EU FP7 CogX
ICT-215181
May 1 2008 (52months)

DR 2.5: Qualitative models of object behaviour, and grasping of novel objects

Renaud Detry[2], Michael Zillich[1], Andreas Richtsfeld[1], Johann Prankl[1], Sergio Roa[3], Sebastian Zurek[4], Yasemin Bekiroglu[2], Thomas Mörwald[1], Markus Vincze[1], Danica Kragic[2], Jeremy Wyatt[4], Geert-Jan Kruijff[3]

[1]*TUW, Vienna* [2]*KTH, Stockholm* [3]*DFKI GmbH, Saarbrücken* [4]*BHAM, Birmingham*

⟨`detryr@kth.se`⟩

Due date of deliverable:	May 25, 2012
Actual submission date:	May 28, 2012
Lead partner:	KTH
Revision:	final
Dissemination level:	PU

This deliverable reports work related to object manipulation. The first problem discussed in this report is the extraction of qualitative models of object behaviour. We present a learning algorithm capable of extracting a discrete representation of a sensorimotor space. We also present a method for identifying different modes of object interactions, which allows us for instance to predict whether an object will turn left, right, or not turn at all if a particular push is applied.

Grasping novel objects is the second problem discussed in this report. We present means of detecting new objects from vision. We also present a novel approach that allows an agent to plan grasps onto novel objects by matching parts of the new object to parts of previously-grasped objects. We finally extend the work done on tactile-based grasp stability estimation in the previous period to allow the robot to assess grasp stability from both touch data *and* task requirements.

Executive Summary

This report presents the work done in the final year of CogX on the topics of (1) modelling qualitative object behaviour and (2) grasping novel objects. Regarding qualitative behaviour models, we first developed an algorithm that extracts probabilistic finite state representations of a dynamical system. This algorithm is applicable to the extraction of qualitative states from sensorimotor data gathered during the execution of a task by a robot. Second, we developed a method for identifying different modes of object interactions, which allowed us for instance to predict whether an object will turn left, right, or not turn at all if a particular push is applied.

Regarding (2), we made three novel contributions. First we developed a method for detecting novel objects in 3D scenes. This method was further integrated in the object tracker discussed in the previous periods, to allow the tracker to trigger texture-based object detection when its belief on the object pose becomes too low. Second, we developed an agent capable of learning the shape of parts by which objects are often grasped, which subsequently allows the agent to plan grasps on partly familiar objects. Finally, we introduced a task model that includes kinematic grasp parameters and tactile signals, which allows the agent to model the stability of a grasp with respect to a given task.

The work presented in this report led to four peer-reviewed conference publications, and six more conference and journal submissions. The work presented here follows up on DR 2.4 (forward models, grasping previously unseen objects) and on DR 5.4 (learning of cross-modal concepts).

Qualitative models of object behaviour

We address the problem of finding qualitative representations of dynamical systems. The task is to infer probabilistic finite-state machines that model the interaction between a robot and an object. In this case, a robot performs pushing actions and sequences of object poses are stored to be used in the learning process. New algorithms were developed for discretisation of sensorimotor spaces and extraction of finite-state probabilistic models [59, 73] (Annexes 2.1 and 2.2). In order to evaluate them, we tested their ability to find qualitative representations of artificial dynamical systems with noisy features, i.e., from data generated by probabilistic finite-state automata where states are gaussian noise distributions.

Role of qualitative models of object behaviour in CogX

Robots need to have capabilities for introspection, abstraction and memory in order to use their acquired knowledge in future tasks. Then, the models that are obtained by the robot during the interaction with objects can be used subsequently to plan actions, to reason, to test a theory of the behaviour of the object after some action. Additionally, given the high-dimensionality of the sensorimotor spaces, it is useful to find more coarse-grained abstractions (concepts) from these interactions, in order to be used for human-driven learning and communication tasks involving language and gestures.

Contribution to the CogX scenarios and prototypes

This work contributes to the Dexter scenario. By learning how objects move when it interacts with them, the agent is able to predict how objects would move if certain manipulation plans are executed. Moreover, the graph-based nature of the representation encodes the probabilistic transitions that lead to subsequent system states which is particularly useful in planning.

Grasping of novel objects

Grasping novel objects is the second problem addressed in this deliverable. It encompasses two sub-problems: detecting novel objects and planning grasps onto novel objects. For the former we learned 3D perceptual grouping principles to segment objects from RGBD images of cluttered scenes and describe objects as grouped surface patches [56, 57, 58] (Annexes 2.3, 2.4, 2.5). These surface patches are described as NURBS [47, 4] (Annexes 2.7 and 2.6), allowing for a flexible and compact representation of objects. Once objects are detected and object models in terms of surface patches are extracted, these can be tracked using methods presented in DR 2.4 [46]. We extended this work to include a self-assessment of the tracker regarding its current performance and object state (moving, occluded) which allows for robust tracking by optimally combining tracking and re-detection as required [48] (Annex 2.8). This work led to a conference and workshop publication at CVPR and CVWW respectively [4, 56] (Annexes 2.6 and 2.3).

Regarding grasping, we present a method that allows a robotic agent to learn prototypical parts by which objects are often grasped, from a set of grasps demonstrated by a teacher [18]. Prototypes subsequently allow the agent to grasp novel objects that contain a part that resembles one of the prototypes. This work led to an ICRA publication [18] (Annex 2.9). We also present a model of task-oriented grasp stability, and means of inferring task stability from tactile data [8] (Annex 2.10). This work was evaluated on the KTH manipulation platform (industrial arm and dexterous hand). The work on task stability partly builds on the contributions presented in DR 2.4 [7].

Role of grasping of novel objects in CogX

One of the aims of CogX is to create an agent that is able to familiarise itself with an unknown environment. While exploring its environment, the agent comes into contact with novel objects. The agent has to grasp some of these objects to fulfil its task. Even when grasping is not required by the task itself, manipulating objects represents an efficient exploration strategy, and it allows the agent to fill object-related knowledge gaps.

In this report, we present means of detecting novel objects from vision, and means of exploiting previously-acquired object knowledge to grasp novel objects. We also present a method for exploiting tactile data *and* task requirements to assess the stability of a grasp. These three contributions improve the efficiency and the robustness with which our agent familiarises itself with novel objects.

Contribution to the CogX scenarios and prototypes

This work contributes to the Dexter scenario, where the agent is required to interact with novel objects. Novel objects first need to be identified as such. This problem is solved using the novelty detection method discussed above. Part-based planning is then used to plan grasps on the objects, provided that partly similar objects have been handled previously. Novelty detection also contributes to the George scenario, where the robot learns novel objects in interaction with a tutor.

1 Tasks, objectives, results

1.1 Qualitative models of object behaviour

1.1.1 Planned work

This deliverable reports work related to Task 2.7:

> Task 2.7: Extracting qualitative states. To be able to perform introspection on possible qualitative cause we require a model that has not only continuous states, but also qualitative states, with qualitative explanations for the transitions between them. We will use the notion of force-aspect graphs to devise a learning algorithm capable of partitioning the continuous configuration space of the modular motor learning predictions into sets of qualitatively similar stable states, plus their basins of attraction. (M33–M39)

This deliverable presents two contributions that address this task. The first contribution is an algorithm that learns a probabilistic finite-state representation of a dynamical system from sensorimotor data. The second contribution is an algorithm that extracts different modes of object-effector interaction in manipulative actions.

This deliverable contributes to the realisation of the sixth measurable objective:

> Objective 6: Methods for perception and manipulation of objects that enable a robot to actively explore objects, to extend its manipulative skills, and its understanding of these.

Extracting qualitative models provides the agent with a compact representation of its sensorimotor space, which allows it to better understand its skills. Compact representations also simplify the decision process required by the high-level planning of actions or exploration, providing the agent with a small set of discrete choices instead of the full range of possible motions.

1.1.2 Actual work performed

We developed an algorithm [59] (Annex 2.1) that extracts probabilistic finite state representations of a dynamical system. The resulting representation has the form of a set of states, with transitions of different probabilities between the states. A dynamical system can be represented as a tuple $\langle I, O, S, P\rangle$, where I, O and S are input, output and state spaces respectively and P a set of conditional probabilities. I, O and S need to be quantised in order to extract qualitative representations of the system. Here, we use a modification of the Growing Neural Gas algorithm [24, 54, 55, 59] for quantisation which is robust for finding the right clusters in the presence of noise (RobustGNG). In order to evaluate the quantisation ability of the algorithm, we performed a clustering task with Gaussian distributions of different types. The algorithm is successful in finding the right number of clusters, by making use of an information-theoretic method, namely the Minimum Description Length (MDL) criterion. The algorithm employs an incremental way of learning where no prior information about the maximal number of clusters or iterations is needed. Its stopping criterion is a measure of graph stability based on MDL, where each node in a graph is a discrete latent variable. To evaluate the extraction of probabilistic machines, we used Noisy Automata where states are Gaussian distributions and transitions are probabilistic. The algorithm was able to infer qualitative states and construct corresponding probabilistic machines which include quantisers for the input space, the output space and the state space, corresponding transitions functions and their probabilities.

The second contribution related to qualitative behaviour models is a method for identifying qualitative states in robot-object interactions [73] (Annex 2.2). The interaction studied here is a robot pushing an object with a single finger. We present means of clustering the sensorimotor data obtained during short exploratory pushes. The sensorimotor data are composed of the starting position and orientation of the finger with respect to the object, and the object displacement that results from the push. Our results demonstrate that the algorithm enumerates states that accord with human judgement. For instance, our system extracts discrete behaviours that correspond to the object turning left, turning right, or moving straight.

We note that the two contributions presented above are similar in spirit, and have complementary roles in this deliverable. The finite state machine model is able to capture complex interactions involving sequences of multiple states and their transition probabilities, while the second contribution focuses on short interactions, and reasons on sensorimotor data captured at the beginning and at the end of each interaction. However, while the state machine is tested on artificial data, the state clustering approach is evaluated on a concrete robot problem, and it includes heuristics that allow the agent to process high-dimensional sensorimotor data efficiently.

1.1.3 Relation to the state-of-the-art

The quantisation algorithm builds upon previous implementations of an algorithm based on Neural Gas algorithm [54, 55], which uses information-theoretic properties to decide the proper number of clusters. The new algorithm is incremental and can work in online settings. A decision to add nodes to a graph is based on an online estimation of error and nodes can also be removed depending on information-theoretic measures. We also incorporated additional efficiency improvements in the learning process.

To extract probabilistic finite-state machines from these dynamical systems, we applied the CrySSMEx algorithm [35] with some modifications to allow the quantisation based on RobustGNG and improve the learning convergence.

The clustering algorithm for extracting qualitative states was inspired by the "push-stability diagram," introduced by Brost [12, 13]. We also used some of the discretisation ideas found in Kuipers' work (e.g. [49]).

1.2 Grasping of novel objects

1.2.1 Planned work

This deliverable reports work related to Task 2.8:

> Task 2.8: Grasping novel objects. Based on our object models, we will investigate the scalability of the system with respect to grasping novel, previously unseen objects. We will demonstrate how the system can execute tasks that involve grasping based on the extracted sensory input (both about the scene and individual objects) and taking into account its embodiment. (M27–M50)

Task 2.8 spans the second half of the project. Grasping novel objects requires (1) the ability to detect novel objects, (2) the ability to *plan* grasps onto novel objects, and (3) the ability to execute the planned grasps robustly. All three points are addressed in this report. The first point is addressed through 3D perceptual grouping. The second point is addressed with a method for planning grasps from partial object snapshots. The third point is addressed with a model of touch-based task stability.

This deliverable contributes to the realisation of the sixth measurable objective:

> Objective 6: Methods for perception and manipulation of objects that enable a robot to actively explore objects, to extend its manipulative skills, and its understanding of these.

The novelty detection and grasping work presented here fulfil both the perception and manipulation objectives. The method for learning graspable parts allows the agent to understand its manipulation skills, by extracting recurrent patterns from the agent's experience.

1.2.2 Actual work performed

A prerequisite for grasping novel objects is detection of these objects in the first place. While this is comparatively easy for simple scenes of isolated objects on a table surface, cluttered scenes containing arbitrary arrangements (such as stacks and piles) of unknown objects still poses a challenge. The first contribution in this section then is a method to segment objects from RGBD images of cluttered scenes. After pre-segmentation of the RGBD input image based on surface normals, surface patches are estimated using a mixture of planes and NURBS (non-uniform rational B-splines) and model selection is employed to find the best representation for the given data. We then construct a graph from surface patches and relations between pairs of patches and perform graph cut to arrive at object hypotheses segmented from the scene. The energy terms for patch relations are learned from user annotated training data, where support vector machines (SVM) are trained to classify a relation as being indicative of two patches belonging to the same object. We show evaluation of the relations and results on a database of different test sets, demonstrating that the approach can segment objects of various shapes in cluttered table top scenes. This work led to a conference and workshop publication at CVPR and CVWW respectively [4, 56] (Annexes 2.6 and 2.3) and submissions to IROS, ICPR and DAGM [57, 58, 47] (Annexes 2.4, 2.5, 2.7).

The second contribution is an extension of the tracking work presented in DR 2.4 [46] with a method for self-assessment of the tracker. Real world settings in object tracking pose challenges such as automatically detecting tracking failure, real-time processing, and robustness to occlusion, illumination, and view point changes. This work presents a 3D tracking system that is capable of overcoming these difficulties using a monocular camera. We present a method of Tracking-State-Detection (TSD) that takes advantage of commercial graphics processors to map textures onto object geometry, to learn textures online, and to recover object pose in real-time. Our system is able to handle 6 DOF object motion during changing lighting conditions, partial occlusion and motion blur while maintaining an accuracy of a few millimetres. Furthermore using TSD we are able to automatically detect occlusions or whether we lost track, and can then trigger a SIFT-based recognition system that is trained during tracking to recover the pose. Evaluations are presented in relation to ground truth pose data and examples present TSD on real-world scenes presented in video sequences. This work led to a conference publication at ROBIO [48] (Annex 2.8).

The third contribution related to grasping novel objects is an agent that has the ability to identify parts by which objects are often grasped [18] (Annex 2.9). As a result, the agent is able to quickly plan grasps onto novel objects that partly resemble objects that it has grasped before. Our agent extracts experience from a set of grasps demonstrated by a teacher. Demonstrations are conducted by placing objects of various shapes and sizes within the robot hand and instructing the robot to close the hand. The final configurations of the hand with respect to the 3D object shapes are used as training data. The agent searches these data for parts that recur in the vicinity of the hand across different grasps. To this end, the agent first extracts shape segments of predefined sizes around the grasping point of each grasp example. This process provides it with a set of prototype *candidates*. The agent then computes pairwise shape similarities between all candidates, and clusters the candidate in the space induced by the similarity measure. The agent only conserves the cluster centres, which altogether form a dictionary of grasp prototypes. By keeping the number of cluster low, we can effectively compress grasping experience into a dictionary that is orders of magnitude smaller than the original set of grasp examples. The dictionary allows the agent to plan grasps from a single partial 3D snapshot of a novel object. The agent attempts to fit all the prototypes to the snapshot, and it executes the grasp that corresponds to the best-fitting prototype. This work led to an ICRA publication [18] (Annex 2.9).

The fourth contribution related to grasping novel objects is a joint model of object grasping parameters, tactile imprints, and task stability [8] (Annex 2.10). In DR 2.4, we presented a general-purpose model of touch-based grasp stability. As noted in the report, however, stability is not an absolute property. Instead, stability largely depends on the task that the agent is performing. For instance, a grasp aimed at seizing a hammer for hitting on a nail needs to be more firm than a grasp aimed at pouring water from a bottle.

We have extended the model of DR 2.4 to include task-related information. The result is a generative model of the class of an object, grasp parameters, task, tactile imprints, and grasp stability. The joint probability of these variables is modelled with a Bayesian network. The model is learned from experiments performed both in simulation and on a real robot. The model allows our agent to reason on any of the variables listed above, given observations of the other variables. For instance, given the tactile feedback gathered after closing the hand on an bottle, the agent is able to decide whether it is safe to use the grasp to pour water off the bottle. If it is not, the agent can compute whether the grasp is good enough to simply transport the bottle. If it is the case, the agent could potentially move the bottle to another location from which it could try a grasp that is better-suited for pouring.

1.2.3 Relation to the state-of-the-art

Various approaches to segment objects either in 2D images or in point clouds exist, where early approaches aimed to formulate generic Gestalt principles to organise 2D scenes into objects. Gestalt principles are also used by Koostra et al. in [38] and [37] where the authors developed a symmetry detector to initialise segmentation based on a Markov Random Field (MRF). Furthermore Koostra et al. developed a quality measure based on Gestalt principles to rank segmentation results. Many state-of-the-art approaches formulate image segmentation as energy minimisation with a Markov Random Field (MRF) [9, 65, 11, 60]. In addition to an appearance model computed from colour and texture, which is commonly used to better distinguish foreground from background, Bergstrom et al. [9] formulate an objective function where it is possible to incrementally add constraints generated through human-robot interaction. In [68] Werlberger et al. propose a variational model for interactive segmentation using a shape prior. This method is based on minimising the Geodesic Active Contour energy. The approach by Hager et al. [28] is able to segment objects from cluttered scenes in point clouds generated from stereo by using a strong prior 3D model of the scene and explicitly modelling physical constraints such as support and handles dynamic changes such as object appearance/disappearance. It is however limited to parametric models (boxes, cylinders), whereas our approach is only limited by the amount and type of training data.

In his groundbreaking paper [33], Horn shows how to approximate the Laplacian by second-order finite differences (FD) on the image grid and solve the resulting algebraic system by a fixed-point scheme. Extensions of Horn's method are too numerous to list here but let us explicitly mention the most recent ones like Harker's and O'Leary's [29, 30] as well as that due to Durou et al. [20], who describe a powerful total-variation-based algorithm capable of resolving discontinuities in the depth map without prior segmentation of the gradient field. Our work relates to the class of kernel methods [21, 50], which can be thought of as mesh-free FEMs in disguise. Similarly, Kovesi applies a basis $\{b_j\}$ of shapelets to the normal adaption problem in scene space [39]. Only a few authors explicitly consider the classical, i.e., non-isogeometric FEM: Hicks employs it for integrating normal fields with three-dimensional support into a foliation of surfaces [32]. Generalisations of Horn's method applicable to such spatially varying normal fields are presented by Balzer [3] and Delaunoy and Prados [17]. None of aforementioned methods is compatible with the geometry representation of contemporary CAD packages. Higher-degree polynomial bases and a multi-scale mechanism are per se possible, at least on polygon meshes, but quite challenging to implement.

A review of the massive body of literature on B-Spline curve fitting would go far beyond the scope of this paper. We briefly give an overview of relevant work and afterwards point out the most common approaches to which we want to apply our methods. One of the most fundamental summaries on B-Splines and least-squares fitting to point-clouds was done in the well know book of Piegl et. al. [51] where they minimise a functional. This method was carefully investigated in [52] where they especially focus on the squared distance function and their approximants used for least-squares fitting. In [66, 67, 10] new point-curve distance functions are introduced to improve the convergence rate and robustness. In [70] Yang et al. propose an active implicit B-Spline model and find the zero set of a bivariate tensor-product B-Spline function using the trust region algorithm [53]. Fitting B-Spline curves to point-clouds in the presence of obstacles is introduced in [22, 23], where they minimise a functional subject to an inequality constraint. Hu et al. [34] present a method where they take advantage of both algebraic and geometric distance minimisation and therefore avoid additional constraints. Often it is necessary to modify an existing curve fitting method to apply to a problem with certain characteristics, such as noise, outliers, unknown degree of freedom (DOF) and so forth [27, 72, 5]. Our approach extends the *Squared Distance Minimisation* (SDM) of [67], i.e. we are modifying their error term for the functional to be minimised. Further we want to overcome the problem of specifying the degree of freedom manually and add control points and knots depending on the error of the curve.

Tracking the pose of an object in image sequences is a classical problem in robot vision, where current approaches aim at improving robustness in tough real-world scenarios [19, 14, 42, 64, 36]. [41] use a combination of edges and textures for tracking. Their approach extracts point features from surface texture and use them together with edges to calculate object pose. This turns out to be very fast as well as robust against occlusion. Our approach not only uses patches but the whole texture, which usually lets the pose converge very quickly to the accurate pose. Since the algorithm runs on the GPU, it is as fast as the method in [41]. More recent approaches aim to solve most of the problems of tracking, such as [63] where the authors are matching the camera image with pre-trained keyframes and then minimising the squared distance of feature points taking into account neighbouring frames. The approach described in [44] uses a modified version of the Active Appearance Model which allows for partial and self occlusion of the objects and for high accuracy and precision. In [16] the authors minimise the optical flow resulting from the projection of a textured model and the camera image. To compensate for shadows and changing lighting they apply an illumination normalisation technique. The work presented in [25] describes an approach for real-time visual servoing using a binocular camera setup to estimate the pose by triangulating a set of feature points. As in our approach [61] takes advantage of robust Monte Carlo particle filtering to determine the pose of the camera with respect to SIFT features, which are localised in 3D using epipolar geometry. Missing in all the above methods is a detection when tracking fails rather than reporting tracking trapped in a local optimum. The proposed tracking state detection (TSD) addresses this problems and we develop an approach to to work fully automatically.

Recent approaches to plan grasps onto novel objects rely on methods that learn a direct mapping from visual cues to grasp parameters. Authors have studied the association of grasping strategies to various kinds of visual cues. Grasps associated to local visual features [62, 45] have the advantage of being easily transferable across objects, as many objects share similar components. However, local features suffer from a poor geometric resolution, which makes it difficult to accurately associate them to the 6D pose of a gripper, let alone finger preshape parameters. Conversely, grasps associated to a model of a whole object [15, 26] benefit from increased geometric robustness, but the resulting models will not apply to novel objects. Authors have explored this trade-off between transferability and robustness by associating grasps to object parts of varying size [2, 6, 31, 43, 71]. An important distinctive point of our work is that we provide the agent with means of optimising this transferability-robustness trade-off internally, by allowing it to select prototypical parts of varying size, depending on their occurrence statistics in the training database. The result is a compact dictionary of parts that lend themselves to grasping.

As argued above, task-related constraints are important for grasp planning. The geometry of a grasp (i.e., the side by which an object is grasped) is often crucial for the execution of a task. This problem has been studied for instance by Xue et al. [69], who manually encoded the expertise about task semantics provided by a human tutor. Another task-related aspect of grasping is that different tasks require different levels of robustness to external object disturbances (in term of the force a grasp is able to apply onto an object). This problem has been studied by Li et al. [40], and more recently by Aleotti et al. [1], who defined task-related grasp quality measures which combined task knowledge with analytical stability measures used in traditional grasp stability studies.

In our work, we combined supervised task learning with experience-based stability learning. This allowed stability to be assessed in a task-oriented manner. This is especially beneficial for energy-efficient control: when a task (e.g., *hand-over*) does not require strong grasping, a relatively smaller gripping force can be applied.

2 Annexes

2.1 Roa et al. "Robust Vector Quantization for Inference of Substochastic Sequential Machines"

Bibliography S. Roa and G.-J. Kruijff: "Robust Vector Quantization for Inference of Substochastic Sequential Machines". Submitted to Journal of Neurocomputing, 2012.

Abstract The article explores the problem of discretizing the continuous evolution of a dynamical system. The article proposes an algorithm to learn a probabilistic discrete state, an input and an output space representation of the system, together with probabilistic transition functions. The method is based on the CrySSMEx algorithm for extracting substochastic finite state machines, and a new Vector quantization algorithm. We performed experiments on Vector quantization with artificial data generated using Gaussian noise distributions. The quantization algorithm is able to find the optimal number of clusters. It induces a good model of the data, avoiding overfitting. Data stemming from Noisy automata were used to test the algorithm for extracting sequential finite state machines. The induced models represent accurately the behavior of these discrete dynamical systems.

Relation to WP The interaction between a robot and an object leads to different object behaviors. These behaviors can be learned by using predicting models inferring the causal relationships in these interactions. Additionally, these models are qualitative representations in which the sensorimotor space is discretized to find meaningful abstractions. The algorithm presented here is used for obtaining those qualitative representations.

2.2 Zurek et al. “Identification of qualitative states from the behaviour of objects”

Bibliography Sebastian Zurek, Marek Kopicki, and Jeremy Wyatt: “Identification of qualitative states from the behaviour of objects”. University of Birmingham, technical report, 2012.

Abstract For a robotic agent interacting with its environment, it is natural to represent its sensory input and motor output as continuous state spaces. This poses a challenge for controlling the behaviour of a robot, since at almost every instant it will observe a novel situation and will have an infinite choice of motor commands that it could deploy. An objective for robotics research is to devise algorithms that can extract qualitative states, in accord with human judgement. We present an algorithm that uses the behaviour of an object, when manipulated and observed by a robot, to discover the qualitative states in perception-action space. Thus we take the definition of a qualitative state to be a set of points in state space that behave similarly under a given action. The algorithm is evaluated by using data from a simulation of a robotic finger pushing an object.

Relation to WP This paper directly addresses the first topic of this deliverable by presenting a method for extracting qualitative states of object behaviours. The paper also shows the applicability of the method to a robot problem.

2.3 Richtsfeld et al. “Towards Scene Understanding Object Segmentation Using RGBD-Images”

Bibliography Richtsfeld, Andreas; Mörwald, Thomas; Prankl, Johann; Balzer, Jonathan; Zillich, Michael; Vincze, Markus: “Towards Scene Understanding Object Segmentation Using RGBD-Images”, Proceedings of the 2012 Computer Vision Winter Workshop (CVWW), 2012.

Abstract We present a framework for detecting unknown 3D objects in RGBD-images and extracting representations suitable for robotics tasks such as grasping. We address cluttered scenes with stacked and jumbled objects where simplistic plane pop-out methods are not sufficient. We start by estimating surface patches using a mixture of planes and NURBS (non-uniform rational B-splines) fitted to the 3D point cloud and employ model selection to find the best representation for the given data. We then construct a graph from surface patches and relations between patches and perform graph cut to arrive at object hypotheses segmented from the scene. The energy terms for patch relations are learned from user annotated training data, where we train a support vector machine (SVM) to classify a relation as being indicative of two patches belonging to the same object given a vector of relation features, such as proximity or color similarity. We show preliminary results demonstrating that the approach can segment objects of various shapes in cluttered table top scenes.

Relation to WP A prerequisite for grasping novel objects (Task 2.8) is the detection of novel objects, i.e. objects for which no instance or category model is available. The above work addresses this problem in a learning framework being essentially only limited by the amount and diversity of training data.

2.4 Richtsfeld et al. "Segmentation of Unknown Objects in Indoor Environments"

Bibliography Richtsfeld, Andreas; Mörwald, Thomas; Prankl, Johann; Zillich, Michael; Vincze, Markus: "Segmentation of Unknown Objects in Indoor Environments", submitted to the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2012.

Abstract We present a framework for segmenting unknown objects in RGBD-images suitable for robotics tasks such as object search, grasping and manipulation. While handling single objects on a table is solved, handling complex scenes poses considerable problems due to clutter and occlusion. After pre-segmentation of the input image based on surface normals, surface patches are estimated using a mixture of planes and NURBS (non-uniform rational B-splines) and model selection is employed to find the best representation for the given data. We then construct a graph from surface patches and relations between pairs of patches and perform graph cut to arrive at object hypotheses segmented from the scene. The energy terms for patch relations are learned from user annotated training data, where support vector machines (SVM) are trained to classify a relation as being indicative of two patches belonging to the same object. We show evaluation of the relations and results on a database of different test sets, demonstrating that the approach can segment objects of various shapes in cluttered table top scenes.

Relation to WP A prerequisite for grasping novel objects (Task 2.8) is the detection of novel objects, i.e. objects for which no instance or category model is available. The above work addresses this problem in a learning framework being essentially only limited by the amount and diversity of training data.

2.5 Richtsfeld et al. "Implementation of Gestalt Principles for Object Segmentation"

Bibliography Richtsfeld, Andreas; Zillich, Michael; Vincze, Markus: "Implementation of Gestalt Principles for Object Segmentation", submitted to the International Conference on Pattern Recognition (ICPR), 2012.

Abstract Gestalt principles have been studied for about a century and were used for various computer vision approaches during the last decades, but became unpopular because the many heuristics employed proved inadequate for many real world scenarios. We show a new methodology to learn relations inferred from Gestalt principles and an application to segment unknown objects, even if objects are stacked or jumbled and tackle also the problem of segmenting partially occluded objects. The relevance of the relations for object segmentation is learned with support vector machines (SVMs) during a training period. We present an evaluation of the relations and show results of the segmentation framework.

Relation to WP A prerequisite for grasping novel objects (Task 2.8) is the detection of novel objects, i.e. objects for which no instance or category model is available. The above work investigates a set of 3D Gestalt principles used in the work reported in Annexes 2.3 and 2.4, for the detection on novel objects based on learning the importance of these Gestalt principles from training examples.

2.6 Balzer et al. “Isogeometric Finite-Elements Methods and Variational Reconstruction Tasks in Vision – A Perfect Match”

Bibliography Balzer, Jonathan; Mörwald, Thomas: “Isogeometric Finite-Elements Methods and Variational Reconstruction Tasks in Vision – A Perfect Match”, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012.

Abstract Inverse problems are abundant in vision. A common way to deal with their inherent ill-posedness is reformulating them within the framework of the calculus of variations. This always leads to partial differential equations as conditions of (local) optimality. In this paper, we propose solving such equations numerically by isogeometric analysis, a special kind of finite-elements method. We will expose its main advantages including superior computational performance, a natural ability to facilitate multi-scale reconstruction, and a high degree of compatibility with the spline geometries encountered in modern computer-aided design systems. To animate these fairly general arguments, their impact on the well-known depth-from-gradients problem is discussed, which amounts to solving a Poisson equation on the image plane. Experiments suggest that, by the isogeometry principle, reconstructions of unprecedented quality can be obtained without any prefiltering of the data.

Relation to WP Fitting of parametric surface models is one part of the work reported in Annex 2.3, where plane and NURBS models compete for an optimal representation of the underlying point cloud data in a model selection framework. As such it is an enabling technology for Task 2.8.

2.7 Mörwald et al. “Fitting B-Spline Curves to Complex Shaped Boundaries”

Bibliography Mörwald, Thomas and Prankl, Johann and Zillich, Michael and Vincze, Markus: “Fitting B-Spline Curves to Complex Shaped Boundaries”, Submitted to the Joint German/Austrian Pattern Recognition Symposium (DAGM-OAGM), 2012.

Abstract Finding the boundary of some region and computing a curve to approximate it best is a common task in computer vision and image processing. This paper describes an approach of fitting B-Splines to 2D point-clouds for robustly finding the boundary of complex shapes. The problems of common B-Spline fitting methods are discussed. New techniques to overcome this problems, namely the Asymmetric Distance Minimization, Error-Adaptive Knot Insertion and Concavity Filling are applied and considered as the main contribution of our work. We will show how our fitting approach leads to satisfying solutions, even by employing a generic initialization scheme and without knowing the required degree of freedom. All improvements are discussed and demonstrated on difficult problems from real sensor data.

Relation to WP The work in this paper is yet another sub-problem of the work reported in Annex 2.6: finding the exact boundary of the data points contributing to a model, projected onto the parametric surface. This is a requirement for constructing precise and “dense” object models with surface patches stitched seamlessly together.

2.8 Mörwald et al. "Self-Monitoring to Improve Robustness of 3D Object Tracking for Robotics"

Bibliography Mörwald, Thomas; Zillich, Michael; Prankl, Johann; Vincze, Markus: "Self-Monitoring to Improve Robustness of 3D Object Tracking for Robotics", IEEE International Conference on Robotics and Biomimetics (ROBIO), 2012.

Abstract In robotics object tracking is needed to steer towards objects, check if grasping is successful, or investigate objects more closely by poking or handling them. While many 3D object tracking approaches have been proposed in the past, real world settings pose challenges such as automatically detecting tracking failure, real-time processing, and robustness to occlusion, illumination, and view point changes. This paper presents a 3D tracking system that is capable of overcoming these difficulties using a monocular camera. We present a method of Tracking-State-Detection (TSD) that takes advantage of commercial graphics processors to map textures onto object geometry, to learn textures online, and to recover object pose in real-time. Our system is able to handle 6 DOF object motion during changing lighting conditions, partial occlusion and motion blur while maintaining an accuracy of a few millimetres. Furthermore using TSD we are able to automatically detect occlusions or whether we lost track, and can then trigger a SIFT-based recognition system that is trained during tracking to recover the pose. Evaluations are presented in relation to ground truth pose data and examples present TSD on real-world scenes presented in video sequences.

Relation to WP While not directly related to detection/grasping of novel objects, this work extends previous work reported in deliverable DR 2.4 on tracking objects for grasping and manipulation. Reasoning about the current state of tracking is an important factor when employing tracking within a larger system that has to make informed decisions, such as aborting a grasping movement in case reliable pose estimates are no longer available.

2.9 Detry et al. "Generalizing Grasps Across Partly Similar Objects"

Bibliography Detry, Renaud; Ek, Carl Henrik; Madry, Marianna; Piater, Justus; Kragic, Danica : "Generalizing grasps across partly similar objects", IEEE International Conference on Robotics and Automation, 2012.

Abstract The paper starts by reviewing the challenges associated to grasp planning, and previous work on robot grasping. Our review emphasizes the importance of agents that generalize grasping strategies across objects, and that are able to transfer these strategies to novel objects. In the rest of the paper, we then devise a novel approach to the grasp transfer problem, where generalization is achieved by *learning*, from a set of grasp examples, a dictionary of object parts by which objects are often grasped. We detail the application of dimensionality reduction and unsupervised clustering algorithms to the end of identifying the size and shape of parts that often predict the application of a grasp. The learned dictionary allows our agent to grasp novel objects which share a part with previously seen objects, by matching the learned parts to the current view of the new object, and selecting the grasp associated to the best-fitting part. We present and discuss a proof-of-concept experiment in which a dictionary is learned from a set of synthetic grasp examples. While prior work in this area focused primarily on shape analysis (parts identified, e.g., through visual clustering, or salient structure analysis), the key aspect of this work is the emergence of parts from *both* object shape *and* grasp examples. As a result, parts intrinsically encode the intention of executing a grasp.

Relation to WP This work is concerned with transferring grasping knowledge across known objects and to novel objects. We developed a method that allows a robot to identify parts by which objects are often grasped, thereby allowing the robot to easily grasp novel objects that contain a familiar part.

2.10 Bekiroglu et al. "A Probabilistic Framework for Task-Oriented Grasp Stability Assessment"

Bibliography Bekiroglu, Yasemin; Song, Dan; Wang, Lu; Kragic, Danica : "A Probabilistic Framework for Task-Oriented Grasp Stability Assessment", KTH Royal Institute of Technology (Technical report), 2012.

Abstract We present a probabilistic framework for grasp modeling and stability assessment. The framework facilitates assessment of grasp success in a goal-oriented way, taking into account both geometric constraints for task affordances and stability requirements specific for a task. We integrate high-level task information introduced by a teacher in a supervised setting with low-level stability requirements acquired through a robot's self-exploration. The conditional relations between tasks and multiple sensory streams (vision, proprioception and tactile) are modeled using Bayesian networks. The generative modeling approach both allows prediction of grasp success, and provides insights into dependencies between variables and features relevant for object grasping.

Relation to WP This work is concerned with the exploitation of touch data *and* task requirements to assess the stability of a grasp. As different tasks impose different constraints on object-gripper bonds, it is important to take tasks into account when assessing stability. The ability to assess stability from touch is particularly important when grasping novel objects, as the configuration of the grasp is less certain than when grasping a known object.

References

[1] J. Aleotti and S. Caselli. Interactive teaching of task-oriented robot grasps. *Robotics and Autonomous Systems*, 58(5):539–550, 2010.

[2] Jacopo Aleotti and Stefano Caselli. Part-based robot grasp planning from human demonstration. In *IEEE International Conference on Robotics and Automation*, 2011.

[3] Jonathan Balzer. A Gauss-Newton Method for the Integration of Spatial Normal Fields in Shape Space. *J. Math. Imaging Vis.* In press.

[4] Jonathan Balzer and Thomas Mörwald. Isogeometric finite-elements methods and variational reconstruction tasks in vision – a perfect match. In *IEEE Conference on Computer Vision and Pattern Recognition*, Providence, RI, USA, Jun 2012.

[5] L Barbieri, F Bruno, M Muzzupappa, and J Pernot. Constrained fitting of B-spline curves based on the force density method. In *International Conference on Innovative Methods in Product Design*, volume 46, 2011.

[6] C. Bard and J. Troccaz. Automatic preshaping for a dextrous hand from a simple description of objects. In *IEEE International Workshop on Intelligent Robots and Systems*, pages 865–872. IEEE, 1990.

[7] Yasemin Bekiroglu, Janne Laaksonen, Jimmy Alison Jørgensen, Ville Kyrki, and Danica Kragic. Assessing grasp stability based on learning and haptic data. *IEEE Transactions on Robotics*, 2011.

[8] Yasemin Bekiroglu, Dan Song, Lu Wang, and Danica Kragic. A probabilistic framework for task-oriented grasp stability assessment. Technical report, KTH Royal Institute of Technology, 2012.

[9] N. Bergström, M. Björkman, and Danica Kragic. Generating object hypotheses in natural scenes through human-robot interaction. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 827–833. IEEE, 2011.

[10] A Blake and M Isard. *Active Contours*, volume 17. Springer, 1998.

[11] Y Y Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary amp; region segmentation of objects in N-D images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 105 –112 vol.1, 2001.

[12] Randy C. Brost. Planning robot grasping motions in the presence of uncertainty. Technical Report CMU-RI-TR-85-12, The Robotics Institute, Carnegie-Mellon University, Pittsburgh, PA, July 1985.

[13] Randy C. Brost. Automatic grasp planning in the presence of uncertainty. *Int. J. Robotics Research*, 7(1):3–17, 1988.

[14] J. Chestnutt, S. Kagami, K. Nishiwaki, J. Kuffner, and T. Kanade. Gpu-accelerated real-time 3d tracking for humanoid locomotion. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007.

[15] Charles de Granville, Joshua Southerland, and Andrew H. Fagg. Learning grasp affordances through human demonstration. In *IEEE International Conference on Development and Learning*, 2006.

[16] Hans de Ruiter and Beno Benhabib. Visual-model-based, real-time 3d pose tracking for autonomous navigation: methodology and experiments. *Autonomous Robots*, 25:267–286, 2008.

[17] A. Delaunoy and E. Prados. Gradient Flows for Optimizing Triangular Mesh-based Surfaces: Applications to 3D Reconstruction Problems Dealing with Visibility. *Int. J. Comput. Vision*, 95:100–123, 2011.

[18] Renaud Detry, Carl Henrik Ek, Marianna Madry, Justus Piater, and Danica Kragic. Generalizing grasps across partly similar objects. In *IEEE International Conference on Robotics and Automation*, 2012.

[19] T. Drummond and R. Cipolla. Real-time tracking of complex structures with on-line camera calibration. In *British Machine Vision Conference (BMVC'99*, pages 574–583, 1999.

[20] J.-D. Durou, J.-F. Aujol, and F. Courteille. Integrating the Normal Field of a Surface in the Presence of Discontinuities. *Proc. EMMCVPR*, pages 261–273, 2009.

[21] S. Ettl, J. Kaminski, M. Knauer, and G. Häusler. Shape reconstruction from gradient data. *Appl. Optics*, 47(12):2091–2097, 2008.

[22] S Flory. Fitting curves and surfaces to point clouds in the presence of obstacles. *Computer Aided Geometric Design*, 26(2):192–202, 2009.

[23] S Flory and M Hofer. Constrained curve fitting on manifolds. *Computer-Aided Design*, 40(1):25–34, 2008.

[24] Bernd Fritzke. A growing neural gas network learns topologies. In *Advances in Neural Information Processing Systems 7*, pages 625–632. MIT Press, 1995.

[25] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendón-Mancha. Binocular visual tracking and grasping of a moving object with a 3d trajectory. *Journal of Applied Research and Technology*, 7(03):259–274, 2009.

[26] C. Goldfeder, M. Ciocarlie, H. Dang, and P.K. Allen. The Columbia grasp database. In *IEEE International Conference on Robotics and Automation*, 2009.

[27] O Grove. From CT to NURBS: Contour Fitting with B-spline Curves. *Computer Aided Design And Applications*, 8(1):3–21, 2011.

[28] Gregory D Hager and Ben Wegbreit. Scene parsing using a prior world model. *The International Journal of Robotics Research*, 2011.

[29] M. Harker and P. O'Leary. Least squares surface reconstruction from measured gradient fields. *Proc. CVPR*, 1:1–7, 2008.

[30] M. Harker and P. O'Leary. Least squares surface reconstruction from gradients: Direct algebraic methods with spectral, Tikhonov, and constrained regularization. *Proc. CVPR*, 1:2529–2536, 2011.

[31] A. Herzog, P. Pastor, M. Kalakrishnan, L. Righetti, T. Asfour, and S. Schaal. Template-based learning of grasp selection. In *The PR2 Workshop (Workshop at IROS'11)*, 2011.

[32] R. Hicks. Designing a mirror to realize a given projection. *J. Opt. Soc. Am. A*, 22(2):323–330, 2005.

[33] B. Horn. Height and gradient from shading. *Int. J. Comput. Vision*, 5(1):37–75, 1999.

[34] Mingxiao Hu, Jieqing Feng, and Jianmin Zheng. An additional branch free algebraic B-spline curve fitting method. *The Visual Computer*, 26(6-8):801–811, 2010.

[35] H. Jacobsson. The crystallizing substochastic sequential machine extractor - `CrySSMEx`. *Neural Computation*, 18(9):2211–2255, 2006.

[36] Georg Klein and Tom Drummond. Robust visual tracking for non-instrumented augmented reality. In *ISMAR IEEE/ACM International Symposium on Mixed and Augmented Reality*, 2003.

[37] Gert Kootstra, Niklas Bergström, and Danica Kragic. Fast and Automatic Detection and Segmentation of Unknown Objects. In *Humanoids*, Bled, 2011.

[38] Gert Kootstra, Niklas Bergström, and Danica Kragic. Gestalt Principles for Attention and Segmentation in Natural and Artificial Vision Systems. In *SPME*, Shanghai, 2011.

[39] P. Kovesi. Shapelets correlated with surface normals produce surfaces. *Proc. ICCV*, 2:994–1001, 2005.

[40] Z. Li and S. Sastry. Task oriented optimal grasping by multifingered robot hands. In *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, volume 4, pages 389–394, January 2003.

[41] Lucie Masson, Michel Dhome, and Frederic Jurie. Robust real time tracking of 3d objects. In *International Conference on Pattern Recognition, ICPR*, 2004.

[42] Philipp Michel, Joel Chestnutt, Satoshi Kagami, Koichi Nishiwaki, James Kuffner, and Takeo Kanade. Gpu-accelerated real-time 3d tracking for humanoid autonomy. In *JSME Robotics and Mechatronics Conference (ROBOMEC'08)*, June 2008.

[43] A. T. Miller, S. Knoop, H. Christensen, and P. K. Allen. Automatic grasp planning using shape primitives. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1824–1829, 2003.

[44] Pradit Mittrapiyanuruk, Guilherme N. Desouza, and Avinash C. Kak. Accurate 3d tracking of rigid objects with occlusion using active appearance models. In *WACV/MOTION*, pages 90–95, 2005.

[45] L. Montesano and M. Lopes. Learning grasping affordances from local visual descriptors. In *IEEE International Conference on Development and Learning*, 2009.

[46] Thomas Mörwald, Marek Kopicki, Rustam Stolkin, Jeremy Wyatt, Sebastian Zurek, Michael Zillich, and Markus Vincze. Predicting the unobservable, visual 3d tracking with a probabilistic motion model. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA11)*, May 2011.

[47] Thomas Mörwald, Johann Prankl, Michael Zillich, and Markus Vincze. Fitting b-spline curves to complex shaped boundaries. In *Joint German/Austrian Pattern Recognition Symposium (DAGM-OAGM) (submitted)*, Graz, Austria, Aug 2012.

[48] Thomas Mörwald, Michael Zillich, Johann Prankl, and Markus Vincze. Self-monitoring to improve robustness of 3d object tracking for robotics. In *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Phuket, Thailand, Dec 2011.

[49] Jonathan Mugan and Benjamin Kuipers. Autonomous learning of high-level states and actions in continuous environments. *IEEE Trans. Autonomous Mental Development*, 4(1):70–86, 2012.

[50] H.-S. Ng, T.-P. Wu, and C.-K. Tang. Surface-from-gradients without discrete integrability enforcement: A gaussian kernel approach. *IEEE T. Pattern Anal.*, 32:2085–2099, 2010.

[51] Les Piegl and Wayne Tiller. *The NURBS book.* Monographs in visual communication. Springer, 1996.

[52] Helmut Pottmann and Michael Hofer. Geometry of the Squared Distance Function to Curves and Surfaces. *Visualization and mathematics III*, (90):223–244, 2003.

[53] M.J.D. Powell. On the global convergence of trust region algorithms for unconstrained optimization. *Math. Prog.*, 29:297–303, 1984.

[54] A.K. Qin and P.N. Suganthan. Robust growing neural gas algorithm with application in cluster analysis. *Neural Networks*, 17(8-9):1135 – 1148, 2004. New Developments in Self-Organizing Systems.

[55] K Qin and N Suganthan. Enhanced neural gas network for prototype-based clustering. *Pattern Recognition*, 38(8):1275–1288, 2005.

[56] Andreas Richtsfeld, Thomas Mörwald, Johann Prankl, Jonathan Balzer, Michael Zillich, and Markus Vincze. Towards scene understanding – object segmentation using rgbd-images. In *Proceedings of the 2012 Computer Vision Winter Workshop (CVWW)*, Mala Nedelja, Slovenia, February 2012.

[57] Andreas Richtsfeld, Thomas Mörwald, Johann Prankl, Michael Zillich, and Markus Vincze. Segmentation of unknown objects in indoor environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (submitted)*, Vilamoura, Algarve, Portugal, Oct 2012.

[58] Andreas Richtsfeld, Michael Zillich, and Markus Vincze. Implementation of gestalt principles for object segmentation. In *21st International Conference on Pattern Recognition (ICPR) (submitted)*, Tsukuba, JAPAN, Nov 2012.

[59] Sergio Roa and Geert-Jan Kruijff. Robust vector quantization for inference of substochastic sequential machines. *Neurocomputing*, 2012. submitted.

[60] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "Grab-Cut": interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, August 2004.

[61] J.R. Sánchez, H. Álvarez, and D. Borro. Towards real time 3d tracking and reconstruction on a gpu using monte carlo simulations. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 185 –192, Oct. 2010.

[62] A. Saxena, J. Driemeyer, and A. Y. Ng. Robotic Grasping of Novel Objects using Vision. *International Journal of Robotics Research*, 27(2):157, 2008.

[63] L. Vacchetti, V. Lepetit, and P. Fua. Stable real-time 3d tracking using online and offline information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004.

[64] Luca Vacchetti, Vincent Lepetit, and Pascal Fua. Combining edge and texture information for real-time accurate 3d camera tracking. In *IEEE/ACM International Symposium on Mixed and Augmented Reality*, 2004.

[65] S Vicente, V Kolmogorov, and C Rother. Joint optimization of segmentation and appearance models. *2009 IEEE 12th International Conference on Computer Vision*, (Iccv):755–762, 2009.

[66] Wenping Wang, Helmut Pottmann, and Yang Liu. Fitting B-Spline Curves to Point Clouds by Squared Distance Minimization. *ACM TOG*, page 41, 2004.

[67] Wenping Wang, Helmut Pottmann, and Yang Liu. Fitting B-spline curves to point clouds by curvature-based squared distance minimization. *ACM Transactions on Graphics*, 25(2):214–238, 2006.

[68] Manuel Werlberger, Thomas Pock, Markus Unger, and Horst Bischof. A Variational Model for Interactive Shape Prior Segmentation and Real-Time Tracking. In *International Conference on Scale Space and Variational Methods in Computer Vision (SSVM)*, Voss, Norway, 2009.

[69] Z. Xue, J. Zoellner, and R. Dillmann. Automatic optimal grasp planning based on found contact points. In *IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics*, pages 1053–1058, 2008.

[70] Zhouwang Yang, Jiansong Deng, and Falai Chen. Fitting unorganized point clouds with active implicit B-spline curves. *The Visual Computer*, 21(8-10):831–839, 2005.

[71] Li (Emma) Zhang, Matei Ciocarlie, and Kaijen Hsiao. Grasp evaluation with graspable feature matching. In *RSS Workshop on Mobile Manipulation: Learning to Manipulate*, 2011.

[72] Xiuyang Zhaoa, Caiming Zhang, Bo Yang, and Pingping Li. Adaptive knot placement using a GMM-based continuous optimization algorithm in B-spline curve approximation. *Computer-Aided Design*, 43(6):598–604, 2011.

[73] Sebastian Zurek, Marek Kopicki, and Jeremy Wyatt. Identification of qualitative states from the behaviour of objects. Technical report, University of Birmingham, 2012.

Submitted to Journal of Neurocomputing, 2012
Provided as part of DR 2.5: Qualitative behaviour models, grasping of novel objects

Robust Vector Quantization for Inference of Substochastic Sequential Machines

Sergio Roa, Geert-Jan Kruijff

German Research Center for Artificial Intelligence (DFKI GmbH)
Stuhlsatzenhausweg 3
D-66123, Saarbrücken, Germany

Abstract

The article explores the problem of discretizing the continuous evolution of a dynamical system. The article proposes an algorithm to learn a probabilistic discrete state, an input and an output space representation of the system, together with probabilistic transition functions. The method is based on the CrySSMEx algorithm for extracting substochastic finite state machines, and a new Vector quantization algorithm. We performed experiments on Vector quantization with artificial data generated using Gaussian noise distributions. The quantization algorithm is able to find the optimal number of clusters. It induces a good model of the data, avoiding overfitting. Data stemming from Noisy automata were used to test the algorithm for extracting sequential finite state machines. The induced models represent accurately the behavior of these discrete dynamical systems.

Keywords: Vector Quantization, Probabilistic Finite-State Machines, Dynamical Systems

1. Introduction

In this article we address the problem of extracting a substochastic finite state machine from a dynamical system. A dynamical system can be described as a sequence of observations, where its corresponding state space encompasses the possible events during the evolution of the system. The

Email addresses: `sergio.roa@dfki.de` (Sergio Roa), `gj@dfki.de` (Geert-Jan Kruijff) current point (state) in the state space can be affected by the input and zero or more previous states. Moreover, one can find patterns in the behavior of dynamical systems in the form of abstract representations like vector clusters. These clusters can be groups of vectors in a time frame, not necessarily at a specific time step, and not necessarily grouped by geometrical similarity. By applying a discretization algorithm, the patterns and rules which are extracted from some dynamical system can be used for other purposes like planning and pattern classification through time (concept learning) [1]. The CrySSMEx algorithm has been used to extract substochastic finite state machines from Recurrent Neural Networks (RNNs) [2, 3], which are also dynamical systems.

CrySSMEx has been tested for inferring machines which model formal grammars learned by RNNs (mainly regular grammars but also context-free grammars). It has also been used in the analysis of a chaotic function and in a simulated robotic scenario [4]. In general terms, CrySSMEx can either perform a quantization of the input and output spaces, or the quantization can explicitly be given in form of symbolic representations. Then, a mechanism based on the minimization of conditional entropy between pairs of inputs and states, and output or next states, is performed to construct the probabilistic automaton, gradually splitting the state space.

However, the quantization mechanism for input and output spaces in CrySSMEx is quite simple, is based on a regular partitioning of the spaces. This may lead to more complex machines, inconsistent ones, or simply longer learning times in the presence of noise.

Therefore, in this paper we present a new method for quantizing spaces by means of an algorithm based on Growing Neural Gas (GNG) [5, 6, 7, 8]. This algorithm (RobustGNG) finds a model which is robust to noise bias and variance in data sets. Moreover, the algorithm is suitable for handling additional data to update the model, which suggests a life-long learning procedure. Parameters like the maximum number of clusters, the maximum number of iterations, or a minimum quantization error do not need to be set a priori. The algorithm implements a stopping criterion based on a learning progress measure and a minimum description length criterion.

The original CrySSMEx algorithm has also some drawbacks that prevent the extraction of optimal machines for dynamical systems with certain symmetrical features and long dependencies among input, state and output tuples across time. We carried out experiments with stochastic systems with such characteristics and we improved CrySSMEx correspondingly to handle these issues.

The new clustering algorithm is tested with artificial data sets where clusters in 2D can be easily identified. RobustGNG is then integrated into CrySSMEx to analyse the behavior of a noisy automaton and infer the corresponding machine. The vector quantization algorithm is able to stop after the right number of clusters has been found. We are also able to extract probabilistic automata from noisy stochastic dynamic systems, which goes beyond a simple clustering of the data sets.

2. Robust Vector Quantization

Quantization or more specifically *Vector Quantization* is used to partition probability distributions [9]. In this way, it can model probability density functions. A related field of research is cluster analysis [10]. The idea is to map a set of high-dimensional vectors to a prototype vector, thereby identifying clusters in the data set that are statistically similar. In order to solve the clustering problem different machine learning and pattern recognition algorithms have been employed, including fuzzy clustering, neural networks, kernel density estimation, nearest-neighbors, principal component analysis, self-organizing maps, among others.

Following the machine learning terminology, a variable that represents a cluster is defined as a latent variable. This variable defines assignments of data points to specific components of a mixture of distributions [11]. Technically, we have a data set $\mathcal{D}$ of points $\mathbf{x}_k \in \mathbb{R}^d$ where $d \in \mathbb{N}$ is the space dimensionality, a set of subsets or distributions $\{S_i \subset \mathcal{D}\}$, a prototype vector $\mathbf{x}_i$ that represents the cluster S_i, and the latent variable or index i. Moreover, some distant function or metric δ is used to evaluate the similarities among points in the data set.

Definition 1. *A quantizer Λ is a quadruple $\langle C, S, \delta, \gamma \rangle$, where C is a set of prototype vectors, $S \subseteq \mathcal{D}$ is a collection of M clusters, δ is a distance or metric function and $\gamma : S \to \{1, 2, \ldots, M\}$ is a function that maps an element $\mathbf{x} \in S_i \subset S$ to its corresponding prototype index i.*

To find proper quantizations a learning algorithm has to be implemented. The work presented here builds upon previous implementations of an incremental algorithm called Growing Neural Gas [5], which is based on a graph whose nodes adapt to the topology of the probability distribution.

Definition 2. *A Growing Neural Gas (GNG) network [5] is a graph $\mathcal{G} = \langle A, \mathcal{C} \rangle$ where A is a set of M (cf. Def. 1) nodes, in which each node $c \in A$ has an associated weight $\mathbf{w}_c \in \mathbb{R}^d$. There exists a set of neighborhood connections $\mathcal{C}$ which are unweighted and symmetric. For each node c, there exists a possibly empty set of neighborhood connections $N_c = \{i \in A \mid (c, i) \in \mathcal{C}\}$. A GNG network can be used as learning algorithm for a quantizer, by using the weights and indices associated to nodes in A as the prototype vectors and indices respectively.*

The algorithm has several strengths. It is an incremental algorithm which starts with 2 nodes and gradually adds more nodes in regions where a global error measure is high. By using Hebbian learning, a winner node and its neighbors gradually move towards regions with higher errors. In this way, the quantization error is minimized and the prototype nodes preserve the topology. However, usually a proper quantization needs to be found so that clusters in the data set are represented by one or at least only a few nodes. For cluster analysis, this algorithm has several issues, some of them described in [7, 8]. For instance, the method suffers from the overfitting problem given the fact that one has to always define the maximum number of nodes M needed for representation. Another important issue is the presence of noise in the form of outliers. As described in [7, 8], some techniques were implemented to improve the convergence of the algorithm for efficient cluster analysis and in the presence of noisy data. Specifically, in [7], an outlier resistant strategy was applied that reduces the influence of outliers in the node weight adaptation process. Additionally, a cluster repulsion scheme was also designed to avoid the prototype coincident problem, namely that two nodes compete for being the representatives of a certain cluster. The complexity of the model is penalized by a Minimum Description Length (MDL) criterion. In this way, models with a minimal MDL can be selected which optimize the number of clusters.

In the algorithms presented in [7, 8] parameters like the maximum number of nodes and the maximum number of iterations for a training epoch are still used. In this article, we propose a new algorithm that is less dependent on a priori parameters. It incorporates also an online calculation of error suitable for setting learning rates and to decide locations for new nodes to insert. The procedure can also decide when to stop according to the MDL principle and a notion of network stability.

In the following sections, we describe and explain the calculation of the different parameters involved in the learning process: the learning rates, the weight updates, the node insertion criterion and the Minimum Description Length calculation. Finally, we present the whole learning algorithm, experimental results and evaluation.

2.1. Learning Parameters

2.1.1. Learning rates and weights update

In [5], fixed learning rates for a winner node s_1 and its neighbors were employed. To balance the contribution of each of them, in [7, 8] learning rates are monotically decreased taking into account the current iteration in an learning epoch and initial and final expected learning rate values. In this work, we propose a new method for obtaining the rates that is dependent on an instantaneous calculation of mean error change ratio that was successful in the experiments.

To obtain an error change ratio, we compute an estimation of the current mean error and a previous mean error. A similar approach was followed in [6], but in the context of a supervised learning task in a non-stationary distribution where the error in prediction is calculated from the expected distance to a target vector. In that work, a ratio is obtained from a short-term average error and a long-term average error to obtain a quality measure for learning that is used for calculating the learning rate. Here, the error change ratio calculation is inspired by the work presented in [12], where an exploratory learning agent selects a learning sample via a learning progress measure derived from a decrease in the mean error rate in prediction.

We obtain a harmonic mean error rate. This way we obtain error measures that are not strongly influenced by outliers. Assuming a data point $\mathbf{x}(t)$ is presented to the network and the corresponding winner weight $\mathbf{w}_{s_1}(t)$ is activated, the inverse error $e_{s_1}(t)$ associated to the node s_1 is:

$$e_{s_1}(t) = \|\mathbf{x}(t) - \mathbf{w}_{s_1}(t)\|^{-1} \tag{1}$$

In this case, we assume $\| \cdot \|$ to be the Euclidean norm. Given a time window parameter τ during which a node s_1 was selected as winner node, a harmonic smoothed mean error rate for the node s_1 at the current time step t and at a previous time step $t - \tau$ is calculated as follows:

$$\begin{aligned} \langle e_{s_1}(t) \rangle &= \left(\tfrac{1}{\theta+1} \textstyle\sum_{i=0}^{\theta} e_{s_1}(t-i) \right)^{-1} \\ \langle e_{s_1}(t-\tau) \rangle &= \left(\tfrac{1}{\theta+1} \textstyle\sum_{i=0}^{\theta} e_{s_1}(t-i-\tau) \right)^{-1}, \end{aligned} \tag{2}$$

where θ is a smoothing parameter.

For every node $k \in N_{s_1} \cup \{s_1\}$, the learning rate $\eta_k(t)$ is obtained in the following way:

$$\eta_k(t) = \begin{cases} \eta_k & \text{if } \exp\left(\frac{\langle e_k(t)\rangle}{\langle e_k(t-\tau)\rangle} - 1\right) > 1 \\ \exp\left(\frac{\langle e_k(t)\rangle}{\langle e_k(t-\tau)\rangle} - 1\right)\eta_k & \text{otherwise,} \end{cases} \tag{3}$$

whereby the default learning rate η_k is modulated by the exponential of the error change ratio $\exp\left(\frac{\langle e_k(t)\rangle}{\langle e_k(t-\tau)\rangle} - 1\right)$. We call this value a learning quality measure. This function is depicted in Figure 1 and serves to moderate the impact of the default learning rate for weights update.

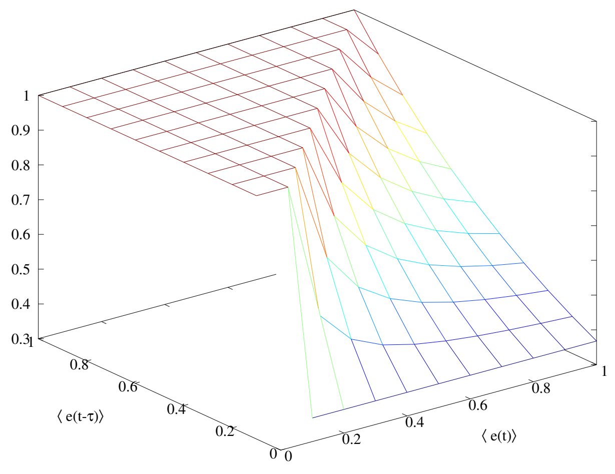


Figure 1: Learning quality function.

In the weight adaptation rule we also incorporated some strategies for outlier resistance based on the work described in [7]. The original formula in [5] is fragile to noisy environments and sensitive to the order of input vectors $\mathbf{x}$. The new proposed rule has the form:

$$\Delta\mathbf{w}_k(t) = \eta_k(t)\sigma_k(t)\frac{\mathbf{x}(t) - \mathbf{w}_k(t)}{\|\mathbf{x}(t) - \mathbf{w}_k(t)\|}, \tag{4}$$

where $\eta_k(t)$ is the learning rate in Eq. 3 and $\sigma_k(t)$ is the parameter for outlier resistance. From now on, we assume the time variable $t = 0$ at the beginning of a growth stage, that is, when a new node is added. The factor $\sigma_k(t)$ is used instead of the absolute distance information $\|\mathbf{x}(t) - \mathbf{w}_k(t)\|$, to mitigate the influence of outliers in the weight adaptation process. $\sigma_k(t)$ is calculated by using a historical restricting distance information $d_k(t)$ [7]:

$$\sigma_k(t) = \begin{cases} d_k(t) & \text{if } \|\mathbf{x}(t) - \mathbf{w}_k(t)\| \geq d_k(t-1) \\ \|\mathbf{x}(t) - \mathbf{w}_k(t)\| & \text{if } \|\mathbf{x}(t) - \mathbf{w}_k(t)\| < d_k(t-1) \end{cases} \tag{5}$$

where $d_k(t)$ serves as a restricting distance for $\mathbf{w}_k(t)$. $d_k(t)$ is updated when k becomes the winner node, i.e., when $k(t) = s_1(t)$:

$$d_k(t) = \begin{cases} \left(\frac{1}{2}\left(d_k(t-1)^{-1} + \|\mathbf{x}(t) - \mathbf{w}_k(t)\|^{-1}\right)\right)^{-1} & \text{if } \|\mathbf{x}(t) - \mathbf{w}_k(t)\| \geq d_k(t-1) \\ \frac{1}{2}\left(d_k(t-1) + \|\mathbf{x}(t) - \mathbf{w}_k(t)\|\right) & \text{if } \|\mathbf{x}(t) - \mathbf{w}_k(t)\| < d_k(t-1), \end{cases} \tag{6}$$

and

$$d_k(0) = \left(\frac{1}{N}\sum_{i=1}^{N} \|\mathbf{x}_i - \mathbf{w}_k^0\|^{-1}\right)^{-1}, \tag{7}$$

where $\mathbf{w}_k^0$ is the initial weight at the beginning of the growth stage and $N = |\mathcal{D}|$. Thus, $d_k(t)$ is initialized at the beginning and reinitialized when a new node is added to the network or when one or more nodes are deleted. We added a value $\varepsilon = 10^{-2}$ to Euclidean distances in order to avoid very big restricting distance values.

As previously mentioned, we also incorporated a factor in the learning rule that acts as a repulsive force and reduces the possibility of the prototype coincident problem, based on the work described in [7]. The final updating rule is then:

$$\begin{aligned} \Delta\mathbf{w}_{s_1}(t) &= \eta_{s_1}(t)\sigma_{s_1}(t)\tfrac{\mathbf{x}(t)-\mathbf{w}_{s_1}(t)}{\|\mathbf{x}(t)-\mathbf{w}_{s_1}(t)\|} \\ \Delta\mathbf{w}_i(t) &= \eta_i(t)\sigma_i(t)\tfrac{\mathbf{x}(t)-\mathbf{w}_i(t)}{\|\mathbf{x}(t)-\mathbf{w}_i(t)\|} \\ &\quad + \exp\left(-\tfrac{\|\mathbf{w}_i(t)-\mathbf{w}_{s_1}(t)\|}{\zeta_i(t)}\right) \cdot \beta\tfrac{\sum_j \|\mathbf{w}_j(t)-\mathbf{w}_{s_1}(t)\|}{|N_{s_1}|}\tfrac{\mathbf{w}_i(t)-\mathbf{w}_{s_1}(t)}{\|\mathbf{w}_i(t)-\mathbf{w}_{s_1}(t)\|} \quad \forall i, j \in N_{s_1}, \end{aligned} \tag{8}$$

where β is usually set to 2 and $\zeta_i(t)$ is a parameter that controls the weakening effect of the distance between winner s_1 and its neighbors. Here, $\zeta_i(t)$ is obtained from the current mean error rate (cf. Eq. 2):

$$\begin{aligned} \zeta_i(t) &= \rho\langle e_i(t)\rangle \\ \zeta_i(0) &= \epsilon. \end{aligned} \tag{9}$$

$\zeta_i(0)$ is the initial value when the node is added to the network, which is set to $\epsilon = 10^{-3}$, and ρ is a constant set to 0.4.

2.1.2. Criterion for node insertion

Definition 3. *The node insertion requirement is fulfilled if for all nodes $c \in A$ the mean error rate e_c is not reduced after T_e learning epochs.*

A new node is added when the node insertion requirement is satisfied. The node is inserted in the proximity of a node q with maximal insertion criterion and its neighbor f with highest insertion criterion. An insertion criterion K_c is defined in our work simply as the highest mean error rate:

$$\begin{aligned} K_q(t) &= \arg\max_{c\in A}(\langle e_c(t)\rangle) \\ K_f(t) &= \arg\max_{c\in N_q}(\langle e_c(t)\rangle) \end{aligned} \tag{10}$$

In [5, 7, 8, 6] the weight calculation is based on an interpolation of the weight vectors of two nodes. The shortcoming of this method is that new nodes can not be added in proper locations due to the topological structure of some datasets. Therefore, in our method the location of the new prototype r is in fact calculated from the location of q and the direction of the mean average error vector $\langle \mathbf{e_q} \rangle$ associated to this node. This quantity is calculated in a similar way as in Eq. 2. When the data point $\mathbf{x}(t)$ is presented to the network, the mean error rate vector of the winner node s_1 is calculated:

$$\langle \mathbf{e}_{s_1}(t)\rangle = \left(\frac{1}{\theta+1}\sum_{i=0}^{\theta} \mathbf{e}_{s_1}(t-i)\right)^{-1}, \tag{11}$$

with error vector $\mathbf{e}_{s_1}(t) = (\mathbf{x}(t) - \mathbf{w}_{s_1}(t))^{-1}$, where each component of $\mathbf{a}^{-1}$ is the inverse of the corresponding component of some vector $\mathbf{a}$. Thus, we set the weight for an inserted node r whose parent node is q as:

$$\mathbf{w}_r(t) = \mathbf{w}_q(t) + 2\langle \mathbf{e}_q(t)\rangle. \tag{12}$$

In some cases, a dislocated node will be deleted according to a criterion explained in Section 2.1.4. In this case, we add two nodes in the next growth stage, by picking up two nodes with maximal insertion criteria. This method accelerates the growth of the network, avoiding convergence to suboptimal results.

2.1.3. Minimum Description Length Principle

We use a Minimum Description Length criterion as proposed in [8] to determine the optimal number of clusters. It is an information-theoretic measure that balances the complexity of the graph and its error.

Definition 4. *Given a data set $\mathcal{D}$ and the set of prototype node weights $\mathcal{W}$, the MDL is defined as [8]:*

$$MDL(\mathcal{D}) = modelL(\mathcal{D}, \mathcal{W}) + errorL(\mathcal{D}, \mathcal{W}), \tag{13}$$

where $errorL(\mathcal{D}, \mathcal{W})$ is the total encoding length or model efficiency and $modelL(\mathcal{D}, \mathcal{W})$ is the model complexity.

The total encoding length and the model complexity for a network $\mathcal{G}$ with M nodes are calculated as follows:

$$\begin{aligned} \text{errorL}(\mathcal{D}, \mathcal{W}) &= \kappa \sum_{i=1}^{M} \sum_{\mathbf{x} \in S_i} \sum_{k=1}^{d} \max\left(\log_2\left(\frac{\|\mathbf{x}_k - \mathbf{w}_{ik}\|}{\varepsilon}\right), 1\right) \\ \text{modelL}(\mathcal{D}, \mathcal{W}) &= KM + N \log_2 M, \end{aligned} \tag{14}$$

where $N = |\mathcal{D}|$, d is the dimension of input vectors, ε is a data accuracy constant usually set to 10^{-4}. K is the number of bits needed to encode a single data vector, which is obtained according to the average value range ν and data accuracy ε: $K = \lceil \log_2\left(\frac{\nu}{\varepsilon}\right) \rceil$. Finally, κ is a parameter to balance the contribution of the model complexity and model efficiency, which is here usually set to 1.3. By setting $\kappa > 1$, we give more weight to the network accuracy in terms of error. The value range is calculated by obtaining the average value in the data set and substracting its lower limit.

2.1.4. Criteria for node deletion

We implemented two different node deletion criteria. The methods solve issues regarding the rearrangement of the network where nodes are not properly located.

In our work, a node is called inactive and can be deleted if at the end of a growth stage it is not a representative for any vector in the dataset. Usually, new prototypes adapt quickly so that they approach regions containing data. However, in some cases clusters might be already represented by other prototypes and the repulsive forces contribute to the isolation of a node. The second method tackles the issue of identifying nodes that are not properly located in the network, so that the clusters they describe are suboptimal. The problem was first observed and described in [8]. They solved the problem by means of an expensive procedure that identifies a set of dislocated prototypes and relocates them. In our case, we simplified the method by allowing only one possible deletion at each growth stage. As in [8], we use the MDL criterion to assess whether a node is dislocated. Assuming that one prototype f is removed, if the MDL value calculated based on the set of nodes $\mathcal{C} \setminus \{f\}$ is smaller than that of $\mathcal{C}$, i.e., if:

$$\begin{aligned}\Delta\mathrm{MDL}(\mathcal{D}, f) \;=\; & -K + N(\log_2(M-1) - \log_2(M) \\ & +\kappa\left(\sum_{i=1, i\neq f}^{M} \sum_{\mathbf{x}\in S_i} \sum_{k=1}^{d} \max\left(\log_2\left(\frac{\|\mathbf{x}_k - \mathbf{w}_{ik}\|}{\varepsilon}\right), 1\right)\right. \\ & \left. - \sum_{i=1}^{M} \sum_{\mathbf{x}\in S_i} \sum_{k=1}^{d} \max\left(\log_2\left(\frac{\|\mathbf{x}_k - \mathbf{w}_{ik}\|}{\varepsilon}\right), 1\right)\right) < 0,\end{aligned} \tag{15}$$

we regard the removed prototype f as dislocated.

Definition 5. *If at the end of a growth stage* $\exists c \in \mathcal{C}$ *such that* $\arg\min_c \Delta MDL(\mathcal{D}, c)$, $\Delta MDL(\mathcal{D}, c) < 0$, *we mark the dislocated node c for deletion. This procedure aims to avoid the overfitting problem.*

2.1.5. Robust GNG Algorithm

We implemented the learning algorithm in such a way that it can decide when to stop. This stopping criterion is based on evaluating the MDL after a number of T_m learning epochs after which no minimal graph has been found, in terms of MDL. The learning process is described in Algorithm 1.

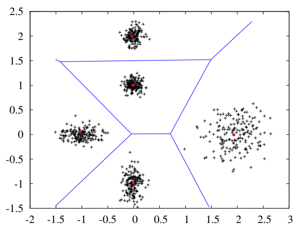
Definition 6. *A graph is stable if the MDL criterion is not reduced after* T_m *learning epochs.*

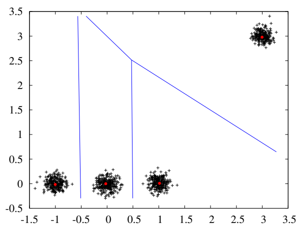
Algorithm 1: RobustGNG($\mathcal{D}, \mathcal{G}$)

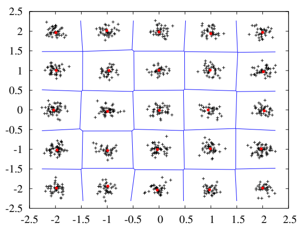
```
Data: A data set $\mathcal{D}$ and a set of 2 not connected nodes $\{c_1, c_2\} \in A$, whose weights are
      initialized randomly considering the data set bounds. Set the constants defined in
      Sections 2.1.1 and 2.1.3. Initialize restricting distances as explained in Eq. 7. Initialize
      smoothed mean error rates with the highest distance among the data set.
Result: A graph $\mathcal{G}$ which is stable
begin
    while Graph $\mathcal{G}$ is not stable do
        for $t = 1$ to $N$ do
            Randomly draw a vector $\mathbf{x}(t) \in \mathcal{D}$;
            Determine winner $s_1$ and second winner $s_2$, where $s_1 = \arg\min_{i \subset A} \|\mathbf{x}(t) - \mathbf{w}_i(t)\|$
            and $s_2 = \arg\min_{i \subset A \setminus \{s_1\}} \|\mathbf{x}(t) - \mathbf{w}_i(t)\|$;
            Update the weights of nodes $k \in N_{s_1} \cup \{s_1\}$ by using Eq. 8;
            if $t \mod \lambda = 0$ where $\lambda$ is a time window constant then
                Calculate MDL using Eqs. 13,14;
                if current MDL is minimal then
                    Store current graph $\mathcal{G}$ as the graph with minimal MDL $\mathcal{G}_{\min}$;
                if current graph is stable then
                    Store $\mathcal{G}_{\min}$ as the resulting stable graph;
                    Finish the algorithm here;
                if insertion requirement is fulfilled (Sec. 2.1.2) then
                    If $M > 2$, delete any nodes if necessary according to Section 2.1.4. If a
                    node is deleted, initialize restricting distances $d_k$ and recalculate MDL.
                    Check if MDL is minimal and store it accordingly. Check if the graph is
                    stable, in which case the algorithm finishes;
                    repeat
                        Determine nodes $q$ and $f$ with maximal insertion criterions $K_q$ and
                        $K_f$ according to Eq. 10;
                        Insert a new prototype $r$ and set its reference vector as in Eq. 12.
                        Interpolate its initial smoothed mean error from $q$ and $f$:
                        $e_r = \frac{1}{2}(e_q + e_f)$;
                        Insert edges connecting the new prototype $r$ with prototypes $q$ and
                        $f$, removing the original one: $\mathcal{C} = \mathcal{C} \cup \{(r, q), (r, f)\}$, $\mathcal{C} = \mathcal{C} \setminus \{(q, f)\}$;
                        Initialize restricting distances as explained in Eq. 7;
                    until only once if a dislocated node in the previous growth stage was
                    deleted ;
            Update the smoothed harmonic mean error rates according to Eq. 2;
            Update the restricting distances $d_k$ as in Eq. 6;
            if a connection between $s_1$ and $s_2$ does not exist already and $s_1$ or $s_2$ has not been
            deleted then
                Create connection: $\mathcal{C} = \mathcal{C} \cup \{s_1, s_2\}$;
                Set the age of the connection $\{s_1, s_2\}$ to 0: $\text{age}_{\{s_1,s_2\}} = 0$;
                Increment the age of all edges emanating from $s_1$:
                $\text{age}_{\{s_1,i\}} = \text{age}_{\{s_1,i\}} + 1, \forall i \in N_{s_1}$;
            Remove edges with age values greater than a constant $\alpha$;
            Remove all nodes without any edge and in such case initialize restricting distances;
    return $\mathcal{G}$
end
```

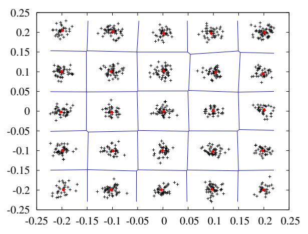
3. Experimental results in Vector Quantization

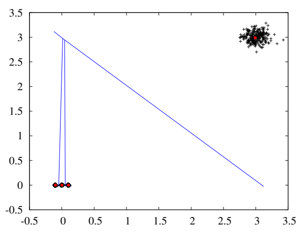
We designed several synthetic probability distributions in $\mathbb{R}^2$ with a variety of topological properties. We also used similar ones to those described in [7, 8]. We added white noise to test the algorithm in the presence of outliers. In Figure 2, different clean data sets are used to illustrate the results. The nodes are shown as red circles and the Voronoi regions they form are also illustrated. The variances of the Gaussian distributions are also variable, having values ranging from 0.01 to 0.3.

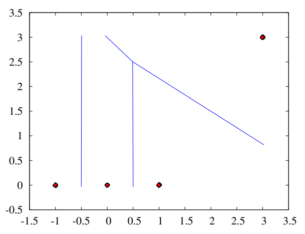

(a) Distribution 1

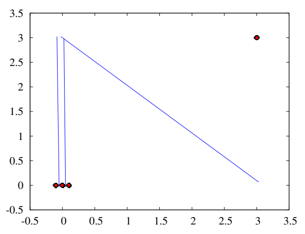

(b) Distribution 2

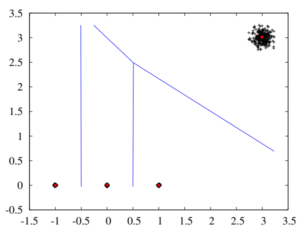

(c) Distribution 3


(d) Distribution 4


(e) Distribution 5


(f) Distribution 6


(g) Distribution 7


(h) Distribution 8

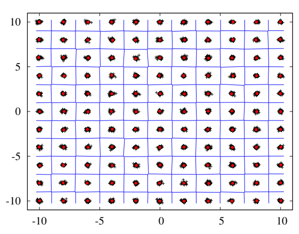
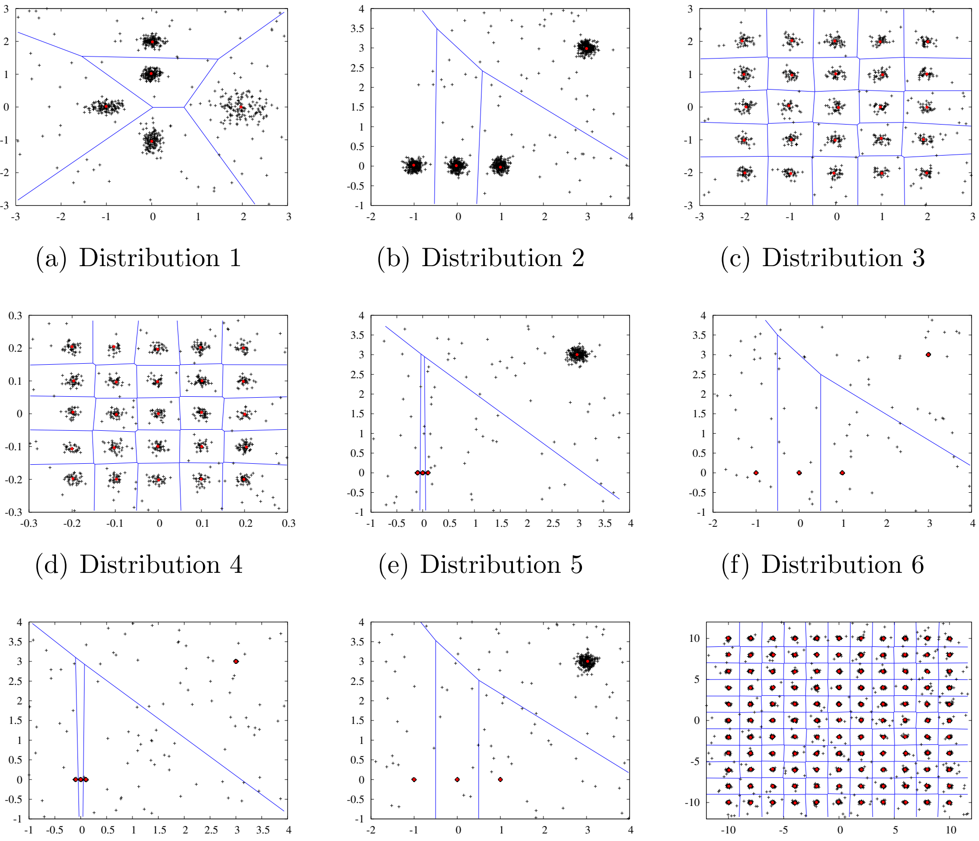

(i) Distribution 9

Figure 2: Synthetic data sets used for testing vector quantization.

The distribution 1 (Fig. 2(a)) consists of 5 Gaussian distributions centered at $[0, 2], [0, 1], [2, 0], [-1, 0]$ and $[0, -1]$ with variances of $[0.1, 0.1], [0.1, 0.1],$

$[0.3, 0.3], [0.2, 0.1]$ and $[0.1, 0.2]$ respectively. The distributions 2, 5, 6, 7 and 8 (Figs. 2(b),2(e), 2(f), 2(g) and 2(h)) have two different variances ($[0.1, 0.1]$ and $[0.01, 0.01]$ respectively). The distributions 3 and 9 (Figs. 2(c) and 2(i)) have a variance of $[0.1, 0.1]$ for every component and the distribution 4 (Fig. 2(d)) a variance of $[0.01, 0.01]$.

We also generate data sets as described above, adding outliers generated by uniformly distributed white noise, as shown in Figure 3.



(a) Distribution 1 (b) Distribution 2 (c) Distribution 3

(d) Distribution 4 (e) Distribution 5 (f) Distribution 6

(g) Distribution 7 (h) Distribution 8 (i) Distribution 9

Figure 3: Synthetic data sets with outliers.

We run 10 experiments for every distribution for evaluation. The parameters we used for all the experiments were $\eta_{s_1} = 0.1$, $\eta_{i \in N_{s_1}} = 0.001$, $T_e = 5$, $T_m = 400$, $\alpha = 50$, $\tau = 60$, $\theta = 100$. λ (the time window for insertions and deletions constant) usually equals the size of the data set, except for the distribution 9 where $\lambda = 1000$. For distributions 2 - 8 we used the parameter $\kappa = 1.3$. For the distributions 1 and 9 we used $\kappa = 1$.

In Table 1 we demonstrate the results of running the experiments on clean distributions. The column "Found" illustrates the number of clusters found when the algorithm failed to find the optimal number of clusters. For every distribution we generated a distinct data set each time (generating 90 data sets in total). In Table 2 we show the corresponding results for noisy data

Table 1: Results for clean data sets.

Distribution	Samples	Successful trials	Clusters	Found
1	800	10	5	
2	1000	10	4	
3	1000	10	25	
4	1000	10	25	
5	1000	10	4	
6	1000	10	4	
7	1000	10	4	
8	1000	10	4	
9	5000	7	121	122

sets. Here an outlier is added to the data set with 9% probability, except for the distribution 9 (5% probability). Distributions 1, 3 and 9 are similar to the ones presented in [7, 8]. Applying the algorithm to these distributions showed comparable and practically identical results, in terms of finding the right number of clusters. In Figure 4, we depict the Mean Absolute Error (MAE)

Table 2: Results for noisy data sets.

Distribution	Samples	Successful trials	Clusters	Found
1	800	10	5	
2	1000	10	4	
3	1000	9	25	26
4	1000	10	25	
5	1000	9	4	2
6	1000	10	4	
7	1000	10	4	
8	1000	10	4	
9	5000	10	121	

and corresponding standard deviation for the 20 trials in each distribution.

In Figure 5 we illustrate the evolution of MDL values. We run an experiment with the Distribution 3 with both clean and noisy data sets. After the

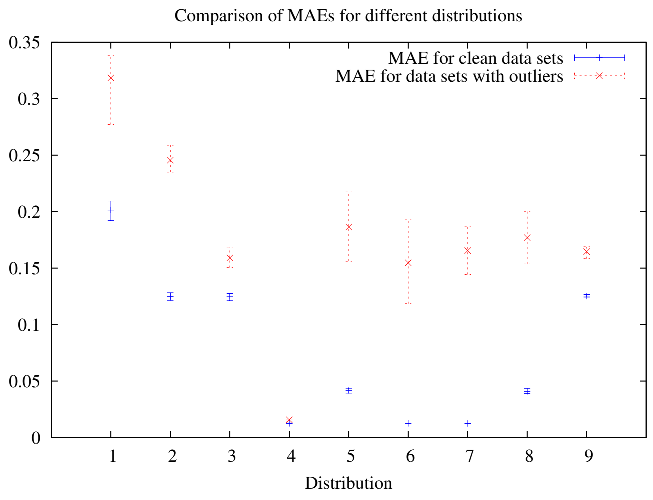


Figure 4: Mean Absolute Error for different clean and noisy synthetic distributions.

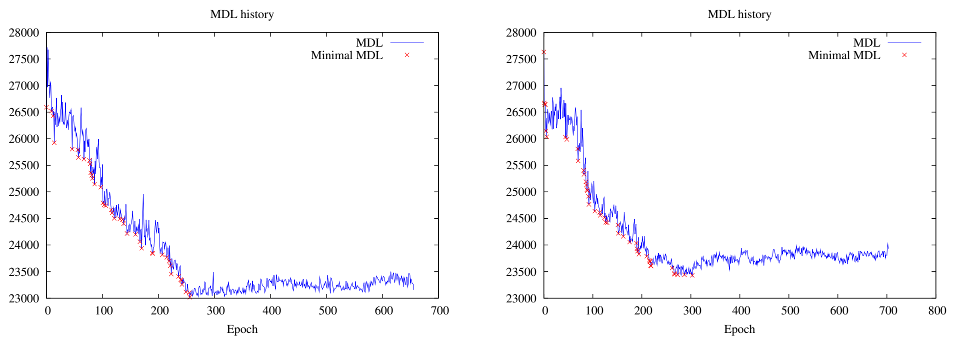
optimal number of clusters is found, MDL does not decrease and eventually, after T_m epochs, the algorithm stops.

The quantization results show a good reliability of the algorithm, taking into account a variety of topological properties of the distributions and in the presence of outliers. The reduction of parameters for the algorithm is an additional gain, compared to previous GNG implementations. Likewise, an online calculation of the error and a learning rate based on this quantity makes the algorithm suitable to be subsequently implemented as an online learning algorithm (cf. discussion in Section 6). The modification of the weight calculation for a new inserted node is useful to find proper quantizations for the distributions 5, 6 and 8. The results also show that the procedure is invariant to different scale parameters (variances), so that it is not necessary to normalize data for the cases presented here.

4. Induction of Substochastic Sequential Machines

4.1. Substochastic Sequential Machines

Definition 7. *A substochastic sequential machine (SSM) is a quadruple* $\langle Q, X, Y, \mathcal{P} = \{p(q_j, y_l|q_i, x_k)\}\rangle$ *where* Q *is a finite set of state elements (SEs),* X *is a finite set of input symbols,* Y *is a finite set of output symbols, and* $\mathcal{P}$ *is a finite set of conditional probabilities (cf. explanation in [2] and eqs.16-18) where* $q_i, q_j \in Q, x_k \in X$ *and* $y_l \in Y$*.*



(a) Clean data set (b) Noisy data set

Figure 5: MDL history for distribution 2.

We use the vector quantization method described in Section 2 for state discretization. A SSM models a situated discrete time dynamical system (SDTDS) for which input (I), output (O) and state (S) spaces are defined, and a transition function $\gamma : S \times I \rightarrow S \times O$. A stochastic dynamical model of such a system is a joint probability mass function p_Ω induced from a transition event set Ω, and quantizer functions Λ_o, Λ_i and Λ_s for output, input and state spaces respectively. Ω consists of selected transition events recorded from a given set of input sequences. Thus, the joint probabilities of observed and quantized transitions (p_Ω) are translated into joint probabilities of SSM transitions according to $\mathcal{P}$. As already mentioned, we define $\Lambda_i(\mathbf{i}(t))$ and $\Lambda_o(\mathbf{o}(t))$ according to the discretization described in Section 2, and $\Lambda_s(\mathbf{s}(t))$ using a modified version of the original state space quantization method (Crystalline Vector Quantizer - CVQ) explained in [2]. Thus, we have:

$$\begin{gathered} p(q_i, x_k, y_l, q_j) = \\ p_\Omega(\Lambda_s(\mathbf{s}(t)) = i, \Lambda_i(\mathbf{i}(t)) = k, \Lambda_o(\mathbf{o}(t)) = l, \Lambda_s(\mathbf{s}(t+1)) = j) \end{gathered} \tag{16}$$

The conditional probability is calculated with:

$$p(q_i, x_k) = \sum_{j=1}^{|Q|} \sum_{l=1}^{|Y|} p(q_i, x_k, y_l, q_j) \tag{17}$$

$$p(q_j, y_l | q_i, x_k) = \begin{cases} \frac{p(q_i, x_k, y_l, q_j)}{p(q_i, x_k)} & \text{if } p(q_i, x_k) > 0 \\ 0 & \text{if } p(q_i, x_k) = 0 \end{cases} \tag{18}$$

Definition 8. *The translation procedure from Ω to an SDTDS and then into an SSM will be called create_machine*$(\Omega, \Lambda_s, \Lambda_i, \Lambda_o)$.

The substochasticity of the extracted machines is due to the possibility that the sample of input sequences in Ω will not necessarily provide examples of all possible input symbols in all possible enumerations of the quantized space of the dynamical system. As a consequence, the probability distributions can become substochastic [2]. The details of the procedure for extracting substochastic sequential machines is described in [2]. In summary, there is a recursive state splitting, starting from only one SE. Then, a decision to split data into different SEs is based primarily on the maximal output entropy $\arg\max H(Y|Q = q_i, X = x_k) = H(\mathcal{P}_y(q_i, x_k))$ and then on the maximal next state entropy $\arg\max H(Q|Q = q_i, X = x_k) = H(\mathcal{P}_q(q_i, x_k))$. This yields that state vectors that convey the most information (i.e., highly indeterministic) are used for splitting [2]. Here, $H(\mathcal{P}) = -\sum_{i=1}^{n} p_i \log p_i$ and $p(q(t+1)) = \mathcal{P}_q(q_i, x_k)$ and $p(y(t)) = \mathcal{P}_y(q_i, x_k)$ are marginal distributions of $\mathcal{P}$. Each split node has associated model vectors that point to other split states, merged ones, or leaf nodes. The model vectors are calculated from the average of the vectors which they represent. Additionally, states are possibly merged if there exists an equivalence relation between two states based on determining when two SEs are not equivalent if they, in their outgoing transitions, share some input symbols and transitions that lead to discrepancies in the future output. The procedure finishes when the machine is deterministic, i.e., when the entropies for all states equal to 0.

4.2. An improved CVQ

As mentioned above, the quantization procedure for the state space is based on the CVQ quantizer. This method has similarities with hierarchical decision trees [2]. In this work, we redefined the quantization procedure of a CVQ, in order to solve issues regarding symmetrical properties present in the evolution of some dynamical systems (e.g. cf. Fig. 8(a) and see Fig. 6 discussed below). A CVQ is defined as follows [2]:

Definition 9. *A CVQ Graph is a quadruple* $CVQ = \langle N_{leaf}, N_{VQ}, N_{Merged}, n_{root} \rangle$ *where* n_{root} *is the root node of the CVQ Graph and the constituents are defined below.*

Definition 10. *A leaf node* $n \in N_{leaf}$ *has only one constituent,* $n = \langle i \rangle$, *where* $i \in \mathbb{N}$ *is an enumeration of the node within the CVQ and* $1 \leq i \leq |N_{leaf}|$.

Definition 11. *A Vector Quantizer (VQ) node $n \in N_{VQ}$ is a tuple $n = \langle \mathcal{M}, \mathcal{H} \rangle$ where $\mathcal{M}$ is a list of L model vectors $[\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_L]$, where $\mathbf{m}_i \in \mathbb{R}^d$ and $\mathcal{H}$ is a nonrepetitive list of child nodes $[h_1, h_2, \ldots, h_L]$ where $h_i \in N_{Leaf} \cup N_{VQ} \cup N_{Merged}$ and $d \in \mathbb{N}$ is the dimensionality of the vector space.*

Definition 12. *A merged node in a CVQ graph, $n \in N_{Merged}$, contains only a "link", $n = \langle n_{group} \rangle$, where $n_{group} \in N_{Leaf} \cup N_{VQ} \cup N_{Merged}$.*

The model vectors in a VQ-Node have associated a list $[\ell_1, \ell_2, \ldots, \ell_L]$ of classifications (labels). An element ℓ_i is a tuple $\langle y(t_i), y(t_i+1) \rangle$ which are the output symbols observed at some time step t_i and the subsequent $t_i + 1$. In this work, we redefine a CVQ quantizer function Λ_{cvq} in terms of a function winner : $N_{Leaf} \cup N_{VQ} \cup N_{Merged} \times \mathbb{R}^d \times \langle \mathbb{N}, \mathbb{N} \rangle \to \{1, 2, \ldots, M\}$:

$$\Lambda_{cvq}(\mathbf{s}(t)) = \text{winner}(n_{root}, \mathbf{s}(t), \langle y(t), y(t+1) \rangle), \tag{19}$$

which in turn is recursively defined as:

$$\text{winner}(n, \mathbf{s}(t), \langle y(t), y(t+1) \rangle) = \begin{cases} ID & \text{if } n \in N_{Leaf} \\ \text{winner}(n_{group}, \mathbf{s}(t), \langle y(t), y(t+1) \rangle) & \\ & \text{if } n \in N_{Merged} \\ \text{winner}(h_w, \mathbf{s}(t), \langle y(t), y(t+1) \rangle) & \\ & \text{if } n \in N_{VQ}, \end{cases} \tag{20}$$

where we determine w, the index of the winning child of a VQ-node according to:

$$w \;=\; \arg\min \|\mathbf{s}(t) - \mathbf{m}_i\|, \quad \text{s.t. } \ell_i = \langle y(t), y(t+1) \rangle. \tag{21}$$

The symmetry problem can be visualized in Fig. 6, where two model vectors with identical geometrical locations have different associated outputs. Thus, depending on the context, one of these model vectors have to be selected to eliminate the ambiguity.

We refer to [2] for more detailed explanations of CVQ training. When a CVQ leaf node is completely split [2], a recursive method is devised that splits data points in regions that separate them on the basis of labels for vectors as seen above. Model vectors are determined by averaging the data points in a region. Given a slightly different labeling present in this work, we reformulate the complete split in Def. 13. In Section 4.3 we describe the procedure for labeling vectors when splitting a data set.

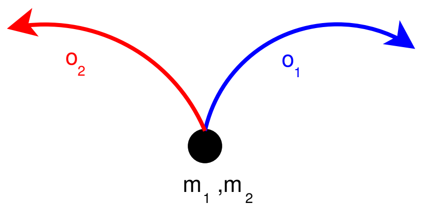


Figure 6: The symmetry issue in model vectors

Definition 13. *The complete split of several VQ nodes using several data sets at once is denoted $cvq = split\_cvq(cvq, D)$ where cvq is the CVQ to be split and $D = [D_1, D_2, \ldots, D_{|\Lambda_{cvq}|}]$ is a list of data sets where D_i is the data set for splitting the leaf node with $ID = i$ (if the node should not be split, then $D_i = \emptyset$). The elements of a data set are pairs $\langle \mathbf{s}, \ell \rangle$ where $\mathbf{s}$ is the data vector and $\ell \in \langle \mathbb{N}, \mathbb{N} \rangle$ is a label or class of the data vector. The leaf nodes are re-enumerated after the completion of all splits.*

4.3. CrySSMEx learning loop

The principal components of the CrySSMEx algorithm are listed below [2]:

- the SDTDS which represents the class of systems for CrySSMEx to analyse.
- the data set, i.e., the SDTDS transition event set Ω.
- SSMs, a subtype of SDTDSs.
- SDTDS transformation into SSM by quantizing input, output and state (cf. Sections 2 and 4.2).
- generation of UNDI-equivalence (universally not decisively inequivalent) sets in SSMs, which is the process that helps to determine when to merge states (function generate_UNDI_equivalence_sets).
- Use of CVQ as a state space quantizer (cf. Section 4.2).
- merging (function merge_cvq) and splitting of CVQ leaf nodes (cf. Definition 13).

- Selection and labeling of state vectors of Ω based on SSM information-theoretic properties (Algorithm 2).

Algorithm 2: collect_split_data($\Omega, \Lambda_i, \Lambda_s, \Lambda_o$)

```
Data: A transition event set, $\Omega$, an SSM, $ssm$, an input quantizer, $\Lambda_i$, a state
      quantizer, $\Lambda_s$ and an output quantizer, $\Lambda_o$.
Result: A list of data sets $D$, one data set per $q \in Q$. An element of each data set
        is described in Def. 13.
begin
    $D = [\emptyset, \emptyset, \dots, \emptyset]$;
    for $\forall \langle \mathbf{s}(t), \mathbf{i}(t), \mathbf{o}(t), \mathbf{s}(t+1) \in \Omega$ do
        $q_i = \Lambda_s(\mathbf{s}(t))$;
        $x_k = \Lambda_i(\mathbf{i}(t))$;
        $y_l = \Lambda_o(\mathbf{o}(t))$;
        $y_m = \Lambda_o(\mathbf{o}(t+1)$;
        $q_j = \Lambda_s(\mathbf{s}(t+1)$;
        if $\exists x_r : H_{ssm}(Y|Q = q_i, X = x_r) > 0$ then
            $x_{max} = \arg\max_{x_r \in X} H_{ssm}(Y|Q = q_i, X = x_r)$;
            if $x_k = x_{max}$ then
                $D_i = D_i \cup \langle \mathbf{s}(t), \langle y_l, y_m \rangle \rangle$;
        else if $\exists x_r : H_{ssm}(Q|Q = q_i, X = x_r) > 0$ then
            $x_{max} = \arg\max_{x_r \in X} H_{ssm}(Q|Q = q_i, X = x_r)$;
            if $x_k = x_{max}$ then
                $D_i = D_i \cup \langle \mathbf{s}(t), \langle y_l, y_m \rangle \rangle$;
    return $D$
end
```

In Algorithm 2, data sets are first split if output is indeterministic. If not, but the next state is not deterministic, data sets are then split and labeled accordingly. Finally, an improved version of the CrySSMEx main loop can be observed in Algorithm 3.

Algorithm 3: CrySSMEx($\Omega, \Lambda_i, \Lambda_o$)

```
Data: An SDTDS transition event set, $\Omega$, an input space quantizer $\Lambda_i$, an output
      space quantizer $\Lambda_o$.
Result: A deterministic SSM mimicking the SDTDS within the domain $\Omega$ as
        described by $\Lambda_o$.
begin
    $i = 0$;
    $ssm^0 = \text{create\_machine}(\Omega, \Lambda_i, \Lambda_{cvq^0}, \Lambda_o)$ ($ssm^0$ has $Q = \{q_1\}$ with all
    transitions to itself);
    repeat
        $i = i + 1$;
        $D = \text{collect\_split\_data}(\Omega, ssm^{i-1}, \Lambda_i, \Lambda_{cvq^{i-1}}, \Lambda_o)$;
        $cvq^i = \text{split\_cvq}(cvq^{i-1}, D)$;
        $ssm^i = \text{create\_machine}(\Omega, \Lambda_i, \Lambda_{cvq^i}, \Lambda_o)$;
        if $ssm^i$ has UNDI-equivalent states then
            $E = \text{generate\_UNDI\_equivalence\_sets}(ssm^i)$;
            $cvq^i = \text{merge\_cvq}(cvq^i, E)$;
            $ssm^i = \text{create\_machine}(\Omega, \Lambda_i, \Lambda_{cvq^i}, \Lambda_o)$;
    until $ssm^i$ is deterministic ;
    return $ssm^i$
end
```

5. Experimental Results of SSMs induction

To test the ability of the algorithm to infer probabilistic machines from noisy data we carried out experiments with noisy automata. Experiments with noisy automata were proposed by [13]. Some GNG-based algorithms adapted to learn time series have already been used to solve this problem [14]. In previous works, algorithms have focused only on prediction but not on models with optimal quantization and optimal rule construction from sequential data as we do here. The goal is to evaluate the density estimating capabilities of a temporal model by reconstructing the transition probabilities of a second order Markov model. Input vectors in $\mathbb{R}^2$ are generated from three Gaussian distributions with means $\mathbf{a} = [0, 0]$, $\mathbf{b} = [1, 0]$ and $\mathbf{c} = [0, 1]$ and common standard deviations σ. Figure 7 shows the automaton where transition probabilities are set depending on the parameter x.

We used the vector quantization algorithms to quantize the input spaces. In this case, the input space is derived from the probability distribution of the Gaussians. Thus, they generate three input clusters $\{a, b, c\}$ associated to means $\{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$. For different values of σ and $x = 0.1$, Figure 8 shows the corresponding quantization for 1000 points.

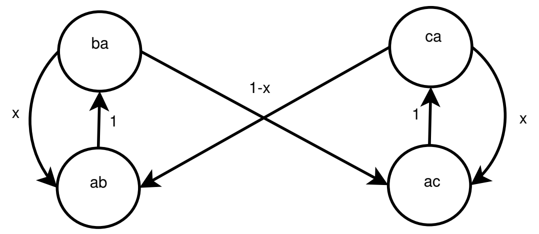
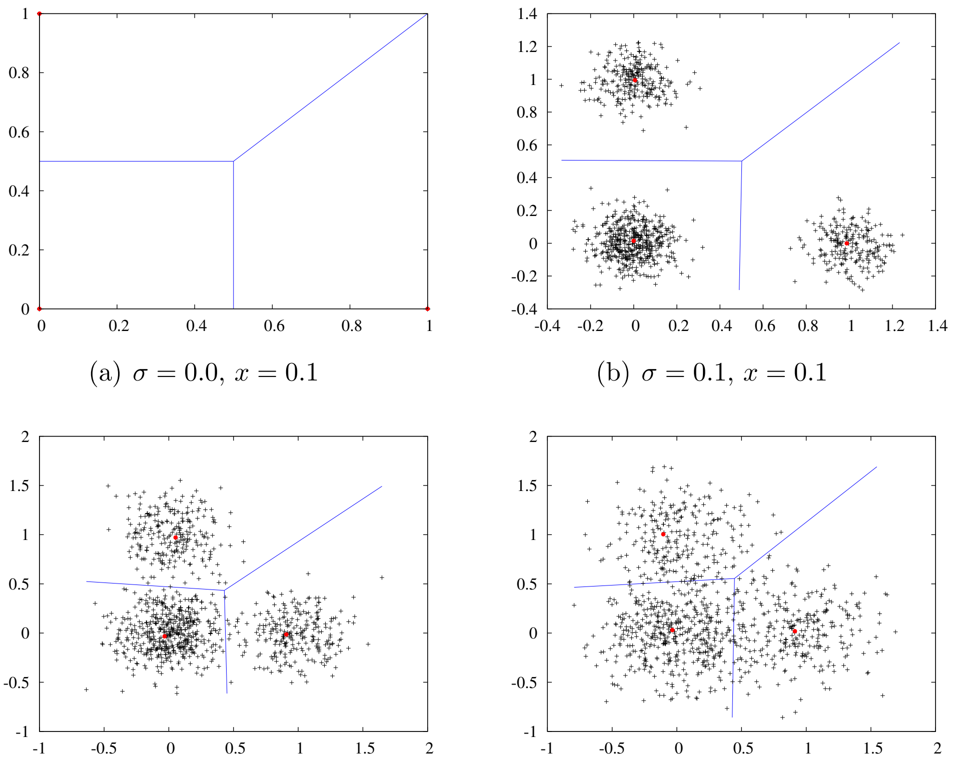


Figure 7: Transition probabilities of the noisy automaton.

Here, we carried out experiments with 10^3 data points with standard deviation $\sigma \in \{0.0, 0.1, 0.2, 0.3\}$ and transition probabilities $x \in \{0.0, 0.1, 0.2, 0.3\}$. In this experiment, an output vector space is derived from the transition between a data point $\mathbf{i}(t-1)$ and $\mathbf{i}(t)$. We define an output vector $\mathbf{o}(t) = \mathbf{i}(t) - \mathbf{i}(t-1)$. Thus, the quantization algorithm RobustGNG is also used for discretization as the output quantizer Λ_o. In this case, it is easy to verify that the output space is discretized in 4 components.

After the quantization of input and output spaces, we run the CrySSMEx algorithm. It is possible to induce either Moore-like SSMs or Mealy-like SSMs. Induced probabilistic Moore machines using the RobustGNG output quantization are visualized in Figure 9. The circles denote the states with corresponding output symbols. Here, an output symbol like ab denotes a transition where the input is a and next input is b. The boxes denote the transitions with corresponding input symbols and transition probabilities when applicable. For further splits the same transition probabilities are obtained but using more VQ-nodes. In Figure 10 the final CVQ tree for $\sigma = 0.0, x = 0.1$ is visualized with the corresponding SSM, where boxes denote VQ-Nodes, points Merge nodes and circles leaf nodes.

In Figure 9(a), we observe that x was found to be 0.0 in both transition cases. In Fig. 9(b), x is induced as 0.11 and 0.10, and in Fig. 9(c) to 0.23 and 0.21 respectively. Thus, transition probabilities were inferred with high accuracy. In Table 3 and 4 we present the reconstructed probabilities based on the induced probabilistic Moore machines. We show results for standard deviations $\sigma \in \{0.0, 0.1\}$. These results are comparable to the ones shown in [14]. Notice that the starting input symbol for all experiments was always a. For greater values of σ, the Gaussian distributions overlap, leading to more complex machines that are more difficult to analyse. In such cases,


(a) $\sigma = 0.0$, $x = 0.1$ (b) $\sigma = 0.1$, $x = 0.1$

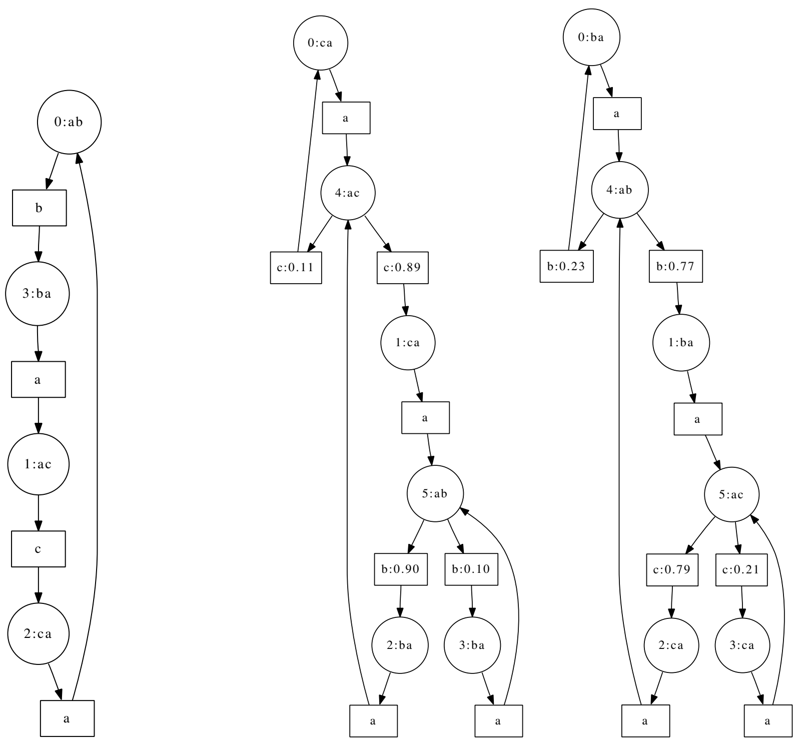
(c) $\sigma = 0.2$, $x = 0.1$ (d) $\sigma = 0.3$, $x = 0.1$

Figure 8: Input space quantization of noisy automata.

a point generated by some Gaussian distribution might be quantized as a member of other Gaussian distribution, since the margins that the Voronoi regions form are not sufficient to split points accordingly. In Fig. 11, the result for $\sigma = 0.2$ and $x = 0.0$ is illustrated.

6. Conclusions and Future Work

In this article, we present a method for extracting substochastic sequential machines from clean and noisy data sets representing dynamical systems. A Vector quantization was implemented in order to discretize the input and output spaces associated to a situated discrete time dynamical system. The method is shown to be successful for cluster analysis with Gaussian noise dis-



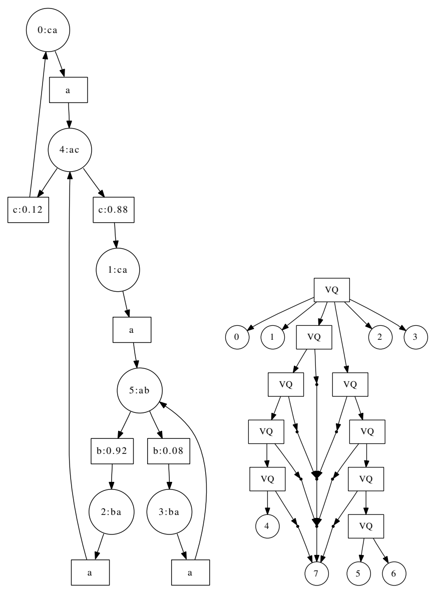
(a) $\sigma = 0.1$, $x = 0.0$ (b) $\sigma = 0.1$, $x = 0.1$ (c) $\sigma = 0.1$, $x = 0.2$

Figure 9: Substochastic Moore Machines extracted from noisy automata

tributions. This quantization method was incorporated in the main loop of the CrySSMEx algorithm, which is useful to extract discrete representations of dynamical systems in a probabilistic automata fashion called substochastic sequential machines. We also improved CrySSMEx to solve issues that involve ambiguities given some context in the evolution of a dynamical system.

The clustering (quantization) algorithm works well when clusters do not overlap. Thus, clusters must lie on a Voronoi region which is convex, given the use of Euclidean distances [9]. However, other metric functions or kernels might also be considered [15, 16]. An additional step is to apply the algorithm in "real" data sets.

Quantization is performed in order to discretize the input and output spaces of a dynamical system. The method presented here builds upon pre-



(a) SSM (b) CVQ Graph

Figure 10: CVQ Graph and corresponding SSM.

vious work on Growing Neural Gas algorithms, and subsequent improvements incorporating a Minimum Description Length criterion to decide the optimal number of clusters. In our method, we implemented an online calculation of error that aims for obtaining learning rates and node insertion and removal criteria. By using the CrySSMEx algorithm, transition probabilities and a state space discretization are obtained by applying a method that uses conditional entropy in terms of states and output functions. The method was tested successfully in an experiment with Noisy Automata, where corresponding substochastic sequential machines were inferred.

In future work, we will investigate proper stopping criteria for CrySSMEx. Techniques like Supervised Vector quantization or alternative methods for obtaining model vectors in the state space can also be explored. Regarding vector quantization, there is still space for improvement when dealing with the prototype coincidence problem. Thus, a more efficient repulsion mechanism or insertion and deletion criteria can be devised. Alternatively, nodes might be added and removed at higher rates as in [6], whereas a decision for deleting a node can be made on the basis of an MDL criterion. The current

Table 3: Reconstructed probabilities for $\sigma = 0.0$.

x	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
$P(a\|ba)$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$P(b\|ba)$	0.0	0.08	0.2	0.32	0.42	0.51	0.63	0.68	0.8	0.91	1.0
$P(c\|ba)$	1.0	0.92	0.8	0.68	0.58	0.49	0.37	0.32	0.2	0.09	0.0
$P(a\|ca)$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$P(b\|ca)$	1.0	0.88	0.8	0.68	0.58	0.52	0.39	0.29	0.18	0.12	0.0
$P(c\|ca)$	0.0	0.12	0.2	0.32	0.42	0.48	0.61	0.71	0.82	0.88	0.0

Table 4: Reconstructed probabilities for $\sigma = 0.1$.

x	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
$P(a\|ba)$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$P(b\|ba)$	0.0	0.1	0.23	0.29	0.39	0.5	0.59	0.68	0.85	0.89	1.0
$P(c\|ba)$	1.0	0.9	0.77	0.71	0.61	0.5	0.41	0.32	0.15	0.11	0.0
$P(a\|ca)$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$P(b\|ca)$	1.0	0.89	0.79	0.72	0.6	0.48	0.41	0.26	0.2	0.09	0.0
$P(c\|ca)$	0.0	0.11	0.21	0.28	0.4	0.52	0.59	0.74	0.8	0.91	0.0

state of the algorithm goes in the right direction for implementing an active and online learning method. First, by calculating online errors, learning rates and insertion criteria in the quantization procedure. Secondly, implementation of information-theoretic measures of predictability and model selection in RobustGNG and CrySSMEx. However, the MDL and restricting distances calculation, and the CrySSMEx learning loop have to be adjusted to accomplish this goal.

References

[1] R. Sun, Beyond simple rule extraction: The extraction of planning knowledge from reinforcement learners, in: Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00)-Volume 2 - Volume 2, IJCNN '00, IEEE Computer Society, Washington, DC, USA, 2000, pp. 105–110.

[2] H. Jacobsson, The crystallizing substochastic sequential machine extractor - `CrySSMEx`, Neural Computation 18 (2006) 2211–2255.

[3] S. L. Frank, H. Jacobsson, Sentence-processing in echo state networks:

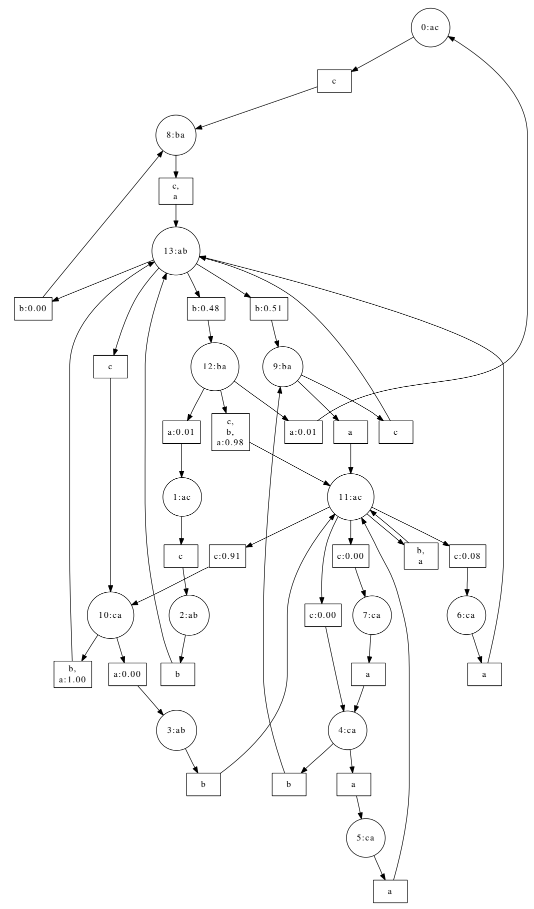


Figure 11: A complex automaton extracted from overlapping Gaussian distributions.

a qualitative analysis by finite state machine extraction, Connection Science 22 (2010) 135–155.

[4] S. Roa, G.-J. Kruijff, On the automatic entropy-based construction of probabilistic automata in a learning robotic scenario, in: Robotics: Science and Systems (RSS) 2010 Workshop: Towards Closing the Loop: Active Learning for Robotics.

[5] B. Fritzke, A growing neural gas network learns topologies, in: Advances in Neural Information Processing Systems 7, MIT Press, 1995, pp. 625–632.

[6] F. H. Hamker, Life-long learning cell structures–continuously learning without catastrophic interference, Neural Networks 14 (2001) 551–573.

[7] A. Qin, P. Suganthan, Robust growing neural gas algorithm with application in cluster analysis, Neural Networks 17 (2004) 1135 – 1148. New Developments in Self-Organizing Systems.

[8] K. Qin, N. Suganthan, Enhanced neural gas network for prototype-based clustering, Pattern Recognition 38 (2005) 1275–1288.

[9] H. Graf, Siegfried ; Luschgy, Foundations of quantization for probability distributions, Lecture notes in mathematics ; 1730, Springer, Berlin, 2000.

[10] A. K. Jain, M. N. Murty, P. J. Flynn, Data clustering: a review, ACM Computing Surveys 31 (1999) 264–323.

[11] C. M. Bishop, Pattern recognition and machine learning, Information science and statistics, Springer, New York [u.a.], 5. (corr. print.) edition, 2007.

[12] P.-Y. Oudeyer, F. Kaplan, V. V. Hafner, Intrinsic motivation systems for autonomous mental development, IEEE Transactions on Evolutionary Computation 11 (2007).

[13] M. Strickert, B. Hammer, S. Blohm, Unsupervised recursive sequence processing, Neurocomputing 63 (2005) 69–97.

[14] A. Andreakis, N. V. Hoyningen-Huene, M. Beetz, Incremental unsupervised time series analysis using merge growing neural gas, in: Proceedings of the 7th International Workshop on Advances in Self-Organizing Maps, WSOM '09, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 10–18.

[15] A. Qin, P. Suganthan, A novel kernel prototype-based learning algorithm, in: Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on, volume 4, pp. 621 – 624 Vol.4.

[16] F.-M. Schleif, A. Gisbrecht, B. Hammer, Accelerating kernel neural gas, in: T. Honkela, W. Duch, M. A. Girolami, S. Kaski (Eds.), ICANN (1), volume 6791 of *Lecture Notes in Computer Science*, Springer, 2011, pp. 150–158.

Sergio Roa works as Researcher in the German Research Center for Artificial Intelligence. He is also PhD candidate in Computer Science at Saarland University in Germany. He holds a M.Sc. degree in Computer Science from the University of Freiburg and an Engineering Diploma in Computer Systems Engineering from the National University of Colombia. His main areas of interest are Artificial Intelligence, Machine Learning and Pattern Recognition, Natural Language Processing and Cognitive Robotics. He has published articles related to robotic learning and natural language processing in different conferences related to artificial intelligence, machine learning and robotics such as RSS, EpiRob, ROBIO, FLAIRS, ACL, ECAI, ICASSP.

Dr. Geert-Jan Kruijff graduated as an engineer of the University of Twente combining Computer Science and analytical Philosophy. Because of his interest in computational linguistics and dialogue systems, he ended up in Prague doing a PhD in mathematical linguistics, finishing in 2001. While in Prague, he also spent two years in Edinburgh, doing a lot of work on categorial grammar and description logic-like semantics. From 2001 until 2004 Geert-Jan worked at the Department of Computational Linguistics at Saarland University, working on grammar formalisms and building LEGO robots. In late 2004 he turned building talking robots into a full-time job at DFKI GmbH, leading several international projects. Their Talking Robots group is now about 14 people strong, plus MSc students. Since early 2010 he is the coordinator for a large international project on HRI and urban search & rescue.

BHAM Technical Report
Provided as part of DR 2.5: Qualitative behaviour models, grasping of novel objects

Identification of qualitative states from the behaviour of objects

Sebastian Zurek, Marek Kopicki, and Jeremy Wyatt
School of Computer Science
University of Birmingham, UK

May 30, 2012

Abstract

For a robotic agent interacting with its environment, it is natural to represent its sensory input and motor output as continuous state spaces. This poses a challenge for controlling the behaviour of a robot, since at almost every instant it will observe a novel situation and will have an infinite choice of motor commands that it could deploy. An objective for robotics research is to devise algorithms that can extract qualitative states, in accord with human judgement. We present an algorithm that uses the behaviour of an object, when manipulated and observed by a robot, to discover the qualitative states in perception-action space. Thus we take the definition of a qualitative state to be a set of points in state space that behave similarly under a given action. The algorithm is evaluated by using data from a simulation of a robotic finger pushing an object.

1 Introduction

A robot interacting with its environment will collect a stream of sensory input and perform some actions. Typically the sensory input and motor output will be represented most naturally as continuous state spaces, possibly changing in continuous time. This poses a challenge for controlling the behaviour of a robot, since at almost every instant it will observe a novel situation and will have an infinite choice of motor commands that it could deploy. Clearly, it would be advantageous to reduce the size of the state space, by projecting to a smaller subspace, or by discretising. Indeed, human perception often chunks sensory experience in this manner.

An objective for robotics research is to devise algorithms that can extract these qualitative states, in accord with human judgement, given a continuous-state representation of a robot's perception-action space. However, it is difficult to define precisely a qualitative state in general, since one can argue that there must be a dependence on the task to be performed by the agent.

In this paper, we present an algorithm that uses the behaviour of an object, when manipulated and observed by a robot, to discover the qualitative states in perception-action space. Thus we take the definition of a qualitative state to be a set of points in state space that behave similarly under a given action.

The remainder of the paper is laid out as follows. In the next section, we explore briefly some previous work on identifying qualitative states in systems where a robotic agent interacts with an environment. Section 3 describes the clustering algorithm that we used to detect qualitative states from a vector field description of the dynamics of a system. Some preliminary results from robotic simulation experiments are reported in section 4. We conclude with a discussion in section 5.

2 Related Work

Oates et al. [1] considered the problem of extracting qualitative states from multivariate time series segments obtained directly from robotic sensors. By using Dynamic Time Warping (DTW) to induce a distance metric between pairs of time series, they were able to extract clusters with a standard hierarchical agglomerative clustering algorithm. They compared these clusters with a classification of the robotic scenarios by human observers, and found good agreement. However, some residual errors proved hard to eliminate, possibly due to the clustering algorithm and the use of DTW which distorted temporal information.

King and Oates [2] used the same techniques (DTW and hierarchical agglomerative clustering) to establish an alphabet that tokenised fixed-length subsequences of the sensory input time series. They proceeded to estimate the conditional probability of a particular sensory experience conditioned on a given action state, and thus were able to link actions with corresponding sensory outcomes (assuming stationarity in the input time series). This method also provided a set of primitive actions of the robot.

Another approach is to start from a symbolic representation of the sensory space. For example, Cohen et al. [3] used the Bayesian Clustering by Dynamics (BCD) algorithm to cluster sequences of sensory tokens. However, the scheme was only applied to univariate time series and had limited success finding a classification that agreed with human judgement.

QLAP (Qualitative Learner of Action and Perception) [4] is an unsupervised learning algorithm that learns states and hierarchical actions in continuous perception-action state spaces. It discretises the state and action space by expressing the values of continuous variables relative to known landmarks (Landmarks are distinguished points in the state space, that can be predefined or learned.)

We can contrast these learning-based approaches with that of Stahovich et al. [5] where the configuration space ("c-space") of a mechanical system is abstracted in a prescribed manner to produce a qualitative c-space ("qc-space"). Qc-space considers those properties of a c-space that must be preserved to ensure that behaviour across systems is qualitatively equivalent.

Finally, of some relevance to our work, the "push-stability diagram," introduced by Brost [6, 7], relates to the problem of grasping polygonally-shaped objects with a parallel-jaw gripper in a 2D setting. The push-stability diagram can be used to plan a stable grasp, and determine the final orientation of the object from any given starting configuration. It illustrates how qualitative states can be identified in a continuous perception-action space.

3 Algorithm to extract qualitative states

Let $\mathcal{S} \subseteq \mathbb{R}^n$ be an n-dimensional state space that describes the joint state of an object and a robotic manipulator (such as a finger or pusher). For a given class of robot actions $\mathcal{A}$, the manipulator can interact with the object resulting in some state change $x \rightarrow x'$. Hence, for each point $x \in \mathcal{S}$ we can associate a vector field $g_a(x)$ such that, for a given action $a \in \mathcal{A}$,

$$g_a(x) \mapsto x' \tag{1}$$

By exploration and observation using, for example, a visual tracking system, the robot can sample $g_a(x)$ at a set of states $\{x_i\}$ and obtain a discretised vector field $\{g_a(x_i)\}$. For concreteness, we will focus on a two-dimensional state space, such as Brost's push-stability diagram [7], in which the push direction δ and object orientation ϕ are the co-ordinates, i.e. $x =< \delta, \phi >$, and where the flow field $\{g(\delta_i, \phi_j)\}$ is available over a regular 2D grid $< \delta_i, \phi_j >$. We can partition the two-dimensional state space into regions using a clustering algorithm that groups neighbouring nodes according to two heuristics:

H1. qualitatively similar behaviour, e.g. $g(\delta, \phi)$ has the same sign, and

H2. smoothness in $g(\delta, \phi)$ – neighbours in state space behave similarly.

Explicitly we assign two nodes (i, j) and (i', j') to the same cluster if

1. they are nearest neighbours: $(|i - i'|, |j - j'|) = (0, 1)$ or $(1, 0)$
2. $\text{sgn}(g_\delta(\delta_i, \phi_j)) = \text{sgn}(g_\delta(\delta_{i'}, \phi_{j'}))$
3. $\text{sgn}(g_\phi(\delta_i, \phi_j)) = \text{sgn}(g_\phi(\delta_{i'}, \phi_{j'}))$
4. $\left|g_\delta(\delta_i, \phi_j) - g_\delta(\delta_{i'}, \phi_{j'})\right| < \theta_1$
5. $\left|g_\phi(\delta_i, \phi_j) - g_\phi(\delta_{i'}, \phi_{j'})\right| < \theta_2$

for some smoothness parameters θ_1 and θ_2, and where $g_\delta(\cdot)$ and $g_\phi(\cdot)$ are the components of the vector field. To compute the clusters, we start at an arbitrary seed node that is unassigned and grow a cluster by extending along those nearest neighbours that satisfy the above constraints. We repeat the process until each node on the grid is assigned to a cluster. Note that we can relax the first heuristic by using a modified sign function, namely:

$$sgn_\epsilon(x) = \begin{cases} 1, & x > \epsilon \\ -1, & x < -\epsilon \\ 0 & otherwise \end{cases} \tag{2}$$

This formulation allows for some noise in the observations, since there is not an abrupt transition between -1 and $+1$, as x crosses zero.

The next section considers a particular scenario, consisting of a robotic finger interacting with an object, that demonstrates how the clustering algorithm performs in a 2D state space.

4 Experiment

As an application of the algorithm described in the previous section, we conducted an experiment in which a robotic finger executed small pushes against an L–shaped object (known as a polyflap). Before each push, the polyflap was at rest, lying with one side flat on a table top. It is assumed that during the push, the polyflap slid along the table, with its base remaining flat on the table surface. Although data from a real robotics study could be used, in this paper, the experimental data was obtained by simulation, using the NVIDIA PhysX physics engine [8]. For a given push, let h be the height of the (centre of the) robotic finger, and x_r be the horizontal displacement of initial contact point from the vertical midline of the polyflap (see Figure 1). If the finger is not in contact with the polyflap at the end of a push, then we use the point nearest to the centre of the robotic finger that lies in the plane defined by the upright plate.

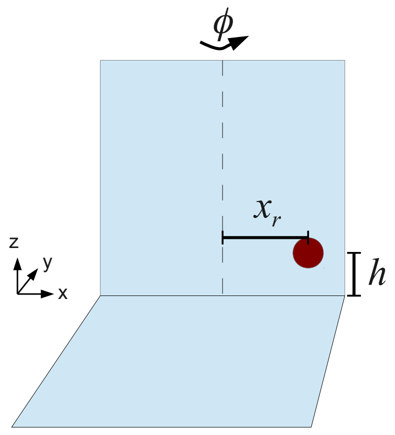


Figure 1: Definition of push parameters h and x_r, and yaw angle ϕ. The tip of the robotic finger is depicted as a red sphere, which is just touching the upright plate of the polyflap. A dashed line marks the vertical midline of the polyflap. The diagram shows the polyflap oriented at $\phi = 180°$.

Throughout the experiment, the push height h and push length are kept constant, but we vary x_r and the initial yaw angle ϕ, which specifies the orientation of the polyflap. A yaw angle ϕ of 180° corresponds to the polyflap's upright plate normal to the y-axis, and its base in front of the upright plate and underneath the robotic finger at the start of a push.

4.1 Representation of state space

Since the polyflap is taken to remain flat on the table, its state can be represented by three parameters: the x and y co-ordinates of its centre (x_p, y_p) and its orientation ϕ about the z-axis. We shall approximate the end of the robotic finger as a point, so that the location of this point (x_f, y_f, z_f) is sufficient to describe the finger state. A push action of the robotic finger can be characterised by push length L and push angle δ. Thus the full state space is 8-dimensional, which is too large to sample densely by real-world exploration or even simulation.

Brost's push-stability diagram (PSD) [6, 7] abstracts over the object's position and the location of the robotic effector – the push effector is idealised as having infinite width. By analogy with Brost, we will focus on the orientation ϕ of the polyflap and ignore (x_p, y_p). However, we are not able to dispense fully with the location of the robotic finger, since unlike Brost's case, the finger has a finite extent and the relative position of finger and polyflap is important. Given the finger is in contact with the polyflap, the parameters (x_f, y_f) can be replaced by a single co-ordinate relative to the position of the polyflap. Furthermore $z_f = h$, the push height, is kept constant, thus we replace (x_f, y_f, z_f) with one parameter x_r, as defined in Figure 1.

Finally by keeping the push length L small and fixed, we obtain the reduced state space:

$$< \delta, x_r, \phi > \tag{3}$$

where the first two co-ordinates determine the push action.

4.2 Results

We simulated pushes for a set of finger positions x_r and initial polyflap orientations ϕ that spanned a grid in $< x_r, \phi >$ space. (Note that the $\phi = 360°$ boundary should be identified with $\phi = 0°$.) Three data sets were computed for push angle $\delta = -45, 0, 45$ degrees, where $\delta = 0°$ defines a push parallel to the y-axis. The push height h and push length L were kept small and fixed. A value of sliding friction was chosen to simulate moderate roughness to prevent a simple translation of the polyflap with little or no rotation.

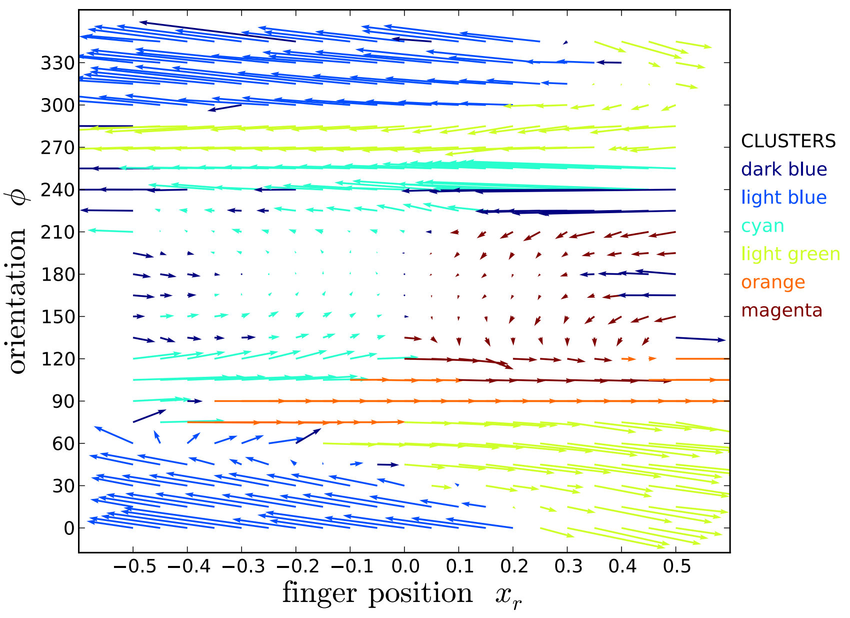


Figure 2: Vector field, for $\delta = 0°$. Five major clusters were identified.

The vector field $g(x_r, \phi)$ (cf. Equation 1) for $\delta = 0°$ is shown in Figure 2. Each arrow in this plot represents a particular push action: the tail of the arrow indicates the state at the start of the push, and the head of the arrow points to the final state at the end of the action.

The clustering algorithm identified five major clusters. Note that a number of small clusters (each with fewer than 20 elements) were detected, but on the plot they are all coloured dark blue. The modified sign function (Equation 2) was employed with $\epsilon_{x_r} = 100$ and $\epsilon_\phi = 0.5°$. The use of a large ϵ_{x_r} renders the value or x_r irrelevant when applying heuristic H1. Thus the x_r component of an arrow, $g_{x_r}(x_r, \phi)$, can change sign (subject to smoothness constraint H2) within a cluster, whereas the ϕ component must have the same sign (or zero up to ϵ_ϕ).

In the idealised case, the two sets of states at $\phi = 90°$ and $\phi = 270°$ are absorbing boundaries where an infinitesimally small finger pusher would not be able to change ϕ using just forward pushes ($\delta = 0°$). On the vector field plot, these boundaries would be depicted by lines of horizontal arrows. The clustering algorithm managed to isolate one cluster (coloured orange in Figure 2) around $\phi = 90°$, but the $\phi = 270°$ boundary is less clearly defined by the vector field and other clusters have merged into this region.

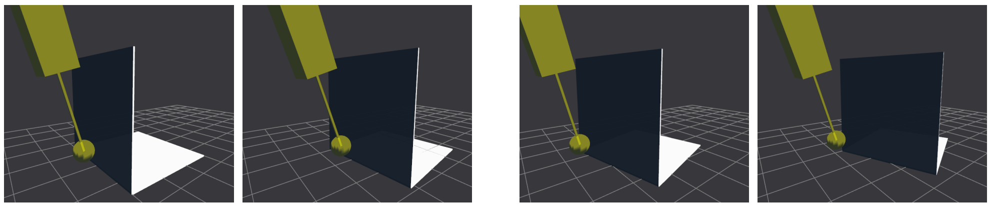


Figure 3: Sample trajectory from the large light-blue cluster in Figure 2. From left to right: the first two panels show the start and end configurations corresponding to the arrow at $(x_r, \phi) = (-0.05, 0)$. The last two panels show the start and end configurations corresponding to the arrow at $(-0.35, 15)$. (Note that the position of the polyflap is reset between the pairs of images (i.e. panel 2 and 3), creating an artefact in which the polyflap appears to move backwards.)

The algorithm detected 4 other qualitative states, that correspond roughly to the four combinations of rotation to the left ($\Delta\phi > 0$) or right ($\Delta\phi < 0$) about the z-axis and whether the base of the polyflap was in front ($\phi \in [90°, 270°]$) or behind ($\phi \in [0°, 90°] \cup [270°, 360°]$) the upright plate. For example, snapshots of a sample trajectory from the light-blue cluster (left rotation, base behind) are shown in Figure 3, whereas Figure 4 depicts a trajectory from the magenta cluster (right rotation, base in front).

Using the same ϵ and θ (smoothness) parameters used to generate the clusters in Figure 2, we applied the clustering algorithm to the case of oblique pushes, for $\delta = -45°$, and $\delta = 45°$. The results shown in Figure 5 demonstrate the robustness of the algorithm. Again four large clusters are identified corresponding to left/right rotation and initial orientation of the polyflap. However, there are also two further clusters where the polyflap does not rotate.

5 Discussion

In this paper we considered a system in which a robot interacts with an object. We proposed a clustering algorithm to extract the qualitative states of such a system, starting from a description of the dynamics of the system as transitions in a state space. For the specific case of a robotic finger pushing a polyflap object, we demonstrated that the algorithm enumerated states that accorded with human judgement.

In principle the algorithm could cluster in state spaces with more than $d = 2$ dimensions. However, larger state spaces (say $d = 4$) would be hard to sample at a reasonable density and it is not clear that the algorithm would perform

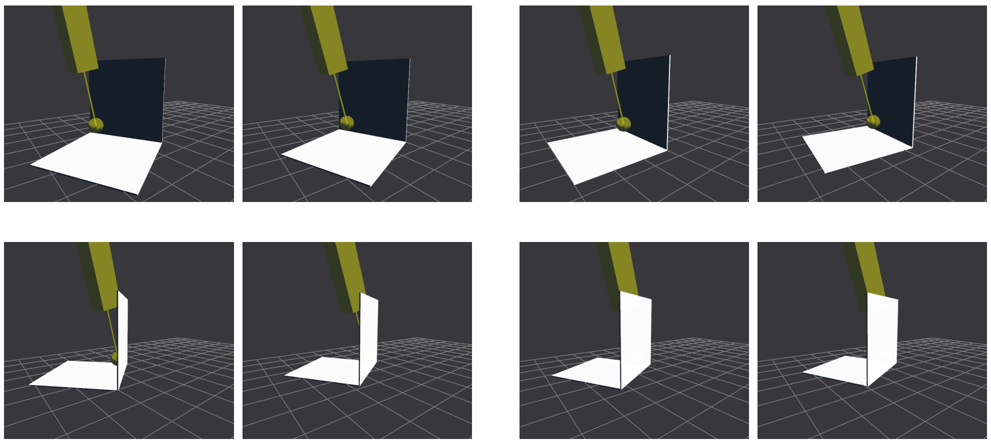


Figure 4: Sample trajectory from the large magenta cluster in Figure 2. The first two panels in the top-left quadrant show the start and end configurations corresponding to the arrow at $(x_r, \phi) = (0.35, 210)$. Successive pairs of panels (top-right, bottom-left, bottom-right) show start and end configurations corresponding to arrows along a flow that ends near $(x_r, \phi) = (0.33, 118)$. (As in Figure 3, the position of the polyflap is reset between each pair of images, creating an artefact in which the polyflap appears to move backwards.)

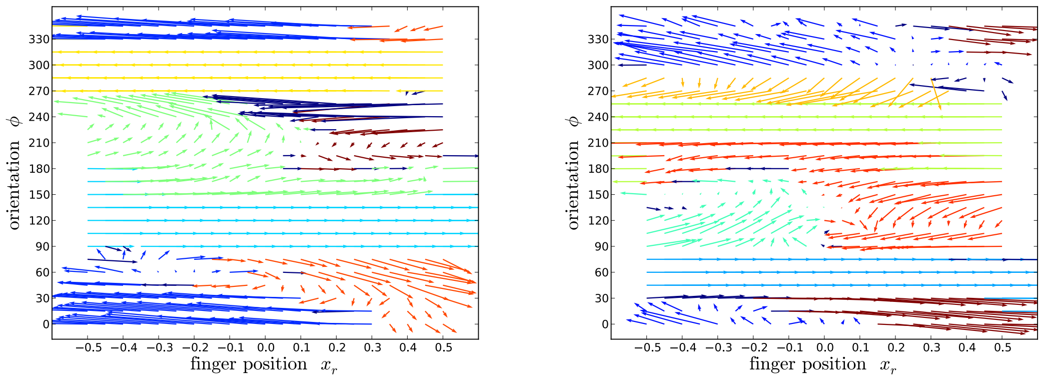


Figure 5: Vector fields for oblique pushes: left panel shows $\delta = -45°$, and right panel $\delta = 45°$.

satisfactorily in this case. Thus one would need to augment the algorithm with a dimensionality reduction strategy.

The algorithm is also loosely specified in the sense that it relies on heuristics whose precise formulation depend on the state space of the system. For example, in the experiment reported in section 4, we chose ϵ_{x_r} such that the x_r component of the state was irrelevant with respect to the qualitative state.

In future, we intend to probe the robustness of the scheme by applying it to data from real robotic experiments. It would also be interesting to investigate more complex environments, such as adding an obstacle to the workspace. The obstacle would alter the behaviour of the object, which would be reflected in a modified vector field of state transitions.

Acknowledgement

The work described in this report was funded by the European Commission's Seventh Framework Programme, contract no. ICT-215181-CogX.

References

[1] T. Oates, M. D. Schmill, and P. R. Cohen, "A method for clustering the experiences of a mobile robot that accords with human judgments," in *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence* (H. A. Kautz and B. W. Porter, eds.), pp. 846–851, AAAI Press / The MIT Press, 2000.

[2] G. King and T. Oates, "The importance of being discrete: Learning classes of actions and outcomes through interaction," in *Canadian Conference on AI* (E. Stroulia and S. Matwin, eds.), vol. 2056 of *Lecture Notes in Computer Science*, pp. 236–245, Springer, 2001.

[3] P. Cohen, M. Ramoni, P. Sebastiani, and J. Warwick, "Unsupervised clustering of robot activities: A Bayesian approach," Tech. Rep. 00-51, Computer Science Department, University of Massachusetts, 2000.

[4] J. Mugan and B. Kuipers, "Autonomous learning of high-level states and actions in continuous environments," *IEEE Trans. Autonomous Mental Development*, vol. 4, no. 1, pp. 70–86, 2012.

[5] T. F. Stahovich, R. Davis, and H. E. Shrobe, "Qualitative rigid-body mechanics," *Artificial Intelligence*, vol. 119, no. 1-2, pp. 19–60, 2000.

[6] R. C. Brost, "Planning robot grasping motions in the presence of uncertainty," Tech. Rep. CMU-RI-TR-85-12, The Robotics Institute, Carnegie-Mellon University, Pittsburgh, PA, July 1985.

[7] R. C. Brost, "Automatic grasp planning in the presence of uncertainty," *Int. J. Robotics Research*, vol. 7, no. 1, pp. 3–17, 1988.

[8] NVIDIA Corporation, PhysX, "Physics simulation for developers," 2009. http://developer.nvidia.com/physx.

Proceedings of the 2012 Computer Vision Winter Workshop (CVWW), 2012
Provided as part of DR 2.5: Qualitative behaviour models, grasping of novel objects

17th *Computer Vision Winter Workshop*
Matej Kristan, Rok Mandeljc, Luka Čehovin (eds.)
Mala Nedelja, Slovenia, February 1-3, 2012

Towards Scene Understanding – Object Segmentation Using RGBD-Images

Andreas Richtsfeld, Thomas Mörwald, Johann Prankl, Jonathan Balzer,
Michael Zillich and Markus Vincze

Vienna University of Technology
Gusshausstrae 25-29, 1040 Vienna

Abstract. *We present a framework for detecting unknown 3D objects in RGBD-images and extracting representations suitable for robotics tasks such as grasping. We address cluttered scenes with stacked and jumbled objects where simplistic plane pop-out methods are not sufficient. We start by estimating surface patches using a mixture of planes and NURBS (non-uniform rational B-splines) fitted to the 3D point cloud and employ model selection to find the best representation for the given data. We then construct a graph from surface patches and relations between patches and perform graph cut to arrive at object hypotheses segmented from the scene. The energy terms for patch relations are learned from user annotated training data, where we train a support vector machine (SVM) to classify a relation as being indicative of two patches belonging to the same object given a vector of relation features, such as proximity or color similarity. We show preliminary results demonstrating that the approach can segment objects of various shapes in cluttered table top scenes.*

1. Introduction

Segmenting unknown objects from generic scenes is one of the elusive goals of computer vision and in general a very ill defined problem. Thanks to the recent introduction of cheap and powerful 3D sensors (such as the Microsoft Kinect or Asus Xtion-PRO) which deliver a dense point cloud plus color for almost any indoor scene, a renewed interest in 3D methods holds the promise to push the envelope slightly further.

In this work we aim at segmenting unknown objects of arbitrary (but reasonably compact) shape from table top scenes, where objects need not be standing isolated but can be jumbled in heaps. An example for such a scene is shown in Fig. 1. More-

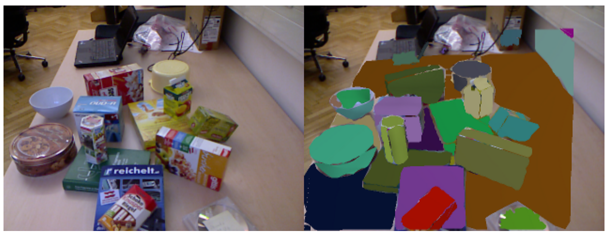


Figure 1. Segmented objects from a cluttered table top scene with stacked and jumbled objects.

over we want a compact and accurate representation of object shapes, suitable in a robotics domain for various manipulation tasks.

The dense and reliable point cloud delivered by a Kinect sensor allows us to robustly fit planar surface patches to parts of the point cloud. These planes are fast to compute and capture a good range of typical man made objects. In order to also model curved objects with high accuracy we furthermore fit NURBS (non-uniform rational B-splines), replacing planes whenever NURBS provide a better fit. We use model selection [12] to find the combination of planes and NURBS optimally explaining the point cloud data.

Segmenting objects from the scene then amounts to identifying groups of surface patches that are likely to belong to the same objects. I.e. we perform perceptual grouping, but not as is more traditionally done in 2D using e.g. edges and junctions, but using 3D surface features and relations. We define several pairwise relations such as proximity or color similarity and form a relation feature vector. Each of these relations indicates to a certain degree that the respective surface patches are likely to belong to the same object, with e.g. closeness being a very good indicator and color similarity being far weaker.

We address this with a learning approach where we use human-annotated ground truth to offline train an SVM to categorize a relation feature vector as either indicating same or different object for a given pair of patches. We then construct a graph from all the surface patches and pairwise relations and use the output of the SVM as the pairwise energy term in a graph cut based segmentation. The resulting segmentation is able to detect many typical objects as they arise in robotics tasks (books, boxes, cups, bowls), provided that single surfaces are big enough to be captured in sufficient detail by the Kinect sensor and that enough training data is provided to the SVM to capture all arising surface relations.

The key novelty in our approach lies in combining planes and NURBS with learned relations in a 3D perceptual grouping approach.

The paper is structured as follows. The next section sets the presented work into context with related work in this research field. Sec. 3 explains the approach in detail for the different components of the framework. Experimental results are shown in Sec. 4, before the work ends with a conclusion and outlook in Sec. 5.

2. Related Work

Various approaches to segment objects either in 2D images or in point clouds exist. Early approaches aimed to formulate generic Gestalt principles to organise 2D scenes into objects. For an overview of this early work in perceptual organisation we want to refer to Boyer and Sarkar [2]. More recently Zillich [22] proposed an any-time perceptual grouping framework to segment convex parts in images. Gestalt principles are also used by Kootstra et al. in [11] and [10]. They developed a symmetry detector to initialize segmentation based on a Markov Random Field (MRF). Furthermore Kootstra et al. developed a quality measure based on Gestalt principles to rank segmentation results.

Many state-of-the-art approaches formulate image segmentation as energy minimization with an MRF [1, 19, 3, 17]. In addition to an appearance model computed from colour and texture, which is commonly used to better distinguish foreground from background, Bergstrom et al. [1] formulate an objective function where it is possible to incrementally add constraints generated through human-robot interaction. In [20] Werlberger et al. propose a variational model for interactive segmentation using a shape prior. This method is based on minimizing the Geodesic Active Contour energy.

Active segmentation is proposed in Mishra et al. [13] and [14] where an image point is fixated and the shortest path in a log polar transformed edge image is computed. In addition to the edges computed from colour and texture information the above authors propose to use the depth image from stereo cameras to improve segmentation.

The approach by Hager et al. [9] is able to segment objects from cluttered scenes in point clouds generated from stereo by using a strong prior 3D model of the scene and explicitly modelling physical constraints such as support and handles dynamic changes such as object appearance/disappearance. It is however limited to parametric models (boxes, cylinders).

The problem of fitting higher order surfaces to point clouds was already addressed by the framework of Leonardis et al. [12]. They segment range images by estimating piecewise linear surfaces, modelled with bivariate polynomials. Furthermore they developed a Model Selection framework, which is used to find the best interpretation of the range data in terms of Minimum Description Length (MDL). Instead of using bivariate polynomials we first describe the scene with simple plane models and then substitute planes with NURBS if the approximation of the point cloud is better in terms of MDL. Additionally, we cluster surface patches to objects depending on learned patch relations.

3. Approach

Our approach consists of four major parts, namely plane fitting, NURBS fitting, model selection and object segmentation. The first two parts abstract from the raw point cloud to surface patches. The model selection part determines the combination of surface patches which optimally represents the underlying point cloud. Object segmentation finally uses relations between surface patches to estimate which of the surface models belong together thus forming object hypotheses.

3.1. Plane Fitting

We chose to fit planes into the point cloud as a first abstraction step. Man-made environments contain many planar surfaces, so planes do in fact quite well describe a good portion of the scenes we are interested in from a robotics point of view. Furthermore, planes are easy to fit using robust methods, as opposed to higher-parametric models such as NURBS or superquadrics.

Plane fitting is typically done with a robust method (e.g. some variant of RANSAC [8]) by sequentially fitting models and removing the inliers, and iterating until there are too few points remaining to support a model. Taking into account the scene layout we are typically going to encounter, we added a additional step. Even though we explicitly tackle cluttered scenes with objects lying around in heaps, often at least these heaps can be separated. To this end we iteratively fit a plane, remove its inliers and then perform clustering based on pairwise Euclidean distances on the remaining point cloud. We then perform RANSAC on the subset of points belonging to a single cluster, leading to a significantly increased inlier rate in comparison to the whole remaining point cloud. For scenes with objects lying on a supporting surface this very effectively speeds up the iterative RANSAC procedure.

For all point cloud operations we use PCL (Point Cloud Library) [18], which provides various RANSAC methods as well as clustering and other basic operations.

The resulting planes represent the point cloud not always optimally, because e.g. a cylindric surface will be represented with a number of planar stripes. This issue is corrected by NURBS fitting and model selection described in the following subsections.

3.2. NURBS Fitting

For representing free-form surfaces there are a number of geometric models available. Most widely used in industry are NURBS (non-uniform rational B-splines). The reasons for their popularity are the convenient manipulation and the ability to represent all conic sections, i.e. circles, cylinders, ellipsoids, spheres and so forth. The possibility for refinement through knot insertion allows for adaption to local irregularities, while selecting a certain degree of freedom gives reason about the measured surface we want to fit to.

A good overview of the properties and advantages of NURBS can be found in Chapter 1.1.2 in [6]. NURBS are a generalisation of B-splines, that allow for exact representation of a wide range of objects. For simplicity we will focus on B-splines for now and will postpone the move to NURBS to future work.

3.2.1 B-splines

The mathematical concept of B-splines would go far beyond the scope of this paper. So for those who are interested let us refer to the well known book by Piegl et al. [15] and start from their definition of B-spline surfaces in Chapter 3.4.

$$\mathbf{S}(\xi,\eta) = \sum_{i=1}^{n}\sum_{j=1}^{m} N_{i,p}(\xi) M_{j,p}(\eta)\mathbf{B}_{i,j} \qquad (1)$$

The basic idea behind this formulation is to manipulate the B-spline surface $\mathbf{S} : \mathbb{R}^2 \to \mathbb{R}^3$ of degree p, by changing the values of the *control grid*. The i, j-element of the control grid is called *control point* $\mathbf{B}_{i,j} \in \mathbb{R}^3$ which defines the B-spline surface at its region of influence determined by the *basis functions* $N_{i,p}(\xi), M_{j,p}(\eta)$. $(\xi, \eta) \in \Omega$ are called parameters defined on the domain $\Omega \subset \mathbb{R}^2$.

Refinement is established by *knot insertion*, i.e. increasing the number of control points, and therefore the degrees of freedom, without changing the surface $\mathbf{S}$. A detailed exposition of knot refinement is available in Chapter 5.3 in [15].

3.2.2 Point-Cloud Fitting

Given a set of points $\mathbf{q}_h \in \mathbb{R}^3$ with $h = 1 \dots k$ and $k > mn$ we want to fit a B-spline surface $\mathbf{S}$ with $n > p$, $m > p$ and $p \geq 1$. Writing Eq. (1) as a linear system

$$\mathbf{s} = \mathbf{A}\mathbf{b} \qquad (2)$$

where $\mathbf{s} \in \mathbb{R}^{k\times 3}$ are points on the B-spline surface. $\mathbf{A} = \mathbf{A}(\xi_h, \eta_h) \in \mathbb{R}^{k\times nm}$ contains the values of the basis functions at (ξ_h, η_h) and the vector of control points $\mathbf{b} \in \mathbb{R}^{nm\times 3}$ is the control grid $\mathbf{B} \in \mathbb{R}^{n\times m\times 3}$ written as vector. The (ξ_h, η_h) are precomputed parameters described in Sec. 3.2.3. We look for a solution of the overdetermined linear system (2) in the least-squares sense, i.e. a minimum of

$$d = \sum_{h=1}^{k} |\mathbf{q}_h - \mathbf{s}_h(\mathbf{b})|^2 \qquad (3)$$

with respect to $\mathbf{b}$.

3.2.3 Initialisation

For minimizing the functional in Eq. (3) the parameters (ξ_h, η_h) for $\mathbf{A}$ in Eq. (2) are required. We compute them by finding the closest point $\mathbf{s}_h(\xi_h, \eta_h)$ on the B-spline surface to $\mathbf{q}_h$ using Newton's method. Therefore a B-spline surface is initialised from the front face of the camera-axis-aligned bounding box of the point-cloud (Fig. 2).

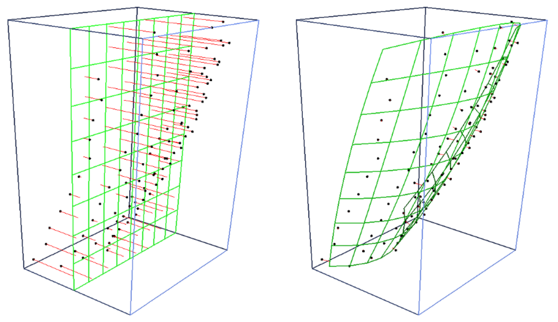


Figure 2. Fitting a B-spline surface (green) by minimizing the closest point distances (red) of a point-cloud (black). For initialisation the camera-axis aligned bounding-box (blue) is used ($m = n = 3$, $p = 2$, $w_a = 1$, $w_r = 0.1$).

3.2.4 Regularisation

To get a smooth surface and to avoid folding we add a regularisation term to Eq. (2) such that a control point tends to lie in the arithmetic mean of its neighbours. For control points at the interior and the boundary of the control grid we consider the 4- and 2-neighbourhood respectively. The regularisation can be written as

$$\mathbf{0} = \mathbf{R}\mathbf{b} \tag{4}$$

and is simply appended to Eq. (2).

$$\begin{bmatrix} \mathbf{s} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} w_a\mathbf{A} \\ w_r\mathbf{R} \end{bmatrix} \mathbf{b} \tag{5}$$

w_a and w_r are the weights defining the influence of the point matching and regularisation.

3.3. Model Selection

For assembling surface patches to object hypotheses, we first need to segment the point cloud into individual patches and estimate the surface model parameters. As shown in the previous section for planes, this can be done with the robust estimation method RANSAC by sequentially fitting models and filtering the inliers. A plane has only three parameters, therefore random sampling is an appropriate approach. For NURBS, where the number of parameters is three times the number of control points, an intractable number of random samples would be necessary to select inliers of a surface patch. For this reason we first explain the point cloud in terms of piecewise planar patches using the sequential RANSAC approach described in Sec. 3.1 and then greedily merge planes and substitute them with NURBS by using Model Selection and a Minimum Description Length (MDL) criterion. In the following paragraph we briefly describe the basic mathematical tool which is introduced by Leonardis et al. [12] for the purpose of range image segmentation and we describe the proposed framework. Our formulation is most similar to the formulation by Prankl et al. [16] who use Model Selection to detect planes in image pairs.

The idea of Model Selection is that the same data point can not belong to more than one surface model. Hence an over-complete set of models is generated and the best subset in terms of an MDL criterion is selected. To select the best model, the savings for each surface hypothesis H can be expressed as

$$S_H = S_{data} - \kappa_1 S_m - \kappa_2 S_{err} \tag{6}$$

where S_{data} is the number of data points N explained by the hypothesis H, S_m stands for the cost of coding different models and S_{err} describes the cost for the error added. κ_1 and κ_2 are constants to weight the different terms. As proposed in [12] we use the number of parameters to define S_m. For the cost S_{err} experiments have shown that the Gaussian error model $\mathcal{N}(\mu_{err}, \sigma^2_{err})$ and an approximation of the log-likelihood has a superior performance. Hence the cost of the error results in

$$S_{err} = -\log \prod_{i=1}^{N} p(f_i|H) = \tag{7}$$

$$\approx \sum_{i=1}^{N} (1 - p(f_i|H)) \tag{8}$$

and accordingly the substitution of Eq. 8 in Eq. 6 yields the savings of a model

$$S_H = \frac{N}{A_m} - \kappa_1 S_m - \frac{\kappa_2}{A_m} \sum_{i=1}^{N} (1 - p(f_i|H)), \tag{9}$$

where A_m is a normalization value for merging two models.

For modelling surface patches we then propose a two step algorithm, where first the savings for individual point clusters are compared and then neighbouring point clusters are greedily merged if the savings of the merged cluster

$$S_{ij} > S_i + S_j \tag{10}$$

is higher than the savings of two individual clusters. Alg. 1 summarizes the proposed surface modelling pipeline.

Algorithm 1 Modelling of surface patches

```
Detect piecewise planar surface patches
for $i = 0 \rightarrow$ number of patches do
    Fit nurbs to patch $i$
    Compute MDL savings $S_{i,nurbs}$ and $S_{i,plane}$
    if $S_{i,nurbs} > S_{i,plane}$ then
        Substitute the model $H_{i,plane}$ with $H_{i,nurbs}$
    end if
end for
Create Euclidean neighbourhood pairs $P_{ij}$ for surface patches
for $k = 0 \rightarrow$ number of neighbours $P_{ij}$ do
    Greedily fit nurbs to neighbouring patches $P_{ij}$
    Compute MDL savings $S_{ij}$ to merged patches
    if $S_{ij} > S_i + S_j$ then
        Substitute individual models $H_i$ and $H_j$ with merged nurbs model $H_{ij}$
    end if
end for
```

3.4. Object Segmentation

The previous sections explained how to find the best representation of a point cloud with different surface patch models. In the last processing step we are now interested in grouping these surface patches into object hypotheses. To this end it is first necessary to figure out which relations between surface patches contribute to the probability that these patches belong together and are part of one and the same object. We define the following relations, which are calculated for neighbouring surface patches:

- r_{ch} ... difference of patch colour
- r_{tr} ... difference of patch texture
- r_{cb} ... colour distance along patch border
- r_{di} ... distance along patch border
- r_{cu} ... curvature along patch border

The first two of the five relations describe differences of global patch properties, while the other three describe differences of local properties along the borders of the patches. The detailed implementation of the relations is explained and discussed in the result Sec. 4.

Each of these relations is defined to produce a value between 0 (same) and 1 (different) - with the exception of r_{cu}, but the degree to which that value indicates two surfaces as belonging to the same object is different for each relation. So a r_{ch} value of 0.3 will typically have a completely different meaning than a r_{tr} value of 0.3. And moreover these will be dependent on the scenes and objects encountered.

We address this with a learning approach. We define relation vectors $\mathbf{r}$

$$\mathbf{r} = \{r_{ch}, r_{tr}, r_{di}, r_{cb}, r_{cu}\} \tag{11}$$

and train an SVM to classify a relation vector as indicating same or different object.

For the offline training phase of the SVM we hand-annotated a set of depth images. Relation vectors between neighbouring surfaces that belong to the same object represent positive training examples, and those between neighbouring patches belonging to different objects or to an object and background represent negative examples.

We use the libsvm package [5], a free SVM software package, with a radial basis function as kernel:

$$\mathbf{K}(x_i, x_j) = e^{\gamma||x_i - x_j||_2} \tag{12}$$

In the online phase, the SVM is capable to provide not only a binary decision $same$ or $notsame$ for each $\mathbf{r}$, but also a probability value $p(same\,|\,\mathbf{r})$ for each decision, based on the theory introduced by Wu et al. [21].

The last processing step makes a global decision and answers the question, which groups of patches form objects. To this end we define a graph, where patches represent nodes and relations represent edges. The graph is not fully connected (which would be computationally prohibitive), as we only define relations between surface patches which are close neighbours. We then employ graph-cut segmentation, introduced by Felzenszwalb et al. [7], using the above probability values as the pairwise energy terms.

4. Experiments

Each learning approach is only as good as its training data, in our case training images for the SVM. The training images must be complex enough, so that the trained SVM can later distinguish between objects, which are e.g. next to each other or stacked. On the other side the images must also contain simple examples to learn the typical relations between the surface patches on the same object.

We created a training set of 27 images together with annotations. All of the images show a tabletop scene, 17 of the images show several boxes and

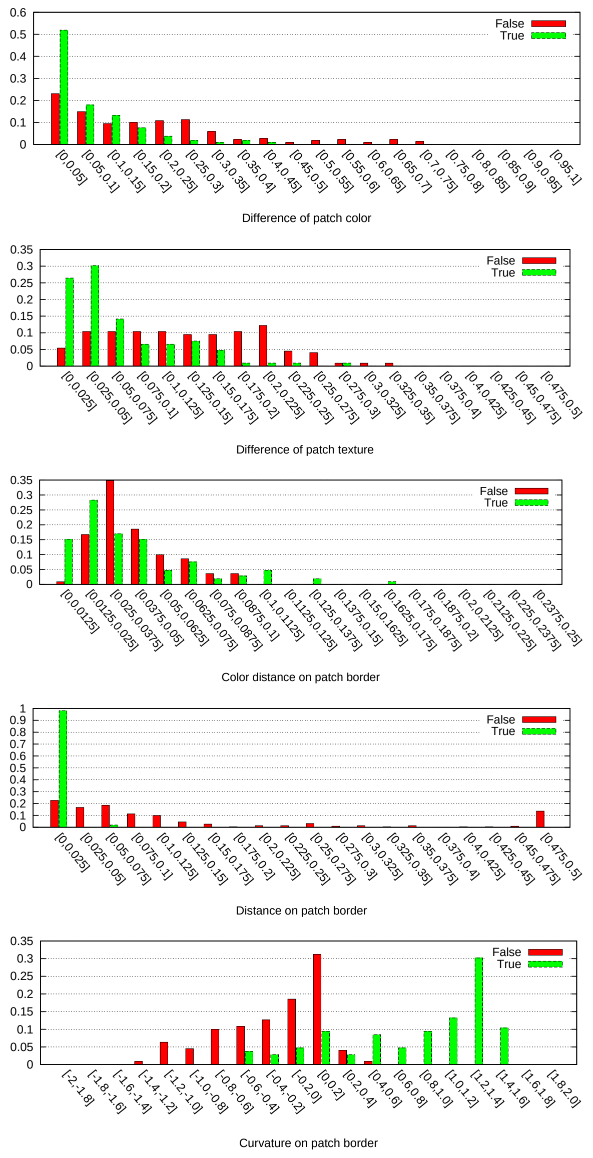


Figure 3. Histograms of relation values for positive and negative examples for relations r_{ch}, r_{tr}, r_{cb}, r_{di}, r_{cu}.

the other ten images show cylindric objects, such as cups, cookie packaging or other kitchenware. The first two rows of Fig. 4 show some of the objects, used for the training period. To give an intuition into the meaning of the different relation values, Fig. 3 shows the distribution of relation values for positive and negative training examples.

The difference of the color histogram is calculated as the Fidelity distance a.k.a. Bhattacharyya coefficient of the UV components in the YUV color space. The comparison of the UV components is less susceptible to brightness changes in the scene than a comparison in the RGB color space. The first histogram in Fig. 3 shows for rising equality (i.e. values close to 0) also a rising number of true examples, while the negative examples are widely distributed over the histogram.

The texture rate of a surface patch is the rate between the number of canny edge pixels on the surface patch and the sum of all surface patch pixels. The distribution shows that surfaces with similar texture rate tend to belong together, with again the negative examples more spread than the positive ones. Shadows on uniform surfaces are a problem, because they can cause edges and thus fake texture.

The third histogram shows again color distance, but now as mean value between neighbouring points (in the image space) on the border of the patches. The distributions of the positive and negative examples are nearly similar. One reason for this is that a lot of our training objects have different colours on neighbouring surfaces, such as the boxes in the top left image of Fig. 4. Another reason for the weak performance of colour is the wrong assignment of colour to 3D points at occlusion boundaries, which is an artefact of the Kinect sensor we were using.

The distance value along patch borders is calculated as mean distance between neighbouring border points. As expected, the positive examples show a high peak for very small distances. Note that for now we do not learn that an object bisected by an occluder should be treated as one object.

The last histogram of Fig. 3 shows the curvature relation, expressed as the mean angle between neighbouring points along patch borders. It shows that positive (convex) curvature typically indicates same object (e.g. two sides of a box joining) whereas negative (concave) curvature typically indicates two different objects.

The results of the evaluation of the SVM and the graph-cut algorithm of four test sets with 50 images is shown in Tab. 1. Examples of the results are shown in Fig. 4. The first four rows show successful examples, the last two image rows show nearly successful segmentation at the systems limit.

5. Conclusion

We presented a framework for segmenting unknown objects in RGBD-images of cluttered table top scenes, by first approximating surfaces with a combination of planes and NURBS and then segmenting the scene based on learned relations between surface patches. One of the problems we are still facing is the combinatorial explosion when trying to replace several planes (think of a couple of stripes ap-

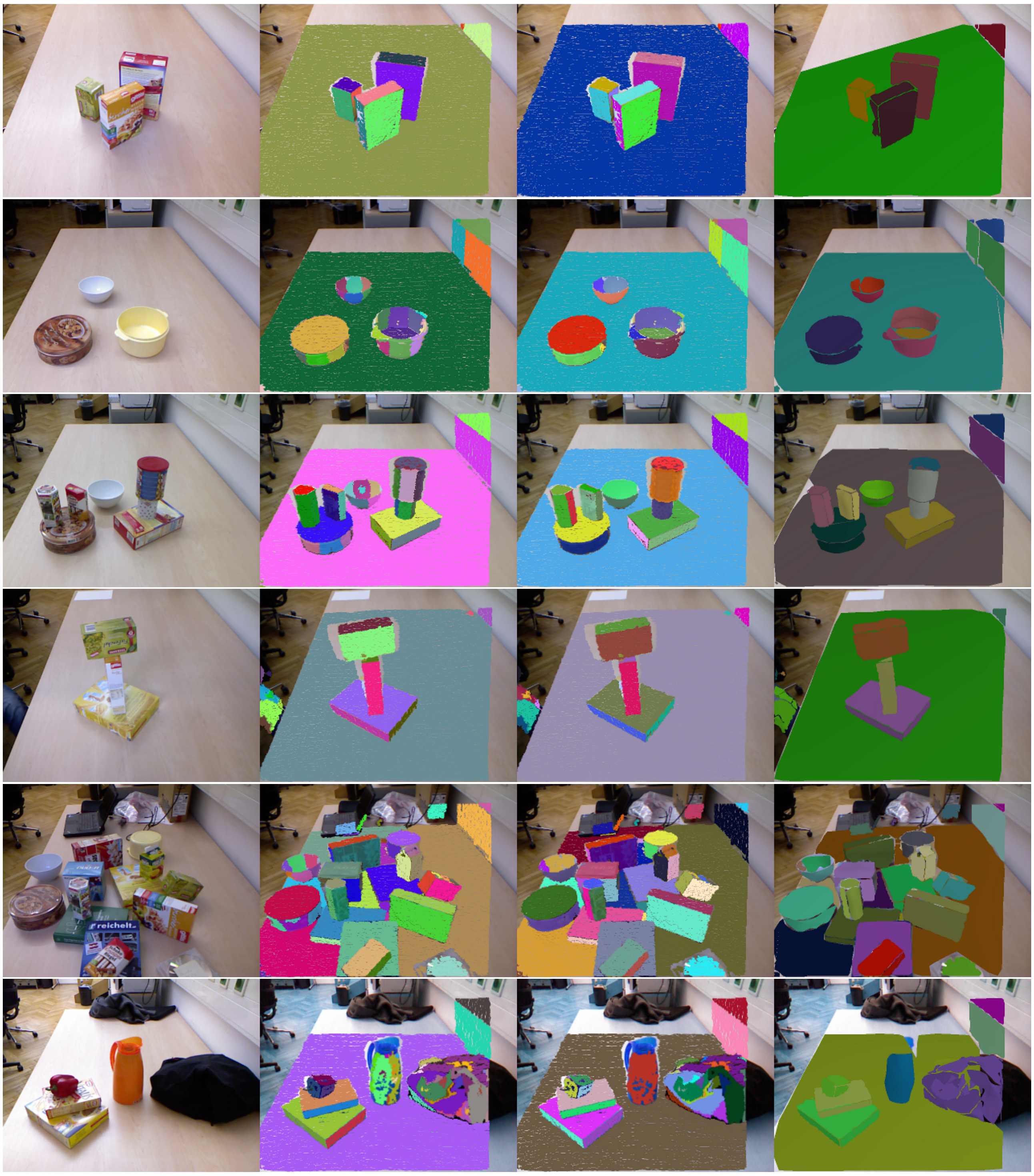


Figure 4. Selected examples of the proposed approach. The first column shows the color image, the second one the estimated planes, the third the plane patches and NURBS after model selection, still as point cloud. The last column shows the segmented object models.

proximating a cylinder) simultaneously with a single NURB, as we would go from pairs to n-tuples with possibly large n. We intend to address this by following not a plane-first-then-NURBS approach, but by first identifying clusters of points forming a single smooth surface area, based on curvature derived from normal vectors. To this end we will employ an improved iterative normal estimation scheme [4] and then trying to locally find the optimal combination of planes and NURBS (again using model selection)

Set	Nr.	SVM acc. (#)	u.seg	o.seg
Boxes	14	100% (158)	0.2%	0.6%
St. Boxes	16	91.5% (224)	1.2%	12.2%
Cylinders	11	91.8% (135)	1.8%	9.1%
Mixed	9	84.7% (406)	6.9%	39.0%

Table 1. Results of object segmentation for different test sets with boxes, stacked boxes, cylinders and mixed objects. Columns: Type of set, number of images, accuracy of SVM prediction (number of relation vectors), under-segmentation and over-segmentation of objects.

for that particular surface area rather than globally for the whole scene.

Acknowledgements

The research leading to these results has received funding from the European Communitys Seventh Framework Programme [FP7/2007-2013] under grant agreement No. 215181, CogX.

References

[1] N. Bergström, M. Björkman, and D. Kragic. Generating object hypotheses in natural scenes through human-robot interaction. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 827–833. IEEE, 2011. 2

[2] K. L. Boyer and S. Sarkar. Perceptual organization in computer vision: status, challenges, and potential. *Comput. Vis. Image Underst.*, 76(1):1–5, 1999. 2

[3] Y. Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary amp; region segmentation of objects in N-D images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 105 –112 vol.1, 2001. 2

[4] F. Calderon, U. Ruiz, and M. Rivera. Surface Normal Estimation with Neighborhood Reorganization for 3D Reconstruction. *Progress in Pattern Recognition Image Analysis and Applications*, 4756:321–330, 2007. 7

[5] C.-c. Chang and C.-j. Lin. LIBSVM : A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27:1—-27:27, 2011. 5

[6] J. A. Cottrell, T. J. R. Hughes, and Y. Bazilevs. Isogeometric Analysis. *Continuum*, page 355, 2010. 3

[7] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision*, 59(2):167–181, Sept. 2004. 5

[8] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cortography. *Communications of the ACM*, 24(6), 1981. 3

[9] G. D. Hager and B. Wegbreit. Scene parsing using a prior world model. *The International Journal of Robotics Research*, 2011. 2

[10] G. Kootstra, N. Bergström, and D. Kragic. Fast and Automatic Detection and Segmentation of Unknown Objects. In *Humanoids*, Bled, 2011. 2

[11] G. Kootstra, N. Bergström, and D. Kragic. Gestalt Principles for Attention and Segmentation in Natural and Artificial Vision Systems. In *SPME*, Shanghai, 2011. 2

[12] A. Leonardis, A. Gupta, and R. Bajcsy. Segmentation of range images as the search for geometric parametric models. *International Journal of Computer Vision*, 14(3):253–277, Apr. 1995. 1, 2, 4

[13] A. K. Mishra and Y. Aloimonos. Active Segmentation. *I. J. Humanoid Robotics*, 6(3):361–386, 2009. 2

[14] A. K. Mishra, Y. Aloimonos, L. F. Cheong, and A. Kassim. Active Segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 6(3):361–386, Aug. 2011. 2

[15] L. Piegl and W. Tiller. *The NURBS book*. Monographs in visual communication. Springer, 1997. 3

[16] J. Prankl, M. Zillich, B. Leibe, and M. Vincze. Incremental Model Selection for Detection and Tracking of Planar Surfaces. In *Proceedings of the British Machine Vision Conference*, pages 87.1—-87.12. BMVA Press, 2010. 4

[17] C. Rother, V. Kolmogorov, and A. Blake. "GrabCut": interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, Aug. 2004. 2

[18] R. Rusu and S. Cousins. 3d is here: Point cloud library (pcl). In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1–4. IEEE, 2011. 3

[19] S. Vicente, V. Kolmogorov, and C. Rother. Joint optimization of segmentation and appearance models. *2009 IEEE 12th International Conference on Computer Vision*, (Iccv):755–762, 2009. 2

[20] M. Werlberger, T. Pock, M. Unger, and H. Bischof. A Variational Model for Interactive Shape Prior Segmentation and Real-Time Tracking. In *International Conference on Scale Space and Variational Methods in Computer Vision (SSVM)*, Voss, Norway, 2009. 2

[21] T. Wu and C. Lin. Probability estimates for multi-class classification by pairwise coupling. *The Journal of Machine Learning Research*, 5:975–1005, 2004. 5

[22] M. Zillich. Incremental Indexing for Parameter-Free Perceptual Grouping. In *31st Workshop of the Austrian Association for Pattern Recognition*, pages 25–32, 2007. 2

Segmentation of Unknown Objects in Indoor Environments

Andreas Richtsfeld, Thomas Mörwald, Johann Prankl, Michael Zillich and Markus Vincze

***Abstract*— We present a framework for segmenting unknown objects in RGBD-images suitable for robotics tasks such as object search, grasping and manipulation. While handling single objects on a table is solved, handling complex scenes poses considerable problems due to clutter and occlusion. After pre-segmentation of the input image based on surface normals, surface patches are estimated using a mixture of planes and NURBS (non-uniform rational B-splines) and model selection is employed to find the best representation for the given data. We then construct a graph from surface patches and relations between pairs of patches and perform graph cut to arrive at object hypotheses segmented from the scene. The energy terms for patch relations are learned from user annotated training data, where support vector machines (SVM) are trained to classify a relation as being indicative of two patches belonging to the same object. We show evaluation of the relations and results on a database of different test sets, demonstrating that the approach can segment objects of various shapes in cluttered table top scenes.**

I. INTRODUCTION

Segmenting unknown objects from generic scenes is an enabler in many robotics tasks, such as object search, grasping and manipulation, but remains one of the ambitious and elusive goals of computer vision and is in general a very ill defined problem. With the recent introduction of cheap and powerful 3D sensors (such as the Microsoft Kinect or Asus XtionPRO) which deliver a dense point cloud plus color for almost any indoor scene, a renewed interest in 3D methods holds the promise to push the envelope slightly further.

In this work we aim at segmenting unknown objects of arbitrary (but reasonably compact) shape from table top scenes, where objects need not stand isolated but can be jumbled in heaps. An example for such a scene is shown in Fig. 1. Moreover we want a compact and accurate representation of object shapes, suitable in a robotics domain for various tasks.

The dense and reliable point cloud delivered by the mentioned sensors allows, after pre-segmentation based on surface normals, to robustly fit planar surface patches to parts of the point cloud. Planes are fast to compute and capture a good range of typical man made objects. In order to also model curved objects with high accuracy we fit NURBS (non-uniform rational B-splines) and replace planes whenever NURBS provide a better fit. We use model selection [11] to find the combination of planes and NURBS optimally explaining the point cloud data.

Segmenting objects from a scene then amounts to identifying groups of surface patches that are likely to belong to the same objects. I.e. we perform perceptual grouping,

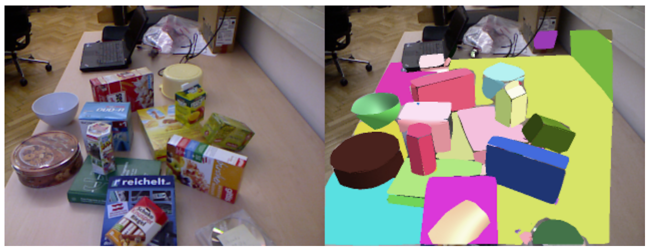


Fig. 1. Segmented objects from a cluttered table top scene.

but not as is more traditionally done in 2D using e.g. edges and junctions, but using 3D surface relations. We define several pairwise relations, based on grouping principles such as proximity, similarity or continuation and create a relation feature vector for each pair of surface patches. We distinguish two types of feature vectors, one for neighbouring and one for non-neighbouring surface patches, each containing relations that are applicable to neighbouring or non-neighbouring surfaces respectively. Each of these relations indicates to a certain degree that the respective surface patches are likely to belong to the same object, with e.g. closeness being a very good indicator and color similarity being far weaker. To arrive at a single value of relatedness of two surface patches from such a vector of very different individual relations we use a learning approach. Human-annotated ground truth data is used to train a SVM for each type of feature vector. The SVMs categorize a relation feature vector as either indicating same or different object for a given pair of patches. We then construct a graph from all the surface patches and pairwise relations and use the output of the SVMs as the pairwise energy term in a graph cut based segmentation. The resulting segmentation is able to detect many typical objects as they arise in service robotics tasks (books, boxes, product packaging, cups, etc.), provided that single surfaces are big enough to be captured in sufficient detail by the sensor and that enough training data is provided to the SVMs to capture all arising surface relations.

The key novelty in our approach lies in a 3D perceptual grouping approach of plane and NURBS representations based on learned relations, suitable for segmenting objects of reasonable size and compactness, even if they are stacked into heaps or are partially occluded.

The paper is structured as follows. The next section sets the presented work into context with related work in this research field. Sec. III shows the structure of the proposed framework, while Sec. IV and Sec. V explains more detailed the components of the framework. Evaluation and results are shown in Sec. VI, before the work ends with a conclusion and an outlook in Sec. VII.

Vision for robotics group (v4r), Automation and Control Institute (ACIN), Vienna University of Technology, 1040 Vienna, Austria
{ari, tm, jp, mz, mv}@acin.tuwien.ac.at

II. RELATED WORK

Various approaches to segment objects either in 2D images or in point clouds exist. Early approaches aimed to formulate generic Gestalt principles to organise 2D scenes into objects. For an overview of this early work in perceptual organisation we want to refer to Boyer and Sarkar [2]. More recently Zillich [23] proposed an any-time perceptual grouping framework to segment convex parts in images. Gestalt principles are also used by Koostra et al. in [10] and [9]. They developed a symmetry detector to initialize segmentation based on a Markov Random Field (MRF). Furthermore Koostra et al. developed a quality measure based on Gestalt principles to rank segmentation results.

Many state-of-the-art approaches formulate image segmentation as energy minimization with a MRF [1], [19], [3], [17]. In addition to an appearance model computed from colour and texture, which is commonly used to better distinguish foreground from background, Bergstrom et al. [1] formulate an objective function where it is possible to incrementally add constraints generated through human-robot interaction. In [20] Werlberger et al. propose a variational model for interactive segmentation using a shape prior. This method is based on minimizing the Geodesic Active Contour energy.

Active segmentation is proposed in Mishra et al. [12] and [13] where an image point is fixated and the shortest path in a log polar transformed edge image is computed. In addition to the edges computed from colour and texture information the above authors propose to use the depth image from stereo cameras to improve segmentation.

The approach by Hager et al. [8] is able to segment objects from cluttered scenes in point clouds generated from stereo by using a strong prior 3D model of the scene and explicitly modelling physical constraints such as support and handles dynamic changes such as object appearance/disappearance. It is however limited to parametric models (boxes, cylinders).

The problem of fitting higher order surfaces to point clouds was already addressed by the framework of Leonardis et al. [11]. They segment range images by estimating piecewise linear surfaces, modelled with bivariate polynomials. Furthermore they developed a Model Selection framework, which is used to find the best interpretation of the range data in terms of Minimum Description Length (MDL). Instead of using bivariate polynomials we first pre-segment the scene, describe the scene with plane or NURBS models, and then decide which combination of planes and NURBS approximates the point cloud better in terms of MDL. Additionally, we cluster surface patches to objects depending on learned patch relations.

III. SYSTEM OVERVIEW

The proposed system consists of two major parts, namely *Data Abstraction* and *Object Segmentation*. Figure 2 shows the processing chain for those parts in detail. Data abstraction starts with pre-segmentation, based on surface normals, before model selection decides whether a plane or a NURBS surface patch fits better to the given pre-segmented point cloud data. The second major part has to decide, which sets of surface patches together form objects. First, relations between surface patches are calculated and feature vectors are built. Support vector machines are used to obtain a meaningful energy term from the feature vector, which is used by a graph cut algorithm to globally optimize the splitting of a graph consisting of surface patches as nodes and relations as edges with energy terms, estimated by the SVMs.

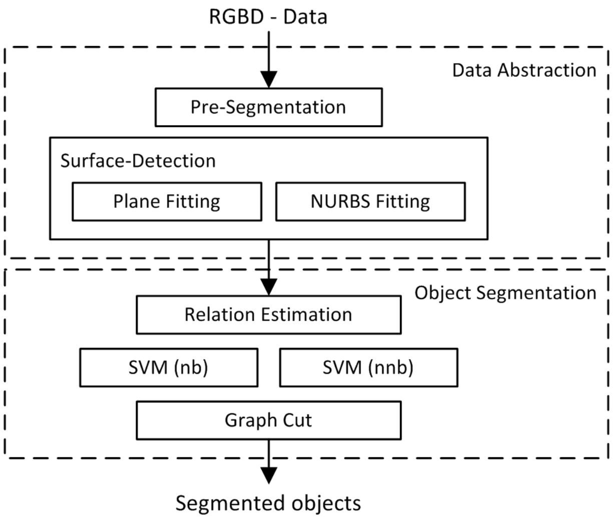


Fig. 2. System overview: RGBD data abstraction into a best plane/NURBS surface model representation and object segmentation using Graph-Cut based on relations trained offline using support vector machines (SVMs).

IV. DATA ABSTRACTION

Object segmentation based on non-abstracted sensor data is a complex task since reasoning on raw sensor data is more difficult than on abstracted data. We propose to abstract real-world sensor data into meaningful pieces before we do object segmentation. Many everyday objects which can be found in households are of compact size and have smooth surfaces, which are sometimes planar, but also free-form surfaces. Representation of such objects requires a simple and computationally efficient planar model as well as a more complex one, such as NURBS (Non-Uniform Rational B-Splines). Our framework pre-segments RGBD data into planar patches, before a model selection algorithm decides whether NURBS fitted to neighbouring patches, or the individual planar patches better represent the underlying data.

A. Pre-Segmentation

For assembling surface patches to object hypotheses, we first need to segment the point cloud into individual patches and estimate the surface model parameters. We use planes and NURBS, thus ideal pre-segmentation amounts to cluster points located on a smooth surface. One possibility would be to compute the surface normals for the point cloud and cluster neighbouring points with similar normals. Estimated normals from point cloud data are noisy or tend to smooth edges and simple clustering of normals would lead to under-segmentation. Hence, we use a stronger plane model to over-segment the point cloud and try to merge neighbouring planes to smooth surfaces in a second step (described in the next Section IV-B).

Plane detection is done by recursively clustering normals in a hierarchical fashion. Therefore a pyramid is created by sub-sampling the point cloud and planes detected in adjacent levels have to compete for points. To select the best subset based on a Minimum Description Length (MDL) criterion we adapted the Model Selection approach proposed by Prankl et al. [15] for plane detection in image pairs. In contrast to Prankl et al. who implemented a QBP approach, we use the model selection criterion in an greedy approach to substitute planes detected at the coarser level with overlapping more detailed ones at the detailed level.

B. Surface Detection

Estimation of surface models can be done with the robust estimation method RANSAC by sequentially fitting models and filtering the inliers. The plane model has only three parameters, therefore random sampling is an appropriate approach. For NURBS, where the number of parameters is three times the number of control points, an intractable number of random samples would be necessary to select inliers of a surface patch. For this reason we greedily merge the pre-segmented planar patches and substitute them with NURBS by using Model Selection and Minimum Description Length (MDL) in a similar way as indicated in Section IV-A.

The idea of Model Selection is that the same data point can not belong to more than one surface model. Accordingly, the savings

$$S_H = \frac{N}{A_m} - \kappa_1 S_m - \frac{\kappa_2}{A_m} \sum_{i=1}^{N} (1 - p(f_i|H)), \qquad (1)$$

for models of neighbouring patches S_i and S_j are compared to the savings of a model fitted to a merged patch S_{ij} and in case S_{ij} is larger the individual patches are substituted. In Eq. 1 N is the number of data points explained by the hypothesis H, S_m stands for the cost of coding different models, $p(f_i|H)$ is the probability (modelled with a Gaussian error model), that a data point f_i belongs to H and A_m is a normalization value representing the size of merged patches. κ_1 and κ_2 are constants to weight the different terms. In the following paragraphs estimation of the model parameter for planes and NURBS is described.

1) Plane Fitting: Even if planes could in principe be modelled as simply specific cases of NURBS, planes have an exceptional role in our framework for reasons of computational efficiency. For the least squares estimation of the plane parameter we use a SVD of the covariance matrix of the data points. On modern CPU/GPU hardware this allows to compute the surface normals of a point cloud with a *sliding window* approach in real time.

2) NURBS Fitting: For representing free-form surfaces there are a number of geometric models available. Most widely used in industry are NURBS (non-uniform rational NURBSs). The reasons for their popularity are the convenient manipulation and the ability to represent all conic sections, i.e. circles, cylinders, ellipsoids, spheres and so forth. The possibility for refinement through knot insertion allows for adaption to local irregularities, while selecting a certain polynomial degree gives reason about the curvature of the measured surface we want to fit to.

A good overview of the properties and advantages of NURBS can be found in Chapter 1.1.2 in [6]. The mathematical concept of NURBS would go far beyond the scope of this paper. So for those who are interested let us refer to the well known book by Piegl et al. [14], which explains the fitting technique used in Chapter 9.4. Initialisation is done by performing a principle component analysis (PCA) on the point-cloud of interest and defining the control points of the NURBS according to the eigenvectors of the PCA.

V. OBJECT SEGMENTATION

The previous section explained how to find the best representation of a point cloud, but now we are interested in grouping these patches into object hypotheses. First we introduce relations between surface patches which contribute to distinguish between patches which belong together or not and create a feature vector of relations. We propose to use two different feature vectors, one for neighbouring and another one for non-neighbouring patches, consisting of different relations. This two step approach follows quite naturally, because different types of relations hold for neighbouring patches e.g. touching along a common boundary, vs. patches separated due to occlusion. Based on the estimated vectors, we train one SVM for each type of vector to predict later the probability of affiliation of surface patches. We build a graph, using the surface patches as nodes and the estimated prediction values from the SVM as edges. Finally, a Graph-Cut algorithm decides globally which surface patches belong together to form a common object hypothesis based on energy minimization.

A. Relation Estimation

The major task of object segmentation is to find relations between surface patches which indicate that they belong to one object. Inspired by Gestalt principles [21], such as proximity, similarity and continuation, which were widely used in cognitive computer vision, we investigated the relevance of several relations, inferred and adapted to 3D-data. By experimental investigations of those relations, we quickly experienced the difference of relevance of features, if patches are neighbours in 3D space or not. We propose to use different relations for neighbouring and non-neighbouring surface patches and introduce two different feature vectors, based on the following relations:

- r_{cu} ... mean curvature along 2D patch border
- r_{di} ... mean distance along 2D patch border
- r_{vdi} ... variance of distance along 2D patch border
- r_{cb} ... difference of color along 2D patch border
- r_{ch} ... difference of patch colour
- r_{tr} ... difference of patch texture
- r_{ga} ... gabor filter match
- r_{fou} ... fourier filter match

- r_{cu3} ... curvature along 3D patch border
- r_{di3} ... distance along 3D patch border
- r_{nm} ... difference of mean surface normals direction
- r_{nv} ... difference of variance of normals direction
- r_{ac} ... mean angle between normals of nearest contour points
- r_{dn} ... mean distance in normal direction of nearest contour points
- r_{md} ... minimum distance between patch borders
- r_{rs} ... relative patch size difference

The feature vector of neighbouring (r_{nb}) and non-neighbouring surface patches (r_{nnb}) are then constructed in the following way:

$$\mathbf{r_{nb}} = \{r_{cu}, r_{di}, r_{vdi}, r_{cb}, r_{ch}, r_{tr}, r_{ga}, r_{fou}, r_{cu3}, r_{di3}\} \quad (2)$$

$$\mathbf{r_{nnb}} = \{r_{ch}, r_{tr}, r_{ga}, r_{fou}, r_{nm}, r_{nv}, r_{ac}, r_{dn}, r_{md}, r_{rs}\} \quad (3)$$

The first four relations are merely suitable for feature vectors of neighbouring patches, because 2D neighbourhood is required to estimate these values. The curvature relation r_{cu} expresses the mean angle between surface normals of neighbouring points along patch borders. r_{di} describes the mean distance between neighbouring patch border points and r_{vdi} is the variance of the distance along the border. Colour difference along the border r_{cb} and color difference of the patch r_{ch} is calculated as the Fidelity distance a.k.a. Bhattacharyya coefficient of the UV components in the YUV color space, because they are less susceptible to brightness changes in the scene than a comparison in the RGB color space. The texture rate of a surface patch is the rate between the number of canny edge pixels on the surface and the sum of all surface pixels. The difference of texture r_{tr} is then defined as difference of the texture rate between two surface patches. The gabor and fourier filter are implemented as proposed in [18]. For the gabor filter six directions (in 30° steps) with five different kernel sizes are used. A feature vector with 60 values is built from the mean and the standard deviation. The gabor filter match r_{ga} is the minimum difference between these two vectors ($d = \sum_{i,j} \sqrt{(\mu_i - \mu_j)^2 + (\sigma_i - \sigma_j)^2}$), when one feature vector gets shifted so that different orientations of the gabor features are matched. This guarantees roughly scale and rotation invariance for the gabor filter. The fourier filter match r_{fou} is again calculated as Fidelity distance of five histograms consisting of 8 bins and filled with the normalised absolute values of the first five coefficients from the DFT. The curvature r_{cu3} and the distance r_{di3} along the 3D borders is compared to the 2D patch border merely calculated if the neighbouring points in the 2D image space are also neighbours in 3D space. It turned out that a good distance for measuring of neighbourhood in 3D is 0.015m. r_{nm} and r_{nv} are the difference of the mean and variance from all normals of a surface patch.

For the next two features the 20% of nearest points between two patches are calculated. r_{ac} and r_{dn} compare then the mean angle between the surface normals and the mean distance in normal direction. Relation r_{md} measures the minimum distance between the patch borders and the last relation r_{rs} describes the relative difference of the patch size of two patches.

B. *Support Vector Machine Learning*

The introduced relation vectors r_{nb} and r_{nnb} are used in an offline phase to train the two SVM using a hand-annotated set of depth images. Feature vectors of patch pairs from the same object represent positive training examples and vectors from pairs between two different objects and objects and background represent negative examples. With this strategy, not only the affiliation of patches to objects, but also the disparity of object patches to other objects or background will be learned.

For the training, as well as prediction and evaluation we use the libsvm package [4], a free SVM software package. As kernel is the radial basis function (RBF) used:

$$\mathbf{K}(x_i, x_j) = e^{\gamma ||x_i - x_j||_2} \quad (4)$$

In the online phase, the SVM is capable to provide not only a binary decision *same* or *notsame* for each relation vector $\mathbf{r}$, but also a probability value $p(same|\mathbf{r})$ for each decision, based on the theory introduced by Wu et al. [22].

C. *Graph Cut Segmentation*

After SVM prediction some estimates may contradict when forming object hypotheses. A globally optimal solution has to be found to overcome vague or wrong local predictions from the two SVMs. To this end we define a graph, where patches represent nodes and relations represent edges. The graph is fully connected, as we defined relations between all surface neighbors, as well as all non-neighbouring surface patches. We then employ graph-cut segmentation, introduced by Felzenszwalb et al. [7], using the probability values from the SVMs as the pairwise energy terms.

VI. RESULTS

Each learning approach is only as good as its training data, in our case training images for the SVM, containing objects of different shape complexity in several assemblies. The training images should be complex enough to distinguish later between objects in the scene. On the other side, images should contain simple examples to learn the typical relations between surface patches from the same object.

TABLE I
NUMBER OF IMAGES AND RELATIONS FROM THE DATABASE.

	learn	r_{nb}	r_{nnb}	test	r_{nb}	r_{nnb}
Boxes	17	157	48	16	393	2489
Stacked Boxes	12	101	58	12	129	1873
Occluded Obj.	8	73	124	7	58	518
Cylindric Obj.	8	104	250	8	84	650
Mixed Objects				12	407	4553
Complex Scene				11	754	28364
TOTAL	45	435	480	66	1825	38447

Evaluation of the proposed system was done on the object segmentation dataset proposed in [16]. The dataset consists of 111 images in six subsets as shown in Tab. I. For the

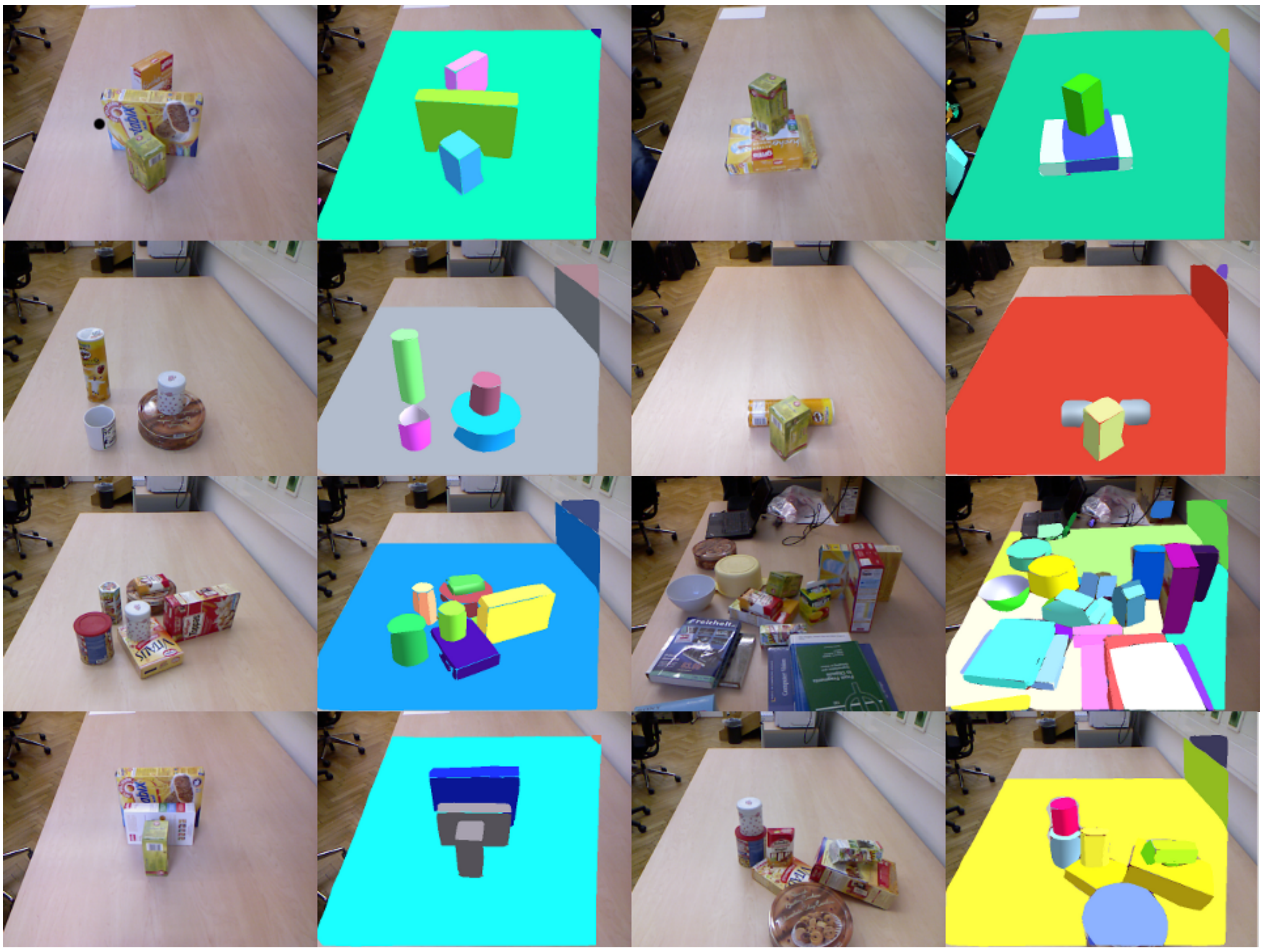


Fig. 3. Selected examples of segmented objects with the proposed approach (objects randomly coloured). The first six examples showing results from each dataset, the last two examples are showing under-segmentation caused by wrong predictions of the SVM_{nnb}.

hole dataset the annotation of objects in the RGBD images is provided to enable learning and also evaluation on these sets. Chen [5] introduced the F-score for evaluation of features for SVM classification. F-score is a technique which measures the discrimination of two sets of real numbers. Given training vectors r_k, $k = 1,\ldots,m$, if the number of positive and negative instances are n_+ and n_-, respectively, then the F-score of the i-th feature is defined as:

$$F(i) = \frac{(\bar{r}_i{}^{(+)} - \bar{r}_i)^2 + (\bar{r}_i{}^{(-)} - \bar{r}_i)^2}{\frac{1}{n_+-1}\sum_{k=1}^{n_+}(r_{k,i}^{(+)} - \bar{r}_i{}^{(+)})^2 + \frac{1}{n_--1}\sum_{k=1}^{n_-}(r_{k,i}^{(-)} - \bar{r}_i{}^{(-)})^2} \quad (5)$$

where $\bar{r}$, $\bar{r}^{(+)}$ and $\bar{r}^{(+)}$ are the average of the i-th feature of the whole, positive and negative data sets, respectively, $r_{k,i}^{(-)}$ is the i-th feature of the k-th negative instance. The numerator indicates the discrimination between the positive and negative sets, and the denominator indicates the one within each of the two sets. The larger the F-score, the more likely this feature is more discriminative. Figure 4 shows the F-score of the learning set from 435 feature vectors for the SVM_{nb} and 480 for the SVM_{nnb}. Relations for the SVM_{nb} are far stronger than for SVM_{nnb} when considering the logarithmic scale. Since the F-score does not reveal mutual information, this does not inevitably mean that all relations together predict utterly precise. The evaluation results from the six different trainings sets are shown in Tab. II. The first column presents the accuracy of the SVM predictions for relations between neighbouring surface patches, followed by the over- and under-segmentation for the case of using just one SVM for graph building. The next three columns show the accuracy of the SVM for non-neighbouring surface

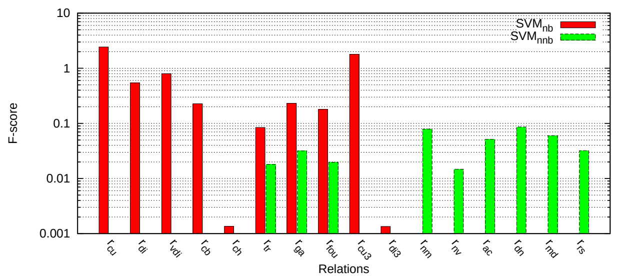


Fig. 4. F-score for relations between surface patches.

TABLE II

RESULTS OF OBJECT SEGMENTATION. SVM ACCURACY, OVER- AND UNDER-SEGMENTATION WHEN USING ONE OR TWO SVM'S.

	SVM_{nb}	F_{os}	F_{us}	SVM_{nnb}	F_{os}	F_{us}
Boxes	88.55%	1.8%	0.2%	98.19%	0.2%	17.2%
St. Boxes	89.15%	1.3%	7.1%	98.99%	0.0%	28.2%
Occl. Obj.	87.93%	16.6%	0.1%	99.23%	0.0%	0.2%
Cyl. Obj.	91.66%	2.6%	0.3%	96.77%	2.6%	3.5%
Mixed Obj.	91.04%	1.9%	19.7%	94.97%	1.3%	39.2%
Complex S.	84.61%	7.0%	8.0%	98.97%	5.4%	146%
TOTAL	87.72%	4.5%	7.9%	98.41%	2.7%	69.5%

patches and the over- and under-segmentation for operation with both SVMs. Over-segmentation F_{os} is defined using the number of correctly assigned object pixels divided by the number of all object pixels and under-segmentation F_{us} using the number of incorrectly assigned pixels divided by the number of all object pixels:

$$F_{os} = 1 - \frac{N_{true}}{N_{all}} \qquad F_{us} = \frac{N_{false}}{N_{all}} \tag{6}$$

The results of the segmentation evaluation in Tab. II show the difference of using one or two SVMs and the associated trade-off between over- and under-segmentation. When using both SVMs, the framework tends to higher under-segmentation, because only one wrongly connected non-neighbouring surface can connect two objects, which can be seen in the last two examples in Fig. 3. On the other hand the second SVM enables handling of occlusions, which can be seen for the results of the occluded objects dataset. For complex scenes is the advantage in over-segmentation not in relation with the disadvantage in under-segmentation. Even when around 98% of the predictions of the SVM_{nnb} are correct, the wrong ones will highly affect the under-segmentation.

VII. CONCLUSION AND FURTHER WORK

We presented a framework for segmenting unknown objects in RGBD-images of cluttered table top scenes, by first approximating surfaces with a combination of planes and NURBS and then segmenting the scene based on learned relations between surface patches. With the proposed approach of using two SVMs for prediction of connectedness between patches, we tackle also the problem of segmenting occluded objects.

One of the problems we are still facing is the weakness of the relations for predictions with the SVM for non-neighbouring surface patches. We intend to address this by further investigations of grouping principles suitable for RGBD-data as well as exploration of different database setups for training of the SVMs.

The presented results of the proposed framework are showing that the approach is promising and has the ability of usage in several indoor robotic tasks where identifying unknown objects or grasping plays a role.

VIII. ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Communitys Seventh Framework Programme [FP7/2007-2013] under grant agreement No. 215181, CogX and by the Austrian Science Foundation under grant agreement No. I513-N23.

REFERENCES

[1] N. Bergström, M. Björkman, and Danica Kragic. Generating object hypotheses in natural scenes through human-robot interaction. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 827–833. IEEE, 2011.

[2] Kim L Boyer and Sudeep Sarkar. Perceptual organization in computer vision: status, challenges, and potential. *Comput. Vis. Image Underst.*, 76(1):1–5, 1999.

[3] Y Y Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary amp; region segmentation of objects in N-D images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 105 –112 vol.1, 2001.

[4] Chih-chung Chang and Chih-jen Lin. LIBSVM : A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27:1—-27:27, 2011.

[5] Yi-wei Chen and Chih-jen Lin. Combining SVMs with Various Feature Selection Strategies. *Strategies*, 324(1):1–10, 2006.

[6] J A Cottrell, T J R Hughes, and Y Bazilevs. Isogeometric Analysis. *Continuum*, page 355, 2010.

[7] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision*, 59(2):167–181, September 2004.

[8] Gregory D Hager and Ben Wegbreit. Scene parsing using a prior world model. *The International Journal of Robotics Research*, 2011.

[9] Gert Kootstra, Niklas Bergström, and Danica Kragic. Fast and Automatic Detection and Segmentation of Unknown Objects. In *Humanoids*, Bled, 2011.

[10] Gert Kootstra, Niklas Bergström, and Danica Kragic. Gestalt Principles for Attention and Segmentation in Natural and Artificial Vision Systems. In *SPME*, Shanghai, 2011.

[11] Aleš Leonardis, Alok Gupta, and Ruzena Bajcsy. Segmentation of range images as the search for geometric parametric models. *International Journal of Computer Vision*, 14(3):253–277, April 1995.

[12] Ajay K Mishra and Yiannis Aloimonos. Active Segmentation. *I. J. Humanoid Robotics*, 6(3):361–386, 2009.

[13] Ajay K Mishra, Yiannis Aloimonos, Loong Fah Cheong, and Ashraf Kassim. Active Segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 6(3):361–386, August 2011.

[14] Les Piegl and Wayne Tiller. *The NURBS book*. Monographs in visual communication. Springer, 1996.

[15] Johann Prankl, Michael Zillich, Bastian Leibe, and Markus Vincze. Incremental Model Selection for Detection and Tracking of Planar Surfaces. In *Proceedings of the British Machine Vision Conference*, pages 87.1—-87.12. BMVA Press, 2010.

[16] Andreas Richtsfeld. The Object Segmentation Database (OSD), http://www.acin.tuwien.ac.at/forschung/v4r, 2012.

[17] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "GrabCut": interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, August 2004.

[18] Ahsan Ahmad Ursani, Kidiyo Kpalma, and Joseph Ronsin. Texture features based on Fourier transform and Gabor filters: an empirical comparison. In *2007 International Conference on Machine Vision*, pages 67–72. Ieee, December 2007.

[19] S Vicente, V Kolmogorov, and C Rother. Joint optimization of segmentation and appearance models. *2009 IEEE 12th International Conference on Computer Vision*, (Iccv):755–762, 2009.

[20] Manuel Werlberger, Thomas Pock, Markus Unger, and Horst Bischof. A Variational Model for Interactive Shape Prior Segmentation and Real-Time Tracking. In *International Conference on Scale Space and Variational Methods in Computer Vision (SSVM)*, Voss, Norway, 2009.

[21] M. Wertheimer. Untersuchungen zur Lehre von der Gestalt. II. *Psychological Research*, 4(1):301–350, 1923.

[22] TF Wu and CJ Lin. Probability estimates for multi-class classification by pairwise coupling. *The Journal of Machine Learning Research*, 5:975–1005, 2004.

[23] Michael Zillich. Incremental Indexing for Parameter-Free Perceptual Grouping. In *31st Workshop of the Austrian Association for Pattern Recognition*, pages 25–32, 2007.

Submitted to the International Conference on Pattern Recognition (ICPR), 2012.
Provided as part of DR 2.5: Qualitative behaviour models, grasping of novel objects

Implementation of Gestalt Principles for Object Segmentation

Andreas Richtsfeld, Michael Zillich and Markus Vincze
Automation and Control Institute (ACIN), Vienna University of Technology
$\{$*ari,mz,vm*$\}$*@acin.tuwien.ac.at*

Abstract

Gestalt principles have been studied for about a century and were used for various computer vision approaches during the last decades, but became unpopular because the many heuristics employed proved inadequate for many real world scenarios. We show a new methodology to learn relations inferred from Gestalt principles and an application to segment unknown objects, even if objects are stacked or jumbled and tackle also the problem of segmenting partially occluded objects. The relevance of the relations for object segmentation is learned with support vector machines (SVMs) during a training period. We present an evaluation of the relations and show results of the segmentation framework.

1. Introduction and related work

A lot of work has be done on investigation but also on implementation of Gestalt principles for visual perception systems. Wertheimer, Köhler and Kofka were the pioneers of studying Gestalt psychology when they started to investigate this theory about 100 years ago. Wertheimer [21] introduced in a seminal paper Gestalt principles and Köhler [9] and Koffka [8] further developed his theories.

Gestalt principles (also called *Gestalt laws*) aim to formulate the regularities according to which the perceptual input is organized into unitary forms, also referred to as wholes, groups, or Gestalten [17]. In visual perception, such forms are the regions of the visual field whose portions are perceived as grouped or joined together, and are thus segregated from the rest of the visual field. While Gestalt psychologists call these phenomena "laws", a more accurate term would be "principles of perceptual organization." These principles are much like heuristics, which are mental shortcuts for solving problems. *Perceptual Grouping* uses then Gestalt principles to group visual features together to meaningful parts (unitary forms or objects). The following *Gestalt principles* were introduced and discussed in [21, 9, 8, 17, 12]: *Proximity, Similarity, Continuity, Symmetry, Closure, Figure-Ground Articulation, Good Form, Simplicity, Common Fate, Common region and Element Connectedness*.

Early approaches to segment objects aimed to formulate generic Gestalt principles to organise 2D scenes into objects. For an overview of this early work in perceptual organisation we want to refer to Boyer and Sarkar [2]. They list approaches along two axes: dimensionality of the sensor signal (2D, 3D, with or without motion) and level of abstraction.

Many state-of-the-art approaches formulate image segmentation as energy minimization with a Markov Random Field (MRF) [1, 19, 3, 15]. Koostra et al. [10] used in addition Gestalt principles and developed a symmetry detector to initialize segmentation based on a MRF. Furthermore Koostra et al. developed a quality measure based on Gestalt principles to rank segmentation results. Werlberger et al. [20] propose a variational model for interactive segmentation using a shape prior. This method is based on minimizing the Geodesic Active Contour energy. The approach by Hager et al. [7] is able to segment objects from cluttered scenes in point clouds generated from stereo by using a strong prior 3D model of the scene and explicitly modelling physical constraints such as support, and handles dynamic changes such as object appearance/disappearance. It is however limited to parametric models (boxes, cylinders). Also the approach of Sala and Dickinson [16], who worked on contour grouping based on a user-defined vocabulary of simple part models, is limited to their vocabulary which allows detection of cylinders and bent or tapered cubic prisms. Ferrari et. al [6] introduced the kAS network, where k contour segments are connected to build up a scale and orientation invariant vocabulary. This enables object class detection on various object classes after a training period.

While in recent years more often contours were used for perceptual grouping approaches, we want to show

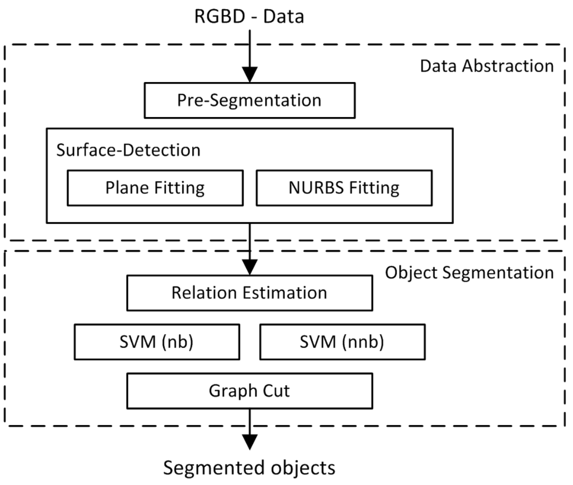


Figure 1. Object segmentation framework

that the introduction of cheap and powerful RGBD-sensors which deliver a dense point cloud plus color for almost any indoor scene, recapturing of well-studied perceptual grouping techniques for segmentation of unknown objects holds the promise to push the envelope slightly further. After abstraction of raw data to a higher level, we address the problem of segmenting unknown objects from cluttered scenes by defining relevant relations which we infer from Gestalt principles and by learning the relations during a training period with hand-annotated image data using support vector machines (SVMs). Finally a Graph-Cut algorithm is used to globally optimize segmentation of unknown objects.

The paper is structured as follows. Section 2 presents the object segmentation framework before Sec. 3 discusses the investigated relations and explains their implementation. Results are presented in Sec. 4, before the work ends with a conclusion and an outlook in Sec. 5.

2. Object segmentation framework

The object segmentation framework consists of two major parts, namely *Data Abstraction* and *Object Segmentation*, which can be seen in Fig. 1 on a more detailed level. Data abstraction starts with pre-segmentation, based on surface normals, before model selection [11] decides whether planar or NURBS (non-uniform rational b-spline) surface patches are fitting better to the given point cloud data. A detailed description of the data abstraction goes beyond this paper, instead we refer to [14].

The second part has to decide which sets of surface patches together form objects. First, relations between surface patches are calculated and two different types of feature vectors, one for neighbouring (nb) patches and one for non-neighbouring (nnb) patches, are built from it. Two SVMs, one for each type of vector are then used to obtain a meaningful energy term from a feature vector, which is afterwards used by a graph cut algorithm [5] to globally optimize splitting of a graph, constructed of surface patches as nodes and estimated energy terms from the SVMs as edges.

3. Relations between surface patches

The crucial task of segmenting unknown objects is to find relations between surface patches indicating that they belong together and form objects, and which of the relations are valid for a wide variety of objects. Based on the above grouping principles, we introduce the following relations between surface patches:

- r_{co} ... similarity of patch colour
- r_{tr} ... similarity of patch texture quantity
- r_{ga} ... gabor filter match similarity
- r_{fo} ... fourier filter match similarity
- r_{rs} ... relative patch size similarity
- r_{cb3} ... color similarity on 3D patch borders
- r_{cu3} ... curvature on 3D patch borders
- r_{di2} ... mean depth on 2D patch borders
- r_{vd2} ... variance of depth on 2D patch borders
- r_{md} ... minimum distance between patches
- r_{nm} ... angle between mean surface normals
- r_{nv} ... difference of variance of surface normals
- r_{ac} ... mean angle of normals of nearest contour p.
- r_{dn} ... mean distance in normal direction of nearest contour points

The first five relations are inferred from the *similarity principle*, which can be integrated in many different ways. Similarity of patch colour r_{co} is implemented by comparing the histograms using the Fidelity distance in the YUV color space. Texture similarity is realised in three ways as difference of texture quantity r_{tr}, as Gabor filter match and as Fourier filter match. The texture quantity is calculated as relation of canny edge pixels to all surface patch pixels, while the Gabor and Fourier filter are implemented as proposed in [18]. For the Gabor filter six directions (in 30° steps) with five different kernel sizes are used. A feature vector with 60 values is built from the mean and the standard deviation. The Gabor filter match r_{ga} is the minimum difference between these two vectors, when one feature vector gets shifted so that different orientations of the Gabor features are matched. This guarantees roughly scale and rotation invariance. The Fourier filter match r_{fo} is again calculated as Fidelity distance of five histograms consisting of 8 bins, filled with the normalised absolute values of the first five coefficients from the DFT. The similarity of size r_{rs} is calculated as the relative difference of the patch size of two patches.

Subsequent relations use the 2D and 3D relationship of neighbouring patches. Colour similarity r_{cb3} and curvature r_{cu3} are calculated along the 3D patch border between surface patches and the mean r_{di2} and the variance of depth r_{vd2} are calculated along borders in the 2D image space. While r_{cb3} represents again a relation inferred from *similarity*, curvature r_{cu3} represents a relation inferred from *continuity* as well as *closure*, which could also be interpreted as a compactness principle in the 3D image space. The mean of depth r_{di2} and the variance of the depth r_{vd2} on 2D patch borders in the image space describe relations inferred from a mixture of the *proximity* and *continuity principle*.

The minimum distance between patches r_{md} is inferred from the *proximity principle* and r_{nm} and r_{nv} are the difference of the mean and variance of the normals of a surface patch and represent the *symmetry* between two patches. For the last two relations the nearest contour points of two patches are calculated. r_{ac} and r_{dn} compare the mean angle between the surface normals and the mean distance in normal direction and are inferred from the *continuity principle*.

With the previous introduced relations, two different types of feature vectors are designed, one for neighbouring patches r_{nb} and one for non-neighbouring surface patches r_{nnb}.

$$\mathbf{r_{nb}} = \{r_{co}, r_{tr}, r_{ga}, r_{fo}, r_{rs}, r_{cb3}, r_{cu3}, r_{di2}, r_{vd2}\} \quad (1)$$

$$\mathbf{r_{nnb}} = \{r_{co}, r_{tr}, r_{ga}, r_{fo}, r_{rs}, r_{md}, r_{nm}, r_{nv}, r_{ac}, r_{dn}\} \quad (2)$$

Feature vectors of type r_{nb} are processed in SVM_{nb} and vectors of type r_{nnb} are processed in SVM_{nnb}. Using a second SVM for prediction of connectedness between non-neighbouring patches solves the problem of segmenting objects when they are split in two parts by partial occlusion.

4. Results

Evaluation of the relations and the proposed object segmentation framework has been done with the database at [13], which consists of table top scenes organized in several learn- and test-sets with various kinds of objects and with different complexities of the scenes.

Figure 3 shows the F-score for each relation used for the SVMs. The F-score is a value describing the discrimination of two sets of real numbers and is explained more detailed in [4]. All relations of feature vector r_{nb}

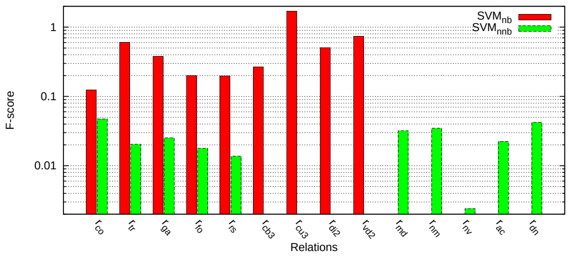


Figure 3. F-score of relations.

have higher F-scores and have therefore higher relevance than the ones used for feature vector r_{nnb}. This implies that neighbouring patches can be easier connected correctly than non-neighbouring patches.

We experienced during evaluation of r_{nb} that relations based on the *similarity principle* are more relevant to connect patches, specifically r_{co}, r_{tr} and r_{fo}. Others are more relevant to separate patches, e.g. r_{cu3} and r_{di2}, which are inferred mainly from the *continuity* and *closure principle*. The former ones influence more the over-segmentation and the latter more the under-segmentation of the object segmentation system.

A single relation of r_{nnb} never leads to a decision that two non-neighbouring patches belong together, because the low prior probability of positive decisions always causes negative decisions of SVM_{nnb} and relations are not relevant enough. Only when using all relations for r_{nnb} the SVM_{nnb} sometimes decides positive, which leads to better results for over-segmentation F_{os} at the cost of worse results for under-segmentation F_{us}. Table 1 shows this results of the object segmentation framework. SVM_l and SVM_t refer to the accuracy of the SVM prediction on the learn- and the test-set, and the latter two columns show over-segmentation F_{os} and under-segmentation F_{us} on the test-set of the object segmentation database. When using relations between non-neighbouring patches, the error of over-segmentation decreases nearly by half, but only at the cost of a higher error of under-segmentation. Which error is more important, depends on the application.

Table 1. Object segmentation results

	SVM_l	SVM_t	F_{os}	F_{us}
$\mathbf{r_{nb}}$	93.5%	87.8%	4.5%	6.5%
$\mathbf{r_{nnb}}$	97.7%	97.0%	29.2%	69.1%
$\mathbf{r_{nb}} + \mathbf{r_{nnb}}$	96.4%	96.0%	2.6%	26.8%

Figure 2 shows a segmentation example[1] (random color). The majority of the objects are correctly segmented, only the bowl causes problems due to the separation into two parts.

[1]More results: http://users.acin.tuwien.ac.at/arichtsfeld/?site=4

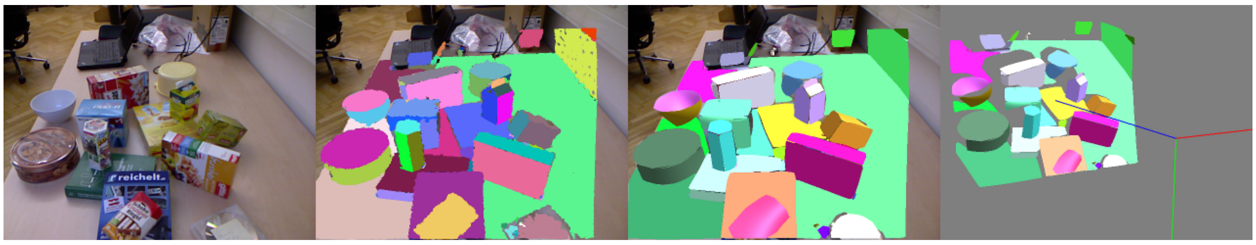


Figure 2. Original image, abstracted surface patches, segmented objects, 3D representation.

5. Conclusion and further work

We presented a repertoire of relations based on perceptual grouping principles which are suitable to segment unknown objects from cluttered scenes. Furthermore, we presented evaluation of these relations and results of the object segmentation framework. One of the problems we are still facing is the weakness of the relations for prediction of connectedness of non-neighbouring surface patches, which allows to segment partially occluded objects. We intend to address this by further investigations of relations based on the *continuity* or *closure principle* using contour segment grouping.

The presented results of the proposed framework shows that grouping approaches produce good results for a wide variety of complex scenes, using Gestalt principles learned from a small set of training examples. The approach is promising and shows applicability in many computer vision tasks where identifying unknown objects plays a role.

References

[1] N. Bergström, M. Björkman, and D. Kragic. Generating object hypotheses in natural scenes through human-robot interaction. In *Intelligent Robots and Systems (IROS)*, pages 827–833. IEEE, 2011.

[2] K. L. Boyer and S. Sarkar. Perceptual organization in computer vision: status, challenges, and potential. *Computer Vision and Image Understanding*, 76(1):1–5, 1999.

[3] Y. Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary amp; region segmentation of objects in N-D images. In *International Conference on Computer Vision (ICCV)*, volume 1, pages 105 –112, 2001.

[4] Y.-w. Chen and C.-j. Lin. Combining SVMs with Various Feature Selection Strategies. *Strategies*, 324(1):1–10, 2006.

[5] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision*, 59(2):167–181, Sept. 2004.

[6] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid. Groups of adjacent contour segments for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(1):36–51, 2008.

[7] G. D. Hager and B. Wegbreit. Scene parsing using a prior world model. *The International Journal of Robotics Research*, 2011.

[8] K. Koffka. *Principles of Gestalt psychology*. Routledge, 1999.

[9] B. W. Köhler. Gestalt Psychology Today. *American Psychologist*, 14(12):727–734, 1959.

[10] G. Kootstra, N. Bergström, and D. Kragic. Fast and Automatic Detection and Segmentation of Unknown Objects. In *Humanoids*, Bled, 2011.

[11] A. Leonardis, A. Gupta, and R. Bajcsy. Segmentation of range images as the search for geometric parametric models. *International Journal of Computer Vision*, 14(3):253–277, Apr. 1995.

[12] S. Palmer and I. Rock. Rethinking perceptual organization: The role of uniform connectedness. *Psychonomic Bulletin & Review*, 1(1):29–55, 1994.

[13] A. Richtsfeld. Object Segmentation Database (OSD), http://www.acin.tuwien.ac.at/forschung/v4r, 2012.

[14] A. Richtsfeld, T. Mörwald, J. Prankl, J. Balzer, M. Zillich, and M. Vincze. Towards Scene Understanding Object Segmentation Using RGBD-Images. In *Proceedings of the 2012 Computer Vision Winter Workshop (CVWW)*, Mala Nedelja, Slovenia, 2012.

[15] C. Rother, V. Kolmogorov, and A. Blake. "GrabCut": interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, Aug. 2004.

[16] P. Sala and S. Dickinson. Contour Grouping and Abstraction Using Simple Part Models. *European Conference on Computer Vision*, 6315:603–616, 2010.

[17] D. Todorovic. Gestalt principles. *Scholarpedia*, 3(12):5345, 2008.

[18] A. A. Ursani, K. Kpalma, and J. Ronsin. Texture features based on Fourier transform and Gabor filters: an empirical comparison. In *International Conference on Machine Vision*, pages 67–72. Ieee, Dec. 2007.

[19] S. Vicente, V. Kolmogorov, and C. Rother. Joint optimization of segmentation and appearance models. *International Conference on Computer Vision (ICCV)*, (Iccv):755–762, 2009.

[20] M. Werlberger, T. Pock, M. Unger, and H. Bischof. A Variational Model for Interactive Shape Prior Segmentation and Real-Time Tracking. In *International Conference on Scale Space and Variational Methods in Computer Vision (SSVM)*, Voss, Norway, 2009.

[21] M. Wertheimer. Untersuchungen zur Lehre von der Gestalt. II. *Psychological Research*, 4(1):301–350, 1923.

Isogeometric Finite-Elements Methods and Variational Reconstruction Tasks in Vision – A Perfect Match

Jonathan Balzer
University of California
Los Angeles, CA 90095
USA
balzer@cd.ucla.edu

Thomas Mörwald
Vienna University of Technology
1040 Vienna
Austria
moerwald@acin.tuwien.ac.at

Abstract

Inverse problems are abundant in vision. A common way to deal with their inherent ill-posedness is reformulating them within the framework of the calculus of variations. This always leads to partial differential equations as conditions of (local) optimality. In this paper, we propose solving such equations numerically by isogeometric analysis, *a special kind of finite-elements method. We will expose its main advantages including superior computational performance, a natural ability to facilitate multi-scale reconstruction, and a high degree of compatibility with the spline geometries encountered in modern computer-aided design systems. To animate these fairly general arguments, their impact on the well-known depth-from-gradients problem is discussed, which amounts to solving a Poisson equation on the image plane. Experiments suggest that, by the isogeometry principle, reconstructions of unprecedented quality can be obtained without any prefiltering of the data.*

1. Introduction

1.1. Motivation

Vision is dominated by inverse problems in the sense that from an observation, one wishes to make inferences about its cause, e.g., shape from shading aims at computing the shape of a Lambertian object from the gray values it induces on the image plane. Inverse problems often struggle with ill-posedness, meaning that they admit no solution at all, or if they do, it is either ambiguous or does not depend continuously on the input data. A general strategy to deal with this issue is to turn away from classical or strict solutions to ones that are merely optimal with respect to an application-dependent cost. Indeed, we assess that energy minimization is ubiquitous in vision. It can be categorized roughly into two classes: A major line of work, following direct discretization and quantization of the objective function, combinatorializes the optimization problem and solves it by one of many graph-based algorithms, among which graph cuts have proven to be particularly successful in recent years [7].

A quite different strategy is restricting all considerations to a *continuous* version of the optimization problem. In that case, the energy takes the form of a functional on an infinite-dimensional linear space of functions. The *calculus of variations* is mainly concerned with deriving a condition of (local) optimality for such functionals, their so-called Euler-Lagrange equation [8]. Generally, the latter is a, possibly nonlinear, partial differential equation (PDE) of arbitrary order and as such rarely solvable by analytical means. In most vision-related works, the remedy of choice is approximating the occurring differential operators by weighted differences of the function values on neighboring grid points, thereby transforming the original PDE into a system of algebraic equations [14].

Digital images and the regular arrangement of their pixels provide an ideal computational grid for the finite-differences method (FDM). This is probably why, in the

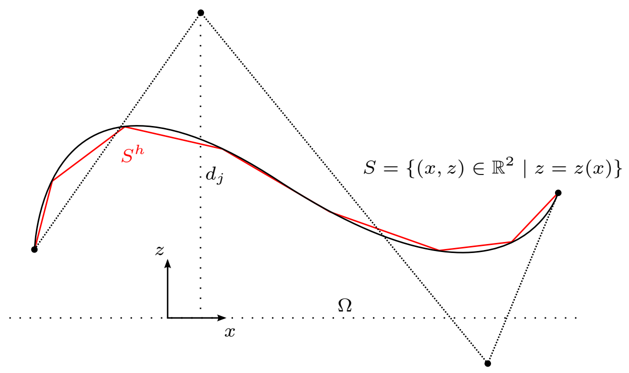


Figure 1. Isogeometric analysis is a finite-elements method to solve partial differential equations defined on a spline surface S (black) with parametric domain Ω. Classical FEMs operate on polygonal meshes such as S^h (red).

realm of variational reconstruction, the finite-elements method (FEM) has been widely neglected, and if not so, it is conducted on unstructured meshes. In this paper, we advocate *isogeometric analysis*, a particular variant of the FEM that operates on B-spline patches. We will argue in Section 2 why it addresses the needs of vision extremely well. To further support our claims and to show the method in action, in Section 3, it is applied to a well-known problem in visual reconstruction, the integration of a gradient field into a depth map. Although just a case study, the proposed algorithm outperforms state-of-the art ones in terms of efficiency, accuracy, and robustness.

2. Finite-element methods for vision

2.1. Conventional approach

To foster a better understanding of what follows let us briefly summarize the fundamental principles guiding the finite-elements method. Its central idea is to approximate the *candidate solutions* $z : S \to \mathbb{R}$ of a PDE posed on a domain $S \subset \mathbb{R}^3$, say a surface, by representing them as linear combinations

$$z(x) = \sum_{j=1}^{n} d_j b_j(x) \tag{1}$$

of some pre-defined basis functions $b_j : S \to \mathbb{R}$. The coefficients d_j are sometimes called the *degrees of freedom* (DOFs). This way, the search for a minimum of the underlying energy functional is confined to a vector of coefficients $\boldsymbol{d} = (d_j) \in \mathbb{R}^n$ in a tractable n-dimensional subspace of the original infinite-dimensional function space. In other words, the underlying functional is discretized directly, not its Euler equation. This bears some resemblance with graph-based optimization, only with the difference that images of solutions, mostly the set of real numbers, remain un-quantized.

The FEM is preferred over the FDM especially when geometry and/or topology of S are more complex than e.g. the image plane's because it inevitably couples digital geometry representation with the shape of the b_j. In fact, as illustrated in Figure 1, S is usually replaced by a polygonal approximation S^h, a collection of *finite elements* (FEs) or *mesh*. The functions[1] $b_j : S^h \to \mathbb{R}$ are then constructed as to model the local behavior of the overall solution (1) by supporting bivariate polynomials locally around elementary geometric entities, like the vertices of the mesh. FEMs admit two modes of refinement: An *h-refinement* increases the local element density. The smoothness of the solution can be controlled by raising the polynomial degree of the basis (*p-refinement*). Let us record in preparation of the next paragraph that any FEM is characterized by 1. the choice of geometric model for S, and 2. the type of basis functions b_j. For a comprehensive introduction to the classical FEM, we refer to [14].

2.2. Isogeometric analysis and its implications

Contemporary product development consists of a design phase in which the geometry of a part is specified by the engineer, and a phase in which its physical properties (e.g., stress resistance, electrical/thermal conductivity, and others that can be modeled by a PDE) are validated at hand of FE analysis. Often, the two phases are carried out iteratively. All the worse, that at each such cycle, a conversion becomes necessary because – for historical reasons – computer-aided design (CAD) systems and conventional FEMs found on totally different representations of the computational domain. This conversion is an error-prone, costly, and time-consuming process, as it almost certainly requires manual assistance by a trained user. As a remedy, Hughes *et al.* [12] propose to refrain from tesselating the CAD-native spline models into polygonal meshes, the input format presumed by current FE solvers. Their insight is that every CAD model already possesses its own set of basis functions to do FE analysis with. The concept truly deserves the predicate *isogeometric*: Translated from greek, it means that *one and the same* representation, a spline basis, is used for specifying the geometry and solving PDEs on it. To our best knowledge, Elguedj *et al.* [5] are the only authors so far to consider isogeometric analysis (IGA) in a vision-related scenario, specifically for optical flow estimation. However, with their background in material testing of metal sheets, they do not elaborate on its significance for reconstruction and vision in general, the study of which will be the first contribution of this paper. In fact, isogeometric methods convince by the following advantages:

1. **Natural parametrization:** Unlike polygonal meshes, splines carry a natural parametrization, i.e., a bijective mapping from the domain S to some subset Ω of Euclidean space. Consequently, not only the FE analysis itself but all other operations that work on planar images extend to curved surfaces in a straightforward manner, cf. Figure 2(a).

2. **Projective invariance:** Projecting the control points of a NURBS surface first and then evaluating it yields the same result as proceeding in reverse order.

3. **Meaningful priors:** A majority of man-made objects have been designed on a computer, and IGA provides a simple mechanism to embrace this prior knowledge directly into the reconstruction process. Smoothing is not only an avoidable preprocessing step. But it also necessitates scale selection, i.e., the choice of the

[1]Note that with slight abuse of notation, we do not distinguish the basis functions on S and its approximation S^h.

"correct" smoothing parameter, which, given a class of CAD surfaces to look for, is almost trivial within the IGA framework, cf. Figure 3.

4. **Direct reverse engineering:** Why initially use splines for synthesis (design) and later point clouds, triangle meshes, or similar for analysis (reconstruction)? The reverse engineering pipeline from optical measurements of a physical prototype back to a digital model can be shortened. IGA makes conversions between different representations obsolete, cf. Figure 2(b) and the video included in the supplemental material.

5. **Multi-scale reconstruction:** Polynomial bases of high degree are realized effortlessly and at any desired scale by parametric splines, whereas constructing quadratic basis functions on polygons is already far from trivial, even on such simple entities as triangles. This is significant in view of the fact that the analysis of images at multiple scales is a fundamental and often-needed technique in image processing and vision.

6. **Accuracy:** The geometric approximation power of polygon meshes is limited. Spline surfaces on the other hand are "continuous" in the sense that, although defined by a finite number of DOFs, they acquire the very shape envisioned by the designer. Furthermore, they can be evaluated *exactly* and at *arbitrary* locations. The same holds true for the results of IGA, which, by definition, are expanded in the same basis as the geometry, thus enabling subpixel resolutions without spending extra interpolation efforts. This becomes particularly important in visualization which unfortunately still requires polygonal representations of both, geometry and solutions. Note, however, that hardware-accelerated direct rendering of splines is on the way, and yet, it is much easier to transform a spline patch into a mesh than vice-versa.

7. **Efficiency and robustness:** IGA features both mechanisms of refinement known from classical FEMs on polygonal surfaces. But opposed to a mesh, a spline surface remains faithful to the original geometry when refined. Loosely speaking, a mesh could need a lot more DOFs to approximate the *geometry* "well" enough than the *solution*. This may result in a substantial computational overhead. The refinement of splines, however, is geometry-independent. Hence, it is possible in principle to tailor the resolution to the requirements of the physical process being modelled by the underlying PDE *not* the geometric model. This has two far-reaching consequences: The content of natural images is known to be concentrated on a much smaller set than the collection of its gray values. Reconstruction based on the FDM always utilizes as many DOFs as there are pixels and is thereby highly redundant. By allocating resources tuned to the part of the data of actual interest, IGA leads to very efficient algorithms, which underlines its potential in real-time applications. This immediately implies the superior robustness of isogeometric methods. As an example, consider the linear case, in which, as we will see later in our case study (Section 3.4), the PDE is transformed into a sparse linear system. The smaller the system matrix, the faster it is to invert, and the better is its condition and hence the numerical stability.

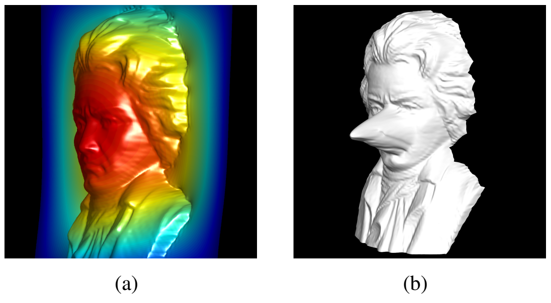


Figure 2. Advantages of the isogeometric approach: (a) PDEs can be solved on patches of arbitrary shape, all with the same framework. This example shows the color-coded solution to a Poisson problem with constant source term under homogenous Dirichlet conditions. (b) The reconstruction is available as a spline patch, readable and editable by all common CAD programs. Beethoven has undergone trimming as well as plastic surgery.

3. A case study: depth from gradients

Numerous computer vision/optical metrology techniques such as photometric stereo, shape from shading, or deflectometry acquire the surface *slope* at points on an unknown object rather than their spatial locations directly. The essence of reconstruction is integrating measured normal or gradient data into a visual surface representation.

3.1. Continuous variational model

A widespread variational formulation of this problem is the following: For the sake of simplicity, we assume that the region of interest on the surface is parameterizable by an orthographic depth map $z : \Omega \to \mathbb{R}$ over the (image) plane $\Omega \subset \mathbb{R}^2$, see Figure 1. Given the measurement of a vector field $\boldsymbol{g}_\mathrm{m} : \Omega \to \mathbb{R}^2$, we wish to find the function z whose gradient ∇z is closest to the data $\boldsymbol{g}_\mathrm{m}$ in terms of the squared L^2-norm, i.e., the function minimizing the Dirichlet-type energy

$$E(z) = \int_\Omega \frac{1}{2} \|\nabla z - \boldsymbol{g}_\mathrm{m}\|^2 d\boldsymbol{x}. \tag{2}$$

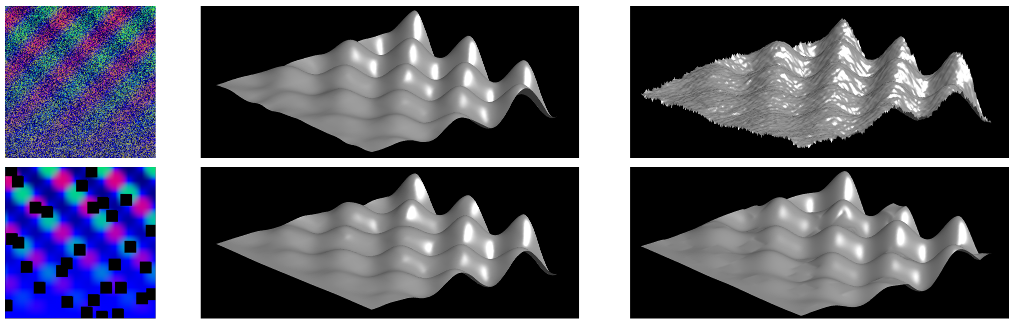

(a) Gradient field (b) IGA reconstruction (raw data) (c) FDM reconstruction (raw data)

Figure 3. An example from the case study in Section 3: By isogeometric analysis, one can put effective and meaningful constraints on the class of admissible shapes and thereby alleviate ramifications of (a) noisy and incomplete data. Of course, results comparable to (b) ours could also be obtained by (c) the FDM. This would come at the cost of preprocessing the data by suitable smoothing/interpolation.

Even if $\boldsymbol{g}_{\mathrm{m}}$ is not integrable and no classical solution exists for which equality $\nabla z = \boldsymbol{g}_{\mathrm{m}}$ holds, the energy (2) implies a projection[2] onto the curl-free portion of $\boldsymbol{g}_{\mathrm{m}}$. This is right in the spirit of Section 1, where we motivated the use of variational models to tackle ill-posed problems. The Euler-Lagrange equation of (2), i.e., a necessary – in the present case even sufficient – condition for optimality, is

$$\Delta z = \operatorname{div} \boldsymbol{g}_{\mathrm{m}}, \tag{3a}$$

in which Δ denotes the Laplace operator, and div the vector field divergence of $\boldsymbol{g}_{\mathrm{m}}$. The unique infinite-dimensional least-squares- or L^2-solution can be deduced from (3a) if and only if complemented by the condition

$$\langle \nabla z, \hat{\boldsymbol{o}} \rangle = \langle \boldsymbol{g}_{\mathrm{m}}, \hat{\boldsymbol{o}} \rangle \tag{3b}$$

on the image boundary $\partial\Omega$ with outer unit normal $\hat{\boldsymbol{o}}$. This is the natural boundary condition arising from the variational principle [4]. It affords that z can move freely above $\partial\Omega$ and thus, the surface adapts optimally to the gradient field there[3]. Note, however, that a solution to above Neumann problem, i.e., Poisson's equation plus aforementioned boundary condition, is unique only up to a scalar: If some $z(x)$ fulfills (3), obviously, the same holds true for $z(x) + c$ with $c \in \mathbb{R}$ because the constant vanishes under differentiation by Δ on the interior of Ω and the directional derivative $\langle \nabla(z + c), \hat{\boldsymbol{o}} \rangle$ on the boundary. Since the integration constant c has no influence on the shape of the surface S parametrized by z, the ambiguity can be resolved either by prescribing the distance of a single point in the computational domain or restricting the search for a solution to all depth maps which are mean-free:

$$\int_{\Omega} z(x)dx = 0. \tag{4}$$

[2]or *Helmholtz-Hodge decomposition* of $\boldsymbol{g}_{\mathrm{m}}$

[3]Opposed to a *Dirichlet* or *essential* boundary condition which fixes the boundary by specifying depth values known in advance (or sometimes even guessed, causing a significant bias, cf. Figure 5(c)).

See [1, Sec. 4.1] for a detailed discussion of both, the boundary and mean-value condition.

3.2. Related work

In his groundbreaking paper [11], Horn shows how to approximate the Laplacian in (3a) by second-order finite differences (FD) on the image grid and solve the resulting algebraic system by a fixed-point scheme. Extensions of Horn's method are too numerous to list here but let us explicitly mention the most recent ones like Harker's and O'Leary's [9] as well as that due to Durou *et al.* [3], who describe a powerful total-variation-based algorithm capable of resolving discontinuities in the depth map without prior segmentation of the gradient field. The inferiority of FD-based methods should be apparent from the discussion in Sections 2.2 and 3.5. Our work relates to the class of kernel methods [6, 15], which can be thought of as meshfree FEMs in disguise. Similarly, Kovesi applies a basis $\{b_j\}$ of shapelets to the normal adaption problem in scene space [13]. Only a few authors explicitly consider the classical, i.e., non-isogeometric FEM: Hicks employs it for integrating normal fields with three-dimensional support into a foliation of surfaces [10]. Generalizations of Horn's method applicable to such spatially varying normal fields are presented by Balzer [1] and Delaunoy and Prados [2]. None of aforementioned methods is compatible with the geometry representation of contemporary CAD packages. Higher-degree polynomial bases and a multi-scale mechanism are per se possible, at least on polygon meshes, but quite challenging to implement.

3.3. B-splines

Here, as a prototypical application of IGA in vision and as the second contribution of this paper, we present the first FE method for gradient field integration based on *B-splines*. In order not to cloud the key ideas by a complicated index calculus, let us assume for the remainder of

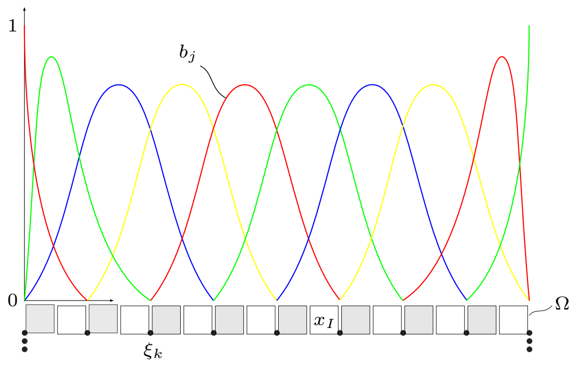


Figure 4. The knots ξ_k, indicated by black dots, uniquely define the B-spline basis functions b_j on the image plane Ω, consisting of pixels x_I (white and gray squares).

the theoretical discussion that, with slight abuse of earlier notation, $\Omega \subset \mathbb{R}$. Everything applies mutatis mutandis to the two-dimensional case. A B-spline is a scalar-valued function of the form (1) with the elements $b_j(x)$ of the basis being compactly supported polynomials defined by the so-called *Cox-de Boor recursion*. An indispensable ingredient of the recursion formula is the fixed *knot vector* $\boldsymbol{\xi} = (\xi_1, \dots, \xi_k, \dots, \xi_{n+p+1}) \in \mathbb{R}^{n+p+1}$. It may be understood as a set of real parameters in calculating the value $b_j(x)$, which we could (but for the sake of notational simplicity will not) indicate by writing $x \mapsto b_j(x; \boldsymbol{\xi})$. The knot vector tiles the parameter domain Ω into smaller intervals (the *finite elements* of IGA), restricted to which, the spline is a polynomial of degree p. Repetition of knots is allowed and leads to a decrease in smoothness. In particular, the spline is interpolating at some ξ_k if and only if ξ_k appears $p+1$ times in $\boldsymbol{\xi}$. Throughout the paper, we assume so-called *open* knot vectors which are interpolating at both ends ξ_1 and ξ_{n+p+1}. Multivariate basis functions can be constructed from tensor products of B-splines in a single variable, the respective Cartesian coordinate of the domain Ω. To extend the image of (1) to higher dimensions, say $d \in \mathbb{N}$, one simply chooses coefficient *vectors* $\boldsymbol{d}_j \in \mathbb{R}^d$. At modeling curves, surfaces, or solids in $\mathbb{R}^3$, these are commonly referred to as *control points*. The knot vector is not to be confused with the much finer pixel grid $\boldsymbol{x}_h = (x_1, \dots, x_I, \dots, x_N) \in \mathbb{R}^N$, which is staggered with respect to $\boldsymbol{\xi}$ and contains no redundant abscissae. The geometric relationship between both is illustrated in Figure 4, alongside with the second-degree B-spline basis induced by the shown knot vector. h-refinement is established by knot insertion. Note that, since the length of $\boldsymbol{\xi}$ is by definition $n+p+1$, during this process, the dimension n of the approximation space must grow with p remaining constant. The effect of degree elevation upon n and the knot vector is less obvious. A detailed exposition of p-refinement, B-splines, and their non-uniform rational generalization NURBS can be found in [16].

3.4. Isogeometric discretization

Multiplication with a test function φ and integration over the domain Ω brings (3) into a variational form, the starting point of any FEM. The equation is said to hold in a weak sense if

$$\int_\Omega \Delta z \varphi dx = \int_\Omega \operatorname{div} \boldsymbol{g}_\mathrm{m} \varphi dx \tag{5}$$

for arbitrary φ in the Sobolev space $H_0^1(\Omega)$, and a fortiori, for all elements of the spline basis i.e. $\varphi \in \{b_1, \dots, b_i, \dots, b_n\}$. Integration by parts and application of the Gauss theorem conveniently relax the differentiability assumptions on both, the solution z and the measured gradient field $\boldsymbol{g}_\mathrm{m}$, by one order:

$$\int_\Omega \langle \nabla z, \nabla b_i \rangle dx = \int_\Omega \langle \boldsymbol{g}_\mathrm{m}, \nabla b_i \rangle dx + \int_{\partial\Omega} \langle \nabla z - \boldsymbol{g}_\mathrm{m}, \hat{\boldsymbol{o}} \rangle b_i dx \tag{6}$$

for all $i \in \{1, \dots, n\}$. Recall that in the strong form (3a), the divergence operator is applied to the data, and that this differentiation amplifies contained noise. Suppose that the boundary condition (3b) holds[4], then the last integral vanishes. The actual discretization is performed by inserting the representation (1) of a candidate solution. The unknown function $z(x)$ is reduced to a finite sequence of unknown coefficients d_j. Thanks to linearity, integration can be restricted to known functions only so that we obtain a linear system $\mathbf{K}\boldsymbol{d} = \boldsymbol{f}$ coupling with each other *stiffness matrix* $\mathbf{K} = ((K_{ij}))$, *displacement* $\boldsymbol{d} = (d_j)$, and *force vector* $\boldsymbol{f} = (f_i)$. The terminology originates from linear elasticity being the primary application of early FEMs and is appropriate regardless of the physical background. In view of (6), we get

$$K_{ij} = \int_\Omega \langle \nabla b_i, \nabla b_j \rangle dx, \quad f_i = \int_\Omega \langle \boldsymbol{g}_\mathrm{m}, \nabla b_i \rangle dx. \tag{7}$$

These integrals are typically computed by Gauss quadrature. In particular, we found the midpoint rule to be sufficient here, which yields, for any two pixels being unit length apart, a simple summation over the grid points x_I in $\boldsymbol{x}_h$:

$$K_{ij} \approx \sum_{I=1}^{N} \langle \nabla b_i(x_I), \nabla b_j(x_I) \rangle, \tag{8a}$$

$$f_i \approx \sum_{I=1}^{N} \langle \boldsymbol{g}_\mathrm{m}(x_I), \nabla b_i(x_I) \rangle. \tag{8b}$$

[4]As a caveat, this must be taken care of later by appropriate modification of the stiffness matrix.

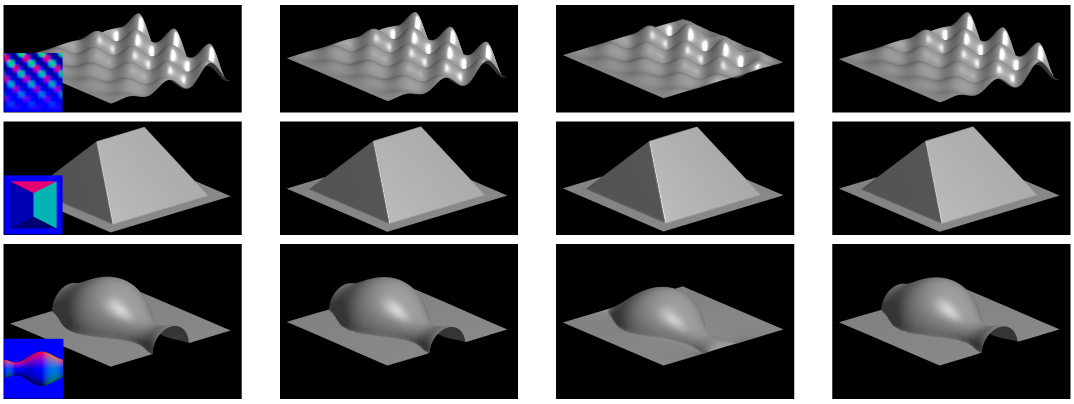

(a) Ground truth and data (b) FDM [4] (c) Kernel method [15] (d) Our method

Figure 5. (a) Ground truth surfaces and gradient fields $\boldsymbol{g}_{\mathrm{m}}$ in false-color representation: The components of corresponding unit normals directly map to the values of the three RGB channels. (b)-(d) Reconstruction results for different numerical methods to solve the Euler-Lagrange equation (3).

	N	s	p	#DOFs	RMSE [px]
Waves	256	6	3	4489	$7.6 \cdot 10^{-5}$
Tent	256	8	1	66049	$6.2 \cdot 10^{-4}$
Vase	256	8	1	66049	$3.3 \cdot 10^{-2}$
Paraboloid	256	2	2	16	$1.3 \cdot 10^{-14}$
Beethoven	256	7	2	16900	$1.3 \cdot 10^{-1}$

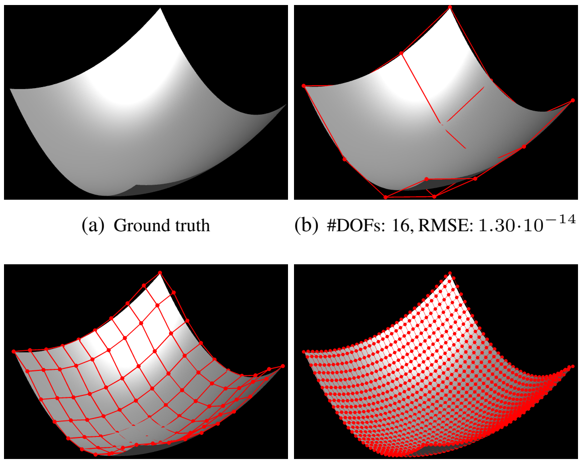
Table 1. Parameters of conducted experiments: N is the size of the input data in one dimension, s the scale, and p the polynomial degree of the spline patch. The number of unknowns is denoted by #DOFs. RMSE stands for the root mean square error between the true and reconstructed gradient fields.

The field $\boldsymbol{g}_{\mathrm{m}}$ is given per pixel x_I. The b_i, b_j, and all of their derivatives can be evaluated at arbitrary locations by the Cox-de Boor formula. Because of their compact support, $\mathbf{K}$ is sparse and thus efficient to invert.

Note that we actually solve the Poisson problem (3) on a planar (but not necessarily degree one) B-spline $\tilde{S} = \Omega$, initially coinciding with the image plane. This simplifies the integrals (7) because the coordinate transformation from Ω to $\tilde{S}$ is just the identity map. A solution $\boldsymbol{d}$ of $\mathbf{K}\boldsymbol{d} = \boldsymbol{f}$ then defines the soughtafter shape S as follows: Since $\tilde{S}$ and S share a common parameter domain Ω, we can simply move the control points of the original patch $\tilde{S}$ by the entries d_j of $\boldsymbol{d}$ in z-direction, cf. Figures 1 and 6.

3.5. Numerical tests

We created a MATLAB implementation of the numerical approach presented in the previous section. In our effort to support reproducibility of research, all data sets and code will be made available online upon publication. The configurations of our IGA solver during various experiments are summarized in Table 1: All images used in the study were


(a) Ground truth (b) #DOFs: 16, RMSE: $1.30 \cdot 10^{-14}$

(c) #DOFs: 100, RMSE: $4.66 \cdot 10^{-16}$ (d) #DOFs: 1156, RMSE: $7.45 \cdot 10^{-16}$

Figure 6. Scale and polynomial degree can be optimally adapted to the data in the isogeometric approach. The reconstruction in (b) remains close to ground truth utilizing as little as 16 unknown coefficients.

of size $N \times N$. We define the scale as the integer $s \in \mathbb{N}$ such that the knot vector consists of $2^{-s} \cdot N$ intervals, e.g., we have $N = 16$ and $s = 1$ in Figure 4. The root mean square error (RMSE) essentially equals (2) normalized with the area of Ω.

Besides the standard FDM, we fed our data into the kernel method (KM) described in [15], which we chose, because it has been published recently, possesses an FEM-flavor, and its implementation is available for download. The results on a first set of examples is shown in Figure 5: Coordinate axes are suppressed because by (4), we are most interested in the shape but less in the exact size of the object. The z-direction should be obvious from each of the

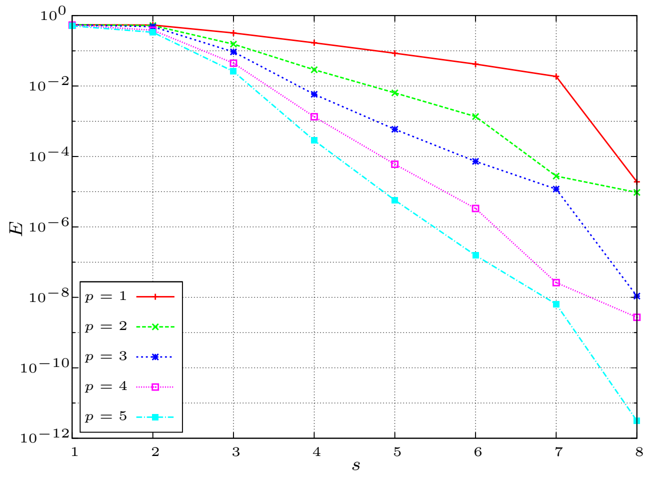


Figure 7. Reconstruction error in dependence on scale s and polynomial degree p for the waves data set.

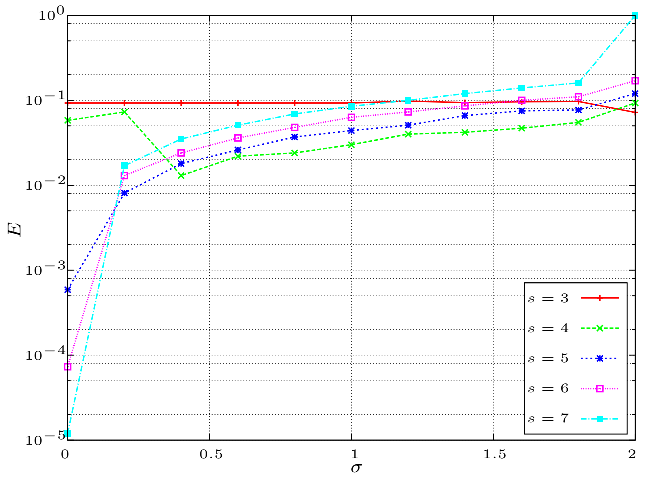


Figure 8. Residual of the energy (2) over the variance σ of noise imposed on the gradient field. These curves were obtained for reconstructions of the waves data set at scales ranging from 3 to 7.

		t_{Assembly} [s]	t_{Solve} [s]	#DOFs	RMSE [px]
Waves	FDM	189.39	3.94	256 × 256	$2.8 \cdot 10^{-3}$
	KM	562.63	338.65	221952	$2.9 \cdot 10^{-1}$
	IGA	22.74	0.82	4489	$7.6 \cdot 10^{-5}$
Tent	FDM	189.39	4.23	256 × 256	$5.6 \cdot 10^{-2}$
	KM	559.92	209.58	221952	$7.7 \cdot 10^{-2}$
	IGA	40.92	8.72	66049	$1.3 \cdot 10^{-1}$
Vase	FDM	189.39	3.84	256 × 256	$5.3 \cdot 10^{-1}$
	KM	557.81	234.85	221952	$2.7 \cdot 10^{-1}$
	IGA	42.12	10.19	66049	$3.3 \cdot 10^{-2}$
Paraboloid	FDM	189.39	4.34	256 × 256	$3.5 \cdot 10^{-4}$
	KM	564.23	357.15	221952	$5 \cdot 10^{-1}$
	IGA	17.35	0.01	16	$1.3 \cdot 10^{-14}$
Beethoven	FDM	189.39	4.18	256 × 256	$5.5 \cdot 10^{-1}$
	KM	557.83	245.4	221952	$3.7 \cdot 10^{-1}$
	IGA	27.17	3.53	16900	$1.3 \cdot 10^{-1}$

Table 2. Quantitative comparison of algorithms: t_{Assembly} is the time needed for equation assembly, t_{Solve} the time for solving the resulting linear system. The value of t_{Assembly} for our implementation is rather pessimistic as assembling the stiffness matrix involves many loops which are handled slowly by the MATLAB kernel.

original surfaces in Figure 5(a). The sinusoidal wave example clearly confirms that unlike previous works, we do not require the unknown surface to be periodic by enforcing the correct boundary condition (3b). It is evident from Figure 5(c) that the KM fails to do so. The visible impression is that all three algorithms resolve the discontinuities in the tent gradient field quite well. A quantitative assessment shows that ours, probably due to its non-local nature, performs slightly worse, see Table 2. The overall reconstruction time $t_{\text{Assembly}} + t_{\text{Solve}}$ does not admit general statements about the performance, it does however allow a direct comparison of algorithms under equal environmental conditions. All ground truth depth maps and their derivatives arose from analytical formulas, except for the synthetic vase, which exhibits infinite gradients unless these are estimated numerically. In fact, we assume that there are no occlusions because we believe that they should be avoided in the measurement process anyway. Recovering information that is just not present in the data by the numerical reconstruction algorithm seems at least somewhat questionable. Still, the isogeometric discretization principle naturally extends to discontinuous depth maps. Only energy functional and optimization method have to be adapted. Section 2.3 of the technical report attached as supplemental material outlines how to proceed.

The outcome of additional experiments back up the claims of Section 2.2: Growing computational demands of the FDM can only be met by naive downsampling of the data. Consider on the other hand the paraboloid in Figure 6, which is known to have degree 2. Thus, in theory, one should be able to express it by only a few quadratic B-splines. Consequently, the number of DOFs for computation by IGA can be reduced to the minimum *without ever affecting the data*. The p/s-diagram in Figure 7 substantiates this further: Since the depth map of the sinusoidal wave can be decomposed into an infinite power series, the reconstruction quality can only grow with s and p. Furthermore, we investigated the influence of two common disturbances of $\boldsymbol{g}_{\text{m}}$ upon the quality of reconstruction, Gaussian noise and clutter, see Figure 3(a). As predicted earlier, opposed to the FDM, our method turns out to be highly robust with respect to both, provided that scale and polynomial degree have been set appropriately. How this can be done in tune with the knowledge about the scene that the image portrays is beyond the scope of this paper. Also, it is fair to say in regard to Figure 3, that competetive results are not impossible to obtain by other methods. The point is that this would

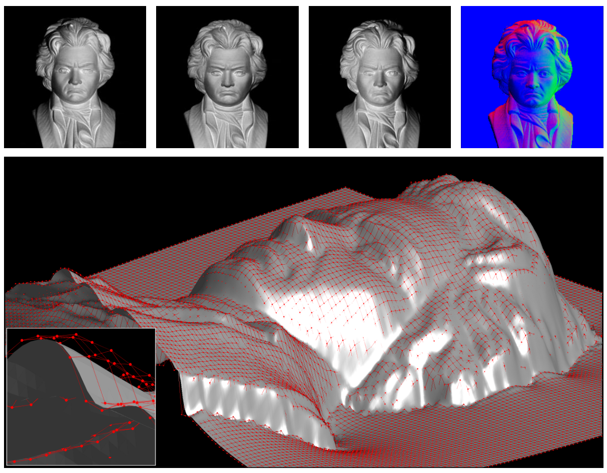


Figure 9. The Beethoven data set (first three images in the top row) was taken from [17]. Gradients (top right) were obtained exploiting the photometric stereo effect and then integrated into a quadratic B-spline patch (bottom). The magnified excerpt shows that the control point lattice is in general not interpolating.

require more or less expensive prefiltering, whereby again, the data cannot remain untouched. Note that the curves in Figure 8 at $\sigma = 0$ and $\sigma = 2.0$ are in reverse order with respect to the error measure E. A simple explanation is that, while for flawless data, higher-order polynomials lead to better approximations, the converse is true in the presence of high-frequency noise which will also be integrated at finer resolutions, cf. top row of Figure 3(c). The result of a final experiment on real-world data is depicted in Figure 9.

4. Open problems

In this paper, we identified isogeometric analysis a general strategy to deal with Euler-Lagrange equations in computer vision and demonstrated its advantages at hand of the depth-from-gradients problem. Although a very promising numerical tool, we would like to point out in closing two particular limitations it entails: First, the construction of a multivariate basis from tensor products of univariate B-splines prohibits a purely local refinement, for inserting a knot in one direction always alters all orthogonal translates of the modified basis function. So-called *T-splines* have been developed to address this issue and found their way into commercially available CAD software. Second, in order to remove the assumption that the unknown surface is parametrizable by a depth map and thus warrant reconstruction in scene space, we have to be able to conduct IGA on surfaces that consist not of one but rather a whole series of patches, not necessarily watertight. This problem has not seen a satisfactory solution, yet, but in our opinion, mathematical techniques such as domain decomposition are quite promising in this respect.

Acknowledgements

This work was conducted while the first author worked at King Abdullah University of Science an Technology. The research of T. Mörwald leading to these results has received funding from the European Communitys Seventh Framework Programme [FP7/2007-2013] under grant agreement No. 215181, CogX.

References

[1] J. Balzer. A Gauss-Newton Method for the Integration of Spatial Normal Fields in Shape Space. *J. Math. Imaging Vis.* In press. 4
[2] A. Delaunoy and E. Prados. Gradient Flows for Optimizing Triangular Mesh-based Surfaces: Applications to 3D Reconstruction Problems Dealing withVisibility. *Int. J. Comput. Vision*, 95:100–123, 2011. 4
[3] J.-D. Durou, J.-F. Aujol, and F. Courteille. Integrating the Normal Field of a Surface in the Presence of Discontinuities. *Proc. EMMCVPR*, pages 261–273, 2009. 4
[4] J.-D. Durou and F. Courteille. Integration of a Normal Field without Boundary Condition. *Proc. PACV*, 1, 2007. 4, 6
[5] T. Elguedj, J. Réthoré, and A. Buteri. Isogeometric analysis for strain field measurements. *Comput. Methods Appl. Mech. Engrg.*, 200:40–56, 2010. 2
[6] S. Ettl, J. Kaminski, M. Knauer, and G. Häusler. Shape reconstruction from gradient data. *Appl. Optics*, 47(12):2091–2097, 2008. 4
[7] P. Felzenszwalb and R. Zabih. Dynamic Programming and Graph Algorithms in Computer Vision. *IEEE T. Pattern Anal.*, 33:721–740, 2011. 1
[8] I. Gelfand and S. Fomin. *Calculus of Variations*. Dover, 2003. 1
[9] M. Harker and P. O'Leary. Least squares surface reconstruction from gradients: Direct algebraic methods with spectral, Tikhonov, and constrained regularization. *Proc. CVPR*, 1:2529–2536, 2011. 4
[10] R. Hicks. Designing a mirror to realize a given projection. *J. Opt. Soc. Am. A*, 22(2):323–330, 2005. 4
[11] B. Horn. Height and gradient from shading. *Int. J. Comput. Vision*, 5(1):37–75, 1999. 4
[12] T. Hughes, J. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Comput. Methods Appl. Mech. Engrg.*, 194(39-41):4135 – 4195, 2005. 2
[13] P. Kovesi. Shapelets correlated with surface normals produce surfaces. *Proc. ICCV*, 2:994–1001, 2005. 4
[14] K. Morton and D. Mayers. *Numerical Solution of Partial Differential Equations*. Oxord University Press, 2005. 1, 2
[15] H.-S. Ng, T.-P. Wu, and C.-K. Tang. Surface-from-gradients without discrete integrability enforcement: A gaussian kernel approach. *IEEE T. Pattern Anal.*, 32:2085–2099, 2010. 4, 6
[16] L. Piegl and W. Tiller. *The NURBS Book*. Springer, 1996. 5
[17] W. Snyder. NC State University Image Analysis Laboratory Database, 2002. 8

Fitting B-Spline Curves to Complex Shaped Boundaries

Anonymous DAGM-OAGM submission

Paper ID 22

Abstract. Finding the boundary of some region and computing a curve to approximate it best is a common task in computer vision and image processing. This paper describes an approach of fitting B-Splines to 2D point-clouds for robustly finding the boundary of complex shapes. The problems of common B-Spline fitting methods are discussed. New techniques to overcome this problems, namely the *Asymmetric Distance Minimization*, *Error-Adaptive Knot Insertion* and *Concavity Filling* are applied and considered as the main contribution of our work. We will show how our fitting approach leads to satisfying solutions, even by employing a generic initialization scheme and without knowing the required degree of freedom. All improvements are discussed and demonstrated on difficult problems from real sensor data.

1 Introduction

A common task in computer vision and image processing is to find the boundary of some region. Often it is important to represent the boundary with a continuous model, or find some smooth approximation of it. The huge amount of literature testify the significance of the topic, yet it seems there is still no generic approach that can handle all the problems arising when trying to solve it. Of course on one hand this is due to the fact that the solution strongly depends on the demands of the user, but on the other hand a lot of work is still done manually or by some non-generic preprocessing.

We aim to automatically find the outer boundary of segmented image patches with noise and clutter and without any prior information of the topology and complexity of the segments. The points to be fitted are computed by edge detection on the labeled image. (see Figure 1(b))

We want to extend existing curve fitting techniques to make them more robust against clutter, noise and discontinuities typically occurring in real world sensor data. Therefore we present a novel solution for approximating unorganized data points with a B-Spline curve and consider the following as main contribution of our work:

- *Asymmetric Distance Minimization (ADM)* that can handle clutter inside while smoothing noise at the boundary.
- *Error-Adaptive Knot Insertion (EAKI)* that automatically accounts for complex shapes and allows for a trivial initialization of the B-Spline curve.

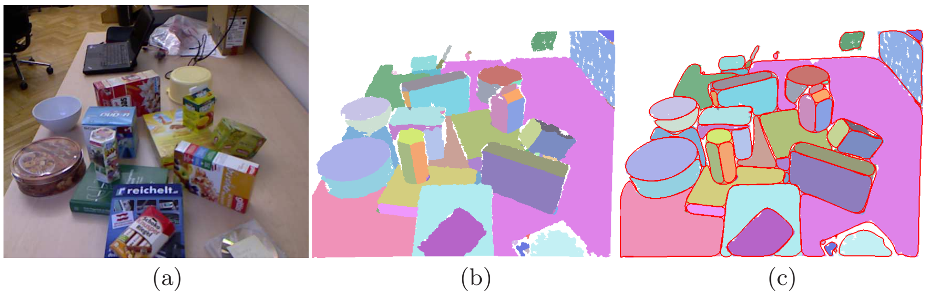


Fig. 1. Finding the contour of continuous regions (surface patches) in an image (a). The segmentation was achieved by taking advantage of depth information (b). The boundary of the segments are computed with our approach (c).

– *Concavity Filling (CF)* to handle special situations like sharp turns and deep, narrow concavities.

The remainder of this paper is organized as follows. In Section 2 we provide a brief outline of the related work followed by a more detailed description of three established methods in the beginning of Section 3. Further we will ascertain the problems of fitting B-Splines and introduce our solutions and discuss the behavior for each of them. In Section 4 we will show the advantage of our approach on real data, namely the Berkeley Image Segmentation Results [7], to illustrate the significance. Finally we will summarize the work presented and point out weaknesses, disadvantages and open problems.

2 Related Work

A review of the massive body of literature on B-Spline curve fitting would go far beyond the scope of this paper. We briefly give an overview of relevant work and afterwards point out the most common approaches to which we want to apply our methods.

One of the most fundamental summary on B-Splines and least-squares fitting to point-clouds was done in the well know book of Piegl et. al. [8] where they minimize a functional like Equation (2). This method was carefully investigated in [9] where they especially focus on the squared distance function and their approximants used for least-squares fitting. In [12, 13, 2] new point-curve distance functions are introduced to improve the convergence rate and robustness.

In [14] Yang et. al. propose an active implicit B-Spline model and find the zero set of a bivariate tensor-product B-Spline function using the trust region algorithm [10]. Fitting B-Spline curves to point-clouds in the presence of obstacles is introduced in [3, 4], where they minimize the functional (2) subject to an inequality constraint.

Hu et. al. [6] present a method where they take advantage of both algebraic and geometric distance minimization and therefore avoid additional constraints.

Often it is necessary to modify an existing curve fitting method to apply to a problem with certain characteristics, such as noise, outliers, unknown degree of freedom (DOF) and so forth [5, 15, 1].

Our approach extends the *Squared Distance Minimization* (SDM) of [13], i.e. we are modifying their error term for the functional to be minimized. Further we want to overcome the problem of specifying the degree of freedom manually and add control points and knots depending on the error of the curve.

3 B-Spline Curve Fitting

Fitting a curve to a point-cloud is the problem of finding a good approximating or interpolating curve for a set of points $\mathbf{p}_k$, $k = 1, 2, \ldots, n$. Common representations for curves are B-Splines, often used in computer graphics, CAD/CAM, computer vision and image processing. We assume the set $\mathbf{p}_k$ to consist of unorganized, scattered data points with considerable non-uniform distributed noise and heavy clutter.

A complete description of B-Splines and their mathematical formulation would go far beyond the scope of this paper, so we want to point the interested reader to the fundamental book [8]. Let us start with the definition of a B-Spline curve as defined in Chapter 3.2 of [8]:

$$\mathbf{C}(t) = \sum_{i=0}^{m} N_i(t)\mathbf{B}_i \tag{1}$$

where N_i are the *basis functions*, $\mathbf{B}_i$ are called *control points* and t is the parameter of the curve. Please note, that for most fitting approaches the number of control points and knots are assumed to be known or given by the user. A common used formulation of fitting a B-Spline curve to a set of points $\mathbf{p}_k$ is to minimize the objective function

$$\begin{aligned} f &= \tfrac{1}{2}\textstyle\sum_{k=1}^{n} e_k + w_s f_s \\ e_k &= ||\mathbf{C}(t_k) - \mathbf{p}_k||^2 \end{aligned} \tag{2}$$

with respect to the control points $\mathbf{B}_i$. Typically a weighted smoothing term $w_s f_s$ is used to obtain a visual satisfying solution. As mentioned in Section 2, Wang et. al. [13] use an error term $e_{SD,k}$ to improve robustness and convergence. We've applied our approach to both e_k and $e_{SD,k}$ experiencing similar behavior as described in [13]. For the reminder of this paper we are referring to the improved error term $e_{SD,k}$.

Unfortunately we have experienced several drawbacks using the formulation above. As depicted in Figure 2 segmentations are often subject to heavy clutter and outliers at the boundary as well as inside. In this Section we will first introduce the *Asymmetric Distance Minimization* to treat clutter inside the boundary. Then we discuss the problem of unknown degree of freedom, and our solution the *Error-Adaptive Knot Insertion*. For especially hard situations like

Submitted to the Joint German/Austrian Pattern Recognition Symposium (DAGM-OAGM), 2012.
Provided as part of DR 2.5: Qualitative behaviour models, grasping of novel objects

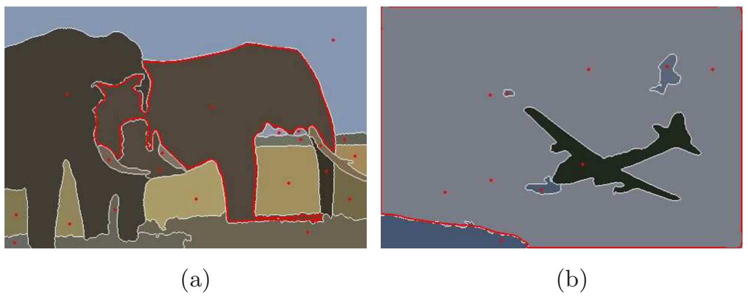


Fig. 2. *Berkeley Dataset:* Deep concavities, sharp turns and close neighboring boundaries (a). Clutter inside the boundary and noise at the boundary (b).

sharp turns and deep and narrow concavities we introduce the *Concavity Filling* method. Due to the robustness of our approach we are able to employ a trivial initialization method which we briefly describe at last in this Section.

3.1 Asymmetric Distance Minimization (ADM)

As shown in Figure 2(b) the image often contains points that do not belong to the boundary, which are located inside of it.

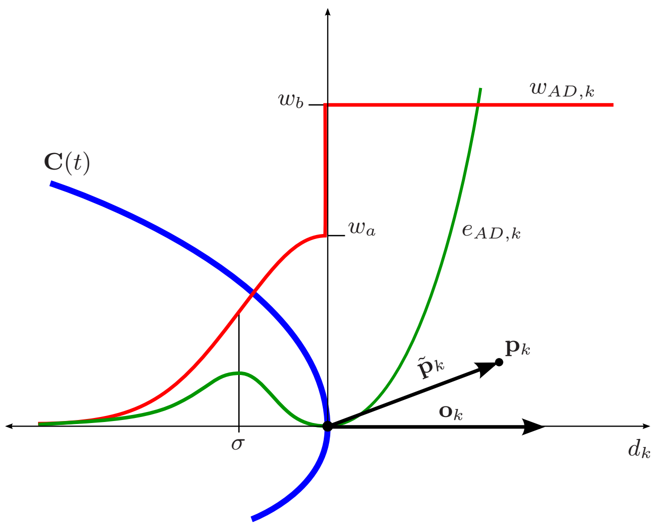


Fig. 3. *Asymmetric Distance:* Weighting function $w_{AD,k}$ (red) and distance term $e_{AD,k}$ (green) for fitting the point $\mathbf{p}_k$ attached to the footpoint $\mathbf{C}(t_k)$. Points outside the B-Spline curve $\mathbf{C}(t)$ (blue) are preferred to inside ones. Note that in general the outward normal vector $\mathbf{o}_k$ is not the negative vector of the second derivative of the curve $\mathbf{C}''(t_k)$.

Similar to [13] we introduce a new asymmetric distance (AD) term such that points that are inside the boundary are weighted less than points outside. Let $\tilde{\mathbf{p}}_k$ be the vector pointing from $\mathbf{C}(t_k)$ to $\mathbf{p}_k$ and $\mathbf{o}_k$ be the outward pointing unit normal vector of the curve at t_k. We define the asymmetric weighting function $w_{AD,k}$ as

$$w_{AD,k} = \begin{cases} w_a e^{-\frac{d_a^2}{2\sigma^2}} & \text{if } d_a < 0 \\ w_b & \text{if } d_a \geq 0 \end{cases} \tag{3}$$

where σ (called *connection width*) defines the width of the transition of the weighting function (see Figure 3) with respect to the signed distance

$$d_a = \tilde{\mathbf{p}}_k^T \mathbf{o}_k \tag{4}$$

The weighting function is multiplied by the distance term e_k, inducing our new asymmetric distance term:

$$e_{AD,k} = w_{AD,k} e_k \tag{5}$$

This forces the curve to the outer boundary points, and neglects points inside, which means that also points of concavities are not considered immediately. Fortunately the half bell-shaped function iteratively closes the gab between the curve and the data points. This is different to most of the other approaches (i.e. [8, 2, 13]) where all points are treated the same in a global sense. Figure 4 shows how easy it is to confuse typical fitting approaches.

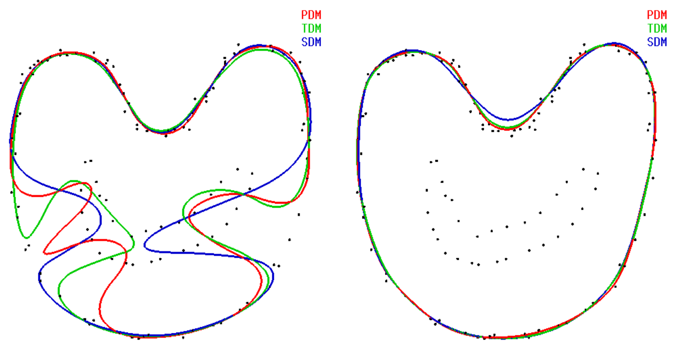


Fig. 4. *ADM:* Typical problem when adding some clutter inside the boundary of a dataset such as Figure 10 of [13] (left). Solving the problem using the asymmetric distance (AD) error term of our approach (right). Red: Point distance [8]. Green: Tangent distance [2]. Blue: Squared distance [13].

3.2 Error-Adaptive Knot Insertion (EAKI)

In real world application the degree of freedom of the boundary is usually unknown. Initialization is typically done by user input or by some simple estimation scheme. We want to introduce a new method that automatically adapts the DOF by iteratively inserting knots to the B-Spline curve at points where the error is above the accuracy specified by the user. This leads to a non-uniform distributed knot vector (i.e. control points), which approximates the point cloud with respect to the accuracy, both at regions of sharp turns and high peak-to-valley height as well as at smooth areas. In other words, control points are placed where they are needed, given a certain accuracy (see Figure 5).

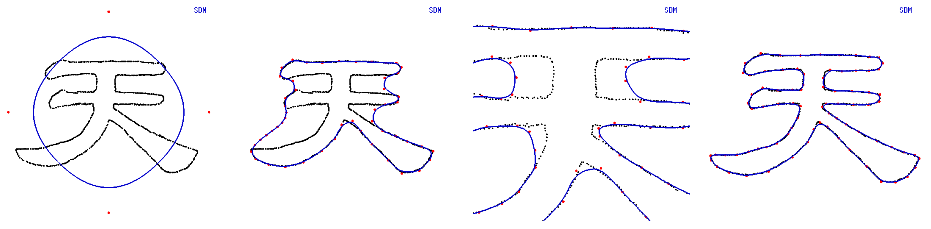


Fig. 5. *EAKI:* Control points (red) are iteratively inserted, automatically adapting the curve (blue) to the required degree of freedom of the outline of the Chinese character *tian*. From left to right: Initial curve, 10, 15 and 30 iterations with 4, 61, 73 and 82 control points respectively. Note the simple initialization and the iterative elevation of the DOF by knot insertion, while the AD term fastens the curve to the points.

During each fitting iteration we measure the distance from every curve point $\mathbf{C}(t_j)$ to the closest point of the point cloud, where t_j are the midpoints of two adjacent elements of the knot vector (see [8] for the definition of a knot vector). If the distance is exceeding the accuracy specified ε_a, a new knot is inserted at the curve parameter point t_j.

Note that at the beginning of the fitting iterations at almost every point t_j a new knot is inserted, since the curve is yet not close enough to the data points. To avoid this, one can add an additional condition such that knots are only inserted if the curve converged (i.e. when it does not change significantly from one iteration to the other).

3.3 Concavity Filling (CF)

Unfortunately noise at the boundary and sharp turns may stop the process of iteratively closing the boundary (Figure 6). This disadvantage is caused by the asymmetric distance minimization, which ignores data points that are inside the boundary and to far away to be considered.

Closest Point Strapping This leads to the idea of explicitly finding data points orthogonal to the boundary and strap the curve to these points. Starting

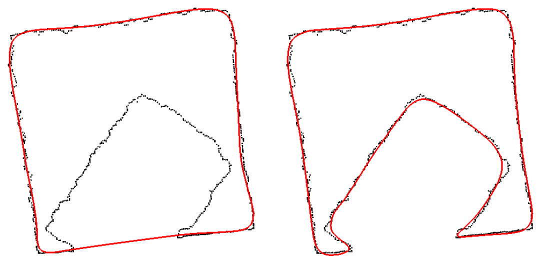


Fig. 6. *Sharp turns:* Without concavity filling methods (a). With *closest point strapping* and *concave smoothing* (b).

form Equation (2), we add the term $w_c f_c$ to the functional to be minimized,

$$\begin{aligned} f &= \tfrac{1}{2}\textstyle\sum_{k=1}^{n} e_{AD,k} + w_s f_s + w_c f_c \\ f_c &= \textstyle\sum_{j=1}^{m} ||\mathbf{C}(t_j) - \mathbf{p}_j||^2 dt \end{aligned} \tag{6}$$

where $\mathbf{p}_j$ is the closest point to the B-Spline curve at t_j. In a similar manner as in Section 3.2 we find the closest point for each midpoint t_j of two neighboring elements of the knot vector. The difference is that we are not using the Euclidean distance to find $\mathbf{p}_j$, since we are searching for points within the curve and close to the straight line normal to the curve. Therefore we define the distance as

$$d_o = \begin{cases} 0 & \text{if } |\tilde{\mathbf{p}}_j| = 0 \\ \infty, & \text{if } \mathbf{o}_j^T \tilde{\mathbf{p}}_j \geq 0,\, |\tilde{\mathbf{p}}_j| \neq 0 \\ \frac{\tilde{\mathbf{p}}_j^T \tilde{\mathbf{p}}_j}{|\mathbf{o}_j^T \tilde{\mathbf{p}}_j|}, & \text{if } \mathbf{o}_j^T \tilde{\mathbf{p}}_j < 0,\, |\tilde{\mathbf{p}}_j| \neq 0 \end{cases} \tag{7}$$

which results in iso-value curves as shown in in Figure 7(a). As we are minimizing Equation (6) the curve is strapped to data points $\mathbf{p}_j$ behind the noisy points. Once the curve is close enough ($d_o < \sigma$) to points in the neighborhood of $\mathbf{p}_j$ the closing effect continues.

Concave Smoothing Similar to other approaches [13] we employ a weighted smoothing term $w_s f_s$, that minimizes the curvature of the curve. Considering the problem described at the beginning of this section we define the term f_s such that it is possible to specify a desired concavity or convexity.

$$f_s = \int_{\Omega} ||\mathbf{C}''(t) + \gamma \mathbf{o}(t)||^2 dt \tag{8}$$

with $\mathbf{o}(t)$ being the outward pointing normal vector at t. In our case γ is usually negative to support filling of concavities. Note that the asymmetric weighting function prohibits concave smoothing at regions where enough boundary points are given (see Figure 7(b)). However, in most cases the combination of both solutions produces best results. Since we compute the outward normal vector $\mathbf{o}$ anyway the additional computational load is insignificant.

Submitted to the Joint German/Austrian Pattern Recognition Symposium (DAGM-OAGM), 2012.
Provided as part of DR 2.5: Qualitative behaviour models, grasping of novel objects

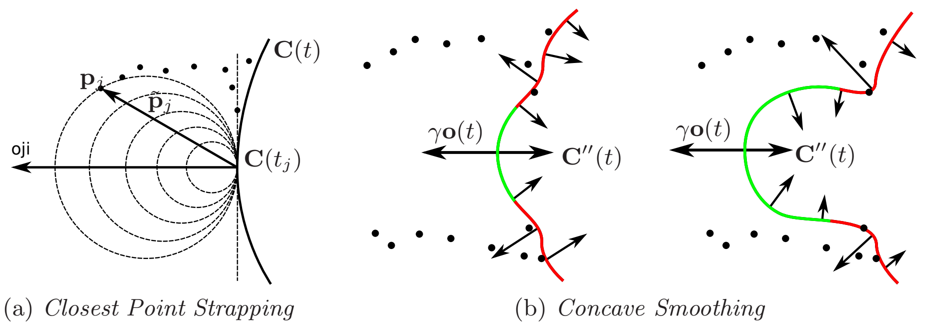


Fig. 7. *Filling concavities*
(a) Iso-value curves for finding the closest point to the curve at point t_j, with respect to the outward pointing normal vector $\mathbf{o}_j$.
(b) With γ in Equation (8) being negative the curve tends to bend inwards at regions where no supportive data points are given (green). Otherwise the data points avoid inward bending of the curve (red).

3.4 Initialization

For initialization we are simply calculating the bounding circle of the point cloud and set the 4 initial control points of the closed periodic B-Spline curve to lie on this circle while being shifted by $\pi/4$ from each other (see Figure 5 left).

4 Results

We employ our method in a generic way to a real world application, namely finding the boundary of segmented images such as Figure 2 and 8. Please note, that segmentation is not the focus of the paper and is therefore treated as given.

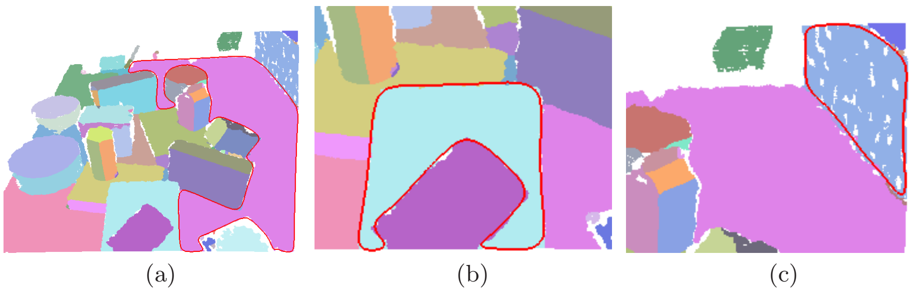


Fig. 8. *Results* of our approach when finding the contours of image segments. (a) Complex shape with sharp turns. (b) Sharp turns and deep concavity. (c) Clutter inside the segmented region. The result for all the segments in the image is shown in Figure 1.

4.1 Segmentation of Continuous Regions

For image segmentation we used the algorithm described in [11]. Please note the complex shape of the right side of the desk, the deep concavity of the blue book in the foreground, as well as the heavy clutter occurring at the right wall in Figure 8. To fit the boundaries of the image segments, the noise is estimated to adjust accuracy ε_a for EAKI and width σ for ADM (w_a and w_b are set to 1.0 and 0.5 for all our experiments, since their influence is vanishing).

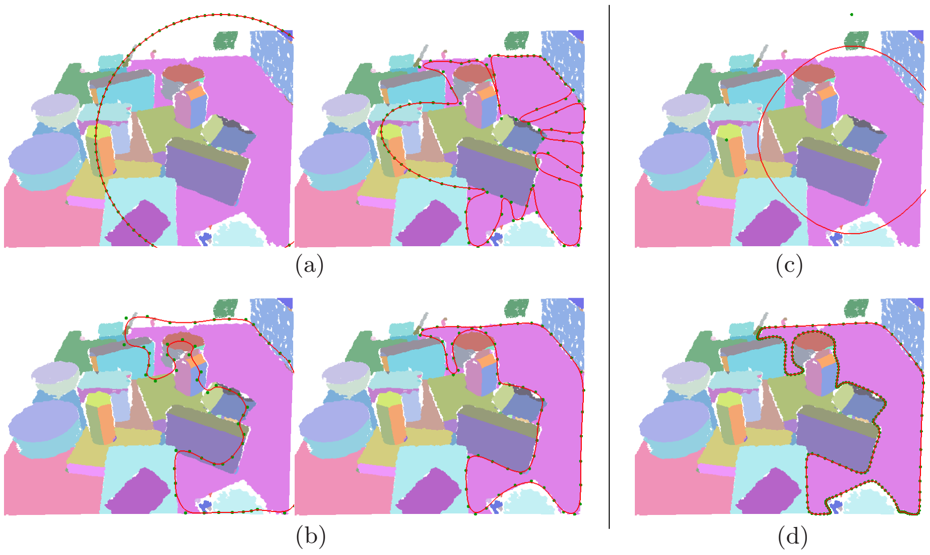


Fig. 9. *Comparison:* (a) SDM with initialization using the bounding circle and a sufficient number of control points. (b) SDM with manual initialization. Note that the accuracy at corners is quite low, due to a too small number of control points. Only if the shape is roughly estimated with a sufficient number of control points SDM produces similar results as our approach. (c) Our method with initialization as described in Section 3.4. (d) Our method after convergence. Note how at points of sharp turns additional control points are inserted by the EAKI.

4.2 Berkeley Segmentation Dataset

We've applied our approach to several segmented images[1] of the Berkeley Segmentation Dataset [7]. The results would exceed the page limit of this paper, so we want to refer to the supplementary videos. The parameters of our approach to produce those videos are not modified. Note the time until convergence with respect to complexity and size of the shape. The data points to be fitted are the border pixels of the respective area (i.e. having the same label) and not the edges separating two regions (white). This is why the curve seems to lie inside of it.

[1] `http://www.cs.berkeley.edu/%7Earbelaez/constrained/results.html`

5 Conclusion

In this paper new techniques for finding continuous representations of boundaries are introduced. We've applied our novel methods to three existing methods and we think that our contribution is similar applicable to other approaches. The parameters defined by the user are formulated in a natural way, like *smoothness*, *accuracy*, *concavity* and *connection width*.

Unfortunately, as mentioned in Section 4, the optimization of the parameters requires implicit knowledge of the sensor which might not be available. This makes it tricky to find an ideal parametrization for different kind of input data. However, we've experienced the algorithm to be very generic when dealing with the same kind of input data.

References

1. Barbieri, L., Bruno, F., Muzzupappa, M., Pernot, J.: Constrained fitting of B-spline curves based on the force density method. In: International Conference on Innovative Methods in Product Design. vol. 46 (2011)
2. Blake, A., Isard, M.: Active Contours, vol. 17. Springer (1998)
3. Flory, S.: Fitting curves and surfaces to point clouds in the presence of obstacles. Computer Aided Geometric Design 26(2), 192–202 (2009)
4. Flory, S., Hofer, M.: Constrained curve fitting on manifolds. Computer-Aided Design 40(1), 25–34 (2008)
5. Grove, O.: From CT to NURBS: Contour Fitting with B-spline Curves. Computer Aided Design And Applications 8(1), 3–21 (2011)
6. Hu, M., Feng, J., Zheng, J.: An additional branch free algebraic B-spline curve fitting method. The Visual Computer 26(6-8), 801–811 (2010)
7. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. Proceedings Eighth IEEE International Conference on Computer Vision ICCV 2001 2(July), 416–423 (2001)
8. Piegl, L., Tiller, W.: The NURBS book. Monographs in visual communication, Springer (1996)
9. Pottmann, H., Hofer, M.: Geometry of the Squared Distance Function to Curves and Surfaces. Visualization and mathematics III (90), 223–244 (2003)
10. Powell, M.: On the global convergence of trust region algorithms for unconstrained optimization. Math. Prog. 29, 297–303 (1984)
11. Richtsfeld, A., Prankl, J., Balzer, J., Zillich, M.: Towards Scene Understanding Object Segmentation Using RGBD-Images. users.acin.tuwien.ac.at (2012)
12. Wang, W., Pottmann, H., Liu, Y.: Fitting B-Spline Curves to Point Clouds by Squared Distance Minimization. ACM TOG p. 41 (2004)
13. Wang, W., Pottmann, H., Liu, Y.: Fitting B-spline curves to point clouds by curvature-based squared distance minimization. ACM Transactions on Graphics 25(2), 214–238 (2006)
14. Yang, Z., Deng, J., Chen, F.: Fitting unorganized point clouds with active implicit B-spline curves. The Visual Computer 21(8-10), 831–839 (2005)
15. Zhaoa, X., Zhang, C., Yang, B., Li, P.: Adaptive knot placement using a GMM-based continuous optimization algorithm in B-spline curve approximation. Computer-Aided Design 43(6), 598–604 (2011)

IEEE International Conference on Robotics and Biomimetics (ROBIO), 2012.
Provided as part of DR 2.5: Qualitative behaviour models, grasping of novel objects

Self-Monitoring to Improve Robustness of 3D Object Tracking for Robotics

Thomas Mörwald, Michael Zillich, Johann Prankl and Markus Vincze

Automation and Control Institute, Vienna University of Technology, AT

***Abstract*— In robotics object tracking is needed to steer towards objects, check if grasping is successful, or investigate objects more closely by poking or handling them. While many 3D object tracking approaches have been proposed in the past, real world settings pose challenges such as automatically detecting tracking failure, real-time processing, and robustness to occlusion, illumination, and view point changes. This paper presents a 3D tracking system that is capable of overcoming these difficulties using a monocular camera. We present a method of Tracking-State-Detection (TSD) that takes advantage of commercial graphics processors to map textures onto object geometry, to learn textures online, and to recover object pose in real-time. Our system is able to handle 6 DOF object motion during changing lighting conditions, partial occlusion and motion blur while maintaining an accuracy of a few millimetres. Furthermore using TSD we are able to automatically detect occlusions or whether we lost track, and can then trigger a SIFT-based recognition system that is trained during tracking to recover the pose. Evaluations are presented in relation to ground truth pose data and examples present TSD on real-world scenes presented in video sequences.**

I. Introduction

Robotic object grasping requires to determine the object's pose and to track the object during the approach with sufficient accuracy (Figure 1, left). After grasping it is necessary to confirm if the grasp was successful and is stable [1], i.e. the object moves together with the end effector without slipping (Figure 1, right). For learning about physical behaviour of objects as in [2–5] the robot has to observe its motion (Figure 1, middle). Again accuracy of tracking but also detecting whether the tracked object has been lost, for example after toppling over, are important to decide whether a certain trajectory should be taken into account for learning. For a robot operating in a complex unpredictable environment, the challenge is to develop a tracking method that is robust to different lighting conditions, partial occlusion, and motion blur.

Today this is achieved best by model-based tracking of objects and numerous solutions using different feature types, models and mathematical frameworks have been developed, where the today's computational power allows for several real-time solutions. However, practical application of these methods is often limited for various reasons. For example, some methods report good results, without giving actual numbers on accuracy [6–9]. Others are capable of handling partial occlusion or changing lighting conditions [9–12] but can not differentiate between deteriorating tracking conditions and lost tracks. Some methods are restricted in their degrees of freedom, e.g. 140 degrees of rotation as in [11], require off-line learning [10] or are limited to either textured [13, 14] or low-textured objects [15]. Also recovery from lost tracks is rarely handled with a few exceptions [13, 14], which are tracking-by-detection approaches.

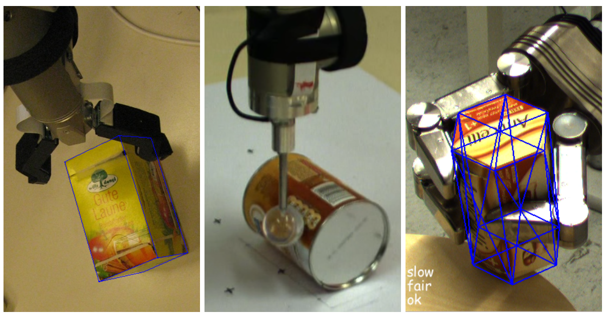


Fig. 1. Tracking for robotic applications. Left: grasping; middle: learning about object motion; right: grasp stability.

Another requirement in robotics is computational efficiency to react to observed situations in time. Consider again the grasping scenario, where we want to use visual servoing to adapt the grasping movement on-line. Hence, we require real-time performance, i.e. processing time within the frame rate of a typical camera (25-50 Hz).

To meet all these requirements we propose to tackle the core problem of detecting tracking failure and take advantage of this supervisory knowledge to achieve automatic object tracking using texture mapping, pose recovery and online learning. Hence, the approach is based on the following methods:

- *Tracking-State-Detection (TSD):* To know whether we are tracking correctly, whether the object is occluded or whether we lost track we employ our novel TSD method. The knowledge of the tracking state, including speed and confidence of tracking, allows for triggering online learning or pose recovery.
- *Texture mapping:* We take advantage of texture, if available, to boost robustness of tracking, especially in cluttered scenes.
- *Pose recovery:* To initialise tracking and recover lost tracks we use distinctive features placed on the object's surface.
- *Online learning:* We learn these feature points and surface texture of the object automatically while tracking.

Our main contribution is the TSD, since it is the key to use the other methods automatically.

The paper proceeds as follows: In Section III-A we formulate tracking as particle filtering using a modified version of the Sequential Importance Resampling (SIR) filter and show how to draw observations by projecting the model into image space. Section III-B describes how to evaluate the particle weights from observations. Section III-C introduces TSD to give evidence of the current tracking quality, speed and whether tracking has lost the object. In Section III-D we show how surface texture of a tracked object can be captured online from the camera image. Section IV briefly explains what methods we use for initialisation and re-detection of the object. In Section V we evaluate our approach with respect to the requirements established above.

Additionally to the results presented in the paper we provide a video in Section V-E to demonstrate robustness and especially our novel Tracking-State-Detection.

II. Related Work

Tracking the pose of an object by analysing a stream of TV images in real-time goes back to the early eighties [16, 17]. One of the first successful approaches of tracking objects based on edges was the RAPiD system [18]. It used points on model edges and searched for corresponding image edges the edge gradient. Subsequent approaches aimed at improving robustness in tough real-world scenarios [6, 7, 10, 19, 20]. Approaches based on globally matching model primitives with primitives extracted from the camera image [21–25] have been used for applications such as robot and car tracking, but were later replaced by improved versions of the RAPiD type.

[9] also use edges and textures for tracking. Their approach extracts point features from surface texture and use them together with edges to calculate object pose. This turns out to be very fast as well as robust against occlusion. Our approach not only uses patches but the whole texture, which usually lets the pose converge very quickly to the accurate pose. Since the algorithm runs on the GPU, it is as fast as the method in [9]. The work presented in [15] uses edge features to track but does not take into account texture information. This makes it less robust against occlusion. Since the search area in that approach is very small, it is also less robust against fast movement and gets caught in local minima.

More recent approaches aim to solve most of the problems of tracking, such as [12] where the authors are matching the camera image with pre-trained keyframes and then minimizing the squared distance of feature points taking into account neighbouring frames. The approach described in [11] uses a modified version of the Active Appearance Model which allows for partial and self occlusion of the objects and for high accuracy and precision. In [26] the authors minimize the optical flow resulting from the projection of a textured model and the camera image. To compensate for shadows and changing lighting they apply an illumination normalization technique.

In [27] the authors introduce real-time tracking to robotic manipulation. They are using the method proposed in [28], where they project the CAD model into image space, and try to minimize a cost functional for the distance to image edges found along the gradients of the edges of the model. The work presented in [29] describes an approach for real-time visual servoing using a binocular camera setup to estimate the pose by triangulating a set of feature points. As in our approach [13] takes advantage of robust Monte Carlo particle filtering to determine the pose of the camera with respect to SIFT features, which are localized in 3D using epipolar geometry.

Missing in all methods is to detect when tracking fails rather than reporting tracking trapped in a local optimum. The proposed TSD proposes to solve this and we develop the approach to make it work automatically.

III. Tracking

The work in this paper identifies the object by using colour and edge information from shape and texture. We project a model, typically consisting of triangles or quads with attached texture, into image space and compare it against the camera image. The pose is estimated using a modified version of the Sequential Importance Resampling (SIR) particle filter [30]. Image processing methods such as Gaussian smoothing and edge extraction as well as pixel-wise comparison of the projected model is accelerated using a typical graphics processing unit (GPU). We first introduce pose estimation and the measure to obtain confidence values from the image data before we explain TSD.

A. Pose estimation

Visual observation of the trajectory of the object is the problem of finding the transformations $\mathbf{T}_t$ given a sequence of images I_t, sampled over the time $t = [1 \ldots t_e]$. The transformations $\mathbf{T}_t$ are represented as

$$\begin{aligned} \mathbf{T}_t(\mathbf{x}_t) &= \begin{bmatrix} \mathbf{R}_t & \mathbf{p}_t \\ \mathbf{0} & 1 \end{bmatrix} \\ \mathbf{R}_t &= \mathbf{R}_t(\alpha, \beta, \gamma) \\ \mathbf{p}_t &= \mathbf{p}_t(x, y, z) \end{aligned}$$

where $\mathbf{R}_t(\alpha, \beta, \gamma)^T$ are rotation matrices and $\mathbf{p}_t = [x, y, z]^T$ translations respectively. This results in a state vector $\mathbf{x}_t = [x, y, z, \alpha, \beta, \gamma]^T$ of 6 DOF. Note that we actually use quaternions to avoid the problems of rotations in Euclidean space.

A particle filter, such as the SIR (Sequential Importance Resampling), explained in [31] and more detailed in [30], estimates the current state $\mathbf{x}_t$ based on the previous state $\mathbf{x}_{t-1}$ and the current observation $\mathbf{y}_t$. Starting from the Bootstrap Filter in [30], Algorithm 1 describes our modified version.

The first modification lies in in step 2a, where we adjust system noise Ω according to the confidence of the previous tracking step c_{t-1}. This means that as the confidence of the particles increases, their degree of distribution decreases, leading to faster convergence and less jitter. Note that from Equation (1) it follows that we do not use a physical motion model. Given the requirements for tracking accuracy and speed for a typical table top scenario we chose a basic

Algorithm 1 Bootstrap Filter, modified with respect to importance sampling.

```
1) Initialisation
    a) For $i = 1,\dots,N$, sample $\mathbf{x}_0^i \sim p(\mathbf{x}_0)$ and set $t = 1$.
2) Importance sampling
    a) For $i = 1,\dots,N$, sample $\tilde{\mathbf{x}}_t^i \sim p(\mathbf{x}_t|\mathbf{x}_{t-1}^i, c_{t-1})$ with
        $$\begin{array}{rcl} p(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i, c_{t-1}) & \sim & \Omega(\mathbf{x}_{t-1}, \sigma_{t-1}^2) \\ \sigma_t & = & (1 - c_t)\sigma_0 \end{array} \quad (1)$$
    b) For $i = 1,\dots,N$ of $\tilde{\mathbf{x}}_t^i$, evaluate the confidences
        $$c_t^i \sim p(\mathbf{y}_t|\tilde{\mathbf{x}}_t^i) \quad (2)$$
       using Equation (6).
    c) Normalize the confidence values for the importance weights
        $$w_t^i = \frac{c_t^i}{\sum_{i=0}^N c_t^i} \quad (3)$$
3) Selection step
    a) Resample with replacement $N$ particles $\mathbf{x}_t^i$ from the set $\tilde{\mathbf{x}}_t^i$ according to the importance weights.
    b) Set $t = t + 1$ and go to step 2
```

standard deviation $\sigma_{0,p}$ of 0.03 m for the translational and $\sigma_{0,\theta} = 0.5$ rad for the rotational degrees of freedom.

The second modification, as proposed already in our previous works [32] and [33], is to use iterative particle filtering for increased responsiveness to rapid pose changes. This means that we perform steps 2 and 3 of Algorithm 1 several times on the same image. Figure 2 shows the improvement over conventional particle filtering when using $k = 8$ iterations with $N = 100$ particles each vs. 1 iteration with 800 particles. It can be seen that the iterative version follows the motion much faster.

To initialise the pose $\mathbf{x}_0$ we use the method described in Section IV.

B. Image Processing and Matching

At time-step t for each particle i, we project the geometric model of the object, described by vertices, faces and textures, into the image space using the transformation $\mathbf{T}^i$ and standard techniques of computer graphics such as perspective transformation and texture mapping. In image space we compute the edges of the model $\mathbf{g}_M^i$ and of the image captured by the camera $\mathbf{g}_I^i$.

For each point (u, v) on the model M in image space we can compute the deviation of the gradients by superimposing the projected model over the image. The match m^i of a particle is defined as the sum of the differences of the gradients, and s_i is a normalising constant given by the sum

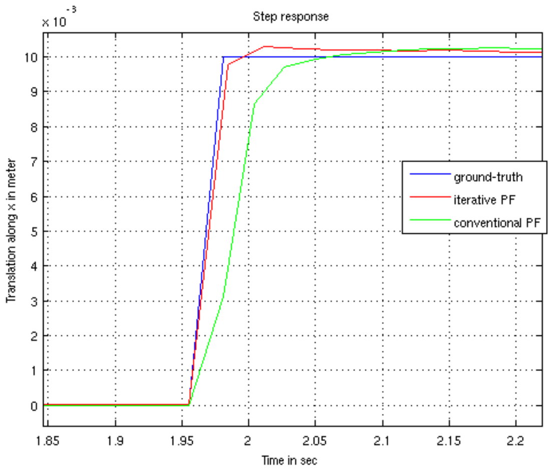


Fig. 2. Step response (1 cm) of conventional and iterative particle filtering using the same amount of particles within one frame, 1x800 and 8x100 respectively.

over all model gradients.

$$\begin{array}{rcl} m^i & = & \sum_{(u,v)\in M} |\mathbf{g}_M^i(u,v) - \mathbf{g}_I^i(u,v)| \\ s^i & = & \sum_{(u,v)\in M} |\mathbf{g}_M^i(u,v)| \end{array} \quad (4)$$

Instead of computing the difference of gradients, the difference of the colour with respect to the hue in HSV (Hue, Saturation, Value) colour space is used:

$$\begin{array}{rcl} m^i & = & \sum_{(u,v)\in M} |h_M^i(u,v) - h_I^i(u,v)| \\ s^i & = & \sum_{(u,v)\in M} |h_M^i(u,v)| \end{array} \quad (5)$$

where h_M^i and h_I^i are the hue values of the projected model and the image respectively. The advantage of using colour based tracking is increased robustness against edge based clutter. Of course it is less robust against changing lighting but the combination of both kinds of cues can significantly improve the overall performance.

We now define the confidence c^i of a particle $\mathbf{x}^i$ as

$$c^i = \frac{1}{2}\left(\frac{m^i}{s^i} + \frac{m^i}{\frac{1}{N}\sum_{j=1}^N s^j}\right) \quad (6)$$

where the first term is simply the match normalised with respect to s_i per particle and the second term is normalised with respect to the mean over all particles, de-weighting particles with a low number of pixels. This prevents the system from getting stuck in poses with a small number of pixels.

The overall confidence of the current observation t is then calculated by simply taking the mean of the confidences

$$c_t = \frac{1}{N}\sum_{i=1}^N c^i \quad (7)$$

C. Tracking-State-Detection (TSD)

As outlined above observing the current state of the tracker is important for assessing the validity of the output as well as allowing to trigger recovery from lost tracks. TSD is a mechanism that gives evidence of tracking speed, quality and overall state in a qualitative and quantitative manner.

1) Speed: The velocity is calculated as the first derivative of the translation $\mathbf{p}_t$ and rotation $\boldsymbol{\theta}_t$ of the pose respectively.

$$\begin{aligned}\dot{\mathbf{p}}_t &= \left[\frac{dx}{dt}, \frac{dy}{dt}, \frac{dz}{dt}\right] \\ \dot{\boldsymbol{\theta}}_t &= \left[\frac{d\alpha}{dt}, \frac{d\beta}{dt}, \frac{d\gamma}{dt}\right]\end{aligned} \tag{8}$$

We first apply a low-pass filter to remove noise and then normalise $\dot{\mathbf{p}}_t$ and $\dot{\boldsymbol{\theta}}_t$

$$\begin{aligned}\tilde{v} &= \max(v, \omega) \\ v &= \frac{1}{fk\sigma_{0,p}} \mathrm{lpf}(|\dot{\mathbf{p}}_t|) \\ \omega &= \frac{1}{fk\sigma_{0,\theta}} \mathrm{lpf}(|\dot{\boldsymbol{\theta}}_t|)\end{aligned} \tag{9}$$

where $fk\sigma_0$ is the maximum velocity the particle filter allows with respect to frame rate f, the number of iterations k and the maximum possible standard deviation $\sigma_t = \sigma_0$. Then we qualify the output by applying thresholds c_{v*} that indicate whether the object is *still* or moving *slow* or *fast* ($v < c_{v1} = 0.01$, $c_{v1} \leq v \leq c_{v2} = 0.1$ and $v > c_{v2}$ respectively).

2) Quality: To give a statement about the quality of the current pose we use Equation (7) which corresponds to the match of a pose hypothesis to the image evidence. Again we classify this measure to obtain qualitative statements by applying thresholds to distinguish if tracking is *good*, *fair* or *bad* ($c_t > c_{q1} = 0.5$, $c_{q1} \geq c_t \geq c_{q2} = 0.3$ and $c_t < c_{q2}$ respectively).

3) State: We decide on the overall tracking state *occluded*, *lost* or tracked *ok* based on confidence and speed, modelling the fact that confidence can decrease as a result of occlusion or motion. To this end we introduce a visibility flag $b_{visible}$ by comparing confidence against a dynamic threshold c_d derived from the current speed.

$$\begin{aligned}b_{visible} &= (\mathrm{lpf}(\max(c_d - c_t, 0)) \leq c_{th,lost}) \\ c_d &= \max(c_{max} - \tilde{v}, c_{min})\end{aligned} \tag{10}$$

where $c_{max} = 0.5$ and $c_{min} = 0.3$ define the range of c_d and $c_{th,lost} = 0.1$ defines the limit to be reached to declare an object to be lost or occluded. We now define an object to be not visible if the low-pass filtered confidence is low with respect to c_d. This dynamic threshold compensates for low confidence during moderate movement. Table I shows how we decide on the tracking state$_t$, based on visibility, speed, quality and previous tracking state$_{t-1}$. ($\neg$ means logical not)

TABLE I
DECISION ON TRACKING STATE

$b_{visible}$	speed	quality	state$_{t-1}$	state$_t$
false	still	-	ok	occluded
false	$\neg$still	-	ok	lost
false	$\neg$still	-	occluded	lost
true	still	good	occluded	ok
true	still	good	lost	ok

If the tracked pose does not move (speed = *still*) and $b_{visible}$ = *false*, the assumption is that the object is temporarily occluded. If the tracked pose however does move while $b_{visible}$ = *false*, this means the tracker drifts off, chasing a wrong local maximum. This follows from the fact that the tracker essentially always follows the local maximum. This definition does not allow detecting occlusion while the object is moving.

To recover from the state where the object is occluded or lost, speed and quality have to be *still* and *good* respectively.

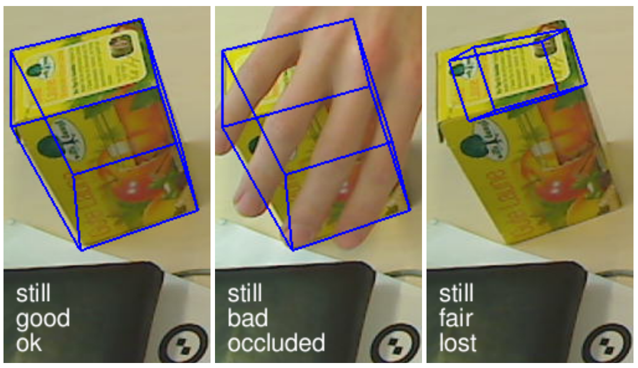


Fig. 3. Tracking-State-Detection: From left to right: ok, occluded and lost tracking.

Note that as we normalise velocity and the confidence the above thresholds apply to a large range of objects and situations.

Figure 4 shows the different values of Equation (10). Between $t = 1.0$ and $t = 2.0$ the object moves, with the confidence going down, but the tracker does not lose it. At about $t = 4.5$ we removed the object from the field of view which was detected at $t = 6.3$. Note how the confidence rises immediately after the object was removed, because the particle filter converged on a false local maximum. This suggests that confidence alone is a bad indicator for successful tracking. Furthermore it shows the importance of a dynamic threshold c_d. At the peak at $t = 1.5$ where the object moves without being lost the confidence is equally low compared to $t = 5$. However, only in the latter case tracking really failed.

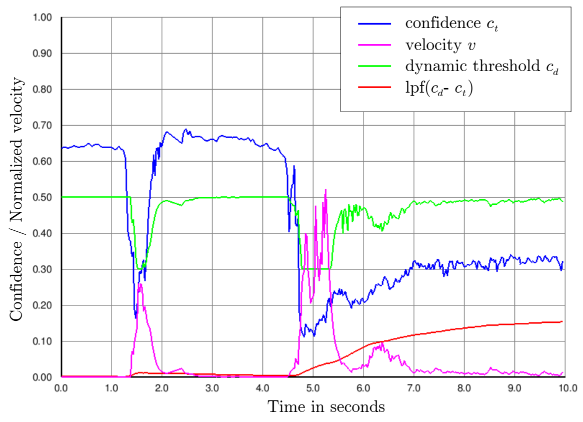


Fig. 4. Tracking-State-Detection: The peak on the left indicates movement without loosing the object, whereas on the right side the tracker detects that it lost the object as the red line rises above the lost-threshold $c_{lost} = 0.1$.

D. Texture Mapping

Tracking is based on a CAD model which (initially) does not include surface texture. This is sufficient for non-textured objects, where all we can observe are edges resulting from occlusion and surface discontinuity. For textured objects the additional edges provided by texture on surfaces significantly improve performance and especially robustness. The question is how to get the texture on the model faces. One possibility is to use a 3D editor to generate the model together with the texture mapping. However, we found that the textures mapped onto the geometry often do not align properly when compared to the real object. So we instead propose a mechanism that grabs the texture from the live camera image. Starting from tracking the wire-frame of the CAD model, culling away occluded edges, we successively capture the colour map from the camera and store it as texture together with the projection matrix that maps a point in model space (i.e. vertices) to image space (i.e. pixel coordinates of the texture). Mapping can be done either manually or automatically.

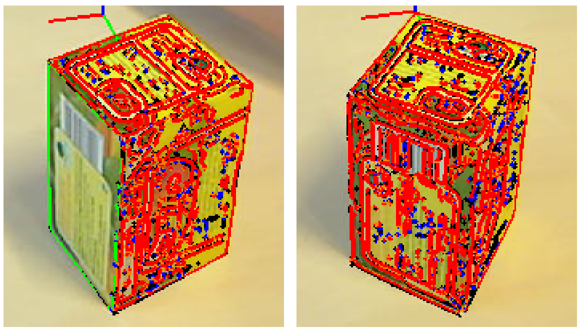


Fig. 5. Successively learning the texture of an object providing correct alignment. (red: matching edges from a textured face, green: matching edges from a non-textured face).

1) Manual mapping: To yield accurately aligned textures without blurring, distortions or occlusion it is best to move the object by hand to a proper position (correct alignment, small angle between view vector and negative face normal, no motion), and trigger texture capturing manually.

2) Automatic mapping: Of course it is more convenient to capture the textures automatically while moving the object, or by moving the camera around the object, e.g. by using a robotic arm with a camera attached to the end effector. To tell whether the object is in a good position with respect to the camera we calculate the angle between the normal of each face of the geometric model and the current view vector with respect to the centre of the object. We apply TSD (Section III-C) to check if the object is neither moving nor occluded and if tracking quality is good.

3) Capture the Texture: Now that we obtained a good pose for a specific face we can copy the image from the camera, cut out the respective image region and generate the UV-coordinates for the vertices. We project the 3D coordinates from model to image space (u, v) using the transformation $\mathbf{T}$ provided by the tracker and the camera projection. Then we compute the bounding rectangle (BR) for all the vertices of the face and re-scale the UV-coordinates to the range $[0 \ldots 1]$ with respect to the BR and store this area of the image.

IV. Initialisation and Recovery

Object detection is used for pose initialisation of learned models and also triggered if the object is lost during tracking as defined by the TSD.

While edges are well suited for fast tracking we use highly discriminating SIFT features for detection, following a fairly standard learning and recognition scheme. During the learning phase SIFT features (again we use a GPU implementation [34]) are detected in keyframes and mapped to the surface model using the known 3D pose from the tracker. SIFT features falling outside the object boundary are discarded. Keyframes are indicated either manually by button press or automatically using the TSD as described in Section III-D. To speed up recognition SIFT features are represented using a codebook (one per object). SIFT descriptors are clustered using an incremental mean-shift procedure and each 3D location on the object surface is assigned to the according codebook entry.

In the recognition phase SIFT features are detected in the current image and matched to the codebook. To robustly estimate the 6D object pose we use the OpenCV pose estimation procedure in a RANSAC [35] scheme. More details and experimental results are given in [36].

V. Results

All experiments were performed on a PC with an Intel Core 2 Quad (Q6600, 2.4 GHz) CPU, a NVIDIA GeForce GTX 285 GPU and a Logitech Webcam Pro 9000 run at a resolution of 640x480 pixels. We evaluated the approach by using virtual rendered image sequences with known ground truth as well as live sequences where we obtain ground truth from a calibration pattern rigidly attached to the object.

A. Evaluation of the Tracking Error

For a measure of the error we used the scheme proposed in Section IV-B in [2], where a large number, $n = [1 \ldots N]$, of randomly chosen points $\mathbf{q}^{1,n}$ are rigidly attached to the object surface at the ground-truth pose and compared to the corresponding points of the tracked pose $\mathbf{q}^{2,n}$.

$$E_d = \frac{1}{N} \sum_{n=1}^{N} |\mathbf{q}_d^{2,n} - \mathbf{q}_d^{1,n}| \qquad (11)$$

with $d \in \{x, y, z\}$.

Before evaluating our method in terms of the above error metric, let us briefly consider the possible sources of errors in our system, such as errors from calibration, geometric modelling, image quantisation and finally the tracking algorithm itself. Concretely we identify the following sources of errors:

- *Mechanical Error:* Positioning the calibration pattern rigidly on the object introduces a small unknown error which can safely be considered to be in sub-millimitre range.
- *Camera Error:* The pose of the calibration pattern is detected with a standard DLT algorithm, followed by a non-linear optimisation of the pose using the sparse bundle adjustment implementation by Lourakis [37].

- *Quantisation Error:* Depending on image resolution a digital camera introduces a pixel quantisation error. In our evaluation we use a resolution of 640x480 with a focal length of ∼500 in pixel-related units. This leads to an error of about 0.5-1.5 mm when tracking at a distance of 0.5-1.5 m parallel to the image plane. This error is even higher for the orthogonal direction, which shows up in Table II.
- *Modelling Error:* For modelling we measured the main dimensions of the objects used, but we used simplified models that do not account for deviations like small details, chamfers or slightly bulging cardboard surfaces. Unfortunately we do not have a measure for the *Modelling Error* but since we mainly used basic shapes, where correct modelling is simple, we assume this error to be negligible.
- *Texturing Error:* We found that textures added during modelling phase do not align properly and therefore introduced the methods described in Section III-D. Manually capturing textures triggered by pressing a button incorporates less error than automatic capturing based on tracking-state-detection.
- *Tracking Error:* The failure of the tracker to accurately locate the local maximum, depending on the challenges posed by current viewing conditions.

B. Accuracy and Precision

Accuracy is defined to be the closeness of a quantity to its actual value, which in our case is measured using Equation (11), where the pose of tracking $\mathbf{p}_t^2$ is compared to the pose of the virtual object and the pose detected by the calibration software respectively. We evaluated the mean accuracy with respect to the poses of several trajectories using

$$E_{acc} = \frac{1}{Jt_e} \sum_{j=1}^{J} \sum_{t=1}^{t_e} E_t \tag{12}$$

where $j = [1 \ldots J]$ are the trajectories of poses $t = [1 \ldots t_e]$ under unchanged conditions, i.e. tracking J times on a sequence of t_e images.

Precision, also called repeatability, is the degree of deviation of a quantity under unchanged conditions, which is measured using Equation (11), where the pose of tracking $\mathbf{q}_{t,j}^{2,n}$ is compared to its own mean with respect to J, the number of repetitions:

$$\mathbf{q}_t^{1,n} = \frac{1}{J} \sum_{j=1}^{J} \mathbf{q}_{t,j}^{1,n} \tag{13}$$

Table II shows the results of the accuracy and precision evaluation, where *static* refers to the pose after convergence and *dynamic* is the mean error of trajectories j, both over a set of trials J. For evaluation we used box shaped and cylindrical objects. The virtual objects give indication about the *Tracking Error* and *Quantisation Error* (all other errors being ruled out), whereas the difference between virtual and real objects are due to *Mechanical*, *Camera*, *Modelling* and *Texture Error*, where we assume the *Modelling* and *Texture Error* to play the main roles.

TABLE II
ACCURACY AND PRECISION

Target Object	Accuracy [mm]				Precision [mm]			
	static		dynamic		static		dynamic	
	x,y	z	x,y	z	x,y	z	x,y	z
box (virt.)	0.4	2.3	1.5	5.6	0.2	1.1	0.7	3.2
box (real)	2.0	5.5	2.6	7.7	1.1	2.9	1.6	4.9
cylinder (virt.)	0.9	4.4	2.4	10.0	0.4	1.9	1.3	5.7
cylinder (real)	3.0	16.5	3.9	21.9	0.5	2.5	1.6	8.8

We evaluated the dynamic errors under the following conditions:

- Linearly moving objects with different velocities
- Rotating objects
- Arbitrary moving objects (i.e. toppling, rolling)
- Partially occluded objects
- Changing illumination

Table II indicates that curved objects are typically harder to track than box-shaped objects.

A typical trajectory for arbitrary movement is shown in Figure 6 where the tracked pose is compared to the virtual with respect to translations, rotations and the error measured by Equation (11). For generating the virtual poses the poses from the pattern detection and bundle adjustment were used and low-pass filtered to remove jitter.

C. Robustness

We tested our approach against various situations including

- fast movement introducing motion blur,
- occlusion,
- changes in lighting,
- large distances, small objects,
- different objects (high resolution, curved surfaces, low texture, . . .).

Since robustness is hard to put in numbers the reader is referred to a video to get an impression how these various challenges are handled.

D. Performance

Processing time during tracking depends, along with computational power, on the complexity of the model as well as on the number of particles used for tracking.

Table III shows the frame rates for different numbers of faces and particles. 2x50, 3x100 and 4x300 indicates 2, 3 and 4 iterations using 50, 100 and 300 particles for each iteration respectively. Figure 7 shows the frames per second on different GPUs with respect to the total number of particles used for tracking.

E. Video

The video[1] shows how we learn texture and feature points during tracking. Then tracking identifies whether if an object

[1]http://www.youtube.com/watch?v=r5xUcDmTY3E

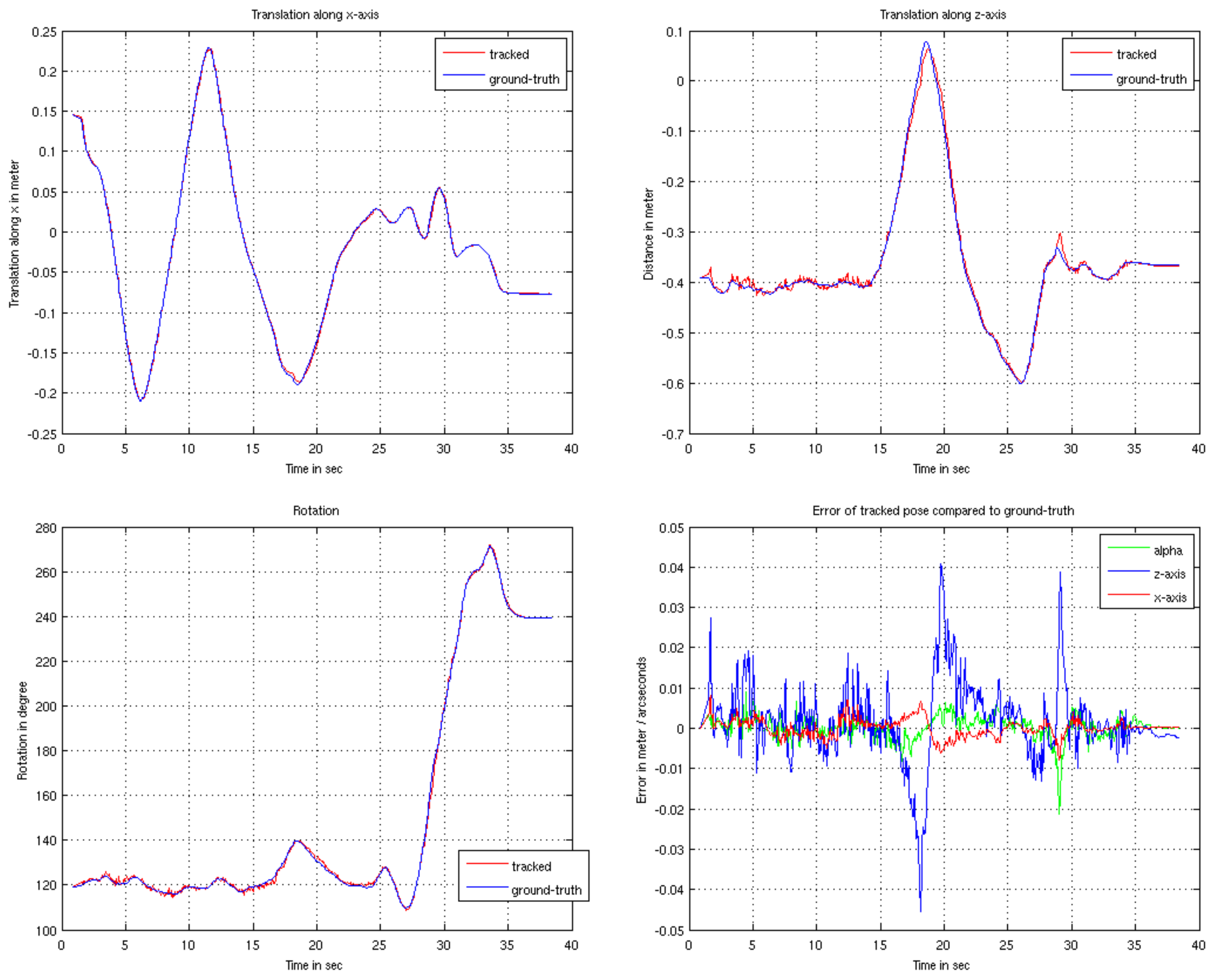


Fig. 6. Trajectory of a tracked virtual object with 45 cm x-translation followed by a 70 cm z-translation and a rotation about the objects y-axis. The lower right figure shows the pose deviations respectively. Note that the jitter results from different location of convergence of the particle filter, due to the errors mentioned in Section V-A.

TABLE III
FRAME RATES

Example Objects	Faces	Frames per Second		
		2x50	3x100	4x300
Box	6	240	100	33
Cylinder (low)	24	220	95	30
Cylinder (mid)	96	210	90	28
Cylinder (high)	384	190	80	25

is occluded or if tracking fails, in which case pose recovery is triggered automatically. Since pose recovery also takes advantage of GPU computing the tracker slows down at this particular moments. Note that we do not interfere with the tracking system via the keyboard other than for changing the display modes.

VI. Conclusion and Future Work

A. Conclusion

We presented a model-based tracking system to accurately follow the pose of an object in real-time. We developed a novel method of Tracking-State-Detection (TSD) that analyses tracking performance to reason about tracking quality, speed and whether the object of interest is occluded or lost. This allows triggering a feature based pose recovery system, texture mapping and online learning. Accuracy, precision and performance of our approach are evaluated carefully to provide a maximum of applicability and give very good results compared to ground truth. Note that we showed tracking results for simple shapes only, which can be easily measured to compare to ground truth. However, our approach is not limited to specific shapes.

B. Future work

There are several improvements for our tracking approach. While the system has no problems in tracking both textured as well as non-textured objects pose recovery only works for the former as it depends on SIFT features. Another open issue is how to combine colour and edges as described in Equation (4) and (5). And while our novel approach for Tracking-State-Detection shows good results for a wide

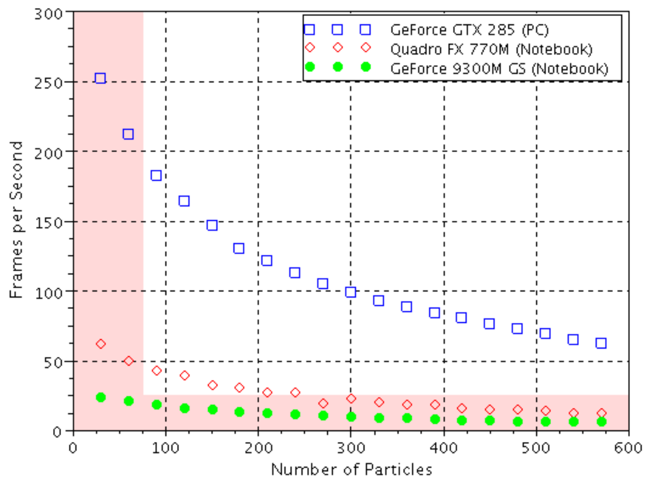


Fig. 7. Frame rate with respect to the number of particles

range of situations, the mathematical formulation in Section III-C is not yet satisfying with respect to generalisability to other tracking methods.

VII. Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement No. 215181, CogX.

References

[1] Y. Bekiroglu, D. Kragic, and V. Kyrki, "Learning grasp stability based on tactile data and hmms," in *RO-MAN, 2010 IEEE*, pp. 132 –137, sept. 2010.

[2] M. Kopicki, R. Stolkin, S. Zurek, T. Mörwald, and J. Wyatt, "Predicting workpiece motions under pushing manipulations using the principle of minimum energy," in *RSS workshop*, 2009.

[3] T. Mörwald, M. Kopicki, R. Stolkin, J. Wyatt, S. Zurek, M. Zillich, and M. Vincze, "Predicting the unobservable, visual 3d tracking with a probabilistic motion model," in *IEEE International Conference on Robotics and Automation, ICRA11*, May 2011.

[4] M. Kopicki, S. Zurek, R. Stolkin, T. Mörwald, and J. Wyatt, "Learning to predict how rigid objects behave under simple manipulation," in *IEEE International Conference on Robotics and Automation, ICRA11*, May 2011.

[5] D. J. Duff, T. Mörwald, R. Stolkin, and J. Wyatt, "Physical simulation for monocular 3d model based tracking," in *IEEE International Conference on Robotics and Automation (ICRA11)*, May 2011.

[6] J. Chestnutt, S. Kagami, K. Nishiwaki, J. Kuffner, and T. Kanade, "Gpu-accelerated real-time 3d tracking for humanoid locomotion," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007.

[7] P. Michel, J. Chestnutt, S. Kagami, K. Nishiwaki, J. Kuffner, and T. Kanade, "Gpu-accelerated real-time 3d tracking for humanoid autonomy," in *JSME Robotics and Mechatronics Conference (ROBOMEC'08)*, June 2008.

[8] G. Klein and D. Murray, "Full-3d edge tracking with a particle filter," in *British Machine Vision Conference*, 2006.

[9] L. Masson, M. Dhome, and F. Jurie, "Robust real time tracking of 3d objects," in *International Conference on Pattern Recognition, ICPR*, 2004.

[10] L. Vacchetti, V. Lepetit, and P. Fua, "Combining edge and texture information for real-time accurate 3d camera tracking," in *IEEE/ACM International Symposium on Mixed and Augmented Reality*, 2004.

[11] P. Mittrapiyanuruk, G. N. Desouza, and A. C. Kak, "Accurate 3d tracking of rigid objects with occlusion using active appearance models," in *WACV/MOTION*, pp. 90–95, 2005.

[12] L. Vacchetti, V. Lepetit, and P. Fua, "Stable real-time 3d tracking using online and offline information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004.

[13] J. Sánchez, H. Álvarez, and D. Borro, "Towards real time 3d tracking and reconstruction on a gpu using monte carlo simulations," in *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 185 –192, Oct. 2010.

[14] M. Özuysal, M. Calonder, V. Lepetit, and P. Fua, "Fast keypoint recognition using random ferns," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009.

[15] M. Vincze, M. Ayromlou, W. Ponweiser, and M. Zillich, "Edge-projected integration of image and model cues for robust model-based object tracking," *The International Journal of Robotics Research*, 2001.

[16] H. H. Nagel, "Representation of moving rigid objects based on visual observations," *Computer*, vol. 14, pp. 29–39, August 1981.

[17] R. Eskenazi and R. T. Cunningham, "Real-time tracking of moving objects in tv images," *IEEE Workshop Pattern Recognition and Artificial Intelligence*, pp. 4–6, April 1978.

[18] C. Harris and A. Blake ed., "Tracking with rigid bodies," *Active Vision*, pp. 59–73, 1992.

[19] T. Drummond and R. Cipolla, "Real-time tracking of complex structures with on-line camera calibration," in *British Machine Vision Conference (BMVC99*, pp. 574–583, 1999.

[20] G. Klein and T. Drummond, "Robust visual tracking for non-instrumented augmented reality," in *ISMAR IEEE/ACM International Symposium on Mixed and Augmented Reality*, 2003.

[21] D. Lowe, "Robust model-based motion tracking through the integration of search and estimation," *International Journal of Computer Vision*, pp. 113–122, 1992.

[22] D. Gennery, "Visual tracking of known three-dimensional object," *International Journal of Computer Vision*, 1992.

[23] D. Koller, K. Daniilidis, and H.-H. Nagel, "Model-based object tracking in monocular image sequences of road traffic scenes," *International Journal of Computer Vision*, 1993.

[24] A. Kosaka and G. Nakazawa, "Vision-based motion tracking of rigid objects using prediction of uncertainties," *International Conference on Robotics and Automation*, 1995.

[25] A. Ruf, M. Tonko, R. Horaud, and H.-H. Nagel, "Visual tracking by adaptive kinematic prediction," in *International Conference on Intelligent Robots and Systems*, 1997.

[26] H. de Ruiter and B. Benhabib, "Visual-model-based, real-time 3d pose tracking for autonomous navigation: methodology and experiments," *Autonomous Robots*, vol. 25, pp. 267–286, 2008.

[27] D. Kragic, A. T. Miller, and P. K. Allen, "Real-time tracking meets online grasp planning," in *IEEE Intl. Conf. on Robotics and Automation*, pp. 2460–2465, 2001.

[28] E. Marchand and P. Bouthemy, "A 2d-3d model-based approach to real-time visual tracking," *IVC*, vol. 19, pp. 941–955, 2001.

[29] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendn-Mancha, "Binocular visual tracking and grasping of a moving object with a 3d trajectory," *Journal of Applied Research and Technology*, vol. 7, no. 03, pp. 259–274, 2009.

[30] A. Doucet, N. De Freitas, and N. Gordon, eds., *Sequential Monte Carlo methods in practice*. Springer, 2001.

[31] A. Doucet, S. Godsill, and C. Andrieu, "On sequential monte carlo sampling methods for bayesian filtering," *Statistics and Computing*, vol. 10, pp. 197–208, 2000.

[32] T. Mörwald, M. Zillich, and M. Vincze, "Edge tracking of textured objects with a recursive particle filter," in *Graphicon 2009*, (Moscow, Russia), 2009.

[33] A. Richtsfeld, T. Mörwald, M. Zillich, and M. Vincze, "Taking in shape: Detection and tracking of basic 3d shapes in a robotics context," in *Computer Vision Winter Workshop*, pp. 91–98, 2010.

[34] C. Wu, "http://www.cs.unc.edu/ ccwu/siftgpu/."

[35] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Comm. of the ACM*, vol. 24, pp. 381–395, 1981.

[36] T. Mörwald, J. Prankl, A. Richtsfeld, M. Zillich, and M. Vincze, "Blort - the blocks world robotic vision toolbox," in *ICRA workshop*, 2010.

[37] M. A. Lourakis and A. Argyros, "SBA: A Software Package for Generic Sparse Bundle Adjustment," *ACM Trans. Math. Software*, vol. 36, no. 1, pp. 1–30, 2009.

IEEE International Conference on Robotics and Automation, 2012.
Provided as part of DR 2.5: Qualitative behaviour models, grasping of novel objects

To appear in the *IEEE International Conference on Robotics and Automation*, 2012.
This copy is for personal use only.
© 2012 IEEE
Prints shortly available at http://ieeexplore.ieee.org.

Generalizing Grasps Across Partly Similar Objects

Renaud Detry Carl Henrik Ek Marianna Madry Justus Piater Danica Kragic

***Abstract*— The paper starts by reviewing the challenges associated to grasp planning, and previous work on robot grasping. Our review emphasizes the importance of agents that generalize grasping strategies across objects, and that are able to transfer these strategies to novel objects. In the rest of the paper, we then devise a novel approach to the grasp transfer problem, where generalization is achieved by *learning*, from a set of grasp examples, a dictionary of object parts by which objects are often grasped. We detail the application of dimensionality reduction and unsupervised clustering algorithms to the end of identifying the size and shape of parts that often predict the application of a grasp. The learned dictionary allows our agent to grasp novel objects which share a part with previously seen objects, by matching the learned parts to the current view of the new object, and selecting the grasp associated to the best-fitting part. We present and discuss a proof-of-concept experiment in which a dictionary is learned from a set of synthetic grasp examples. While prior work in this area focused primarily on shape analysis (parts identified, e.g., through visual clustering, or salient structure analysis), the key aspect of this work is the emergence of parts from *both* object shape *and* grasp examples. As a result, parts intrinsically encode the intention of executing a grasp.**

I. Introduction: Challenges in Grasp Planning

This paper studies the planning of grasping actions, or, in other words, the problem of exploiting perceptual data to select a wrist position and finger configuration to which a hand can be transported in order to grasp an object. The wrist position (or *grasping point*) corresponds to the region of the object towards which the hand will move. The finger configuration (or *hand preshape*) corresponds to the angles to which finger joints are set prior to coming in contact with the object.

Grasp planning is a complex problem. A grasp must bind a hand to an object, and prevent the object from subsequently slipping or escaping. Configurations which lead to a collision between the hand and the object or other obstacles must be avoided, and task-related constrains must be verified (certain tasks restrain the number of possible grasps, as a knife should be held specifically by its handle when the task is to cut something). Perceptual data, usually provided by vision, are noisy and often limited to a single viewpoint. For dexterous grasping, the space of action parameters (hand positions and configurations) quickly becomes high-dimensional (a human hand has twenty-five degrees of freedom – six for the wrist position and orientation, and nineteen for the finger joint angles). Yet, despite the complexity of the problem, the frequent recurrence of grasping in everyday tasks imposes an ability to plan grasps quickly.

In robotics, grasp planning traditionally relies on contact-force analysis [3], [34]. Force analysis bases planning on a reconstruction of the geometry and physical properties of the objects that surround the agent. Provided that such a reconstruction is available, the agent searches the space of hand configurations for the configuration that best verifies grasping constraints (binding configuration, no collisions, task compatibility). In practice, the applicability of force analysis is limited by the difficulty of obtaining accurate models of object geometry, mass, and friction characteristics. Also, as the space of hand configurations is high dimensional, the optimization procedure underlying force analysis is computationally expensive. These shortcomings motivated the community to rethink the planning problem, leading for instance Borst et al. [5] to demonstrate that finding the globally optimal grasp is often not strictly worth the computational effort, as for many tasks an average grasp (in the force-analysis sense) is acceptable. The bigger leap however came with a class of methods that parted drastically from the traditional planning philosophy. Instead of searching for a grasp that optimally satisfies the various (vision-dependent) grasping constraints, these methods extract, from the agent's experience, a function that directly maps visual perceptions to grasp parameters, with the advantage of *implicitly* capturing the object's physical properties, and avoiding a costly search through the high-dimensional space of hand configurations [7], [21], [25], [30], [36].

Numerous behavioral studies tend to support the existence of similar processes in the human grasping system. It has been shown for instance that humans often grasp objects by preshaping their hand during its transportation towards the object [18], then compliantly refining the grip upon contact [19]. Concurrently, neurophysiological studies suggested that, in monkeys, the cortex encodes a set of prototype grasps, which are selectively triggered by visual stimuli [26]. It thus seems plausible, as proposed, for instance, by Johansson et al. [19], that the human grasping system relies on a set of prototypical motor programs that are selected and parametrized by visual input, therefore acting as a direct mapping from vision to action. Humans arguably possess the most sophisticated grasping system known today, being able to plan complicated grasps in just a few hundreds of

R. Detry, C. H. Ek, M. Madry, and D. Kragic are with the Centre for Autonomous Systems and the Computer Vision and Active Perception Lab, CSC, KTH Royal Institute of Technology, Stockholm, Sweden. Justus Piater is with the Intelligent and Interactive Systems group, Institute of Computer Science, University of Innsbruck, Austria. Email: {detryr,chek,madry,danik}@csc.kth.se, justus.piater@uibk.ac.at

This work was supported by the Swedish Foundation for Strategic Research, the Belgian National Fund for Scientific Research (FNRS), and the EU projects CogX (FP7-IP-027657) and TOMSY (IST-FP7-Collaborative Project-270436).

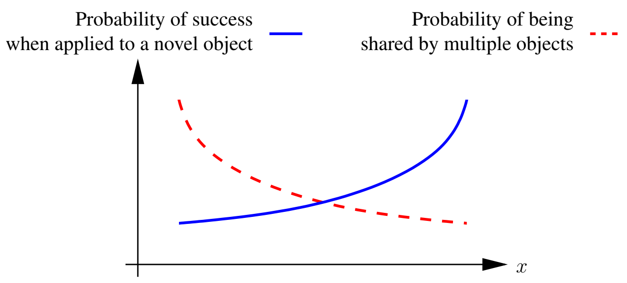


Fig. 1: Robustness-transferability trade-off in feature-based grasp planning. The x axis corresponds to the amount of information encoded by a part. Highly informative parts allow for a robust grasp application. However, these are less likely to be shared across objects.

milliseconds [17]. We believe that the possibility that such an efficient system be based on a direct vision-action mapping is a strong argument for researching vision-action mappings for robotics.

To learn a vision-to-grasp mapping for one specific object, an agent usually collects a set of grasp examples, and lets machine-learning algorithms construct a grasp predictor from these. Such a model allows the agent to quickly produce grasping plans for the object on which it trained. However, collecting grasp examples is an expensive, time-consuming process. A major focus in grasp learning is to develop methods that produce useful manipulation models from as few data as possible. A natural means of limiting the need for examples is to try and adapt memories of previous objects to the planning of a grasp onto a novel object. Many objects share similarities in shape, and similarities in grasp affordances, and both are often correlated. When a novel object appears, instead of starting to learn from scratch, an agent may instead attempt to apply to it the strategies it has acquired for partly similar objects. To this end, means of linking grasps to certain object *features* have been researched, in the hope of transferring grasps across objects that share the same features. The challenge of this task is to decide which visual cues should be captured by the features. Intuitively, a feature should capture no more no less than the specific cues that predict the applicability of a grasp. If a feature misses important cues, it risks predicting faulty grasps. If a feature includes cues that are not directly related to grasping, its transferability to other objects will be impeded. Designing a feature for grasp generalization thus involves a robustness-transferability trade-off, as illustrated in Fig. 1.

A number of methods for vision-based grasping learn a mapping from image features, such as local gradients or SIFT, to grasp parameters [24], [25], [30]. One advantage of these methods is their conceptual elegance:

1) Extract features from images of a set of objects.
2) Label these features as good or bad grasping point, either with the help of a teacher [30] or through autonomous exploration [24].
3) Learn a grasp classifier.

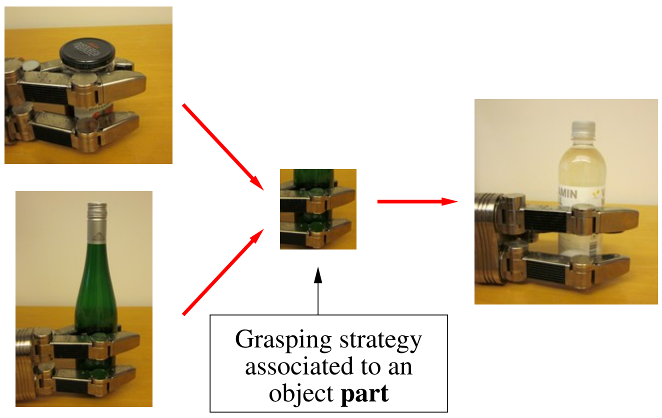


Fig. 2: Learning part-grasp associations. The agent will identify, within its visuomotor experience, recurrent associations of object *parts* and successfully executed grasps. These grasps will then be applicable to novel objects that share the same part.

4) Transfer grasps by classifying features obtained from images of novel objects.

Unfortunately, these methods also come with their shortcomings. From a practical viewpoint, the geometric information provided by a local feature detector is generally poor. As grasping is an intrinsically 3D interaction, it largely relies on 3D object properties, such as shape, which are only partly captured by 2D image features. It is thus difficult to link, for example, a 3D gripper orientation to an image feature.

Across the range of visual cues that have been used for designing grasp planners, 3D shape has lead to particularly good results. By contrast to methods based on image features, methods that link grasp parameters to a shape model [1], [9], [11], [14], [23] benefit from an increased geometric robustness, which makes it easier to preshape the hand to approximate object shapes, and accurately position and orient the wrist and fingers with respect to the object. Mapping grasps to 3D cues is supported by behavioral and neurophysiological studies. Behavioral studies have demonstrated the reliance of human grasping on 3D shape [16], while neurophysiologists have observed a mapping from 3D shape to action prototypes in monkeys [27].

II. Learning Shape Prototypes for Generalizing Grasps

In the rest of the paper, we present an adaptive grasp planner that learns a mapping from object shape to grasp parameters.

A. From Part to Grasp

Linking grasp parameters to the shape of the whole body of an object limits the applicability of the model to that particular object. In order to transfer grasps across objects, we instead explore the linking of grasp parameters to object parts. In order to allow the agent to generalize its acquired knowledge to novel objects, we propose to provide it with means of identifying, within its visuomotor experience, recurrent associations of object *parts* and successfully executed grasps. For instance, the agent may have successfully transported objects such as bottles, cans, and jars, which have different sizes, but which can be seized by applying the same power grasp to their side. We propose to provide the agent with means of understanding, from a set of such examples, that any object that presents a cylindrical part can be grasped sideways with a wide-palm grasp (Fig. 2).

B. Previous Work on Part-based Grasping

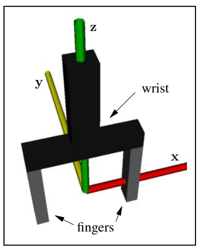
Part-grasp associations have been previously suggested and studied by several research groups [2], [23], [36]. In the earlier work, the definition of parts was often either hard-coded [23], or driven by shape analysis [1], [2], [36]. There is however an increasing interest for defining parts based on grasping experience [10], [12], [15], [22], [37]. For instance, Herzog et al. [15] and Zhang et al. [37] presented two exciting data-driven approaches where a part describes an object's shape in a fixed-size region around a grasping point. These approaches are further discussed below.

C. Method

Our work aims at learning, from a set of grasp examples, a dictionary of prototypical parts by which objects are often grasped. A key property that we wish to allow our agent to extract from experience is the spatial extent of grasp-predicting parts. For instance, in the case presented in Fig. 2, we wish our agent to learn that the relevant part is a 10cm-high cylinder. The the tap of the jar or the conic upper part of the bottle should be ignored, as they are not shared by the two objects.

Training data are provided to the agent in the form of a set of grasps demonstrated onto objects known to the agent. (The agent has previously acquired 3D point clouds that model the shape of the objects.) A grasp is parametrized by the 6D pose of the wrist (3D position and 3D orientation), and by the 6D pose of the object. Our method works as follows: First, the agent generates, from the grasp examples, a large number of part candidates of varying sizes (Section III). Most of the candidates will not generalize well. However, it is our hope that for every set of objects that share a graspable part, each object will yield one candidate that approximately captures that part. The candidates that recur across objects are identified by clustering part candidates (Section IV). Dense clusters will contain parts by which objects are often grasped, which are thus promising for grasping novel objects.

The central parts of all clusters will form the dictionary used by the agent to grasp novel objects. An important aspect of our work appears at this point. As the dictionary of parts is only formed from cluster centers, it is allowed to be orders of magnitude smaller than the set of grasp examples initially provided to the agent. In the data-driven approaches discussed above [15], [37], each grasp example yields a part. By contrast, in our work, a grasp example only "votes" for the potential inclusion of a part into the dictionary, which provides us with a means of controlling the size of the dictionary in order to keep the computational cost of planning a grasp onto a novel object reasonably low.



(a) Gripper reference frame

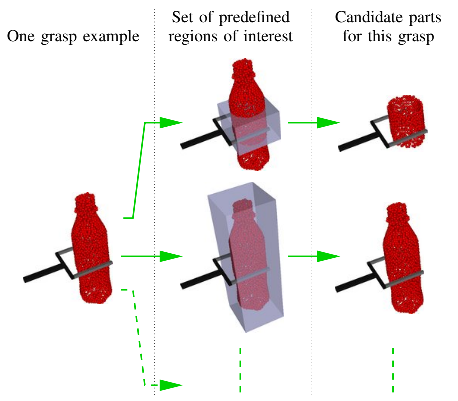

One grasp example
Set of predefined regions of interest
Candidate parts for this grasp
(b) Generating part candidates

Fig. 3: Generating part candidates. The black and grey renderings on each image represent the pose of the gripper set for a sideways grasp on the soda bottle. Parts of varying sizes are generated by defining several box-shaped regions of interest centered on the gripper.

Also, in our work, parts emerge from both object shape and grasp examples. A key result is our ability to optimize the robustness-transferability trade-off discussed above. Not only the shape, but also the spatial extent (or size) of the parts that form the dictionary depend on the available grasp data. Our approach involves an explicit search for recurrent patterns within the agent's visuomotor experience, which leads to the identification of parts that directly predict grasp applicability.

III. Generating Part Candidates

Part candidates are generated by extracting object surface segments of varying size in the vicinity of grasps demonstrated by a teacher. Parts are thus represented, as the object from which they are extracted, by point clouds. This process is illustrated for a soda bottle in Fig. 3. Surface segments are extracted using a set of predefined regions of interest (ROI). These regions are centered on the gripper, as the applicability of a grasp is largely conditioned by the shape of the surface in the direct vicinity of the grasping point. ROI sizes should *a priori* vary in all directions. However, the preshape of the gripper at the time of the grasp can limit the number of regions that are interesting to look at. For instance, in the case shown in Fig. 3, it is reasonable to limit the ROI width along the $\mathbf{x}$ axis of the gripper to the distance that separates both fingers, as the object will usually not be larger that this gap. With more sophisticated hands, grasp preshapes can further constrain the definition of ROIs.

IV. Extracting Dense Clusters of Parts

Graspable parts that generalize are discovered by clustering part candidates. Dense groups of similarly-shaped candidates correspond to shapes onto which grasps can be applied in order to seize several different objects. These shapes are thus likely to predict grasp applicability for novel objects.

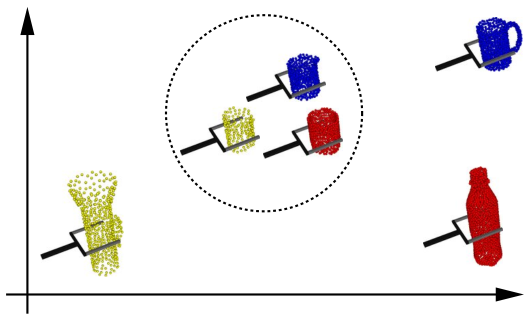


Fig. 4: Finding parts that allow for transferring grasps to a novel object. The three outer "parts" (which correspond to entire objects), will not generalize well. By contrast, the three center parts, which represent a piece of the flashlight, cup, and soda bottle, are very similar to each other. As there exist a shape similarity across these three parts extracted from different objects of the training database, it seems reasonable to assume that the grasps related to these parts are potentially applicable to novel objects.

In Fig. 4, none of three outer parts would be applicable to other objects. The three middle parts, by contrast, encode a shape-grasp relation that would be applicable to an object that has a cylindrical part of a similar diameter.

Clustering part candidates requires the definition of a measure of shape (dis)similarity. This measure is defined in the next section. Section IV-B details the clustering algorithm.

A. *Measuring Part Dissimilarity*

This section defines a measure part dissimilarity. We note that, as we ultimately aim at using parts for predicting gripper poses, we must measure the (dis)similarity of *gripper-relative* shapes. In other words, a cylindrical part grasped from the side should *not* be similar to the same cylindrical part grasped from the bottom.

In this work, a part is represented by a point cloud defined in a reference frame that corresponds to the 6D pose of the grasp associated to that part. Let $P = \{x_i\}_{i\in[0,n]}$ and $Q = \{y_i\}_{i\in[0,m]}$ denote the point-cloud representations of two parts, with all x_i's and y_i's belonging to $\mathbb{R}^3$. Let us then denote by d^* an asymmetric measure of dissimilarity of P and Q, with

$$d^*(P,Q) = \sum_{i=0}^{n} \min_{j\in[0,m]} f(x_i, y_j), \tag{1}$$

where

$$f(x,y) = \begin{cases} \frac{\|x-y\|}{T} & \text{if } \|x-y\| \le T, \\ 1 & \text{if } \|x-y\| > T. \end{cases} \tag{2}$$

The dissimilarity d^* is often used as error function for point-cloud alignment. In our experiments, the threshold T is set to two centimeters.

We define the dissimilarity of two parts P and Q as

$$d(P,Q) = d^*(P,Q) + d^*(Q,P). \tag{3}$$

The dissimilarity d is symmetric in its arguments. It amounts to the sum of the Euclidean distances between the points of P and their nearest neighbor in Q, and the points of Q and their nearest neighbor in P.

B. *Clustering Parts*

The dissimilarity measure defined in the previous section provides us with a qualitative tool for reasoning on the recurrence of shape-gripper associations across grasp examples. As expressed in the conceptual illustration of Fig. 4, we wish to find a geometric configuration with dense clusters of parts induced by our similarity measure. Dense clusters will correspond to parts that frequently occur within our database. These parts are therefore likely to be useful for grasping novel objects.

The measure described in IV-A provides a global dissimilarity measure between each item in the database from which we can generate a distance matrix

$$D_{ij} = d(P_i, P_j) \tag{4}$$

for all the entries in the database. In order to interpret the data we wish to find a geometrical configuration of the datapoints where the Euclidean distance corresponds to the dissimilarity measure we defined. One possibility is to directly apply classical multi-dimensional scaling [8] to the distance matrix. However, in this paper we are interested in finding a geometrical configuration which suits interpreting the data in terms of clusters. In order to do so we introduce additional flexibility by first interpreting the distance matrix in terms of an inner-product of Gram matrix. Distance matrices and Gram matrices can be interchanged [29] as data inducing representations. Dependent on applications there are benefits associated with each view-point. Here the use of a Gram matrix allows us to view the matrix as a covariance matrix; this approach is well known as the "kernel-trick" [4]. To that end, we use a squared exponential function to apply a non-linear transform of the space that the dissimilarity measure induces,

$$k(P,Q) = e^{-\frac{d(P,Q)^2}{\sigma}}. \tag{5}$$

The squared exponential function induces a geometrical space well-suited for clustering as it will push points that are close together closer and move points far apart even further apart. The parameter σ controls the strength of this transformation.

Discovering part clusters could be achieved directly on the distances defined above (4). However, in order to facilitate the illustration of our method in the experiments presented below, we first recover a low-dimensional approximation of the data, then cluster the data in this low-dimensional space. We recover a d dimensional approximation of the data by solving the following minimization problem,

$$\hat{\mathbf{C}} = \text{argmin}_{\mathbf{C}} ||\mathbf{K} - \mathbf{C}||_{\text{F}}^2, \tag{6}$$

where $\mathbf{K}$ is the Gram matrix whose elements are defined by $k(P_i, P_j)$ for all the entries in the database, and the rank of $\mathbf{C}$ is constrained to be at most d. The solution can be found in close form through an eigenvalue problem and is well-known as kernel principal component analysis [31].

Having resolved a geometrical representation of the data, we wish to partition the space in such a manner that we can discover atomic classes of grasps independent of object type. We proceed through a two-stage process. First, we want to group each point in the database into a small number of classes. Secondly, we wish to explain each class by a single representative grasp. Underpinning our approach is the notion that the dissimilarity measure contains this desired structure. This assumption implies that the grouping can be cast as a clustering problem. Clustering is a well-studied problem within computer science and datamining. It has been used extensively to create compact representations of data using mixture models [35] or for application scenarios where a significant amount of prior information about the partitioning is available [6].

The dissimilarity measure $d(\cdot,\cdot)$ is defined between each point in the database. This allows us to construct a graph $G \in \{\mathcal{V},\mathcal{E}\}$ where each grasp is represented by a node $v_i \in \mathcal{V}$ with edges $e_{ij} \in \mathcal{E}$ connecting associated nodes. We wish to find a partitioning that respects the dissimilarity measure $d(\cdot,\cdot)$. To that end, we construct a fully connected graph. The edge weights are $e_{ij} = \mathbf{C}_{ij}$, *i.e.*, inversely proportional to the dissimilarity between the grasps according to our measure. In order to partition the space, it now remains to cut the graph into disjoint regions each representing a cluster.

In this paper we employ the normalized cuts [33] approach to partition the graph. The $\text{cut}(\mathcal{A},\mathcal{B})$ of a graph G into two sets of disjoint nodes $\mathcal{A}$ and $\mathcal{B}$ is defined as,

$$\text{cut}(\mathcal{A},\mathcal{B}) = \sum_{i\in\mathcal{A},j\in\mathcal{B}} e_{ij}. \tag{7}$$

The normalized cuts algorithm finds the partitioning of the graph that minimizes the following objective function,

$$\text{cut}_{\text{normalized}}(\mathcal{A},\mathcal{B}) = \frac{\text{cut}(\mathcal{A},\mathcal{B})}{\text{assoc}(\mathcal{A},\mathcal{V})} + \frac{\text{cut}(\mathcal{A},\mathcal{B})}{\text{assoc}(\mathcal{B},\mathcal{V})}, \tag{8}$$

$$\text{assoc}(\mathcal{A},\mathcal{V}) = \sum_{i\in\mathcal{A},j\in\mathcal{V}} e_{ij}. \tag{9}$$

The denominator grows with increasing node sets which works to penalize creating very small clusters.

V. Proof of Concept

We now present a proof-of-concept experiment which illustrates the method suggested above. The experiment is realized on synthetic data consisting of seven two-finger grasps demonstrated on four objects (see Fig. 5a and Fig. 5b).

Three sets of regions of interest were defined for the three grasp types present in the database. Three ROIs were defined for "cylindrical" grasps, which correspond to the grasps number 1, 2 and 3 of Fig. 5b. Four ROIs were defined for the parallel grasps (4, 5, 6), and six ROIs for the pinch grasp (7). We note that, in the case of the synthetic data studied in this paper, considering cylindrical, parallel and pinch grasps is purely anecdotal. However, in

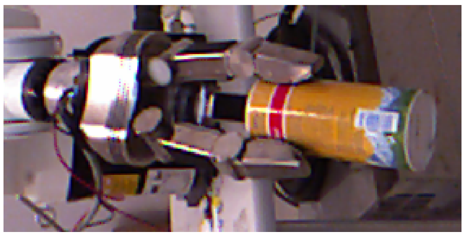


Fig. 6: Cylindrical grasp preshape. The finger-surface normals at the contact points are 120° apart.

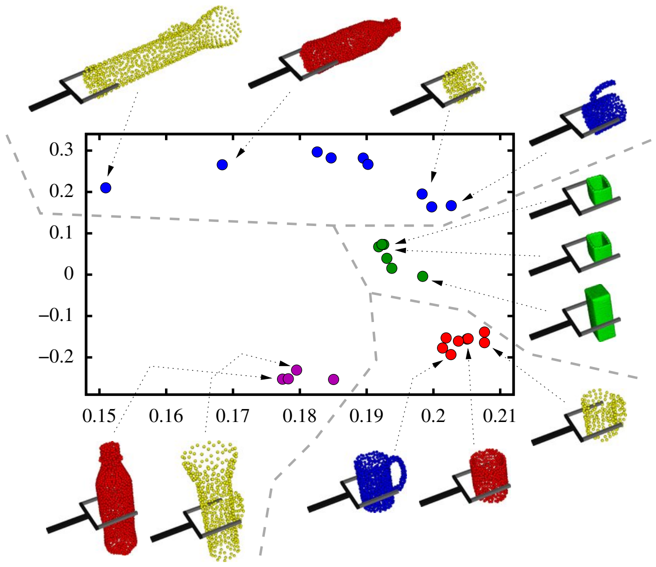
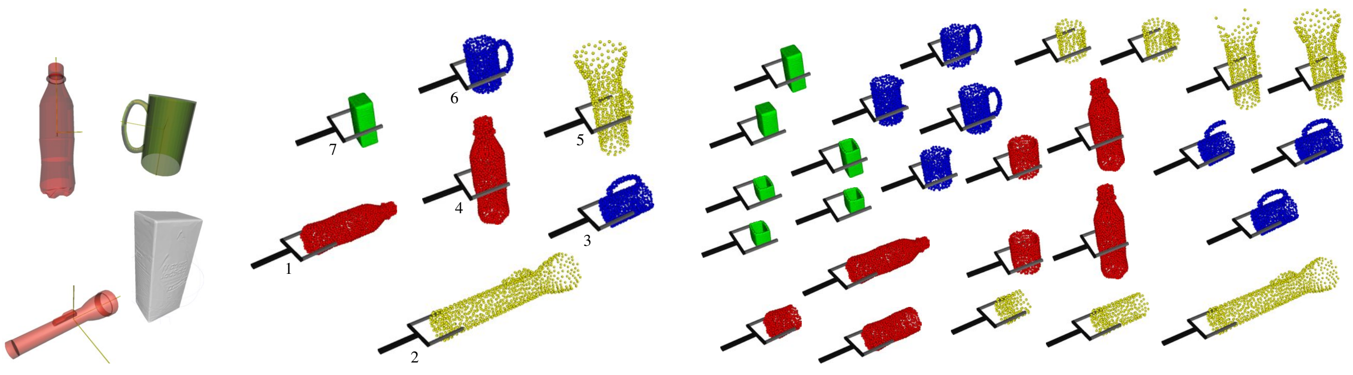


Fig. 7: Two-dimensional approximation of candidates' geometric configuration, computed from the dissimilarity measure of Section IV-A. Dot colors indicate the data cluster to which a datapoint (part candidate) belongs (see text for details). The colors of the dots within the plot and the colors of the parts surrounding the plot are unrelated. We note that the vertical and horizontal axes are not equally scaled.

a real-case scenario, the hand preshape used for a given grasp would allow us to limit the number of parts that need to be considered as candidates. For instance, with a cylindrical grasp (Fig. 6), generating ROIs that differ in size in a direction perpendicular to the palm of the hand is more important than considering variations along directions parallel to the palm. With a parallel grasp (for instance, Fig. 2), ROIs of various lengths in a direction parallel to the palm are necessary. These observations motivated the definition of different sets of ROIs for the different types of grasps shown in Fig. 5. The part candidates generated with these ROIs are shown in Fig. 5c.

As explained in Section IV-B, kernel PCA provides us with low-dimensional approximations of our data. A two-dimensional approximation is show in Fig. 7. This plot shows that the dissimilarity measure of Section IV-A properly separates candidate parts in groups of similarly-shaped parts. These groups can be correctly identified by the clustering algorithm of Section IV-B, as reported by the colors associ-



(a) Object set (b) Demonstrated grasps (c) Part candidates

Fig. 5: Experimental data. Three of the objects are cylinders of different sizes, and one is a box. Seven grasps are synthetically demonstrated to the agent. for the cylinders, both sideways and top-down grasps are demonstrated. Fig. (c) shows the candidate parts computed from the grasps of Fig. (b). Part colors indicate which object a part is segmented from.

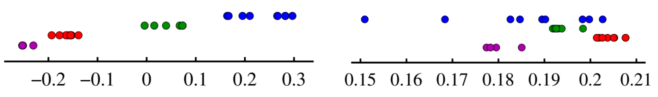


Fig. 8: Projection of the data (candidate parts) onto the first (left) and second (right) principal components of the data. Colors indicate the data cluster to which a datapoint belongs (see text for details). The elevation of the datapoints above horizontal axes is meant to help identifying clusters.

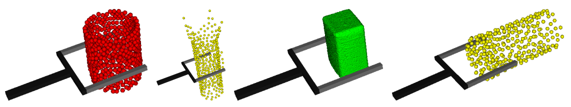


Fig. 9: Prototype parts. These parts correspond to the centers of the clusters of Fig. 7.

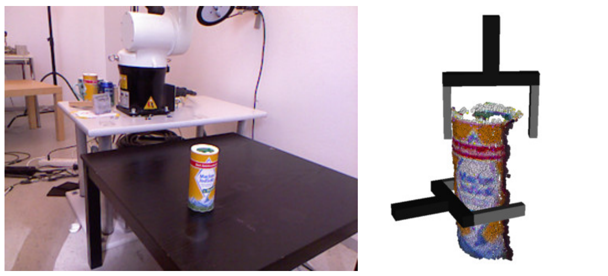


Fig. 10: Grasping a novel object using a dictionary of parts. The rightmost image shows the grasps suggested by the first and last prototypes of Fig. 9, respectively approaching the object from the side and from the top.

ated to the datapoints. In this paper, the number of clusters was determined by inspection. However, BIC-like criterions that compute an optimal number of clusters could be used instead [32]. We note that the two axes of this plot are not equally scaled. The data shows a larger variance along the vertical axis than along the horizontal axis. Fig. 8 shows the projection of the data onto its first and second principal components (which correspond to the vertical and horizontal axes of Fig. 7, respectively). Fig. 8 indicates that the first component contains enough information to identify most of the clusters computed from the dissimilarity measure. The second component leads to a clear separation of the purple and red clusters.

Despite the modest number of data, computing the central point of each cluster allows us to identify a set of prototypical graspable parts. These parts are shown in Fig. 9. We emphasize that despite its reliance on complete object shape models for learning prototypical parts, the method presented above is applicable to predicting grasps onto novel objects perceived through a single 3D snapshot. Fig. 10 illustrate the application of the first and last prototypes of Fig. 9 to a novel object. The right side of Fig. 10 shows the point-cloud representation of the scene (captured by a depth sensor), and the two grasps suggested by the prototypes. The parts are aligned to the object using the pose estimation method of Detry et al. [13].

VI. Discussion

The dissimilarity measure of Section IV-A provides a direct channel for injecting expert knowledge into to the method presented above. By choosing suitable dissimilarities, one can let a variety of desirable visuomotor strategies emerge from data clustering. For instance, one may argue that similarly-shaped parts may predict similar grasps despite a scale difference. Basing a similarity measure on a mix of local shape features (Spin images [20], or FPFH [28]) and global shape features (for instance, the first few moments of a point cloud) has the potential of robustly representing shape while being invariant, to some extent, to scale. Such a measure would allow an agent to understand that cylinders of different radii can be grasped in similar ways. Simultaneously, the distance matrix of Eq. 4 would be much simpler to compute from a set of compact shape features than from the original point-cloud representations. Using shape features would effectively move some of the computational effort out of the distance-matrix computation (quadratic in the number of candidate parts), into a process linear in the number of candidate parts.

Grasp preshapes were discussed in the previous section, albeit remaining of anecdotal use. In a real-world scenario involving a dexterous hand, preshape is an essential grasping property. In such a scenario, a dissimilarity measure would benefit from the availability of preshape parameters, as it would provide an additional cue for separating unrelated parts.

VII. Conclusion

We reviewed the challenges associated to robotic grasping and the importance of devising means of transferring grasping strategies across objects. We then depicted a method that allows an agent to identify, within its visuomotor experience, graspable parts that generalize across objects. Part candidates are first generated by extracting object surface segments in the vicinity of grasps demonstrated by a human. Candidates are then clustered by means of nonlinear dimensionality reduction and unsupervised learning algorithms. The central elements of the resulting clusters are selected to form a dictionary of prototypical parts that can then be used for grasping novel objects. As the dictionary of parts is only formed from cluster centers, it is allowed to be orders of magnitude smaller than the set of grasp examples initially provided to the agent. A grasp example only "votes" for the potential inclusion of a part into the dictionary, which provides us with a means of controlling the size of the dictionary in order to keep the computational cost of planning a grasp onto a novel object reasonably low. Finally, not only the shape, but also the spatial extent (or size) of the parts that form the dictionary depend on the available grasp data. Prototypical parts are selected based on their recurrence across experienced grasps, which leads to the identification of parts that strongly predict grasp applicability.

References

[1] J. Aleotti and S. Caselli. Part-based robot grasp planning from human demonstration. In *IEEE International Conference on Robotics and Automation*, 2011.

[2] C. Bard and J. Troccaz. Automatic preshaping for a dextrous hand from a simple description of objects. In *IEEE International Workshop on Intelligent Robots and Systems*, pages 865–872. IEEE, 1990.

[3] A. Bicchi and V. Kumar. Robotic grasping and contact: a review. In *IEEE International Conference on Robotics and Automation*, 2000.

[4] C. M. Bishop. Pattern recognition and machine learning, 2006.

[5] C. Borst, M. Fischer, and G. Hirzinger. Grasping the dice by dicing the grasp. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 4, pages 3692–3697, 2003.

[6] Y. Boykov and M.-P. Jolly. Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D Images. In *International Conference on Computer Vision*, 2005.

[7] J. Coelho, J. Piater, and R. Grupen. Developing haptic and visual perceptual categories for reaching and grasping with a humanoid robot. In *Robotics and Autonomous Systems*, volume 37, pages 7–8, 2000.

[8] M. Cox and T. Cox. Multidimensional scaling. *Handbook of data visualization*, Jan. 2008.

[9] C. de Granville, J. Southerland, and A. H. Fagg. Learning grasp affordances through human demonstration. In *IEEE International Conference on Development and Learning*, 2006.

[10] R. Detry. *Learning of Multi-Dimensional, Multi-Modal Features for Robotic Grasping*. PhD thesis, University of Liège, 2010. Supervisor: Justus Piater.

[11] R. Detry, D. Kraft, O. Kroemer, L. Bodenhagen, J. Peters, N. Krüger, and J. Piater. Learning grasp affordance densities. *Paladyn. Journal of Behavioral Robotics*, 2(1):1–17, 2011.

[12] R. Detry and J. Piater. Grasp generalization via predictive parts. In *Austrian Robotics Workshop*, 2011.

[13] R. Detry, N. Pugeault, and J. Piater. A probabilistic framework for 3D visual object representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(10):1790–1803, 2009.

[14] C. Goldfeder, M. Ciocarlie, H. Dang, and P. Allen. The Columbia grasp database. In *IEEE International Conference on Robotics and Automation*, 2009.

[15] A. Herzog, P. Pastor, M. Kalakrishnan, L. Righetti, T. Asfour, and S. Schaal. Template-based learning of grasp selection. In *The PR2 Workshop (Workshop at IROS'11)*, 2011.

[16] Y. Hu, R. Eagleson, and M. A. Goodale. Human visual servoing for reaching and grasping: The role of 3-d geometric features. In *IEEE International Conference on Robotics and Automation*, 1999.

[17] L. S. Jakobson and M. A. Goodale. Factors affecting higher-order movement planning: a kinematic analysis of human prehension. *Experimental Brain Research*, 86(1):199–208, 1991.

[18] M. Jeannerod. The timing of natural prehension movements. *Journal of Motor Behavior*, 1984.

[19] R. S. Johansson. Sensory input and control of grip. In *Novartis Foundation Symposium*, pages 45–58, 1998.

[20] A. E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21:433–449, 1999.

[21] I. Kamon, T. Flash, and S. Edelman. Learning to grasp using visual information. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 2470–2476, 1996.

[22] J. Kim. M. eng. Master's thesis, Massachusetts Institute of Technology, 2007.

[23] A. T. Miller, S. Knoop, H. Christensen, and P. K. Allen. Automatic grasp planning using shape primitives. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1824–1829, 2003.

[24] L. Montesano and M. Lopes. Learning grasping affordances from local visual descriptors. In *IEEE International Conference on Development and Learning*, 2009.

[25] A. Morales, E. Chinellato, A. H. Fagg, and A. P. del Pobil. Using experience for assessing grasp reliability. *International Journal of Humanoid Robotics*, 1(4):671–691, 2004.

[26] G. Rizzolatti, R. Camarda, L. Fogassi, M. Gentilucci, G. Luppino, and M. Matelli. Functional organization of inferior area 6 in the macaque monkey. *Experimental Brain Research*, 71(3):491–507, 1988.

[27] G. Rizzolatti and G. Luppino. The cortical motor system. *Neuron*, 31(6):889–901, 2001.

[28] R. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (FPFH) for 3D registration. In *IEEE International Conference on Robotics and Automation*, 2009.

[29] M. Saric, C. H. Ek, and D. Kragic. Dimensionality Reduction via Euclidean Distance Embeddings. Technical report, KTH, Royal Institute of Technology, Stockholm, 2011.

[30] A. Saxena, J. Driemeyer, and A. Y. Ng. Robotic Grasping of Novel Objects using Vision. *International Journal of Robotics Research*, 27(2):157, 2008.

[31] B. Schölkopf and A. Smola. Kernel principal component analysis. *Artificial Neural Networks—ICANN'97*, 1997.

[32] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[33] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[34] K. B. Shimoga. Robot grasp synthesis algorithms: A survey. *The International Journal of Robotics Research*, 15(3):230, 1996.

[35] D. Song, C. H. Ek, K. Huebner, and D. Kragic. Multivariate Discretization for Bayesian Network Structure Learning in Robot Grasping. In *International Conference on Robotics and Automation*, pages 1–8. Royal Institute of Technology, 2011.

[36] J. D. Sweeney and R. Grupen. A model of shared grasp affordances from demonstration. In *International Conference on Humanoid Robots*, 2007.

[37] L. E. Zhang, M. Ciocarlie, and K. Hsiao. Grasp evaluation with graspable feature matching. In *RSS Workshop on Mobile Manipulation: Learning to Manipulate*, 2011.

A Probabilistic Framework for Task-Oriented Grasp Stability Assessment

Yasemin Bekiroglu, Dan Song, Lu Wang and Danica Kragic

***Abstract*—We present a probabilistic framework for grasp modeling and stability assessment. The framework facilitates assessment of grasp success in a goal-oriented way, taking into account both geometric constraints for task affordances and stability requirements specific for a task. We integrate high-level task information introduced by a teacher in a supervised setting with low-level stability requirements acquired through a robot's self-exploration. The conditional relations between tasks and multiple sensory streams (vision, proprioception and tactile) are modeled using Bayesian networks. The generative modeling approach both allows prediction of grasp success, and provides insights into dependencies between variables and features relevant for object grasping.**

I. Introduction

A lot of current work in robotics is inspired by human goal-directed behavior [18]. In humans, goal-directedness is obtained through multiple development stages, both through the sensorimotor *exploration* (trial and error) and through the *observation* of others interacting with the world (imitation learning) [24]. The former is addressing the problem of learning through self-experience in order to associate the sensorimotor signals to the direct motor effects. The latter involves human supervision, which is especially beneficial for efficient learning of complex tasks. Robotic approaches often focus on just one of these two aspects. Linking between the two is often through manual encoding [32] or applied to simple tasks [29, 22, 24, 21]. The main challenges originate from the differences in commonly adopted representations [15].

The gap between the representations is especially visible when dealing with robot grasping tasks. For example, if a robot is given a high-level task command, e.g., *pour me a cup of coffee*, it needs to make decision on which object to use, how the hand should be placed around the object, and how much gripping force should be applied so that the subsequent manipulation is stable. Several sensory streams (vision, proprioception and tactile) are relevant for manipulation. The problem domain and hence the state space becomes high-dimensional involving both continuous and discrete variables with complex relations. Traditional dynamic systems approaches in robotics e.g., [14] focus mainly on optimal planning and control of hand trajectories, hence the state space only includes kinematic parameters of the arm and hand. The relations between many grasping-relevant variables mentioned above can not be addressed simultaneously.

Probabilistic frameworks based on graphical models have proved to be powerful in various fields with high-dimensional complex problem domains [29, 6, 10, 21]. Graphical models encode the relations between variables through their probabilistic conditional distributions. Such distributions do not require the variables to have same underlying representations. Therefore, high-level symbolic variables such as task goals can be naturally linked to the low-level sensorimotor variables such as hand configuration. Furthermore, the model can be combined with the probabilistic decision making where grasp plan and control can be performed through inference even with noisy and partial observations [28].

Some recent work in the area [27] exploited these strengths and linked the grasp plan to the manipulation tasks through Bayesian networks (BNs). The work emphasized the geometric constraint of a task for planning grasps based on simulated vision inputs. Tasks, however, also require various manipulations: *pouring* needs rotating a bottle that contains liquid, and *hand-over* needs only parallel transportation. The stability demand therefore differs due to different manipulations requested by tasks.

In this paper, we integrate this task-dependency with stability assessment. A method combining self-exploration and supervision is implemented, where self-exploration enables the robot to learn about its own sensorimotor ability (how to grasp an object to stably lift and manipulate it), while human tutoring helps the robot to associate its sensorimotor ability to high-level goals. In particular, we use a probabilistic model to integrate the semantically expressed goal of a task with a set of continuous features. We present an extensive evaluation of the proposed approach on a real robot platform equipped with multiple sensory modalities (vision, proprioception and tactile). The results show that the proposed model accurately estimates grasp success both at the stage of planning (before execution in real environments) and during grasp execution.

II. Related Work

Planning and executing a grasp that is robust and stable is an important topic in grasp research (see [25] for a recent review). The quality measures of stability are mostly based on *force-closure* of a grasp wrench space. A force-closure grasp means that any disturbing external forces can be balanced by the forces applied at the contacts. However these approaches assume perfect knowledge of the contacts between the hand and the object, which is usually an unrealistic demand on real setups. On the other hand, experience based approaches where the robot learns good grasping configurations through real execution [5, 3, 20] have proved to be successful.

But a good grasp should not only be stable, it also needs to be suitable for the task, i.e., *what do you want to do after* *you lift the object*. Very few work has put effort on planning grasps in a goal-directed manner. Xue et al. [32] manually encoded the expertise about task semantics provided by a human tutor. A recent work [27] used Bayesian networks to learn the grasping task constraints that depends on a set of geometric attributes from both objects and grasps (e.g., hand positions). However manipulation tasks do not just concern geometric constraints. A *pouring* task not only requires the bottle opening to be unblocked, but also needs the grasp to be stable enough to rotate the bottle. We need to link task information with stability in real world scenarios.

A natural extension is to combine supervised task learning with experience-based stability learning. This allows stability to be assessed in a task-oriented manner. This is especially beneficial for energy-efficient control: when a task (e.g., *hand-over*) does not require strong grasping for difficult manipulations (e.g., waving for the *hammering* task), a relatively smaller gripping force can be applied. Combining task with stability was rarely studied. Some work [1, 17] defined task-related grasp quality measures which combined task knowledge with analytical stability measures used in traditional grasp stability studies. Such approaches therefore also suffer from partial and uncertain knowledge of the world in real setups.

Probabilistic learning is a powerful paradigm for modeling and reasoning about the noisy and uncertain real world data [29, 6, 10, 21]. For robot grasping, planning and control rely heavily on vision sensing with typically noisy and incomplete observations. Probabilistic approaches combining vision and tactile sensing [11] provided an on-line estimate of belief states which were used to plan the next action. Toussaint et al. [29] proposed a coherent control, trajectory optimization, and action planning architecture by applying the inference-based methods across all levels of representation. Montesano et al. [21] used Bayesian networks to learn object affordances, and applied them to goal-directed motion planning.

However, no one has proposed a model that addresses both task-oriented grasp planning and stability-oriented grasp execution in real environments. In this paper we close the learn-plan-execute loop where the robot learns task knowledge from human teaching, and grounds this knowledge in low-level sensorimotor systems through self-exploration (manipulating the object) in a real environment. We use Bayesian networks to model conditional relations between task and stability knowledge with a multitude of features from vision (simulated, in this work), proprioception, and tactile sensing. The generative modeling approach provides a flexible framework to guide detailed grasp planning and execution in a task-directed way.

III. Models

We use X to denote a set of features relevant for grasping tasks T. Here X comes from three groups of features, $\{O, A, H\}$, where O denotes object feature set, A action feature set and H haptic feature set. Detailed feature description can be found in Section IV-A. We propose to use a generative approach, the Bayesian network [23], to model this grasp space. The goal is to apply the model for both task classification $P(T|X)$ and inferring the distribution of one variable conditioned on a task and other variables $P(X_i|T, X_j)$. $P(T|X)$ predicts how likely a grasp will succeed for a task, and $P(X_i|T, X_j)$ conveys domain knowledge such as the expected value of a tactile feature given a task and an object. To evaluate BN's classification performance, we compare it with a discriminative approach, Kernel Logistic Regression (KLR). In this section, we provide an overview of the two modeling approaches.

A. Kernel Logistic Regression

Kernel Logistic Regression is a nonlinear probabilistic classification model. Given a class variable (in this paper, the task T) and the input feature set (in this paper $X \subsetneqq \{O, A, H\}$ as seen in Tab. I), KLR models the probability of the class variable $P(T|X)$ through a weighted sum of the similarities (kernels $\mathcal{K}$) between a testing point $\mathbf{x}$ and each training point $\mathbf{x}_i$ [33]:

$$p(\mathbf{t}|\mathbf{x}; \mathbf{w}) = \frac{1}{1 + \exp\{-\sum_{i=1}^{n} w_i \mathcal{K}(\mathbf{x}, \mathbf{x}_i)\}} \tag{1}$$

In this paper we choose $\mathcal{K}$ to be a Gaussian kernel. Training a KLR model is to find the weight vector $\mathbf{w}$ that maximizes the regularized probability of the data

$$-\sum_{i=1}^{n} \log p(y_i|x_i; w_i) + \eta \,\mathrm{trace}(\mathbf{w}\mathbf{K}\mathbf{w}^T) \tag{2}$$

where $\mathbf{K}$ is the kernel Gram matrix, with $K_{ij} = \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)$, and η is the regularization constant. During training, the kernel bandwidth parameters and η are chosen by cross-validation.

B. Bayesian Network

A Bayesian network [23] is a probabilistic model that encodes the joint distribution of a set of random variables $V = \{V_1, V_2, \ldots, V_m\}$. Each node in the network represents one variable, and the directed arcs represent conditional independencies. Given a structure of the network S and a set of local conditional probability distributions (CPDs) of each variable V_i, the joint distribution of all the variables can be decomposed as

$$p(\mathbf{v}) = p(\mathbf{v}|\boldsymbol{\theta}, S) = \prod_{i=1}^{m} p(\mathbf{v}_i|\mathbf{pa}_i, \boldsymbol{\theta}_i, S)\ , \tag{3}$$

where $\mathbf{pa}_i$ denotes the parents of node V_i, and the parameter vector $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_m)$ specifies the CPDs. Learning a BN includes discovering from a dataset: *1)* how one variable depends on others ($\boldsymbol{\theta}$), and *2)* what the conditional in-dependencies between different variables are (S). The former is an instance of parameter learning and the latter of structure learning. Various algorithms and techniques have been developed to learn a BN in different model and data conditions (see [9] for a review).

In this paper, we use the Bayesian network to model the joint distribution of a set of task and stability-relevant variables

Tab. I, i.e., $V = \{T, X\}$ where $X \subsetneqq \{O, A, H\}$. To correctly describe a manipulation task, both conceptual high-level information and continuous low-level sensorimotor variables are needed. The variables in this work are both discrete (e.g., $task, obcl$), and continuous (most O, A, H features). The continuous features such as hand grasp configuration $fcon$ can be high-dimensional with complex probabilistic distributions.

Learning BN structures from both continuous and discrete data is difficult, particularly when continuous data is high-dimensional and sampled from complex distributions. Most algorithms for structure learning only work with discrete variables. Therefore, a common approach is to convert the mixed modeling scenario into a completely discrete one by discretizing the continuous variables [7]. In this paper we use a two-step discretization scheme. For a high-dimensional continuous variable X, the data in original observation space is first projected to a low-dimensional intrinsic space, and then a parametric mixture model (multi-variate Gaussian mixture) is learned to model the data distribution in this intrinsic space,

$$p(\mathbf{x}) \propto \prod_{k=1}^{M} \lambda_k N(\mathbf{x}|\mathbf{u}_k, \Sigma_k^{-1}). \qquad (4)$$

where $\mathbf{u}_k$ and Σ_k are the mean and covariance of each Gaussian component, and λ_k is the mixing proportion. The parameters of the mixture model are learned using the standard EM approach. The number of the clusters for each variable is found through cross-validation where the task classification performance with the BN is maximized.

We use a greedy search algorithm to find the network structure (the directed acyclic graph, or DAG) in a neighborhood of graphs that maximizes the network score (Bayesian information criterion [26]). The search is local and in the space of DAGs, so the effectiveness of the algorithm relies on the initial DAG. As suggested by Leray and Francois [16], we use another simpler algorithm, the maximum weight spanning tree [4], to find an oriented tree structure as the initial DAG.

C. Inference in Bayesian Networks

A trained network defines factorization of the joint distribution of the observations, $p(V) = p(T, O, A, H)$, in terms of a network of conditional dependencies. We can now compute the posterior distribution of one or group of variables given the observation on others. A common way of this computation is to convert the graph into a tree, then apply the junction tree algorithm [12], an algorithm of local message passing to compute the distribution of the interests. The output of the network is in terms of a multinomial distribution over each of the discrete states of the network,

$$\mu_{ijk} = p(\mathbf{v}_i \to \mathbf{u}_{ik}|\Pi_i = \mathbf{U}_j). \qquad (5)$$

stating as "*the probability of variable V_i is at its discrete state $\mathbf{u}_{ik}$ when a set of other variables Π_i is observed to be at the state $\mathbf{U}_j$*". We will now describe how we can recover a continuous estimate from this distribution.

Each point on the low-dimensional intrinsic space corresponds to a point over the original observation space. Therefore in order to acquire a continuous estimate we need to determine a distribution over the intrinsic space associated with the multi-nominal distribution from the tree. During data discretization (see Eq. (4)) we have learned a mixture model $p(\mathbf{x})$ defining the continuous distribution of X_i in the intrinsic space that is not conditioned on other variables (see Eq. (4)). The multinomial distribution μ_{ijk} defines a conditional distribution over the discrete states of X_i. Combining μ_{ijk} with the mixture model results in the following conditional mixture model,

$$p(\mathbf{x}_i|\Pi_i = \mathbf{U}_j) \propto \prod_{k=1}^{M_i} \mu_{ijk} N(\mathbf{x}_i|\mathbf{u}_{ik}, \Sigma_{ik}^{-1}). \qquad (6)$$

We can then sample from the above distribution in order to find locations over the intrinsic space.

D. Generative Model

A Bayesian network is a generative model where not only the class probabilities $p(T|X)$ can be inferred as KLR, but also the class conditional distributions can be predicted $p(X|T)$. This means: we can apply a BN model in estimating the posterior distribution of a task given observations of the object and action features, i.e., to classify T. We can also find, given an assigned task, the posterior distribution of the object features. This provides an evaluation of the task affordance of an object, hence allowing the robot to select objects in complex environments.

In addition, Bayesian networks allow us to infer the domain knowledge through data. The network structure depicts an influence diagram illustrating the conditional relations between different variables. Also the class conditional on feature variables provides an intuitive evaluation of task and stability-related requirements.

Another strength of the BN is its ability to infer the grasp success with partial observation. In the scenarios of task-based grasp adaptation (see Fig. 7), this is especially important because we can predict grasp success given observation on only object features and grasp parameters planned in the simulation environment. Grasp replan therefore can be initiated without having to execute a bad grasp using real robot platforms. Though this can also be done using discriminative models, each observation condition requires training of a separate model.

IV. Data Collection

A. Feature Description

Tab. I shows the most important features used in this work. T denotes task and it is labeled by a human tutor by observing the generated grasps both in simulation and in the real world. O and A represent the object and action feature sets respectively. They are extracted from the grasp planner in the simulation environment. H represents the haptic features which are obtained when the grasp is executed with the robot on real objects.

The object features include object class identity *obcl*, the three dimensional $size$, and overall convexity shape $cvex$. The action features A describe the hand pose (position and

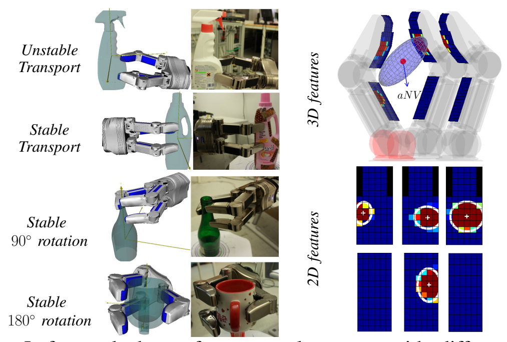


Fig. 1. Left panel shows four example grasps with different stability conditions. Right panel illustrates tactile related features.

TABLE I
FEATURE SET WITH DIMENSIONALITY D (LOW/HIGH) AND NUMBER OF DISCRETE STATES M (OPTIMIZED FOR EACH OF THE THREE TASKS [HAND-OVER, POURING, DISHWASHING]).

	Name	D	M	Description
T	$task$		2	Binary task identifier
O_1	$obcl$		2	Object class
O_2	$size$	3		Object dimensions
O_3	$cvex$	1	[5, 5, –]	Convexity value $[0, 1]$
A_1	dir	4		Quaternion hand orientation
A_2	$npos$	3	16	Unit grasping position
A_3	rad	1	[15, 14, 14]	Radius of $npos$
H_1	$fcon$	2/7	[7, 7, 7]	Final hand configuration
H_2	iG	5/30		2D pressure distribution
H_3	iC	3/12	[11, –, –]	2D pressure centroid
H_4	pG	3/9		3D pressure distribution
H_5	pC	3		3D pressure centroid
H_6	aNV	2/3	[–, 5, 5]	Average normal vector
H_7	pOV	2/3		Pressure orientation vector
H_8	pOA	2		Pressure orientation angle

orientation) in the object-centered coordinate system. We decompose the grasp position into a unit sphere $npos$ and the radius rad for visualization purpose in the inference results.

For haptic features H, we have the final hand configuration $fcon$ from proprioception when a grasp is completed, and a set of tactile-related features (see Fig. 1) derived from the 6 tactile sensors on the Schunk dextrous hand. iG carries information about the contact centroids (iC) and the distribution of the pressure in the vertical and horizontal directions locally for each sensor array. It is calculated based on image moments up to order 2. pG is the 3D version of iG with respect to the wrist frame and therefore includes the overall contact centroid (pC) and pressure distribution considering all sensors. aNV is the 3D normal vector weighted by the tactile readings, pOV is the unit vector of aNV, and pOA represents the spherical coordinates of pOV.

We emphasize that the grasp representation does not have to be non-redundant, e.g., iG contains information of e.g., iC. Such an "over-representation" of the featured variables allows us to select most "representative" variables for efficient learning and inference, and also to use BNs to identify the importance of, and dependencies between these variables in the scenarios of robot grasping tasks.

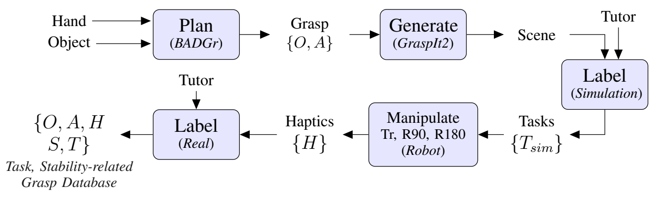


Fig. 2. Data Generation Process: The three tasks are: hand-over, pouring and dishwashing, each of which has to satisfy one of the stable manipulations: transport (Tr), 90° rotation (R90), and 180° rotation (R180), respectively.

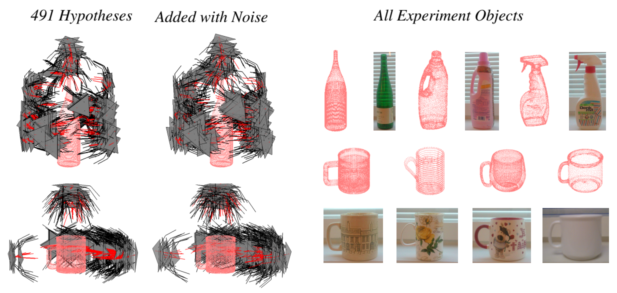


Fig. 3. Data Collection: The left panel shows all grasps generated on the two classes of objects (mugs and bottles). The right panel shows all the objects.

B. Data Generation Process

The hand we use in experiments is the 7-dof Schunk dextrous hand equipped with tactile array sensors. The hand is attached to the 6-dof Kuka arm that is mounted on a robust shelf. We have a corresponding Schunk hand model in GraspIt! [19] for grasp planning and feature extraction in simulation environment. The objects we have used are typical home-environment objects consisting of three bottle instances and four mug instances. The 3D models capture the similar sizes and shapes of the real objects as shown in Fig. 3.

Fig. 2 shows the schematic of the data generation process. To extract those features in Tab. I, we first generate grasp hypotheses using the grasp-planner BADGr [13], and evaluate them as scenes of object-grasp configurations in a grasp simulator, GraspIt! [19]. Each grasp hypothesis is first visualized in GraspIt! by a human tutor who associates it with a task label from simulation (T_{sim}). Then the hypothesis that is good for at least one task T_{sim} is used on the robot platform to perform a set of manipulations on the corresponding real object (see Fig. 1). The human tutor assesses the stability for each manipulation. If a grasp that is previously considered to be good for a task, e.g., *pouring* (by label T_{sim}) results in unstable 90°rotation which is a required manipulation for *pouring* task, then it will be considered to be bad for *pouring* in the final task label T. In this paper, we experiment with three tasks: *hand-over*, *pouring*, and *dishwashing*, each of which has to satisfy one of the three manipulations: *transportation* (Tr), 90°*rotation* (R90), and 180°*rotation* (R180) respectively. Note that these conditions are determined heuristically based on our life experience, e.g., since dishwashing needs to place a mug upside-down into the dishwasher machine, we need to be able to stably rotate the object 180°.

Our robot platform implements an inverse-kinematics path planning given the target hand pose. And the grasping is position-controlled given the final hand configuration. During data generation, our goal is to execute the planned hypotheses around the object (see Fig. 3). To avoid that some grasps is not reachable, we place the object in a known location in front of the robot, and manually rotate the object along the vertical axis by a 45° increment to place the hypotheses in the robot's working space.

Because of the uncertainty introduced in both the motor system and the manual placement, the real hand pose A will not precisely represent the values generated in simulation. This uncertainty is simulated by adding the Gaussian noise to the variables. Fig. 3 shows all the grasps generated on bottles and mugs in both clean (left) and noisy (right) versions. The resulting grasping position will have noise with standard deviation about $0.4 \rightarrow 1.1$ (cm) in the three dimensions.

V. Model Selection

Model selection is a process including three steps: 1) dimension reduction, 2) variable selection (using the low-dimensional representation), and finally 3) optimizing data discretization.

We employ a nested stratified cross-validation design throughout for all the classification models, in which the outer loop of cross-validation estimates the performance of the optimised classifiers while the inner loop is used to find the best parameter configuration for each classifier. In both loops 10-trial hold-out split is used with 20% as the testing data. We use the area (AUC) under the Receiver Operator Characteristic (ROC) curve as the performance metric for both KLR and BN. The classifiers' outputs $p(T|X)$ are thresholded to derive the ROCs.

1) Dimensionality Reduction: There are many techniques for dimension reduction (see [31] for a review). Ideally a cross-validation process should be used to select optimal technique and their parameters. However in this paper, we have many steps for model selection, a full-scale model selection will be very expensive. Considering the main focus of the paper is not to evaluate dimension reduction techniques, we decide to select a single method. We choose Kernel PCA because its capability to model non-linear manifolds which is a character of our problem domain. The intrinsic dimensionality of a given variable is determined automatically using a maximum likelihood-based approach [30]. Tab. I shows the intrinsic dimensions together with their original dimensionality on a set of variables.

2) Variable Selection: We use the HITON algorithm [2] to perform the optimal variable selection for the three tasks. HITON works by first inducing the Markov Blanket of the target variable to be classified. In this paper the target is the binary task variable T, and its Markov Blanket is denoted by $MB(T)$. Then support vector machine is used to further remove the unnecessary variables in the $MB(T)$ in a greedy hill-climbing fashion. The performance metric is the task classification rate. Exhaustive search through all subsets of features returned in $MB(T)$ is prohibitive, so we adopt a set of heuristics to form a smaller search space: 1) the subset must include $obcl$ and $npos$ because we are interested to infer the conditionals involving these variables, 2) there must be at most two features in each of the O, A and H feature sets. We adopt a stopping point at a 95% threshold of classification accuracy. The subset of features with the highest score discovered up to this point is selected as the satisfactory set of features. Fig. 4 shows which variables have been selected for each of the three tasks.

3) Optimizing Data Discretization: This is a step for only Bayesian networks. The structure learning requires discrete data. However, this leads to loss of information. When the resolution is low (i.e., a few discrete states), the variance in the original continuous domain that is discriminative may be smoothed out. On the other hand, for the variables that are not discriminative, a high resolution will jeopardize the classification performance due to the curse of dimensionality. We therefore want to find an optimal granularity M in Eq. (4)), on the variables of interest ($\{cvex, rad, fcon, iC, aNV\}$). The optimal granularity maximizes the task classification performance (AUCs) with the BNs. Tab. I shows the resulting M for each of the three tasks.

VI. Model Evaluation

We evaluate the Bayesian network-based modeling framework in two aspects: classification performance, and how we can use the generative model for understanding the problem domain.

For classification performance, we compare the BNs with the discriminative model KLR under two observation conditions: the partial observation when only simulated object and action variables are observed ($T|O, A$), and the full observation when haptic information also is available after grasp execution in the real environment ($T|O, A, H$). We perform this over 50 trials with 20% hold-out splits. In each trial, we train three models: 1) $KLR(O, A, H)$ with all selected variables, 2) $KLR(O, A)$ with only simulated variables, and 3) $BN(O, A, H)$ with all selected variables. We do not need to train $BN(O, A)$ because task probability can be inferred in BNs with partial observations. When training KLRs, we use the continuous low-dimensional representation. And when training BNs, we use the optimal discrete data. In each trial, both structure and parameters of the BNs are learned. Since each trial uses different set of training data, the resulting structure can be different.

We then show the inference results on some interested variables ($npos$, and a selected H feature). We choose $npos$ because it represents from which direction the hand is placed relative to the object, and therefore is a very intuitive variable to exhibit task constraints. We choose one tactile-related feature in order to show that the BN can be used to produce an expectation over sensor images given task conditions. For each variable, we evenly sample a set of points $\mathbf{x}$ in the low-dimensional space for easy visualization. For each sampled point, a conditional likelihood is obtained given the 3 tasks and object class $p(\mathbf{x}|task, obcl)$ to generate the *likelihood maps* (Fig. 6).

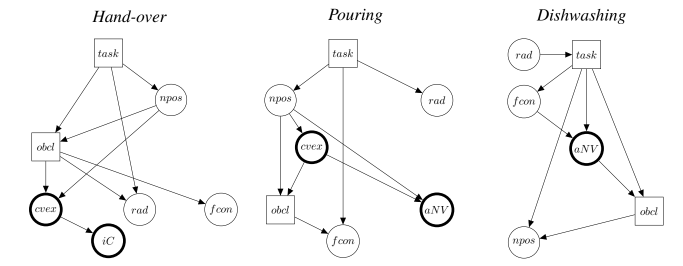


Fig. 4. Structure of BNs: DAGs of the three BNs each of which models one binary classification of one task. Square nodes represent discrete variables and circled nodes continuous ones. The differences in variable selection among the three tasks are highlighted by thick border of the nodes.

A. Network Structures

Fig. 4 shows the best Bayesian network structures (DAGs) for the three tasks. The nodes represent the variables selected using HITON algorithm. The differences in selected variables are highlighted by the thick-bordered nodes.

When we look at the haptic features H, *hand-over* selects iC, whereas *pouring* and *dishwashing* both select aNV. iC is a 2D feature characterizing the pressure centroid of each tactile sensor pad on the fingers, whereas aNV is a 3D feature that summarizes the overall pressure distribution considering also the finger configurations. In other words, aNV encompasses stronger information that may be relevant to stability especially when task demands stronger grasping like *pouring* or *dishwashing*.

As to the network structure, all three tasks have direct conditional relations with $npos$ and rad. This is natural since the position of the hand relative to the object is an important factor influencing both the affordance of a task (from which direction to approach the object $npos$), and its stability requirements (how far away the hand is from the object center of mass rad). For *dishwashing* T is directly connected to aNV, whereas for *pouring* T influence aNV through $npos$. This may be due to that *dishwashing* requires a manipulation with 180° rotation, which, compared to 90° rotation for *pouring*, is much more demanding in terms of grasp stability. So the task success for *dishwashing* depends on aNV even if the $npos$ is also observed.

B. Classification

Fig. 5 shows the ROC curves for task classification results averaged over 50 trials. Tab. II shows the mean and standard deviation of the AUCs. In general, the BNs with partial observations have good classification performance for all 3 tasks (average AUC around 0.86). Under full observation, KLR models perform a lot better than BNs. However, we note that when haptic features are not observed, KLR's performance drops a lot compared to BNs. To confirm this, we conduct a two-sample t-test on AUC scores over 50 trials of the experiment. The hypothesis is: "*The classification performance with full observation is* 0.07 *greater than the performance with partial observation*", briefed as "full $\triangle 0.07 >$ partial". The results show that at the significance level 0.05, the hypothesis is accepted for the KLR, but rejected for BN. In other words,

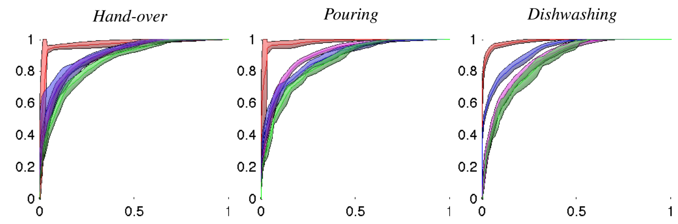


Fig. 5. Classification: The average ROC curves for three tasks. Red is KLR with full observation (O, A, H). Pink is KLR with partial observation (O, A). Blue is BN with full observation (O, A, H). Green is BN with partial observation (O, A). The transparent regions represent the one standard deviation of the true positive rate.

TABLE II
MEAN AND STANDARD DEVIATION OF AUCS FOR THE THREE TASKS. SAMPLE SIZE (NSAMPLES) FOR EACH TASK IS ALSO SHOWN.

Task (Nsamples)	KLR full	KLR partial	BN full	BN partial
Hand-over (1026)	0.97 (0.01)	0.90 (0.01)	0.90 (0.04)	0.86 (0.01)
	$\triangle 0.07 > *$			
Pouring (1143)	0.98(0.01)	0.90 (0.01)	0.88 (0.02)	0.86 (0.02)
	$\triangle 0.07 > *$			
Dishwashing (831)	0.98 (0.01)	0.87(0.02)	0.92 (0.01)	0.86 (0.02)
	$\triangle 0.07 > *$			

KLR with partial observation performs similar to BN with both observation conditions. Looking closely, when haptic features are not observed, the performance drop for *dishwashing* task in the BN is higher than for the other two tasks. This is related to the differences in the task demand of grasping stability which has explained the structural differences before (Fig. 4). For example, when aNV is not observed in *dishwashing*, $p(T|X)$ loses more useful information than in *pouring*.

C. Inference

Fig. 6 exhibits the likelihood maps of two features. The brighter color indicates higher probability. On the left side shows the results on $p(npos|task, obcl)$. For *pouring* task, one should not grasp from the top of the mug or bottle, which is reflected by the dark color on the $npos$ sphere, but top grasps are allowed for *hand-over* task. Among the two object classes, only mug affords *dishwashing* task, which is indicated by complete dark maps.

On the right side shows the results of $p(iC|task, obcl)$ for *hand-over* task and $p(aNV|task, obcl)$ for the other two tasks. We observe clear differences in these "haptic images" both between two different object classes, also between different tasks. This reflects different "haptic expectations" given task conditions.

D. Model Application

We conclude the paper by a task-oriented, stability-based grasp adaptation scenario. The goal is to demonstrate one way of applying the proposed probabilistic framework. Fig. 7 depicts a two-step grasp adaptation process, where the first step predicts if a planned hypothesis affords an assigned task (from the simulated O, A features) before it is executed on real robots, and the second step predicts if the grasp affords manipulation demanded by the task once the grasp has been executed. Here the haptic sensory inputs H are available

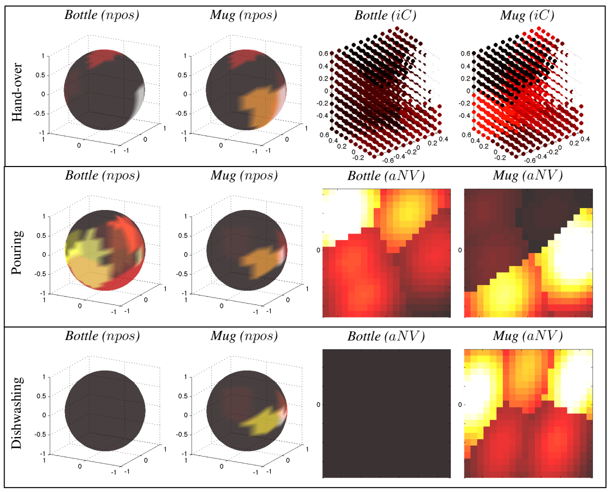


Fig. 6. Inference: Likelihood maps of the continuous variables conditioned on task and object class. Left side shows $p(npos|task, obcl)$ for all the three tasks. On the right side, $p(iC|task, obcl)$ is obtained for *hand-over* and $p(aNV|task, obcl)$ is obtained for the other two tasks.

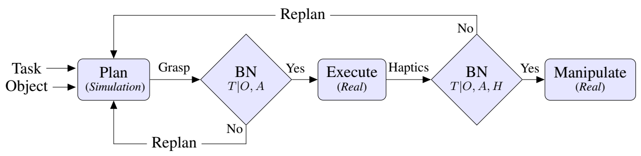


Fig. 7. Application Diagram: Task-based grasp adaptation.

which allows more accurate prediction with full observation $p(T|O, A, H)$ before the object is lifted. Such a *double-guarded* system is beneficial to efficiently plan and execute the robot grasping.

Fig. 8 demonstrates a grasp adaptation process for the input *pour with this detergent bottle*. The top row shows the grasp hypotheses sequentially produced by a planner. Before they are executed on the real robot platform $p(T|O, A)$ rejected the first 3 hypotheses. This is reflected by the location of data point (green dot) in the dark region of $npos$ likelihood maps. The grasp replan is triggered until the fourth hypothesis is found to be good for grasp execution. It is however predicted to fail under full observation $p(T|O, A, H)$ (aNV is in the dark region of the likelihood map). A replan is again triggered until a good grasp is found with full observation.

VII. Conclusion

We have proposed a unified probabilistic framework using Bayesian networks to assess grasp stability in a task-oriented manner. The framework combines human supervision and self-exploration during manipulation to encode task-dependent stability requirements. The learned network could successfully predict task performance of a grasp both in terms of its geometric requirements and in terms of the stability demands for the subsequent manipulations. This is very important contribution. Since the high-level task goals are seamlessly linked to low-level haptic sensory outputs, grasp plan and control become more efficient and goal-oriented. In addition, the generative model allows us not only to predict task success, but also convey domain knowledge. We can infer structural dependencies between different variables, and form conditional expectations on interested features.

The work opens a broad avenue for future research. Firstly the current system considers only a single time instance at grasp completion. Temporal data is certainly necessary if on-line grasp adaptation and control is needed. We can explore dynamic models such as Dynamic Bayesian Networks for this purpose. Secondly robot grasping is not stationary. Task requirements may change in different time, contexts or environments. Sensory signals may also change over time. Therefore the model needs to be updated incrementally. Learning algorithms that allow incremental data discretization and structure update of BNs are needed. Finally, the current system does not consider dynamic control problems and the reaching motion. Following the idea of [29], the current likelihood map of the grasping position $npos$ that encodes task affordances can be combined with the reachability map in [8] to compose a task-oriented path plan, trajectory optimization, and grasp control system.

Acknowledgments

This work was supported by EU IST-FP7-IP GRASP and FP7-IP-027657 CogX. The authors would like to thank Renaud Detry and Carl Henrik Ek for their help in structuring the paper.

References

[1] J. Aleotti and S. Caselli. Interactive teaching of task-oriented robot grasps. *Robotics and Autonomous Systems*, 58(5):539–550, 2010.

[2] C. F. Aliferis, I. Tsamardinos, and A. Statnikov. HITON: a novel Markov Blanket algorithm for optimal variable selection. *AMIA Annu Symp Proc.*, pages 21–25, 2003.

[3] Y. Bekiroglu, R. Detry, and D. Kragic. Learning tactile characterizations of object- and pose-specific grasps. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1554–1560, 2011.

[4] C. Chow and C. Liu. Approximating Discrete Probability Distributions with Dependence Trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.

[5] R. Detry, E. Başeski, M. Popović, Y. Touati, N. Krüger, O. Kroemer, J. Peters, and J. Piater. Learning Continuous Grasp Affordances by Sensorimotor Exploration. In *From Motor Learning to Interaction Learning in Robots*, pages 451–465. Springer-Verlag, 2010.

[6] B. Douillard, D. Fox, F. Ramos, and H. D. Whyte. Classification and Semantic Mapping of Urban Environments. *The Int. Journal of Robotics Research*, 30(1):5–32, 2010.

[7] L. D. Fu and I. Tsamardinos. A Comparison of Bayesian Network Learning Algorithms from Continuous Data. In *AMIA*, 2005.

[8] M. Gienger, M. Toussaint, and C. Goerick. Task Maps in Humanoid Robot Manipulation. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 2758–2764, 2008.

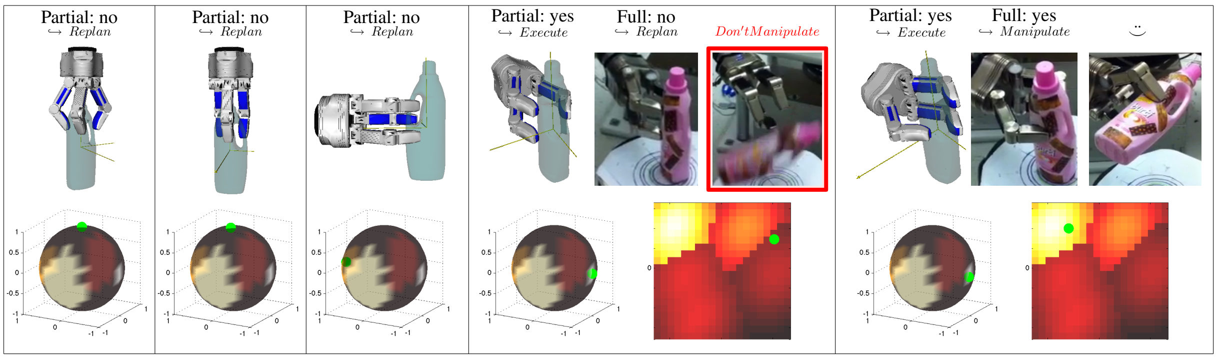


Fig. 8. Application: Two-loop grasp adaptation when task is *pouring* with the detergent bottle following the flowchart in Fig. 7.

[9] D. Heckerman. A Tutorial on Learning With Bayesian Networks. Technical report, Microsoft Research, 1996.

[10] T.M. Hospedales and S. Vijayakumar. Structure inference for Bayesian multisensory scene understanding. *IEEE transactions on pattern analysis and machine intelligence*, 30(12):2140–2157, 2008.

[11] K. Hsiao, L. Kaelbling, and T. Lozano-Perez. Task-Driven Tactile Exploration. In *Robotics: Science and Systems*, 2010.

[12] C. Huang and A. Darwiche. Inference in Belief Networks: A Procedural Guide. *Int. Journal of Approximate Reasoning*, 15:225–263, 1994.

[13] K. Huebner. BADGr - A Toolbox for Box-based Approximation, Decomposition and GRasping. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems: Workshop on Grasp Planning and Task Learning by Imitation*, 2010.

[14] A. J. Ijspeert, J. Nakanishi, and S. Schaal. Movement Imitation with Nonlinear Dynamical Systems in Humanoid Robots. In *IEEE Int. Conf. on Robotics and Automation*, pages 1398–1403, 2002.

[15] N. Krüger, C. Geib, J. Piater, R. Petrick, M. Steedman, F. Wörgötter, A. Ude, T. Asfour, D. Kraft, D. Omrčen, A. Agostini, and R. Dillmann. ObjectAction Complexes: Grounded abstractions of sensorymotor processes. *Robotics and Autonomous Systems*, 59(10):740–757, October 2011.

[16] P. Leray and O. Francois. BNT Structure Learning Package: Documentation and Experiments. Technical report, Université de Rouen, 2006.

[17] Z. Li and S. Sastry. Task oriented optimal grasping by multifingered robot hands. In *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, volume 4, pages 389–394, January 2003.

[18] A. N. Meltzoff. *Elements of a Developmental Theory of Imitation*, pages 19–41. Cambridge University Press, Cambridge, MA, USA, 2002.

[19] A. T. Miller and P. K. Allen. GraspIt! A Versatile Simulator for Robotic Grasping. *IEEE Robotics and Automation Magazine*, 2004.

[20] L. Montesano and M. Lopes. Learning Grasping Affordances from Local Visual Descriptors. In *IEEE Int. Conf. on Development and Learning*, 2009.

[21] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor. Learning Object Affordances: From Sensory–Motor Coordination to Imitation. *IEEE Transactions on Robotics*, 24(1):15–26, 2008.

[22] E. Oztop, D. Wolpert, and M. Kawato. Mental State Inference using Visual Control Parameters. *Cognitive Brain Research*, 22(2):129–151, 2005.

[23] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[24] R. Rao, A. Shon, and A. Meltzoff. A Bayesian Model of Imitation in Infants and Robots. In *Imitation and Social Learning in Robots, Humans, and Animals*, pages 217–247, 2004.

[25] A. Sahbani, S. El-Khoury, and P. Bidaud. An overview of 3D object grasp synthesis algorithms. *Robotics and Autonomous Systems*, 60(3):326 – 336, 2012.

[26] G. Schwarz. Estimating the Dimension of a Model. *Annals of Statistics*, 6(2):461–464, 1978.

[27] D. Song, K. Huebner, V. Kyrki, and D. Kragic. Learning Task Constraints for Robot Grasping using Graphical Models. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2010.

[28] M. Toussaint, L. Charlin, and P. Poupart. Hierarchical POMDP Controller Optimization by Likelihood Maximization. In *Conf. on Uncertainty in Artificial Intelligence (UAI)*, 2008.

[29] M. Toussaint, N. Plath, T. Lang, and N. Jetchev. Integrated motor control, planning, grasping and high-level reasoning in a blocks world using probabilistic inference. In *IEEE Int. Conf. on Robotics and Automation*, 2010.

[30] L. J. P. van der Maaten. Matlab Toolbox for Dimensionality Reduction (v0.7.1b), 2010.

[31] L. J. P. van der Maaten, E. O. Postma, and H. J. van den Herik. Dimensionality Reduction: A Comparative Review, 2008.

[32] Z. Xue, J. Zoellner, and R. Dillmann. Automatic Optimal Grasp Planning based on Found Contact Points. In *IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics*, pages 1053–1058, 2008.

[33] M. Yamada, M. Sugiyama, and T. Matsui. Semi-supervised speaker identification under covariate shift. *Signal Processing*, 90(8):2353–2361, 2010.