shows a schematic setup of this task. Using our framework we are facing two main issues: In contrast to the standard task, the placing order is important because not every order would enable the assembling of the puzzle. Thus, the logic of the state machine module is extended by a sorting algorithm. Furthermore, a simple placing strategy is not suited for a robust assembling of the task due to uncertainties of the calculated object pose and calibration errors. Our approach extends the former placing strategy by a *swing-in* motion. The *swing-in* motion is defined as a sinusoidal movement of the TCP which superposes the regular placing motion. That way, the puzzle pieces reach their target pose despite the several uncertainties. This strategy improved the success-rate considerably by exploiting the compliance of the torque controlled joints and preventing jamming.

C. Moving Conveyor Belt

In this task the robot has to pick up red cubes from a conveyor belt and place them in a target zone (see Fig. 12). The cubes are dropped on the conveyor belt while the moving speed (in x'–direction) of the belt increases. Thus, for each cube i the robot has a decreasing cycle time $\Delta t_{c,i}$ to pick the cube and to place it in the target zone. Since time is an issue, our focus is to speed up the *pick-and-place* procedure. Our strategy avoids the exact calculation of the

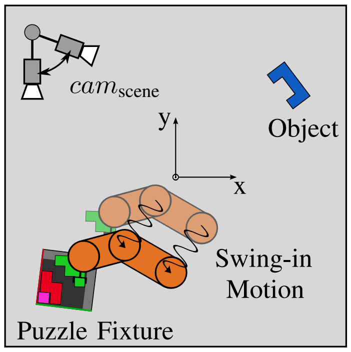

Fig. 11. Top-View of the scene *Puzzle Assembly*. Several puzzle pieces have to be localized and placed into a puzzle frame. Approaching the puzzle frame, a *swing-in* motion is used to account for inaccuracies exploiting the torque control of the manipulator.

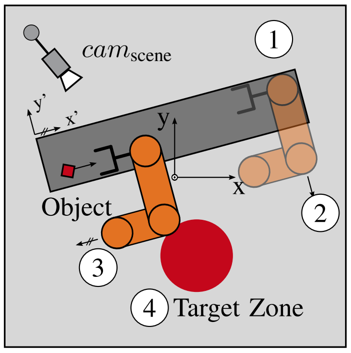

Fig. 12. Top-View of the scene *Moving conveyor belt*. The object (red) is dropped on the conveyor belt at random position and has to be put into the target zone.

rendezvous point. Instead, the grasping procedure (compare Alg. 2) is simplified. Due to a sufficient gripper width, we choose frontally to grip each object (compare Fig. 12). Thus, the robot first moves in the predefined gripping pose (the conveyor belt height is estimated at the beginning by the scene-camera). Then it estimates the object's center (and not the complete pose) relative to the middle of the conveyor belt using the TCP cam (1). Based on this position the robot adjusts its y'–position relative to the conveyor belt and moves towards the red cube on a straight line (3). Thereby, the planning problem can be reduced to an analytically solvable one dimensional problem taking into account imprecise timing information. Using this strategy, both vision and motion planning processing times are negligible and speed depends mainly on the robot's capabilities.

IV. Application and Outlook

In this section we discuss further challenges and the applicability of our framework to real industrial applications, such as those included in further stages of the *EuRoC*. Challenge 2 of the *EuRoC* project proposes mobile manipulators addressing the scenarios *Logistics* and *Robotic Co-Workers* as described in [29]. In this regard, robots shall be enabled to work autonomously in unstructured environments while respecting requirements for safe human-robot collaboration. Facing the application in real-world scenarios, several additional requirements compared to the simulation have to be met by the software framework. The main challenges include the precise positioning of parts, the safe operation in unstructuredand dynamically changing environments sharing workspace with humans and more complex conditions for the different sensors (e.g. vision module).
Our presented general strategy and framework is inherently flexible and adaptable to new tasks, scenarios and real-world applications. As stated before, changes or even the complete re-write of individual modules are easily implemented. In the following we give an overview on how each respective module could be redefined for such scenarios.

1) State Machine: Aiming for long-term autonomy, a robust operation has to be ensured. The proposed framework has proven to be flexible as well as easily adaptable to task-specific strategies and enables a comprehensive error handling. These requirements are crucial for an autonomous operation in a dynamical environment. A good assessment of possible events -which needs to be done for each specific application- is crucial for an autonomous operation in a dynamical environment.

2) Vision: The problem of image processing in real world scenarios is naturally more challenging than in simulated, fixed ones. Real RGB-D sensors present much more complex difficulties (e.g. non-Gaussian noise, calibration and alignment errors) and may not be the most adequate ones. Nevertheless, point cloud processing algorithms can be used with other, more precise kinds of 3D and RGB sensors. An initial "scan environment" routine may not be enough in dynamic enviroments and the environment map would have to be constantly updated. Some tools, e.g. *Octomap*, provide probabilistic updating capabilites which also help dealing with sensor uncertainties. Additionally, there are numerous works and algorithms (also available in *PCL*) which help deal with sensor fusion and SLAM.
Naturally, the identification of objects is much more complex in real-world scenarios as in a simulated environment with 6 possible colors. Still, this is an extended subject of research and there are several strategies (e.g. identifiers) that help dealing with it. Pose estimation algorithms depend strongly on each particular case, so they always have to be adapted -or completely changed- accordingly.

3) Motion Planning: Regarding the motion planning module, we define robustness as the ability to find a collision-free trajectory in case there exist one or otherwise to return an error in a reasonable amount of time. Generated trajectories have to be as short and smooth as possible or better, optimal w.r.t. time and effort. Furthermore, it has to deal with dynamic environment and unforeseeable constraints. Due to a disadvantageous initial guess, the planner sometimes fails at finding a feasible solution in case the target pose is given in $\mathcal{W}$-space instead of $\mathcal{C}$-space. This issue can be solved by the adaption of sampling based algorithms designed for redundant systems, e.g. as proposed by [30]. Regarding dynamic environments, we plan to develop a new module, that monitors the current environment while moving and stops the system smoothly in case an unforeseen collision impends.

4) Grasping: The current implementation of the grasping module is limited to a fixed amount of object shapes. Object data is given by a combination of shape primitives. For real-world applications, it might be desirable to use CAD-data for the grasping pose calculation. However, this calculation can be done in advance and the use of pre-generated catalogues seems applicable. In addition, part-specific end-effectors may be advantageous in terms of robustness and precision.

5) State Observer: Acting as main interface for sensor feedback to the state machine, the state observer is crucial to increase the robustness of the overall control system. Improving autonomy in (real) dynamic environments the use of sophisticated algorithms for localization, multi-sensor fusion and disturbance estimation are necessary. Safety in terms of industrial standards plays an important role and has to be added to current design.

V. Conclusion

Mobile robot platforms with dexterous manipulators and integrated vision and force/torque sensors are a promising approach to cope with future demands on flexible, small-scale and autonomous production. In this paper, we introduced a well structured and extensible software framework that consists of various independent modules with state-of-the-art algorithms and concepts. This framework has proven its worth at stage 1 of the *EuRoC* project coming out on top of 39 challenger teams. Looking into possible applications in real-factory environments, we set up a basis on which further extensions or adaptions can be built on.

Acknowledgments

Special thanks go to our motivated and brilliant students T. Blume, F. Bräu, M. Moein, P. Seiwald, A. Seppålå, T. Smith, N. Tekles, I. Uygur, S. Yousefpour whose contribution was essential to the team's success.

References

[1] M. Hägele, K. Nilsson, and J. N. Pires, "Industrial Robotics," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer Berlin Heidelberg, 2008, ch. 42, pp. 963–986.

[2] S. D. International Federation of Robotics (IFR), *World Robotics 2012 - Industrial Robots*. Frankfurt am Main: VDMA, 2012.

[3] "European Robotics Challenges - webpage," 2015. [Online]. Available: http://www.euroc-project.eu/

[4] "Willow Garage PR2 - webpage," 2015. [Online]. Available: https://www.willowgarage.com/pages/pr2/overview

[5] "KUKA youbot - webpage," 2015. [Online]. Available: http://www.kuka-labs.com/en/service_robotics/research_education/youbot/

[6] M. Hägele, W. Schaaf, and E. Helms, "Robot Assistants as Simple and Effective Tools in Manufacturing Environments," in *Advances in Human-Robot Interaction*, E. Prassler, G. Lawitzky, A. Stopp, G. Grunwald, M. Hägele, R. Dillmann, and I. Iossifidis, Eds. Springer Berlin Heidelberg, 2005, ch. 7, pp. 347–358.

[7] D. Hackett, J. Pippine, A. Watson, C. Sullivan, and G. Pratt, "An overview of the darpa autonomous robotic manipulation (arm) program," *Journal of the Robotics Society of Japan*, vol. 31, no. 4, pp. 326–329, 2013.

[8] L. Righetti, M. Kalakrishnan, P. Pastor, J. Binney, J. Kelly, R. Voorhies, G. Sukhatme, and S. Schaal, "An autonomous manipulation system based on force control and optimization," *Autonomous Robots*, vol. 36, no. 1-2, pp. 11–30, 2014.

[9] N. Hudson, T. Howard, J. Ma, A. Jain, M. Bajracharya, S. Myint, C. Kuo, L. Matthies, P. Backes, P. Hebert, T. Fuchs, and J. Burdick, "End-to-end dexterous manipulation with deliberate interactive estimation," in *IEEE International Conference on Robotics and Automation*, May 2012, pp. 2371–2378.

[10] J. Bagnell, F. Cavalcanti, L. Cui, T. Galluzzo, M. Hebert, M. Kazemi, M. Klingensmith, J. Libby, T. Y. Liu, N. Pollard, M. Pivtoraiko, J.-S. Valois, and R. Zhu, "An integrated system for autonomous robotics manipulation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2012, pp. 2955–2962.

[11] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation*, vol. 2, no. 1, pp. 14–23, 1986.

[12] K. Abd, K. Abhary, and R. Marian, "A scheduling framework for robotic flexible assembly cells," *KMUTNB: International Journal of Applied Science and Technology*, vol. 4, no. 1, pp. 31–38, 2013.

[13] B. Dhillon, A. Fashandi, and K. Liu, "Robot systems reliability and safety: A review," *Journal of quality in maintenance engineering*, vol. 8, no. 3, pp. 170–212, 2002.

[14] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[15] C. Wöhler, *3D Computer Vision: Efficient Methods and Applications (Chapter 2)*. Springer, 2012.

[16] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 1–4.

[17] "Point Cloud Library - webpage," 2015. [Online]. Available: http://www.pointclouds.org

[18] R. B. Rusu, "Semantic 3d object maps for everyday manipulation in human living environments," Dissertation, Technische Universität München, München, 2009.

[19] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3d recognition and pose using the viewpoint feature histogram," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 2155–2162.

[20] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013.

[21] S. M. LaValle, "Planning Algorithms," *Methods*, vol. 2006, p. 842, 2006.

[22] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *IEEE International Conference on Robotics and Automation*, May 2009, pp. 489–494.

[23] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization," in *Proceedings of Robotics: Science and Systems*, 2013.

[24] I. A. Sucan and S. Chitta, "Moveit!" 2015. [Online]. Available: http://moveit.ros.org

[25] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012.

[26] J. Pan, S. Chitta, and D. Manocha, "FCL: A general purpose library for collision and proximity queries," in *IEEE International Conference on Robotics and Automation*, 2012, pp. 3859–3866.

[27] J. D. Wolter, R. A. Volz, and A. C. Woo, "Automatic generation of gripping positions," *IEEE Transactions on Systems, Man and Cybernetics*, no. 2, pp. 204–213, 1985.

[28] N. Vahrenkamp, M. Do, T. Asfour, and R. Dillmann, "Integrated grasp and motion planning," in *IEEE International Conference on Robotics and Automation*, May 2010, pp. 2883–2888.

[29] EuRobotics, "The strategic research agenda for robotics in europe," 2009. [Online]. Available: http://www.robotics-platform.eu/

[30] D. Berenson, S. S. Srinivasa, D. Ferguson, A. Collet, and J. J. Kuffner, “Manipulation planning with Workspace Goal Regions,” in *IEEE International Conference on Robotics and Automation*, 2009.