The EU Framework Programme for Research and Innovation H2020
Research and Innovation Action

CENTAURO

Deliverable D6.3 Autonomous Single-Arm Pick and Place Manipulation Skills

Dissemination Level: Public

Project acronym:	CENTAURO
Project full title:	Robust Mobility and Dexterous Manipulation in Disaster Response by Fullbody Telepresence in a Centaur-like Robot
Grant agreement no.:	644839
Lead beneficiary:	UBO - University of Bonn
Authors:	D. Pavlichenko, D. Rodriguez, M. Schwarz
Work package:	WP6 Manipulation
Date of preparation:	2018-01-16
Type:	Report
Version number:	1.0

Document History

Version	Date	Author	Description
0.1	2017-09-11	DP	Initial version
0.2	2017-09-27	DR	Autonomous grasping
0.3	2017-09-27	MS	Updated object perception
0.9	2018-01-16	DP	Include Pisa experiment
1.0			Submitted version

Executive Summary

This deliverable describes our approach to autonomous single-arm pick and place for the CENTAURO system.
The deliverable presents the two main components involved in this task: object perception and manipulation planning. Object perception is handled by different components developed at LIU and UBO, while the manipulation planning stages have been developed at UBO only. We describe the developed methods in detail and report on successful initial experiments using the integrated CENTAURO system.

Contents

1 Introduction

This CENTAURO Deliverable D6.3: Autonomous single arm pick and place manipulation skills presents implemented methods which allow the CENTAURO system to perform autonomous single-arm manipulation. These methods can be divided into following main categories: perception of the workspace, grasp generation and arm trajectory planning. These methods are covered by tasks T6.1: Object and workspace perception and T6.2: Collision-aware motion generation. Combined, they represent a solution for T6.3: Single-arm object pick and place. Figure 1 gives an overview over the components involved in this deliverable.

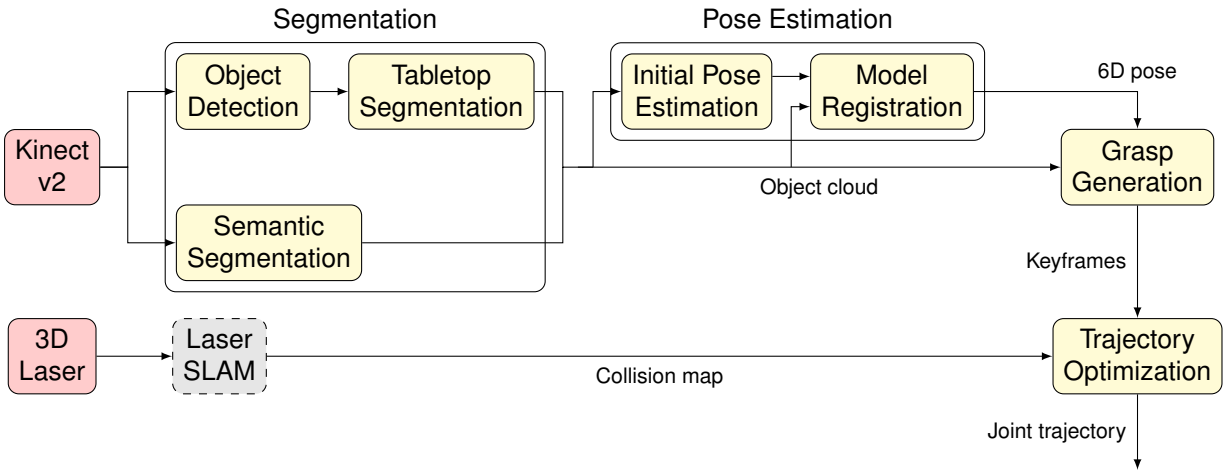


Figure 1: Overview of the pipeline for autonomous manipulation. Sensors are colored red, pipeline components yellow, and external modules from other workpackages are colored grey.

2 Manipulation Setup

In this section we describe setup used in the CENTAURO system to perform autonomous manipulation.

2.1 Calibration of Non-overlapping Cameras

The CENTAURO robot is equipped with three 2-megapixel color cameras, mounted in such a way that their views have only a minimal overlap. This camera configuration provides the operator with the largest possible combined field of view with a minimum number of cameras. However, it also prevents estimation of the relative camera poses with standard calibration methods. To address this problem we have developed a new calibration method [16], where additional cameras are used temporarily to create the required overlap. In comparison to previous methods developed for the same purpose, such as [6] and [1], this new approach has considerable lower calibration error and increased robustness to noise.

To evaluate the impact of additional cameras providing significant overlap, we experiment on synthetic data since the accuracy cannot be determined without ground truth or additional sensors. Briefly, we generate noise-free images, apply camera noise, estimate the extrinsic camera parameters and evaluate the result.

We experiment with synthetic 1600 x 1200-pixel cameras with very wide-angle lenses (focal length 1.67mm) configured either as a pair of cameras 90 degrees apart, three cameras at 45 degrees, and five cameras at 22.5 degrees, outward facing on a circle. Images are generated from a synthetic scene with 3D modeling software. In each case, five checkerboard calibration patterns (7-by-9 squares) are manually placed, one at a time, to maximally cover the joint field of view, as shown in figure 2. Images are rendered and the associated ground-truth image 2D coordinates of all pattern landmarks are stored. Errors caused by the camera - Gaussian blur ($\sigma = 0.5$), vignetting ($\cos^4(x)$, scaled to 0.25 gain at left/right image edge) and shot noise (Poisson distribution, approximated as $N(x, 0.1\sqrt{x} + 2)$, $px \in [0, 255]$ pixel intensity) - are simulated and added to the images. The calibration pattern landmarks are then located with sub-pixel precision (`findChessboardCorners()` in OpenCV). Finally, Gaussian noise ($N(0, \sigma_d)$ with $\sigma_d \in [0.0, 0.7]$) is added to the landmark coordinates to simulate residual camera distortion error. The coordinates are then used to estimate the relative transform between the two cameras in a pair.

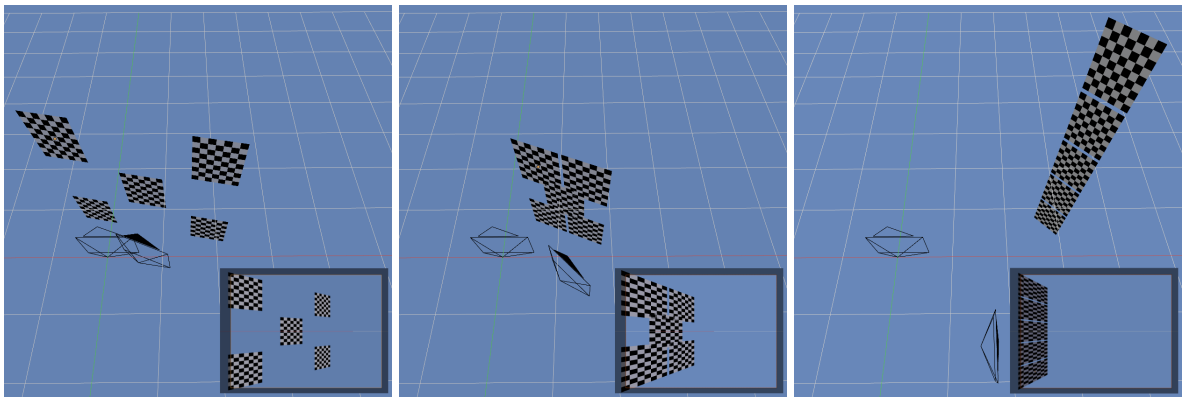


Figure 2: Camera and pattern setups with 22.5, 45 and 90 degree rotations between cameras. Insets show the pattern placements in the viewport of the right camera in the pair. Their appearance in the left camera are mirror-symmetric. The ground-plane grid lines are 1 meter apart.

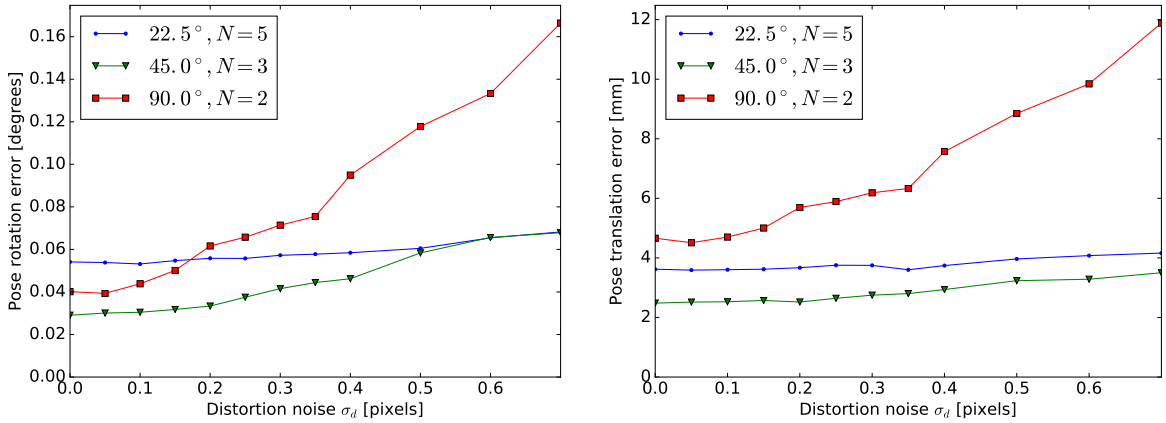


Figure 3: Expected pose rotation errors and expected pose translation errors as functions of added ϵ_d noise of magnitude σ_d.

.

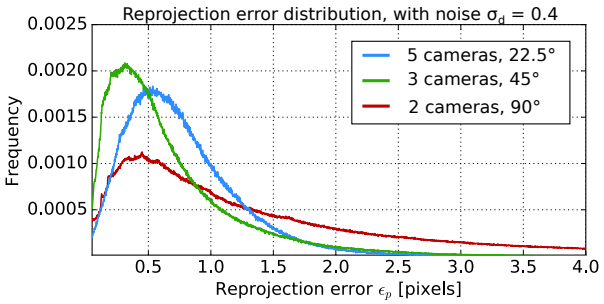


Figure 4: Distribution of reprojection errors

.

For each of the camera configurations, we generate $M = 1000$ calibration runs, with different noise samples added each time to acquire a reasonable sampling of the error distribution. N-view camera setups are simulated by linking the relative poses $P_{i,j}$ of pairs of cameras such that $P_{N,1} = \prod_{k=1}^{N-1} P_{k+1,k}$. For example, in a three-view setup, $P_{31} = P_{32}P_{21}$. In addition we compute the reprojection errors of 10,000 random points $\vec{y} = (x, y, z, 1)^T$, uniformly distributed across the image plane at uniformly random depths $z \in [0.1, 100]$.

The mean pose rotation and translation errors in $P_{N,1}$ are shown in figure 3. It is apparent that one additional camera during calibration (i.e N=3) exhibits greater robustness to noise compared to no additional camera. More cameras (i.e N=5) do not improve the robustness.

The distribution of reprojection errors is shown in figure 4, with coordinate noise $\sigma_d = 0.4$ chosen as an example. It is clear that the distribution for N=3 cameras has a lower error overall than either alternative.

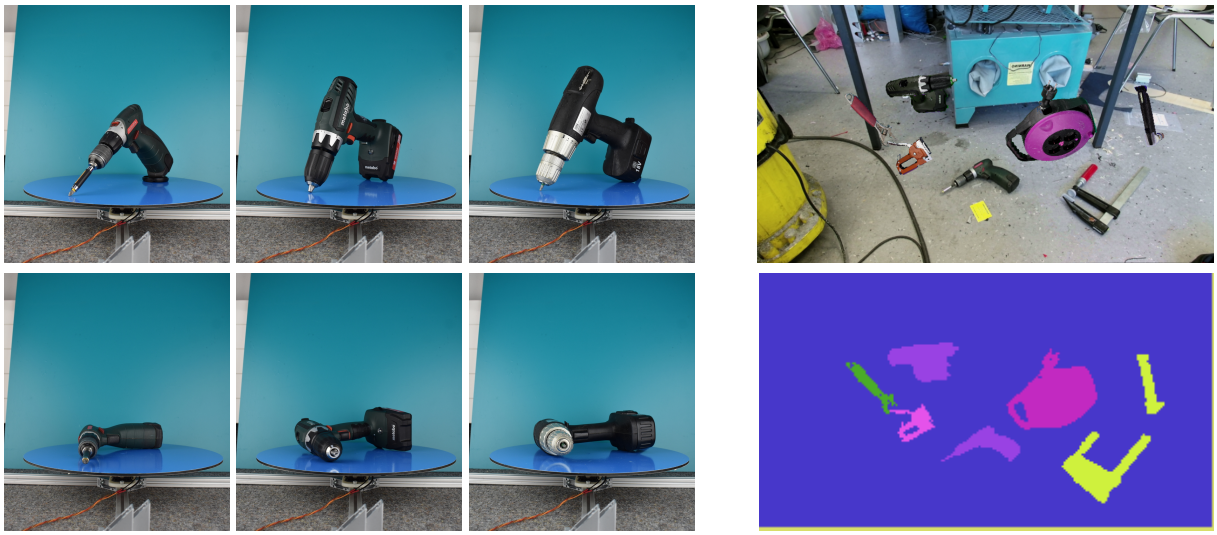


Figure 5: Turntable capture and scene synthesis. Left: Different drills on the turntable as captured by a DSLR camera. Right: Synthetic training scene generated by inserting new objects into the scene. The top image shows the resulting color image, the bottom shows synthetic ground truth for training the segmentation model.

3 Object Perception

In order to plan a grasp an object, the exact position and orientation have to be known. The target object has to be detected and localized. In this section, we discuss object perception methods which were implemented. In general, the object perception pipeline detects individual objects, segments them, and estimates the object pose.

3.1 Object Detection and Segmentation

To enable manipulation of the workspace, the CENTAURO robot has to detect and determine the pose of useful objects in its environment, for example tools. For the purpose of object detection and semantic segmentation, we train CNN-based models on a dataset of tools. In the project, two alternative detection and segmentation pipelines have been developed, with focus on either speed or precision.

3.1.1 YOLO with Tabletop Segmentation

The first alternative is based on the YOLO object detector [15]. The rectangles output by the detector also contain background pixels. If we assume that the objects are placed on a planar surface, which constitutes most of the background, most of it can be removed. This is done by estimating planes in the depth image and then remove all points belonging to the significantly supported planes. After this step, in the ideal case the remaining points belong to the objects.

3.1.2 Semantic Segmentation

A more integrated pipeline was developed using semantic segmentation, which directly produces pixel- (or point-)wise segmentation. Previous efforts have been based on a semantic segmentation network based on the OverFeat feature extractor [8]. While the gains from using

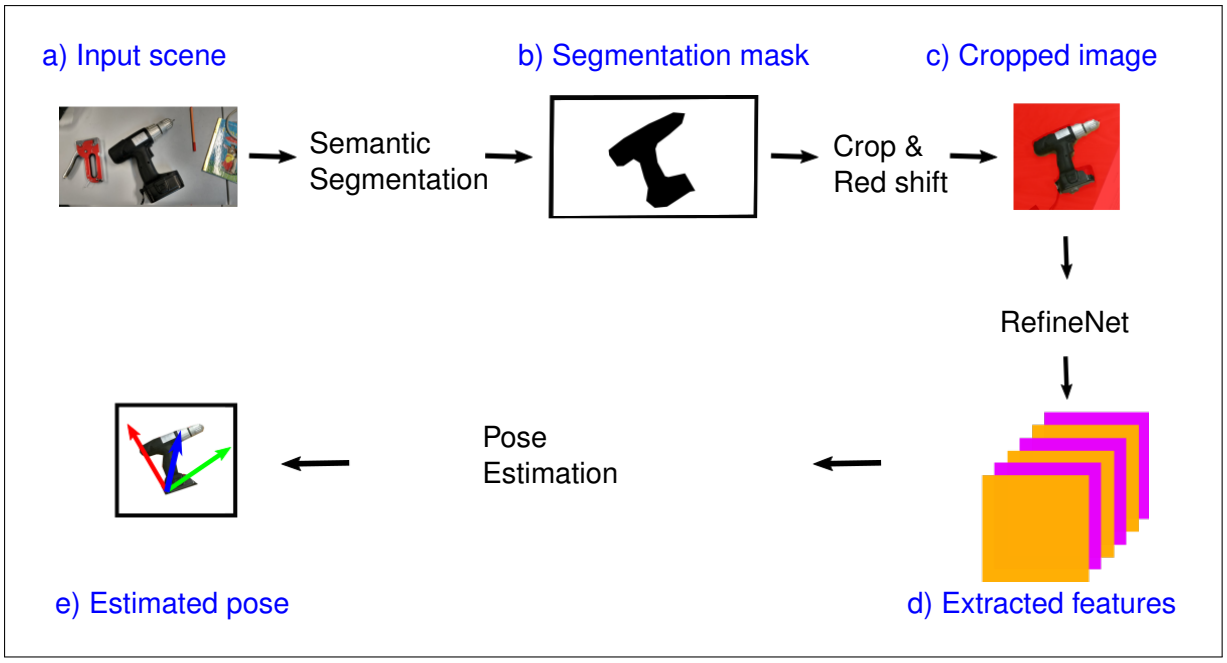


Figure 6: Steps in pose estimation.

pretrained CNNs as feature extractors or finetuning them to the task at hand are immense, CNNs pretrained for classification suffer from low spatial resolution in their later stages.

Our current pipeline switched to a more modern architecture called RefineNet [12], which addresses this problem by subsequently upsampling and merging higher-level feature maps with lower-level features of higher spatial resolution—creating a representation of the input image with both highly semantic information and high spatial resolution, which is ideally suited for semantic segmentation.

Deep learning methods require large amounts of training data. We address this problem by generating new training scenes using data captured from a turntable setup. Automatically extracted object segments are inserted into precaptured scenes (see Fig. 5). For details on the capturing and scene synthesis pipeline, we refer to [20].

3.2 Pose Estimation Network

To facilitate autonomous grasping of objects, we need the 6D pose of the objects. We augmented the semantic segmentation pipeline with an additional CNN to estimate the 5D pose (rotational, and X and Y of the translational components) of the objects from the RGB crops of the objects from the scene. The pose estimation module is trained with the features extracted from the RefineNet module discussed in the Sec. 3.1.2. The steps during the inference of the pose estimation module is shown in Fig. 6. The crop is determined based on the expected maximum size of the objects in the scene from semantic segmentation results and the pixels in the crop which do not belong to the object are pushed towards red. This representation encodes the segmentation results and decreases attention to the background (which may contain other objects). To generate the ground truth poses for training the network, the data acquisition pipeline described in [20] was extended to record turntable poses automatically and fuse captures with different object poses or different objects (see Fig. 5) with minimal user intervention.

We evaluated two different types of CNN architectures shown in Fig. 7. The single-block output variant predicts six values (rotation represented as a unit quaternions and x and y of

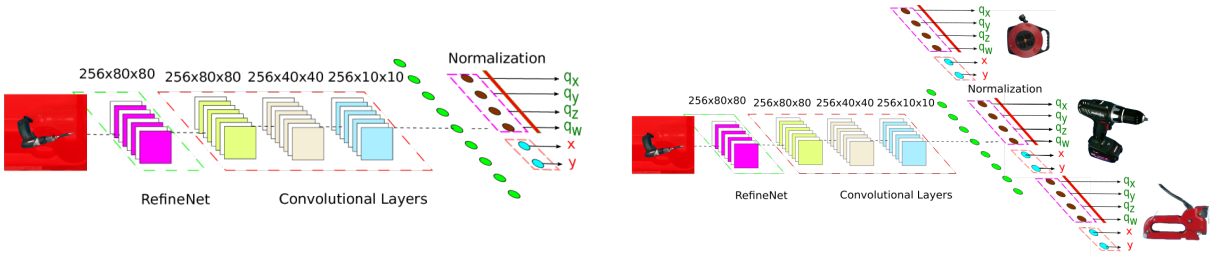


Figure 7: Pose estimation network architecture. Left: Single-block output variant. Right: Multi-block output variant.

Table 1: Average pose estimation errors on turntable training and evaluation dataset.

	Translation [pix]		Rotation [°]	
	train	val	train	val
Driller	8.3	9.9	7.6	10.2
Extension box	9.8	12.8	7.8	9.9
Stapler	6.3	8.1	4.3	6.4

translation) by taking RefineNet features of the crops of the objects as input, whereas the multi-block output variant predicts a set of six values for each category of the objects, with each set corresponding to one category. While computing the loss for the multi-block output variant, only the output corresponding to the relevant category of the object is subjected to the loss function and only the part of the network relevant for the output block is updated during back-propagation step. Our evaluation showed that the single-block variant performed slightly better in the presence of occlusion, and was thus chosen for use in the CENTAURO project. Table 1 shows quantitative results of training the pose estimation model on three objects captured with the turntable setup.

3.3 Feature-based Point Cloud Registration

In this work we integrate descriptive high-dimensional features into a probabilistic registration framework [4]. In contrast to the from point-wise color observations that were used in [5], these high-dimensional features capture the geometrical properties of the local neighborhood and are typically based on histograms. We use the Point Feature Histograms (PFH) [19], due to its discriminative power and invariance to rigid transformations. However, other types of invariant features can also be employed in our framework. The PFH uses both the locations and normals of points in a fixed-sized neighborhood. For each pair of points in the neighborhood, three angular features are extracted using an invariant reference frame. The descriptor is then constructed as a 3-dimensional histogram of these angles for all pairs, resulting in a feature vector of dimension $5^3 = 125$.

To incorporate the high-dimensional histogram representation into a probabilistic framework, we cluster the features using K-means. In this way, each histogram is labeled by the corresponding cluster index, and the observed feature of a specific point is given as the index of its histogram vector. Finally and similar to [5], this representation is used in an expectation maximization framework, where the likelihood of a categorical distribution over all observed features is maximized.

Using the Stanford Lounge Dataset, we compare our method to standard ICP [2], to color-

Table 2: A comparison our approach with state-of-the-art registration methods on the Stanford Lounge Dataset. The results are reported in terms of average and standard deviation of the inlier rotation errors, together with average failure rate.

	Avg. err	Std. dev.	Failure rate (%)
ICP[2]	4.32×10^{-2}	2.53×10^{-2}	15.70
Color GICP[11]	1.72×10^{-2}	1.75×10^{-2}	1.27
JRMPS[7]	1.78×10^{-2}	1.35×10^{-2}	3.67
CPPSR[5]	1.54×10^{-2}	1.08×10^{-2}	1.00
Ours [4]	1.54×10^{-2}	1.08×10^{-2}	0.6

supported generalized ICP (GICP) [11], and to the probabilistic registration methods JRMPS [7] and CPPRSR [5], see Table 2. Our method achieves state-of-the-art result in terms of error, comparable to CPPRSR, while significantly reducing the registration failure rate.

3.4 Object Detection and Pose Estimation

The model point-cloud is acquired by scanning an instance of each object. These models can be deployed on the input data generated by the Kinect v2, which provides both color and depth information. By combining the detection with the depth information we obtain a 3D point-cloud representation of the object.

Once the object points have been extracted from the sensor data, we can find the 6D object pose relative to the camera. This is done by aligning the detected object points to a model point-cloud using the algorithm described in Section 3.3, or the previous method in [5]. In Figure 8 we see an example where a detected hammer, denoted as input, is aligned to the model hammer.

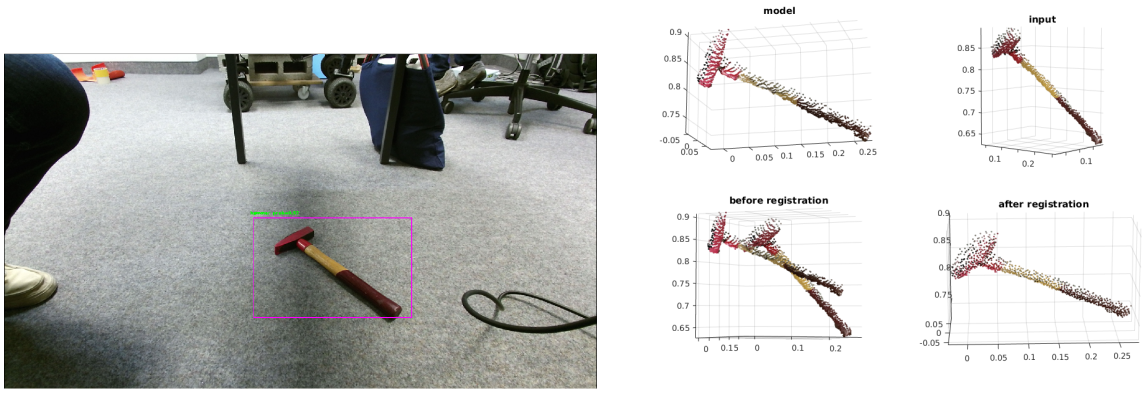


Figure 8: Object detection and pose estimation. Left: YOLO detection of hammer. Right: Pose estimation of detected hammer. The input hammer is aligned to the model hammer using the method described in Section 3.3.

4 Manipulation Planning

In order to perform an actual physical manipulation with an object it is necessary to carefully plan all the actions. Such planning allows to successfully accomplish a task in safe way for the robot and its surroundings. Manipulation planning can be decomposed into two main steps:

- Grasping generation
- Arm trajectory generation

During the first step a feasible grasp is generated. It should allow to hold the object reliably and its corresponding pre-grasp pose should be reachable by the arm. In the second step a feasible trajectory of the arm to reach the pre-grasp pose is generated. The trajectory should approach given pre-grasp pose as fast as possible while avoiding any collisions. When the pre-grasp pose is reached, the object can be grasped and the next iteration of the manipulation planning can start: given the task, next arm position is generated together with a trajectory necessary to reach it.

4.1 Grasping Generation

The method for generating the grasping is based on the observation that objects within a category are often similar in their shapes and usage. This approach transfer grasping skills from known instances to novel instances of an object category such as drills, hammers, screwdrivers, among others. This method incorporates category-level information by learning a shape space of the category.

Our method is divided into two phases: a learning phase and an inference phase. In the learning phase, the objective is to create a class-specific linear model of the transformations that a class of objects can undergo. We do this by first selecting a single model to be a canonical instance of the class, and then we find the transformations relating this instance to all other instances of the class using Coherent Point Drift (CPD) [13]. We then find a linear subspace of these transformations, which becomes our transformation model for the class. In the inference phase, the objective is: given a newly observed instance, search this subspace of transformations to find the transformation which best relates the canonical instance to the observed instance. Grasping information from the canonical model is also transformed to the observed instance and used to generate the final motion of the robot.

We define a category or class as a set of objects which share the same topology and a similar extrinsic shape. We find a descriptor vector of the deformation field that the canonical model has to undergo to transform into the training sample. Finally, we apply Principle Component Analysis (PCA) on this matrix to find a lower-dimensional manifold of deformation fields for this class.

With the transformation model we can now start registering the canonical shape to novel instances in order to estimate the underlying transformation. The parameters of the transformation are given by the latent vector plus an additional rigid transformation. The rigid transformation is meant to account for minor misalignments in position and rotation between the target shape and the canonical shape at the global level. We concurrently optimize for shape and pose using gradient descent. We want to find an aligned dense deformation field which when applied to the canonical shape $\mathbf{C}$, minimizes the distance between corresponding points in the observed shape $\mathbf{O}$.

From experiences with an anthropomorphic robotic hand (Fig. 9), namely the multi-fingered Schunk hand [18], we observed that a functional grasping cannot be described only by the final

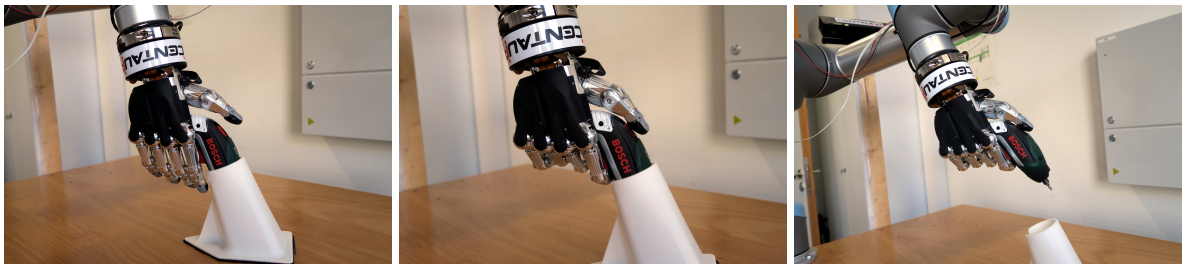


Figure 9: Anthropomorphic grasping of a drill with the Schunk hand.

joint configuration of the entire system (manipulator plus hand). During the motion, the hand can make use of geometrical and frictional advantage constraints to increase the chances of a successful grasping. For instance, for grasping drills, we can first make a lateral contact with the palm followed by a contact of the thumb with the rear part of tool, creating static frictional forces that may help during the grasp.

We represent a grasping action as a set of parametrized motion primitives. The parameters of the motion primitives are poses expressed in the same coordinate system of the shape of the object. These poses need to be defined only for the canonical model. For new instances, the poses are found by warping the poses of the canonical model to the instance. Because the warping process can violate the orthogonality of the orientation, we orthonormalize the warped poses.

We tested our method on the *Drill* category. We obtained the models from two online CAD databases: GrabCad[1] and the 3DWarehouse[2]. We trained the *Drill* models with their canonical shape and 9 additional samples. We evaluated the performance of our method's robustness to noise, misalignment, and occlusion and compared our results against results given by CPD. Each test was run on a full view of the object and on six different partial views of the object. To obtain the partial views, we used ray-casting on a single view of a tessellated sphere.

The results of our experiments are plotted in Fig. 10. We use the abbreviation CLS (Categorical Latent Space) for referring to our method. When a shape is fully visible with or without noise, CPD outperforms our method. This is rather to be expected as CPD is the source of training data from which we build our registration model. On the other hand, when the observed shape is misaligned, our method outperforms CPD, which can be explained by the additional rigid transformation component of our method. Moreover, when the shape becomes partially occluded, the method also outperforms CPD thanks to the topological information that lies in the latent space, which is not available in CPD.

For testing the grasping transference, we conducted a set of experiments using the multi-fingered Schunk hand, which has a total of 20 Degrees of Freedom (DoF) from which 9 are fully actuated. The experiments were carried out in the Gazebo simulation environment. The tests were performed on the *Drill* category. Due to the reduced number of meshes, we used cross-validation and created five transformation models leaving two testing samples for each transformation model. A trial was defined as successful if after the execution of the motion, the drill was held by the robotic hand. We got a success rate of 0.8 on grasping the drills. The successful grasping is shown in Fig. 11,

More details on this method can be found in [17].

[1] `https://grabcad.com/library`
[2] `https://3dwarehouse.sketchup.com/`

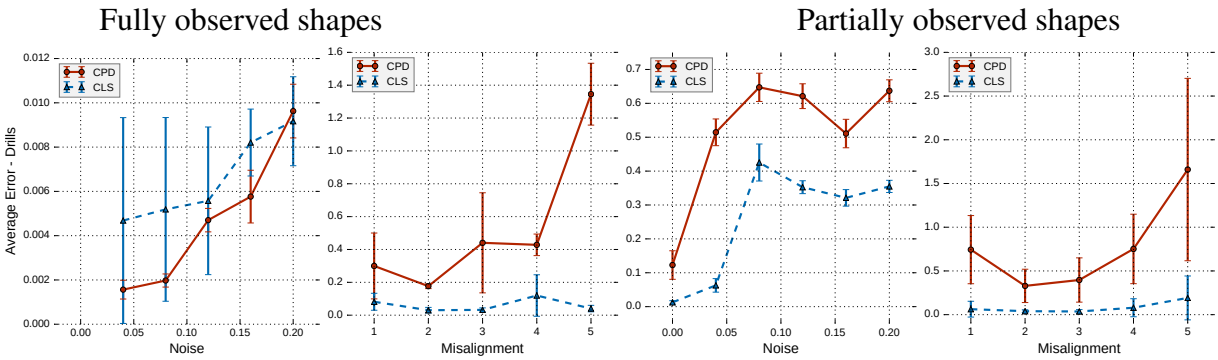


Figure 10: A comparison of our method (CLS dotted blue line) against CPD (red line). The first two plots show the results on fully observed shapes, while the last two, on partially observed shapes. Our method outperforms CPD on partial views of objects and misaligned fully observed shapes.

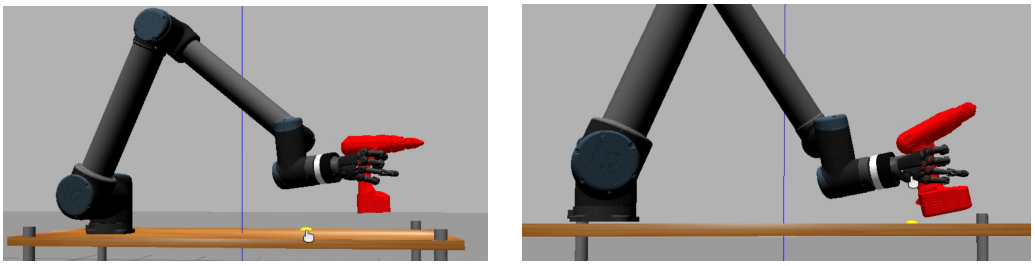


Figure 11: On the left, the grasping of the canonical shape is performed. On the right, the grasping is transferred to a new instance.

4.2 Arm Trajectory Generation

In order to reach a generated grasp or deliver the grasped object to a target position it is necessary to generate a collision free feasible trajectory. We solve this problem by extending a popular optimization method: Stochastic Trajectory Optimization for Motion Planning (STOMP) [10]. In the original STOMP, the duration of the trajectory is predefined and fixed. We propose a way to optimize duration as well. We further introduce a novel cost computation policy that allows to take full control over the computational effort, which lowers the runtime. We propose a cost function with five components: collisions, joint limits, orientation constraints, joint torque, and duration minimization. Each cost component is normalized to have values in the interval $[0, 1]$. By introducing an importance weight for each component, it is possible to set priorities, which allows obtaining qualitatively different trajectories according to the user preferences. Optimization is performed in two phases to reduce computations: with simplified and full cost functions. Our method is described in details in [14].

In order to evaluate the method we performed various experiments in simulation and with real robots. Experiments included comparison of runtime and success rate against several other planning methods: Lazy Bi-directional KPIECE (LBKPIECE)[3], which uses a discretized representation of projected state space in order to find a solution and is a combination of [21] and [3], RRTConnect [9] from OMPL [22], STOMP-Industrial[4], which is a newer implementation of the original STOMP. The experiments demonstrated that our method outperformed compared algorithms and that each component of the method performs intended optimization.

[3] `http://ompl.kavrakilab.org/classompl_1_1geometric_1_1LBKPIECE1.html`
[4] `https://github.com/ros-industrial/industrial_moveit`

Detailed descriptions of the experiments and their results can be seen in [14].

STOMP-New was tested with an Kuka iiwa manipulator on an omnidirectional mobile base. This configuration resembles the general principle present in the Centauro robot: self-moving body with an arm to perform manipulation upon arrival to the target location. The objective was to pick up different objects in different locations across the arena. There were three different objects: metal engine support parts of two sizes and engine pipes. Our method was used to plan trajectories for the iiwa arm to reach a pre-grasp pose and to deliver a grasped object to a pre-release pose, which is a classical pick and place task. In order to plan collision-free trajectories with objects of different shape during one run, rough approximations of these objects were attached and detached from our collision model on the fly. The algorithm planned several dozen trajectories during the challenge which allowed for successfull pick and place. Typical trajectory for object delivery is depicted in Fig. 12.

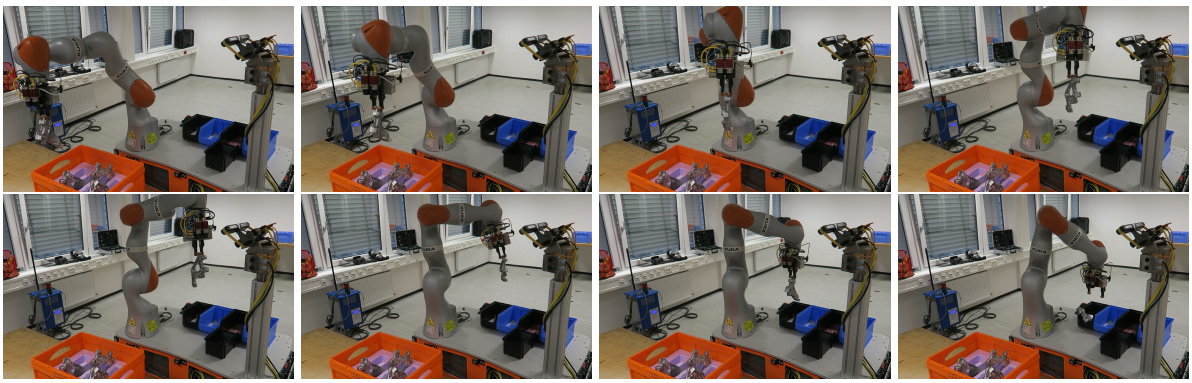


Figure 12: Execution of planned trajectory for object delivery to a pre-release pose.

In addition, we performed a replanning experiment, which shows the capability of our method to perform a quick replanning when an obstacle interferes with the already planned trajectory. An initial trajectory was planned with usual setting. However, right after planning process has finished, an obstacle was inserted on the way. Since during execution the remainder of the trajectory is being checked for collisions using new measurements, potential collision was detected. The execution of the trajectory was stopped and the replanning was performed, which allowed to achieve avoidance of initially unconsidered obstacle. In Fig. 13 one can see the robot executing the final trajectory, replanned to avoid the box.

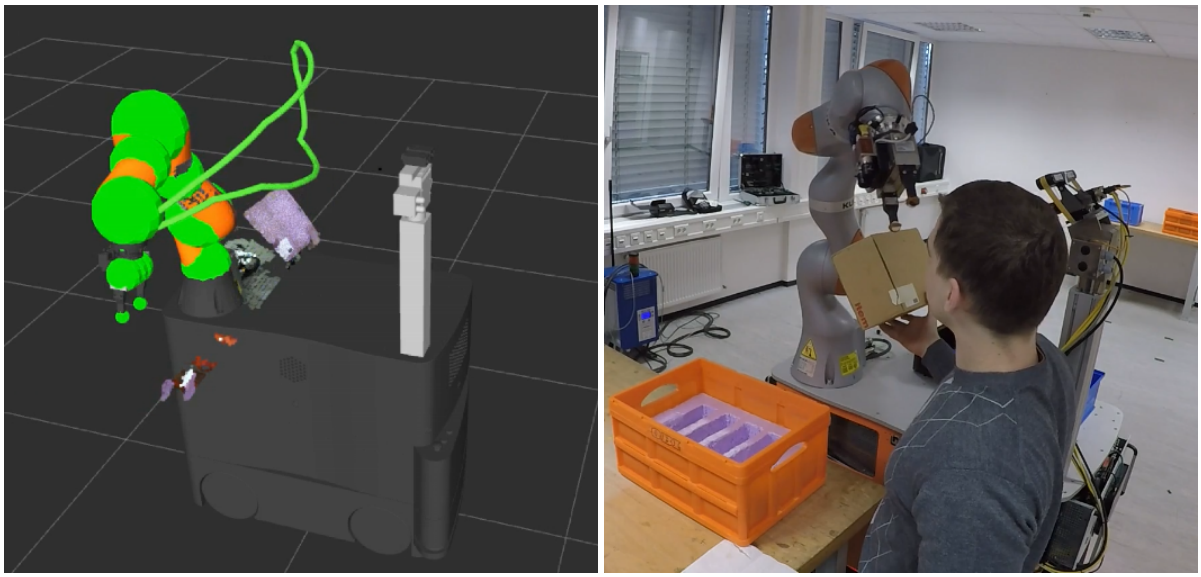


Figure 13: Replanning with iiwa arm. **Left**: Two trajectories: initial (lower line) and replanned (upper line), which was produced when previously unconsidered obstacle (box in the middle) interfered the initial trajectory. **Right**: iiwa arm executing a replanned trajectory, avoiding the previously unconsidered obstacle (box).

5 Integrated Manipulation Skills

Integrating together methods described in previous sections, we obtain a system which is capable to perform autonomous pick-and-place manipulation. Initial integration occured during the integration meeting at SSSA in Pisa during October 2017.

In a first test, we tested the trajectory optimization method described in Section 4.2 in isolation. The test demonstrated correctness of the interfaces and that the component is ready to be integrated. We performed optimization of arm trajectory using low duration cost importance weight in order to obtain relatively slow movement due to safety concerns. All other weights were in their neutral positions of 0.5. The optimization yielded a smooth trajectory in 0.39 s, which was successfully executed on CENTAURO robot.

A second experiment confirmed integration of most components of the object perception and grasp generation pipeline by grasping a drill autonomously. The semantic segmentation module (Section 3.1.2) was trained on synthetic scenes generated from turntable captures of different drills. It was then used to segment live point cloud data from the Kinect2 observing a novel drill instance (see Fig. 14), which was not part of the training set. The pose estimation

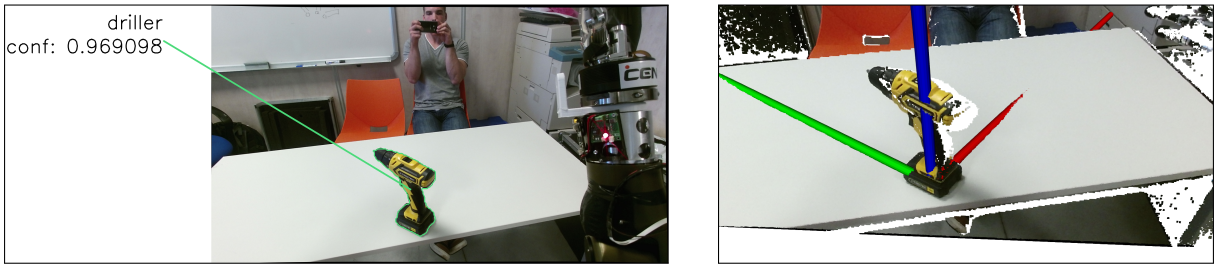


Figure 14: Object Perception during the drill grasping experiment. Left: Post-processed object contour from semantic segmentation. Right: Kinect2 3D point cloud with estimated drill pose from the pose estimation network.

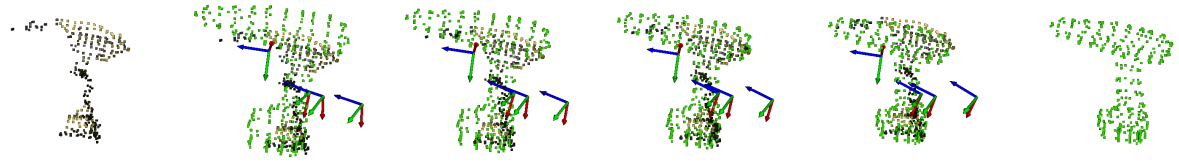


Figure 15: Transferring grasping knowledge to the presented novel instance. The input point cloud is at the leftmost while the inferred shape is at the rightmost.

network (Section 3.2) predicted an initial pose estimate. Then the grasp generation module (Section 4.1) was used to register the point cloud to a canonical model and transfer a predefined grasp from the canonical model onto the observed instance (see Fig. 15). Finally, the calculated trajectory was executed and the drill grasped successfully (see Fig. 16).

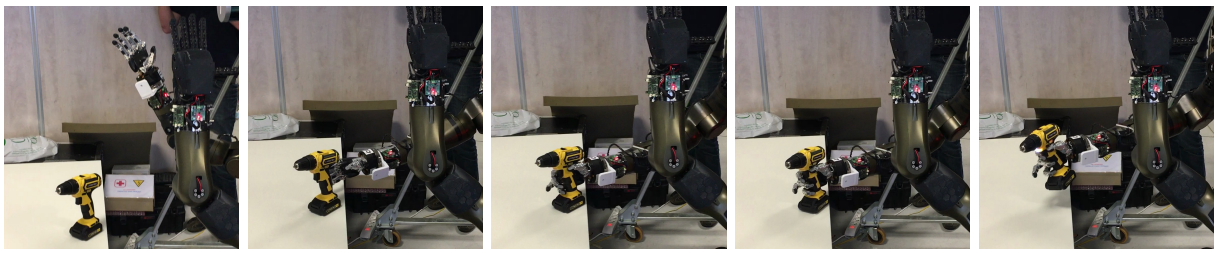


Figure 16: Autonomous drill grasping experiment during the Pisa Integration Meeting.

6 Future Work

The next steps in the project related to this work package can be summarized as:

- Incorporate the Kinect v2 data into trajectory optimization to obtain a collision map of higher density in the goal region.
- Perform experiments for autonomous placing, as only picking was tested.
- Perform experiments with left arm for pick and place of heavier objects.
- Evaluate the system on different kinds of objects.

As in any research project, it is expected that after the evaluation steps some of the current functionalities may have to be modified, extended, or even replaced.

Appendix

The following papers are enclosed below in the order of their appearance in the report:

1. Diego Rodriguez, Corbin Cogswell, Seongyong Koo, and Behnke Sven. Transferring grasping skills to novel instances by latent space non-rigid registration. In *IEEE International Conference on Robotics and Automation (ICRA)*, (Accepted) May 2018

2. Dmytro Pavlichenko and Sven Behnke. Efficient stochastic multicriteria arm trajectory optimization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017

Transferring Grasping Skills to Novel Instances by Latent Space Non-Rigid Registration

Diego Rodriguez, Corbin Cogswell, Seongyong Koo, and Sven Behnke

***Abstract*— Robots acting in open environments need to be able to handle novel objects. Based on the observation that objects within a category are often similar in their shapes and usage, we propose an approach for transferring grasping skills from known instances to novel instances of an object category. Correspondences between the instances are established by means of a non-rigid registration method that combines the Coherent Point Drift approach with subspace methods.**

The known object instances are modeled using a canonical shape and a transformation which deforms it to match the instance shape. The principle axes of variation of these deformations define a low-dimensional latent space. New instances can be generated through interpolation and extrapolation in this shape space. For inferring the shape parameters of an unknown instance, an energy function expressed in terms of the latent variables is minimized. Due to the class-level knowledge of the object, our method is able to complete novel shapes from partial views. Control poses for generating grasping motions are transferred efficiently to novel instances by the estimated non-rigid transformation.

I. Introduction

People can be given a screwdriver that they never saw before, and they will immediately know how to grasp and operate it by transferring previous manipulation knowledge to the novel instance. While this transfer happens effortless in humans, achieving such generalization in robots is challenging. Manipulating novel instances of a known object category is still an open problem. Although the manipulation of known objects can be planned offline, many open-world applications require the manipulation of unknown instances. Our approach transfers manipulations skills to novel instances by means of a novel latent space non-rigid registration (Fig. 1).

Robots are often equipped with RGB-D sensors to perceive their environment in 3D. In order to reconstruct the full shape of an object—desirable for planning grasping—multiple views of the object must be taken and fused into a single 3D model. However, robots are not always able to obtain the required views for generating the full model, because of obstructions or unreachable observation poses, e.g., an object lying at the inner corner of a shelf. Fortunately, it is frequently not necessary to measure occluded object parts if they can be inferred from prior object knowledge.

In this paper, we propose a method for generating grasping motions for novel instances from a single view by making use of category-level extrinsic shape information residing in a learned latent space. Our method accumulates object shape from multiple known instances of a category in a canonical model. The learned latent space of shape variations enables a category-specific shape-aware non-rigid registration procedure that establishes correspondences between the novel view and the canonical model. By linearly interpolating between low-dimensional shape parameters of the known instances, and by extrapolation in this shape space, a manifold of plausible shapes belonging to the modeled category can be generated.

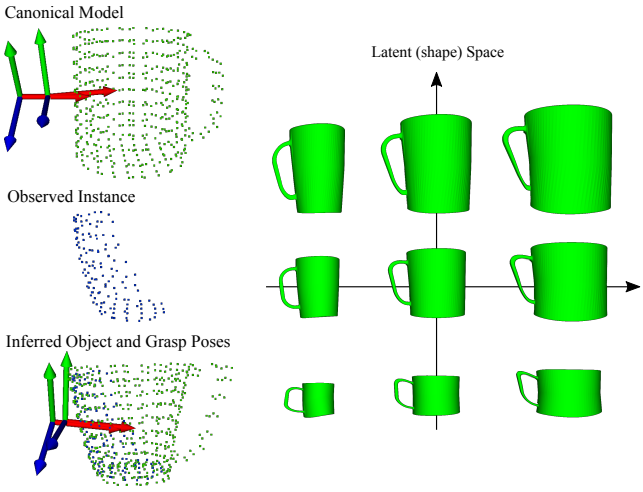


Fig. 1: Control poses are transferred to the shape of a novel instance by latent space non-rigid registration and used to generate a valid grasping motion.

Our method finds a transformation from the canonical model to a view of a novel instance in the latent space—linearly interpolated and extrapolated from other transformations found within the class—which best matches the observed 3D points. This estimates the shape parameters of the novel instance and allows for inference of its occluded parts. The non-rigid transformation maps control poses from the canonical model to the novel instance, which are then used to generate the grasping motion for it.

The remainder of this paper is organized as follows. In the next section, we discuss related works. Section III gives an overview of our approach. The necessary background is then presented in Section IV. In Section V, we describe our approach in detail. We evaluated our method in multiple experiments which are presented in Section VI.

II. Related Work

A. Non-Rigid Registration

Since Chen and Medioni [1] introduced the Iterative Closest Point (ICP) algorithm, numerous variations have

[1]All authors are with the Autonomous Intelligent Systems (AIS) Group, Computer Science Institute VI, University of Bonn, Germany, {rodriguez, koo, behnke}@ais.uni-bonn.de

been proposed for rigid registration [2], [3]. For non-rigid registration, priori restrictions or regularization on the motion or deformation of the points between sets are often imposed. Different transformation priors such as isometry [4]–[6], elasticity [7], conformal maps [8]–[10], thin-plate splines [11], [12], and Motion Coherence Theory [13] have been used to allow for or to penalize different types of transformations.

Many methods use non-rigid registration for surface reconstruction [14]–[19]. Brown and Rusinkiewicz [12] and Li *et al.* [14] proposed surface reconstruction methods for global non-rigid registration. Methods such as Li *et al.* [15] and Zollhöfer *et al.* [20] use a Kinect depth camera to capture an initially low-resolution 3D surface and use non-rigid registration to continuously add higher-frequency details to the model with each new frame.

Recently, Newcombe *et al.* [19] proposed a non-rigid dense surface reconstruction method using dense non-rigid registration. From a sequence of depth images, they calculate a dense 6-dimensional warp field between frames. They then undo the transformations between each frame and rigidly fuse the scans into one canonical shape. The dense warp field is estimated by minimizing an energy function. Although this method is able to deform a scene in real-time toward a canonical shape, the use optical flow constraints makes this method inadequate for large deformations or strong changes in illumination and color.

B. Class-Level Shape Spaces

To create a parameterized space of shapes, several methods have been proposed. Blanz and Vetter [21] create a morphable model of faces able to create novel faces and to interpolate between faces using a few parameters. Similarly, Allen *et al.* [11] create a shape space of human bodies using human body range scans with sparse 3D markers. Hasler *et al.* [22] extend this space to include pose, creating a unified space of both pose and body shape. This allows them to model the surface of a body in various articulated poses more accurately.

Other approaches as the one presented by Nguyen *et al.* [23] establish shape correspondences by creating collections of similar shapes and optimizing the mapping at a global scale. Huang *et al.* [24] also use collections of shapes to enforce global consistency. They create a small collection of base shapes from which correspondences are established between all other shapes.

Burghard *et al.* [25] propose an approach to estimate dense correspondences on a shape given initial coarse correspondences. They use the idea of minimum description length to create a compact shape space of related shapes with strongly varying geometry. However, this method does not perform well in the presence of noise or incomplete scans. Engelmann *et al.* [26] learn a compact shape manifold which represents intra-class shape variance, allowing them to infer shape in occluded regions or regions where data might be missing or noisy such as on textureless, reflective, or transparent surfaces. This approach however does not give correspondences between points and do not offer any kind of transformation, which limits its applicability to transferring grasping knowledge.

C. Transferring Grasping Skills

Stueckler *et al.* [27] proposed a method for manipulation skill transfer using non-rigid registration. The registration finds a non-rigid transformation between a known instance where manipulation information—such as grasp poses or motion trajectories—are available and a novel instance from the same class. The non-rigid transformation is applied to map trajectory control poses towards the newly observed instance, which allows for use of the novel object. We extend this work by modeling shape and grasping not for single known instances, but within a category, which allows for learning typical variations of a canonical model.

Alternative methods as in [28] transfer grasp poses by segmenting the objects in primitive shapes according to their RGB-D appearance. Because this method is based on templates primitives, it is not able to handle occlusions efficiently. In [29] and in [30], grasp poses are transferred using the same contact warp method which minimizes the distance between assigned correspondences. However, both approaches require a fully observed object and focus only on the final result of grasping, namely on grasp poses, while our approach addresses the process of grasping.

III. Problem Statement

We propose a new approach for transferring grasping skills based on a novel method for non-rigid registration. This method incorporates category-level information by learning a shape space of the category. By incorporating this information into the registration, we can avoid unlikely shapes and focus on deformations actually observed within the class. Thus, the resulting grasping motion is able to tolerate the noise and occlusions typical of data measured by 3D sensors. Furthermore, the generated space can be used to create novel instances and to interpolate and extrapolate between previous ones.

Our method is divided into two phases: a learning phase and an inference phase (Fig. 2 and 3). In the learning phase, the objective is to create a class-specific linear model of the transformations that a class of objects can undergo. We do this by first selecting a single model to be a canonical instance of the class, and then we find the transformations relating this instance to all other instances of the class using Coherent Point Drift (CPD) [13]. We then find a linear subspace of these transformations, which becomes our transformation model for the class. In the inference phase, the objective is: given a newly observed instance, search this subspace of transformations to find the transformation which best relates the canonical shape to the observed instance. Grasping information from the canonical model is also transformed to the observed instance and used to generate the grasping motion of the robot.

The transformations are represented by a dense deformation field plus a single rigid transformation; the latter is

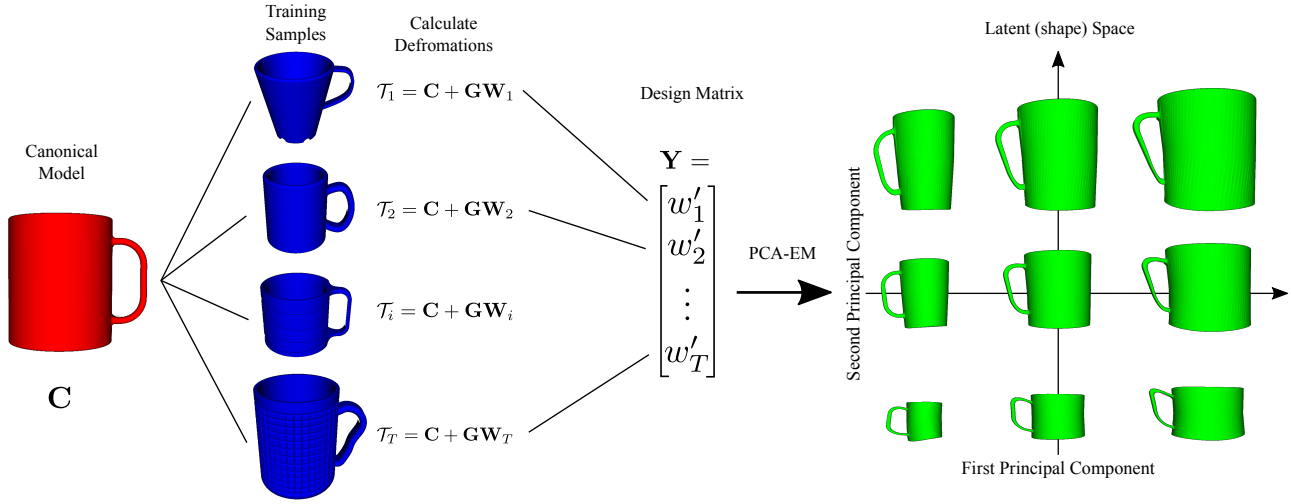


Fig. 2: Learning of the latent space. First, the deformations $\mathbf{W}_i$ between each instance $\mathcal{T}_i$ and the canonical model $\mathbf{C}$ are calculated using CPD. These deformations are then expressed as column vectors $\mathbf{w}_i$ and assembled into the design matrix $\mathbf{Y}$. Using PCA-EM, the principal components which constitute the latent shape space are extracted.

included to account for small global misalignments. This allows points defined in the space of the canonical shape to be transformed into the space of an observed instance. These points do not need to be known a priori for the learning phase and can be added at any time after the registration has completed.

The main contributions of this paper are:

1) a novel learned non-rigid registration method that finds a low-dimensional space of deformation fields; this method can be used to interpolate and extrapolate between instances of a class of objects to create novel instances and
2) the application of this non-rigid registration method to transfer grasping skills.

IV. Coherent Point Drift

We briefly review the Coherent Point Drift (CPD) [13] method on which our non-rigid registration is based.

Given two point sets, a template set $\mathbf{S}^{[t]} = (\mathbf{s}_1^{[t]}, ..., \mathbf{s}_M^{[t]})^T$ and a reference point set $\mathbf{S}^{[r]} = (\mathbf{s}_1^{[r]}, ..., \mathbf{s}_N^{[r]})^T$, CPD tries to estimate a deformation field mapping the points in $\mathbf{S}^{[t]}$ to $\mathbf{S}^{[r]}$. The points in $\mathbf{S}^{[t]}$ are considered centroids of a Gaussian Mixture Model (GMM) from which the points in $\mathbf{S}^{[r]}$ are drawn. CPD seeks to maximize the likelihood of the GMM while imposing limitations on the motion of the centroids. CPD imposes a smoothness constraint on the deformation of the points in the form of motion coherence, which is based on Motion Coherence Theory [31]. The idea is that points near each other should move coherently and have a similar motion to their neighbors. Maximizing the likelihood of the GMM is equivalent to minimizing the energy function:

$$E(\mathbf{S}^{[t]}, \psi) = -\sum_{n=1}^{N} \log \sum_{m=1}^{M} e^{-\frac{1}{2\sigma^2}\left\|\mathbf{s}_n^{[r]} - \mathcal{T}(\mathbf{s}_m^{[t]}, \psi)\right\|^2} + \frac{\lambda}{2}\phi(\mathbf{S}^{[t]}) \tag{1}$$

where $\mathcal{T}(\mathbf{s}_m^{[t]}, \psi)$ is a parametrized transformation from the template point set to the reference set. The first term of Eq. (1) penalizes the distance between points after applying the transformation $\mathcal{T}$, and the second term is a regularization term which enforces motion coherence.

For the non-rigid case, the transformation $\mathcal{T}$ is defined as the initial position plus a displacement function v:

$$\mathcal{T}(\mathbf{S}^{[t]}, v) = \mathbf{S}^{[t]} + v(\mathbf{S}^{[t]}), \tag{2}$$

where v is defined for any set of D-dimensional points $\mathbf{Z}_{N\times D}$ as:

$$v(\mathbf{Z}) = G(\mathbf{S}^{[\mathbf{t}]}, \mathbf{Z})\mathbf{W}. \tag{3}$$

$G(\mathbf{S}^{[\mathbf{t}]}, \mathbf{Z})$ is a Gaussian kernel matrix which is defined element-wise as:

$$g_{ij} = G(\mathbf{s}_i^{[t]}, \mathbf{z}_j) = \exp^{-\frac{1}{2\beta^2}\left\|\mathbf{s}_i^{[t]} - \mathbf{z}_j\right\|^2}, \tag{4}$$

$\mathbf{W}_{M\times D}$ is a matrix of kernel weights, and β is a parameter that controls the strength of interaction between points. An additional interpretation of $\mathbf{W}$ is as a set of D-dimensional deformation vectors, each associated with one of the M points of $\mathbf{S}^{[t]}$. For convenience in the notation, $\mathbf{G}_{M\times M}$ will be denoted $\mathbf{G}(\mathbf{S}^{[t]}, \mathbf{S}^{[t]})$. Note that $G(\cdot,\cdot)$ can simply be computed by Eq. (4), but the matrix $\mathbf{W}$ needs to be estimated.

CPD uses an Expectation Maximization (EM) algorithm derived from the one used in Gaussian Mixture Models [32] to minimize Eq. (1). In the E-step, the posterior probabilities matrix $\mathbf{P}$ is estimated using the previous parameter values. To add robustness to outliers, an additional uniform probability distribution is added to the mixture model. This matrix $\mathbf{P}$ is defined element-wise as:

$$p_{mn} = \frac{e^{-\frac{1}{2\sigma^2}\left\|\mathbf{s}_n^{[r]} - (\mathbf{s}_m^{[t]} + G(m,\cdot)\mathbf{W})\right\|^2}}{\sum_{m=1}^{M} e^{-\frac{1}{2\sigma^2}\left\|\mathbf{s}_n^{[r]} - (\mathbf{s}_m^{[t]} + G(k,\cdot)\mathbf{W})\right\|^2} + \frac{\omega}{1-\omega}\frac{(2\pi\sigma^2)^{\frac{D}{2}}}{N}} \tag{5}$$

where ω reflects the assumption on the amount of noise.

In the M-step, the matrix $\mathbf{W}$ is estimated by solving the equation:

$$(\mathbf{G} + \lambda\sigma^2 d(\mathbf{P1})^{-1})\mathbf{W} = d(\mathbf{P1})^{-1}\mathbf{P}\mathbf{S}^{[r]} - \mathbf{S}^{[t]} \quad (6)$$

where $\mathbf{1}$ represents a column vector of ones and $d(\cdot)^{-1}$ is the inverse diagonal matrix.

In our method, we use the canonical shape $\mathbf{C}$ for the deforming template shape $\mathbf{S}^{[t]}$ and each training example $\mathbf{T}_i$ as the reference point set $\mathbf{S}^{[r]}$. Hence, the transformations $\mathcal{T}_i$ are defined as

$$\mathcal{T}_i(\mathbf{C}, \mathbf{W}_i) = \mathbf{C} + \mathbf{G}\mathbf{W}_i \quad (7)$$

where $\mathbf{W}_i$ is the $\mathbf{W}$ matrix computed by taking training example $\mathbf{T}_i$ as the reference point set $\mathbf{S}^{[r]}$.

V. Method

Our learning-based approach has an initial training phase in which the latent space for a class of objects is built. With the latent space of the object class, inference can be performed to find a transformation which relates one of the shapes from the class called the canonical shape to fully or partially-observed novel instances. The transformation is represented by a dense deformation field plus a single rigid transformation. Finally, we transform all the grasping information to the new instance.

A. Categories and Shape Representation

We define a category or class as a set of objects which share the same topology and a similar extrinsic shape. A category is composed of a set of training sample shapes and a canonical shape $\mathbf{C}$ that will be deformed to fit the shape of the training and observed samples. To represent a shape, we use point clouds, which can be generated from meshes by ray-casting from several viewpoints on a tessellated sphere and then down-sampling with a voxel grid filter. The canonical shape can be chosen by heuristics or chosen as the shape with the lowest reconstruction energy after finding the deformation field of all other training examples. Fig. 4 shows both the mesh and the resulting uniformly-sampled point clouds of object samples of two different categories.

Each class must specify a canonical pose and reference frame, which are essential for initial alignment. For example: for *Mugs*, we can align all instances such that their top is open upward with all the handles pointing towards the right.

B. Low-Dimensional Deformation Field Manifold

Once we have a set of training examples and a canonical shape, we need to find the deformations from the canonical shape to all other training shapes. Here we make use of the Coherent Point Drift method [13].

CPD provides a dense deformation field, allowing us to find deformation vectors for novel points, even those added after the field is created, which in turn allows us to apply the method for transferring grasping skills. Additionally, CPD allows us to create a feature vector representing the deformation field. Importantly, this vector has the same length for all training examples and elements in the vector correspond with the same elements in another. This will allow us to construct a latent space in later steps.

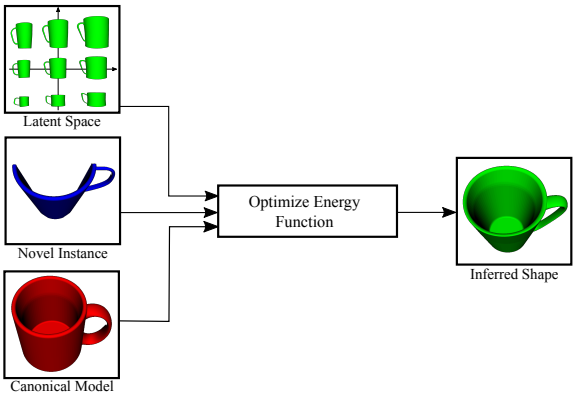


Fig. 3: The canonical shape (red) is matched against a partially-occluded target shape (blue) by optimizing Eq. (12) in terms of the latent parameters plus a rigid transformation. The resulting shape is shown in green on the right.

From Eq. (7), we see that the deformation field function is defined by the matrices $\mathbf{G}$ and $\mathbf{W}$. Moreover, from Eq. (4) we see that $\mathbf{G}$ is a function only of the canonical shape and remains constant for all training examples. Thus, the entire uniqueness of the deformation field for each training example is captured by the matrix $\mathbf{W}$.

For each training example $\mathbf{T}_i$, we take the matrix $\mathbf{W}_i$ from its deformation field and convert it into a row vector $\mathbf{y}_i \in \mathbb{R}^{p=M\times D}$, which becomes the feature descriptor of that deformation field. The vectors are then normalized resulting in zero-mean and unit-variance vectors which are then assembled into a design matrix $\mathbf{Y}$. Finally, we apply Principle Component Analysis (PCA) on this matrix to find a lower-dimensional manifold of deformation fields for this class.

PCA finds a matrix $\mathbf{L}_{p\times q}$ of principle components that can be used to estimate a matrix of n observation vectors $\hat{\mathbf{Y}}_{n\times p}$ given a small set of q latent variables, i.e., $q \ll p$:

$$\hat{\mathbf{Y}} = \mathbf{X}\mathbf{L}^T. \quad (8)$$

For a new normalized set of observations $\mathbf{Y_o}$, the latent variables can be found by:

$$\mathbf{X} = \mathbf{Y_o}\mathbf{L}. \quad (9)$$

We find the matrix L using the PCA Expectation Maximization (PCA-EM) algorithm [33]. We choose this method over analytical algorithms due to its superior performance in situations with high dimensions and scarce data and also because of its greater numerical stability.

Much like with CPD, we alternate between an E- and M-step. The E-step is given by:

$$\mathbf{X} = \mathbf{Y}\mathbf{L}^T(\mathbf{L}\mathbf{L}^T)^{-1} \quad (10)$$

whereas the M-step is defined by:

$$\mathbf{L} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}. \quad (11)$$

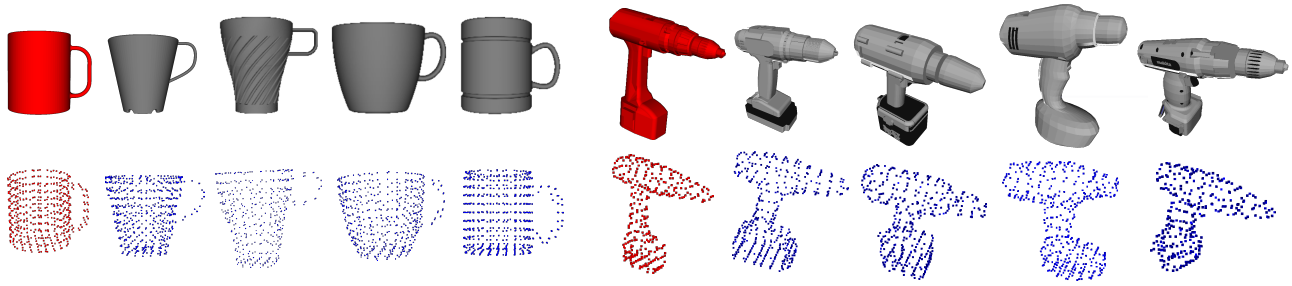


Fig. 4: Original meshes and the uniformed sampled point clouds of instances of the *Mug* and *Drill* categories. The canonical models are colored red.

This method is shown to converge to a local minimum using standard EM convergence proofs [34]. Additionally, it has been shown that the only stable local extremum is the global maximum [35], [36], meaning the algorithm will always converge to the correct result with enough iterations.

Using Eq. (9), a deformation field can now be described by only q latent parameters. Similarly, any point $\mathbf{x}$ in the latent space can be converted into a deformation field transformation by first applying Eq. (8) and converting the result into a $\mathbf{W}_{M\times D}$ matrix after the respective denormalization. Thus, moving through the q-dimensional space linearly interpolates between the deformation fields.

The matrix $\mathbf{L}$ and the canonical shape $\mathbf{C}$ together represent the transformation model for a class. Figure 2 gives an overview of the training phase which is also summarized in Algorithm 1.

C. Shape Inference for Novel Instances

With the transformation model, we can now start registering the canonical shape to novel instances in order to estimate the underlying transformation. The parameters of the transformation are given by the q parameters of the latent vector $\mathbf{x}$ plus an additional seven parameters of a rigid transformation $\boldsymbol{\theta}$. The rigid transformation is meant to account for minor misalignments in position and rotation between the target shape and the canonical shape at the global level.

We concurrently optimize for shape and pose using gradient descent. We expect local minima, especially with regard to pose, therefore our method, as CPD and ICP, requires an initial coarse alignment of the observed shape. We want to find an aligned dense deformation field which when applied to the canonical shape $\mathbf{C}$, minimizes the distance between corresponding points in the observed shape $\mathbf{O}$. Specifically, we want to minimize the energy function:

$$E(\mathbf{x},\boldsymbol{\theta}) = -\sum_{m=1}^{M}\log\sum_{n=1}^{N} e^{\frac{1}{2\sigma^2}\|\mathbf{O}_n-\Theta(\mathcal{T}_m(\mathbf{C}_m,\mathbf{W}_m(\mathbf{x})),\boldsymbol{\theta})\|^2}, \tag{12}$$

where the function Θ applies the rigid transformation given $\boldsymbol{\theta}$ parameters.

When the minimum is found, we can transform any point or set of points into the observed space by applying the deformation field using Eq. (3) and Eq. (2) and then applying the rigid transformation Θ. Algorithm 2 summarizes the inference process.

Algorithm 1 Building the latent space for a Category

Input: A set of training shapes $\mathbf{E}$ in their canonical pose and reference frame

1: Select a canonical shape $\mathbf{C}$ via heuristic or pick the one with the lower reconstruction energy.
2: Estimate the deformation fields between the canonical shape and the other training examples using CPD.
3: Concatenate the resulting set of $\mathbf{W}$ matrices from the deformation fields into a design matrix $\mathbf{Y}$.
4: Perform PCA on the design matrix $\mathbf{Y}$ to compute the latent space of deformation fields.

Output: A canonical shape $\mathbf{C}$ and a latent space of deformation fields represented by $\mathbf{L}$.

D. Transferring Grasping Skills

From experiences with an anthropomorphic robotic hand, namely the multi-fingered Schunk hand [37], we observed that a functional grasping cannot be described only by the final joint configuration of the entire system (manipulator plus hand). During the motion, the hand can make use of geometrical and frictional constraints to increase the chances of a successful grasping. For instance, for grasping drills, we can first make a lateral contact with the palm followed by a contact of the thumb with the rear part of tool, creating static frictional forces that may help during the grasp. Therefore, we use the term *grasping* when we refer to the complete action and the term *grasp* for the result of this action.

We represent a grasping action as a set of parametrized motion primitives. The parameters of the motion primitives are poses expressed in the same coordinate system of the shape of the object. These poses need to be defined only for the canonical model. For new instances, the poses are found by warping the poses of the canonical model to the instance. Because the warping process can violate the orthogonality of the orientation, we orthonormalize the warped quaternions. Additional parameters of the motion primitives such as velocities and accelerations can also be derived from the warped poses.

Algorithm 2 Shape Inference for a Novel Instance

Input: Transformation model ($\mathbf{C}$, $\mathbf{L}$) and observed shape $\mathbf{O}$

1: Compute matrix $\mathbf{G} = G(\mathbf{C}, \mathbf{C})$ (Eq. 4).
2: Use gradient descent to estimate the parameters of the underlying transformation ($\mathbf{x}$ and $\boldsymbol{\theta}$) until the termination criteria is met. To calculate the value of the energy function, in each iteration:
- Using the current values of $\mathbf{x}$ and $\boldsymbol{\theta}$:
1) Use Eq. (8) to create vector $\hat{\mathbf{Y}}$ and convert it into matrix $\mathbf{W}$.
2) Use Eq. (3) and Eq. (2) to deform $\mathbf{C}$.
3) Apply the rigid transformation Θ to the deformed $\mathbf{C}$.

Output: Non-rigid transformation given by deformation field description $\mathbf{W}$ and rigid transform $\boldsymbol{\theta}$.

VI. Experimental Results

We tested our method on two categories: *Mugs* and *Drills*. We obtained the models from two online CAD databases: GrabCad[1] and the 3DWarehouse[2]. The meshes were converted into point clouds by ray-casting from several viewpoints on a tessellated sphere and down-sampling with a voxel grid filter. We trained the *Mug* and *Drill* models with their canonical shape and additional 21 and 9 samples, respectively. The meshes of some of the samples together with the uniformly sampled point clouds are shown in Fig. 4.

We evaluated the robustness of our method to noise, misalignment, and occlusion and compared our results against results given by CPD. Noise was added to each point by randomly sampling a point from a normal distribution and scaling it by a noise factor. Misalignment was generated by adding a rigid transformation to the observed shape. For the translation, we uniformly sampled a three-dimensional unit vector and multiplied it by the following factors: $[0.01, 0.02, 0.03, 0.04, 0.05]$. For the orientation, a three-dimensional unit rotation axis was uniformly sampled and combined with the following angles: $[\pi/4, \pi/8, 3\pi/16, \pi/4, 3\pi/8]$, making use of the axis-angle representation. Each test was run on a full view of the object and on six different partial views of the object. To obtain the partial views, we used ray-casting on a single view of a tessellated sphere.

In all experiments, we parametrized CPD with the following values $\beta = 1$ and $\lambda = 3$. For creating the shape space, the number of latent variables was set to capture at least 95% of the variance of each class, and the class canonical shapes were selected by experts.

Fig 5 shows, for both categories, how the canonical shape is deformed to a single view of an observed instance. Note that our method is able to reconstruct occluded parts, as for example, the handles of the mugs.

For the evaluation, we take the noiseless fully-observed shape as the ground truth and the following error function:

$$E(\mathbf{D}, \mathbf{O}^*) = \frac{1}{N} \sum_{n=1}^{N} \min_m \left(\|\mathbf{O}_n^* - \mathbf{D}_m\|^2 \right) \qquad (13)$$

where $\mathbf{D}$ is the transformed canonical shape and $\mathbf{O}^*$ is the ground truth shape. To compute the final error, we average the errors resulting from each partial view and from each test sample.

The results of our experiments are plotted in Fig. 7. We use the abbreviation CLS (Categorical Latent Space) for referring to our method. When a shape is fully visible with or without noise, CPD outperforms our method. This is rather to be expected as CPD is the source of training data from which we build our registration model. On the other hand, when the observed shape is misaligned, our method outperforms CPD, which can be explained by the additional rigid transformation component of our method. Moreover, when the shape becomes partially occluded, the method also outperforms CPD thanks to the topological information that lies in the latent space, which is not available in CPD.

For testing the grasping transfer, we conducted a set of experiments using the five-fingered Schunk hand, which has a total of 20 Degrees of Freedom (DoF) from which 9 are fully actuated. The experiments were carried out in the Gazebo simulation environment. The tests were performed on the *Drill* category. Due to the reduced number of meshes, we used cross-validation and created five transformation models leaving two testing samples for each transformation model. A trial was defined as successful if after the execution

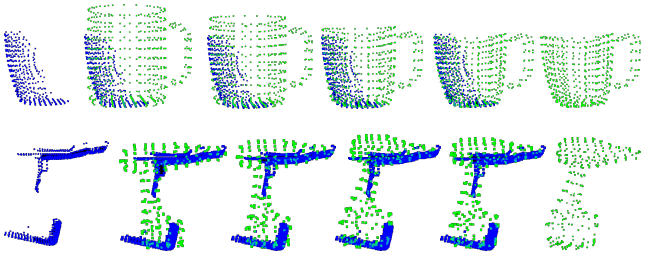


Fig. 5: Given a partial view of the object (leftmost), the canonical objects are deformed for the *Mug* and *Drill* categories. The resulting point cloud is shown rightmost.

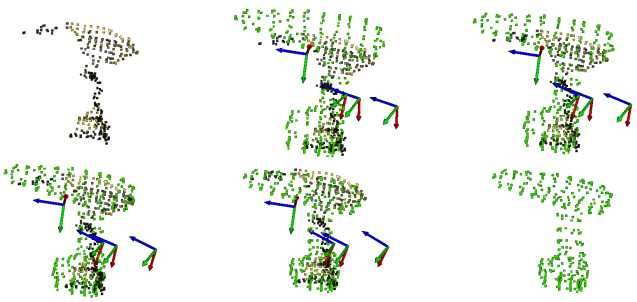


Fig. 6: Registration of a partially occluded point cloud of an object coming from real sensory data. The input point cloud is at the top leftmost while the inferred shape is at the bottom rightmost.

[1] https://grabcad.com/library
[2] https://3dwarehouse.sketchup.com/

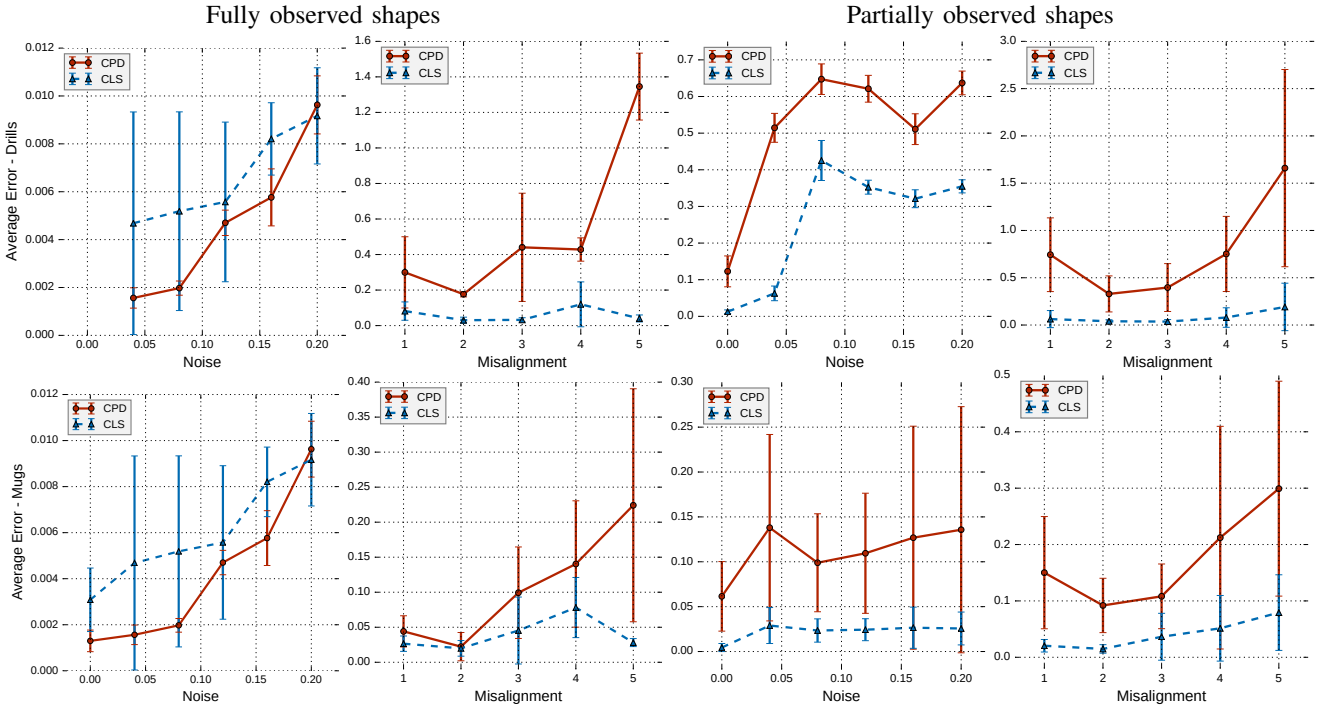


Fig. 7: A comparison of our method (CLS dotted blue line) against CPD (red line). The first row of plots corresponds to the *drill* category while the second row refers to the *cup* category. On each case, the first two plots show the results on fully observed shapes, while the last two, on partially observed shapes. Our method outperforms CPD on partial views of objects and misaligned fully observed shapes.

of the motion, the drill was held by the robotic hand. We got a success rate of 0.8 on grasping the drills.

Failure cases occurred mainly for one reason, i.e., the size of the testing object was too small compared to the canonical shape. This indicated that the transformation model was not general enough, mainly due to the reduced number of training samples. The small size of the objects also caused collisions between the fingers at the moment of grasping the handle. However, we expect to tackle this problem in the future by inferring information from the shape such as the number of fingers required for grasping.

We tested our approach with real sensory data coming from a Kinect v2 sensor [38]. We trained a single latent space model with 9 training samples and the canonical model. Our method was able to generate plausible category-alike shapes and grasping poses in each of three different trials. The inference in average took 6 s on a Corei7 CPU 2.6 GHz with 16GB RAM. This demonstrates that our method can be used on-board in real robotic platforms. The result of the registration is presented in Fig. 6.

VII. Conclusion

We presented a novel approach for transferring grasping skills based on a latent space non-rigid registration method, which is able to handle fully observed and partially observed shapes. We demonstrated its robustness especially to noise and occlusion. We evaluated our method on two sets of shapes and found that while the method performed slightly worse than conventional CPD on noiseless, fully observed shapes, when shapes were partially occluded, our method was much more able to recover the true underlying shape and reported lower average error. Successful application of the non-rigid registration for transferring grasping skills was also demonstrated both with synthetic and real sensory data.

In the future, we would like to extend our approach to more complex objects that require a part-based modeling approach.

Acknowledgements

This work was supported by the European Union's Horizon 2020 Programme under Grant Agreement 644839 (CENTAURO) and the German Research Foundation (DFG) under the grant BE 2556/12 ALROMA in priority programme SPP 1527 Autonomous Learning.

References

[1] Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image and Vision Computing*, vol. 10, no. 3, pp. 145–155, 1992.

[2] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *3-D Digital Imaging and Modeling. Proceedings of Third International Conference on*, IEEE, 2001, pp. 145–152.

[3] D. Holz, A. E. Ichim, F. Tombari, R. B. Rusu, and S. Behnke, "Registration with the Point Cloud Library: A modular framework for aligning in 3-D," *IEEE Robotics & Automation Magazine*, vol. 22, no. 4, pp. 110–124, 2015.

[4] A. M. Bronstein, M. M. Bronstein, and R. Kimmel, "Efficient computation of isometry-invariant distances between surfaces," *SIAM Journal on Scientific Computing*, vol. 28, no. 5, pp. 1812–1836, 2006.

[5] A. Tevs, M. Bokeloh, M. Wand, A. Schilling, and H.-P. Seidel, "Isometric registration of ambiguous and partial data," in *Computer Vision and Pattern Recognition, CVPR. IEEE Conference on*, IEEE, 2009, pp. 1185–1192.

[6] M. Ovsjanikov, Q. Mérigot, F. Mémoli, and L. Guibas, "One point isometric matching with the heat kernel," in *Computer Graphics Forum*, Wiley Online Library, vol. 29, 2010, pp. 1555–1564.

[7] D. Hahnel, S. Thrun, and W. Burgard, "An extension of the ICP algorithm for modeling nonrigid objects with mobile robots," in *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, Acapulco, Mexico, 2003, pp. 915–920.

[8] B. Lévy, S. Petitjean, N. Ray, and J. Maillot, "Least squares conformal maps for automatic texture atlas generation," in *ACM Transactions on Graphics (TOG)*, vol. 21, 2002, pp. 362–371.

[9] Y. Zeng, C. Wang, Y. Wang, X. Gu, D. Samaras, and N. Paragios, "Dense non-rigid surface registration using high-order graph matching," in *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, IEEE, 2010, pp. 382–389.

[10] V. Kim, Y. Lipman, and T. Funkhouser, "Blended intrinsic maps," *ACM Transactions on Graphics (Proc. SIGGRAPH)*, vol. 30, no. 4, Jul. 2011.

[11] B. Allen, B. Curless, and Z. Popović, "The space of human body shapes: Reconstruction and parameterization from range scans," in *ACM Transactions on Graphics (TOG)*, vol. 22, 2003, pp. 587–594.

[12] B. J. Brown and S. Rusinkiewicz, "Global non-rigid alignment of 3-d scans," in *ACM Transactions on Graphics (TOG)*, ACM, vol. 26, 2007, p. 21.

[13] A. Myronenko and X. Song, "Point set registration: Coherent point drift," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 12, pp. 2262–2275, 2010.

[14] H. Li, R. W. Sumner, and M. Pauly, "Global correspondence optimization for non-rigid registration of depth scans," in *Computer Graphics Forum*, Wiley Online Library, vol. 27, 2008, pp. 1421–1430.

[15] H. Li, B. Adams, L. J. Guibas, and M. Pauly, "Robust single-view geometry and motion reconstruction," in *ACM Transactions on Graphics (TOG)*, vol. 28, 2009, p. 175.

[16] J. Süßmuth, M. Winter, and G. Greiner, "Reconstructing animated meshes from time-varying point clouds," in *Computer Graphics Forum*, Wiley Online Library, vol. 27, 2008, pp. 1469–1476.

[17] M. Wand, P. Jenke, Q. Huang, M. Bokeloh, L. Guibas, and A. Schilling, "Reconstruction of deforming geometry from time-varying point clouds," in *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, Barcelona, Spain, 2007, pp. 49–58.

[18] M. Wand, B. Adams, M. Ovsjanikov, A. Berner, M. Bokeloh, P. Jenke, L. Guibas, H.-P. Seidel, and A. Schilling, "Efficient reconstruction of nonrigid shape and motion from real-time 3D scanner data," *ACM Transactions on Graphics (TOG)*, 2009.

[19] R. A. Newcombe, D. Fox, and S. M. Seitz, "Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 343–352.

[20] M. Zollhöfer, M. Nießner, S. Izadi, C. Rehmann, C. Zach, M. Fisher, C. Wu, A. Fitzgibbon, C. Loop, C. Theobalt, *et al.*, "Real-time non-rigid reconstruction using an RGB-D camera," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, p. 156, 2014.

[21] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3D faces," in *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, ACM Press/Addison-Wesley Publishing Co., 1999, pp. 187–194.

[22] N. Hasler, C. Stoll, M. Sunkel, B. Rosenhahn, and H.-P. Seidel, "A statistical model of human pose and body shape," in *Computer Graphics Forum*, Wiley Online Library, vol. 28, 2009, pp. 337–346.

[23] A. Nguyen, M. Ben-Chen, K. Welnicka, Y. Ye, and L. Guibas, "An optimization approach to improving collections of shape maps," in *Computer Graphics Forum*, Wiley Online Library, vol. 30, 2011, pp. 1481–1491.

[24] Q.-X. Huang, G.-X. Zhang, L. Gao, S.-M. Hu, A. Butscher, and L. Guibas, "An optimization approach for extracting and encoding consistent maps in a shape collection," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 6, p. 167, 2012.

[25] O. Burghard, A. Berner, M. Wand, N. Mitra, H.-P. Seidel, and R. Klein, "Compact part-based shape spaces for dense correspondences," *ArXiv preprint arXiv:1311.7535*, 2013.

[26] F. Engelmann, J. Stückler, and B. Leibe, "Joint object pose estimation and shape reconstruction in urban street scenes using 3D shape priors," in *German Conference on Pattern Recognition*, Springer, 2016, pp. 219–230.

[27] J. Stueckler, R. Steffens, D. Holz, and S. Behnke, "Real-time 3D perception and efficient grasp planning for everyday manipulation tasks.," in *ECMR*, 2011, pp. 177–182.

[28] N. Vahrenkamp, L. Westkamp, N. Yamanobe, E. E. Aksoy, and T. Asfour, "Part-based grasp planning for familiar objects," in *IEEE-RAS 16th International Conference on Humanoid Robots*, 2016, pp. 919–925.

[29] T. Stouraitis, U. Hillenbrand, and M. A. Roa, "Functional power grasps transferred through warping and replanning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 4933–4940.

[30] H. B. Amor, O. Kroemer, U. Hillenbrand, G. Neumann, and J. Peters, "Generalization of human grasping for multi-fingered robot hands," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 2043–2050.

[31] A. L. Yuille and N. M. Grzywacz, "The motion coherence theory," in *Computer Vision., Second International Conference on*, IEEE, 1988, pp. 344–353.

[32] C. M. Bishop, *Neural networks for pattern recognition*. Oxford university press, 1995.

[33] S. T. Roweis, "Em algorithms for pca and spca," in *Advances in Neural Information Processing Systems 10*, M. I. Jordan, M. J. Kearns, and S. A. Solla, Eds., MIT Press, 1998, pp. 626–632.

[34] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977.

[35] M. E. Tipping and C. M. Bishop, "Mixtures of probabilistic principal component analyzers," *Neural computation*, vol. 11, no. 2, pp. 443–482, 1999.

[36] ——, "Probabilistic principal component analysis," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 61, no. 3, pp. 611–622, 1999.

[37] S. W. Ruehl, C. Parlitz, G. Heppner, A. Hermann, A. Roennau, and R. Dillmann, "Experimental evaluation of the schunk 5-finger gripping hand for grasping tasks," in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2014, pp. 2465–2470.

[38] P. Fankhauser, M. Bloesch, D. Rodriguez, R. Kaestner, M. Hutter, and R. Siegwart, "Kinect v2 for mobile robot navigation: Evaluation and modeling," in *International Conference on Advanced Robotics (ICAR)*, 2015, pp. 388–394.

Efficient Stochastic Multicriteria Arm Trajectory Optimization

Dmytro Pavlichenko and Sven Behnke

***Abstract*— Performing manipulation with robotic arms requires a method for planning trajectories that takes multiple factors into account: collisions, joint limits, orientation constraints, torques, and duration of a trajectory. We present an approach to efficiently optimize arm trajectories with respect to multiple criteria. Our work extends Stochastic Trajectory Optimization for Motion Planning (STOMP). We optimize trajectory duration by including velocity into the optimization. We propose an efficient cost function with normalized components, which allows prioritizing components depending on user-specified requirements. Optimization is done in two stages: first with a partial cost function and in the second stage with full costs. We compare our method to state-of-the art methods. In addition, we perform experiments on real robots: centaur-like robot Momaro and an industrial manipulator.**

I. Introduction

Autonomous robots are required to interact with unstructured environments. This introduces a need for manipulation trajectory planning. A feasible trajectory is expected to satisfy multiple criteria: being collision-free, being within joint limits, and being smooth. In addition, minimization of torques may allow to operate longer with limited power supply and to operate heavy objects with more safety for the motors. Minimized duration of a trajectory allows finishing a task faster. In addition, orientation constraints are often required when manipulating orientation-dependent objects. Planning a trajectory considering all these factors is a challenging task.

In this paper, we address these challenges by extending a popular optimization method: Stochastic Trajectory Optimization for Motion Planning (STOMP) [1]. In the original STOMP, the duration of the trajectory is predefined and fixed. We propose a way to optimize duration as well. We further introduce a novel cost computation policy that allows to take full control over the computational effort, which lowers the runtime. We propose a cost function with five components: collisions, joint limits, orientation constraints, joint torque, and duration minimization. Each cost component is normalized to have values in the interval $[0, 1]$. By introducing an importance weight for each component, it is possible to set priorities, which allows obtaining qualitatively different trajectories according to the user preferences. Optimization is performed in two phases to reduce computations: with simplified and full cost functions (Fig. 1).

We evaluate our approach by performing experiments of different difficulty levels in simulation in comparison with three state-of-the-art planning methods. We demonstrate the effects of optimization of different cost components. Finally, we perform experiments with real robots to demonstrate that our approach can be applied to real-world problems.

The authors are with the Autonomous Intelligent Systems (AIS) Group, Computer Science Institute VI, University of Bonn, Germany, {pavlichenko, behnke}@ais.uni-bonn.de

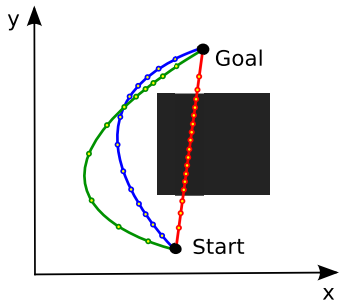


Fig. 1. Two-phased optimization utilized in our approach. Red: initial trajectory, going through the obstacle; Blue: result of the first phase – collision-free trajectory; Green: result of the second phase – trajectory optimized with full costs. Points on trajectories correspond to collision checks. We utilize adaptive collision checking density: closer to the obstacles collision checking is performed with increased density.

II. Related Work

Manipulation planning has been investigated by many researchers, as it is essential for a vast range of autonomous manipulation robots. Sampling-based methods [2], [3] are popular for addressing planning problems. One example are Rapidly-Exploring Random Trees (RRTs) [4]. James et al. [5] utilize RRTs for planning feasible trajectories for a robotic arm. This method produces non-smooth trajectories which are far from being optimal. Thus, a postprocessing step is needed.

RRTs find paths from a start configuration to any configuration in the search space to find an optimal solution. However, this approach introduces many unnecessary computations. Gammell et al. [6] represent a set of perspective solutions as a prolate hyperspheroid. Unnecessary exhaustive search across the whole search space is avoided by this technique.

Batch Informed Trees (BIT*) [7] combine incremental graph search and sampling-based techniques. The initial ellipsoid subset of samples is incrementally expanded with new batches of configurations. This allows exploring a search space and finding an optimal solution. BIT* utilizes a heuristic in order to bias the search towards potential improvement of the solution.

The Fast Marching Tree algorithm (FMT*) [8] is a probabilistic sampling-based planning method. The approach is a combination of single-query and multiple-query algorithms. Lazy dynamic programming recursion is utilized in order to

grow a tree of trajectories. This technique is aimed to reduce the number of collision checks. The method was proven to be asymptotically optimal.

Bidirectional Informed RRT* (BI^{2}RRT*) [9] is an extension of Informed RRT*. Bi-directional search is utilized in order to find an optimal solution faster. A greedy connect heuristic is used in order to find an initial feasible solution quickly. The method enables the use of task-specific constraints through first-order retraction [10].

Another work based on RRT* is Ball Tree + RRT* (BT+RRT*) [11]. The approach combines the Ball Tree algorithm with RRT*. The Ball Tree algorithm maintains a tree in similar to RRT manner. However, each vertex is represented by a ball in the configuration space instead of a point. This allows obtaining solutions using much sparser trees. BT+RRT* utilizes a memoization technique, which allows to reduce the amount of collision checks. The method achieved lower runtime than RRT and RRT*. However, these methods only take into account a geometry of a robot. In [12] a method for acceleration-limited planning, based on RRT is presented. Non-iterative steering method is defined, which allows to take into account velocities and accelerations and find feasible solutions fast. The method is capable of solving problems with non-zero initial and/or goal velocities.

Optimization-based methods are used in order to obtain smooth trajectories which are optimized with respect to the required criteria. In Covariant Hamiltonian Optimization for Motion Planning (CHOMP) [13], a covariant gradient technique is used, which requires a gradient of the cost function. The idea is similar to the elastic bands planning [14], where the trajectory is pushed away from the obstacles by repelling forces. CHOMP quickly converges to the locally optimal trajectory. A signed distance field is used as an environment representation, which allows obtaining gradients even for non-collision-free parts of the trajectory. However, as many gradient-based methods, CHOMP suffers from local minima.

T-CHOMP [15] is an extension of CHOMP. The configuration space is extended by one dimension for time. This allows to optimize in space-time. The idea behind this work is similar to ours, as we add one dimension representing time as well. The authors mention that the implementation is very sensitive to parameters and without fine tuning it may output a "collision-free" solution where the robot slowly goes through the obstacles. We avoid this effect by designing a cost function which penalizes such situations.

STOMP [1] adopted the environment representation from CHOMP. However, instead of using the gradient of the cost function, it uses a sampling technique for cost minimization. This allows to use non-differentiable cost functions and decreases the risk of being stuck in a local minimum.

Local multiresolution for STOMP was proposed by Steffens et al. [16]. The initial part of the trajectory is planned with a high resolution and parts of the trajectory which are located further away in time are planned with lower resolution. This allows to decrease the runtime and allows to apply the method in dynamic environments. However, the duration of the trajectory is fixed as in the original method.

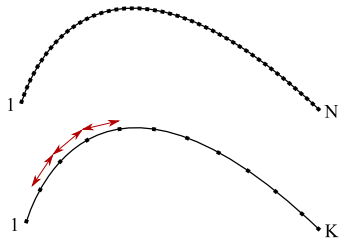


Fig. 2. Top: trajectory of original STOMP with a large number of keyframes. Bottom: small number of keyframes in our modification; transitions between them are evaluated.

III. Method

Our method is based on STOMP [1], which demonstrated a good performance in optimizing high-dimensional trajectories with respect to multiple criteria. In this section we discuss our extension of this method.

A. Original STOMP

In STOMP [1], a planning task is considered as an optimization problem. The objective is to find a trajectory which has the minimal cost according to a given cost function.

The input of STOMP is an initial trajectory Θ which consists of N keyframes $\boldsymbol{\theta}_i \in \mathbb{R}^J$ in a joint space with J joints. The keyframes are equally spaced in time and discretize a predefined fixed duration T. A naïve initial trajectory that is often used is the linear interpolation between the given start and goal configurations $\boldsymbol{\theta}_{start}$ and $\boldsymbol{\theta}_{goal}$. During the optimization process, start and goal configurations remain unchanged. STOMP outputs an optimized trajectory. The optimization problem is formulated as:

$$\min_{\tilde{\Theta}} \mathbb{E}\left[\sum_{i=1}^{N} q(\tilde{\boldsymbol{\theta}}_i) + \frac{1}{2}\tilde{\Theta}^\top R \tilde{\Theta}\right], \tag{1}$$

where $\tilde{\Theta} = \mathcal{N}(\Theta, \Sigma)$ is a noisy joint parameter vector, given that Θ is the mean and Σ is the covariance. $q(\tilde{\boldsymbol{\theta}}_i)$ is a state cost function which includes: obstacle costs, torque costs, and constraint costs. Each state $\tilde{\boldsymbol{\theta}}_i$ of the trajectory $\tilde{\Theta}$ is evaluated using this cost function. The term $\tilde{\Theta}^\top R \tilde{\Theta}$ is the sum of squared accelerations along the trajectory, which are computed using a finite differencing matrix.

B. Proposed Cost Function

In contrast to the original STOMP, we propose to define the state costs not for individual keyframes, but for transitions between them (Fig. 2). Given a trajectory Θ which consists of N keyframes, the state costs are computed as:

$$q(\Theta) = \sum_{i=0}^{N-1} q(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{i+1}), \tag{2}$$

where $q(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{i+1})$ is a cost for the transition from the configuration $\boldsymbol{\theta}_i$ to $\boldsymbol{\theta}_{i+1}$. This extended cost computation allows for taking control over the number of cost computations. Given a pair of keyframes, it is possible to determine the number of intermediate configurations which have to be checked in order to cover the transition from $\boldsymbol{\theta}_i$ to $\boldsymbol{\theta}_{i+1}$ with a required

precision. This ensures that only necessary computations are made. This model allows to plan trajectories with a substantially smaller number of keyframes, e.g. 10-20. We keep the second term of the original cost function (1) of STOMP. The optimization problem now is defined as:

$$\min_{\tilde{\Theta}} \mathbb{E}\left[\sum_{i=0}^{N-1} q(\tilde{\boldsymbol{\theta}}_i, \tilde{\boldsymbol{\theta}}_{i+1}) + \gamma \tilde{\Theta}^\top R \tilde{\Theta}\right], \tag{3}$$

where $\gamma \in [0, 1]$ is the importance weight of the control costs. We define the transition cost function:

$$\begin{aligned} q(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{i+1}) =& q_o(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{i+1}) + q_l(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{i+1}) + q_c(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{i+1}) \\ &+ q_d(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{i+1}) + q_t(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{i+1}), \end{aligned} \tag{4}$$

where q_o is an obstacle cost, which penalizes collisions and being close to the obstacles, q_l is a joint limit cost which penalizes violations of joint limits, q_c is a constraint cost which penalizes violation of any custom constraints of the end-effector position/orientation, q_d is a duration cost, which penalizes long durations and q_t is a torque cost which penalizes high torques. Each cost component function $q_j(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{i+1})$ is designed so that:

$$q_j(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{i+1}) = \begin{cases} \lambda_j \cdot q_{j_v}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{i+1}), & \text{if } \boldsymbol{\theta}_i \rightarrow \boldsymbol{\theta}_{i+1} \text{ is valid} \\ q_{j_{nv}}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{i+1}), & \text{otherwise} \end{cases}, \tag{5}$$

where $q_{j_v}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{i+1}) \in [0, 1]$ is a cost for a valid transition, defined for the cost component q_j. The term $\boldsymbol{\theta}_i \rightarrow \boldsymbol{\theta}_{i+1}$ corresponds to a transition from the configuration $\boldsymbol{\theta}_i$ to $\boldsymbol{\theta}_{i+1}$. A transition from $\boldsymbol{\theta}_i$ to $\boldsymbol{\theta}_{i+1}$ is considered to be valid with respect to the cost component q_j if there are no critical violations of the constraints defined for q_j. For example, a collision with an obstacle is a critical violation with respect to the obstacle cost function. The function $q_{j_{nv}}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{i+1}) \gg 1$ defines a cost for non-valid transitions. This is done in order to prevent the algorithm from decreasing costs for the valid transitions in order to compensate high costs of the invalid transition. At the same time, when the transition is valid, the cost component is scaled within the interval $[0, 1]$, which allows to utilize a system of weights $\lambda_j \in [0, 1]$ in order to set a relative importance of the cost components. This allows for defining properties for optimizing cost components, which may differ for each particular situation.

1) Obstacle costs: Obstacle costs penalize collisions with the environment, self collisions and being close to the obstacles. We adopt a signed Euclidean Distance Transform (EDT) [17] representation for the environment as in the original STOMP. This representation requires an assumption that the environment is static during the optimization and motion execution. We utilize this assumption, and divide the robot body into static and dynamic parts. The static part is not involved in the planned movement, e.g., the robot base. It is represented with another instance of EDT, and must be precomputed only once, while the EDT for the environment must be precomputed before each new planning task. The dynamic part is represented with a set of spheres, as in the original STOMP. They are checked for collisions with both

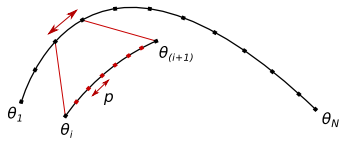


Fig. 3. In order to evaluate a transition, a discretization precision p is determined. Each of obtained intermediate configurations is evaluated with a cost function. The maximum cost is chosen to be the cost for the transition.

distance fields and against each other. In order to eliminate unnecessary checks between spheres, they are partitioned into collision groups. Pairs of collision groups to be checked are stored in an Allowed Collision Matrix (ACM).

To estimate the obstacle cost $q_o(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{i+1})$ for the transition from configuration $\boldsymbol{\theta}_i$ to $\boldsymbol{\theta}_{i+1}$, we determine a set of equally spaced intermediate configurations Ω. To find $|\Omega|$, a link which moves for the longest Euclidean distance d is determined using forward kinematics. In order to determine the movement distance of the link, a specific point is assigned for each link for which the distance is measured. We define this point to be in the place where the successor link is connected. Given a required precision p, $|\Omega|$ is computed as $|\Omega| = \frac{d}{p}$ (Fig. 3).

In situations where the obstacles are far away, a low precision is used, while in situations when the obstacles are close, a high precision is used. In order to estimate the precision p for a particular transition, we estimate the distance d_{obst} to the closest obstacle as: $d_{obst} = \min(dist(\boldsymbol{\theta}_i), dist(\boldsymbol{\theta}_{mid}), dist(\boldsymbol{\theta}_{i+1}))$, where $dist(\boldsymbol{\theta}_i)$ is a function which estimates the minimum distance from the robot dynamic part to the obstacles for a given configuration $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_{mid}$ is the middle configuration between $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_{i+1}$. Given a finest allowed precision p_{max}, we compute the precision p as:

$$p = \max\left(\frac{d_{obst}}{2}, p_{max}\right). \tag{6}$$

Given a set Ω which consists of $|\Omega|$ uniformly spaced intermediate configurations obtained by linear interpolation from $\boldsymbol{\theta}_i$ to $\boldsymbol{\theta}_{i+1}$, the obstacle cost for the transition is:

$$q_o(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{i+1}) = \max\left(q_o(\boldsymbol{\theta}_j) | \forall \boldsymbol{\theta}_j \in \Omega, q_o(\boldsymbol{\theta}_{i+1})\right), \tag{7}$$

where $q_o(\boldsymbol{\theta}_i)$ is the obstacle cost which determines how feasible a particular configuration $\boldsymbol{\theta}_i$ is:

$$q_o(\boldsymbol{\theta}_i) = \begin{cases} C_o \cdot |d_{min} - d_{obst}|, & \text{if } d_{obst} \leq d_{min} \\ 0, & \text{if } d_{obst} \geq d_{max} \\ \lambda_o \cdot \left(1 - \frac{d_{obst} - d_{min}}{d_{max} - d_{min}}\right), & \text{otherwise} \end{cases}, \tag{8}$$

where $\lambda_o \in [0, 1]$ is the importance weight for the obstacle costs, d_{min} is a minimum acceptable distance to the obstacles and d_{max} is a maximum distance to the obstacles which the algorithm should take into consideration. d_{obst} is the distance to the nearest obstacle. $C_o \gg 1$ is a predefined constant which ensures that unfeasible configurations have very high costs. In case a configuration $\boldsymbol{\theta}_i$ is feasible and the distance to the nearest obstacle falls in the interval $(d_{min}, d_{max}]$, the cost function is scaled within the interval $[0, 1]$.

2) Joint limit costs: These costs penalize violations of joint limits. As any other cost component in our cost function, joint limit costs estimate the cost for a given transition from a configuration $\boldsymbol{\theta}_i$ to $\boldsymbol{\theta}_{i+1}$. The joint limit costs q_l are formulated as:

$$q_l(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{i+1}) = \max\big(q_l(\boldsymbol{\theta}_j) | \forall \boldsymbol{\theta}_j \in \Omega, q_l(\boldsymbol{\theta}_{i+1})\big), \quad (9)$$

where Ω is a set of intermediate configurations for the transition from $\boldsymbol{\theta}_i$ to $\boldsymbol{\theta}_{i+1}$. To define $|\Omega|$, a constant precision p_l is used: $|\Omega| = \frac{d}{p_l}$, where d is a distance determined in the obstacle cost computation. Given upper and lower bounds on joint positions $\boldsymbol{\theta}_{max}$ and $\boldsymbol{\theta}_{min}$ and a configuration $\boldsymbol{\theta}_k$, we compute a maximum deviation $\Delta\boldsymbol{\theta}_k$ from the limits. If there is a violation of a joint limit, $\Delta\boldsymbol{\theta}_k$ has a negative value, or is zero. Otherwise, $\Delta\boldsymbol{\theta}_k$ has a positive value and represents the smallest deviation within the limit. Joint limit cost is defined as:

$$q_l(\boldsymbol{\theta}_k) = \begin{cases} C_l \cdot (|\Delta\boldsymbol{\theta}_k| + 1) & \text{if } \Delta\boldsymbol{\theta}_k \leq 0 \\ \frac{1}{\varepsilon^2}\Delta\boldsymbol{\theta}_k^2 - \frac{2}{\varepsilon}\Delta\boldsymbol{\theta}_k + 1, & \text{if } 0 < \Delta\boldsymbol{\theta}_k < \varepsilon \\ 0, & \text{otherwise} \end{cases}, \quad (10)$$

where $C_l \gg 1$ is a predefined constant. The term ε is a magnitude of a considered safety margin. We do not include a corresponding importance weight λ_l, as this cost is of significant importance in any situation. Positions which are close to the actuator limits may cause harm to the actuators, that is why we employ a smooth cost to penalize positions which are close to the joint limits. This cost function is based on a quadratic function, such that the largest considered deviation of ε leads to the cost value close to 0, meanwhile the deviation close to 0 leads to the cost value close to 1. In our work we use the value $\varepsilon = 0.1$ rad.

3) Custom constraint costs: This cost component is similar to the joint limit costs and preserves any custom constraints on the end-effector position/orientation. We apply the same procedure for the constraint costs q_c, as for joint limit costs: Given a set of intermediate configurations Ω, we compute the cost q_c for each of them, and choose the maximum cost. We record the magnitude of the largest deviation from constraints $\Delta\boldsymbol{\theta}_k$ as described for the joint costs. Custom constraint costs are defined as follows:

$$q_c(\boldsymbol{\theta}_k) = \begin{cases} C_c \cdot (|\Delta\boldsymbol{\theta}_k| + 1) & \text{if } \Delta\boldsymbol{\theta}_k \leq 0 \\ 0, & \text{otherwise} \end{cases}, \quad (11)$$

where $C_c \gg 1$ is a predefined constant which penalizes any violations of custom constraints. We do not include a corresponding importance weight λ_c, as this cost has either a very large value, which penalizes violations, or is 0.

4) Duration costs: Duration costs penalize long durations and, hence, allow to minimize a duration of a trajectory. In order to have a mechanism to influence the duration, the velocity of the joint with longest path is added to the configuration space. We make the assumption that a duration necessary to execute a trajectory is bounded by the duration necessary to execute the trajectory of the joint with the longest path. So, before evaluating a trajectory, the joint with the longest path is determined, and the duration estimation is performed with respect to this joint. By restricting the velocity information to a single joint, we avoid doubling the dimensionality of the optimization problem. The extended configuration now consists of a joint vector and a value v for the velocity: $\hat{\boldsymbol{\theta}}_i = \langle \boldsymbol{\theta}_i, v \rangle$.

We prevent the velocity from exceeding the limits $0 < v < v_{max}$ when generating noisy trajectories by clipping it to the limit. Given the desired velocity for the transition from $\hat{\boldsymbol{\theta}}_i$ to $\hat{\boldsymbol{\theta}}_{i+1}$, it is possible to estimate the duration t for this transition. Before the optimization process is started, we estimate a maximum acceptable duration t_{max} for one transition in order to scale the duration costs from 0 to 1. We define t_{max} as: $t_{max} = \frac{t_{total}}{N-1}$, where N is the number of keyframes and t_{total} is the duration of the initial trajectory executed with low velocity.

In order to provide an additional level of safety for optimized trajectories, we introduce an additional constraint on the velocity which depends on distance to the obstacles. The closer the robot is to an obstacle, the lower is the allowed velocity. This constraint is represented as a set V of tuples of a form $\langle v, d \rangle$, where v is the maximum allowed velocity when the distance to the nearest obstacle is less than d. We determine the duration costs $q_d(\hat{\boldsymbol{\theta}}_i, \hat{\boldsymbol{\theta}}_{i+1})$ as:

$$q_d = \begin{cases} C_v, & \text{if } \exists \langle v, d \rangle \in V : v < v_{\hat{\boldsymbol{\theta}}_i} \wedge d > d_{\hat{\boldsymbol{\theta}}_i} \\ \lambda_d \cdot \frac{t}{t_{max}}, & \text{if } t \leq t_{max} \\ C_d \cdot (t + 1), & \text{otherwise} \end{cases}, \quad (12)$$

where $\lambda_d \in [0, 1]$ is the importance weight for the duration costs, $C_d \gg 1$ and $C_v \gg 1$ are predefined constants, which penalize exceeding of the duration limit and obstacle-velocity constraints. The terms $v_{\hat{\boldsymbol{\theta}}_i}$ and $d_{\hat{\boldsymbol{\theta}}_i}$ are the velocity and the distance to the closest obstacle, respectively, measured for a set of intermediate configurations Ω between $\hat{\boldsymbol{\theta}}_i$ and $\hat{\boldsymbol{\theta}}_{i+1}$, which were defined in the obstacle cost computation.

5) Torque costs: The purpose of this cost component is to penalize high torques and ensure that torque limits are not exceeded. In order to evaluate torque costs of the transition from $\hat{\boldsymbol{\theta}}_i$ to $\hat{\boldsymbol{\theta}}_{i+1}$, we find a set $\hat{\Omega}$ of intermediate configurations which are uniformly distributed along the transition with given constant precision p. We define the torque costs for the transition as:

$$q_t(\hat{\boldsymbol{\theta}}_i, \hat{\boldsymbol{\theta}}_{i+1}) = \max\big(q_t(\hat{\boldsymbol{\theta}}_j) | \forall \hat{\boldsymbol{\theta}}_j \in \hat{\Omega}, q_t(\hat{\boldsymbol{\theta}}_{i+1})\big). \quad (13)$$

The torques τ affecting motors are expressed as a function of joint positions and their derivatives: $\tau = f(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \ddot{\boldsymbol{\theta}})$. As we have the velocity, it is possible to estimate the acceleration as well. We use the RBDL library [18] to compute torques. The torque costs of the configuration $\hat{\boldsymbol{\theta}}_i$ are:

$$q_t(\hat{\boldsymbol{\theta}}_i) = \begin{cases} C_t \cdot (\max_{j \in J}(\tau_j - \tau_{max}) + 1), & \text{if } \tau_j > \tau_{max} \\ \lambda_t \cdot \frac{\sum_J \tau_j}{J \cdot \tau_{max}}, & \text{otherwise} \end{cases}, \quad (14)$$

where $\lambda_t \in [0, 1]$ is the importance weight of the torque cost, τ_{max} is a maximum allowed torque for a single motor,

and $C_t \gg 1$ is a predefined constant. In the first row of the equation above, we penalize any exceeding of the torque limit by large cost $\gg 1$. In the second row, we produce a cost $\in [0, 1]$ which penalizes high torques.

C. Optimization Process

In the previous subsections, the multicriteria cost function was discussed. However, this complex function leads to a complex solution space with many disjoint local minima. In this subsection, we describe how extended STOMP is applied to find feasible trajectories more effectively.

One of the most severe barriers on the way to a feasible trajectory are obstacles. Often, the initial trajectory is going through obstacle regions. Thus, finding a collision-free trajectory is the first problem which must be solved. However, our cost function consists of five components, two of which are not relevant for this phase: duration and torque costs. While the algorithm attempts to leave the region of collisions, these costs are not important, as the current solution is not feasible anyway. Using them could slow down convergence, as the components may pull the trajectory in different directions. Moreover, these components would introduce additional computations.

In order to address this issue, the optimization process is split into two consecutive phases. In the first phase, we use a simplified cost function. It consists of: obstacle costs, joint limit costs, and constraint costs. Optimization with the simplified cost function continues until a valid trajectory is found. After this, the configuration space is extended with velocity, and the second phase of the optimization starts, where the full cost function with five components is used. This phase continues until a termination criterion is met.

In certain situations, the algorithm can get stuck in a local minimum. In order to prevent failures or unnecessary exhaustive runs in these cases, we apply the algorithm in an iterative manner. If the algorithm cannot improve the solution during a given number of iterations, and the current best solution is not valid, the optimization starts from scratch. In this case, the best solution from the previous iteration is used as initialization. Initial noise standard deviation tends to explore previously unseen areas and leads to discovery of a valid solution. The maximum number of replanning attempts M is predefined and we use the value $M = 5$.

This restarting approach shows better results than attempts to solve the problem in one exhaustive run of the algorithm. The separation of the optimization process into two parts with simplified and full cost functions allows to decrease the time of the optimization.

IV. Experiments

In order to evaluate our method, we performed experiments in simulation and on real robots. Our centaur-like robot Momaro [19] was used in almost all experiments. Momaro has two 7 DOF arms and a torso yaw joint, which allows to enlarge the workspace. In all experiments with Momaro, the planning is performed for 8 DOF: one 7 DOF arm and the torso yaw. The volume of the workspace covered by the signed distance field is $2.0 \times 1.5 \times 1.5$ m. The distance field has a resolution of 1.5 cm. All start and goal configurations used in the experiments were defined manually. A linear interpolation in joint space between start and goal configuration was used as an initial trajectory.

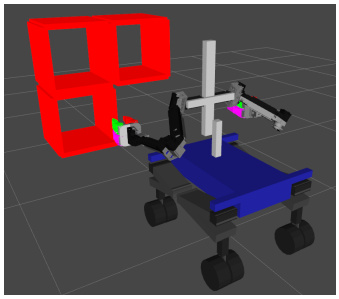


Fig. 4. Environment for the shelf experiment. Momaro stands in front of the shelf with three cells.

A. Simulation Experiments

We performed the experiments in simulation on a desktop computer with Quad-core 4.00 GHz Intel Core i7-4790K CPU, 32 GB of RAM, 64 bit Kubuntu 14.04 with 4.2.0-42 kernel using ROS Indigo Igloo. All evaluated algorithms ran on a single core.

1) Shelf experiment: The purpose of this experiment is to model an every-day task. We constructed a shelf with three $35 \times 35 \times 35$ cm cells. The thickness of a shelf border is 3 cm. The robot stands in front of the shelf with an arm in a neutral position (Fig. 4). In addition to the neutral configuration, there are three more configurations, where the hand is located inside the first, the second and the third cell, respectively.

The experiment consists of 12 tasks which are formed by all possible transitions between the four configurations. Each task is performed 100 times to average out noise from obtained measurements. In the shelf scenario, two more series of tasks of higher difficulty were designed. The initial set of configurations is referred to as "Easy". The gripper was immersed 11 cm deeper into the cells, which made the task harder as the gripper had to travel more in the tight space of the cells. We refer to this experiment as "Hard". Finally, an orientation constraint for the gripper was introduced. We kept "Hard" configurations, but now the gripper had pitch and roll constrained to deviate no more than ± 0.2 rad from the initial orientation. We refer to this as "Hard constrained" task.

We performed this experiment with four algorithms: Lazy Bi-directional KPIECE (LBKPIECE)[1], which uses a discretized representation of projected state space in order to find a solution and is a combination of [20] and [21], RRT-Connect [5] from OMPL [22], STOMP-Industrial[2], which is a newer implementation of the original STOMP, and our method, which is referred to as STOMP-New. We set the time

[1] http://ompl.kavrakilab.org/classompl_1_1geometric_1_1LBKPIECE1.html

[2] https://github.com/ros-industrial/industrial_moveit

TABLE I

COMPARISON OF SUCCESS RATE AND AVERAGE RUNTIME.

	Difficulty level		
	Easy	Hard	Hard constrained
Algorithm	Success rate runtime [s]	Success rate runtime [s]	Success rate runtime [s]
LBKPIECE	0.94 2.47 ± 1.08	0.93 2.46 ± 0.85	- -
STOMP-Industrial	0.87 0.87 ± 0.86	0.76 1.47 ± 1.01	- -
RRT-Connect	0.97 0.29 ± 0.18	0.96 0.85 ± 0.58	0.97 1.22 ± 1.04
STOMP-New	**1.0** **0.09 ± 0.02**	**1.0** **0.18 ± 0.11**	**0.99** **0.28 ± 0.21**

TABLE II

COMPARISON OF AVERAGE RUNTIME FOR SIMPLIFIED AND FULL COST.

	Difficulty level		
	Easy	Hard	Hard constrained
Simplified costs	0.09 ± 0.02	0.18 ± 0.11	0.28 ± 0.21
Full costs	0.12 ± 0.04	0.23 ± 0.19	0.48 ± 0.32
Runtime growth	33%	28%	71%

limit for LBKPIECE and RRTConnect to be 5 seconds. We set the maximum iteration number for STOMP-Industrial and STOMP-New to be 100. At each iteration, ten trajectories are sampled. Trajectories of STOMP-New consist of ten keyframes and 50 keyframes for STOMP-Industrial. We made efforts to tune the planners, so that they demonstrate their best performance. In this experiment, we used the simplified cost function in STOMP-New, as the compared methods do not optimize the duration or motor torques. The obtained average success rates and runtimes are shown in Table I. There are no results for LBKPIECE and STOMP-Industrial for "Hard constrained" test, as the orientation constraints were not realized in these implementations.

One can observe that all algorithms except STOMP-Industrial achieved a success rate close to 1.0. However, the average runtime differs significantly. The slowest method is LBKPIECE. The second slowest algorithm is STOMP-Industrial, which has a noticeable improvement in runtime in comparison with LBKPIECE. STOMP-New and RRTConnect have shown the best performance, having high success rates and low runtimes. Our method achieved three to four times lower runtime than RRTConnect. The average time for computation of EDT in STOMP-New was 0.033 s. The improvement in runtime in comparison to STOMP-Industrial was achieved by the proposed cost function, which reduces the computations. In addition, smaller keyframe numbers which are available for our method, contribute to the speed up. On average 33 collision checks per trajectory were performed by our method during this experiment.

In order to estimate a runtime growth as well as possible success rate degradation when using full costs instead of simplified costs, we performed the shelf experiment one more time using full costs. During this experiment, we set all cost importance weights to a neutral 0.5 value.

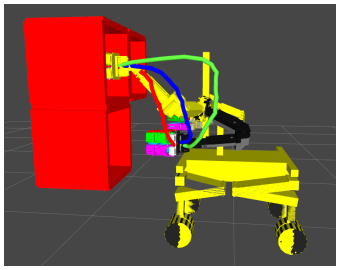


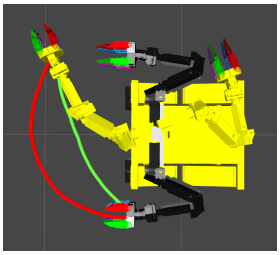
Fig. 5. Trajectories obtained with different obstacle cost importance weights. Red: 0.0; Blue: 0.5; Green: 1.0. The larger the weight is, the larger distance to the obstacles is kept by the robot. Black: start pose; Yellow: goal pose.

The comparison of the obtained runtimes is shown in Table II. The success rate remained the same, thus we do not show it. For both "Easy" and "Hard" unconstrained tests, the runtime grew by approximately 30%. For the test with orientation constraints, the runtime growth reached 71% which is explained by many disjoint local minima caused by the constraints, which is harder to overcome. However, the full cost function allowed to obtain trajectories with lower durations and torques. We demonstrate the effects of these components in the next subsections. Overall, the runtime growth is not critical. Our method may be used in a frequent-replanning manner for acting in dynamic environments.

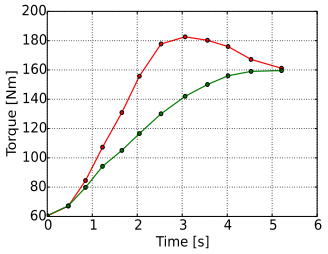
2) Obstacle costs: In this experiment, we demonstrate how different obstacle cost component importance weights influence the obtained solutions. We took a task from the shelf experiment and obtained solutions with three different obstacle importance weights, shown in Fig. 5. While the trajectory with the value of obstacle cost weight 1.0 is the safest, as it moves the arm very far from the obstacles, this trajectory is the longest and has the longest duration. The trajectory with lowest obstacle weight is the fastest to execute, but includes movements close to the obstacles.

3) Torque optimization: In order to demonstrate how torque optimization influences the resulting trajectory, we performed an additional experiment. The initial configuration is a default position of the robot with bent elbow. The goal configuration has fully extended arm and the torso is rotated. The weight of the end-effector is being increased by 5 kg representing a heavy object in the hand. The optimization is performed two times: with and without torque minimization. The obtained trajectories are depicted in Fig. 6. Without torque minimization, all joints move uniformly towards the goal. With torque minimization turned on, the trajectory is different. The arm in the extended state experiences high torques due to gravity. The optimizer avoids this effect by keeping the arm bent and rotating the torso first. Only when this movement is finished, the arm is extended, which results in lower total torque.

4) Duration optimization: In this experiment, we demonstrate the behaviour of our duration cost component. We took a task from the shelf experiment and adjusted the goal configuration in a way that it is located very close to the cell border, so that in the end the robot must move close

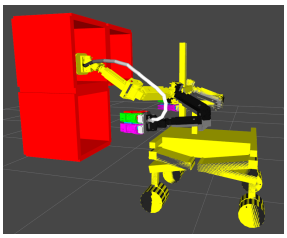


(a)

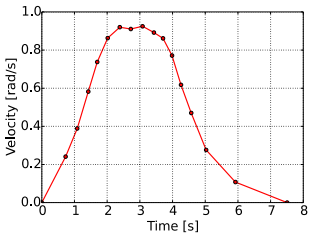


(b)

Fig. 6. Comparison of trajectories obtained with/without torque optimization. The robot is assumed to hold 5 kg. Red: without torque optimization; Green: with torque optimization. **(a)** Trajectories of the end-effector. Black: start pose; Yellow: goal pose. With torque optimization the robot first rotates the torso and only then extends the arm, which results in lower torque. **(b)** Magnitude of the total torque. Without optimization (upper line) the torque grows faster and reaches unnecessary high values. While with optimization (lower line) total torque grows slower.



(a)

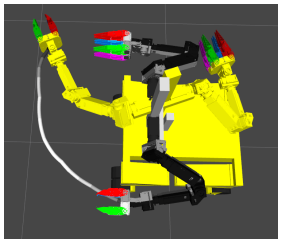


(b)

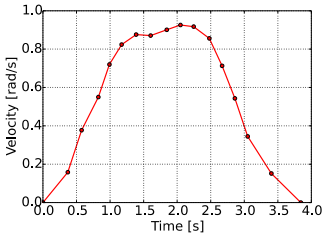
Fig. 7. Duration optimization. In order to make the movement safer, the optimizer maintains lower velocities in region near the obstacles. This makes an earlier deceleration necessary, which results in a trajectory with longer duration, but with a higher safety level. **(a)** The grey-scale line represents the trajectory of the end-effector. The brighter the segment is, the larger is the velocity during that segment. Black: start pose; Yellow: goal pose. **(b)** Velocity vs time.

to the obstacle. The obtained solution is shown in Fig. 7 As the movement starts and ends in a static state, both initial and goal velocities were set to 0 rad/s. One can see that the algorithm attempts to reach the maximum allowed velocity (1.0 rad/s) during the first part of the trajectory and then maintains this velocity. However, as further movement is done close to the obstacle, the deceleration is started in advance and the movement is continued towards the goal with low velocity, which makes the motion safer.

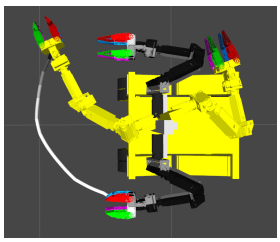
To demonstrate replanning capabilities of our method, we performed an additional experiment. The optimization is done two times: in the first case start and goal velocities are 0 rad/s. In the second case, the initial velocity is 0.7 rad/s and the goal velocity is 0 rad/s. The experiment shows that our method can be used for replanning when the robot is already moving. Resulting trajectories are shown in Fig. 8. One can observe that in both cases the velocity smoothly grows towards its maximum allowed value (1.0 rad/s). It stays on this level and then decreases until the goal value is reached. The trajectory which starts with 0.7 rad/s velocity has smaller duration (2.78 s) than the trajectory which starts with 0 rad/s (3.81 s), which is an expected result.



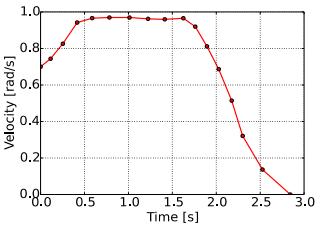
(a) Trajectory with both start and goal velocities 0 rad/s.



(b) Velocity vs time for (a).

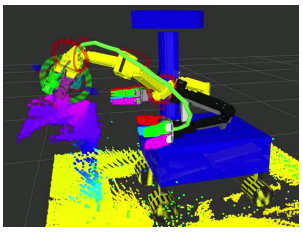


(c) Trajectory with start velocity 0.7 rad/s and goal velocity 0 rad/s.

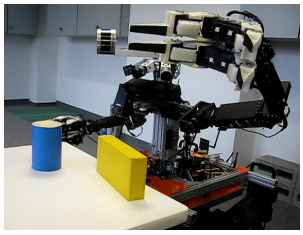


(d) Velocity vs time for (c).

Fig. 8. Example of a duration optimization. Black: start pose; Yellow: goal pose. The grey-scale lines represent the trajectories of the end-effector. The brighter the segment is, the larger is the velocity. In the first case **(a)** the initial and the goal velocities are 0 rad/s. In the second case **(c)** the initial velocity is 0.7 rad/s. In both cases the optimizer attempts to reach the maximum velocity and keeps it as long as possible before deceleration. However, as in **(c)** the velocity is non-zero initially, there is less time spent for acceleration, and hence, the overall duration is smaller.



(a)



(b)

Fig. 9. Obstacle avoidance with a real robot. **(a)** Planned trajectory which avoids the obstacle in order to reach a pre-grasp pose. **(b)** Momaro executing the planned trajectory.

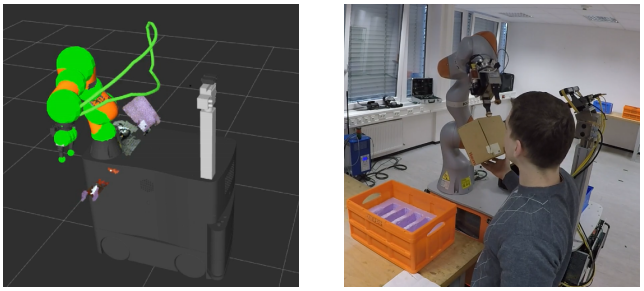
B. Robot Experiments

To demonstrate that our method can be applied in reality, experiments with real robots were performed. The videos of the experiments are available online[3].

1) Momaro: We used our method to reach a pre-grasp pose with the Momaro robot. As shown in Fig. 9, Momaro successfully avoided an obstacle placed on the way. This demonstration was shown live during the review meeting of the CENTAURO[4] project. Different obstacle cost importance weights were used to avoid the obstacle with different margins, to demonstrate planning of safe or fast trajectories.

[3] http://www.ais.uni-bonn.de/videos/IROS_2017_Trajectory_Optimization

[4] https://www.centauro-project.eu



(a) (b)

Fig. 10. Replanning with iiwa arm. **(a)** Two trajectories: initial (lower line) and replanned (upper line), which was produced when previously unconsidered obstacle (box in the middle) interfered the initial trajectory. **(b)** iiwa arm executing a replanned trajectory, avoiding the previously unconsidered obstacle (box).

2) KUKA arm: STOMP-New was used during the Showcase evaluation of the KittingBot project in the European Robotics Challenge 2 (EuRoC)[5]. In this challenge, the KUKA miiwa robot was used, which consists of an iiwa manipulator on an omnidirectional mobile base. The objective was to pick up different engine parts in different locations across the arena. Our method was used to plan trajectories for the iiwa arm to reach a pre-grasp pose and to deliver a grasped object to a pre-release pose. There were three different objects: metal engine support parts of two sizes and engine pipes. In order to plan trajectories with these objects during one run, rough approximations of these objects were attached and detached from our collision model. In addition, we performed a replanning experiment, which shows the capability of our method to perform a quick replanning when an obstacle interferes with the already planned trajectory. In Fig. 10 one can see the robot executing the final trajectory, replanned to avoid the box.

V. Conclusions

In this paper, we presented an approach for optimization of arm trajectories with respect to multiple criteria that extends STOMP. Trajectory duration is optimized by including a velocity into the configuration space. We proposed a multi-component cost function, which includes the following cost components: collisions, joint limits, orientation constraints, joint torque and trajectory duration. The components are normalized and have importance weights assigned. These weights allow to prioritize component optimization and obtain trajectories with different properties. The cost function is designed to evaluate a trajectory efficiently, keeping the computational load as low as possible. It is easy to extend the cost function with any additional costs as long as they are normalized. We evaluated our method in simulation and on real robots. Our approach demonstrated high success rate and low runtime, making it suitable for frequent replanning in dynamic environments.

[5] `http://www.euroc-project.eu/index.php`

References

[1] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning", in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2011.

[2] R. B. Rusu, I. A. Sucan, B. Gerkey, S. Chitta, M. Beetz, and L. E. Kavraki, "Real-time perception-guided motion planning for a personal robot", in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2009, pp. 4245–4252.

[3] R. Diankov, N. Ratliff, D. Ferguson, S. Srinivasa, and J. Kuffner, "BiSpace planning: Concurrent multi-space exploration", in *Robotics: Science and Systems (RSS)*, 2008.

[4] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning", Tech. Rep., 1998.

[5] J. James J. Kuffner and M. Steven, "RRT-Connect: An efficient approach to single-query path planning", in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2000, pp. 995–1001.

[6] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic", in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2014, pp. 2997–3004.

[7] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch Informed Trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs.", in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2015, pp. 3067–3074.

[8] L. Janson and M. Pavone, "Fast Marching Trees: A fast marching sampling-based method for optimal motion planning in many dimensions", in *Robotics Research: The 16th International Symposium (ISRR)*. 2016, pp. 667–684.

[9] F. Burget, M. Bennewitz, and W. Burgard, "BI2RRT*: An efficient sampling-based path planning framework for task-constrained mobile manipulation", in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2016, pp. 3714–3721.

[10] M. Stilman, "Global manipulation planning in robot joint space with task constraints", *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 576–584, 2010.

[11] A. Perez, S. Karaman, A. Shkolnik, E. Frazzoli, S. Teller, and M. R. Walter, "Asymptotically-optimal path planning for manipulation using incremental sampling-based algorithms", in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[12] T. Kunz and M. Stilman, "Probabilistically complete kinodynamic planning for robot manipulators with acceleration limits", in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2014, pp. 3713–3719.

[13] N. Ratliff, M. Zucker, J. A. D. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning", in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2009.

[14] O. Brock and O. Khatib, "Elastic Strips: A framework for motion generation in human environments", *The International Journal of Robotics Research*, vol. 21, no. 12, pp. 1031–1052, 2002.

[15] A. Byravan, B. Boots, S. Srinivasa, and D. Fox, "Space-Time functional gradient optimization for motion planning", in *Proceedings - IEEE Int. Conf. on Robotics and Automation (ICRA)*. 2014, pp. 6499–6506.

[16] R. Steffens, M. Nieuwenhuisen, and S. Behnke, "Continuous motion planning for service robots with multiresolution in time", in *13th Int. Conf. on Intelligent Autonomous Systems (IAS)*. 2016, pp. 203–215.

[17] Q.-Z Ye, "The signed Euclidean distance transform and its applications", *9th Int. Conf. on Pattern Recognition (ICPR)*, 1988.

[18] M. L. Felis, "RBDL: An efficient rigid-body dynamics library using recursive algorithms", *Autonomous Robots*, pp. 1–17, 2016.

[19] M. Schwarz, T. Rodehutskors, D. Droeschel, M. Beul, M. Schreiber, N. Araslanov, I. Ivanov, C. Lenz, J. Razlaw, S. Schüller, D. Schwarz, A. Topalidou-Kyniazopoulou, and S. Behnke, "NimbRo Rescue: Solving disaster-response tasks through mobile manipulation robot Momaro", *Journal of Field Robotics*, 2016.

[20] I. A. Şucan and L. E. Kavraki, "Kinodynamic motion planning by interior-exterior cell exploration", in *Algorithmic Foundation of Robotics VIII: Selected Contributions of the Eight Int. Workshop on the Algorithmic Foundations of Robotics*. 2010, pp. 449–464.

[21] R. Bohlin and L. E. Kavraki, "Path planning using lazy PRM", in *Int. Conf. on Robotics and Automation (ICRA)*, 2000, pp. 521–528.

[22] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library", *IEEE Robotics & Automation Magazine*, pp. 72–82, 2012.

References

[1] Amit Agrawal. Extrinsic camera calibration without a direct view using spherical mirror. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2368–2375, 2013.

[2] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Robotics-DL tentative*, pages 586–606. International Society for Optics and Photonics, 1992.

[3] R. Bohlin and L. E. Kavraki. Path planning using lazy PRM. In *International Conference on Robotics and Automation (ICRA)*, pages 521–528, 2000.

[4] Martin Danelljan, Giulia Meneghetti, Fahad Khan, and Michael Felsberg. Aligning the dissimilar: A probabilistic feature-based point set registration approach. In *International Conference on Pattern Recognition*, 2016.

[5] Martin Danelljan, Giulia Meneghetti, Fahad Shahbaz Khan, and Michael Felsberg. A probabilistic framework for color-based point set registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1818–1826, 2016.

[6] Sandro Esquivel, Felix Woelk, and Reinhard Koch. Calibration of a Multi-camera Rig from Non-overlapping Views. In *Pattern Recognition*, pages 82–91, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[7] Georgios D Evangelidis, Dionyssos Kounades-Bastian, Radu Horaud, and Emmanouil Z Psarakis. A generative model for the joint registration of multiple point sets. In *European Conference on Computer Vision*, pages 109–122. Springer, 2014.

[8] F. Husain, H. Schulz, B. Dellen, C. Torras, and S. Behnke. Combining semantic and geometric features for object class segmentation of indoor scenes. *IEEE Robotics and Automation Letters*, 2(1):49–55, May 2016.

[9] Jr. James, J. Kuffner and M. Steven. RRT-Connect: An efficient approach to single-query path planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 995–1001, 2000.

[10] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal. STOMP: Stochastic trajectory optimization for motion planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[11] Michael Korn, Martin Holzkothen, and Josef Pauli. Color supported generalized-icp. In *Computer Vision Theory and Applications (VISAPP), 2014 International Conference on*, volume 3, pages 592–599. IEEE, 2014.

[12] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. Refinenet: Multi-path refinement networks with identity mappings for high-resolution semantic segmentation. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[13] Andriy Myronenko and Xubo Song. Point set registration: Coherent point drift. *IEEE transactions on pattern analysis and machine intelligence*, 32(12):2262–2275, 2010.

[14] Dmytro Pavlichenko and Sven Behnke. Efficient stochastic multicriteria arm trajectory optimization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[15] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *arXiv preprint arXiv:1506.02640*, 2015.

[16] Andreas Robinson, Mikael Persson, and Michael Felsberg. Robust accurate extrinsic calibration of static non-overlapping cameras. In *International Conference on Computer Analysis of Images and Patterns*, pages 342–353. Springer, 2017.

[17] Diego Rodriguez, Corbin Cogswell, Seongyong Koo, and Behnke Sven. Transferring grasping skills to novel instances by latent space non-rigid registration. In *IEEE International Conference on Robotics and Automation (ICRA)*, (Accepted) May 2018.

[18] S. W. Ruehl, C. Parlitz, G. Heppner, A. Hermann, A. Roennau, and R. Dillmann. Experimental evaluation of the schunk 5-finger gripping hand for grasping tasks. In *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014)*, pages 2465–2470, Dec 2014.

[19] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, and Michael Beetz. Learning informative point classes for the acquisition of object model maps. In *Control, Automation, Robotics and Vision, 2008. ICARCV 2008. 10th International Conference on*, pages 643–650. IEEE, 2008.

[20] Max Schwarz, Christian Lenz, Germán Martín García, Seongyong Koo, Arul Selvam Periyasamy, Michael Schreiber, and Sven Behnke. Fast object learning and dual-arm coordination for cluttered stowing, picking, and packing, 2018. Submitted to Int. Conf. on Robotics and Automation (ICRA), `http://www.ais.uni-bonn.de/papers/ICRA_2018_Schwarz.pdf`.

[21] Ioan A. Şucan and Lydia E. Kavraki. *Kinodynamic Motion Planning by Interior-Exterior Cell Exploration*, pages 449–464. 2010.

[22] Ioan A. Şucan, Mark Moll, and Lydia E. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, pages 72–82, 2012.