Reaching and grasping kitchenware objects

Rui Figueiredo*, Ashwini Shukla†, Duarte Aragão*, Plinio Moreno*,
Alexandre Bernardino*, José Santos-Victor*, Aude Billard†
*Faculty of Electrical and Computer Engineering, Institute de Sistemas e Robótica,
Instituto Superior Técnico, 1049-001 Lisbon, Portugal
{ruifigueiredo, daragao@gmail.com, plinio, alex, jasv}@isr.ist.utl.pt
†Learning Algorithms and Systems Laboratory, École polytechnique fédérale de Lausanne,
EPFL-STI-I2S-LASA, Station 9, CH 1015, Lausanne, Switzerland
{ashwini.shukla, aude.billard}@epfl.ch

***Abstract*— We integrate software components that allow efficient and successful grasping of kitchenware objects. The contributed components include: The object pose detector, the gripper reaching motion and the grasp hypothesis selection. The object pose detector of Drost et. al. [10] is improved, considering rotationally symmetric objects. The reaching motion execution combines two independent dynamical systems: The approach direction system and its tangent space [21]. The coupling provides a robust reaching component that copes with several gripper configurations. The grasp hypothesis selection filters the object poses by considering the table orientation.**

I. Introduction

Grasping and manipulation of known objects have reached already the industrial applications, where the environment is totally predefined in order to avoid failures (e.g. [1]). In such controlled environments, the common approach is to design the shape of the fingers that fits best with the few objects considered and the type of grasps. In addition, the sequence of object poses are known in advance so the trajectory of the arm is fully preprogrammed. By removing any of the constraints of the industrial scenarios, grasping known objects turns into a difficult problem due to uncertainty in the object pose, selection of the best grasping hypothesis and singularities in the trajectories during the reaching phase. We present a grasping system that addresses these problems with the following components: (i) a point cloud-based object pose detector, (ii) a grasping hypothesis selection and (iii) a reaching algorithm based on coupled dynamical systems. The software components are deployed in a scenario with the KUKA LightWeight Robot (LWR) [18], the Universal Gripper WSG 50 [20] with two sensor fingers WSG-FMF [19] and an Asus Xtion PRO [3].

Figure 1 shows the components of the grasping system. The object dataset contains the point clouds and grasping hypotheses (grasping point location and gripper orientation), which are learnt off-line. The grasping hypotheses are obtained from the interaction between the BADGr software [11] and graspIt! [16]. BAGDr applies a box decomposition heuristic that generates a reduced number of hypotheses and graspIt! is able to compute force-closure metrics [5], [6], [9], [7] for each hypothesis. After the off-line selection of grasping hypotheses, the object models are updated with this information. The table top segmentation component is a ROS service [17] that fits the dominant plane in the point cloud through RANSAC and finds the point cloud clusters on top of the table.

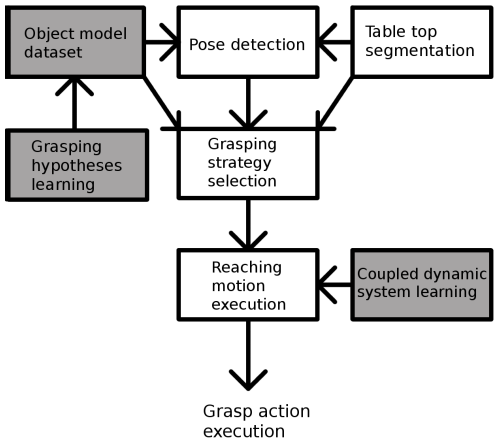

Fig. 1. Grasping system overview. Each box represents a software component of the system. The boxes filled with gray color are executed off-line and the rest are execute on-line

The pose detector component reads the object dataset and receives the clusters provided by the table top segmentation. The object pose detector introduced by Drost et. al. [10] is an efficient and robust method for matching point clouds to known objects. The technique relies on a hash table (i.e., global model) that provides efficient matching

This work is supported by the European Community's 7th Framework Programme, grant agreement First-MM-248258 and by FCT (ISR/IST plurianual funding through the PIDDAC Program)

between local features and point pairs. In addition to the model, a two step matching procedure provides robustness: Firstly, a local voting selects promising object candidates and secondly, a clustering procedure selects the most likely candidates. The advantages of this matching method include: robustness to occlusion, noise and partial view of the object. We introduce an important extension to the pose detector, dealing efficiently with rotationally symmetric objects that are common in many environments (e.g. kitchenware objects like cups, glasses, cans, plates). We reduce drastically the computational complexity of [10] when dealing with this kind of objects. The output of the pose detector is the top candidate for each cluster provided by the table top segmentation.

The grasping hypothesis selection reads the grasping hypotheses from the object dataset and receives: (i) the object poses from the detector and (ii) the table's surface normal. The grasping strategy algorithm just selects the hypothesis whose approach vector is oriented towards the the surface normal but with opposite sense. The selected grasping hypothesis is then sent to the reaching motion module.

Planning and control of constrained grasping motions have often been studied as two separate problems in which one first generates the arm motion [4], [13] and then shapes the hand to grasp stably the targeted object [15], [2]. The sheer complexity of each of these two problems when controlling high dimensional armhand systems has discouraged the use of a single coherent framework for carrying out both tasks simultaneously. We apply the coupled dynamical system based controller of [21], whereby two dynamical systems driving the direction of approach and the corresponding tangent space are coupled. This offers a compact encoding for reach-to-grasp motions that ensures fast adaptation with zero latency for re-planning. Each dynamical system is represented by a Gaussian Mixture Model (GMM), whose parameters are learnt off-line. During the on-line phase, a weighted contribution of the dynamical systems converge to the target grasping hypothesis.

The system briefly explained above allow us to perform successful grasping actions of kitchen objects such us mugs and glasses. Section II explains in detail the object pose detection method, section III the coupled dynamical system, section IV the grasping hypothesis selection, section V presents the experiments and section VI the conclusions and future work.

II. Object pose detection

Each object model is represented by a set of points and associated surface normals, i.e. surflets [22]. Let M be the set of all model surflets, $M = \{s_i^m, i = 1..N\}$ – upper indices m and s will be further used to distinguish model from scene, respectively. An object description suitable for object recognition and pose estimation is created through the analysis of all possible permutations of surflet pairs. Let A be the set of all surflet pairs, $A = \{(s_r^m, s_t^m), r \neq t\}$, which has cardinality $|A| = N \times (N-1)$.

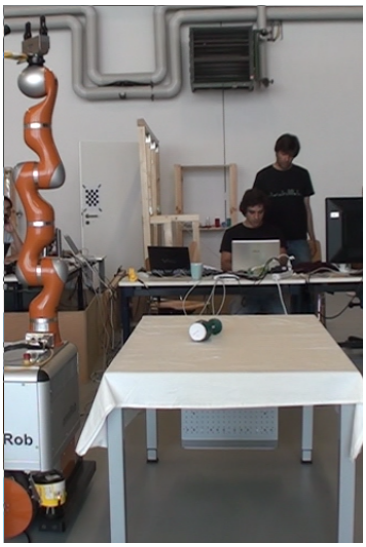

Fig. 2. Experimental scenario

A. Model Description

For each surflet pair (s_r, s_t), we compute a descriptive 4-element feature vector as illustrated in figure 4. This could be formally described by the following expression:

$$F(s_r, s_t) = (f_1, f_2, f_3, f_4) = (\|d\|, \angle(n_r, d), \angle(n_t, d), \angle(n_r, n_t)) \quad (1)$$

The data structure used to represent the model description is a hash table for quick retrieval, in which the key value is given by the discrete point pair feature while the mapped value is the respective surflet pair. Since one key could be associated with several model surflet pairs, each slot of the hash table contains a list of surflet pairs with similar discrete feature.

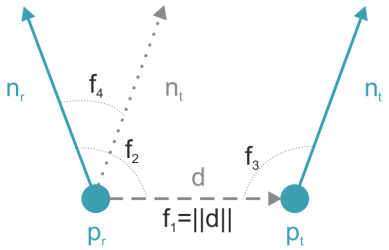

Fig. 4. Point pair feature descriptor

1) Dealing with rotational symmetry: In order to efficiently deal with rotationally symmetric objects, we incorporate a strategy that reduces drastically the size of A, by discarding redundant surflet pairs, thus increasing dramatically the recognition runtime performance. To accomplish this, a Euler angle representation [8], is used to describe orientation. In our work we chose the X-Y-Z Euler representation since we assume that the object axis of symmetry is aligned with the z axis of the object reference coordinate frame. During the creation of the model description, for each surflet pair, we compute the transformation with respect to the object model

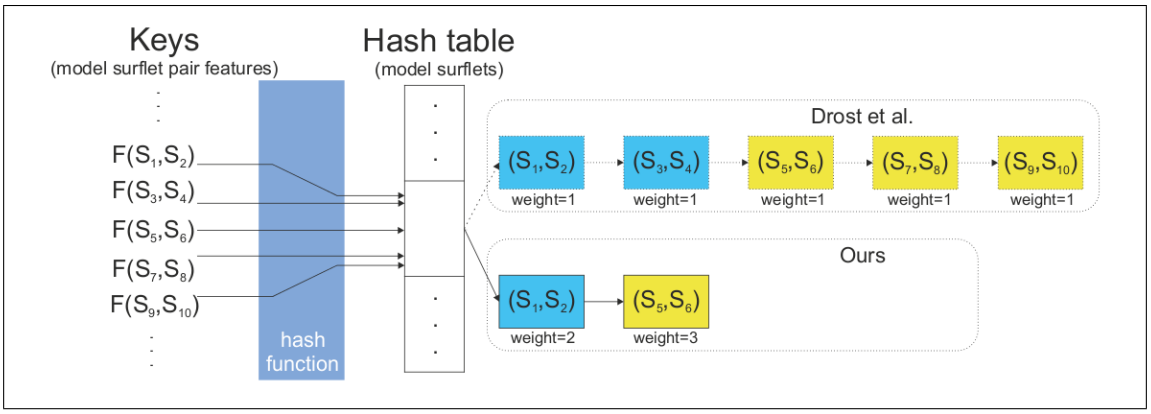

Fig. 3. Example of surflet pairs with similar feature stored in the same slot of the hash table, during the creation of the object model description.

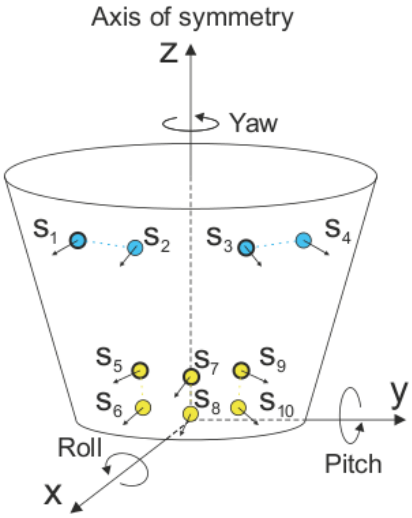

Fig. 5. An example of a rotationally symmetric object model. All illustrated surflet pairs have similar discrete feature. In the figure, pairs represented with similar color are redundant.

reference frame (see section II-B) that aligns it with each similar pair already stored in the hash table. If there is at least one pair for which the alignment transformation has no translation and no roll and pitch components on the rotation as expressed on eqs. (2) and (3) respectively, then, this surflet pair corresponds to a rotation of the other (homologous) around the symmetry axis, and is therefore redundant and discarded.

$$d < d_{\text{th}} \tag{2}$$

$$\alpha_{\text{yaw}=0} < \alpha_{\text{th}} \tag{3}$$

The weight of the homologous surflet pair, stored in the hash table, is then incremented by 1. This process is clearly ilustrated in figures 5 and 3.

B. Pose Estimation

A set of reference surflets on the scene $R_s \subset S$ – where S is the set of all scene surflets, $S = \{s_i^s, i = 1..N\}$ – is randomly chosen and each of them is paired with all the other surflets on the scene. For each scene surflet pair $(s_r^s, s_t^s) \in S^2$ we compute a point pair feature $F(s_r^s, s_t^s)$ and then, using the extracted feature, we obtain a set of model surflet pairs whose feature is similar to it. From every match between a scene surflet pair $(s_r^s, s_t^s) \in S^2$ and a model surflet pair $(s_r^m, s_t^m) \in M^2$, we are able to extract the rigid transformation that aligns the matched model with the scene. This is done by first computing the transformations $T_{m\to g}$ and $T_{s\to g}$ that align s_r^m and s_r^s, respectively, to the object reference coordinate frame x axis, and secondly the rotation α around the x axis that aligns p_t^m with p_t^s. The final transformation that aligns the model with the scene is then computed considering the ensuing expression:

$$T_{m\to s} = T_{s\to g}^{-1} R(\alpha) T_{m\to g} \tag{4}$$

The transformations $T_{m\to g}$ and $T_{s\to g}$ translate p_r^m and p_r^s, respectively, to the reference coordinate frame origin and rotates their normals n_r^m and n_r^s onto the x axis. After applying these two transformations, p_t^m and p_t^s are still misaligned. The transformation $R(\alpha)$ applies the final rotation needed to align these two points.

The transformation expressed in eq. (4) can be parametrized by a surflet on the model and a rotation angle α. In [10], this pair (s_r^m, α) is mentioned as the *local coordinates* of the model with respect to reference point s_r^s.

1) Voting Scheme: This method uses a voting scheme similar to the GHT for pose estimation. For each scene reference surflet, a two-dimensional accumulator array that represents the discrete space of local coordinates is created. The number of rows, N_m, is the same as the number of model sample surflets $|M|$, and the number of columns N_{angle} is equal to the number of sample steps of the rotation angle α.

The voting procedure goes as follows: considering a given reference surflet s_r^s on the scene surface S, we pair it with every other surflet $s_t^s \in S$. For each resulting surflet pair we search on the model surface for similar surflet pairs, with the aim of finding where it might be in the model. This is done by querying the model descriptor for surflet pairs with similar feature. The computed feature $F(s_r^s, s_t^s)$ is used as an index to the model hash table and a list of matched surflet pairs, with similar feature, is returned. For

every match (s_r^m, s_t^m) the rotation angle α is computed and a vote is placed in the accumulator array by incrementing the position correspondent to the local coordinates (s_r^m, α), by the $weight$ of the matched model surflet pair. After pairing s_r^s with all s_t^s, the highest peak – i.e. the position with more votes – in the accumulator corresponds to the optimal local coordinate. In the end, all retrieved pose hypotheses whose position and orientation do not differ more than a predefined threshold are clustered together.

To deal with symmetry, before clustering, we collapse all redundant hypotheses to a single pose. This additional step removes the rotational component around the object axis of symmetry, i.e. yaw, ensuring that all redundant poses are gathered in the same cluster, therefore allocating less resources and reducing the number of computations. We were able to discard near 93% surflet pairs during the creation of the model description, and reduce the number of computations during pose recognition. As shown in figure 6, the recognition rate drops lightly for high levels of noise due to sampling effects, but the recognition time performance increases significantly. For $|S| \approx 5000$, our method achieves a recognition time 120 times faster than [10].

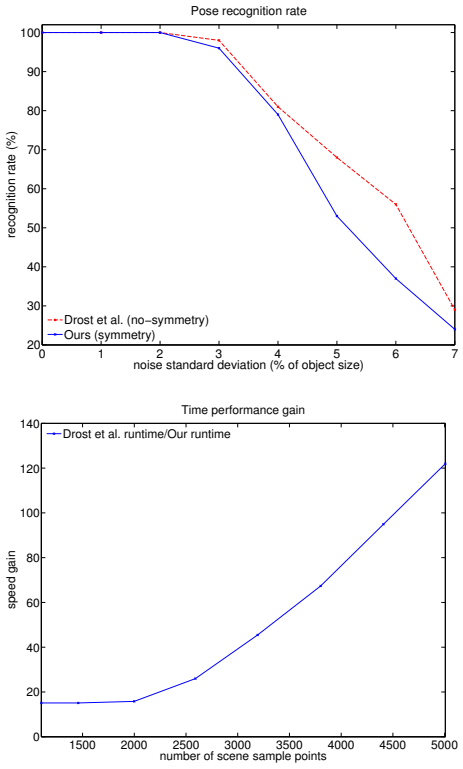

Fig. 6. Comparison results of our method against the method of Drost et. al. , with $|R_s| = 0.05\,|S|$ reference points.

III. Encoding and Executing Reaching Motions with Coupled Dynamical Systems

In this section, we explain the coupled dynamical system (CDS) model [21] used to generate trajectories for the robot's end-effector. It is worth mentioning here that for executing reach-to-grasp motions, it does not suffice to have a simple

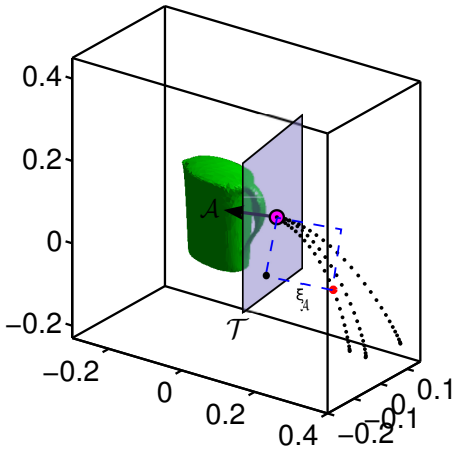

Fig. 7. Separating the demonstration data (black dots) for reaching and grasping an object into two subspaces - $\mathcal{A}$ (approach direction) and $\mathcal{T}$ (its tangent space). The inference model is learned as a joint distribution on the two subspaces $\mathcal{P}(\boldsymbol{\xi}_{\mathcal{A}}, \boldsymbol{\xi}_{\mathcal{T}})$.

trajectory interpolator. This is due to two key reasons - a) The dynamics of the motion must be smooth and may vary from object to object; b) Grasping an object requires maintaining a specific approach direction towards the end of the motion. Next, we will explain the CDS model with respect to reach-to-grasp motions.

We model the reaching motion in two different but coupled subspaces : the direction of approach $\mathcal{A}$ and the corresponding tangent space $\mathcal{T}$. The approach consists of learning two different dynamical systems each operating in one of the above mentioned subspaces and one *inference* model that learns the task metric for coupling the two subspaces. Each of these dynamical systems are learned as a Guassian Mixture Model (GMM) [14]. The two dynamic models are then coupled during task execution using the inference model. Such a spatial coupling ensures that the trajectories retain the approach direction while following the learned dynamics as closely as possible, even in the presence of arbitrary perturbations. Here we briefly summarize the CDS model and the task execution algorithm. For more details the reader is referred to [21].

2) Model Learning: Let $\boldsymbol{\xi}_{\mathcal{A}} \in \mathbb{R}$ denote the Cartesian position of the end-effector along the direction of approach and $\boldsymbol{\xi}_{\mathcal{T}} \in \mathbb{R}^2$ the same in its tangent space as shown in Fig. 7. We record several human demonstrations of reaching and grasping the object and separate them in the two subspaces for learning independent dynamical systems off them. For convenience, we place the attractors of the dynamical systems at the desired grasping point. In other words, the motion is expressed in a coordinate frame attached to the object to be grasped, with its origin at the grasping point and any one of the axes aligned with the direction of approach.

The following three joint distributions, learned as separate GMMs, combine to form the CDS model:

1) $\mathcal{P}\left(\boldsymbol{\xi}_{\mathcal{T}}, \dot{\boldsymbol{\xi}}_{\mathcal{T}} | \boldsymbol{\theta}_{\mathcal{T}}\right)$: encoding the dynamics along the approach direction, called the *master* subsystem
2) $\mathcal{P}(\boldsymbol{\xi}_{\mathcal{A}}, \boldsymbol{\xi}_{\mathcal{T}} | \boldsymbol{\theta}_{\mathcal{AT}})$: encoding the joint probability dis-

Algorithm 1 Grasp Execution using CDS

Input: $\boldsymbol{\xi}_{\mathcal{T}}(0)$; $\boldsymbol{\xi}_{\mathcal{A}}(0)$; $\boldsymbol{\theta}_{\mathcal{T}}$; $\boldsymbol{\theta}_{\mathcal{A}}$; $\boldsymbol{\theta}_{\mathcal{AT}}$; α; β; Δt; ϵ

Set $t = 0$

repeat:

Update Tangent Space Motion:
$$\dot{\boldsymbol{\xi}}_{\mathcal{T}}(t) \sim \mathcal{P}\left(\dot{\boldsymbol{\xi}}_{\mathcal{T}} | \boldsymbol{\xi}_{\mathcal{T}}; \boldsymbol{\theta}_{\mathcal{T}}\right)$$
$$\boldsymbol{\xi}_{\mathcal{T}}(t+1) = \boldsymbol{\xi}_{\mathcal{T}}(t) + \dot{\boldsymbol{\xi}}_{\mathcal{T}}(t)\Delta t$$

Infer Approach:
$$\tilde{\boldsymbol{\xi}}_{\mathcal{A}}(t) \sim \mathcal{P}\left(\boldsymbol{\xi}_{\mathcal{A}} | \boldsymbol{\xi}_{\mathcal{T}}; \boldsymbol{\theta}_{\mathcal{AT}}\right)$$

Update Approach:
$$\dot{\boldsymbol{\xi}}_{\mathcal{A}}(t) \sim \mathcal{P}\left(\dot{\boldsymbol{\xi}}_{\mathcal{A}} | \beta\left(\boldsymbol{\xi}_{\mathcal{A}} - \tilde{\boldsymbol{\xi}}_{\mathcal{A}}\right); \boldsymbol{\theta}_{\mathcal{A}}\right)$$
$$\boldsymbol{\xi}_{\mathcal{A}}(t+1) = \boldsymbol{\xi}_{\mathcal{A}}(t) + \alpha\dot{\boldsymbol{\xi}}_{\mathcal{A}}(t)\Delta t$$

$t \leftarrow t + 1$

until: *Convergence* $\left(\|\dot{\boldsymbol{\xi}}_{\mathcal{A}}(t)\| < \epsilon \quad \text{and} \quad \|\dot{\boldsymbol{\xi}}_{\mathcal{T}}(t)\| < \epsilon\right)$

tribution of the state variables along the approach direction and its tangent subspace, called the *inference* subsystem

3) $\mathcal{P}\left(\boldsymbol{\xi}_{\mathcal{A}}, \dot{\boldsymbol{\xi}}_{\mathcal{A}} | \boldsymbol{\theta}_{\mathcal{A}}\right)$: encoding the dynamics in the tangent subspace, called the *slave* subsystem

Here $\boldsymbol{\theta}_{\mathcal{T}}$, $\boldsymbol{\theta}_{\mathcal{A}}$ and $\boldsymbol{\theta}_{\mathcal{AT}}$ denote the parameter vectors of the GMMs encoding the master, slave and the inference models respectively. The distributions in 1) and 3) above are learned using the *stable estimator of dynamical system* (SEDS) technique [14] which ensures that the learned dynamical system has a single globally and asymptotically stable attractor. This in turn ensures that the overall coupled system will terminate at the desired targets for both hand pose and finger joint angles. The probability distribution in 2) does not represent a dynamics[1], and hence is learned using a variant of SEDS where we maximize the likelihood of the model under the constraint:

$$\mathbb{E}\left[\boldsymbol{\xi}_f \,|0\right] = \mathbf{0}. \tag{5}$$

It is to be noted that the *master* dynamical system runs independently and the *slave* adapts accordingly to maintain the desired coupling. This implies that perturbations in the master subsystem are also mirrored onto the slave in order to maintain the desired coupling behavior.

At execution time, the overall scheme works as shown in Algorithm 1. Note that the free parameters α, β of the model allow to tune the approach behavior of the trajectories by changing their sensitivity w.r.t the different error terms. High values of α ensure strict following of the coupling constraints whereas high values of β make the behavior more sensitive to deviations from the desired coupling.

IV. Grasping Hypothesis Generation

We implemented a two-stage approach in order to build the map between grasp poses and regions of known object models, namely: (i) (Off-line) selection of promising hypotheses using simulation tools and (ii) (On-line) execution of the grasping hypotheses on the actual robot.

[1]Here the dimension of input and output variables is not equal. SEDS can only be applied for learning dynamics, where the inputs are positions and outputs are velocities and hence have the same dimensionality.

The selection of grasping hypotheses relies on the BADGr software [11] that decomposes the object model into a hierarchical set of oriented bounding boxes [12], which allows to define one approach vector and two gripper orientations for each face of the boxes. Then, collision detection algorithms using the approach direction and the gripper model select the reachable hypotheses of the computed bounding boxes. Every reachable hypothesis (grasping approach vector and gripper orientation) is sent to GraspIt! [16], where the fingertips of the gripper are closed. Since the grasp wrench space can not be computed in most of the gripper configurations, the hypotheses are selected by checking for collisions on both fingertips. If there are collisions in only one fingertip, the gripper is translated along the longitudinal direction up to the distance to the other fingertip in that direction. All the promising gripper configurations are stored for grasping execution. Figures 8 and 9 illustrate the grasping trials using the BADGr boxes and GraspIt! simulation on some models.

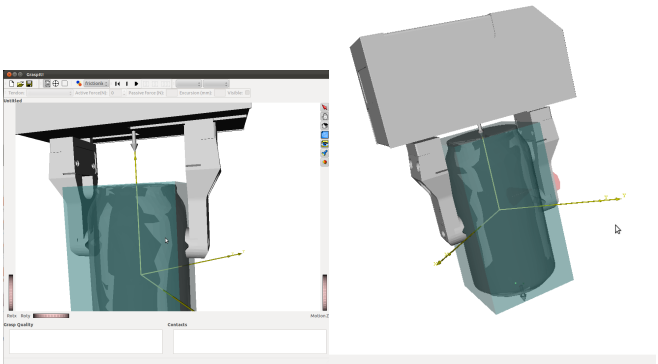

Fig. 8. Screenshot samples during the grasping trials in GraspIt! [16]. Left side shows the detail of the gripper and a soda can within the GraspIt! window and right side shows a grasping hypothesis candidate.

V. Experiments

The object pose detector, the grasping hypothesis selection and the reaching motion execution are independent software components interconnected through ROS topics.

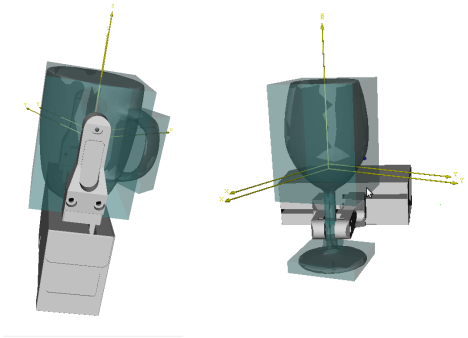

Fig. 9. Samples of successful grasping trials in GraspIt!.

The object pose detector loads the object models and grasping hypotheses at an initial step. Then, at each execution of its callback, the detector: (i) Calls the table top segmentation service, (ii) computes the best object pose hypothesis for each cluster and (iii) publishes the object id and its pose on a ROS topic.

The grasping hypothesis selection receives the object pose hypothesis and the table orientation. Then, at each execution of its callback, the selection algorithm computes the best grasping hypothesis and publishes it on a ROS node.

Finally, the reaching motion execution: (i) reads the best hypothesis from the topic, (ii) computes the arm trajectory using the CDS, (iii) commands the gripper closing and (iv) moves the arm up to evaluate the grasp.

The pipeline described above is executed 10 times, grasping two objects: (i) A mug and (ii) a wine glass. The system grasped successfully 7 out of 10 times, failing 3 times due to (i) very difficult configurations of the arm to reach the grasping hypothesis and (ii) contact between the hand and the object before grasping. Figure 10 shows two of the successful grasping experiments.

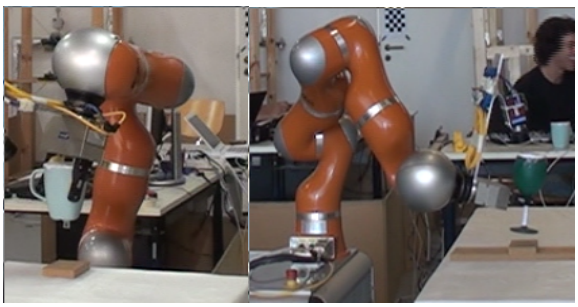

Fig. 10. Samples of successful grasping trials

VI. Conclusions

We presented the software components of a system for reaching and grasping kitchenware objects with a two-fingered gripper. The most important components are: (i) Object pose detector, (ii) Grasping hypothesis selection and (iii) Reaching motion generation and grasp execution. The components interact between each other through ROS topics and are able to grasp successfully both fragile and heavy objects. Future work should consider the on-line collision detection in order to take advantage of the CDS approach.

References

[1] Scape technologies. Available from: `http://www.scapetechnologies.com`.

[2] *2008 IEEE International Conference on Robotics and Automation, ICRA 2008, May 19-23, 2008, Pasadena, California, USA*. IEEE, 2008.

[3] ASUS. Asus xtion PRO. Available from: `http://www.asus.com/Multimedia/Motion_Sensor/Xtion_PRO/`.

[4] Dmitry Berenson, Siddhartha Srinivasa, David Ferguson, Alvaro Collet Romea, and James Kuffner. Manipulation planning with workspace goal regions. In *IEEE International Conference on Robotics and Automation (ICRA '09)*, May 2009.

[5] Antonio Bicchi. On the closure properties of robotic grasping. *I. J. Robotic Res.*, 14(4):319–334, 1995.

[6] Christoph Borst, Max Fischer, and Gerd Hirzinger. Grasp planning: How to choose a suitable task wrench space. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation, ICRA 2004, April 26 - May 1, 2004, New Orleans, LA, USA*, pages 319–325. IEEE, 2004.

[7] M. Buss, H. Hashimoto, and J.B. Moore. Dextrous hand grasping force optimization. *Robotics and Automation, IEEE Transactions on*, 12(3):406 –418, June 1996.

[8] John J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 1989.

[9] Dan Ding, Yun-Hui Liu, and Shuguo Wang. The synthesis of 3-d form-closure grasps. In *ICRA*, pages 3579–3584, 2000.

[10] B. Drost, M. Ulrich, N. Navab, and S. Ilic. Model globally, match locally: Efficient and robust 3d object recognition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 998–1005, june 2010.

[11] K. Hubner. BAGDr software. Available from: `http://www.csc.kth.se/~khubner/badgr`.

[12] Kai Huebner. BADGr–A toolbox for box-based approximation, decomposition and grasping. *Robotics and Autonomous Systems*, 60(3):367–376, 2012.

[13] J. A. Ijspeert, J. Nakanishi, and S. Schaal. movement imitation with nonlinear dynamical systems in humanoid robots. In *international conference on robotics and automation (icra2002)*, 2002.

[14] Seyed Mohammad Khansari-Zadeh and Aude Billard. Learning stable non-linear dynamical systems with gaussian mixture models. *IEEE Transaction on Robotics*, 27(5):943–957, 2011.

[15] A. Miller, S. Knopp, H.I. Christensen, and P. Allen. Automatic grasp planning using shape primitives. In *Intl Conf on Robotics and Automation*. IEEE, Taipai, Taiwan, September 2003.

[16] A.T. Miller and P.K. Allen. Graspit! a versatile simulator for robotic grasping. *Robotics Automation Magazine, IEEE*, 11(4):110–122, dec. 2004.

[17] M. Muja and M. Ciocarlie. Table top segmentation package. Available from: `http://www.ros.org/wiki/tabletop_object_detector`.

[18] KUKA robotics. Kuka lightweight robot (LWR). Available from: `http://www.kuka-robotics.com/en/products/addons/lwr/`.

[19] Weiss robotics. Sensor finger WSG-FMF. Available from: `http://www.weiss-robotics.de/gripper-systems/gripper-accessories/sensor-finger-wsg-fmf.html`.

[20] Weiss robotics. Universal gripper WSG 50. Available from: `http://www.weiss-robotics.de/gripper-systems/gripper-modules/universal-gripper-wsg-50.html`.

[21] Ashwini Shukla and Aude Billard. Coupled dynamical system based armhand grasping model for learning fast adaptation strategies. *Robotics and Autonomous Systems*, 60(3):424–440, 2012.

[22] Eric Wahl, Ulrich Hillenbrand, and Gerd Hirzinger. Surflet-pair-relation histograms: A statistical 3d-shape representation for rapid classification. *3D Digital Imaging and Modeling, International Conference on*, 0:474, 2003.