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

DR 2.2: Active Vision, Learning and Manipulation

Michael Zillich, Thomas Mörwald, Andreas Richtsfeld, Kai Zhou, Johann Prankl, Markus Vincze, Alper Aydemir, Oscar J. Rubio, Danica Kragic, Marek Kopicki, Damien J. Duff, Rustam Stolkin

TUW, Vienna
⟨`zillich@acin.tuwien.ac.at`⟩

Due date of deliverable:	July 31 2010
Actual submission date:	July 30 2010
Lead partner:	TUW
Revision:	final
Dissemination level:	PU

The content of WP2 is defined rather broadly as Object perception and manipulation. Perception encompasses active search strategies and attention, object detection, recognition and tracking. Regarding object manipulation we investigate grasping of known and unknown objects, as well as pushing of objects in order to learn predictive physics models. For year 2 we can report progress on a selection of these sub-tasks as well as integration. Accordingly the work reported in this deliverable falls into several groups, namely vision routines, modular motor learning from push interactions, and grasping of learned objects.

Executive Summary

WP2 groups a large number of different tasks, from vision to learning and manipulation. All these tasks revolve around objects, their detection. learning appearance and physical models and interacting with these objects. Incompleteness of knowledge permeates all of these tasks. Objects are initially seen only from one side and the system needs to actively complete appearance models for the occluded parts. Searching for known objects, with all the variations in position, scale and lighting, can be dramatically improved by actively guiding the search using knowledge about learned room layouts and typical object locations. Learning the physical behaviour of objects requires lots of experimental interactions, each of which is perturbed slightly by tiny details of the physical world like specks of dirt changing local friction coefficients. The system thus must incorporate all these non-deterministic observations into a physics model that can express not only its predictions about object movements but also its uncertainty about what it has learned. Moreover object models are rarely completely accurate (think of slightly bulging sides, or rounded-off edges of a cereal box). Hardly visibly noticeable, these details can be crucial for obtaining a stable grasp on an object. So the inherently incomplete visual information needs to be augmented with tactile feedback in order to get a better understanding of grasp stability.

Reflecting these different sub-tasks, this deliverable groups several reports on vision routines for active visual search, modular motor learning, and grasping of learned objects.

Vision routines for active visual search

An actively exploring robot poses fundamentally different vision problems than e.g. object recognition in a data base. When searching a given object in a cluttered real world scene it is crucial to first obtain good views that are likely to contain the object. Most vision methods become significantly simpler once the object is likely to be big and centered in the image. To this end we reviewed several approaches to active visual search. Supporting surfaces are places likely to contain objects. So we extended our prior work on a plane pop-out attentional operator to handle more complex cases with multiple planes in a 3D stereo point cloud. Regarding the searched objects themselves we investigated methods to incrementally learn object models, while tracking objects and observing a dynamic scene.

Role of active visual search in CogX

Active visual search constitutes a prime example of a knowledge gathering action. It starts with weak hypotheses of where to find objects, based on general principles (objects are often resting on supporting surfaces) as well as task knowledge (certain rooms are more likely to contain certain objects). These then need to be actively confirmed or rejected by devising a search strategy to optimally explore the environment. Location is one aspect of object knowledge and active visual search deals with obtaining that knowledge in a principled manner. Object geometry and appearance are other aspects of object knowledge. Being able to to build object models incrementally, based on knowledge about their completeness (see DR.2.3) also constitutes an important part of a system that continuously extends itself.

Contribution to the CogX scenarios and prototypes

Locating objects in a large environment is an integral part of Dora the exploring robot. Also for Dexter and George objects need to be located, however in smaller workspaces, i.e. typically on table tops.

Modular motor learning

This part of the CogX project has looked at ways of applying machine learning to the problems of pushing manipulation with a single finger. The main thrust of the research has been the development of a modular prediction system, in which multiple learned experts (or "modules") combine their beliefs about what kinds of work-piece motions will result from various applied pokes or pushes of the robot finger.

Role of modular motor learning in CogX

Push prediction fits into the CogX aims of self understanding and self extension in two main ways. Firstly the learned modular predictor can detect a knowledge gap by inferring that none of the predictors, that it has trained on previously experienced contexts, is making good predictions about the present context (a "context" being a particular kind of object motion behaviour in response to applied pushes). Secondly, the system can then decide to train a new predictor or predictive expert module on the new context, thereby filling that knowledge gap. Related material is discussed in more detail in DR 2.3.

Contribution to the CogX scenarios and prototypes

The pushing prediction work forms the basis of the Dexter scenario, in which the robot is tasked with learning about the world through a series of experimental pushes applied to various objects. Dexter receives information about the effects of his pushing actions, by observing the resulting work-piece motions with a vision system. We are interested in pushing because it is a particularly simple and fundamental kind of manipulation. Pushing with a single finger is also fundamentally a part of more complex grasping with multiple fingers. Furthermore, we regard pushing as merely one example domain, and the learning techniques might be applied more generally to enable machines to make predictions about the motions of interacting bodies under other circumstances.

Grasping learned objects

For the second period of the project, we have continued with the use of early grasping strategies and grasp generation on known and unknown objects and the assessment of grasp stability using tactile sensing.

Role of grasping in CogX

Grasping generic objects is a notoriously hard problem that can not be addressed simply by analytic or pre-programmed solutions. Noisy and incomplete sensor data (unseen back sides of objects, unknown friction parameters) and the abundance of different object shapes and matching grasps require a learning based approach. Estimation of the likelihood of grasp stability based on visual and haptic feedback poses an interesting problem in self understanding and extension.

Contribution to the CogX scenarios and prototypes

Grasping is one part of the Dexter scenario where the robot learns about the world by interacting with objects (the other part being simpler - in terms of contact relationships - manipulations like pushing).

1 Tasks, objectives, results

As outlined in the executive summary, this deliverable presents work carried out in year 2 on the various sub-tasks contained in WP 2, covering a wide area between vision, learning and manipulation. While the following three sections report on individual sub-tasks separately we want to emphasise that they all contribute to their respective aspects of a system that self-understands and self-extends, be it by augmenting visual with haptic information, by building incremental visual models or by learning predictive physical models based on observing these visual models.

1.1 Vision routines for active visual search

1.1.1 Planned work

Work on visual routines concerns Tasks 2.3 and 2.4:

> **Task 2.3:** *Active segmentation. Use haptic information, pushing and grasping actions i) for interactive scene segmentation into meaningful objects, and ii) for extracting more detailed object models (visual and haptic). Also use information inside regions (surface markings, texture, shading) to complement contour information and build denser and more accurate models. (M16 - M39)*

Planned work from Task 2.3 for this reporting period up to month 27 was to build more complete object models by accumulating features while actively viewing objects from different views.

> **Task 2.4:** *Active Visual Search. Survey the literature and evaluate different methods for visual object search in realistic environments with a mobile robot. Based on this survey develop a system that can detect and recognise objects in a natural (possibly simplified) environment. (M1 - M27)*

Planned work on Task 2.4 (Active Visual Search) was to i) produce a document that presented a survey of the methods used in the literature for visual search, ii) make an initial implementation to test some of the ideas.

1.1.2 Actual work performed

Regarding Task 2.3 (Active segmentation) we concentrated our work on complementing edge based object models already used for tracking (and presented in DR.2.1) with distinctive texture based features. Using distinctive features allows recognition of objects in complex scenes, where purely edge-based methods can easily get lost in clutter. We build these models incrementally while they are already being tracked. This allows online active refinement of object knowledge. Observation currently however is still passive, as we are only beginning to exploit results in pushing and grasping of objects and work on *active* observation will be addressed in the later half of Task 2.3 (months 28 - 36).

Figure 1 gives an overview of the steps involved. We start by detecting basic 3D shapes such as boxes or cylinders based on perceptual grouping of edges. Alternatively we can also supply a more complex wireframe 3D model (beyond basic shapes), e.g. as found on google 3D warehouse (`http://sketchup.google.com/3dwarehouse`). We then start tracking the wireframe and augment it with texture edges which adds robustness to the tracker. This is not sufficiently robust however to detect the object in a cluttered (edge-rich) environment. So we place SIFT features extracted from the live image and place them on the 3D surface of the model. These SIFT features with their associated 3D position and orientation on the object are then used in a RANSAC based robust pose estimation procedure to detect the object and recover its 3D pose. The procedure is robust to scene clutter, scale changes and partial occlusions.

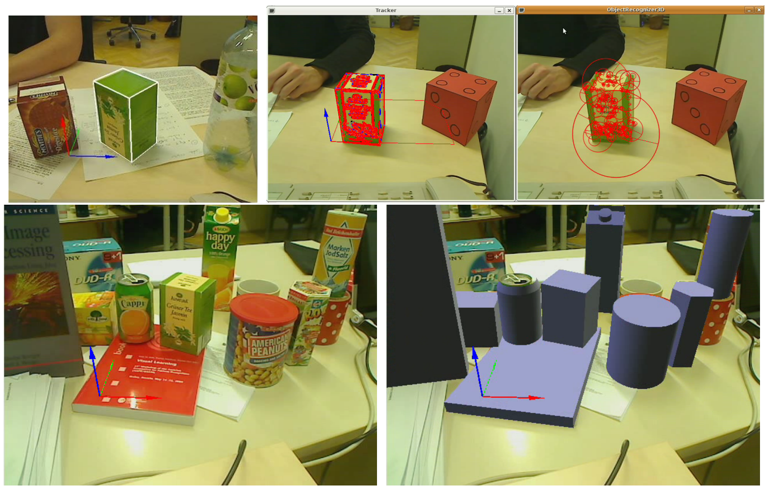


Figure 1: Edge based detection and tracking (top left and center) and SIFT feature on the object (top right). Recognition results for 10 objects (bottom)

Object poses are expressed as probability density functions and represented using a particle filter. This allows us to express not only small uncertainties regarding the exact object pose but also several distinct tracking solutions in the form of multi-modal distributions. Uncertainty about object identity is expressed via a confidence measure based on the number of correctly matched SIFT features.

The above chain of processing is implemented as a collection of CAST components and was also released to the public domain as a stand-alone toolbox (BLORT - The Blocks World Robotic Vision Toolbox) and was reported in [27, 21] (Annexes 2.1 and 2.2 respectively). The toolbox can be downloaded at `http://users.acin.tuwien.ac.at/mzillich/?site=4`

We also continued work on view-based object recognition. Contrary to the approach presented above we do in this case not assume a known object shape, but rather collect features of views of the object (we again use SIFT features). We only assume a SOI, a space of interest around the object, as given by the plane pop-out attentional operator. This approach is thus more general and complements the above shape based approach in more difficult cases where object shape is not available. The output of recognition is a distribution over object identity and view and is thus allows to detect an instance of an object but is not suited for manipulating objects, where exact shape and 3D pose are required.

Segmenting generic unknown objects from a scene is a notoriously difficult problem. Observing dynamic scenes over a sequence of images offers the possibility to group scene elements based on consistent common motion. In [25] (Annex 2.3) we developed a method for segmenting and tracking piecewise planar objects from a dynamic scene. The approach is based on identifying homographies using interest points. The main idea is to embed Minimal Description Length (MDL) based model selection in an iterative scheme. Thus existing planes compete with newly created plane hypotheses to ensure that interest points are assigned to the best currently available hypothesis. Additionally hypothesis generation can be guided to unexplained regions. This method avoids the bias towards dominant planes typical for other iterative methods (such as iterative RANSAC), and it limits the search space which leads to a faster explanation of the entire image in terms of piecewise planar surfaces.

In a closely related work we identify multiple plane hypotheses in depth images, again avoiding troublesome iterative RANSAC schemes. We use particle swarm optimisation to simultaneously search for multiple solutions for a plane fit in 3D point cloud data. Particles converging on tentative solutions create "repulsive" forces for other particles, thus biasing those to explore different regions of the search space. Results on detecting multiple supporting surfaces as input for the plane pop-out attentional operator (see Figure 2) are reported in [34] (Annex 2.4).

Both above approaches segment given data (dynamic 2D images or a static depth image) in terms of a given model type (e.g. planes) by having multiple hypotheses compete in order to find a globally most consistent interpretation. These competing hypotheses represent the uncertainty about data association, i.e. which pieces belong to the same model.

Regarding active visual search we conducted a literature review [1] (Annex 2.5). A lot of work in computer vision that mentions object search essentially tries to locate an object in the image, i.e. the object is already

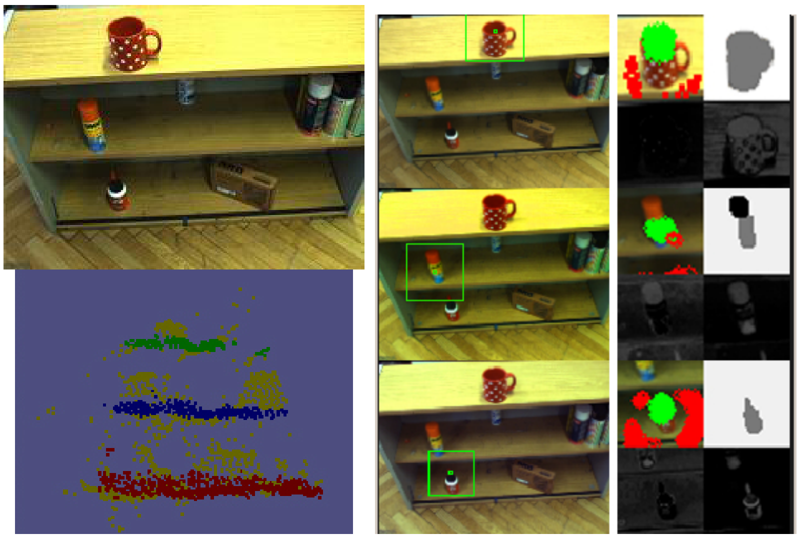


Figure 2: Detection of multiple supporting planes (coloured in red, green and blue) in stereo data (left) and some popping-out object hypotheses with their segmentations (right)

assumed to be present in the image. Object search in this context is different from the case where a mobile robot is being tasked with finding an object in an office or home environment. In this setting assuming that the set of objects that the robot can recognise are already in the field of view is clearly unrealistic. The goal of object search, as defined in this report, then is to calculate the set of sensing actions (typically camera movements) which brings the target object in the sensor field of view in an as efficient manner as possible. We reviewed the active visual search literature: from the classic art gallery problem, which is a purely geometric approach to purely probabilistic approaches such as pursuit/evasion problems as well as hybrid approaches. Deliverable DR.3.1 (Annex 2.4.2) reports work on the implementation of an active search strategy which utilises knowledge of qualitative spatial relations between objects, namely the relation "on" (i.e. physical support) to guide the selection of camera views.

1.1.3 Relation to the state-of-the-art

Various systems to incrementally and interactively build up object models have been proposed; based on edges [3] or interest points [13]; with various levels of required prior object knowledge from complete CAD models [32], rough bounding shapes [23], planar objects [24, 14] to mere image regions of interest [29, 28] or no limitation at all other than object texture [26]. Our BLORT toolbox uses a selection of methods designed to allow quick and user-friendly interactive acquisition of objects with a possibility of fully automatic operation, handling textured and untextured objects. This comes at the cost of limitations in terms of shape complexity during the initial detection step, which are however not too severe for a wide array of robotic manipulation tasks,

Given a fixed threshold to detect inliers incremental methods favour planes detected first over subsequent planes by greedily consuming features [33, 17, 22]. Recently developed approaches [31, 5, 10] overcome this drawback by treating hypotheses equally, but plane hypotheses have to be created independently of each other and thus it is not possible to restrict the search space, which leads to higher computational complexity. We propose model selection based on the MDL principle: Instead of creating all hypotheses at once, pruning models and then using model selection, we propose to embed model selection in an incremental scheme and thus guide randomised selection of interest points to compute more likely plane hypotheses. This allows us to avoid an additional hypotheses pruning step without a decrease in performance. Finally, this formulation allows us to explicitly introduce priors, hence we can detect and track planes in one scheme which is not possible in any of the approaches described above.

Another approach for robust multi-model estimation is based on Particle Swarm Optimization (PSO), which is a metaheuristic optimization method first introduced for simulating social behaviour of birds [18, 8]. PSO maintains a swarm of candidate solutions (particles) in parameter space, and moves these particles in the search-space according to their own personal best solution found so far and the swarm's global best solution so far. We extend this approach by replacing the globally best solution with a list of locally best solutions thus allowing the estimation of multiple models.

1.2 Modular motor learning

1.2.1 Planned work

Work on modular motor learning concerns Task 2.5:

> **Task 2.5:** *Modular motor learning theory. Using 3D contour based shape descriptors plus haptic and proprioceptive information, extend the modular motor learning theory to predict and control object trajectories and contact relations. Extend object models with attributes that can be learned/detected only in contact with the object, e.g. weight. This will be investigated for both pushing and grasping actions. (M7 - M21)*

Planned work included learning algorithms for prediction and control during pushing manipulations of rigid 3D objects, and self understanding and self extension with respect to knowledge about how objects will respond to pushes.

Significant progress has been made with advancing several new theories for learning of object behaviours under manipulative actions. This includes implementing, testing and comparison of various different kinds of modular combinations of predictive experts. These systems learn to predict the motions of objects which will result from pushes applied by a robot finger.

We have also implemented algorithms for context identification (identifying which of several different kinds of learned workpiece behaviours is currently being experienced). This system leads to a theory for self understanding, i.e. detection of knowledge gaps as new contexts which do not correspond to any previously experienced contexts. This approach to self understanding is currently being implemented, and it also ties in closely with the same theory and mathematics that has been applied for solving very different problems of self understanding in other CogX robot prototype scenarios, e.g. the George scenario. We have not yet implemented a solution for self extension (filling knowledge gaps once they have been detected), however the theory for doing this has been formulated and is currently being implemented. The theory and implementation work on context discrimination is described in more detail in DR 2.3.

Work has also been done in some additional areas, such as fusion of learned motion predictors into visual tracking algorithms to achieve enhanced tracking performance. This is enabling ongoing experiments in which online learning is used to mutually refine both visual tracking and workpiece motion models.

1.2.2 Actual work performed

Push prediction We have been investigating push prediction scenarios, in which a robot arm equipped with a single rigid finger applies pushes to various rigid objects, and the resulting work piece motions are visually tracked. We are interested in pushing because it is a particularly simple and fundamental kind of manipulation. Pushing with a single finger is also fundamentally a part of more complex grasping with multiple fingers. Furthermore, we regard pushing as merely one example domain, and the learning techniques might be applied more generally to enable machines to make predictions about the motions of interacting bodies under other circumstances.

To predict the results of any particular push, the planned finger trajectory is broken down into 150 small "sub-pushes" of length a few mm each. A prediction is made about the pose that the work-piece will have moved to by the end of the first such sub-push. This predicted pose is then used as the starting work-piece position at the beginning of the next sub-push, and so on, so that a complete prediction is made over the entire push (all 150 sub-pushes). This entire prediction is made before the robot begins executing the push, so that the resulting predictions are achieved without any recursive estimation (e.g. predictor-corrector type visual feedback of poses during and throughout the push).

We have investigated a variety of different machine learning approaches, in which the robot learns to predict the work-piece motions that will result from various applied pushes. Extensive technical descriptions of these methods, and the results achieved with them, can be found in [20] and [19] (Annexes 2.6 and 2.7 respectively).

1) Global experts - We have explored the use of kernel density estimation to learn a "global" model for how an entire object will move when subjected to a push. A data-base of training examples is developed, which consists of many "sub-pushes" for which the work-piece pose before and after and the robot finger pose before and after the sub-push have been recorded. In this case the work-piece is described simply by the pose of a single rigid-body co-ordinate frame which is attached to the work-piece.

Now, a kernel density estimation scheme is used to train a probability density function which yields probabilities over the possible end poses that the work-piece may end up occupying, given the starting pose and the finger motion.

This method works well if it is applied to a single object, and if a good set of training data is available for that object. However, the method has difficulty in generalising by making predictions about new objects, with novel shapes that are different to those trained on.

2) Local (finger tip) expert - To improve generalisation performance, we have explored the addition of another "expert" which makes predictions about the motion of a small surface patch of the object, relative to the part of the finger-tip which contacts it. This expert is also trained as a kernel density estimator, but is trained on the before and after poses, not of a frame fixed in the overall work-piece, but of a frame fixed in the small patch of the work-piece surface at the point where the finger pushes it. Fundamentally, the reason that this assists generalisation is that, even though the gross motion behaviours of bodies will vary greatly from one body to another, the behaviour of a small surface patch will be very similar between different objects, so that information learned on one object can be usefully transferred to making predictions about another object.

Because the global predictor and local predictor are both described as probability distributions, the opinions of these two "modular experts" can easily be combined, simply as a product of densities. This is more effective than a sum of experts approach since that could produce an "average" prediction which is inconsistent with either of the original expert predictions. In contrast, product of densities allows one prediction to constrain the other, resulting in a combined prediction of a region of work-piece pose, which is of high probability according to both of the original (global and local) expert predictions.

The global expert tends to encode information about gross body motions, while the local expert helps constrain the global predictions, by ruling out regions of pose which are incompatible with the contacting finger motions.

3) Multiple surface patch contact experts - Further improvements in accuracy and shape generalisation can be attained by incorporating more experts, which are learned distributions of the relative motions of small parts of surfaces of the object and its surroundings. In this approach, additional KDE distributions are trained on data about the relative motions of small surface patches of the work-piece and corresponding small surface patches on supporting parts of the environment, e.g. the table top.

Again, these distributions can be combined with the predictions of the global and local (finger-tip) experts, to further constrain the predicted motion and further enhance accuracy, by combining as a product of densities. The surface contact experts provide constraints on the predicted motion, by ruling out possible motions that are incompatible with the supporting surfaces - e.g. ruling out motions which would result in the work-piece penetrating the table top. These additional experts can improve the generalisation capabilities of the system, because the relative motions of small patches of object and small patches of supporting surface may be very similar amongst many different objects, so that information learned by experiencing one object might usefully be applied to making predictions about a different object.

4) Regression learning - We have also developed an alternative method of learning to predict work piece motions, based on the LWPR regression learning technique. In this approach, an LWPR system is trained to map inputs (finger motion and work-piece starting pose) onto outputs (final work-piece pose). This approach has some advantages in terms of speed, and makes accurate predictions about known objects. However, it has difficulty generalising in certain cases where the system is presented with objects with previously un-encountered shapes, and it has difficulty making predictions about pushes in previously un-encountered directions. This approach can be thought of as encoding similar kinds of information as the “global” KDE expert.

5) Minimum energy principle - We have also developed an alternative method for making “global” type predictions, without having to do learning on training data. This method makes use of the physics principle of minimum energy, and predicts in favour of those work piece motions which end up in minimum energy states and require the least work to achieve. For every candidate work-piece pose change, a series of cost function terms that are proportional to the various kinds of work necessary for that pose change are added up (e.g. increase in gravitational potential energy, work done against friction, etc.) - these energy costs can then be converted into a probability distribution, by using a Boltzmann distribution to convert energies into probabilities. Once this is in the form of a distribution, it can be combined with the predictions of the KDE experts by the same system of products of experts, so that the minimum energy predictor offers an alternative to a learned global expert, in situations where the system is confronted by a new object for which it has no previous training data to learn from.

6) Deformable objects - Additionally, we have begun to investigate ways of learning to predict the deformations of non-rigid bodies when subjected to one or two finger pushing operations, although this work is still in preliminary stages. So far a model has been formulated for representing a deforming body as a mesh of masses and springs, and ongoing work is looking at how to train the parameters of this model from data extracted from video sequences in which bodies deform under pushes from a robot finger.

Prediction for vision Throughout this work, a particle filter has been used for visual tracking of the work piece during pushing manipulations (BLORT, mentioned above and in Annex 2.2), where each particle represents a candidate work piece pose. At each iteration of the tracking algorithm, all particles are propagated according to a dynamics model. Typically the dynamics is unknown and so the "dynamics model" merely consists of randomly perturbing the particles by a small amount of Gaussian noise at each time step. Clearly, improved performance can be expected if a more accurate model of work piece motions could be substituted for this random perturbation.

Hence, we have been investigating the use of a dynamics prediction models, to propagate each of the particles during pushing operations, and this yields improved tracking. We have experimented both with using learned predictors and also physics simulation software for propagating particles from one time step to the next. In both cases, it has become clear that successful tracking depends on explicitly representing uncertainty, and in maintaining multiple hypotheses about workpiece trajectories and their likelihoods.

In the case of using learned motion predictors, a "chicken and egg" situation now arises, where vision is used to capture the motions of work pieces in order to train a motion predictor, however accurate vision is not itself possible without the use of the learned motion prediction for particle propagation as part of the predictor-corrector recursive tracking algorithm. Hence, we have recently begun investigating a "boot-strapping" approach, in which both vision and motion prediction are mutually refined via online learning over a sequence of many pushes.

We have simultaneously been investigating the use of physics simulation software, to improve the performance of visual motion estimation algorithms in extracting physically plausible motions of moving objects from video sequences [7] (Annex 2.8). In this approach, a candidate trajectory is assigned a cost according to a cost function which includes a component that represents deviation from visual observations and a component that represents deviation from physically plausible trajectories as predicted by the physics engine. Hence, by iteratively optimising to minimise this cost function, knowledge of object dynamics can be factored into the extraction of a trajectory from an image sequence. The cost function approach enables some leeway in deviating from physics predictions, which is necessary to handle the large amount of uncertainty that arises with respect to initial conditions, object characteristics and physical parameters.

1.2.3 Relation to the state-of-the-art

Previous work on pushing manipulation is predominantly restricted to simple 2D objects sliding over a planar surface. In contrast we tackle real 3D objects under pushes which can make them tip or topple as well as slide. More importantly, previous work relies either on analytical physics solutions to predicting workpiece motion, or on the use of physics simulation software. In contrast, our work does not depend on pre-programmed physics models, but involves enabling the robot to learn dynamics behaviours by simply observing a relationship between actions and outcomes. This is useful in practical terms, since it is actually very difficult to make physics simulation software deliver accurate predictions about the interactions of real objects. Furthermore, we show how learned systems can generalise, to make useful predictions about new objects which are different from those trained on. In contrast, physics approaches are rarely able to make predictions outside the one context for which they are tuned, and tuning a physics engine to accurately replicate real object motions for even a single context can be very difficult.

The use of Kernel Density Estimators, to predict distributions over the possible motions of the work-piece, enables the modular approach, by which the opinions of multiple experts can be combined as a product of densities, and this is a novel approach to predicting the outcomes of push manipulations.

1.3 Grasping learned objects

1.3.1 Planned work

Work on grasping concerns Task 2.2:

> **Task 2.2:** *Early grasping strategies. Based on the visual sensory input extracted in Task 2.1, define motor representations of grasping actions for two- and three-fingered hands. The initial grasping strategies will be defined by a suitable approach vector (relative pose with respect to object/grasping part) and preshape strategy (grasp type). (M7 - M21)*

1.3.2 Actual work performed

A major problem for grasping of unknown objects is that grasp synthesis needs to be performed from data a robot can extract on-line. This is a difficult problem due to the high dimensionality of the problem, incomplete and uncertain information about the environment and the objects to be grasped, and lack of generalisable measures of quality for grasp planning. We have studied two important problems related to this: i) the methodology and representations for grasp selection on unknown objects, and learning from experience for grasping of objects similar in shape (familiar objects). Many of the state-of-the-art methods generate grasps on unknown objects by using different methods for segmentation of raw sensory data such as point clouds generated from stereo vision or laser data. However, the decision if an effort in terms of a segmentation or part-based decomposition is in fact worthy is often not treated in the literature. Rarely, a set of differently granulated decision criteria and representations is evaluated to see if some partitioning of the object in fact is reasonable or not. In [30] (Annex 2.9), we develop a system for robot grasp selection that copes with known and unknown objects where the focus is not to find the most stable grasp, but a grasp that is force-closure and feasible, collision-free and constrained by the parameters of the robotic hand, given available sensor data. The core part of the paper is the study of different representations necessary for implementing grasping tasks on objects of different complexity. We show how to select a grasp satisfying force-closure, taking into account the parameters of the robot hand and collision-free paths. Our implementation takes also into account efficient computation at different levels of the system regarding representation, description and grasp hypotheses generation.

1.3.3 Relation to the state-of-the-art

A robot grasping cycle involves data representation, shape description and grasp hypotheses generation. Most of the recent work on robotic grasping relies on 3D data although there are approaches producing grasp hypotheses using 2D image features, e.g. [2]. A range of state-of-the-art methods synthesise 3D object shapes from point clouds by using superquadrics [11] or other shape primitives such as boxes [15]. Assuming that an arbitrary point cloud has to be approximated, a single primitive is obviously not enough for many objects. The more complex the shape is, the more primitives have to be used to represent its individual parts. Multiple methods approach this problem by a variety of segmentation methods, [11, 15, 4].

Overall, there has been a lot of work on grasp planning on different levels: path planning [6], planning on 3D mesh models [9] and databases [12], planning on shape primitives [16]. A general problem is that for any kind of single-view 3D sensor system, a generated point cloud of an object or a scene is not complete. Grasp selection may also benefit from assessing the shape complexity. If complexity is high, it is worthy to segment the object into graspable parts. The segmentation of a model into its parts is also necessary for task-constrained grasping of simple objects.

2 Annexes

2.1 Richtsfeld et al. "Taking in Shape: Detection and Tracking of Basic 3D Shapes in a Robotics Context"

Bibliography Richtsfeld, Andreas; Mörwald, Thomas; Zillich, Michael; Vincze, Markus: "Taking in Shape: Detection and Tracking of Basic 3D Shapes in a Robotics Context", Computer Vision Winter Workshop (CVWW), 2010, p. 91-98

Abstract Segmenting and tracking generic objects in an unknown dynamic scene remains an elusive goal for computer vision. In this paper we tackle a simplified problem, namely detecting and tracking objects from a class of basic shapes (cuboids, cylinders, cones, spheres) in scenes containing a ground plane. We use perceptual grouping of edges to identify generic views of basic shapes, instantiate 3D wire-frame models assuming that objects initially rest on the ground and subsequently track these using a particle-filter based tracker. During tracking we augment the wire-frame models with surface texture which substantially increases the robustness of tracking with respect to background clutter as well as occlusions and lighting effects.

Relation to WP This work combines two methods (detection of shape primitives and edge based tracking) to successively build detailed object models. More information in the form of surface texture is added online as the object is tracked and increases robustness of tracking, thus reducing pose uncertainty. While the system in this case does not explicitly reason about reducing pose uncertainty by adding texture (self understanding), this approach presents an example of online and continuous self extension.

2.2 Mörwald et al. "BLORT - The Blocks World Robotic Vision Toolbox"

Bibliography Mörwald, T.; Prankl, J.; Richtsfeld, A.; Zillich, M.; Vincze, M.: "BLORT - The Blocks World Robotic Vision Toolbox", Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation (in conjunction with ICRA 2010)

Abstract The vision and robotics communities have developed a large number of increasingly successful methods for tracking, recognising and on-line learning of objects, all of which have their particular strengths and weaknesses. A researcher aiming to provide a robot with the ability to handle objects will typically have to pick amongst these and engineer a system that works for her particular setting. The work presented in this paper aims to provide a toolbox to simplify this task and to allow handling of diverse scenarios, though of course we have our own particular limitations: The toolbox is aimed at robotics research and as such we have in mind objects typically of interest for robotic manipulation scenarios, e.g. mugs, boxes and packaging of various sorts. We are not aiming to cover articulated objects (such as walking humans), highly irregular objects (such as potted plants) or deformable objects (such as cables). The system does not require specialised hardware and simply uses a single camera allowing usage on about any robot. The toolbox integrates state-of-the art methods for detection and learning of novel objects, and recognition and tracking of learned models. Integration is currently done via our own modular robotics framework, but of course the libraries making up the modules can also be separately integrated into own projects.

Relation to WP Object models are augmented further beyond surface texture with information that allows recognition in cluttered environments. As before these models are learned online while objects are tracked, allowing to continuously extend object knowledge as new views appear. Maintaining models in such a way as to allow online extension as opposed to an offline learning phase followed by an online recognition/tracking phase is a crucial requirement of a system that wants to self-understand and self-extend.

2.3 Prankl et al. "Incremental Model Selection for Detection and Tracking of Planar Surfaces"

Bibliography Prankl, J.; Zillich, M.; Leibe, B; Vincze, M.: "Incremental Model Selection for Detection and Tracking of Planar Surfaces", BMVC 2010, To appear.

Abstract Man-made environments are abundant with planar surfaces which have attractive properties and are a prerequisite for a variety of vision tasks. This paper presents an incremental model selection method to detect piecewise planar surfaces, where planes once detected are tracked and serve as priors in subsequent images. The novelty of this approach is to formalize model selection for plane detection with Minimal Description Length (MDL) in an incremental manner. In each iteration tracked planes and new planes computed from randomly sampled interest points are evaluated, the hypotheses which best explain the scene are retained, and their supporting points are marked so that in the next iteration random sampling is guided to unexplained points. Hence, the remaining finer scene details can be represented. We show in a quantitative evaluation that this new method competes with state of the art algorithms while it is more flexible to incorporate prior knowledge from tracking.

Relation to WP This work complements the above work which is based primarily on shape primitives. Here we detect and track planar surface patches in dynamic scenes. The MDL based model selection scheme allows us to successively segment the data as being explained in terms of multiple models thus successively explaining the scene in terms of planar patches (where freeform surfaces are approximated by small patches). This work broadens the range of objects and scenes we can handle. Moreover model selection fits well into the idea of reasoning about uncertainty (self understanding), as it explicitly evaluates competing alternative hypotheses against each other to find a globally most consistent interpretation.

2.4 Zhou et al. "Multi-model Fitting Using Particle Swarm Optimization for 3D Perception in Robot Vision"

Bibliography Kai Zhou, Michael Zillich, Markus Vincze, Alen Vrečko and Danijel Skočaj: Multi-model Fitting Using Particle Swarm Optimization for 3D Perception in Robot Vision, submitted to IEEE International Conference on Robotics and Biomimetics (ROBIO 2010)

Abstract Attention operators based on 2D image cues (such as color, texture) are well known and discussed extensively in the vision literature but are not ideally suited for robotic applications. In such contexts it is the 3D structure of scene elements that makes them interesting or not. We show how a bottom-up exploration mechanism that selects spaces of interest (SOIs) based on scene elements that pop out from planes is used within a larger architecture for a cognitive system. This mechanism simplifies the object localization as single plane detection, which is however not practical when dealing with real scenes that contains objects with complicated structures (e.g. objects in a multi-layer shelf). Therefore, the key work required for this situation is the multi-plane estimation, which is solved in this paper using Particle Swarm Optimization (PSO).

Relation to WP This work presents another approach to the important and difficult problem of multiple model estimation, where here we use particle swarm optimization. Again the models are planes, this time however extracted from noisy stereo data. The goal is the same, namely to successively explain the data as being composed of multiple instances of a model plus remaining noise. This work feeds into 3D attention (plane pop-out) and thus into the processing chain of visual search.

2.5 Aydemir “Object Search with Mobile Robots”

Bibliography A. Aydemir: “Object Search with Mobile Robots”, Technical Report, KTH, 2010

Abstract This report presents methods that enables a mobile robot efficiently search an environment for an object.

Object search entails ascertaining the location of a specific object, or of one or more of a given class of objects, and doing so in an efficient manner. Object search is an important component of a mobile robots skill repertoire. The need for utilizing objects in the robotics framework becomes more and more crucial if robots will one day coexists with humans. Previous work (Vasudevan et al. 2007), has shown that the human spatial understanding is based heavily on the objects that are in the environment. The same objects can also be used for various fields such as navigation and localization. Objects not only help defining the space around them but also accommodate certain functional interpretations. This is why different modes of interaction such as grasping and manipulation attracts much attention in the research community.

Relation to WP This report is central to the work package. It presents a thorough literature research that forms the basis for subsequent work in active visual search in this work package and also work in WP3. Active visual search is an example of a knowledge gathering action based on incomplete knowledge. The geometric reasoning underlying many approaches represents a case of self understanding: the agent reasons about its own viewpoint in relation to obstacles and free space, and then plans actions to maximise the chances of gathering more information. Moreover active visual search is simply a prerequisite for any task that includes object search in a realistic real world setting, as all object recognisers have limitations in terms of scale change and occlusion.

2.6 Kopicki et al. "Learning to predict how rigid objects behave under simple manipulation"

Bibliography Marek Kopicki, Rustam Stolkin, Sebastian Zurek, Jeremy Wyatt: "Learning to predict how rigid objects behave under simple manipulation", Technical Report, University of Birmingham, 2010

Abstract An important problem in robotic manipulation is the ability to predict how objects behave under manipulative actions. This ability is necessary to allow planning of object manipulations. Physics simulators can be used to do this, but they model many kinds of object interactions poorly, and unless there is a precise description of an objects properties its behaviour will remain quite uncertain in many cases. An alternative is to learn a motion model for objects by interacting with them. In this report we specifically address the problem of learning to predict the interactions of rigid bodies in a probabilistic framework, and demonstrate the results in the domain of robotic push manipulation. During training a simulated robot finger applies pushes to objects, and learns to predict their resulting motions. The learning does not make explicit use of physics knowledge, nor is it restricted to domains with any particular physical properties. The prediction problem is posed in terms of estimating probability densities over the possible rigid body transformations of an object under a known action. We show how to make this density estimation problem tractable by factorization, and describe the differences with an approach based on regression. This factorization creates many simpler estimation problems, and allows us to recombine the resulting predictors to model the constraints imposed by surface contacts. We show how even a very simple product of experts can generalize from learned examples: successfully predicting object motions for previously unseen object poses, push directions and objects with novel shape. We show results with a variation in the number of experts. Performance is evaluated through a combination of virtual experiments in a physics simulator, and real experiments with a 5-axis arm equipped with a simple, rigid finger.

Relation to WP This work is relatively central to the workpackage. Vision is used for tracking manipulated objects, to inform learning. Learning is used to enable prediction, i.e. predicting the motions of manipulated objects, given applied robotic pushes. Prediction is necessary for control - in order to apply the correct manipulations to manoeuver the workpiece in a desired direction, it is necessary to be able to predict the effects that different possible manipulations will cause. Furthermore, recent and ongoing work is exploring ways by which these learned predictions can themselves be used to enhance vision. While this paper does not explicitly deal with the issues of knowledge gaps and self extension, the probabilistic representations of uncertainty that are described here, are central to the theories for self understanding and self extension that have more recently been formulated and are currently being implemented.

2.7 Kopicki et al. "Predicting workpiece motions under pushing manipulations using the principle of minimum energy"

Bibliography Marek Kopicki, Rustam Stolkin, Sebastian Zurek, Thomas Mörwald, Jeremy Wyatt: "Predicting workpiece motions under pushing manipulations using the principle of minimum energy", Proceedings of the RSS workshop on Representations for Object Grasping and Manipulation in Single and Dual Arm Tasks, 2010

Abstract We are investigating the problem of predicting how objects behave under manipulative actions. In particular, we wish to predict the workpiece motions which will result from simple pushing manipulations by a single robotic fingertip. Such interactions are themselves fundamental components of multi-fingered grasping and other complex interactions. Physics simulators can be used to do this, but they model many kinds of object interactions poorly, being dependent on detailed scene descriptions and parameters, which in practice are often difficult to tune. Additionally, we have previously investigated ways of learning to predict, by employing density estimation techniques to learn, from many example pushes, a probabilistic mapping between applied pushing motions and resulting workpiece motions. In contrast, this paper presents an alternative approach to prediction, which does not rely on learning but infers the likelihood of possible workpiece motions by using the simple physics principle of minimum energy. This approach is advantageous in situations where insufficient prior knowledge is available for training our learned predictors. In such situations, possible strategies include either training learned predictors on unrealistic simulation data, or making use of the simple physics approach which requires no training. We show that the second of these strategies performs significantly better, and approaches the performance of learned predictors are trained on observations of real object motions.

Relation to WP This paper contributes to the prediction and control components of the work package. It concerns making predictions about the motions of manipulated bodies. The method outlined here provides an alternative to learned predictors (that are the main thrust of the work) in certain situations where insufficient training data may be available for accurate dynamics learning. The method is designed to represent uncertainty probabilistically, in such a way that the predictions of this method can be conveniently combined with the predictions made by our learned experts.

2.8 Duff et al. "Motion estimation using physical simulation"

Bibliography Damien Jade Duff, Jeremy Wyatt, and Rustam Stolkin: "Motion estimation using physical simulation", Proceedings of the International Conference on Robotics and Automation (ICRA), 2010

Abstract We consider the task of monocular visual motion estimation from video image sequences. We hypothesise that performance on the task can be improved by incorporating an understanding of physically likely and feasible object dynamics. We test this hypothesis by incorporating a physical simulator into a least-squares estimation procedure. We initialise a full trajectory estimate using RANSAC followed by gradient descent refinement. We present results for 2D image sequences consisting of single ambiguous, visible or occluded balls, as well as results for 3D computer-generated sequences of objects in free flight with added long-tailed noise and outliers. Results suggest that restricting the estimation to allow only motions that are feasible according to the physics simulator can produce marked improvement when the observed object motion is within the limits of the physics simulator and its world model. Conversely, merely penalising deviations from feasible physical dynamics produces a consistent but incremental improvement over more common dynamics models.

Relation to WP This paper contributes to the active vision part of the work package, by providing an alternative method for extracting the trajectory of interacting bodies from video sequences. Further, it contributes to the learning and prediction parts of the work package, in that our learned predictors are trained by learning the relationship between applied robotic pushes and the resulting motions of pushed objects, which must be extracted from video sequences by a vision system. This work is also interesting in that it investigates the difficulties of physics engines, which may be solved by our alternative learned dynamics models, and thus provides a useful benchmark for comparison. It also shows how these difficulties require motion models which take into account multiple hypotheses and probabilistic representations of uncertainty, and so contributes to our understanding of the issues of gaps in knowledge of objects.

2.9 Rubio et al. "Representations for Object Grasping and Learning from Experience"

Bibliography Rubio, Oscar J.; Huebner, Kai; Kragic, Danica: "Representations for Object Grasping and Learning from Experience", IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010). To appear.

Abstract We study two important problems in the area of robot grasping: i) the methodology and representations for grasp selection on known and unknown objects, and ii) learning from experience for grasping of similar objects. The core part of the paper is the study of different representations necessary for implementing grasping tasks on objects of different complexity. We show how to select a grasp satisfying force-closure, taking into account the parameters of the robot hand and collision-free paths. Our implementation takes also into account efficient computation at different levels of the system regarding representation, description and grasp hypotheses generation.

Relation to WP This work deals with grasping of known and novel objects. It maintains a database of already encountered objects together with triplets of contact points for a stable grasp. If an object is encountered again or an object with a similar shape, it will be grasped by adapting the already learned grasp triplet. If that proves unsuccessful, a new specific grasp will be generated for that object and added to the database. The system thus continuously evaluates itself and extends its grasping knowledge.

References

[1] Alper Aydemir. Object search with mobile robots. Technical report, KTH, 2010.

[2] J. Bohg and D. Kragic. Learning Grasping Points with Shape Context. *Robotics and Autonomous Systems*, 2010.

[3] M. Brown, T. Drummond, and R. Cipolla. 3D Model Acquisition by Tracking 2D Wireframes. In *Proc. British Machine Vision Conference (BMVC)*, 2000.

[4] L. Chevalier, F. Jaillet, and A. Baskurt. Segmentation and Superquadric Modeling of 3D Objects. *Journal of Winter School of Computer Graphics, WSCG'03*, 2003.

[5] T.-J. Chin, H. Wang, and D. Suter. Robust fitting of multiple structures: The statistical learning approach. In *IEEE International Conference on Computer Vision (ICCV)*, 2009.

[6] Rosen Diankov, Siddhartha Srinivasa, David Ferguson, and James Kuffner. Manipulation Planning with Caging Grasps. In *IEEE International Conference on Humanoid Robots*, pages 285–292, 2008.

[7] Damien Duff, Jeremy Wyatt, and Rustam Stolkin. Motion estimation using physical simulation. In *Proceedings of the International Conference on Robotics and Automation*, 2010.

[8] R. Eberhart and Y. Shi. Comparing inertia weights and constriction factors in particle swarm optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 1, pages 84–88, 2000.

[9] Sahar El-Khoury and Anis Sahbani. On Computing Robust N-Finger Force-Closure Grasps of 3D Objects. In *IEEE International Conference on Robotics and Automation*, pages 2480–2486, 2009.

[10] D. Fouhey, D. Scharstein, and A. Briggs. Multiple plane detection in image pairs using j-linkage. In *20th International Conference on Pattern Recognition (ICPR 2010), To appear*, 2010.

[11] C. Goldfeder, P. K. Allen, C. Lackner, and R. Pelossof. Grasp Planning Via Decomposition Trees. In *IEEE International Conference on Robotics and Automation*, pages 4679–4684, 2007.

[12] C. Goldfeder, M. Ciocarlie, H. Dang, and P. K. Allen. The Columbia Grasp Database. In *IEEE International Conference on Robotics and Automation*, pages 3343–3349, 2009.

[13] Iryna Gordon and David G Lowe. What and where: 3D object recognition with accurate pose. In J. Ponce, M. Hebert, Schmid. C., and A. Zisserman, editors, *Toward Category-Level Object Recognition*, chapter What and where: 3D object recognition with accurate pose, pages 67–82. Springer, 2006.

[14] Michael Grabner, Helmut Grabner, and Horst Bischof. Learning Features for Tracking / Tracking via Discriminative Online Learning of Local Features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'07)*, 2007.

[15] K. Huebner, S. Ruthotto, and D. Kragic. Minimum Volume Bounding Box Decomposition for Shape Approximation in Robot Grasping. In *IEEE Int'l Conf. on Robotics and Automation*, pages 1628–1633, 2008.

[16] Kai Huebner, Kai Welke, Markus Przybylski, Nikolaus Vahrenkamp, Tamim Asfour, Danica Kragic, and Rüdiger Dillmann. Grasping Known Objects with Humanoid Robots: A Box-based Approach. In *International Conference on Advanced Robotics*, 2009.

[17] Y. Kanazawa and H. Kawakami. Detection of planar regions with uncalibrated stereo using distributions of feature points. In *British Machine Vision Conference (BMVC)*, 2004.

[18] J. Kennedy and R. Eberhart. Particle Swarm Optimization. In *Proceedings of IEEE International Conference on Neural Networks*, pages 1942–1948, 1995.

[19] Marek Kopicki, Rustam Stolkin, Sebastian Zurek, Thomas Morwald, and Jeremy Wyatt. Predicting workpiece motions under pushing manipulations using the principle of minimum energy. In *Proceedings of the RSS workshop on Representations for Object Grasping and Manipulation in Single and Dual Arm Tasks*, 2010.

[20] Marek Kopicki, Rustam Stolkin, Sebastian Zurek, and Jeremy Wyatt. Learning to predict how rigid objects behave under simple manipulation. Technical report, University of Birmingham, 2010.

[21] T. Mörwald, J. Prankl, A. Richtsfeld, M. Zillich, and M. Vincze. BLORT - The Blocks World Robotic Vision Toolbox. In *Best Practice in 3D Perception and Modeling for Mobile Manipulation (in conjunction with ICRA 2010)*, 2010.

[22] G. Lopez Nicolas, J.J. Guerrero, O.A. Pellejero, and C. Sagues. Computing homographies from three lines or points in an image pair. In *CIAP 2005*, pages 446–453, 2005.

[23] M. Özuysal, V. Lepetit, F. Fleuret, and P. Fua. Feature Harvesting for Tracking-by-Detection. In *European Conference on Computer Vision*, volume 3953, pages 592–605, 2006.

[24] Mustafa Özuysal, Pascal Fua, and Vincent Lepetit. Fast Keypoint Recognition in Ten Lines of Code. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[25] J. Prankl, M. Zillich, B Leibe, and M. Vincze. Incremental Model Selection for Detection and Tracking of Planar Surfaces. In *BMVC*, 2010.

[26] Pan Qi, Gerhard Reitmayr, and Tom Drummond. ProFORMA: Probabilistic Feature-based On-line Rapid Model Acquisition. In *Proc. British Machine Vision Conference (BMVC)*, 2009.

[27] Andreas Richtsfeld, Thomas Mörwald, Michael Zillich, and Markus Vincze. Taking in Shape: Detection and Tracking of Basic 3D Shapes in a Robotics Context. In *Computer Vision Winder Workshop (CVWW)*, pages 91–98, 2010.

[28] Hayko Riemenschneider, Michael Donoser, and Horst Bischof. Robust Online Object Learning and Recognition by MSER Tracking. In *Proc. 13th Computer Vision Winter Workshop (CVWW)*, 2007.

[29] Peter M. Roth, Michael Donoser, and Horst Bischof. On-line Learning of Unknown Hand Held Objects via Tracking. In *Proc. 2nd International Cognitive Vision Workshop (ICVS)*, Graz, Austria, 2006.

[30] Oscar J. Rubio, Kai Huebner, and Danica Kragic. Representations for Object Grasping and Learning from Experience. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010. To appear.

[31] Roberto Toldo and Andrea Fusiello. Robust multiple structures estimation with j-linkage. In *ECCV 08: Proceedings of the 10th European Conference on Computer Vision*, pages 537–547, 2008.

[32] L. Vacchetti, V. Lepetit, and P. Fua. Stable Real-Time 3D Tracking using Online and Offline Information. *PAMI*, 2004.

[33] E. Vincent and R. Laganiere. Detecting planar homographies in an image pair. In *Proceedings of the 2nd International Symposium on Image and Signal Processing and Analysis (ISPA)*, pages 182–187, 2001.

[34] Kai Zhou, Michael Zillich, Markus Vincze, Alen Vrecko, and Danijel Skocaj. Multi-model fitting using particle swarm optimization for 3d perception in robot vision. In *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2010. submitted.

Computer Vision Winter Workshop 2010, Libor Špaček, Vojtěch Franc (eds.)
Nové Hrady, Czech Republic, February 3–5
Czech Pattern Recognition Society

Taking in Shape: Detection and Tracking of Basic 3D Shapes in a Robotics Context

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

Automation and Control Institute (ACIN), Vienna University of Technology
{arichtsfeld, moerwald, zillich, vincze} @acin.tuwien.ac.at

Abstract Segmenting and tracking generic objects in an unknown dynamic scene remains an elusive goal for computer vision. In this paper we tackle a simplified problem, namely detecting and tracking objects from a class of basic shapes (cuboids, cylinders, cones, spheres) in scenes containing a ground plane. We use perceptual grouping of edges to identify generic views of basic shapes, instantiate 3D wire-frame models assuming that objects initially rest on the ground and subsequently track these using a particle-filter based tracker. During tracking we augment the wire-frame models with surface texture which substantially increases the robustness of tracking with respect to background clutter as well as occlusions and lighting effects.

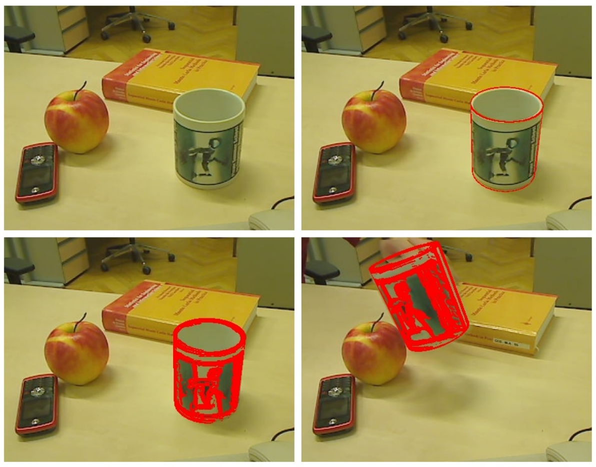


Figure 1: Detecting and tracking a mug on a typical table-top scene: The red contour of the top right image indicates a mug detected as a cylinder, which is then instantiated as a 3D wire-frame model by assuming it rests on the ground plane. Knowing the shape and position of the object model, the surface texture is captured from the camera image (bottom left image) and attached to the wire-frame model. This allows us to robustly track the object even at positions in degenerate views, partially occluded or across heavy background clutter (bottom right image).

1 Introduction

Perceiving 3D shape plays a major role in computer vision applications for robotics tasks. Most obviously this is true for tasks involving manipulation of objects, be it grasping, simple pushing or just observing a human handling an object. While tracking based on 3D models of objects has a long history and many increasingly sophisticated methods exist, robustness and speed remain a challenge to this date. Also the question of how sufficiently accurate object models are acquired in the first place is often left open.

In this work we present a system that aims to close this gap between model acquisition from an unknown scene and robust tracking of models. We make several simplifying assumptions. Having in mind robotics applications we assume a robot mounted camera which allows us to at least assume the ground plane (or table plane) while the rest of the scene remains of course unknown. Furthermore we restrict our search to objects belonging to simple shape classes (cuboids, cylinders, cones and spheres) which are detected in generic views from edge images. Note that there are in principle no limits to shape complexity for the tracker once we have a model (in fact more “non-regular” shapes simplify the task for tracking) but that the difficulty of segmenting objects from edge images limits the scope of shapes we can handle.

Fig. 1 shows how we proceed from 2D images to tracked 3D objects.

The remainder of the paper is organized as follows. Section 2 reviews related work. Section 3 explains detection of shapes while their tracking is detailed in Section 4. Section 5 shows experimental results and Section 6 concludes the paper.

2 Related Work

Detection of geometric shapes based on perceptual grouping of edges is a well known topic in computer vision with an abundance of literature since the eighties. [1] introduced geons as a means of describing objects of arbitrary shape by components and [3] discuss the potential of geons in computer vision systems. [20] present a method based on joint statistical constraints defined on complex wavelet transforms to represent and detect geons. These representations are however not size invariant and a sliding window is needed to cover the whole image at different positions and scales. Moreover the test images are rather artificial with grey objects on a black background. Note that our proposed approach does not aim to detect all geons but is tailored to a subset.

[9] uses perceptual grouping of edge segments to reduce the complexity of detecting 3D models in edge images and shows impressive results on highly cluttered images. This however requires precise CAD-like 3D models given a-priori which we want to avoid. Approaches such as [13], [6] and [14] use groups of edge fragments to detect learned classes of shapes and show impressive results on databases. Our models differ in that they are only topological models of generic views of basic shapes such as an arrangement of lines and ellipses forming a cylinder, where the limited number of these shapes allows us to ignore learning.

[4] combine qualitative and quantitative object models to detect and track ten object primitives (box, cylinder, tapered cylinder etc). That system is still brittle in the presence of texture and clutter. To this end [15] describe objects beyond simple shapes but still of limited complexity (cups, hats, desks) with qualitative, parts-based shape abstraction base on a vocabulary of 2D part models corresponding essentially to closed contours of various shapes. Their system can extract such representations from images containing textured objects as well as complex backgrounds.

A major challenge in perceptual grouping is the combinatorial explosion when identifying possible groups of image features as the number of possible groups grows exponentially with group size. [9] addresses the problem of exponential run-time complexity using a grid overlaid on the image indexed by line endpoints. A typical problem of indexing is the appropriate choice of bin size. [18] use further curve parameters to construct index spaces of higher-parametric models and also addressed the problem of bin size and indices close to bin boundaries. [22] proposes to use indexing in the image space where search lines emanating from the ends of image edges are used to find collinearities and junctions and finally closed contours. Search lines are grown incrementally over processing time, thus avoiding problematic distance thresholds.

An overview of monocular model-based 3D tracking is given in [8]. With respect to tracking based on Monte Carlo particle filtering the authors state the following:

> We attribute this relative lack of popularity to two different causes. First a large number of particles, perhaps as many as several thousands when the motion is poorly defined, can be required, which slows down the tracking process.

To cope with this problem our particle filter based approach uses texture information, which is more distinctive than only using geometry edges and therefore requires a smaller number of particles. Secondly we implemented the major part of the particle filter on the graphics processing unit (GPU) which allows us to evaluate many particles efficiently in parallel.

Tracking based on surface texture is of course not new. [10] also use edges and texture for tracking. They extract point features from surface texture and use them together with edges to calculate object pose. This turns out to perform very fast and robust with respect to occlusion. Our approach not only uses patches around a few interest points but the whole texture. Also [17] fuse interest points (FAST features) from surface texture and geometry edges for improved tracking. Whereas in both of the above approaches texture and edges are treated explicitely with different underlying operators our approach treats them the same using one underlying operator, namely edge extraction.

Recent approaches typically rely on (possibly combinations of) edge contours, interest point descriptors and optical flow [2, 19, 21]. While interest point descriptors (such as SIFT, FAST etc.) and optical flow are becoming faster to compute thanks in part to GPU implementations edges are still considerably faster to compute but are of course inherently less discriminative than interest point descriptors.

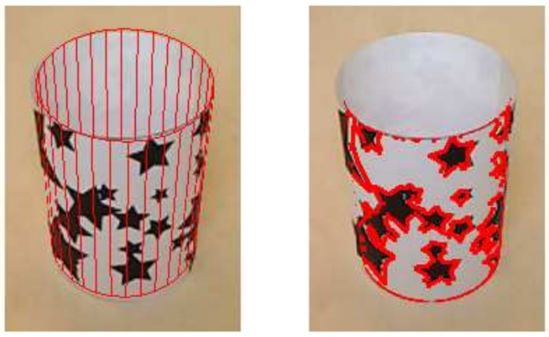


Figure 2: Edges from geometry vs. edges from texture

The work presented in this paper is based on [7] where the authors also take advantage of the GPU by projecting a wire-frame model into the camera image. A particle filter with a Gaussian noise model is used to evaluate the confidence level with respect to the pose.

Our approach not only uses geometry edges but also edge features from textures which extends the class of trackable models to those that have curved surfaces as illustrated on the right of Fig. 2. This is because in a standard 3D model curvature is approximated by triangles and quadrangles which would produce virtual edges which do not correspond to the actual edges as shown on the left of Fig. 2.

We found [12] to be the closest related work to our tracking part, but instead of computing the cross-correlation of a pixel patch we propose to evaluate the match between the edge gradients of the rendered model and the camera image. This means that we have fewer comparisons for each pixel which makes it faster. Furthermore edges are more robust against changing lighting conditions. Our approach was partially presented in [11], with major modifications regarding the particle filtering.

3 Detection of Basic Shapes

In the following sections we show how we detect generic views of basic shapes in edge images by employing an incremental perceptual grouping approach. Having detected 2D shapes we then use a ground plane assumption to generate 3D shape models which are subsequently handed to the tracker.

3.1 Incremental Indexing and Anytimeness

Perceptual grouping in our system is based on the work of [22] which provides an anytime solution to finding junctions between edge segments and subsequently closed contours avoiding the need for arbitrary distance thresholds and [16] which adds higher level features. Indexing is used to efficiently identify candidates for junctions, where the indexing space is the image itself. Each edge endpoint defines a set of search lines consisting of tangential and normal search lines. These search lines are drawn into the index image using Bresenham line drawing. Whenever two lines index into the same bin, i. e. their search lines intersect, we create a new junction. Depending on the types of search lines intersecting we form an L-junction, a collinearity or a T-junction between the respective originating lines. If more than two lines intersect, the according number of pairwise junctions are created. Shortest path search in the resulting graph consisting of edges and junctions then finds closed contours.

In order to avoid the definition of certain length thresholds for search lines they are drawn incrementally, continuously checking for junctions. So the longer we search, the more junctions and eventually closed contours will be found, where "easy" cases typically pop out fast and "difficult" ones (broken edges, partial occlusions, more clutter) follow later. This allows us to stop processing anytime, e. g. after a certain frame time has elapsed or, if we happen to know that we expect precisely three cylinders in the scene, after having found three cylinders.

3.2 Perceptual Grouping

We then define a hierarchy of grouping principles to enable efficient abstraction of image edges into basic geometric Gestalts as shown in Fig. 3. Triggered by the incrementally growing search lines referred to in the above section, lower level Gestalts such as closures or ellipses are formed and in turn trigger formation of higher level Gestalts such as cuboids and cylinders. Concretely cuboids are defined as three overlapping "flaps", where a flap is defined as two rectangles connected along one edge. Ellipses are derived from convex groups of intersecting arcs. Cylinders are defined using ellipse junctions: search lines emanating from the major semi-axes of an ellipse meeting straight lines. Two ellipses and two parallel straight lines (possibly of course comprised itself of several collinear lines) thus make up a cylinder. Cones are defined likewise.

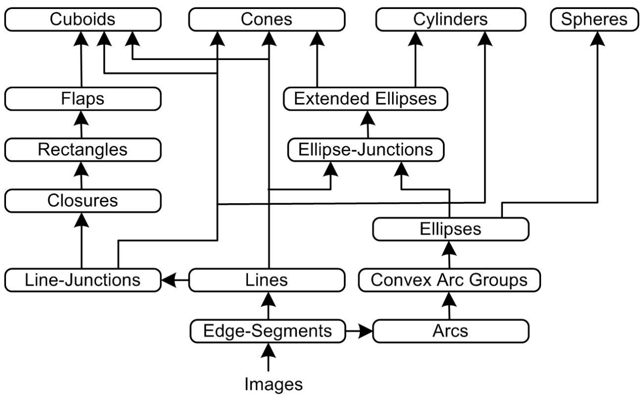


Figure 3: Abstraction hierarchy: from edges to basic shapes

Note that as we move up the abstraction hierarchy the corresponding Gestalts get more and more distinctive. So while we will generally find lots of closures, rectangles and ellipses are already somewhat less frequent. Finally cuboids comprised of three flaps or cylinders and cones being composed of a specific topological arrangement of lines and ellipses already rarely appear accidentally. The next section explains how we further reduce the number of false hypotheses.

3.3 Ranking and Masking

With longer processing time the number of estimated hypotheses will grow exponentially, because the more junctions are found between edges the more combinatorial possibilities for higher level groups will appear. Basically this means we will start accumulating "crappy" hypotheses. However many of these higher level hypotheses will share lower level features, thus essentially providing different interpretations for the same underlying data. We are obviously only interested in the best interpretation.

So we rank all hypotheses according to their significance, which is derived from properties such as parallelism, closeness or completeness of Gestalt. We then traverse the list of hypotheses in order of decreasing significance and mark all visited lower level features. Whenever a hypothesis finds one of its constituent features already marked, it will be masked by the higher ranked owner of that feature.

This additional pruning step gets rid of most false hypotheses. Remaining accidental groupings that happen to constitute valid hypotheses could be identified by changing the viewpoint or more generally by observation over time as typically only correct hypotheses, i. e. actual shapes will be stable over viewpoints. In the context of this paper the tracker described in Section 4 takes care of these as only correct hypotheses will produce stable tracks.

3.4 From 2D to 3D

The following tracking procedure in Section 4 requires a 3D wire-frame model of the detected object shape as well as an initial pose estimate relative to the camera. Note that everything so far is purely 2D, i. e. we detect projections of shapes in generic views onto the image plane. Assuming a camera with known elevation and tilt angle and further assuming that detected objects (cubes, cones, cylinders and spheres) rest on the ground, allows us to convert them to 3D shapes. We intersect view rays with the ground plane and thus obtain 3D position on the plane as well as unambiguous size. Note that we restricted our search to objects belonging to a limited number of simple shape classes. This allows us to "fill in" the unseen backside from simple symmetry considerations.

4 Tracking

Once we have detected objects and generated wire-frame models along with initial pose estimates we initialize a model based 3D pose tracker. Remember that we view our work in the context of robotic applications where we expect to observe manipulated objects or manipulate them ourselves. So it is essential that we have robust and fast estimates of object trajectories.

To improve robustness we enhance the wire-frame models with surface texture, captured directly from the camera image. Tracking based solely on wire-frame edges already provides reasonable performance in many cases but is insufficient for rotationally symmetric objects and runs into problems with degenerate views and heavy background clutter. Adding texture edges provides much more dense information for the tracker to "snap" on to, allowing degenerate views as well as large scale partial occlusions.

To meet real-time requirements we harness the parallel computing power of modern graphics processing units (GPU) for image processing. Graphics boards are designed to render virtual scenes as realistically as possible. The basic idea is to compare those virtual scenes with an image captured from reality. Texturing is a common method of simulating realistic surfaces. In this paper, the edges of those textures are used for comparison. Fast progress in computer graphics will soon allow the inclusion of more and more optical effects such as shadows, reflections, shading, occlusions or even smoke, fire, water or fog. This requires of course available information about the environment and the target object, namely the 3D geometry, color and material and furthermore light sources, occlusions and shadow casting objects and so forth.

The algorithm for tracking is illustrated by Fig. 4. It is a modified version of the well known bootstrap filter in [5] applied to vision based object tracking. With respect to computational costs it can be separated into image processing and particle filtering.

4.1 Image Processing

In the following an object is described by the geometry of its surface S (approximated by polygons and vertices $\mathbf{v}$) and its 6 DOF pose $\mathbf{x}$. Furthermore with the

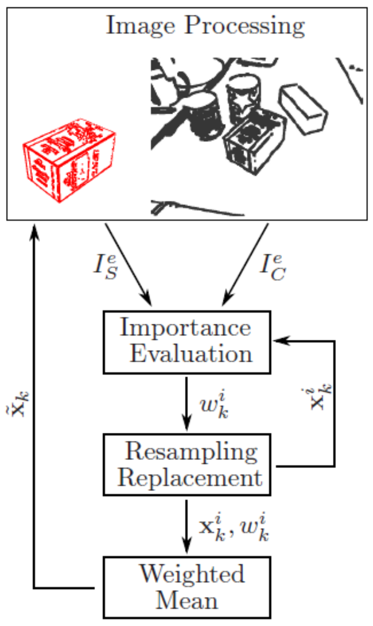


Figure 4: Block scheme of particle filtering

information of the geometry S and the initial pose $\mathbf{x}_d$ of the object as described in Section 3 the color texture I_S can be retrieved from the camera image I_C and mapped onto the surface S.

In the image processing stage first the edge image I_C^e of the incoming camera image I_C is computed. Now the edge image of the object I_S^e and the camera edge image I_C^e could be used to calculate the importance weights as described in Section 4.2. However there are runtime considerations requiring another intermediate step. The particle filter requires rendering the model several hundred times at different poses $(\mathbf{x}^i | i = 1, \ldots, N)$ - which is fast using the GPU - but also extracting edges from the rendered images, which would be too slow to allow frame rate tracking. Therefore we perform the step of extracting edges from a rendered object image only once. To this end we calculate the weighted mean pose of the previous tracking step

$$\tilde{\mathbf{x}}_k = \sum_{i=1}^{N} w_{k-1}^i \mathbf{x}_{k-1}^i \quad (1)$$

and use this "representative" pose for rendering and subsequent edge extraction from the rendered image. This edge image is then re-projected to the object in world space, temporarily replacing the original surface color texture S with a corresponding edge texture S^e. This edge texture is then projected several hundred times instead of the color texture. Again this projection is fast as it uses GPU functions.

Note that the obvious solution of generally and permanently replacing color texture with edge texture in the object model does not work. 1-pixel wide edges when projected at different scales and viewing angles will suffer badly from aliasing effects severely affecting edge matching. Generating, for each new input image,

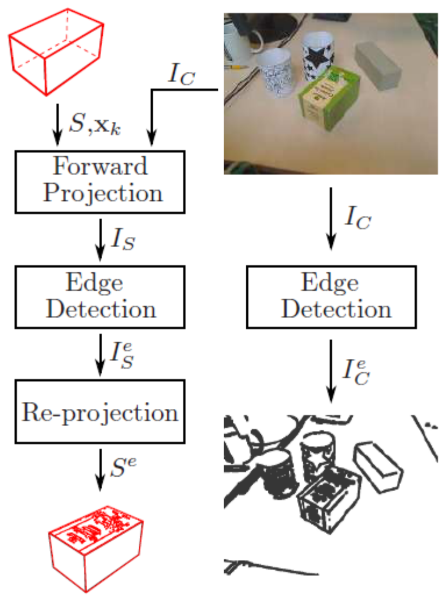


Figure 5: Block scheme of image processing with a tea box. The decision whether a face uses texture information or not depends on the viewing angle and confidence level c_k of the tracking step.

the edge texture once as the edges appear for a "representative" view of the object and using that for rendering of all other (similar) views substantially reduces such effects. Of course this relies on the assumption that the various object pose hypotheses represented by the particles are in fact similar enough, which for normal tracking situations they are.

4.2 Particle Filtering

For each tracking step the particle filter executes the methods shown in Fig. 4. First the particles $\mathbf{x}_0^i, i = 1, \ldots, N$, representing the pose of the object, are generated using Gaussian noise. Then the confidence level c_k^i and importance weight w_k^i of each particle $\mathbf{x}^i$ are evaluated by matching its corresponding edge image against the edge image of the camera I_C^e. According to the importance weights the set of particles is resampled and then perturbed using again Gaussian noise.

The loop formed by the blocks "Importance Evaluation" and "Resampling with Replacement" is executed several times (2-5 times depending on the power of the processor) before proceeding to the step "Weighted Mean". We refer to this as Iterative Particle Filtering. This increases accuracy, and therefore also increases robustness especially when the object is moving.

First particles $\hat{\mathbf{x}}_k^i$ are resampled from the prior particle distribution $\mathbf{x}_{k-1}^i$ according to the importance weights. Then $\hat{\mathbf{x}}_k^i$ is perturbed by adding Gaussian noise $\mathbf{n}(\boldsymbol{\sigma})$ with a standard deviation scaled by the prior confidence level c_{k-1}^i:

$$\begin{aligned} \mathbf{x}_k^i &= \hat{\mathbf{x}}_k^i + \mathbf{n}(\boldsymbol{\sigma}) \\ i &= 1, \ldots, N \end{aligned}$$

The standard deviation is evaluated by

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}(c_{k-1}^i, \mathbf{m}_w)$$

$\mathbf{m}_w$ is the transformation from the normalized Gaussian noise in the range of $[0, \ldots, 1]$, to the metric world coordinates with respect to the object size.

Each particle is tested against the camera image and a confidence level is calculated. To this end the correlation between the gradients of the edges $\mathbf{g}_{S_i}(u, v)$ and $\mathbf{g}_C(u, v)$ is evaluated by comparing the direction of the edges at each image point (u, v).

$$\begin{aligned} \mathbf{g}_{S^i}(u, v) &= \begin{pmatrix} I^e_{S^i,x}(u, v) \\ I^e_{S^i,y}(u, v) \end{pmatrix} \\ \mathbf{g}_C(u, v) &= \begin{pmatrix} I^e_{C,x}(u, v) \\ I^e_{C,y}(u, v) \end{pmatrix} \end{aligned}$$

The angles between those vectors are calculated, producing the edge correlation image Φ_i:

$$\begin{aligned} \phi &= \arccos\left(\mathbf{g}_{S^i} . \mathbf{g}_C\right) \\ \Phi^i(u, v) &= \begin{cases} 1 - \frac{2\phi}{\pi} & \text{if} \quad \phi < \pi/2 \\ 1 - \frac{2|\phi - \pi|}{\pi} & \text{if} \quad \phi > \pi/2 \\ 0 & \text{if} \quad (u, v) \notin \mathbf{v}'_S \end{cases} \end{aligned}$$

Note that it is assumed that the result of the arccos() function lies within 0 and π. The image Φ^i now contains the degree of correlation between the pose suggested by the particle i and the camera image. The angular deviation of the edge angles Φ^i is scaled to the range of 0 to 1.

The confidence level c^i and importance weight w^i are evaluated as follows:

$$\begin{aligned} c_k^i &= \frac{1}{2}\left(\frac{m^i}{n^i} + \frac{m^i}{n_{max}}\right) \\ w_k^i &= (c_k^i)^p \end{aligned} \qquad (2)$$

with

$$\begin{aligned} m^i &= \sum_{u,v} \Phi^i(u, v) \\ n^i &= \sum_{u,v} |I^e_{S^i}(u, v)| \\ n_{max} &\propto \max(n^i | i = 1, \ldots, N) \end{aligned}$$

The first term within the brackets of equation (2) is the percentage of matching edge pixels m^i with respect to the non-matching edge pixels n^i. Calculating the confidence level only with this term would cause the tracker to lock when only one side of the 3D object is visible. If in this degenerate view the object is rotated slightly, another side of the object becomes visible. The particle representing this rotation would typically get less weight than the prior front facing particle. This is because the number of matching pixels m^i grows slower than the number of not-matching pixels n^i when rotating the object out of the front side view. This effect is amplified by the fact that edge detection for strongly tilted faces is less accurate.

The second term allocates more weight to the total number of matching pixels m^i which is intrinsically higher for the rotated particle. n_{max} is the maximum number of visible edge pixels in the actual area and scales the pixels to the proper range. To scale the range of the outcome to $[0, \dots, 1]$ the terms are devided by the sum of their maximum values.

The weights of the particles are calculated by raising c_k^i to the power of p, which controls the speed of convergence of the particles. With a higher power p, w_k^i increases, which leads to more particles assigned to $\mathbf{x}_k^i$ when resampling the whole particle distribution and therefore to a faster convergence.

As explained in Section 4.1 for projection and reprojection of the model, a single pose $\tilde{\mathbf{x}}_k$ is required. We use the weighted mean of the particle distribution as in equation (1), because it shows good results, both in terms of accuracy and smoothness of the resulting pose.

5 Experimental Results and Evaluation

We made experiments and evaluations with the proposed basic shape detector and with the model tracker. The detector and the tracker are running as separated threads in a distributed framework, using shared memory for the exchange of geometry and pose information of objects. While the detector is triggered only every second to detect major changes in the scene, the tracker runs with high priority at frame rate to achieve high pose accuracy. For each new object appearing in the scene, the detector drops a new geometric model to the shared memory. On the other hand the tracker updates the pose of the models in the shared memory, which allows the system to identify a re-detection.

5.1 Detection of Basic Shapes

The incremental grouping method has been evaluated experimentally with a mobile robot moving among simple geometric 3D objects. Fig. 6 shows an example image, with several objects. The picture indicates a typical problem of grouping, namely that shadows or image noise create spurious features such as lines or arcs. A grouping into higher level Gestalts sometimes accidentally includes a wrong feature. With the incremental approach object detection depends on processing time. The first two images present the cluttered edge image and the extended search lines after 468 ms from the voting image. The following images show the detected object shapes after 328 ms and 468 ms processing time.

The proposed method is able to detect non-textured as well as textured objects in real world scenes. As can be expected, the amount of texture, background clutter and occlusion limits detection rates. Gaps in edges due to occlusion are filled in by search lines if they are not too large (otherwise filling in would take rather long). Adjoining faces with the same color tend to lead to weak edges if lighting is uniform, leading to edge detction failure. Also too much texture over shape edges (e.g. for colourful packaging) will cause problems for detection of the geometry edges. Fig. 7 shows results from two indoor table-top scenes.

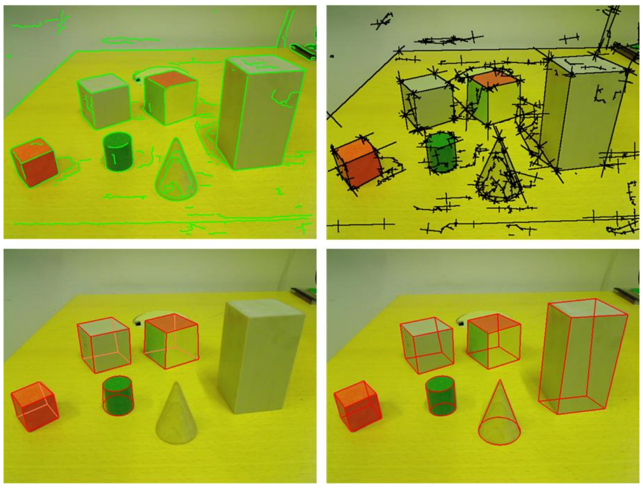


Figure 6: Edge image, voting image with grown search lines and object shapes after 328 ms and 468 ms processing time.

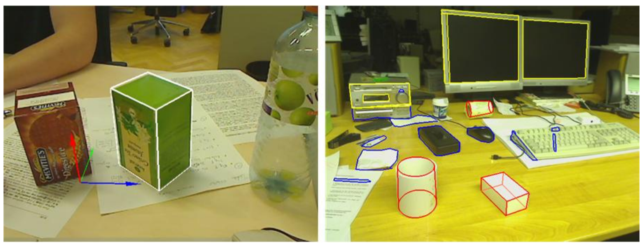


Figure 7: The left image shows a detected textured tea box on cluttered background. The right one illustrates detected object shapes from cubes and cylinders (red) as well as rectangles (yellow) and closures (blue) from lower-level Gestalts in an office scene.

To evaluate the capabilities of the perceptual grouping approach, we explored a playground scene containing several cubes with a mobile robot using different processing times. Tab. 1 shows the results of the different runs using an Intel Core2Duo with 2.5 GHz. The whole scene consists of 148 images within 417 cubes to detect. Our approach allows only detection of basic shapes in generic views. With degenerated views, when only one or two surfaces of a cube are visible, only a detection of rectangles or flaps is possible, which may indicate a cube at this position but is not sufficient to build a cube hypothesis. Therefore it is impossible to reach a perfect detection rate of 100 percent with this method of grouping.

Fig. 8 shows the detection rate graphically from the playground scene with the different processing times. As expected the detection rate increases with increased processing time but also the rate of falsely detected cubes. Under 400 ms the false positives (FP) are few but will increase steadily with increasing processing time. I. e. the longer we search the more "hallucinated" shapes will appear.

Please note that the first 150 ms of processing are used up by Canny edge detection (we use Deriche edge extraction and self-adjusting hysteresis thresholds) and line and arc fitting. Only after that fixed amount of time the incremental processing elements take place and start filling the hierarchy of Gestalts. Hence the detection rate curve starts at 150 ms.

Table 1: True and false positive and negative (TP, FP, TN, FN) detection from a playground scene with four cubes in 148 images at different processing times.

Processing time	TP	FP	TN	FN
150 ms	211	0	-	206
200 ms	294	0	-	123
300 ms	340	1	-	77
400 ms	359	12	-	58
600 ms	378	28	-	39

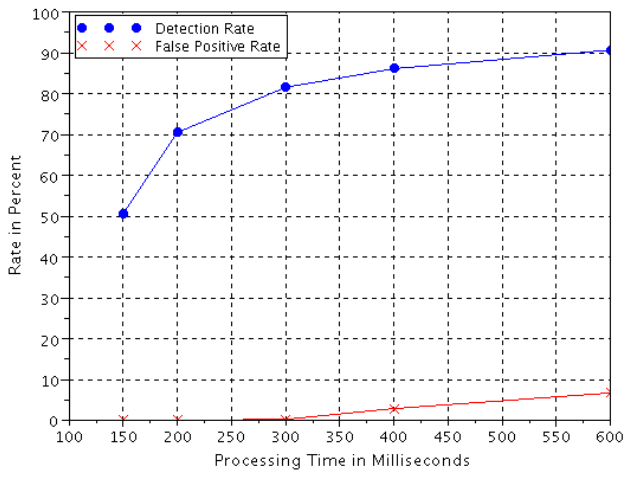


Figure 8: Detection rate from a playground sequence with 148 images and 417 possible cube detections.

5.2 Tracking of Textured Objects

The advantage of tracking using the edges of the object texture is its high robustness against changing lighting conditions, occlusion, reflections and background clutter. In other words, they are very distinctive, robust and only a relatively small portion of the surface needs to be visible to determine the correct pose. Of course the latter is only true if the visible surface is rich in texture.

Robustness comes with at the cost of speed, which corresponds to the number of particles used. Fig. 10 illustrates the dependency between frame rate and number of particles, where the red shaded area indicates either too low frame rate (bottom) or too few particles (left) which causes jittering and loss of object. We experienced that running the particle filtering loop several times within one image but with less particles further increases robustness without wasting calculation time.

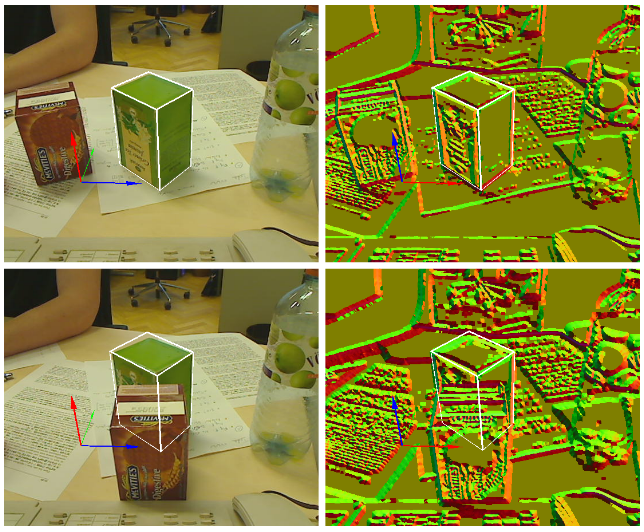


Figure 9: Robustness of tracking. The detected box (upper left image) is occluded by another box in the next frames (lower left image). The algorithm is still capable to track the object although accuracy and the maximum trackable speed drops. The right images show the output of the edge detection algorithm used and illustrate the high level of clutter in the scene (different colors represent edge orientations).

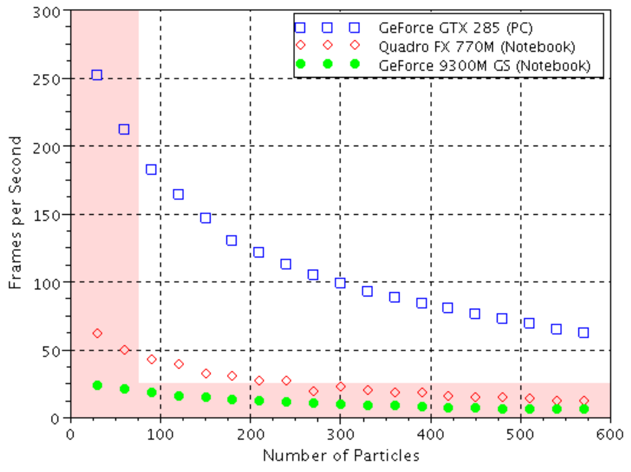


Figure 10: Frame rate with respect to the number of particles

6 Conclusion and Further Work

We presented a system for detection and tracking of basic geometric objects. Shape detection is based on a hierarchical perceptual grouping system where the use of an incremental processing approach eliminates the need for many thresholds and parameters in various Gestalt principles. Assuming known camera elevation and tilt we use the ground plane to generate 3D wire-frame models which are subsequently covered with surface texture and tracked by a particle filter based tracker.

Tracking converges quickly to the correct pose and is able to handle large deviations, for example when initializing. Also partial occlusion, reflections, light changes, shadows and cluttered background are handled thanks to the increase in robustness with the use

Table 2: Frame rate with respect to number of polygons of the geometrical model with different number of recursions and particles, computed on a GeForce 285 GTX

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
Complex Scene	1536	160	60	18

of texture edges besides simple wire-frame edges. Exploiting the power of a graphics processing unit allows high tracking speed at frame rate.

We are currently working to remove the ground plane assumption required to get from 2D to 3D shape by using stereo, where we perform stereo matching not on pixel but on higher feature level. These are generally much more distinctive than small image patches used in dense stereo and matching of edges allows high accuracy in the stereo reconstruction.

Tracking rate is limited by the number of particles required to obtain sufficiently accurate pose. The number of particles can be reduced by employing a better motion model (right now we simply assume a 0-th order motion model, i. e. a static object). A first order motion model could already improve tracking during smooth trajectories. But actually it is the non-smooth trajectories that pose the real problems and here first order motion models would not help. So we plan to incorporate predictive models from a physics simulation to predict events such as sudden stops when a falling object hits the ground.

Another important issue is the ability of the tracker to report lost tracks, which would result in a re-detection request to the detector. Deciding when a tracked object is truly lost and not just partially occluded or entering strong shadow is non-trivial. To this end we are working on robust tracking confidence measures.

Acknowledgement

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] I. Biederman. Recognition-by-components: A theory of human image understanding. Psychological Review, Vol. 94(2):115–147, 1987.

[2] T. Brox, B. Rosenhahn, J. Gall, and D. Cremers. Combined region- and motion-based 3D tracking of rigid and articulated objects. Transactions on Pattern Analysis and Machine Intelligence, 32, 2009.

[3] S. Dickinson, R. Bergevin, I. Biederman, J. Eklund, R. Munck-Fairwood, A. K. Jain, and A. Pentland. Panel report: The potential of geons for generic 3-d object recognition. In Image and Vision Computing, Vol. 15(4):277–292, 1997.

[4] S.J. Dickinson and D. Metaxas. Integrating Qualitative and Quantitative Object Representations in the Recovery and Tracking of 3-D Shape. In L. Harris and M. Jenkin, editors, Computational and Psychophysical Mechanisms of Visual Coding, pages 221–248. Cambridge University Press, New York, 1997.

[5] A. Doucet, N. De Freitas, and N. Gordon, editors. Sequential Monte Carlo methods in practice. Springer, 2001.

[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. Klein and D. Murray. Full-3d edge tracking with a particle filter. Proc 17th British Machine Vision Conference, 2006.

[8] V. Lepetit and P. Fua. Monocular model-based 3d tracking of rigid objects: A survey. Foundations and Trends in Computer Graphics and Vision, Vol. 1, No. 1, 2005.

[9] D. G. Lowe. Three-dimensional object recognition from single two-dimensional images. Artificial Intelligence, Vol. 31(3):355–395, 1987.

[10] L. Masson, M. Dhome, and F. Jurie. Robust real time tracking of 3d objects. ICPR '04: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 4, 2004.

[11] T. Moerwald, M. Zillich, and M. Vincze. Edge tracking of textured objects with a recursive particle filter. 19th International Conference on Computer Graphics and Vision, Moscow, Russia, 2009.

[12] E. Murphy-Chutorian and M.M. Trivedi. Particle filtering with rendered models: A two pass approach to multi-object 3d tracking with the gpu. Computer Vision and Pattern Recognition Workshop, 2008.

[13] Randal C. Nelson and Andrea Selinger. A cubist approach to object recognition. Technical Report TR689, Dept. of Computer Science, Univ. of Rochester, 1998.

[14] B. Ommer and J. Malik. Multi-Scale Object Detection by Clustering Lines. In International Conference on Computer Vision, 2009.

[15] P. Sala and S. Dickinson. Model-Based Perceptual Grouping and Shape Abstraction. In The Sixth IEEE Computer Society Workshop on Perceptual Grouping in Computer Vision (POCV), 2008.

[16] A. Richtsfeld and M. Vincze. Basic object shape detection and tracking using perceptual organization. International Conference on Advanced Robotics (ICAR), pages 1–6, June 2009.

[17] Edward Rosten and Tom Drummond. Fusing points and lines for high performance tracking. In IEEE International Conference on Computer Vision, volume 2, pages 1508–1511, October 2005.

[18] S. Sarkar and K. L. Boyer. A computational structure for preattentive perceptual organization: Graphical enumeration and voting methods. IEEE Transactions on System, Man and Cybernetics, Vol. 24(2):246–266, 1994.

[19] T. Schoenemann and D. Cremers. A combinatorial solution for model-based image segmentation and real-time tracking. Transactions on Pattern Analysis and Machine Intelligende, 32, 2009.

[20] X. Tang, K. Okada, and C. von der Malsburg. Represent and Detect Geons by Joint Statistics of Steerable Pyramid Decomposition. Technical Report 02-759, Computer Science Department, University of Southern California, 2002.

[21] M. Werlberger, T. Pock, M. Unger, and H. Bischof. A variational model for interactive shape prior segmentation and real-time tracking. SSVM, 2009.

[22] Michael Zillich and Markus Vincze. Anytimeness Avoids Parameters in Detecting Closed Convex Polygons. In The Sixth IEEE Computer Society Workshop on Perceptual Grouping in Computer Vision (POCV), 2008.

BLORT - The Blocks World Robotic Vision Toolbox

T. Mörwald, J. Prankl, A. Richtsfeld, M. Zillich and M. Vincze

***Abstract*— The vision and robotics communities have developed a large number of increasingly successful methods for tracking, recognising and on-line learning of objects, all of which have their particular strengths and weaknesses. A researcher aiming to provide a robot with the ability to handle objects will typically have to pick amongst these and engineer a system that works for her particular setting. The work presented in this paper aims to provide a toolbox to simplify this task and to allow handling of diverse scenarios, though of course we have our own particular limitations: The toolbox is aimed at robotics research and as such we have in mind objects typically of interest for robotic manipulation scenarios, e.g. mugs, boxes and packaging of various sorts. We are not aiming to cover articulated objects (such as walking humans), highly irregular objects (such as potted plants) or deformable objects (such as cables). The system does not require specialised hardware and simply uses a single camera allowing usage on about any robot. The toolbox integrates state-of-the art methods for detection and learning of novel objects, and recognition and tracking of learned models. Integration is currently done via our own modular robotics framework, but of course the libraries making up the modules can also be separately integrated into own projects.**

I. Introduction

Even with the large pool of powerful methods for learning, recognition and tracking of objects available today, putting together a system that "simply works" can be a time-consuming task. Objects need to be recognised and tracked, and learning of models should not be too cumbersome, if possible on-line and require no extra hardware besides the robot itself. Simple as well as complex scenes should be covered, with no need to either place markers on objects or paint them in bright colours. A sufficiently large number of objects should be detectable and trackable in real-time, with no constraints on object type or shape. And if the task requires manipulation full 6D object pose and shape are required.

Recent increasingly successful approaches in object recognition and tracking are typically based on some variant of interest points and a combination of offline training and online recognition. Training often requires either hand-labelling of images or presentation of objects on specialised equipment like turn tables, which is sometimes not desirable. Approaches for learning by showing use e.g. optical flow for segmenting the object of interest from the background and then train an interest points-based model, accumulating new interest points as the object is rotated. These approaches however typically create a "sparse" 3D model consisting of several interest points rather than a model of the actual physical surface. Calculating points of contact for grasping or touching an object however requires a dense surface model such as a triangle mesh. Furthermore not all methods are amenable to real-time tracking, which is needed as soon as the robot or a human handles objects and the scene becomes dynamic.

T. Mörwald, J. Prankl, A. Richtsfeld, M. Zillich and M. Vincze are with the Vienna University of Technology, Vienna, Austria, email: [moerwald, prankl, arichtsfeld, zillich, vincze]@acin.tuwien.ac.at

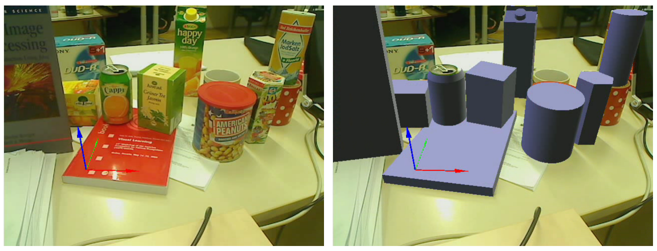


Fig. 1. Scene and recognised/tracked objects

With BLORT, the blocks world robotic vision toolbox, we combine state-of-the-art methods for learning, recognising and tracking objects modelled as dense triangle meshes. Fig. 1 shows an example of a scene containing several (partly occluded) objects and their overlaid 3D models after recognition. Specifically we employ a combination of edge-based detection of basic geometric shapes, fast edge-based monte carlo particle filter tracking and SIFT-based recognition. The rationale behind this particular choice of methods is twofold. First we want to enable human tutor driven learning-by-showing as well as completely automatic on-line model acquisition by the robot. For the latter we have to make simplifying assumptions to allow segmentation of unknown objects and it is here where blocks world enters: detection of novel objects is limited to simple geometric shapes, namely cuboids and cylinders. While this is admittedly a considerable limitation, it allows experiments where a robot interacts with a (simple) scene fully autonomously. More complex objects can be acquired in a learning-by-showing manner but require known 3D triangle mesh models to start with. Second, using edge-based methods covers both textured and untextured objects. Tracking will be improved by available surface texture but works satisfactorily without. SIFT-based recognition obviously requires texture but can be substituted with the edge-based detection.

As a motivating example let us consider a scenario where a human tutor shows a new object to the robot within a learning setup that is intended to make things easy initially, i.e. the tutor basically puts the object on a table and says something like "This is a tea box." The system detects this new object

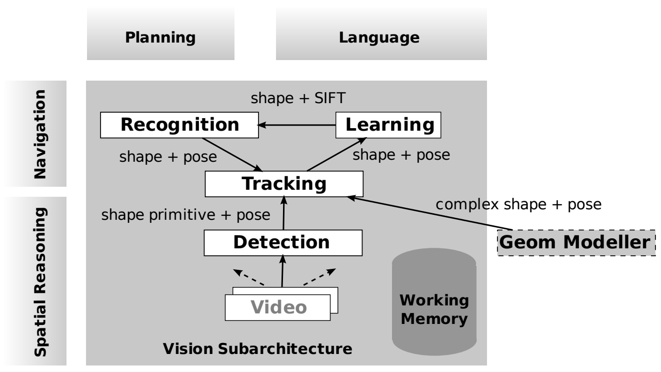


Fig. 2. System overview: interplay between detection, tracking and recognition

and starts tracking it. The tutor then picks up the object and shows it from different sides while the system learns the different object views. Alternatively the robot could go round the object itself, e.g. using an arm-mounted camera. The system is then able to recognise the learned object and re-initialise the tracker in more general scenes, with all the background clutter and varying lighting that are typical of robotic scenarios.

Fig. 2 shows the interplay between detection, tracking and recognition within our robotics framework, where other parts of the system concerned with localisation, planning, language understanding etc. are only hinted at. The framework is based on a software architecture toolkit [1] which handles issues like threading, lower level drivers and communication via shared working memories. Tracking starts when a new object is detected automatically or a model is supplied explicitly. While the object is tracked, initially only based on its wireframe model, surface texture and SIFT features are collected and mapped onto the object surface. This builds up a SIFT-based model that is used later to re-initialise the tracker.

The remainder of this paper is organised as follows. After reviewing related work in Section II we describe detection of novel objects in Section III, followed by tracking in Section IV and recognition in Section V. Experimental evaluation is presented in Section VI followed by conclusion and outlook in Section VII. Note that size constraints prevent us from going into the details of the involved methods and the reader is referred to referenced previous work. The focus of this paper lies on showing the interplay between methods.

II. Related Work

There are many approaches for obtaining 3D object models. The simplest option would be to just download a model from the rich selection on Google 3D Warehouse, if the particular object happens to be modelled (which is quite likely for things like coke cans or bottles). Professional solutions for building 3D models from multiple images such as 2D3® exist but tend to be costly. Systems like the low-cost laser scanning system by Winkelbach et al. [2] or the projected-light system by Rusinkiewicz et al [3] allow quick and easy capture of full 3D models but require special hardware setups.

A very convenient way to construct models is by simply showing an object and adding model information as it is rotated. The system by Brown et al. [4] builds 3D wireframe models of simple geometric objects. The user however has to initialise a 2D wireframe model by selecting image lines and model acquisition takes around 5 minutes. Vacchetti et al. [5] track a 3D CAD model (which has to be specified in advance) and augment it with distinctive features, first from user supplied keyframes in a training phase and also online while tracking. Only a very rough 3D model (e.g. just an ellipsoid) is needed for the system by Özuysal et al. [6], which learns object models for tracking-by-detection by "harvesting" features. The user aligns the model with the first image where keypoints based on randomised tree classifiers are learned and mapped to the 3D model. New features are successively added while the object is tracked using the available features. In [7] the same authors use marking of the outline of a planar object in a single training image in a similar tracking-by-detection approach. Tracking-by-detection has the advantage that lost tracks are recovered immediately. The models are however somewhat costly in terms of memory (in the order of hundred MB) which quickly becomes an issue even on modern hardware. Grabner et al. [8] use a similar approach of accumulating keypoint classifiers while tracking. Only a region of interest near the object center is needed for initialisation. The approach requires no explicit 3D model but also assumes planar objects. Roth et al. [9] use background subtraction to initialise an MSER-based tracker and incrementally learn a PCA model of the object while it is tracked. These models however do not represent 3D shape as would be needed by typical robotics tasks. Riemenschneider et al. [10] take a similar approach of tracking based on MSER but learn a model based on a SIFT codebook. Again however the models do not represent 3D shape. Pan et al.'s ProFORMA system [11] allows to interactively build high quality dense triangle meshes by reconstructing and tracking a model while it is rotated by the viewer. No constraints are placed on the type of object to be modelled other than it be textured.

Gordon and Lowe [12] build a 3D model composed of SIFT descriptors in an offline training phase by performing structure and motion estimation. The online phase then uses RANSAC to estimate 6D pose from 2D-3D correspondences. The system though is geared at augmented reality applications and the scene is not segmented into objects. Collet et al. [13] extend the above for application in the robotics domain, specifically by augmenting RANSAC with a Mean-Shift clustering step to allow recognition of multiple instances of the same object. The system does require manual segmentation of the object in each training image though. Furthermore the obtained sparse 3D points model has to be manually aligned with a CAD model of the object, so the whole procedure requires considerable user intervention.

Tracking of 3D object models has a long history (see Lepetit and Fua [14] for an overview) and we are here mostly concerned with approaches that use surface texture or combine it with model geometry. Masson et al. [15] use point features from surface texture in conjunction with edges to increase robustness especially with respect to occlusion. Rosten and Drummond [16] present a similar approach fusing FAST features with edges. Both of the above treat edges and texture point features independently and fuse them explicitly. Klein and Murray [17] take advantage of the large processing power of recent GPUs for tracking a wire-frame model using a particle filter. Murphy and Trivedi [18] follow a similar approach but use surface texture by computing the cross-correlation of pixel patches.

Detection of geometric shapes based on perceptual grouping of edges is a well known topic in computer vision with an abundance of literature since the eighties. Approaches such as [19], [20] and [21] use groups of edge fragments to detect learned classes of shapes and show impressive results on databases. Our models differ in that they are only topological models of generic views of basic shapes such as an arrangement of lines and ellipses forming a cylinder, where the limited number of these shapes allows us to ignore learning. Dickinson and Metaxas [22] combine qualitative and quantitative object models to detect and track ten object primitives (box, cylinder, tapered cylinder etc). That system is still brittle in the presence of texture and clutter. To this end Sala and Dickinson [23] describe objects beyond simple shapes but still of limited complexity (cups, hats, desks) with qualitative, parts-based shape abstraction base on a vocabulary of 2D part models corresponding essentially to closed contours of various shapes. Their system can extract such representations from images containing textured objects as well as complex backgrounds.

III. Detection of Novel Objects

Detection of unknown objects, with segmentation from a possibly cluttered background is a notoriously difficult problem and we thus have to make several simplifying assumptions. We use a robot mounted camera which allows us to at least assume the ground (or table) plane and that objects for training initially rest on this ground plane. Furthermore we restrict our search to objects belonging to simple shape classes (cuboids and cylinders) which are detected in generic views from edge images. Detection is based on an incremental perceptual grouping approach and outputs 2D projections of shape primitives. We then use the ground plane constraint to generate 3D wireframe models.

A. *Incremental Indexing and Anytimeness*

Perceptual grouping is based on previous work of Zillich and Vincze [24] which provides an anytime solution to finding junctions between edge segments and subsequently closed contours avoiding the need for arbitrary distance thresholds and Richtsfeld and Vincze [25] which adds higher level primitives such as cuboids, cylinders and cones. Indexing is used to efficiently identify candidates for junctions, where the indexing space is the image itself. Each edge endpoint defines a set of search lines consisting of tangential and normal search lines. These search lines are drawn into the index image using Bresenham line drawing, essentially voting for junctions. Whenever two lines index into the same bin, i.e. their search lines intersect, we create a new junction. Depending on the types of search lines intersecting we form an L-junction, a collinearity or a T-junction between the respective originating lines. If more than two lines intersect, the according number of pairwise junctions are created. Shortest path search in the resulting graph consisting of edges and junctions then finds closed contours.

In order to avoid the definition of certain length thresholds for search lines they are drawn incrementally, continuously checking for junctions. So the longer we search, the more junctions and eventually closed contours will be found, where "easy" cases typically pop out fast and "difficult" ones (broken edges, partial occlusions, more clutter) follow later. This allows us to stop processing any time, e. g. after a certain frame time has elapsed or, if we happen to know that we expect precisely three cylinders in the scene, after having found three cylinders.

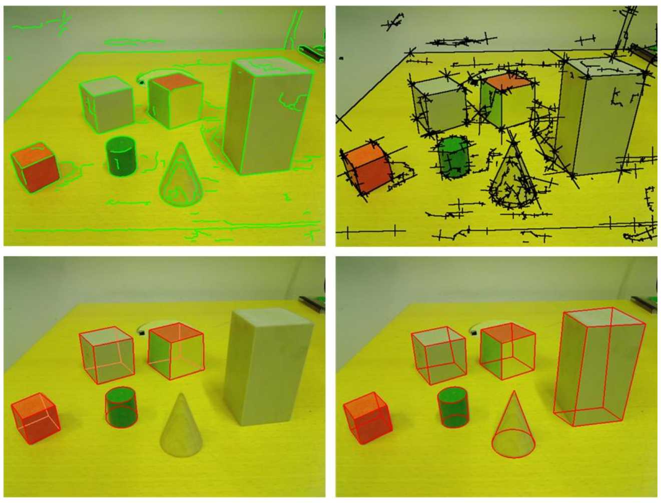


Fig. 3. Edge image, voting image with growing search lines and object shapes after 300 ms and 450 ms processing time.

B. *Perceptual Grouping*

We then define a hierarchy of grouping principles to enable efficient abstraction of image edges into basic geometric primitives. Triggered by the incrementally growing search lines referred to in the above section, lower level primitives such as closures or ellipses are formed and in turn trigger formation of higher level primitives such as cuboids and cylinders as seen in Fig. 3. Concretely cuboids are defined as three rectangles pairwise connected along an edge. Cylinders are defined as two ellipses and two parallel straight lines. Note that as we move up the abstraction hierarchy the corresponding primitives get more and more distinctive. So while we will generally find lots of closures, rectangles and ellipses are already somewhat less frequent. Finally cuboids comprised of three rectangles or cylinders being composed of a specific topological arrangement of lines and ellipses already rarely appear accidentally.

C. From 2D to 3D

The following tracking procedure in Section IV requires a 3D wire-frame model of the detected object shape as well as an initial pose estimate relative to the camera. Note that everything so far is purely 2D, i. e. we detect projections of shapes in generic views onto the image plane. Assuming a camera with known elevation and tilt angle and further assuming that detected objects (cubes, cones, cylinders and spheres) rest on the ground, allows us to convert them to 3D shapes. We intersect view rays with the ground plane and thus obtain 3D position on the plane as well as unambiguous size and fill in the unseen backsides from simple symmetry considerations.

IV. Tracking Known Objects

Tracking is based on edges and a particle filter for 6D pose estimation. The tracker uses geometry edges (surface discontinuities and contour edges) as well as edges resulting from surface texture (if present) both of which are treated the same, i. e. we make no explicit distinction between tracking edges and tracking texture. Texture is mapped onto the surface while tracking as part of the learning procedure. Texture edges are generally more numerous and thus lead to improved robustness. All time-consuming parts of the tracker (such as matching of edge images) are implemented on a GPU allowing tracking at frame rate. More details can be found in Moerwald et al.[26] and Richtsfeld et al. [27].

The algorithm for tracking is illustrated in Fig. 5. It is a modified version of the well known bootstrap filter in Doucet et al. [28] applied to vision based object tracking. With respect to computational costs it can be separated into *image processing* and *particle filtering*.

A. Image Processing

In the following an object is described by the geometry of its surface S (approximated by polygons and vertices $\mathbf{v}$) and its 6 DOF pose $\mathbf{x} = [R \,|\, t]$. Furthermore with the information of the geometry S and the initial pose $\mathbf{x}_d$ of the object as described in Section III the colour texture I_S can be retrieved from the camera image I_C and mapped onto the surface S.

In the image processing stage shown in Fig. 4 first the edge image of the incoming camera image is computed. This is to be compared with the edge images of all projected object hypotheses represented by the particles to finally weight each particle with a matching score. Rendering each hypothesis with its texture is fast using the GPU but edge detection on each rendered image would take too long. On the other hand storing thinned edge images as surface texture instead of normal colour texture leads to bad aliasing effects for smaller projected scales. So we forward-project the colour texture once using a "representative" current pose (the weighed mean of all particles), do edge detection on the rendered image and back-project the edge image as (temporary for this frame) surface texture. Thus rendering of the edge texture happens

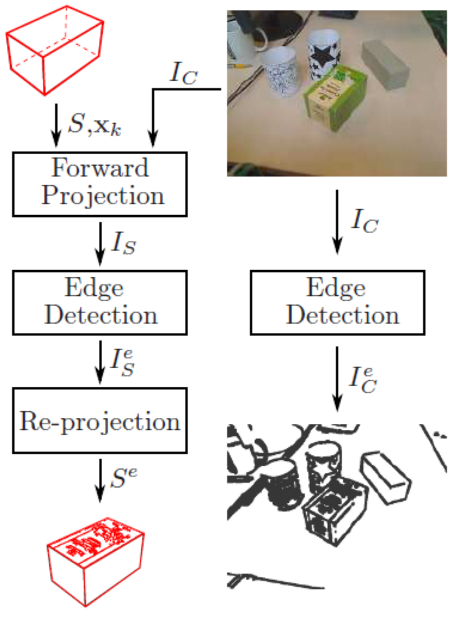


Fig. 4. Block scheme of image processing. The decision whether a face uses texture information or not depends on the viewing angle and confidence level c_k of the tracking step.

at or near the appropriate current scale and aliasing effects are drastically reduced.

B. Particle Filtering

For each tracking step the particle filter executes the methods shown in Fig. 5. First the particles $\mathbf{x}_0^i, i = 1, \ldots, N$, each representing a different pose hypothesis of the object, are generated using Gaussian noise. Then the confidence level c_k^i and importance weight w_k^i of each particle $\mathbf{x}^i$ are evaluated by matching its corresponding edge image against the edge image of the camera I_C^e. According to the importance weights the set of particles is resampled and then perturbed using again Gaussian noise.

The loop formed by the blocks "Importance Evaluation" and "Resampling with Replacement" is executed several times (2-5 times depending on the power of the processor). We refer to this as Iterative Particle Filtering. It increases accuracy and increases robustness especially in cases where the object is moving fast.

First particles $\hat{\mathbf{x}}_k^i$ are resampled from the prior particle distribution $\mathbf{x}_{k-1}^i$ according to the importance weights. Then $\hat{\mathbf{x}}_k^i$ is perturbed by adding Gaussian noise with a standard deviation scaled by the prior confidence level c_{k-1}^i. Each particle is tested against the camera image and its current confidence level is calculated. To this end the correlation between the gradients of the edges of the camera edge image and the particle edge image is evaluated by comparing the direction of the edges at each image point (u, v), producing the edge correlation image Φ_i. The image Φ^i now contains the degree of correlation between the pose suggested by the particle i and the camera image. The angular deviation of the edge angles Φ^i is scaled to the range of 0 to 1.

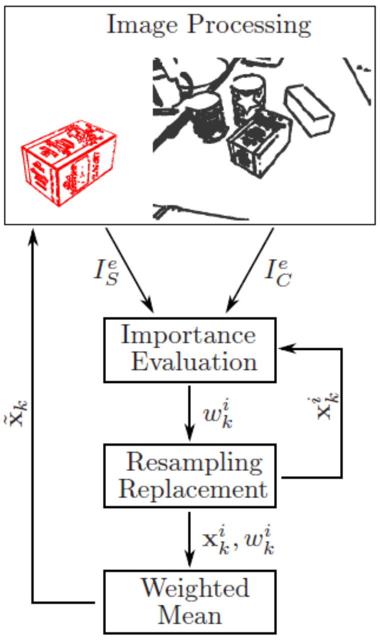


Fig. 5. Block scheme of particle filtering

The confidence level c^i and importance weight w^i are evaluated as follows:

$$c_k^i = \frac{1}{2}\left(\frac{m^i}{n^i} + \frac{m^i}{n_{max}}\right) \quad (1)$$
$$w_k^i = (c_k^i)^p$$

with

$$m^i = \sum_{u,v} \Phi^i(u,v)$$
$$n^i = \sum_{u,v} |I_{S^i}^e(u,v)|$$
$$n_{max} \propto \max(n^i | i = 1, \dots, N)$$

The first term within the brackets of Eq. (1) is the percentage of matching edge pixels m^i with respect to the total visible edge pixels n^i. Calculating the confidence level only with this term would cause the tracker to lock when only one side of the 3D object is visible. If in this degenerate view the object is rotated slightly, another side of the object becomes visible. The particle representing this rotation would typically get less weight than the prior front facing particle as the number of matching pixels m^i grows slower than the number of non-matching pixels n^i when rotating the object out of the front side view. This effect is amplified by the fact that edge detection for strongly tilted faces is less accurate. The second term allocates more weight to the total number of matching pixels m^i which is intrinsically higher for the rotated particle. n_{max} is the maximum number of visible edge pixels in the actual area and scales the pixels to the proper range.

The weights of the particles are calculated by raising c_k^i to the power of p, which controls the speed of convergence of the particles. With a higher power p, w_k^i increases, which leads to more particles assigned to $\mathbf{x}_k^i$ when resampling the whole particle distribution and therefore to a faster convergence. As explained in Section IV-A for projection and re-projection of the model, a single representative pose $\tilde{\mathbf{x}}_k$ is required, where we use the weighted mean of the particle distribution.

V. Learning and Recognising Objects

While edges are well suited for fast tracking we use highly discriminating SIFT features for recognition (where again we use a GPU implementation [29]). For recognition we follow a standard approach similar to Gordon and Lowe [12] and Collet et al. [13] but our training phase differs in that we do not build a sparse 3D SIFT point model via bundle adjustment but use the 3D pose and object geometry already provided by the tracker and simply map SIFT features onto that.

During the learning phase SIFT features 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 by the user via a button press. According to Lowe's findings [30], that SIFT can be reliably detected up to a view point change of about 30°, about 6 keyframes taken from around the object plus 2 for top and bottom are typically sufficient. A larger number of keyframes will result in improved recognition rates for tough cases like large scale occlusions or scale changes, but comes at the cost of increased recognition time per object.

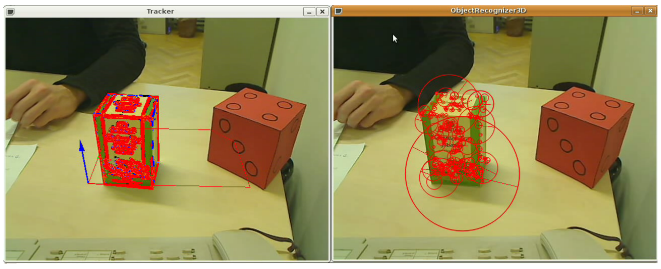


Fig. 6. Learning phase: Edge-based tracking (left) and learned SIFT features (right) during learning phase

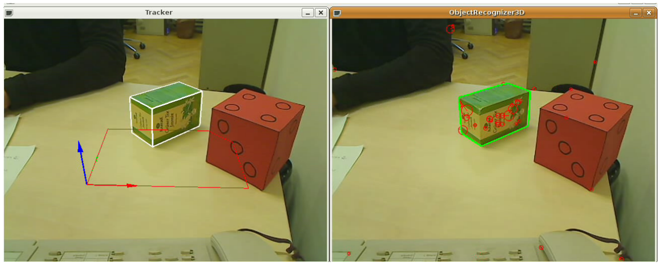


Fig. 7. Recognition phase: Re-initialised tracker (left) after SIFT based recognition (right)

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 with the codebook. According to the codebook entry each matched feature has several corresponding 3D model locations. To robustly estimate the 6D object pose we use the *OpenCV* pose estimation procedure in a RANSAC [31] scheme with a probabilistic termination criterion, where the number of iterations necessary to achieve a desired detection probability is derived from an estimate of the inlier ratio, which is taken to be the inlier ratio of the best hypothesis so far. So the number of RANSAC iterations is adapted to the difficulty of the current situation and accordingly easy cases quickly converge.

To distinguish between hallucinating false detections and correct object locations we define the object confidence

$$p(o\,|\,m, f_t) = \frac{n_{inlier}}{n_{detected}}$$

of an object o for a given image frame f_t and an object model m as the ratio between the matched interest points n_{inlier} and the number of detected interest points $n_{detected}$ located within the boundary projected to the current image. This provides a good estimate independent of the total number of features in the model and independent of current scale.

Fig. 6 shows an example of the learning phase. On the left we see the tracked objects with overlaid texture edges. The right image shows the detected SIFT features of that view. Fig. 7 shows a recognised object on the right (again with overlaid SIFT features) and the re-initialised tracker on the left.

VI. Experimental Results

We present preliminary results. To evaluate our toolbox we learned 10 object models and recognised them in different lighting conditions and scales. Some objects were simple shapes (box- and cylinder-shaped packaging) that were acquired using the basic shape detector as explained in Section III. Others were more complex (though still of the packaging sort) and for these we downloaded models from Google 3D Warehouse. Furthermore we qualitatively evaluated tracking performance in cases of severe occlusion and scale change.

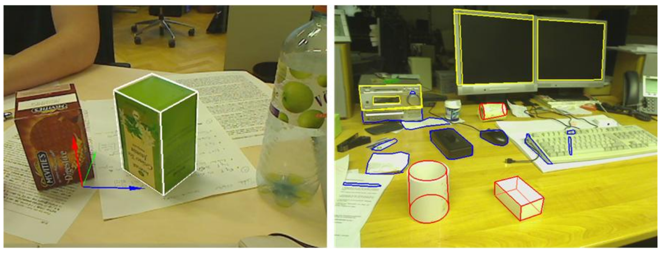


Fig. 8. Detecting a box (left) and the limits of shape detection (right) where red indicates detected shapes and other colours lower level primitives.

Fig. 8 (left) shows a typical case of learning a box-shaped object. Note that shape detection is able to handle moderate amounts of background clutter, no clean white background is required but a typical office table suffices. Also surface texture (with the spurious edges it creates) can be handled as long as the texture does not cross object sides, thus rendering geometry edges all but invisible. Fig. 8 (right) shows the limits of shape detection. The toilet roll lying sideways in the background is detected but results in a bad model and the black box-shaped object is lacking internal geometry edges (only its contour is detected) due to low contrast and thus is not detected as a cuboid.

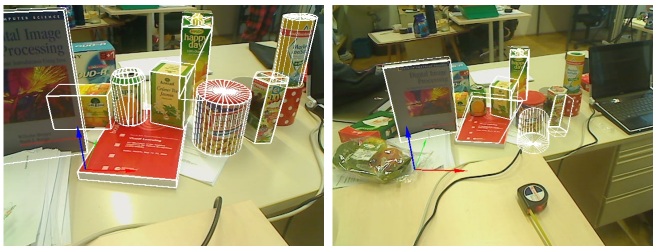


Fig. 9. Good (left) and bad (right) examples of recognition at 0.5 m and 1.0 m respectively, illustrating how recognition performance for some objects degrades with object distance

Table I shows object recognition rates for two lighting situations (sunny noon and artificial light) and two distances of the camera from the scene (0.5 and 1.0 m) and Fig. 9 shows some typical examples. Recognition rates for each scene were taken over 10 images of that scene. As can be seen some objects like the green "Jasmin" tea box (in the center) with its good texture are very stable, while others like the "JodSalz" (yellow cylinder on the right) suffer badly from scale change. Lighting variations did not matter much, as is to be expected from SIFTs invariance to lighting.

TABLE I
Recognition rates in percent and avg. recognition times per object for different lighting situations and distances, over 10 runs each.

light	noon		artificial	
distance [m]	0.5	1.0	0.5	1.0
Cappy	90	30	100	0
GuteLaune	100	0	100	10
HappyDay	100	70	100	90
Jasmin	100	100	100	100
JodSalz	100	0	100	40
Koala	100	100	100	100
peanuts	100	100	100	80
Digital	100	100	100	100
Visual	90	20	100	100
avg. time [s]	1.023	2.772	0.829	2.403

Note that even in cases where the recogniser's object pose is slightly mis-aligned the tracker, once initialised, typically "snaps" on to the correct object pose quickly.

Fig. 10 shows qualitative results of tracking robustness. In Fig. 10 (top) an object is occluded by the humans hand while being moved and a stable track can be maintained. In Fig. 10 (center) one object is moved in front of the other leading to

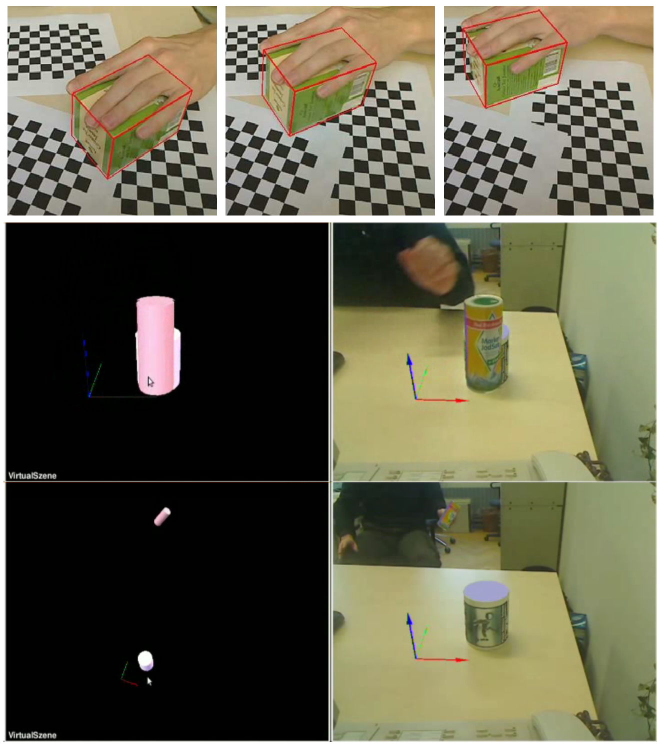


Fig. 10. Robustness of tracking against large scale occlusion (top and center) and large scale variation as well as background clutter (bottom).

even more severe occlusion of the latter. The tracker can still hold on to the occluded object, though in this case it is not moving. Generally accuracy and the maximum trackable speed drop with increasing occlusion. Fig. 10 (bottom) shows robustness with respect to large scale change.

TABLE II

FRAME RATE WITH RESPECT TO NUMBER OF POLYGONS OF THE GEOMETRICAL MODEL WITH DIFFERENT NUMBER OF RECURSIONS AND PARTICLES, COMPUTED ON A GEFORCE 285 GTX

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
Complex Scene	1536	160	60	18

Table II shows frame rates of tracking for different models and different number of particles per model. Tracking rate does not depend too strongly on the number of faces in the triangle mesh of the model and is directly related to the number of particles. Note that tracking accuracy and maximum trackable speed increase with number of particles. The number of particles can be changed dynamically for each object while tracking to e.g. maintain a given frame rate or desired accuracy.

Fig. 1 shows a scene with 9 tracked objects, which are all static. So a few particles per object suffice. As soon as an object moves, a higher number of particles will be required to maintain a stable and accurate track. Also other attentional cues (such as a human hand approaching an object that is likely to be moving soon) could be used to intelligently schedule the pool of particles amongst the objects. This is left for future research.

VII. Conclusion and Future Work

We have presented a toolbox for learning, recognising and tracking objects aimed at robotics research, concentrating on classes of objects that are typical of grasping and fetch-and-carry tasks, namely containers and packaging of various sorts. The toolbox can handle textured as well as non-textured objects and provides shape and full 6D pose in real-time. The simplifying "blocks world" assumption was made for the case of simple shapes (cuboids and cylinders) which can be detected and tracked fully automatically. More complex shapes are handled given that a 3D wireframe model is initially available. While we do not claim to have a silver bullet to cover all problems and scenarios we believe that our simplifying assumptions pose no major limitations for a large class of interesting robotics tasks, such as fetch-and-carry, tutor-driven learning of objects or interacting with a scene composed of unknown simple objects.

Results presented are preliminary and a more detailed evaluation regarding model accuracy and tracking accuracy is needed. Also a more thorough and structured evaluation of recognition performance with regard to scene and model complexity. Several improvements are planned for the individual components. The ground plane assumption required for creating 3D wireframe models from 2D edge images can be removed by simply employing line-based stereo (in cases where the robot provides stereo cameras). The inlier ratio for the RANSAC step of the SIFT based recogniser can be improved taking into account visibility and discarding SIFTs with a surface normal pointing away from the camera. Tracking already provides a fully dynamic allocation of particles for each object and an intelligent scheduling strategy for making best use of the pool of available particles, possibly including additional attentional mechanisms is a promising topic. Regarding learning, we will replace user intervention by key press with an automatic procedure to determine good keyframes for learning SIFTs automatically.

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. Hawes and J. Wyatt, "Engineering intelligent information-processing systems with cast," *Advanced Engineering Infomatics*, vol. 24, no. 1, pp. 27–39, 2010.

[2] S. Winkelbach, S. Molkenstruck, and F. M. Wahl, "Low-Cost Laser Range Scanner and Fast Surface Registration Approach," in *Proceedings of the DAGM*, ser. LNCS, vol. 4174, 2006, pp. 718–728.

[3] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy, "Real-time 3D Model Acquisition," in *Proc. SIGGRAPH*, 2002.

[4] M. Brown, T. Drummond, and R. Cipolla, "3D Model Acquisition by Tracking 2D Wireframes," in *Proc. British Machine Vision Conference (BMVC)*, 2000.

[5] L. Vacchetti, V. Lepetit, and P. Fua, "Stable Real-Time 3D Tracking using Online and Offline Information," *PAMI*, 2004.

[6] M. Özuysal, V. Lepetit, F. Fleuret, and P. Fua, "Feature Harvesting for Tracking-by-Detection," in *European Conference on Computer Vision*, vol. 3953, 2006, pp. 592–605.

[7] M. Özuysal, P. Fua, and V. Lepetit, "Fast Keypoint Recognition in Ten Lines of Code," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[8] M. Grabner, H. Grabner, and H. Bischof, "Learning Features for Tracking / Tracking via Discriminative Online Learning of Local Features," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'07)*, 2007.

[9] P. M. Roth, M. Donoser, and H. Bischof, "On-line Learning of Unknown Hand Held Objects via Tracking," in *Proc. 2nd International Cognitive Vision Workshop (ICVS)*, Graz, Austria, 2006.

[10] H. Riemenschneider, M. Donoser, and H. Bischof, "Robust Online Object Learning and Recognition by MSER Tracking," in *Proc. 13th Computer Vision Winter Workshop (CVWW)*, 2007.

[11] P. Qi, G. Reitmayr, and T. Drummond, "ProFORMA: Probabilistic Feature-based On-line Rapid Model Acquisition," in *Proc. British Machine Vision Conference (BMVC)*, 2009.

[12] I. Gordon and D. G. Lowe, "What and where: 3D object recognition with accurate pose," in *Toward Category-Level Object Recognition*, J. Ponce, M. Hebert, Schmid. C., and A. Zisserman, Eds. Springer, 2006, ch. What and where: 3D object recognition with accurate pose, pp. 67–82.

[13] A. Collet, D. Berenson, S. S. Srinivasa, and D. Ferguson, "Object recognition and full pose registration from a single image for robotic manipulation," in *ICRA'09: Proceedings of the 2009 IEEE international conference on Robotics and Automation*. Piscataway, NJ, USA: IEEE Press, 2009, pp. 3534–3541.

[14] V. Lepetit and P. Fua, "Monocular Model-Based 3D Tracking of Rigid Objects: A Survey," *Foundationas and Trends in Computer Graphics and Vision*, vol. 1, no. 1, pp. 1–89, 2005.

[15] L. Masson, M. Dhome, and F. Jurie, "Robust Real Time Tracking of 3D Objects," in *Proc. of the 17th International Conference on Pattern Recognition (ICPR)*, vol. 4, 2004.

[16] E. Rosten and T. Drummond, "Fusing Points and Lines for High Performance Tracking." in *IEEE International Conference on Computer Vision*, vol. 2, 2005, pp. 1508–1511.

[17] G. Klein and D. Murray, "Full-3D Edge Tacking with a Particle Filter," in *Proc. British Machine Vision Conference (BMVC)*, vol. 3, Sept. 2006, pp. 1119–1128.

[18] E. Murphy-Chutorian and M. Trivedi, "Particle Filtering with Rendered Models: A Two Pass Approach to Multi-Object 3D Tracking with the GPU," in *Computer Vision on GPU Workshop (CVGPU)*, 2008, pp. 1–8.

[19] R. C. Nelson and A. Selinger, "A cubist approach to object recognition," Dept. of Computer Science, Univ. of Rochester, Tech. Rep. TR689, 1998. [Online]. Available: citeseer.nj.nec.com/article/nelson98cubist.html

[20] 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*, vol. 30, no. 1, pp. 36–51, 2008.

[21] B. Ommer and J. Malik, "Multi-Scale Object Detection by Clustering Lines," in *International Conference on Computer Vision*, 2009.

[22] S. J. Dickinson and D. Metaxas, "Integrating Qualitative and Quantitative Object Representations in the Recovery and Tracking of 3-D Shape," in *Computational and Psychophysical Mechanisms of Visual Coding*, L. Harris and M. Jenkin, Eds. Cambridge University Press, New York, 1997, pp. 221–248.

[23] P. Sala and S. Dickinson, "Model-Based Perceptual Grouping and Shape Abstraction," in *The Sixth IEEE Computer Society Workshop on Perceptual Grouping in Computer Vision (POCV 2008)*, 2008.

[24] M. Zillich and M. Vincze, "Anytimeness Avoids Parameters in Detecting Closed Convex Polygons," in *The Sixth IEEE Computer Society Workshop on Perceptual Grouping in Computer Vision (POCV 2008)*, 2008.

[25] A. Richtsfeld and M. Vincze, "Basic Object Shape Detection and Tracking using Perceptual Organization," in *International Conference on Advanced Robotics (ICAR)*, June 2009, pp. 1–6.

[26] T. Mörwald, M. Zillich, and M. Vincze, "Edge Tracking of Textured Objects with a Recursive Particle Filter," in *19th International Conference on Computer Graphics and Vision (Graphicon), Moscow*, 2009, pp. 96–103.

[27] 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 Winder Workshop (CVWW)*, 2010, pp. 91–98.

[28] A. Doucet, N. De Freitas, and N. Gordon, Eds., *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

[29] C. Wu, "http://www.cs.unc.edu/ ccwu/siftgpu/."

[30] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[31] 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.

Incremental Model Selection for Detection and Tracking of Planar Surfaces

Johann Prankl[1]
prankl@acin.tuwien.ac.at
Michael Zillich[1]
zillich@acin.tuwien.ac.at
Bastian Leibe[2]
leibe@umic.rwth-aachen.de
Markus Vincze[1]
vincze@acin.tuwien.ac.at

[1] Automation and Control Institute
Vienna University of Technology
Austria
[2] UMIC Research Centre
RWTH Aachen
Germany

Abstract

Man-made environments are abundant with planar surfaces which have attractive properties and are a prerequisite for a variety of vision tasks. This paper presents an incremental model selection method to detect piecewise planar surfaces, where planes once detected are tracked and serve as priors in subsequent images. The novelty of this approach is to formalize model selection for plane detection with Minimal Description Length (MDL) in an incremental manner. In each iteration tracked planes and new planes computed from randomly sampled interest points are evaluated, the hypotheses which best explain the scene are retained, and their supporting points are marked so that in the next iteration random sampling is guided to unexplained points. Hence, the remaining finer scene details can be represented. We show in a quantitative evaluation that this new method competes with state of the art algorithms while it is more flexible to incorporate prior knowledge from tracking.

1 Introduction

Regardless whether indoor or outdoor, man-made environments are abundant with planar structures. Walking on a planar street in a city we can observe many planar surfaces. Entering a supermarket even more objects consisting of planar structures can be found, such as box-shaped packaging located on shelves surrounded by a planar floor and walls.

Because of their attractive properties, planes are used in various vision tasks. Examples include application for camera self-calibration [25, 28] and for feature matching and grouping [7, 15]. Planes are also used for 3D reconstruction and scene analysis [1, 3, 6, 10, 18] and in robotics applications for obstacle detection [14], camera localization [19] and for object recognition [23].

Our overall goal is to build a cognitive robotic experimentation framework. The rationale behind our system is to enable human tutor driven learning-by-showing as well as completely automatic on-line model acquisition by the robot.[1] Schindler et al. [24] use a model

[1]More details about the cognitive robotic framework can be found in [20]

selection framework for multibody Structure-and-Motion estimation of image sequences. In contrast we use model selection to detect piecewise planar surfaces necessary to attach affordances such as graspable and stability. Our model is simpler but enables the robot to interact in more realistic environments. Therefore we developed an incremental model selection method, where planes once detected are tracked and serve as priors in subsequent images. This approach formalizes model selection for plane detection with Minimal Description Length (MDL), which is well-known for segmentation of range images [12] and robust object detection [11]. We describe plane hypotheses using the 2D projective transformation (homography) computed from four interest point pairs in two uncalibrated images. In an iterative scheme tracked planes and new planes computed from randomly sampled interest points are evaluated, the hypotheses which best explain the scene are retained and their supporting points marked so that in the next iteration random sampling is guided to unexplained points. Hence, the remaining finer scene details can be detected.

After a review of the related work, we give an overview of the approach in Section 2 and its core parts, namely how to select good plane hypotheses (Section 2.1) and the model selection scheme (Section 2.2). Section 3 describes the post-processing and Section 4 shows how to incorporate plane tracking. Finally, results of the experiments are shown in Section 5.

1.1 Related work

Various approaches for plane detection in uncalibrated image pairs exist. Most of them use a hypothesize-and-test framework. A popular method for detecting multiple models is to use the robust estimation method RANSAC [4], to sequentially fit the model to a data set and then to remove inliers. To generate plane hypotheses Vincent et al. [29] use groups of four points which are likely to be coplanar to compute the homography. To increase the likelihood that the points belong to the same plane they select points lying on two different lines in an image. In contrast Kanazawa et al. [9] define a probability for feature points to belong to the same plane using the Euclidean distance between the points. Both approaches use a RANSAC scheme, iteratively detect the dominant plane, remove the inliers and precede with the remaining interest points.

The success of the plane computation depends on the coplanarity of four matched points. Nicolas et al. [13] propose to use the knowledge of the first homography to guide the computation of further homographies and thus reduce the set of points/lines to three pairs. After the first iteration they use three points to generate hypotheses. The selection scheme is also iterative and RANSAC based. Piazzi et al. [22] also need only three corresponding points. They propose to use two cameras aligned to the same orientation to compute the normal vector to a plane. The normal vector is then used to cluster triangles. This approach does not use RANSAC, instead sequentially similar neighbouring normal vectors of Delaunay triangles are clustered. Lourakis et al. [16] first estimate the fundamental matrix and the epipoles. Then the homography is computed for each set of point and line feature and a voting scheme is applied. The homography with the highest number of votes is selected and refined using Least Median of Squares. This approach also detects a plane and removes the inliers in an iterative scheme.

More recent approaches concentrate on robust estimation of multiple structures. Toldo et al. [27] propose j-linkage, an approach based on random sampling and conceptual data representation. Each point is represented with the characteristic function of the set of random models that fit the point. Then agglomerative clustering is used to group points belonging to the same model. In [5] this method is used for the robust detection and matching of multiple planes. For hypothesis generation random sampling is used. To select the best hypotheses this approach clusters homographies. In [2] Chin et al. propose a novel Mercer kernel for the robust estimation problem which elicits the potential of two points to have emerged from the same underlying structure and thus can cope with more than 90% outliers. While random sampling is used to generate hypotheses, principal component analysis and spectral clustering are applied for robust fitting. The methods of Toldo et al. [27], Fouhey et al. [5] as well as Chin et al. [2] use clustering schemes and avoid to remove inliers and iterative detection of planes.

Given a fixed threshold to detect inliers incremental methods favour planes detected first over subsequent planes by greedily consuming features. Recently developed approaches overcome this drawback by treating hypotheses equally, but plane hypotheses have to be created independently of each other and thus it is not possible to restrict the search space, which leads to higher computational complexity. We propose model selection based on the MDL principle: Instead of creating all hypotheses at once, pruning models and then using model selection, we propose to embed model selection in an incremental scheme and thus guide randomized selection of interest points to compute more likely plane hypotheses. This allows us to avoid an additional hypotheses pruning step without decrease of performance. Finally, this formulation allows us to explicitly introduce priors, hence we can detect and track planes in one scheme which is not possible in any of the approaches described above.

2 Approach

We developed a method to detect multiple planes in image pairs. The idea is to embed Minimal Description Length (MDL) based model selection in an iterative scheme. Thus existing planes compete with newly created hypotheses to ensure that interest points are assigned to the best current available hypothesis. Additionally hypothesis generation can be guided to unexplained regions. This method avoids the bias towards dominant planes typical for iterative methods, and it limits the search space which leads to a faster explanation of the entire image in terms of piecewise planar surfaces.

Algorithm 1 Plane detection

```
$P \leftarrow 0$, $T \leftarrow 0$
$k \leftarrow 0$, $\varepsilon \leftarrow M/N$, $S \leftarrow 0$
while $\eta = (1-\varepsilon^M)^k \leq \eta_0$ do
    $T \leftarrow P$
    Add $Z$ random plane hypotheses to $T$
    Select plane hypotheses from $T$ and store in $P$
    Count number of explained interest points (inliers) $I$ for $P$
    if $I > S$ then
        $S \leftarrow I$
        $\varepsilon \leftarrow S/N$
    end if
    $k \leftarrow k+1$
end while
```

Algorithm 1 shows our proposed method for plane detection. In each iteration a small number Z of new plane hypotheses T is computed which have to compete with the selected hypotheses P of the last iteration. The termination criterion is based on the true inlier ratio ε and the number of samples M which are necessary to compute the homographies. As long as we do not know these values we use the best estimate available up to now. For ε that is the ratio of the number of explained interest points S of the current best plane hypotheses and the number of matched interest points N to explain. Accordingly M is the number of plane hypotheses currently selected multiplied with the minimal set of interest points $m = 4$ to compute one plane homography. Furthermore in Algorithm 1 k is the number of iterations, η stands for the probability that no correct set of hypotheses is found and η_0 is the desired failure rate. Due to the incremental scheme it is possible to guide the computation of new hypotheses to unexplained regions.

2.1 Incremental computation of good hypotheses

One of the key issues of approaches which use random samples is to select *good* features. Our method addresses this fact in two ways. In [21] Myatt et al. propose to bias random selection towards clusters in a multi-dimensional space. Following this approach the first interest point A is selected randomly. Then all other points are ordered in increasing Euclidean distance from A and further three nearby points are selected, depending on their position in the sorted list using a Gaussian distribution. The second assumption is that in the following iteration already selected homographies are good and thus the selection of the first interest point A is biased to unexplained interest points.

2.2 Minimal Description Length (MDL) based model selection

In each iteration selected homographies of the last iteration have to compete with newly sampled hypotheses. For the selection, the idea is that the same feature cannot belong to more than one plane and that the model cannot be fitted sequentially. Thus an over-complete set of homographies is generated and the best subset in terms of a Minimum Description Length criterion is chosen. The basic mathematical tool for this is introduced in [12] and adapted in [11]. We briefly describe the general formulation and explain our specific version for plane detection. To select the best model, the savings for each hypothesis H are expressed as

$$S_H = S_{data} - \kappa_1 S_{model} - \kappa_2 S_{error} \tag{1}$$

where in our case S_{data} is the number of interest points N explained by H and S_{model} stands for the cost of coding the model itself. We use one model (the homography of a plane) and thus $S_{model} = 1$. S_{error} describes the cost for the error added, which we express with the log-likelihood over all interest points f_i of the plane hypothesis H. Experiments have shown that the Gaussian error model

$$p(f_i|H) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\varepsilon_i^2}{2\sigma^2}\right) \tag{2}$$

in conjunction with an approximation of the log-likelihood comply with our expectations. Thus the cost of the error results in

$$\begin{aligned} S_{error} &= -\log \prod_{i=1}^{N} p(f_i|H) = -\sum_{i=1}^{N} \log(p(f_i|H)) && (3) \\ &= -\sum_{i=1}^{N} \sum_{n=1}^{\infty} \frac{1}{n}(1-p(f_i|H))^n \approx N - \sum_{i=1}^{N} p(f_i|H) && (4) \end{aligned}$$

where $\log(p(f_i|H))$ is the log-likelihood that an interest point belongs to the plane. For ε_i we use the Euclidean distance of inliers to the estimated homography. Substitution of Equation 4 into Equation 1 yields the merit of a model

$$S_H = -\kappa_1 + \sum_{k=1}^{N} ((1-\kappa_2) + \kappa_2 p(f_k|H)) \tag{5}$$

An interest point can only be assigned to one model. Hence, overlapping models compete for interest points which can be represented by interaction costs

$$s_{ij} = -\frac{1}{2} \sum_{f_k \in H_i \cap H_j} ((1-\kappa_2) + \kappa_2 \min\{p(f_k|H_i), p(f_k|H_j)\}) \,. \tag{6}$$

In contrast $s_{ii} = S_{H,i}$ represents a merit term of a plane hypothesis. Finding the optimal possible set of homographies for the current iteration leads to a Quadratic Boolean Problem (QBP)[2]

$$\max_n \mathbf{n}^T S\mathbf{n} \;,\; S = \begin{bmatrix} s_{11} & \cdots & s_{1N} \\ \vdots & \ddots & \vdots \\ s_{N1} & \cdots & s_{NN} \end{bmatrix} \tag{7}$$

where $\mathbf{n}= [n_1, n_2, \cdots, n_N]$ stands for the indicator vector with $n_i = 1$ if a plane hypothesis is selected and $n_i = 0$ otherwise. We embed model selection in an iterative algorithm to keep the number of hypotheses tractable. Furthermore experiments have shown that a greedy approximation gives good results and thus the solution can be found very fast.

3 Splitting of clusters

Plane hypotheses often capture interest points that match the underlying homography by chance. To account for this we introduce a postprocessing step and split them into separate planes. For this, we build a neighbourhood graph of the interest points of a plane using the Delaunay triangulation. Then the mean and the standard deviation of the distance between connected interest points is computed and edges longer than s times the standard deviation are removed. For each split plane hypothesis, we verify that the support is high enough, i.e. that their merit still surpasses κ_1. We found that for our scenarios a factor of $s = 1$ works best and leave it fixed for all following experiments. Figure 1 shows the edges of the Delaunay neighbourhood graph in white and the plane hypothesis split into two groups of interest points. The dark cluster is accepted and the weaker white group is rejected, since it does not surpass the base cost κ_1.

4 Combined detection and tracking of planes

One of the key benefits of our algorithm is that prior knowledge can be introduced easily. We exploit this in image sequences where detected planes are propagated to subsequent frames. For this, the interest points of planes detected in the previous image pair are matched with interest points of the current frame, followed by a robust homography estimation using

[2] QBP assumes pairwise interaction, which in our case can be violated. But this is still a good approximation because interaction always increases cost, yielding a desirable bias against weak hypotheses.

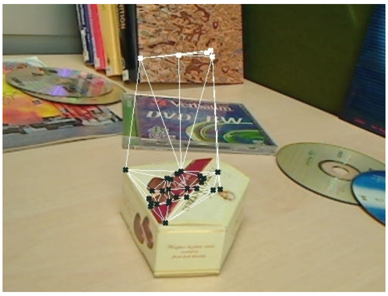


Figure 1: A plane (top surface of packaging) accidentally picks up interest points in the background. Therefore we split the interest point clusters (coloured dots) using the mean and standard deviation of edges of a Delaunay graph (white edges).

RANSAC. Thus Algorithm 1 is extended with tracked planes $P_{tracked}$, which are used to initialize P. Following $P_{tracked}$ the initialization value of the inlier ratio ε increases to the number of accumulated interest points of the tracked planes divided by the total number of matched interest points. Hence, plane detection already starts with an initial guess of planes which have to survive the following hypothesis selection stage.

5 Experiments

For all experiments we use SIFT, the well known interest point proposed by Lowe [17]. SIFT features are matched in image pairs using the Euclidean distance of the descriptors and matches are accepted if the NNDR (nearest/next ratio) d is below 0.8. To compute the homography we follow Hartley [8], that is points are normalized to zero mean and scaled to get an average distance of $\sqrt{2}$ from the origin. Then the homography is estimated using the Direct Linear Transform (DLT) algorithm.

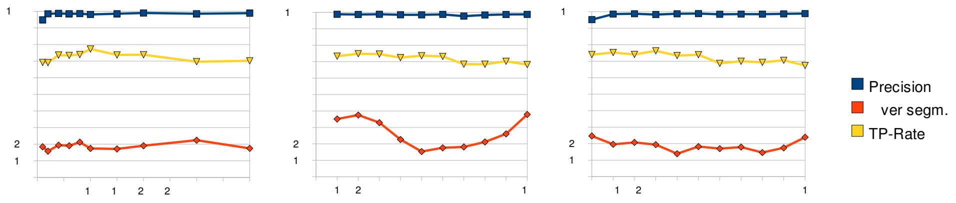
To test our method we use two completely different sets of images. Motivated by our cognitive robotic scenarios the first set of images show packaging of arbitrary shapes typically found in a supermarket (see Figures 5). We placed each object in front of a low textured background as well as in a highly cluttered scene. For comparison, we additionally test the system with the houses data sets published by the Visual Geometry Group at the University of Oxford (see Figures 6). To get ground truth we manually marked all planes in the foreground and dominant ones of the background, resulting in 231 planes in 56 images. To test the tracking of our method the packaging data set consists of 8 sequences with 4 subsequent images, whereas we used 6 sets from Oxford also with 4 images, but these images are not ordered in a sequence.

Three numbers are computed to compare the methods, that is the feature based precision

$$p_{pr} = \frac{n_{f,tp}}{n_{f,tp} + n_{f,fp}} \tag{8}$$

which is the ratio of the number of inliers $n_{f,tp}$ correctly located on a ground truth plane and the total number of features per detected plane $n_{f,tp} + n_{f,fp}$. The second number is the oversegmentation rate

$$p_{ov} = \frac{n_{p,fp}}{n_{p,tp} + n_{p,fp}} \tag{9}$$



(a) Num. of rand. hypotheses Z (b) Parameter κ_1 (c) Parameter κ_2

Figure 2: Parameter optimisation

per plane which indicates if an algorithm splits a plane in several parts. $n_{p,fp}$ the number of false positives is the number of detected planes minus the number of correct detected planes $n_{p,tp}$. Furthermore we computed the plane based true-positive rate (tp-rate)

$$p_{tp} = \frac{n_{p,tp}}{n_{p,tp} + n_{p,fn}} \tag{10}$$

which describes the ratio of the correctly detected planes $n_{p,tp}$ and the total number of planes $n_{p,tp} + n_{p,fn}$.

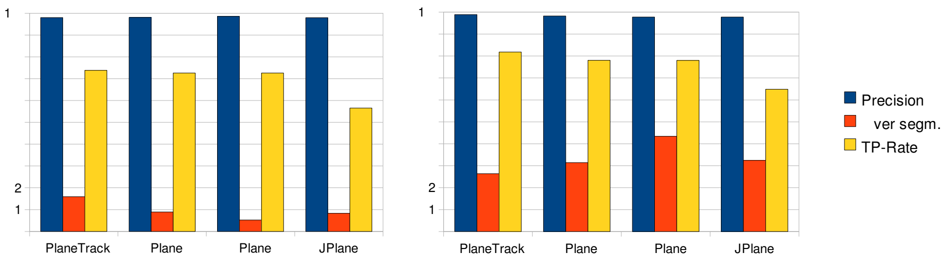
5.1 Parameter optimisation

To get an idea about the behaviour of the parameters of our algorithm we tested it with the first half of the packaging data set. For this, we vary the parameters: number of random hypotheses $Z = [1...40]$, $\kappa_1 = [1...10]$ and $\kappa_2 = [0...1.]$ and plot the performance measures. Figure 2 shows that our algorithm is quite robust against variation of the parameters. While the over-segmentation-rate in Figure 2(a) is almost constant the precision slightly increases at the beginning and the tp-rate has a peak at $Z = 10$. The Parameter κ_1 mostly influences the over-segmentation-rate, the tp-rate slightly decreases and thus we set $\kappa_1 = 5$ to the minimum of p_{ov}. In Figure 2(c) it can be seen, that the Parameter κ_2 is stable in a wide range. We set it to $\kappa_2 = 0.3$ where the tp-rate has a maximum and over-segmentation-rate is rather low.

5.2 Plane detection

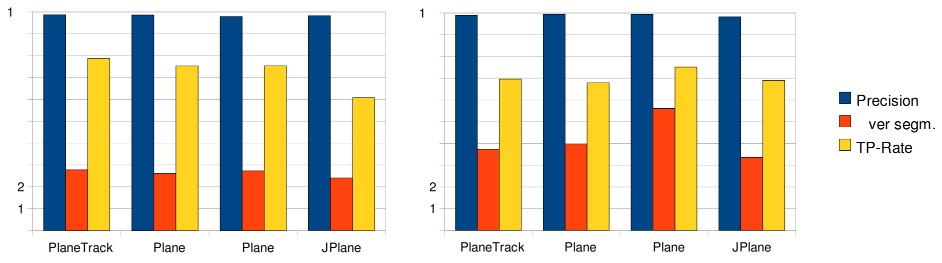
For this test all images of the packaging data set and the oxford houses data set are used. We compare the proposed method with/without tracking of planes *(SPlaneTrack/SPlane)* with iterative plane detection *(DPlane)* and j-linkage based plane detection *(JPlane, see [5])*. DPlane is a simple RANSAC based method, where in each iteration dominant planes are detected, supporting features marked and excluded in the following iteration. For all the following tests we use our own implementation of the algorithms.

The experimental evaluation shows that our model based selection method outperforms the other methods in terms of tp-rate and lower over-segmentation especially for complex scenes. Although it is not optimized for outdoor environments of the Oxford houses it competes with JPlane. Only the incremental RANSAC approach DPlane has a higher tp-rate of the cost of over segmentation. If one compares Figure 3(a) with the Figures 3(b) and 4(b) an interesting detail can be seen. Although we use the same postprocessing step for the methods (see Section 3) for the case of highly cluttered images over-segmentation increases for DPlane and JPlane, while it remains low for SPlane. Comparing Figure 3(a) and 3(b)



(a) (b)

Figure 3: Comparison of plane detection methods. Left graph shows the plane detection result for images with no background texture. Tested images of the right graph have a highly textured background.



(a) (b)

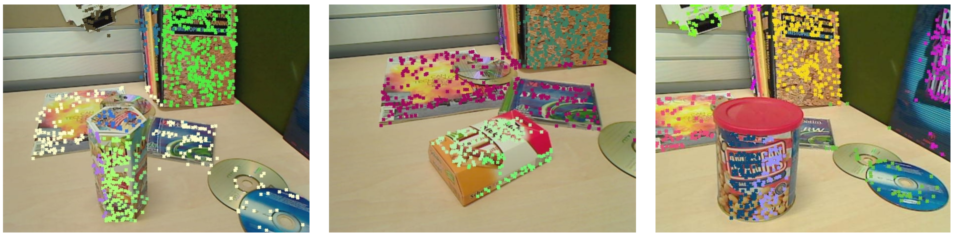
Figure 4: Results for all our images (left) and for the oxford houses data set (right).

it seems that all methods have a higher tp-rate in case of a cluttered background. For foreground objects some of the marked ground truth planes are very small and thus easily missed, while the background planes of the cluttered scenes are generally rather large and thus more easily detected, which explains the higher overall tp-rate for this scenes.

Table 1 shows the results for the packaging data set depict in Figure 4(a). Comparing SPlane and DPlane it can be seen that although SPlane converges faster and the mean number of random sample per image is lower the tp-rate is higher. A reason therefore is that incrementally filtering out of interest points which support planes detected first in DPlane leads to a decreasing inlier/outlier ratio and thus to a increasing number of samples for planes detected later. In contrast SPlane treats all planes together and thus the number of samples has an appropriate lower value. In Figure 5 planes detected in our packaging data set are depicted in different colours. Whereas in Figure 5(a) and 5(b) flat objects located on the groundplane are "correctly" detected as one plane, they are separated in Figure 5(c). Because of the foreground object, which separates the magazine and the CD's, our postprocessing described in

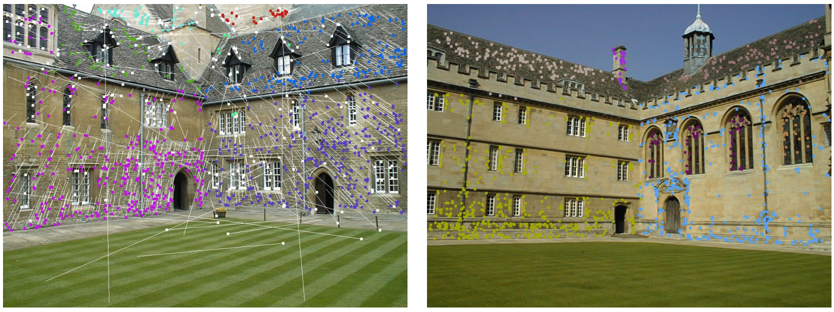
method	precision	ov. segm. rate	tp-rate	num. samples
SPlaneTrack	0.987	0.278	0.787	2455
SPlane	0.986	0.261	0.753	2407
DPlane	0.979	0.273	0.753	8278
JPlane	0.983	0.241	0.608	5000

Table 1: Comparison of different methods for plane detection (packaging data set)



(a) (b) (c)

Figure 5: Examples of our packaging data set.



(a) MertonCollege1 (b) WadhamCollege

Figure 6: Examples of the Oxford Visual Geometry data set.

Section 3 also splits this plane in two separate groups. Figure 6(a) shows MertonCollege1 from the Oxford data set. Features assigned to detected planes are drawn in different colours, whereas false matches, inaccurate features and features which belong to not accepted weak planes are depicted in white. The white lines indicate the motion of the interest points in the image pair. Another example of the Oxford data set is shown in Figure 6(b).

6 Conclusion and Further Work

We formalize model selection with Minimal Description Length to detect multiple planes in images. Planes once detected are tracked and serve as priors in subsequent images. Instead of creating all hypotheses at once an incremental scheme is proposed where tracked planes as well as planes of former iterations serve as prior to guide randomized selection of interest points to unexplained regions. Hypotheses which best explain the image are selected using an MDL criterion and retained to the next iteration. Although the method is incremental planes have to compete for features and thus features are treated equally. The result of our algorithm is planes specified by 2D homographies and sparse point clouds. For future work we want to extend the post-processing to get a dense piecewise planar object model. One possibility would be to introduce a multi-label segmentation using a Markov Random Field (MRF) optimization and graph-cuts, e.g., such as proposed by Sudipta et al. [26] and Micusik et al. [18].

Acknowledgement

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 and No. 215821, GRASP.

References

[1] C. Baillard and A. Zisserman. Automatic reconstruction of piecewise planar models from multiple views. In *Computer Vision and Pattern Recognition (CVPR 1999)*, pages II: 559–565, 1999.

[2] T.-J. Chin, H. Wang, and D. Suter. Robust fitting of multiple structures: The statistical learning approach. In *IEEE International Conference on Computer Vision (ICCV)*, 2009.

[3] Antonio Criminisi, Ian D. Reid, and Andrew Zisserman. Duality, rigidity and planar parallax. In *ECCV '98: Proceedings of the 5th European Conference on Computer Vision-Volume II*, pages 846–861, London, UK, 1998. Springer-Verlag.

[4] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[5] D. Fouhey, D. Scharstein, and A. Briggs. Multiple plane detection in image pairs using j-linkage. In *20th International Conference on Pattern Recognition (ICPR 2010)*, To appear 2010.

[6] Friedrich Fraundorfer, Konrad Schindler, and Horst Bischof. Piecewise planar scene reconstruction from sparse correspondences. *Image and Vision Computing*, 24(4):395 – 406, 2006.

[7] Luc Van Gool, Marc Proesmans, and Andrew Zisserman. Planar homologies as a basis for grouping and recognition. *Image and Vision Computing*, 16(1):21 – 26, 1998. Geometric Modeling and Invariants for Computing Vision.

[8] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521623049, 2000.

[9] Y. Kanazawa and H. Kawakami. Detection of planar regions with uncalibrated stereo using distributions of feature points. In *British Machine Vision Conference (BMVC)*, 2004.

[10] Robert Kaucic, Richard Hartley, and Nicolas Dano. Plane-based projective reconstruction. *Computer Vision, IEEE International Conference on*, 1:420, 2001.

[11] Bastian Leibe, Aleš Leonardis, and Bernt Schiele. Robust object detection with interleaved categorization and segmentation. *Int. J. Comput. Vision*, 77(1-3):259–289, 2008.

[12] Aleš Leonardis, Alok Gupta, and Ruzena Bajcsy. Segmentation of range images as the search for geometric parametric models. *Int. J. Comput. Vision*, 14(3):253–277, 1995.

[13] G. Lopez Nicolas, J.J. Guerrero, O.A. Pellejero, and C. Sagues. Computing homographies from three lines or points in an image pair. In *CIAP 2005*, pages 446–453, 2005.

[14] Manolis I. A. Lourakis and Stelios C. Orphanoudakis. Visual detection of obstacles assuming a locally planar ground. In *ACCV '98: Proceedings of the Third Asian Conference on Computer Vision-Volume II*, pages 527–534, London, UK, 1997. Springer-Verlag.

[15] M.I.A. Lourakis, S.V. Tzurbakis, A.A. Argyros, and S.C. Orphanoudakis. Using geometric constraints for matching disparate stereo views of 3d scenes containing planes. In *ICPR '00: Proceedings of the International Conference on Pattern Recognition*, page 1419, Washington, DC, USA, 2000. IEEE Computer Society.

[16] M.I.A. Lourakis, A.A. Argyros, and S.C. Orphanoudakis. Detecting planes in an uncalibrated image pair. In *British Machine Vision Conference (BMVC)*, 2002.

[17] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[18] B. Micusik and J. Kosecka. Piecewise planar city 3d modeling from street view panoramic sequences. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:2906–2912, 2009.

[19] Roger Mohr, Luce Morin, and Enrico Grosso. Relative positioning with uncalibrated cameras. In *Geometric invariance in computer vision*, pages 440–460, Cambridge, MA, USA, 1992. MIT Press.

[20] Thomas Mörwald, Johann Prankl, Andreas Richtsfeld, Michael Zillich, and Markus Vincze. Blort - the blocks world robotic vision toolbox. In *Best Practice in 3D Perception and Modeling for Mobile Manipulation at ICRA 2010*, Anchorage, Alaska, 2010.

[21] D.R. Myatt, P.H.S. Torr, S.J. Nasuto, J.M. Bishop, and R. Craddock. Napsac: High noise, high dimensional robust estimation - it's in the bag. In *British Machine Vision Conference (BMVC)*, 2002.

[22] J. Piazzi and D. Prattichizzo. Plane detection with stereo images. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference*, pages 922 –927, may 2006.

[23] C. A. Rothwell, A. Zisserman, D. A. Forsyth, and J. L. Mundy. Planar object recognition using projective shape representation. *Int. J. Comput. Vision*, 16(1):57–99, 1995.

[24] Konrad Schindler, David Suter, and Hanzi Wang. A model-selection framework for multibody structure-and-motion of image sequences. *Int. J. Comput. Vision*, 79(2): 159–177, 2008.

[25] Peter F. Sturm and Stephen J. Maybank. On plane-based camera calibration: A general algorithm, singularities, applications. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 1:1432, 1999.

[26] N. Sinha Sudipta, Drew Steedly, and Richard Szeliski. Piecewise planar stereo for image-based rendering. In *International Conference on Computer Vision (ICCV 2009)*, September 2009.

[27] Roberto Toldo and Andrea Fusiello. Robust multiple structures estimation with j-linkage. In *ECCV '08: Proceedings of the 10th European Conference on Computer Vision*, pages 537–547, Berlin, Heidelberg, 2008. Springer-Verlag.

[28] Bill Triggs. Autocalibration from planar scenes. In *ECCV '98: Proceedings of the 5th European Conference on Computer Vision-Volume I*, pages 89–105, London, UK, 1998. Springer-Verlag.

[29] E. Vincent and R. Laganiere. Detecting planar homographies in an image pair. In *Image and Signal Processing and Analysis, 2001. ISPA 2001. Proceedings of the 2nd International Symposium on*, pages 182 –187, 2001.

Multi-model Fitting Using Particle Swarm Optimization for 3D Perception in Robot Vision

Kai Zhou, Michael Zillich, Markus Vincze, Alen Vrečko and Danijel Skočaj

Abstract— **Attention operators based on 2D image cues (such as color, texture) are well known and discussed extensively in the vision literature but are not ideally suited for robotic applications. In such contexts it is the 3D structure of scene elements that makes them interesting or not. We show how a bottom-up exploration mechanism that selects spaces of interest (SOIs) based on scene elements that pop out from planes is used within a larger architecture for a cognitive system. This mechanism simplifies the object localization as single plane detection, which is however not practical when dealing with real scenes that contains objects with complicated structures (e.g. objects in a multi-layer shelf). Therefore, the key work required for this situation is the multi-plane estimation, which is solved in this paper using Particle Swarm Optimization (PSO).**

I. INTRODUCTION

Imagine a robot entering a room with the task to locate an object, say the ubiquitous coffee mug. This is quite a challenge as the mug might be partially occluded on a cluttered office desk, hidden in shadow on a shelf, too far away and only a few pixels large or simply out of the current view. This highlights the importance of attention mechanisms for robotic vision applications, as has also been argued previously e.g. in [1]. While much of the object recognition literature operates on centered objects which are large in the image (at least for training) a major problem in robotic applications is to get such nice views in the first place.

Although attention has been the subject of much interest in the psychology and vision literature, relatively little is concerned with attention based on 3D cues (see [2] for a good overview).

However, psychophysical studies show that spontaneous, exploratory eye movements are not only dependent on 2D features such as contrast and edge intensity as used in popular saliency models [3] [4] but are also influenced by the 3D structure of the visual scene, e.g. for a slanted plane follow the depth gradient [5] or fall on the 3D center of gravity of objects rather than the 2D c.o.g. of the projection [6].

Several authors have addressed the issue of attention based on 3D cues. [7] combines disparity, image flow and motion cues into an attentional operator that is designed to follow close moving targets. [8] extend the standard Koch & Ullman [9] model of visual attention based on color images with a depth channel. Similarly [10] combine two 2D saliency maps from the reflectance and range image of a 3D laser range scanner and show improved object recognition performance.

Putting an emphasis on biological plausibility the authors of [11] extend their Selective Tuning Model of attention to the binocular case to select areas and disparities of optimal matches between left and right images. Moreover their model handles issues of binocular rivalry, i.e. can put attention on a salient region in one eye when the corresponding region in the other eye is occluded.

Showing the use of attention in a robotic system [1] present a strategy for a mobile robot equipped with a stereo head to search for a target object in an unknown 3D environment that optimizes the probability of finding the target given a fixed cost limit in terms of total number of robotic actions required for detection. Their approach maintains a 3D grid of detection likelihood that is used to plan next best positions and views.

Most of the above mentioned approaches handle 3D attention by treating the disparity image like another channel next to color. [12] use the 3D reconstructed point cloud for segmenting objects from a ground plane as a preprocessing step in a robotic scenario for learning object properties. The segmentation from stereo data, which can be of unsatisfactory accuracy depending on the amount of available texture, is refined with graph-cut segmentation in the color image. However they focus on the affordance learning for manipulation and apply RANSAC (RANdom SAmple Consensus) [13] to fit a plane to the table surface. The well-recognised limitation of the RANSAC scheme, which is that it permits the estimation of single model, restricts the practical application such as dealing with real scenes that contains objects with complicated structures (e.g. multi-layer shelf, sofa and connected tea table).

We extend that approach to a wider range of scenes and develop a 3D attention operator for a robotic system aimed at various indoor tasks. Among these are object recognition and learning of objects and their properties. We make the assumption that objects presented to the robot for learning as well as objects the robot is asked to pick up are resting on supporting surfaces such as tables, shelves or simply the floor. Accordingly we place attention on anything that sticks out from supporting surfaces.

Our major contribution is a novel multi-model estimation approach based on particle swarm optimisation. The resulted multiple supporting planes serve in our robot system as a fundamental component for bottom-to-up exploration. In Section II the related work of model-estimation is outlined. The subsequent sections describe the system overview, the

Kai Zhou, Michael Zillich and Markus Vincze are with Institute of Automation and Control, Vienna University of Technology, Austria. `[zhou, zillich, vincze]@acin.tuwien.ac.at`

Alen Vrečko and Danijel Skočaj are with the Visual Cognitive Systems Laboratory, University of Ljubljana, Slovenia. `[alen.vrecko, danijel.skocaj]@fri.uni-lj.si`

employed methods in detail and present the results.

II. RELATED WORK

In model estimation from a set of observed data with outliers, the RANSAC approaches based on randomly selected hypotheses of models prove efficient. Each hypothesis is evaluated by means of a cost function $f(.)$. The hypothesis that minimize $f(.)$, i.e. $h = \mathbf{arg\,min}\, f(.)$ is considered as the solution of the estimation. As the candidate hypotheses are discrete, the RANSAC algorithms offer only approximate results. Meanwhile, in order to ensure that all the inlier samples are covered by the hypotheses, a massive number of hypotheses must be generated in most RANSAC-based algorithms. The inlier ratio is required to be defined before the hypotheses generation, either by the user [13][14] or by pre-calculation [15][16]. The RANSAC approach is originally designed for single model estimation and a few literatures report some successful extensions to multi-model fitting [17][18].

Since the hypotheses used by RANSAC are usually described with parameters (e.g. a line $r(\theta) = r_0 \sec(\theta - \varphi)$ in polar coordinate system is represented with two parameters $[r, \theta]$), the target models can be alternatively estimated in parameter space[19] [20] [21]. The parameter-space analysis approaches cope naturally with multi-model estimation. Diverse statistic analysis methods such as skewness and kurtosis detection are deployed in the parameter space of the model. In [19], Xu et al. build a histogram and detect the peaks of the histogram in parameter space to get the best solution; Subbarao and Meer [20] utilize mean-shift to cluster the manifold structures, which in essence applies the Gauss-Newton algorithm to find the minimum of an elaborate cost function in parameter space.

Recent developments[14][15][16] in robust multi-model estimation lie in a large part in automatic process ending. Instead of evaluate each hypothesis individually, the candidates are checked by means of their distribution and the relationships between one hypothesis and the other ones are also considered (such as J-linkage in [14] and Mercer kernel in [16]). However, the search space of solutions is still limited to all the pre-selected hypotheses as for the RANSAC family.

Recently in [21], Delong et al. proposed an energy minimization based algorithm, which is widely used for labelling problems in computer vision such as image segmentation and stereo matching. By controlling the label number in the solution, this method proves to be effect for multi-model fitting.

All the above mentioned methods deliver discrete solutions and therefore are sensitive to uneven distributed data sets, i.e. once the distribution of inliers is uneven, some models are dropped because of the manifold regularization.

We propose a novel method to estimate multiple models in parameter space. The hypothetical models are mapped to parameter space and represented as particles, which can move continuously. Particle swarm optimization (PSO) is employed to search for the optimal position in the parameter space, where the corresponding model minimizes the cost function. Since the PSO is originally designed to simulate social behaviours aimed at a global optimization, information sharing and mutual cooperation among particles(solutions) are incorporated as in [14][15][16]. Therefore, the model fitting can be automatically ended up when all the optimal models are found out. The other advantage of our algorithm is that it delivers the estimation results model by model suited for real-time performance in mobile robot system, differing from the global analysis algorithms in which all the estimated models are output as a whole solution after the entire estimation process[14] [15] [16].

III. SYSTEM OVERVIEW

Figure 1 shows (part of) the visual processing happening within a larger robotics framework. The framework is based on a software architecture toolkit [22] which handles issues like threading, lower level drivers and communication via shared working memories.

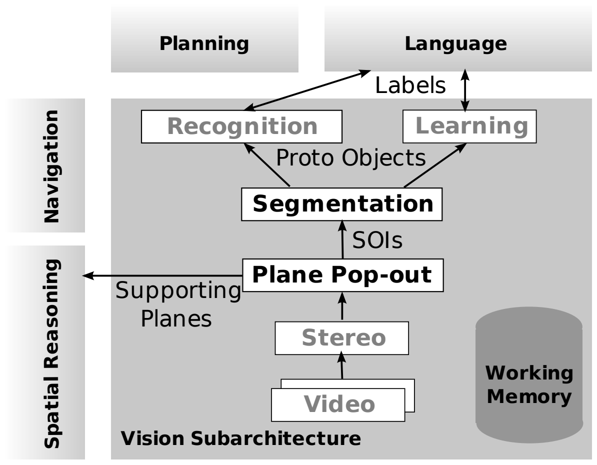


Fig. 1. System overview: Attention driven visual processing in a cognitive robotics architecture

Attention in this context serves several purposes. First of all there is the obvious usage as a primer for costly object recognition. 3D point cloud from stereo reconstruction is used to extract dominant planes as well as things sticking out from these planes, a process which we refer to as *plane pop-out*. Those sticking-out parts form spherical 3D spaces of interest (SOIs) (termed so to avoid confusion with typical 2D image regions of interest - ROIs) which are handed to the segmentation component. The segmenter is coarsely initialised with colours obtained from back-projected 3D points inside the SOIs and then it refines the projected contour of the SOIs generating what we term proto-objects. These form an intermediate level more object-like (i.e. more likely to correspond to an actual scene object) than just “stuff” that caught our attention but not quite recognised (labelled) objects yet. Proto-objects are subsequently handed to the (SIFT-based) recogniser which only needs to process the segmented image regions.

IV. THE PROPOSED APPROACH

There are of course potentially many planes in an indoor environment but we are only interested in supporting, i.e. horizontal planes. We either know the tilt angle of the camera from the mounting on the mobile platform and thus know the horizontal direction or, in case we don't have a calibrated system, we find it based on the assumption that the ground plane is the dominant plane in the first view when entering a new room (initialisation phase).

A. MULTI-MODEL FITTING

Here we describes how the plane estimation problem can be effectively solved with particle swarm optimization. Given input data (point cloud from stereo) $\mathcal{L} = \{x_i\}_{i=1,\dots,N}$ and plane hypotheses $\mathcal{H} = \{\theta_j\}_{j=1,\dots,M}$ which are generated by individuals randomly sampled from $\mathcal{L}$, the plane should be represented with D parameters which means the optimal plane can be estimated with searching a D dimensional space, also each plane can be considered as a particle in the D dimensional space. A cost function $f(\theta)$ is defined to evaluate the current position of model θ in the D dimensional parameter space. The problem of plane model estimation is to find $\mathbf{arg\,min}\{f(\theta), \theta \in \mathbb{R}^D\}$.

Particle Swarm Optimization (PSO) is a metaheuristic optimization method which was first intended for simulating social behaviour of birds[23]. PSO maintains a swarm of candidate solutions(called particles), and makes these particles fly around in the search-space according to their own personal best found position and the swarm's best known position. To rate each particle, PSO introduces a fitness function which has a similar functionality as the energy function used in [21]. Each particle has a velocity that directs its movement and is adjusted iteratively. After several iterations, particles are converged at one position where the evaluation function has the minimum value.

$$\begin{aligned} v_{jd}^{t+1} &= \chi(w v_{jd}^t + c_1 r_1(p_{jd}^t - \theta_{jd}^t) + c_2 r_2(p_{gd}^t - \theta_{jd}^t)) \\ \theta_{jd}^{t+1} &= \theta_{jd}^t + v_{jd}^{t+1} \end{aligned} \tag{1}$$

where $v_j = (v_{j1}, v_{j1}, \dots, v_{jD})$ denotes the velocity of particle j in d dimension ($1 \leq d \leq D$), $p_j = (p_{j1}, p_{j1}, \dots, p_{jD})$ is the j^{th} particle's own best position found so far, and p_{gd} is the global best current position of the swarm. For all the particle in d dimensions, v^{t+1}, θ^{t+1} are updated velocity and new position derived from this new velocity. As in [24], χ is the "constriction factor"which is introduced to limit the maximum velocity and the dynamic search area on each dimension. w, c_1, c_2 are inertia weight and two acceleration constants, where c_1 is the factor that influences the "cognitive"behaviour, i.e., how much the particle will follow its own best solution, and c_2 is the factor for "social"behaviour, i.e., how much the particle will follow the swarm's best solution. c_1 and c_2 are typically set to 2.1 and 2.0, χ is a function of c_1 and c_2 as reflected in (2). r_1 and r_2 are even distributed random numbers in the interval $[0, 1]$. The iteration continues till a termination criterion (e.g. number of iterations performed, or adequate fitness reached) is met, then the p_{gd} holds the best found solution. The termination criterion as well as the fitness function used in our approach are addressed later. Algorithm 1 lists the scheme of PSO.

$$\begin{aligned} \chi &= \frac{2}{|2 - c - \sqrt{c^2 - 4c}|} \\ c &= c_1 + c_2 \end{aligned} \tag{2}$$

Algorithm 1 PSO scheme

```
1: ∀$\theta_j$ Initialise $v_j, p_j$ randomly
2: while maximum iteration or minimum error criteria is
   not attained do
3:     ∀$\theta_j$, evaluate each particle using fitness function $f(\theta_j)$
4:     if $f(\theta_j) < f(p_j)$ then
5:         Replace $p_i$ with $\theta_j$
6:     end if
7:     if $f(\theta_j) < f(p_g)$ then
8:         Replace $p_g$ with $\theta_j$
9:     end if
10:    ∀$\theta_j$, calculate new $v_j$ and update $\theta_j$ according to (1)
11: end while
```

We don't simply use the current best position of the swarm but select one particle from a global best list using roulette wheel selection. This roulette wheel selection mechanism [25] is introduced to promote the diversification of global (assuming) optima and avoid the local optima caused by premature convergence. To implement the roulette wheel selection, a list of particles with the n best fitness values is first stored. These particles are rated using their fitness and the fittest individuals have the largest share of the roulette wheel. If f_k is the fitness of particle k, its probability of being selected as the global best is as follows:

$$P_{f=f_k} = \frac{f_k}{\sum_{m=1}^{n} f_m} \tag{3}$$

Since this roulette wheel selection mechanism is called whenever the particle needs to update its velocity, the cells on the roulette (equivalent with the length of the global best list) should be reasonably limited. According to how many models should be estimated, we address the fitness function and termination criterion respectively.

Easy case: one model estimation: We use the general cost function the same with the RANSAC family as the fitness function in PSO. It can be formulated as follows:

$$f(\theta^t) = \sum_{x_i \subset \mathcal{L}} \mathrm{Loss}(dist_{x_i \theta^t}) \tag{4}$$

where i denotes the index of each input data and θ is the model hypothesis. $dist$ is an error function such as geometric distance between individual data and model hypothesis. The loss function is a truncation function which can be illustrated as follows:

$$\text{Loss}(dist) = \begin{cases} dist & \text{if } |dist| < const \\ const & \text{otherwise} \end{cases} \tag{5}$$

We use a time-based measure to detect the convergence (termination criterion) as in [26]. The swarm is said to have converged if the global best location has not moved for s iterations, this measure is invariant to the scale of the problem space.

Fitness Function "Deflation": When more than one model exists, the fitness function has more than one optimum, therefore the swarm probably plunges into one of the following behaviours: either it can find only one optimum, or it will wander and reciprocate among optima. We propose a "deflation" function $g(\theta)$ to perform a transformation of the fitness function $f(\theta)$ once there is any (possibly local) optimum being found, this transformation lessens the fitness of the optima found as well as the around area of these optimums, it seems as the nearby area of the optima found converges at this position and the particles in this area has the lower probability to be selected as the global best position.

$$g(\theta) = f(\theta) \prod_{k \subset \mathcal{O}} (\lambda \mathrm{e}^{-\lambda D_{\theta\bar{\theta}_k}} + 1) \tag{6}$$

where $\bar{\theta}$ is one optima found and $D_{\theta\bar{\theta_k}}$ denotes the distance between current particle and the k^{th} optimum found. The set $\mathcal{O}$ contains all the optima found. λ is a parameter which is inversely proportional to the size of the nearby convergence area of the optima found.

Algorithm 2 PSO scheme for multiple model fitting

```
1: $\forall\theta_j$ Initialise $v_j, p_j$ randomly
2: while Completion criteria is not attained do
3:     if sizeof($\mathcal{O}$) == 0 then
4:         $\forall\theta_j$, $E(\theta_j) = f(\theta_j)$
5:     else
6:         $\forall\theta_j$, $E(\theta_j) = g(\theta_j)$
7:     end if
8:     if $E(\theta_j) < f(p_j)$ then
9:         Replace $p_i$ with $\theta_j$
10:    end if
11:    if $E(\theta_j) < f(p_g)$ then
12:        Replace $p_g$ with $\theta_j$
13:    end if
14:    $\forall\theta_j$, calculate new $v_j$ and update $\theta_j$ according to (1)
15:    if Convergence is detected then
16:        $\forall\theta_j$ Reinitialise $v_j, p_j$
17:        $\mathcal{O}$ →pushback($p_g$)
18:    end if
19: end while
```

Each convergence can be detected in the same way as in the one-model estimation problem described before. The completion criteria, which means all the optima are found, can be derived by the analysing the stability of the global best list used in roulette wheel selection. Once all the optima are found, the fitness of new hypothetical particles counterbalances with each other, the stability of global best list reflects this phenomenon.

B. OBJECT SEGMENTATION

Plane fitting is called iteratively until no more horizontal planes with reasonable size can be found. Then the remaining points sticking out from these planes are segmented using 3D flood-filling and the resulting clusters together with a bounding sphere form SOIs. Note that the bounding sphere is slightly larger than the actual point cluster to ensure it also contains a part of the plane points, which is needed for the following segmentation step. Figure 4 shows the different scenes and corresponding reconstructed point cloud. Different planes are shown in different colours and remaining sticking out points are shown in yellow. Because of the inherent limitation of stereo reconstruction at poorly textured surface parts and shadowing effects between left and right camera, we refine the results using 2D colour based segmentation.

The 2D segmentation is based on *energy minimisation with graph cuts*. The back-projected 3D points within the SOI provide colour and spatial cues for the object and its background. The cost function for the object combines the *colour cost* with the *spatial cost*, while the cost function for the background consists of the *colour cost* component only. The *spatial cost* is simply the distance between the point and the nearest object's back-projected 3D point. The *colour cost*, on the other hand, is the average distance between the point's colour and the K nearest colours from the sample (K is determined based on the sample size). Besides foreground and background cost functions, there is a third cost function with a fixed cost to cover the areas, where both former functions have high costs. While these areas are considered uncertain and might be resolved on higher levels of the system's cognition, they are meanwhile deemed as background by the recogniser.

The distance between two colours is calculated in the HLS colour space:

$$\Delta HLS = \Delta^2 S + (1 - \Delta S)\Delta HL \tag{7}$$

$$\Delta HL = \bar{S}\Delta H + (1 - \bar{S})\Delta L, \tag{8}$$

where ΔH, ΔL and ΔS are the distances between the two colour's HLS components, while $\bar{S}$ is the average saturation of the two colours. All the parameters are normalised to values between 0 and 1. The H distance has to be additionally normalised and truncated because of its circular space. The contribution of each colour component to the overall distance between the two colours is thus determined by the saturation difference and saturation average.

The code for the graph cut algorithm was kindly provided by Boykov, Kolmogorov and Veksler [27][28][29].

V. EXPERIMENTAL RESULTS

To demonstrate the progress of convergence, first we tested our PSO based multi-model estimation algorithm on synthetic 2D data. In figure 2, this progress is illustrated in

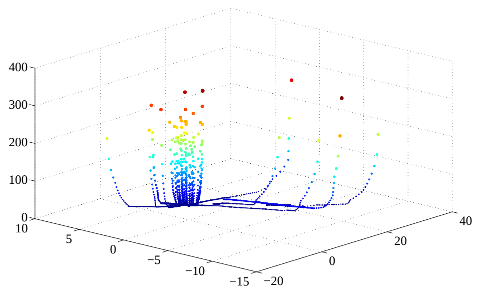


Fig. 2. Convergence of all the particles. The horizontal plane is the 2 dimensional parameter space of line fitting problem. The vertical axis is the value of fitness function, the smaller value denotes the better fitness.

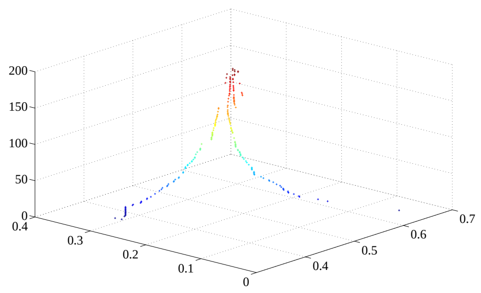


Fig. 3. Movement of global best, the horizontal plane is the 2 dimensional parameter space of line fitting problem and vertical axis is the iteration times of PSO

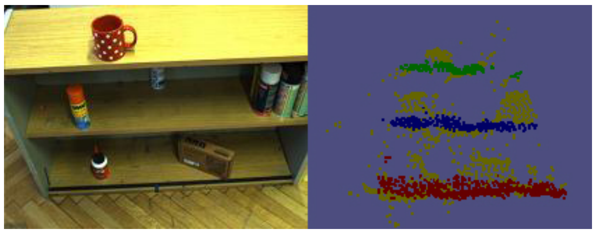
parameter space. Figure 3 shows that after 100 iterations, the global best location is almost stable and static, this is the criteria for convergence detection in our experiments. The results on synthetic data are reported in figure 6 and 7. As shown in figure 4, our system are tested on the different real scene. The multiple layers of shelf are estimated in figure 4(a), the chair surface and floor are detected in figure 4(b).

Figure 5 shows a typical result for a shelf consisting of three planes. The results of the subsequent segmentation step are shown in the right part of figure 5. On the left side, the SOI are marked on the original image with a bounding rectangle, the following 2D graph-cut segmentation would only process in these rectangles. The right side zooms on these area, the top images show the position of back-projected 3D points (green for object, red for background) and the segmentation (grey for object, white for background), the bottom images represent the graph cut cost functions for object and background where the brighter colour denotes greater cost. We can see that despite the fact that the backprojected 3D points are not very precise due to rather large noise, the graph-cut segmentation can be successfully initialised and provides a precise object contour.

VI. CONCLUSION

We present an attentional mechanism based on plane pop-out in 3D stereo data and its use within a robotic framework. The robust and simultaneous detection of multiple supporting planes is the major contribution of this paper. To our knowledge, Particle Swarm Optimisation (PSO) is the first used as the workhorse to solve the multi-model estimation problem. As PSO is a variant of metaheuristics, theoretically other metaheuristic methods could also be applied in multi-model fitting. Our successful application of PSO points out the new direction of future studies.

Future work will on one hand focus on more robust extraction of supporting planes in cases where only small textured parts of the plane are visible as in the case of (densely filled) shelves. To this end we plan to fuse cues from line-based stereo with dense stereo. On the other hand we are currently integrating the recently proposed segmentation with fixation method by [30] as an alternative to the more generic graph-cut segmentation used now.


(a) multi-layer shelf scene

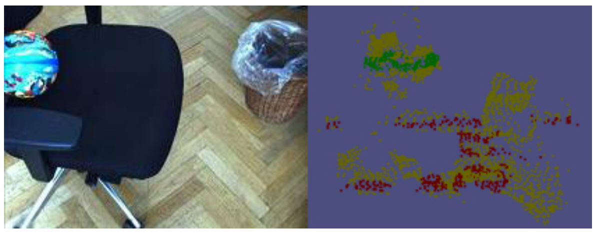

(b) chair and floor scene

Fig. 4. PSO based multi-plane estimation is tested with two different scenes, the detected planes are illustrated in the right images, note that the figure is best viewed in color.

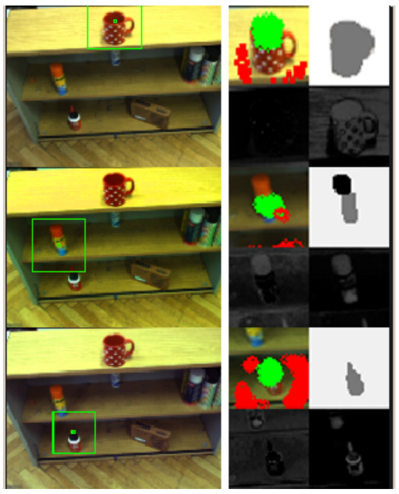


Fig. 5. Segmentation of back-projected spaces of interest

VII. ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Communitys Seventh Framework

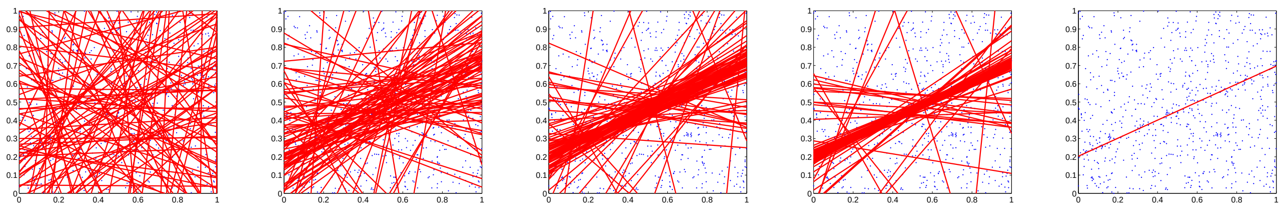


Fig. 6. The demonstration of 2D line fitting using PSO. The procedure of PSO iteration is illustrated: the initial 100 particles are showed in the first figure, the next three figures are captured after 50, 100 and 150 iterations, the last figure is the final estimation result. Input data of 800 points, with 100 liners and 700 gross outliers.

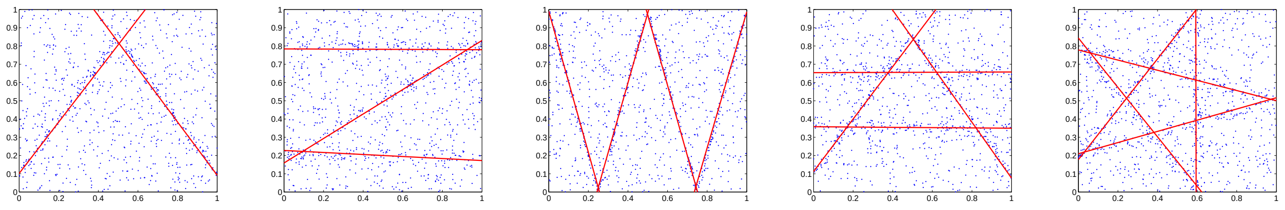


Fig. 7. The multiple line fitting results are depicted, either with the models that have spatial overlap (last two) or just occlude each other. Input data of each test is with 100 points per line and 700 gross outliers.

Programme [FP7/2007-2013] under grant agreement No. 215181, CogX.

REFERENCES

[1] J. K. Tsotsos and K. Shubina, "Attention and Visual Search : Active Robotic Vision Systems that Search," in *The 5th International Conference on Computer Vision Systems*, 2007.

[2] S. Frintrop, E. Rome, and H. Christensen, "Computational Visual Attention Systems and their Cognitive Foundations: A Survey," *ACM Transactions on Applied Perception*, 2009.

[3] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Transactions on PAMI*, vol. 20, no. 11, pp. 1254–1259, Nov 1998.

[4] J. Tsotsos, S. Culhane, W. Wai, Y. Lai, N. Davis, and F. Nuflo, "Modeling visual attention via selective tuning," *Artificial Intelligence*, vol. 78, no. 1–2, pp. 507–547, 1995.

[5] M. Wexler and N. Quarti, "Depth Affects Where We Look," *Current Biology*, vol. 18, pp. 1872–1876, Dec. 2008.

[6] D. Vishwanath and E. Kowler, "Saccadic localization in the presence of cues to three-dimensional shape," *J. Vis.*, vol. 4, no. 6, pp. 445–458, May 2004.

[7] A. Maki, P. Nordlund, and J.-O. Eklundh, "A computational model of depth-based attention," in *Proceedings of the 13th ICPR*, vol. 4, Aug 1996, pp. 734–739.

[8] N. Ouerhani and H. Hugli, "Computing visual attention from scene depth," in *ICPR 2000*, vol. 1, 2000, pp. 375–378.

[9] C. Koch and S. Ullman, "Shifts in selective visual attention: towards the underlying neural circuitry," *Human Neurobiology*, pp. 219–227, 1985.

[10] S. Frintrop, E. Rome, A. Nuchter, and H. Surmann, "A bimodal laser-based attention system," *J. of Computer Vision and Image Understanding (CVIU), Special Issue on Attention and Performance in Computer Vision*, vol. 100, no. 1-2, pp. 124–151, 2005.

[11] N. D. B. Bruce and J. K. Tsotsos, "An attentional framework for stereo vision," in *In Proc. of Canadian Conference on Computer and Robot Vision*, 2005.

[12] B. Ridge, D. Skočaj, and A. Leonardis, "Unsupervised learning of basic object affordances from object properties," in *Proceedings of the Fourteenth Computer Vision Winter Workshop*, 2009, pp. 21–28.

[13] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[14] R. Toldo and A. Fusiello, "Robust multiple structures estimation with j-linkage," in *Proceedings of the European Conference of Computer Vision*, 2008, pp. 537–547.

[15] W. Zhang and J. Kosecka, "Nonparametric estimation of multiple structures with outliers," in *Dynamical Vision Workshop*, 2006.

[16] T. Chin, H. Wang, and D. Suter, "Robust fitting of multiple structures: The statistical learning approach," in *ICCV*, 2009.

[17] M. Zuliani, C. S. Kenney, and B. S. Manjunath, "The multiransac algorithm and its application to detect planar homographies," in *IEEE International Conference on Image Processing*, 2005.

[18] C. V. Stewart, "Bias in robust estimation caused by discontinuities and multiple structures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, pp. 818–833, 1997.

[19] L. Xu, E. Oja, and P. Kultanen, "A new curve detection method: randomized hough transform (rht)," *Pattern Recogn. Lett.*, vol. 11, no. 5, pp. 331–338, 1990.

[20] R. Subbarao and P. Meer, "Nonlinear mean shift for clustering over analytic manifolds," in *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, pp. 1168–1175.

[21] A. Delong, A. Osokin, H. N. Isack, and Y. Boykov, "Fast approximate energy minimization with label costs," in *CVPR*, 2010.

[22] N. Hawes and J. Wyatt, "Engineering intelligent information-processing systems with cast," *Advanced Engineering Infomatics*, vol. 24, no. 1, pp. 27–39, 2010.

[23] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks*, 1995, pp. 1942–1948.

[24] R. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proceedings of the 2000 IEEE Congress on Evolutionary Computation*, 2000, pp. 84–88 vol.1.

[25] T. Bäck, *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms.* Oxford, UK: Oxford University Press, 1996.

[26] S. Bird and X. Li, "Enhancing the robustness of a speciation-based pso," in *Proceedings of the 2006 IEEE Congress on Evolutionary Computation*, 2006.

[27] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Transactions on PAMI*, vol. 26, no. 9, pp. 1124 – 1137, 2004.

[28] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," in *ICCV (1)*, 1999, pp. 377–384. [Online]. Available: citeseer.ist.psu.edu/article/boykov99fast.html

[29] V. Kolmogorov and R. Zabih, "What energy functions can be minimized via graph cuts?" *IEEE Transactions on PAMI*, vol. 26, no. 2, pp. 147 – 159, 2004.

[30] A. Mishra, Y. Aloimonos, and C. L. Fah, "Active Segmentation with Fixation," in *ICCV*, 2009.

KTH Numerical Analysis
and Computer Science

Department of Numerical Analysis and Computer Science

Object Search with Mobile Robots

Alper Aydemir

TRITA-NA-yynn

Alper Aydemir
Object Search with Mobile Robots

Report number: TRITA-NA-yynn
Publication date: June 2010
E-mail of author: aydemir@kth.se

Object Search with Mobile Robots

Alper Aydemir
KTH – Numerical Analysis and Computer Science
SE-100 44 Stockholm
SWEDEN

July 29, 2010

Abstract

This report presents methods that enables a mobile robot efficiently search an environment for an object.

1 Introduction

Object search entails ascertaining the location of a specific object, or of one or more of a given class of objects, and doing so in an efficient manner. Object search is an important component of a mobile robots skill repertoire. The need for utilizing objects in the robotics framework becomes more and more crucial if robots will one day coexists with humans. Previous work [19] has shown that the human spatial understanding is based heavily on the objects that are in the environment. The same objects can also be used for various fields such as navigation and localization. Objects not only help defining the space around them but also accomodate certain functional interpretations. This is why different modes of interaction such as grasping and manipulation attracts much attention in the research community.

Previous work is based on the assumption that the object in question is assumed to be already in the field of view of the robot or in its immediate sensory reach. An often stated reason for this is tasks such as object recognition and manipulation poses already hard enough challenges. However as we crack these challenges, the assumption of the object being readily available in the field of view of robot's sensors can no longer be hold. It is simply unrealistic and too nave to assume that all the object that a mobile robot would need to recognize and manipulate will always inside the immediate sensory reach of the robot. This report is an attempt to document the sparse work and state future directions.

The goal of object search, as defined in this report, is to calculate the set of sensing actions which brings the target object in the sensor field of view in an as efficient manner as possible. For this set to represent an efficient search plan, it should consist of minimum number of sensing actions with the highest object detection probability. To construct and execute such a plan, the searcher must actively adjust its sensor parameters to obtain the highest quality data. This approach is also known as active visual search [2]. Considering the case of searching for a 3D object in 3D space the solution to active visual search problem is far from trivial. A uninformed search, i.e. without any prior information on the target object's location, inevitably suffers from the curse of dimensionality, proportional with the sensor parameters that can be actively controlled. Furthermore, [23] has shown that the solution to object search problem is NP-complete. Therefore the active visual search problem needs to be addressed accordingly. In particular, the sensor planning problem for object search also often adressed as the next-best view problem [14, 6, 3]:

How and where to choose the position of the next sensor to be deployed in order to maximize the total amount of information each time?

This report spans works that involve the case of mobile robot equipped with a sensor capable of verifying the presence of an object. Typically the sensor is a video camera which has limited effective range and field of view. Some authors also assumes an infinite ranged sensor with a line-of-sight sensing capability. We assume that the robot can move in the free space of the environment according a certain spatial representation. The obvious solution for finding objects is the exhaustive blind search over all possible sensing actions that the environment and the searcher can afford. However, as stated above, such an approach is intractable even considering the simplest of sensors. The dimensionality of the search space and the task of looking for an object with an arbitrary pose in the 3D space makes this approach impractical.

2 Background

The methods presented in previous work attempting to solve the object search problem are diverse in their solutions.The diversity in methods stems mainly from different spatial representations used (2D-3D metric representation of space, graph representation etc.), various assumptions prior knowledge on object location and the implementation on various robotic systems with different capabilities (RFID sensing, line-of-sight etc.) . Therefore, usually it is not possible to directly compare bulk of the previous work in a common quantitative benchmark setting. However there it is possible to group most of the work under the following categories:

- Homogenous approaches
  - Purely geometric
  - Purely probabilistic (Bayesian)
- Hybrid approaches

Purely geometric methods are concerned with the full or near-full coverage of the search space. The methods used in this category usually has a fine detailed metric map of the environment. The task of the searcher is to leave as little unchecked volume as possible. On the other hand purely probabilistic approaches deals with the recognition and presence probability of an object in a given node. The reason that they are purely probabilistic is because the searcher movements are modeled in a graph like spatial representation. In this approaches it is generally assumed that if the robot and the object are in the same graph node, the object is detected. Therefore interfering effects such as occlusion and other sensor model shortcomings are ignored. Hybrid approaches tries to merge both of the above mentioned methods. The environment configuration is not ignored and a probability of object being at a location is assigned to the pieces of the environment. These approaches differ among themselves in how much geometry or bayesian inference is considered.

Most previous work from the computer vision field that mentions object search tries to locate the contour of the object in the image domain. The object is assumed to be present in the image. Object search in this context is different from the case where a mobile robot is being tasked with finding an object in an office or home environment. In this setting assuming that the set of objects that the robot can recognize are already in the field of view is clearly unrealistic. Therefore previous works that searches for objects in the image domain [16, 20] are not in the scope of this report. Furthermore, the pursuit/evasion literature considering various scenarios such as an UAV locating a ground object or an evader actively seeking to avoid a searcher is out of the scope of this text although presented methods may apply.

3 Homogeneous approaches

3.1 Purely geometric

Let Ω representing the environment configuration and f a sensing action. The exact parameters determining f can be the searchers position and sensor specific parameters. Assume that the sensor that is being used to detect an object has a known field of view and range, that is the coverage of a single sensing action is known. Now let $U_n = f_0..f_n$ a set of sensing actions, $C(U_n)$ the subset of Ω that is covered by U_n. The aim of purely geometric approaches is to find the solution to the following:

$$\underset{n}{\operatorname{argmax}}\ A(U_n) \tag{1}$$

Although the problem formulation seems simple, considering the space of possible Ω configurations and sensor models, the solution is far from trivial. A traditionally well researched problem in computational geometry that concerns with coverage is the so called art gallery problem. The art gallery problem poses the following question: -What is the minimum number of non-moving guards and their positions to entirely monitor an art gallery?

Although art gallery theorems do not explicitly try to solve the problem of finding objects, coverage aspect of the object search makes art gallery problems a base solution in cases where there is lack of a priori knowledge on the object location. Imagine looking for an object that can be anywhere inside a room. Then one's best shot to making sure that the room contains the object is to look everywhere inside the room. This connection makes art gallery problem a base line case of object search.

The majority of existing work on art gallery problem are grouped into art gallery theorems that deals with covering 2D polygon shapes. There are many other different variations of the art gallery problem such as, the watchmen's route problem where non-moving guards leave their place to moving guards. Looking at an art gallery from above, one can abstract it to a polygon. A guard is defined as a point inside the polygon with a field of view similar to a sensor pose. Şafak [7] in his 2009 master's thesis provides an extensive introduction to art gallery theorems and discusses their usage in the object search context, this thesis is added as an appendix.

Another approach which avoids on relying a complex world model is proposed by Tovar et al. [12]. The method introduces a reduced visibility graph T_g by tracking the appearance and disapperance of gaps using a range sensor. The sensor is assumed to be able to track any topological change related to gaps, e.g. appearance, disappearance, merging or splitting of gaps. Everytime a critical event in sensor measurements is detected a new topological node is added to T_g. T_g is used to generate optimal paths connecting nodes. Assuming that the world is 2-D, objects create the same type of gap event that causes the robot to add a new node. Therefore, from robots spatial perspective there is no difference between a navigation node and an object node in T_g. Typically the gap which is closest to the object is assumed to be the place of the object. It is assumed that by travelling to the nearest gap to the object, the robot can reach objects location. One obvious limitation of this approach is that often object supporting planar surfaces such as tables and shelves do not appear as discontinuities in the 2-D sensor output.

Sarmiento et al. [11] presents a method to calculate a minimal trajectory to locate an object in a known 3D environment W .G is a hidden guard set meaning that members of G are not visible to each other. First, the algorithm samples the environment using the method in [13]. The goal of sampling is to approximate W and finally represent W as a set of convex regions. The sampling is done by randomly sampling the interior of W . Initially G the hidden guard set of W is empty. For each sample, the algorithm checks whether the sample point is visible from at least one of the guards in G. If not, the sample is added in G as a new guard otherwise it is added to the sample set S. An initial set of sensing locations G together with a probabilistic measure of how good is the coverage is the output of sampling process. Then taking the initial samples as basis, the environment is divided into convex regions. The resulting convex set is then used to produce a graph representing connectivity of the space which will be used to compute a new set of sensing locations. A greedy search is performed on this graph to find the trajectory which minimizes the time to find an object. No prior information about the object is considered during the search.

3.2 Purely probabilistic approaches

Purely probabilistic approaches abstracts the environment in a way that effects resulting from the metric properties of the environment such as occlusion, interference etc. are ignored. Typically the well researched pursuit/evasion problem operates on such a framework. The case of a mobile robot looking for an search is a specialization of pursuit/evasion in that the target does not actively try to avoid the searcher. Therefore most methods from pursuit/evasion can be applied to the object search problem in the abstract sense. However ignoring geometrical constraints often results in less than desirable performance in real world experiments.[9] formulated in such a probabilistic framework but tries to tackle the issue of finding an object with a mobile robot in a real world setting. In [9] where a searcher robot is tasked to find the location of a coffee mug. The search space is abstracted to an undirected graph structure $G(N, E)$

that represents the topology of the environment. A searcher robot can be at any vertex N in the graph at time t ($s(t)$) and can move between two vertices provided that exists an edge E in between. Similary the target is located in one of the vertices($e(t)$). In the case of both the object and the target are in the same vertex the target is assumed to be immediately detected. Each node N is associated with a probability of having the object O in it. $P(C|s(t) = e(t))$ is the probability of detecting the object when both the robot and the target object is in the same node. Then the searcher's objective function is defined as:

$$J(U) = \sum_{t=0}^{T} \gamma^{\tau} P(C|s(t) = e(t))P((t) = e(t)) \tag{2}$$

The searcher's aim is to find the set of $S = [s(1)..s(T)]$ that maximizes $J(U)$ with $\gamma = [0, 1]$, $\tau = t + \alpha(s(t))$. The term γ^{τ} acts as a discount on the $J(U)$ forcing the searcher to favor a S with lesser cardinality. A unique aspect of this work is to couple the object search and the subsequent action. The assumption made here is that the cost of action is dependent on the N the object is found. This cost is defined as the duration of action. Therefore, by letting $\alpha(s(t))$ as the expected cost of action if the target is found at $s(t)$, the action is injected into the search plan maximization formulation. The authors attempts to formulate the problem by a Partially Observable Markov Decision Process (POMDP). They cite a previous work where the same authors show that using a POMDP, the joiint planning space becomes intractable even in the case of a few graph nodes. The authors then propose the finite-horizon path enumeration algorithm (FHPE) which is more scalable then a POMDP solution at the cost of approximating it [10]. The experiments are performed on a mobile robot that attempts to find a mug in three possible locations in the environment. They note that on a 3.0 GHz P4 computer with 2GB RAM is the limit of what the current state-of-the-art POMDP solvers. The experiments are done comparing two different POMDP solvers and the proposed algorithm.

Lau et al. [18] devised an algorithm with the a priori knowledge of a map with places and edges connecting places (much like a topological map) (fig. 3). The environment is assumed to be decomposed into regions which are represented as topological nodes in the map. The size, structure and distances between each place is known. The probability of the target being located inside a node in the map is updated after each search operation. This update rule is different from Yes method in that it operates on a larger piece ofspace rather then view point. Search time for each node and cost of traveling between nodes are also given. Size of the nodes matters: nodes with larger areas are expected to have several targets. The algorithm does not provide a solution for object searching in the smaller scale , the concern here is the visit sequence of nodes and not what the searcher should do once it arrives to a node.

4 Hybrid and various approaches

Hybrid methods combine the probabilistic framework with geometric constraints. This often involves tesselating a metric representation and assigning a probability density value to a subset of the environment. Tsotsos et al. starts attacking the object search problem by showing that looking for a known object with an arbitrary pose in 3D in a 3D space is NP-hard [17]. The formulation of the problem is given in [22] as follows: Let Ω be the search region whose structure is known *a priori*. To discretize the search region, Ω is tessellated into identically sized cells, $c_1...c_n$ and $\Omega = \cup_{i=1}^{n} c_i$. The area outside of the search region is represented by a single cell c_0. A sensing action s is then defined as taking an image of Ω from a view point v and running a recognition algorithm to determine whether the target object o is present or not. In the general case, the parameter set of s consists of camera position (x_c, y_c, z_c), pan-tilt angles (p, t), focal length f and a recognition algorithm a; $s = s(x_c, y_c, z_c, p, t, a)$. The cost of a search plan $S = s_0...s_i$ is then given as $C(S)$.

A search agent starts with an initial probability density function (PDF) on target object location over Ω. The authors examines the case where there is exactly one target object in the environment either inside or outside the search region. This means that all cells will be dependent and every sensing action will influence the values of all cells. Let β be a successful detection event and α_i the event that the center of o is at c_i. The probability update rule after each s with a non-detection result is then:

$$\mathbf{p}(\alpha_i|\neg\beta) = \frac{\mathbf{p}(\alpha_i)(1 - \mathbf{p}(\beta|\alpha_i))}{\mathbf{p}(\alpha_0) + \sum_{j=1}^{n} \mathbf{p}(\alpha_j)(1 - \mathbf{p}(\beta|\alpha_j))} \tag{3}$$

Note that for $i = 0$, $\mathbf{p}(\beta|\alpha_i) = 0$, i.e. a successful detection is impossible if the object is outside the search region . Therefore after each sensing action with a non-detection result the probability mass inside Ω shifts towards c_0 and the rest of Ω which was not in field of view. Now that the environment is defined and an update rule is given the next step is to define how to select the best next view given a PDF. First, candidate robot positions are generated by randomly picking samples from the traversable portion of Ω. This results in several candidate sensor poses.

The next best view point is then defined as:

$$\operatorname*{argmax}_{j=1..N} \sum_{i=1}^{n} \mathbf{p}(\alpha_i) V(c_i, j) \tag{4}$$

Where N is the number of candidate view points and V is defined as:

$$V = \begin{cases} 1, & \text{if } c_i \text{ is inside of the } j^{\text{th}} \text{ view cone} \\ 0, & \text{otherwise} \end{cases}$$

A similar framework is also applied on different platforms [18] Saidi et al. looked at the problem in similar way as the previosly described formulation. He used a rating function composed of probability of detecting the object (DP), the newvolume of Ω that will be seen (NI) and the cost in time/energy to reach a new view point (MC). Together the rating function is deinfed as

$$RF(c) = \alpha_D PDP(c) + \alpha_N INI(c) + \alpha_M CMC(c) \tag{5}$$

with c representing the sensor parameters for a candidate view cone and $\alpha_D P$, $\alpha_N I$, $\alpha_M C$ being weighting factors. The algoritm is implemented on a real humanoid robot and experiments are performed. The experiments involved a humanoid robot searching for an object in a room of $6x4x1.5$ meter in size. He reported that on average it took 15 views for the robot to find the target object.

4.1 Object Search using semantic a priori information

We see the world in scenes and the context predicates the objects to expect in it (for a review of the relation of objects and context see [4]). When thinking about a specific object, quite generally one cannot help but picture the place the object usually appears or other functionally related objects. What we visually perceive imposes what kind of objects to expect in the scene and their spatial layouts relative to each other. Biederman has shown that in human subjects when an object with a certain context is shown, the recognition of other contextually related objects are performed faster [5]. In some cases, objects that cannot be recognized in isolation can be inferred in the presence of contextual information. From a active visual search point of view, any spatial hint about the location of target object given by a prior contextual source is valuable.

Further, Biederman defines five types of properties that outlines the characteristics of a scene : support (objects tends to be physically supported and not float in midair), position (in known scenes objects appear in similar locations), probability (objects appears in some scenes and not others), size (objects have typical size relations with other objects), interposition (objects hides what is behind them). In this representation, apart from the interposition relation, all the others hints about the possible object locations in real-world scenes.

While no previous work on object search explicitly make use of the theoritically proven relations between objects, Garvey, in his 1976 technical note, mentioned indirect search as a way to facilitate the object search task [8]. Indirect search is about using an intermadiate object to guide the search which hints about the location of the target object (he experiments with a phone resting on a table which falls in line with the support relation of Biedermans scene characteristics). The additional a priori information limits the search space significantly in some cases by hinting on the target object location. As in most of the previous work on object search, Garvey does not consider searching in a large-scale spatial representation . Instead the search domain is limited to a captured image. Wixson and Ballard [21] provide a careful study of the behavior of search efficiency using intermediate objects. They concluded that for indirect search to be efficient, the intermediate object should be recognizable at low cost and its spatial relation to the target object should point to a relatively small space. The authors showed that using relatively typical values there is an eight fold increase in the search search efficiency compared to direct search.

A robot that interacts with people and objects needs to be able to acquire and communicate semantic information in order to function in the real world. The semantics of an environment presents strong cues for the location objects and can therefore act as a prior on object locations in the real worl. Ekvall et al. presented a system that combines SLAM and object search. First a robot is guided through the environment and a partial metric map is built consisting of several rooms. For each room, multiple positions in the navigable free space is determined autonomously. Then during object search the robot visits each of these positions to perform the object recognition step. This step involves first generating object hypothesis in the image domain based on receptive-field histograms. This results in areas of the image where the object is likely to be present. The next step is then to zoom in these regions and run a SIFT based object recognizer.

The idea of performing indirect search by exploiting the spatial relations between objects are presented in [1, 15]. Here the authors first provide a definition for the spatial relations *ON*. The mechanical support relation *ON* is highly relevant to the everyday placement of objects. Therefore a robot that can perceive and evaluate spatial relations can make use of this information while searching for objects. As an example, if a book is known to be on the table then a robot can used this information to constrain its search space. The search algorithm proposed by the authors follows the formulation of Tsostsos. However an *ON* relation between the target object and a secondary object is given to the robot beforehand. Based on that the authors performs experiments and show that under the indirect search strategy, the number of view cones until the object is found decreases significantly. Another strategy that the authors proses is direct informed search. In this case the robot has the same a priori information as in indirect search but still directly looks for the target object. Compared to uninformed search where no prior is given, direct informed search performs better than uninformed search but worse than the indirect search.

5 Discussion and Conclusion

The task of reliable and efficient object search is still an open question in robotics. As of now, it seems that the inevitable need for a mobile robot having to locate and interact with objects has not received the attention it deserves. A mobile robot that nears human performance levels should be able to fuse different kinds of information, spatial configuration of the world, visual features, semantic information and so on. This means a unification across multiple disciplines in the robotics field; semantic mapping, computer vision, cognitive robotics to name a few. While the purely geometric methods may guarantee that the searcher will leave no place in the environment unchecked, generally the time it takes to complete the search task is simply intractable. The searcher simply treats all parts of the environment equally whereas most object whereabouts are not distributed uniformly in the world. On the other hand, purely probabilistic methods that works on the graph level fails to take into account the real world considerations that a robot will face such as occlusion, illumination, signal reflection and so on. Therefore, one should augment the purely probabilistic methods with taking the configuration of the environment into account. The pursuit/evasion and planning literature provides strong cues for possible methods. Semantic information can then act as a way to get priors that are inline with the current state of the world.

References

[1] Alper Aydemir, Kristoffer Sjöö, and Patric Jensfelt. Object search on a mobile robot using relational spatial information. In *Proc. of the 11th Int Conference on Intelligent Autonomous Systems (IAS-11)*, August 2010.

[2] R. Bajcsy. Active perception. In *Proceedings of IEEE*, 1988.

[3] J. E. Banta, Yu Zhien, X. Z. Wang, G. Zhang, M. T. Smith, and Mongi A. Abidi. Best-next-view algorithm for three-dimensional scene reconstruction using range images. volume 2588, pages 418–429. SPIE, 1995.

[4] Moshe Bar and Shimon Ullman. Spatial context in recognition. *Perception*, 25:324–352, 1993.

[5] Irving Biederman. On processing information from a glance at a scene: some implications for a syntax and semantics of visual processing. In *UODIGS '76: Proceedings of the ACM/SIGGRAPH workshop on User-oriented design of interactive graphics systems*, pages 75–88, New York, NY, USA, 1977. ACM.

[6] C. Connolly. The determination of next best views. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 432 – 435, mar 1985.

[7] Görkem Şafak. The Art-Gallery Problem: A Survey and an Extension. Master's thesis, Royal Institute of Technology, Stockholm, Sweden, 2009.

[8] Thomas David Garvey. *Perceptual strategies for purposive vision.* PhD thesis, Stanford, CA, USA, 1976.

[9] Geoffrey Hollinger, Dave Ferguson, Siddhartha Srinivasa, and Sanjiv Singh. Combining search and action for mobile robots. In *ICRA'09: Proceedings of the 2009 IEEE international conference on Robotics and Automation*, pages 800–805, Piscataway, NJ, USA, 2009. IEEE Press.

[10] Geoffrey Hollinger and Sanjiv Singh. Proofs and experiments in scalable, near-optimal search by multiple robots. In *Robotics: Science and Systems*, June 2008.

[11] An Object In, Ro Sarmiento, Rafael Murrieta-cid, and Seth Hutchinson. A sample-based convex cover for rapidly finding. In *In Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 3486–3491, 2005.

[12] Without Geometric Maps, Benjamin Tovar, and Steven M. Lavalle. Optimal navigation and object finding. In *In Proc. IEEE International Conference on Robotics and Automation*, pages 464–470, 2003.

[13] J p. Laumond and C. Nissoux. Visibility-based probabilistic roadmaps for motion planning. *Journal of Advanced Robotics*, 14:2000, 2000.

[14] Richard Pito and Ruzena Bajcsy. A solution to the next best view problem for automated cad model acquisition of free-form objects using range cameras. In *Proceedings of the SPIE Symposium on Intelligent Systems and Advanced Manufacturing*, 1995.

[15] Kristoffer Sjöö, Alper Aydemir, and Patric Jensfelt. Mechanical support as a spatial abstraction for mobile robots. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'10)*, October 2010.

[16] Antonio Torralba, Monica S. Castelhano, Aude Oliva, and John M. Henderson. Contextual guidance of eye movements and attention in real-world scenes: the role of global features in object search. *Psychological Review*, 113:2006, 2006.

[17] J. K. Tsotsos. On the relative complexity of active vs. passive visual search. *International Journal of Computer Vision*, 7(2):127–141, 1992.

[18] John K. Tsotsos and Ksenia Shubina. Attention and visual search : Active robotic vision systems that search. In *International Conference on Computer Vision Systems ICVS'07*, page 539, Washington, DC, USA, 2007. IEEE Computer Society.

[19] Shrihari Vasudevan, Stefan Gächter, and Roland Siegwart. Cognitive spatial representations for mobile robots perspectives from a user study. In Proc. ICRA Workshop: Semantic Information in Robotics (ICRA - SIR 2007), 2007.

[20] Kai Welke, Tamim Asfour, and Rüdiger Dillmann. Active multi-view object search on a humanoid head. In *ICRA'09: Proceedings of the 2009 IEEE international conference on Robotics and Automation*, pages 2041–2047, Piscataway, NJ, USA, 2009. IEEE Press.

[21] Lambert E. Wixson and Dana H. Ballard. Using intermediate objects to improve the efficiency of visual search. *Int. J. Comput. Vision*, 12(2-3):209–230, 1994.

[22] Yiming. Ye. *Sensor planning for object search.* PhD thesis, 1998.

[23] Yiming Ye and John K. Tsotsos. Sensor planning in 3d object search: its formulation and complexity. In *4th International Symposium on Artificial Intelligence nad Mathematics*, pages 3–5, 1996.

Learning to predict how rigid objects behave under simple manipulation

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

July 8, 2010

Abstract

An important problem in robotic manipulation is the ability to predict how objects behave under manipulative actions. This ability is necessary to allow planning of object manipulations. Physics simulators can be used to do this, but they model many kinds of object interactions poorly, and unless there is a precise description of an object's properties its behaviour will remain quite uncertain in many cases. An alternative is to learn a motion model for objects by interacting with them. In this report we specifically address the problem of learning to predict the interactions of rigid bodies in a probabilistic framework, and demonstrate the results in the domain of robotic push manipulation. During training a simulated robot finger applies pushes to objects, and learns to predict their resulting motions. The learning does not make explicit use of physics knowledge, nor is it restricted to domains with any particular physical properties. The prediction problem is posed in terms of estimating probability densities over the possible rigid body transformations of an object under a known action. We show how to make this density estimation problem tractable by factorization, and describe the differences with an approach based on regression. This factorization creates many simpler estimation problems, and allows us to recombine the resulting predictors to model the constraints imposed by surface contacts. We show how even a very simple product of experts can generalize from learned examples: successfully predicting object motions for previously unseen object poses, push directions and objects with novel shape. We show results with a variation in the number of experts. Performance is evaluated through a combination of virtual experiments in a physics simulator, and real experiments with a 5-axis arm equipped with a simple, rigid finger.

1 Introduction

This report presents and compares several algorithms which can learn to predict the motions of a rigid object that result from an applied robotic pushing action. These algorithms do not rely on any understanding or encoding of Newtonian mechanics, but can be trained in simple online experiments in which a robot arm applies random pushes to objects of interest and extracts the resulting motions using a vision system. Properties of objects, and their interactions, are learned as distributions.

Pushing operations are encountered frequently in robotics, but have received relatively little attention in the research community. Push manipulations are interesting in that they provide a large number of unstable positions. They are also important in that push contacts are fundamental to more complex tasks such as grasping.

Mason [1] was the first to identify pushing operations as fundamental to manipulation, especially grasping. Mason develops a detailed analysis of the mechanics of pushed, sliding objects and determines conditions required for various 2D motions of a pushed object. [2] attempts to put quantitative bounds on the rate at which these predicted motions occur. [3] developed a method for finding the set of all possible motions of a sliding object, in response to an applied push. More recently, [4] has developed path planning techniques for push manipulation of 2D sliding objects, based on the use of a physics simulator for prediction.

The above work is restricted to planar sliding motions of effectively 2D objects. In contrast, there is comparatively little literature which addresses the far more complex problems of predicting the results of push manipulations on real 3D bodies, which are free to tip or roll. It is possible to use physics simulators to predict the motions of interacting rigid bodies, however this approach is reliant on explicit knowledge of the objects, the environment and key physical parameters which can be difficult to tune. Once a physics simulator has been set up for a particular scenario, it is not generalizable to new objects or novel situations.

Machine learning approaches have been developed to learn pre-specified binary affordance classes, e.g. rolling versus non-rolling objects [5], or liftable versus non-liftable objects [6]. [7] present experiments where a robot arm coupled to a vision system learns affordances (e.g. rolling or sliding) of various different objects by applying pushes and then observing the resulting motions. This kind of approach is limited, in that affordances learned for a specific object and push action, may not be generalizable to a new object, pose or push direction. Furthermore, although certain primitive classes of motion, e.g. "rolling", may be predicted, such systems cannot predict an explicit 6-DOF rigid body motion for the pushed object.

In contrast, we present a system which can learn to predict the explicit 3D rigid body transformations that will result when an object in an arbitrary orientation is subjected to an arbitrary push. The probabilistic nature of the learning enables generalization to previously unseen push directions and object poses. Furthermore, the system is often able to successfully predict the behaviours of novel objects with previously unencountered shapes.

Our preliminary work in this field appeared as a short conference paper in [8]. This report extends the work in several ways: we extend the two expert approach to show how it can generalize to a combination of many experts of arbitrary number; we develop a regression approach and compare it against our density estimation approach, showing how regression methods can struggle when generalizing to new objects with novel shapes; we include a far more extensive set of simulation experiments and additionally support these with real experiments carried out using a 5-axis robot arm, equipped with a finger, which pushes real 3D objects on a table top.

2 Representing interactions of rigid bodies

2.1 A three body system

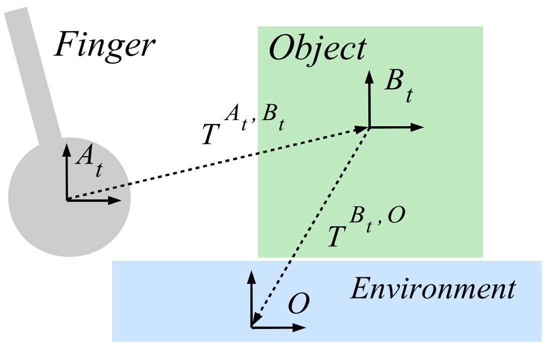


Figure 1: 2D projection at time t of a robotic finger with frame A_t, an object with frame B_t, and a ground plane with constant frame O.

Consider three reference frames A, B and O in a 3-dimensional Cartesian space. For example frame A can represent a robotic finger which pushes an object with frame B which in turn is placed on a table top with frame O as in Figure 1. While frame O is fixed, A and B change in time and are observed at discrete time steps $\ldots, t-1, t, t+1, \ldots$ every non-zero Δt. A frame X at time step t is denoted by X_t, a rigid body transformation between a frame X and a frame Y is denoted by $T^{X,Y}$ (see Figure 2).

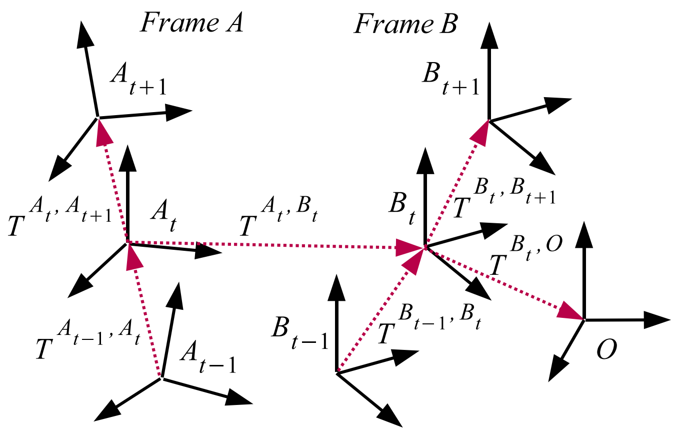


Figure 2: A system consisting of three interacting bodies with frames A and B in some constant environment with frame O can be described by six rigid body transformations T^{A_t,B_t}, $T^{B_t,O}$, T^{A_{t-1},A_t}, $T^{A_t,A_{t+1}}$, T^{B_{t-1},B_t}, and $T^{B_t,B_{t+1}}$.

From classical mechanics we know that in order to predict a state of a body, it is sufficient to know its mass, velocity and a net force applied to the body. We do not assume any knowledge of the mass and applied forces, however the transformations of a body, with attached frame B, over two time steps T^{B_{t-1},B_t} and $T^{B_t,B_{t+1}}$ encode its acceleration - the effect of the applied net force. Therefore, if the net force and the body mass are constant, the transformations T^{B_{t-1},B_t} and $T^{B_t,B_{t+1}}$ provide a complete description of the state of a body at time step t in absence of other bodies. A triple of transformations $T^{B_t,O}$, T^{B_{t-1},B_t} and $T^{B_t,B_{t+1}}$ provide a complete description of a state of a body in some fixed frame of reference O which accounts for a constant or stationary environment. Similarly, transformations $T^{A_t,O}$, T^{A_{t-1},A_t} and $T^{A_t,A_{t+1}}$ provide such a description for some other body with frame A.

The state of a system consisting of three bodies with frames A and B in some constant environment with frame O can be described by the six transformations as it is shown in Figure 2, where $T^{A_t,O}$ has been replaced by a relative transformation T^{A_t,B_t}. The transformation $T^{B_t,O}$ can be omitted, if the environment does not affect the motion of the bodies or it is explicitly modelled by one of them.

2.2 Body frame representation

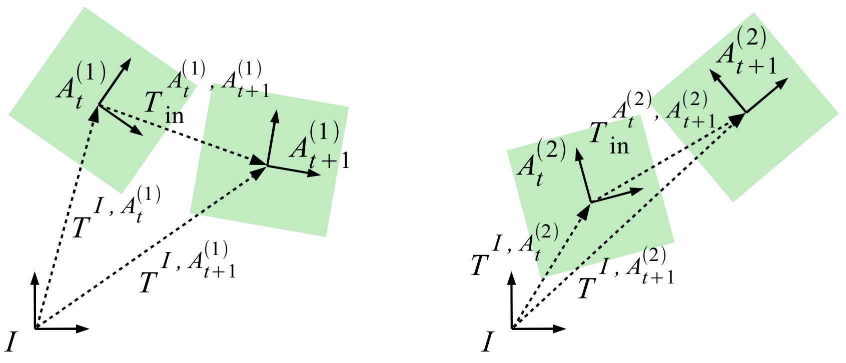


Figure 3: In the above two scenes a pose change between time step t and $t+1$ as observed in instantaneous object body frame $A^{(1)}$ and the same object in another instantaneous body frame $A^{(2)}$ given inertial frame I are both the same. However because transformations $T^{I,A^{(1)}}$ and $T^{I,A^{(2)}}$ are different, the corresponding transformations in the inertial frame are also different, i.e. $T_{in}^{A_t^{(1)},A_{t+1}^{(1)}} \neq T_{in}^{A_t^{(2)},A_{t+1}^{(2)}}$.

We expect that the behaviour of interacting bodies represented by rigid body transformations as in Figure 2 shares some statistical similarities *independently* on their global poses with respect to some current inertial frame I. Consider two scenes (1) and (2) as shown in Figure 3. A pose change between time step t and $t+1$ as observed in instantaneous object body frame $A^{(1)}$ and the same object in another instantaneous body frame $A^{(2)}$ given inertial frame I are both the same. However because transformations $T^{I,A^{(1)}}$ and $T^{I,A^{(2)}}$ are different, the corresponding transformations in the inertial frame (with subscript *i*) are also different, i.e. $T_{in}^{A_t^{(1)},A_{t+1}^{(1)}} \neq T_{in}^{A_t^{(2)},A_{t+1}^{(2)}}$.

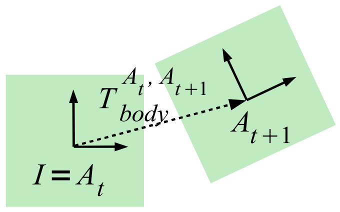


Figure 4: The body frame transformation $T_{body}^{A_t,A_{t+1}}$ represents movement instantaneous object frame A_t at time t overlaps with inertial frame I.

Instead of using inertial frame-dependent transformation $T_{in}^{A_t,A_{t+1}}$, one can represent object transformations as observed in the object body frame. Body frame transformation $T_{body}^{A_t,A_{t+1}}$ (with subscript *b*) is obtained by moving instantaneous frame A, so that at time t it overlaps with inertial frame I (see Figure 4). Given some instantaneous object frame A_t at time t, transformation $T_{in}^{A_t,A_{t+1}}$ and because $T^{I,A_{t+1}} = T_{in}^{A_t,A_{t+1}} T^{I,A_t} = T^{I,A_t} T_{body}^{A_t,A_{t+1}}$, one can obtain transformation $T_{body}^{A_t,A_{t+1}}$ in the body frame as follows:

$$T_{body}^{A_t,A_{t+1}} = (T^{I,A_t})^{-1} T_{in}^{A_t,A_{t+1}} T^{I,A_t} \tag{1}$$

Similarly from a given transformation in body frame, instantaneous object frame A_t at t and using Equation 1, one can obtain expression for transformation $T_{in}^{A_t,A_{t+1}}$ in the inertial frame

$$T_{in}^{A_t,A_{t+1}} = T^{I,A_t} T_{body}^{A_t,A_{t+1}} (T^{I,A_t})^{-1} \tag{2}$$

If not stated otherwise in further discussion we will keep subscripts *in* while dropping subscripts *body* assuming that all transformations $T^{X,Y}$ are transformations in the body frame X obtained from $T^{X,Y} \equiv T_{body}^{X,Y} = (T^{I,X})^{-1} T_{in}^{X,Y} T^{I,X}$.

3 Prediction learning as a regression problem

The prediction problem can now be stated as: given we know or observe the starting states and the motion of the pusher, $T^{A_t,A_{t+1}}$, predict the resulting motion of the object, $T^{B_t,B_{t+1}}$. This is a problem of finding a function:

$$f : T^{A_t,B_t}, T^{B_t,O}, T^{A_{t-1},A_t}, T^{B_{t-1},B_t}, T^{A_t,A_{t+1}} \rightarrow T^{B_t,B_{t+1}} \tag{3}$$

Function 3 is capable of encoding all possible effects of interactions between rigid bodies A and B, providing their physical properties and applied net forces are constant in time. Furthermore, it can be learned purely from observations for some fixed time delta Δt.

There are two problems related to relying on such a function:

1. **Limited or no generalization capability.** A function approximating interactions between bodies A and B cannot be used for any other bodies of e.g. different shape or mass. This is because function 3 implicitly encodes information about the surfaces of A and B, which play a critical role in collisions. In this way a slight change of the objects' shape can cause a dramatic deviation of the predicted transformation $T^{B_t,B_{t+1}}$. Consequences of surface deformations are further discussed in the experimental section 5.

2. **Dimensionality problem.** For a rigid body transformation represented as a set of 6 or 7 numbers, the domain of function 3 has 30 or 35 dimensions. This problem can be alleviated by a quasi-static assumption introduced in the following section 3.1.

3.1 Quasi-static assumption

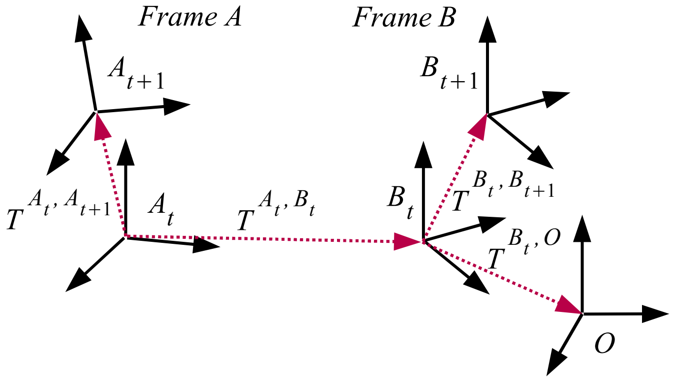


Figure 5: In quasi-static conditions two interacting bodies with frames A and B in constant environment with frame O can be described by only four rigid body transformations T^{A_t,B_t}, $T^{B_t,O}$, $T^{A_t,A_{t+1}}$ and $T^{B_t,B_{t+1}}$.

In many robotic operations, manipulations are slow, one can assume quasi-static conditions, and it is often possible to ignore all frames at time $t-1$ as it is shown in Figure 5. This conveniently reduces the dimensionality of the problem to 18 dimensions if rigid body transformations are represented by 6 numbers using e.g. Euler angles. Function 3 can then be rewritten in a simpler form:

$$f : T^{A_t,B_t}, T^{B_t,O}, T^{A_t,A_{t+1}} \longrightarrow T^{B_t,B_{t+1}} \tag{4}$$

The quasi-static assumption comes at a price however. Function 4 no longer encodes the complete information about a state of the system, in particular the velocity of frame A and B. For example transformations $T^{A_t,B_t}, T^{B_t,O}, T^{A_t,A_{t+1}}$ comprising the domain of Function 4 can be exactly the same for two very different situations. In the first situation a robotic finger approaches a non-moving (yet) object, while in the second the object begins to move after being pushed by the finger. Still the movement of the finger, the relative pose between the finger and the object as well as the object pose on the table are all the same. This ambiguity can be removed by encoding explicitly the causal relation between the finger movement and the object movement. In order to do this, two phases of the approaching before touching the object and after touching are split by detecting a moment the finger touches the object[1].

4 Predicting rigid body motions using multiple experts

4.1 Combining local and global information with two experts

It is clear that we need to enable generalization of predictions with respect to changes in shape. Consider two objects lying on a table top. Figure 6 shows two situations that are identical except for the shape of object A. It is clear that the same transformation of A's position will lead to different motions for object B in each case, mostly due to a potentially infinite number of ways shapes A and B can vary. How can we then encode the way in which the shapes of A and B alter the way they behave? We use a product of several densities to approximate the density over the rigid body transformation instead of a single value as given in the function 4.

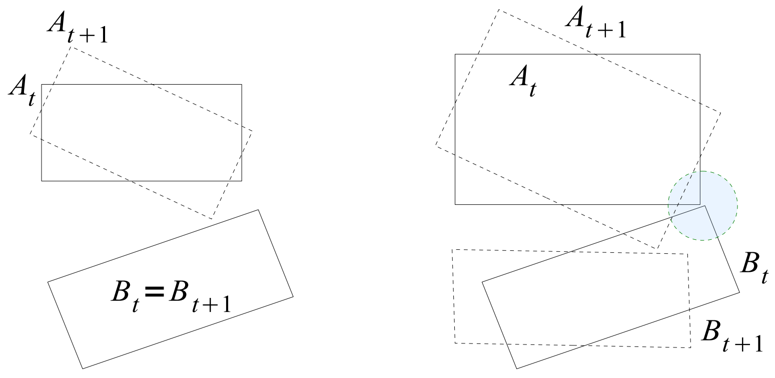


Figure 6: Two scenes, each with two objects on a table top, viewed from above. Between the two scenes only the shape of A is different. Yet when A moves the resulting transformation $T^{B_t,B_{t+1}}$ will be quite different. This shows that our predictors must take some aspect of the shape of A and B into account.

[1]In practice no algorithm could work without applying this trick.

In the simplest case one can approximate two densities, conditioned on local and global information respectively. We define the global information to be the information about changes of the pose of the whole object. The local information is specified by changes of the pose of the surfaces of A and B at the contact point, or the point of closest proximity, between the object and the finger. We model this local shape as a pair of planar surface patches, of limited extent (see Figure 7). Statistically, the greater the starting distance between these local surface patches of A and B, and/or the smaller the magnitude of the transformation $T^{A_t,A_{t+1}}$, the less likely it is that the objects will collide, and hence the less likely it is that the pose of shape B will change between t and $t+1$, or equivalently the more likely that the transformation $T^{B_t,B_{t+1}}$ will be an identity transformation Id. On the other hand, if the local surfaces A and B are close a large portion of possible transformations $T^{A_t,A_{t+1}}$ will cause collisions.

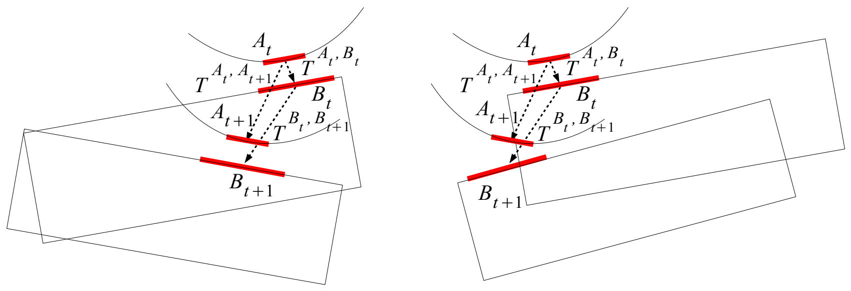


Figure 7: Two scenes, each with two objects on a table top, viewed from above. Local shapes A and B, transformations $T^{A_t,A_{t+1}}$ and T^{A_t,B_t} are the same in each scene. Still, the transformation $T^{B_t,B_{t+1}}$ is different because local shapes belong to different parts of objects.

Transformations T^{A_t,B_t}, $T^{A_t,A_{t+1}}$ and $T^{B_t,B_{t+1}}$, observed over many experimental trials for many different objects form a conditional distribution:

$$\{T^{B_t,B_{t+1}}|T^{A_t,A_{t+1}},T^{A_t,B_t}\} \tag{5}$$

While conditional distribution 5 for global frames may become unimodal, for local shapes is highly multi-modal. To see this consider two scenes with two objects, where the initial conditions are identical (Figure 7). Local shapes A and B, transformations $T^{A_t,A_{t+1}}$ and T^{A_t,B_t} are the same in each scene. Still, the transformation $T^{B_t,B_{t+1}}$ is different because local shapes belong to different parts of objects.

Consider a 2D projection at time t of a robotic finger with global frame A_t, an object with global frame B_t, and a ground plane with constant global frame O (Figure 8). Similarly, local frames A_t^l and B_t^l describe local shapes belonging to a finger and an object. The global conditional density function can be defined as:

$$p_{global}(T^{B_t,B_{t+1}}|T^{A_t,A_{t+1}},T^{A_t,B_t},T^{B_t,O}) \tag{6}$$

and similarly a local conditional density function as:

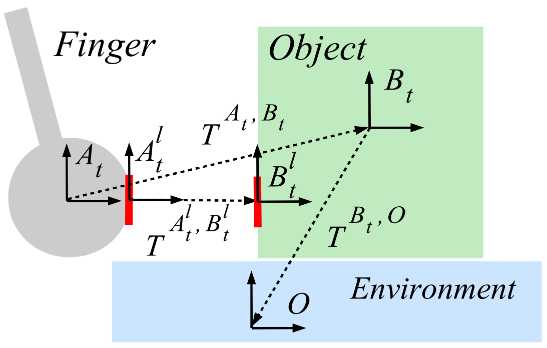


Figure 8: 2D projection at time t of a robotic finger with global frame A_t, an object with global frame B_t, and a ground plane with constant global frame O. Local frames A_t^l and B_t^l describe the local shape of the finger and an object at their point of closest proximity.

$$p_{local}(T^{B_t^l,B_{t+1}^l}|T^{A_t^l,A_{t+1}^l},T^{A_t^l,B_t^l}) \tag{7}$$

The only problem is to determine the above three transformations in the body frame of the local shapes. For a particular situation shown in Figure 8 from object rigidity and using Equation 1 we have:

$$T^{A_t^l,A_{t+1}^l} = (T^{I,A_t^l})^{-1}T_{in}^{A_t,A_{t+1}}T^{I,A_t^l} \tag{8a}$$

$$T^{B_t^l,B_{t+1}^l} = (T^{I,B_t^l})^{-1}T_{in}^{B_t,B_{t+1}}T^{I,B_t^l} \tag{8b}$$

where I is the inertial frame. $T^{A_t^l,B_t^l}$ can be determined directly from the shape frame:

$$T^{A_t^l,B_t^l} = (T^{I,A_t^l})^{-1}T_{in}^{A_t^l,B_t^l}T^{I,A_t^l} \tag{9}$$

To predict the rigid body transformation of an object when it is in contact with others we are faced with how to represent the constraints on motion provided by the contacts. We do this using a product of experts. The experts represent by density estimation which rigid body transforms are (in)feasible for each frame of reference. In the product, only transformations which are feasible in both frames will have high probability. For the finger-object scenario a prediction problem can then be defined as finding that transformation $T_{in}^{B_t,B_{t+1}}$ in the inertial frame which maximizes the product of the two conditional densities (experts) 6 and 7:

$$\max_{T_{in}^{B_t,B_{t+1}}} p_{global}((T^{I,B_t})^{-1}T_{in}^{B_t,B_{t+1}}T^{I,B_t}|T^{A_t,A_{t+1}},T^{A_t,B_t},T^{B_t,O})\times$$
$$p_{local}((T^{I,B_t^l})^{-1}T_{in}^{B_t,B_{t+1}}T^{I,B_t^l}|T^{A_t,A_{t+1}},T^{A_t^l,B_t^l}) \tag{10}$$

where identity 8b has been used.

Starting with some initial state of the finger A_0 and the object B_0, and knowing a trajectory of the finger $A_1, \ldots A_N$ over T time steps, one can now predict a whole trajectory of an object $B_1, \ldots B_N$ by sequentially solving a problem of maximization of the product 10.

There are two major advantages of using such products of densities, e.g. over attempting to directly approximate the function of equation 3:

1. **Generalization.** Even small differences in a local object surface can cause very different reactions $T_{in}^{B_t,B_{t+1}}$ for some given action $T_{in}^{A_t,A_{t+1}}$ (see the experimental section 5). However, such changes are unlikely to be predicted by a global density function alone. Hence, computing $T_{in}^{B_t,B_{t+1}}$ as the maximizer of the product of densities, equation 10, enhances the ability of the system to generalize between different objects and actions, because both local and global densities must simultaneously support the predicted motion hypothesis $T_{in}^{B_t,B_{t+1}}$ (see Figure 9).

2. **More efficient movement encoding and learning.** Combining information from both local and global frames, allows objects' properties to be separated into those that are common to many objects and those that are specific to the particular object in question. Common properties (e.g. impenetrability) tend to be encoded in the local surface patches distribution, function 7, whereas the global density function 6 encodes information specific to the object, such as its overall behaviour. The global density function 6 tends not to require many learning trials to provide accurate predictions, when combined with the local density function 7, which is shared or common to many different objects or situations. Thus this combination provides a movement encoding and learning method which is highly efficient.

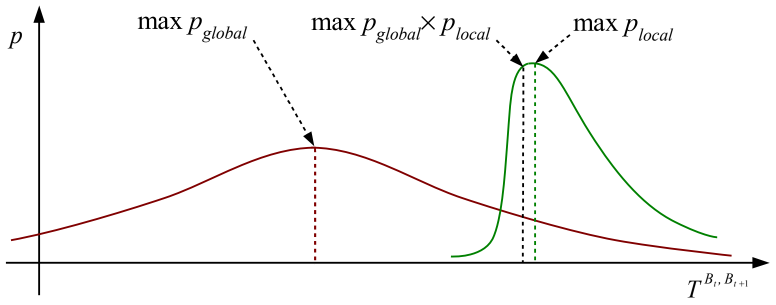
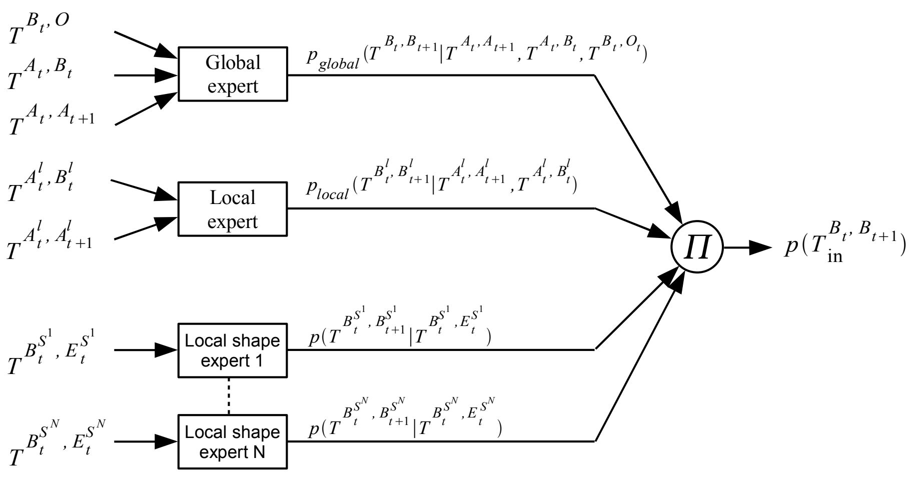


Figure 9: Varying object shape: maxp_{global} alone (equivalently regression) does not provide correct predictions as well as any linear combination of e.g. maxp_{global} and maxp_{local} cannot be used either. Maximum of the product of p_{global} and p_{local} provides approximate but correct predictions.

4.2 Incorporating information from additional experts



Global expert
Local expert
Local shape expert 1
Local shape expert N
$T^{B_t,O}$
T^{A_t,B_t}
$T^{A_t,A_{t+1}}$
$T^{A^l_t,B^l_t}$
$T^{A^l_t,A^l_{t+1}}$
$T^{B^{S^1}_t,E^{S^1}_t}$
$T^{B^{S^N}_t,E^{S^N}_t}$
$p_{global}(T^{B_t,B_{t+1}}|T^{A_t,A_{t+1}},T^{A_t,B_t},T^{B_t,O_t})$
$p_{local}(T^{B^l_t,B^l_{t+1}}|T^{A^l_t,A^l_{t+1}},T^{A^l_t,B^l_t})$
$p(T^{B^{S^1}_t,B^{S^1}_{t+1}}|T^{B^{S^1}_t,E^{S^1}_t})$
$p(T^{B^{S^N}_t,B^{S^N}_{t+1}}|T^{B^{S^N}_t,E^{S^N}_t})$
Π
$p(T^{B_t,B_{t+1}}_{\text{in}})$

Figure 10: Inputs and outputs of learned prediction system. The approach described in section 4.1, uses only local and global experts. This can be extended to include opinions from multiple local shape experts represented by coordinate frames S^N.

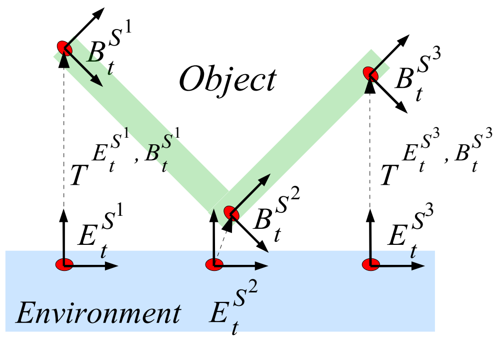


Figure 11: Co-ordinate frames can be attached to an arbitrary number of local shapes, and local experts can be learned for each of these frames, predicting a distribution of how the frame may move next, given where it is at the present time step.

In addition to learning how an object moves in response to a push, it is desirable if we can also incorporate learned information about the inherent tendencies of parts of an object to move in various directions with respect to the environment or any other objects, but regardless of whether it is being pushed or not. This additional information may help when predicting the motions of previously unseen objects, because it provides some prior knowledge about what kinds of motions are possible and which are not.

We can incorporate this additional information by attaching an arbitrary number of additional coordinate frames B^{sn_t} to various parts of the object. We then learn densities for the future motions of each of these frames, conditioned only on their relative pose $T^{E^{S^k_t},B^{S^k_t}}$ with respect to a corresponding pose $E^{S^k_t}$ of a patch on a ground plane at the present time step, ignoring any information about the motions of the pushing finger. For the k-th such frame, we estimate *the local contact conditional density*:

$$p(T^{B^{S^k_t},B^{S^k_{t+1}}} | T^{E^{S^k_t},B^{S^k_t}}) \tag{11}$$

which represent probability density over possible rigid body transformations in the body frame of the k-th local contact. The subsequent motion of the object in the inertial frame can now be predicted as:

$$\begin{aligned}\max_{T_{in}^{B_t,B_{t+1}}} & p_{global}((T^{I,B_t})^{-1} T_{in}^{B_t,B_{t+1}} T^{I,B_t} | T^{A_t,A_{t+1}}, T^{A_t,B_t}, T^{B_t,O}) \times \\ & p_{local}((T^{I,B^l_t})^{-1} T_{in}^{B_t,B_{t+1}} T^{I,B^l_t} | T^{A_t,A_{t+1}}, T^{A^l_t,B^l_t}) \times \\ & \prod_{k=1\ldots N} p((T^{I,B^{S^k_t}})^{-1} T_{in}^{B_t,B_{t+1}} T^{I,B^{S^k_t}} | T^{E^{S^k_t},B^{S^k_t}})\end{aligned} \tag{12}$$

4.3 Incorporating additional information into the global conditional density function

4.3.1 Finger-object contact

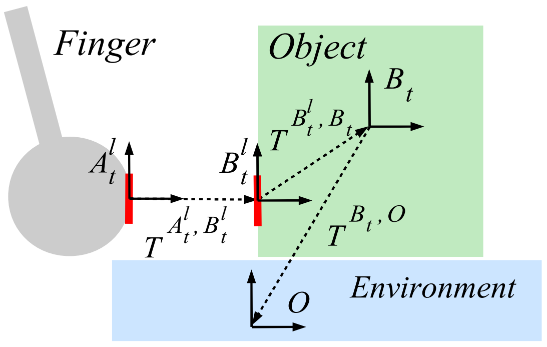


Figure 12: An alternative setup to the one from Figure 8 redefines the global conditional density function using local frame B^l_t.

The global conditional density function as in Equation 6 describes "global" behaviour of all three interacting objects. From physics we know that contacts between objects are responsible for forces and therefore determine the objects behaviour. Contacts between an object and the environment can be arbitrarily complex and they are all handled by term $T^{B_t,O}$ in density 6. However assuming that there is only one contact between the finger and an object one can use more efficient representation of the global conditional density function, in which term T^{A_t,B_t} is replaced by $T^{B_t^l,B_t}$ or equivalently by $T^{A_t^l,B_t}$ (see Figure 12). This representation is simpler because it takes into account only a local shape (B_t^l or A_t^l)-object relation instead of a more complex finger-object relation. The global conditional density function can be then redefined as:

$$p_{global}(T^{B_t,B_{t+1}}|T^{A_t,A_{t+1}},T^{B_t^l,B_t},T^{B_t,O}) \tag{13}$$

In general it is safer to use more general definition 6. However incorporating explicitly the finger-object contact information might be beneficial in some cases, as it is shown in the experimental part of this section (see Section 5.3).

4.3.2 Parametrization of objects' properties

Any additional knowledge about properties of the interacting objects (including the robotic finger and the environment) which vary in time can also be incorporated into the global conditional density function. These properties may involve shape variations such as length of an object, but also properties which cannot be expressed by shape such as friction, restitution, viscosity, dislocated centre of mass, etc. The global conditional density function can be then expressed as:

$$p_{global}(T^{B_t,B_{t+1}}|T^{A_t,A_{t+1}},T^{A_t,B_t},T^{B_t,O},X) \tag{14}$$

where $X \in \mathbb{R}^n$ is some n-dimensional vector describing additional objects' properties.

4.4 Learning as density estimation

We use memory-based learning in which all *learning samples* are stored during learning. The learning samples create a global joint distribution, local joint distribution and N local contact joint distributions:

$$\{T^{A_t,B_t},T^{B_t,O},T^{A_t,A_{t+1}},T^{B_t,B_{t+1}}\} \tag{15}$$

$$\{T^{A_t^l,B_t^l},T^{A_t,A_{t+1}},T^{B_t,B_{t+1}}\} \tag{16}$$

$$\{T^{B^{S_t^k},B^{S_{t+1}^k}},T^{E^{S_t^k},B^{S_t^k}}\} \qquad \text{for } k = 1\dots N \tag{17}$$

We address $3D$ rigid bodies, subject to 6-DOF transformations, so that distributions 15, 16 and 17 have $4\times 6 = 24$, $3\times 6 = 18$ and $2\times 6 = 12$ dimensions respectively. During prediction conditional densities 6, 7 and 11 are created online from learning sample sets (i.e. from the above joint distributions).

Consider N D-dimensional sample vectors X_i drawn from some unknown distribution. We would like to find an approximation of this distribution in the form of a density function $p(X)$. Kernel density methods with Gaussian kernels (see e.g. [9])

estimates the density $p(X)$ for any given vector X as a sum of N identical multivariate Gaussian densities centred on each sample vector X_i:

$$p(X) = C_{norm} \sum_{i=1...N} \exp\left[-\frac{1}{2}(X - X_i)^T \mathbf{C}^{-1}(X - X_i)\right] \tag{18}$$

where a constant $C_{norm} = [N(2\pi)^{D/2}|\mathbf{C}|^{1/2}]^{-1}$ and $\mathbf{C}$ is a $D \times D$ sample covariance matrix. For simplicity, we assume that $\mathbf{C}$ is diagonal. The above equation can be re-written in a new simpler form ([9]):

$$p(X) = \frac{1}{N} \sum_{i=1...N} \left[\prod_{j=1...D} K_{h_j}(X^j - X_i^j)\right] \tag{19}$$

where K_{h_j} are 1-dimensional Gaussian kernel functions:

$$K_{h_j}(X^j - X_i^j) = \frac{1}{(2\pi)^{1/2} h_j} \exp\left[-\frac{1}{2}\frac{(X^j - X_i^j)^2}{h_j^2}\right] \tag{20}$$

and D parameters h_j are called bandwidth $H \equiv (h_1, \dots, h_D)$. The bandwidth H is estimated from all distribution learning samples using the “multivariate rule-of-thumb” see [9].

Let us decompose each D-dimensional sample vector X_i into two vectors: K-dimensional Y_i and L-dimensional Z_i so that $X_i \equiv (Y_i, Z_i)^T$ and $D = K + L$. Knowing bandwidth H or equivalently diagonal covariance matrix $\mathbf{C}$ for sample set $\{X_i\} \equiv \{(Y_i, Z_i)^T\}$, we can compute conditional density $p(Z|Y)$ for some given vectors Y and Z using the following two step procedure:

1. Find a set of M weighted samples $\{(Z_i, w_i)\}$ representing a conditional distribution for given vector Y, such that Y_i which corresponds to Z_i lies within some predefined maximum Mahalanobis distance d_{max} to vector Y. Mahalanobis distance d_i between sample vector Y_i and vector Y is defined as:

$$d_i = (Y - Y_i)^T \mathbf{C}_Y^{-1}(Y - Y_i) \tag{21}$$

where diagonal covariance $\mathbf{C}_Y$ is defined as:

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_Y & 0 \\ 0 & \mathbf{C}_Z \end{bmatrix} \tag{22}$$

Weights w_i are computed from distance d_i as:

$$w_i = \exp[-d_i/2] \tag{23}$$

and normalized for all M weights w_i. Normalized weight w_i can be interpreted as a probability of generating Y_i from a multivariate Gaussian centred at Y with covariance $\mathbf{C}_Y$.

2. Compute conditional probability density $p(Z|Y)$ as:

$$p(Z|Y) = \sum_{i=1...M} w_i \exp\left[-\frac{1}{2}(Z-Z_i)^T \mathbf{C}_Z^{-1}(Z-Z_i)\right] \tag{24}$$

For simplicity, the density product 10 is maximized using the differential evolution optimization algorithm [10]. This requires the ability to evaluate and sample from each distribution comprising product 10.

All conditional distributions are represented as a weighted set of samples $\{(Z_i, w_i)\}$. Computation of a probability density for some given vector Z is realized as in Equation 24. Sampling consists of a two step procedure:

1. Choose vector Z_i from a set of samples $\{(Z_i, w_i)\}$ using an importance sampling algorithm with importance weights w_i ([11]).
2. Sample from a multivariate Gaussian centred at Z_i with covariance $\mathbf{C}_Z$.

5 Results

5.1 Introduction

5.1.1 Experimental setup

We have tested the introduced prediction algorithms in simulation experiments using PhysX physics engine [12]. We also performed real experiments using 5-axis Katana robotic manipulator [13] equipped with a single rigid finger and where we capture the motion of the polyflap using a vision model-based tracking system [14]. Simulation experiments are useful in that they can provide large amounts of test data with perfectly known ground-truth although not necessary corresponding to reality. We have also demonstrated the capability of these systems to predict motions of a real object being pushed by a real robot.

We present a series of experimental scenarios demonstrating two prediction problems described below:

Generalization with respect to novel actions Predictions where actions during learning and prediction are drawn from the same (interpolation) or different (extrapolation) action sets. All shapes during learning and prediction are fixed.

Generalization with respect to novel shapes Predictions where shapes during learning and prediction are drawn from the same (interpolation) or different (extrapolation) shape sets. All actions during learning and prediction are drawn from the same action set.

All the above prediction problems have been addressed in simulation environments in Section 5.2 and Section 5.3. Real experiments are presented in Section 5.4 addressing only the first prediction problem mostly due to limitations of the vision tracker.

In each test scenario, we compare the performance of three approaches (further called predictors):

1. A multiple expert method with global, local and local shape experts (see Section 4).
2. A single global expert method as a ground-truth comparison to the multiple expert method (see Section 3).
3. A state of the art regression method described in Section 5.1.4.

Multiple experimental trials were performed, in which a robotic arm equipped with a finger performs a random pushing movement of length approximately 25 cm towards an object placed at a random initial pose (Figure 26). In each experiment data samples are stored over a series of such random trials. Each trial lasts 10 seconds, while data samples are stored every $1/15$th of a second.

We performed 10-fold cross-validation where at the beginning of each experiment all the trials are randomly partitioned into 10 subsets. Prediction was then subsequently performed (10 times) on each single subset, while learning was always performed on the remaining 9 subsets of these trials. All the results were then averaged to produce a single estimation.

5.1.2 Performance measure

In all experiments, we take the output of either a physics simulator or the tracked pose of a real object to be ground-truth, and compare it against predictions forecast by the learned system. Prediction performance is evaluated as follows.

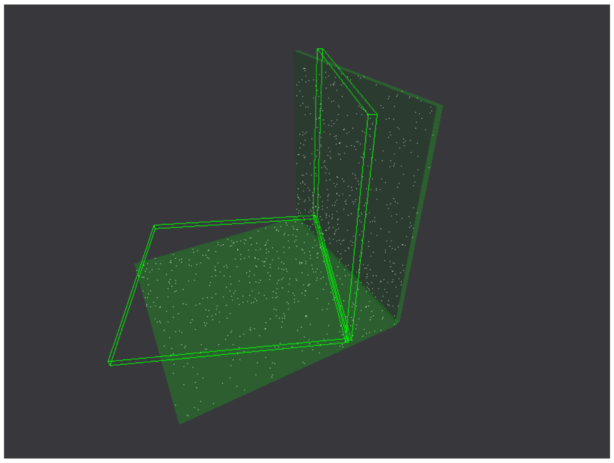


Figure 13: Randomly chosen points (white dots) rigidly attached to the object at the ground-truth pose (solid colour). Not shown corresponding points are also attached to the object at the predicted pose (wire-frame).

At any particular time step, t, a large number, N, of randomly chosen points $p_n^{1,t}$, where $n = 1 \dots N$, are rigidly attached to an object at the ground-truth pose, and the corresponding points $p_n^{2,t}$ to an object at the predicted pose (see Figure 13). At time step t, an average error E_t can now be defined as the mean of displacements between points on the object at the predicted pose and points on the object at the ground-truth pose:

$$E_t = \frac{1}{N} \sum_{n=1...N} |p_n^{2,t} - p_n^{1,t}| \tag{25}$$

Note that for each robotic push action, we predict approximately 150 consecutive steps into the future, with no recursive filtering or corrector steps, hence it is expected that errors will grow with range from the initial object pose. We therefore find it more meaningful to normalize all errors with respect to an "average range", R_t, of the object from its starting position, defined as:

$$R_t = \frac{1}{N} \sum_{n=1...N} |p_n^{1,t} - p_n^{1,0}| \tag{26}$$

For a test data set, consisting of K robotic pushes, each of which breaks down into many consecutive predictions over T time steps, we can now define an *normalized average error*:

$$E_{av} = \frac{1}{K} \sum_{k=1...K} \frac{1}{T} \sum_{t=1...T} \frac{E_t}{R_t} \tag{27}$$

For each set of test data, we also report an *normalized final error*, E_f which represents the typical discrepancy between prediction and ground truth that has accumulated by the end of each full robotic push:

$$E_f = \frac{1}{K} \sum_{k=1...K} \frac{|p_n^{2,T} - p_n^{1,T}|}{R_T} \tag{28}$$

The normalized average error and the normalized final error for each prediction method and in each experiment are collected in Table 1.

5.1.3 Prediction with multiple experts

The multi-expert method uses local contact experts in order to improve prediction of the object motion. We tested two types of objects: a polyflap (an object consisting of two flat square flanges) and a box. In both cases apart from the local expert describing finger-object contact behaviour (see Section 4.1), 3 additional experts are used (see Figure 14):

Polyflap case: The first edge type is represented by 2 identical experts attached to the front and the top part of a polyflap. Another edge in the middle of a polyflap is represented by a separate expert.

Box case: All edges are assumed to be of the same type therefore all 3 experts are identical although attached to different parts of a box.

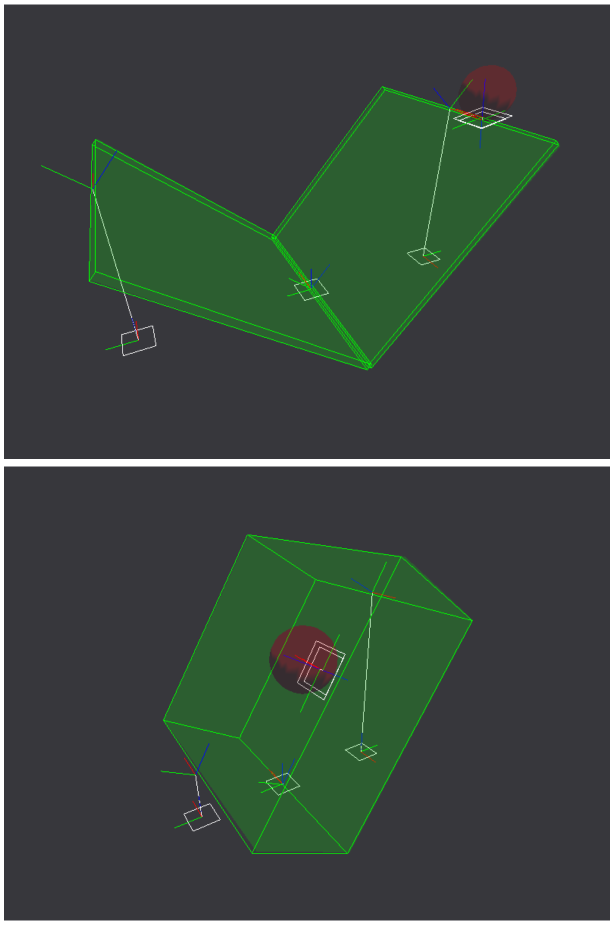


Figure 14: In the polyflap case (top panel) the first edge type is represented by 2 identical experts attached to the front and the top part of a polyflap (small white surface patches with reference frames). A separate expert represents the edge in the middle of a polyflap. In the box case (bottom panel) there are 3 identical experts attached to 3 different parts of a box.

If two or more experts are identical they all share the same joint distribution during learning and prediction.

The bandwidth of all distributions were additionally tuned by hand and kept constant throughout all the experiments.

5.1.4 Prediction by regression

To provide a comparison with a more conventional function approximation approach, we employed a powerful regression algorithm – Locally Weighted Projection Regression (LWPR) [15]. LWPR is an online algorithm that uses dimensionality reduction to learn non-linear mappings from high-dimensional input spaces, by using a set of local models.

Thus we attempted to predict the rigid body motion of an object (frame B) subjected to a push from a finger (frame A), by learning a non-linear function G. The function G is analogous to the global expert of the probabilistic approach in section 4.

The input domain for the regression was an 18 dimensional vector formed by the concatenation of three 6-DOF transformations, where each transformation was represented as a displacement and three Euler angles. The 6 dimensional output vector was simply the change in pose of object B, so that

$$G : T^{A_t,B_t}, T^{B_t,O}, T^{A_t,A_{t+1}} \rightarrow T^{B_t,B_{t+1}}. \tag{29}$$

The parameters for LWPR were tuned by hand on a basic test set. The (diagonal) distance metric **D** was held fixed, and initialized so that the local model receptive fields were small for the finger-object interaction T^{A_t,B_t}, and large for the other input dimensions.

The regression scheme was implemented using the LWPR software library [16].

5.2 Generalization to predict motions from novel actions

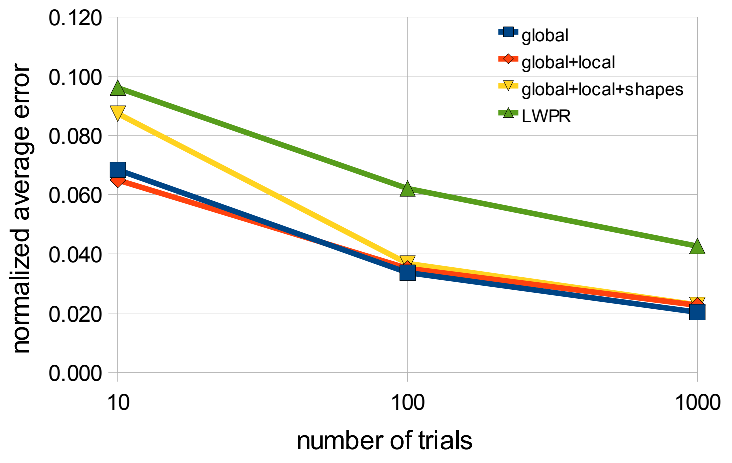
5.2.1 Interpolative generalization of push directions

In **Experiment 1** with a virtual robot, pushes with random directions were applied to a polyflap of a fixed shape. Some pushes were randomized about a direction approximately orthogonal to the contacted polyflap face, and some were randomized about a direction at an oblique angle. Both orthogonal and oblique pushes were used during training, new examples of each class of push direction were then used in testing. We consider this a test of “interpolative” generalization, in that, although the test pushes are different from any particular push in the training set, the directions of the test pushes lie within a space of directions that are reasonably spanned by training examples.

Figure 15 shows how the average and final prediction error decreases with increased number of trials used in learning for four tested prediction methods. The density-based methods outperforms LWPR regression in each case. It is also clear that using the introduced prediction error in the interpolative generalization case, multiple expert methods do not show any advantage over the single global expert method if they are learned on the same data.

5.2.2 Extrapolative generalization of push directions

In **Experiment 2**, only the orthogonal type of pushes were used during training, pushes being applied to both the front and rear surfaces of a polyflap. However, only the oblique type pushes were used during testing. We consider this to be a test of “extrapolative” generalization, in that the push directions used in testing are all qualitatively different from those used in training - the test push directions do not lie in the same region of data covered by the training examples.



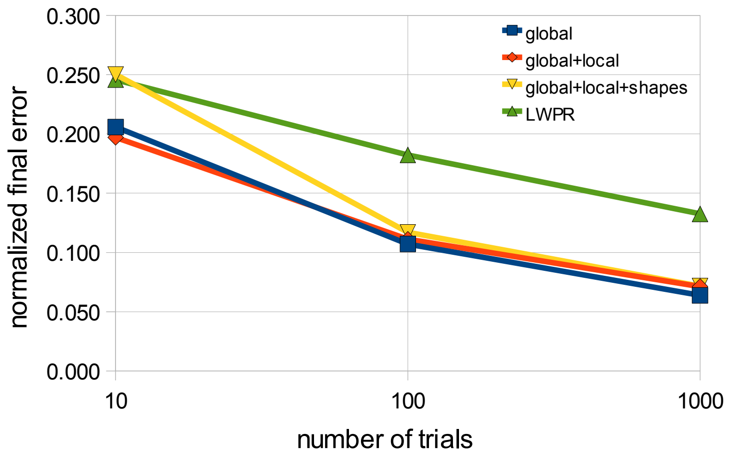


Figure 15: Experiment 1. Decrease in average (top) and final (bottom) prediction errors with increased number of trials used in learning, for four different prediction method.

The regression method makes the smallest error mostly because it manages to differentiate between tipping/toppling and forward pushing as it is shown in Figure 16. The single expert method predicts only tipping and toppling, thus the multiple experts method which also uses the global density is unable to overcome this error. Predictions of the regression and the single expert frequently violates impenetrability constraint of a rigid body. Because multiple experts help in preserving this constraint, the multiple expert method tends to make smaller errors than the single expert one as it is shown in Figure 17. Predicted trajectories are usually more stable so as the final error E_f (Figure 16).

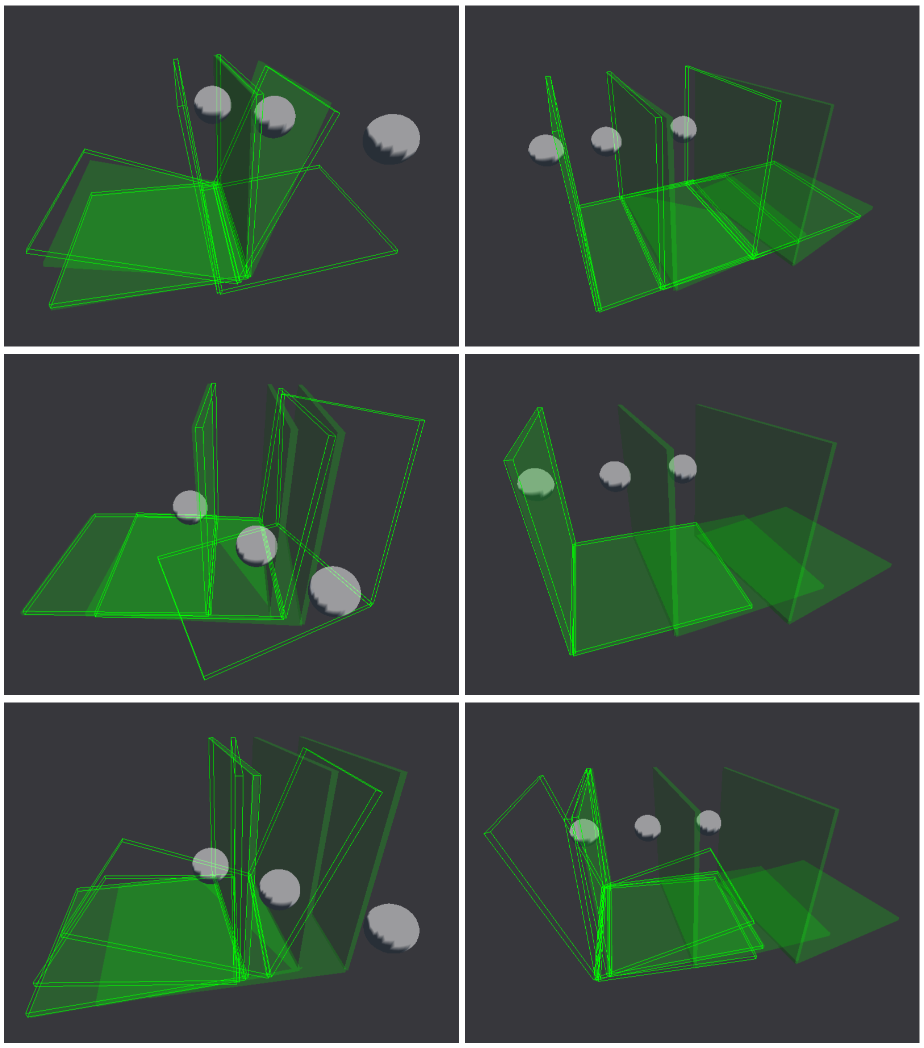
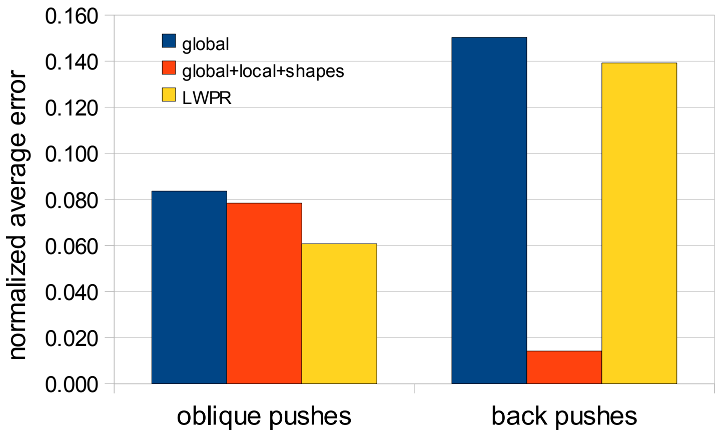


Figure 16: Experiment 2 and 3. All methods frequently make an error regarding whether to tip or to topple (top left). However, the regression method manages to differentiate between tipping/toppling (top left) and forward pushing (middle left), while the density-based methods predict only the first type of behaviour (bottom left). The multiple expert method steadily predicts a simple forward movement (top right) which is nevertheless far closer to the actual polyflap behaviour than physics-violating predictions of the regression and the single expert methods (middle right and bottom right). The ground-truth and the predicted poses are shown as solid and wire-frame shapes respectively.

In **Experiment 3**, only the orthogonal type pushes to the front of a polyflap were



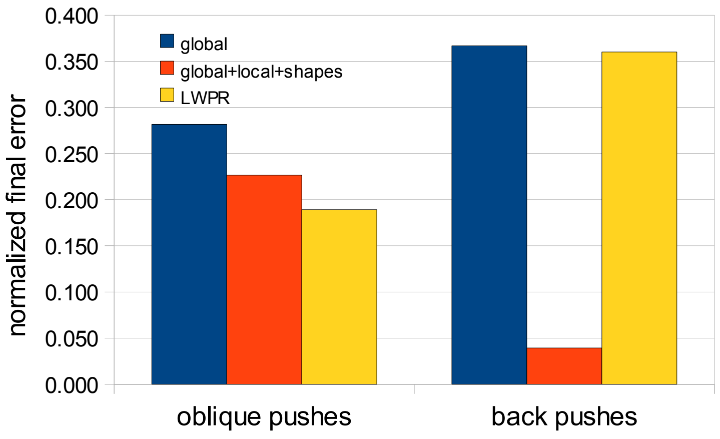


Figure 17: Experiment 2 and 3. Extrapolative action generalization errors for oblique pushes and back pushes.

used in training, along with oblique type pushes. In contrast, the system was tasked to make predictions for orthogonal type pushes applied to the back of a polyflap.

The regression method and the single expert method fail to predict the polyflap behaviour (see Figure 17). Although the multiple expert method steadily predicts a simple forward movement of a polyflap it does not violate physics and roughly corresponds to the actual behaviour (Figure 16).

5.3 Generalization to objects with novel shapes

5.3.1 Extrapolative generalization to novel shapes

In **Experiment 4**, training data featured only an ordinary polyflap constructed from two square faces joined at right angles. However, predictions were then made for test data in which the geometry of the polyflaps varied randomly in several ways, see Figure 18. Variations included: the angle at which the two rectangular flanges of the polyflap were connected; the width of each rectangular flange; the transverse offset of one flange with respect to the other. We again consider this to be an example of "extrapolative" generalization, in that the different kinds of test shape are not spanned by the single shape used during training.

All prediction methods performed well with a slight advantage of the multiple expert method (Figure 20). A typical error is related to inability to differentiate between tipping/toppling behaviour of a polyflap (Figure 18).

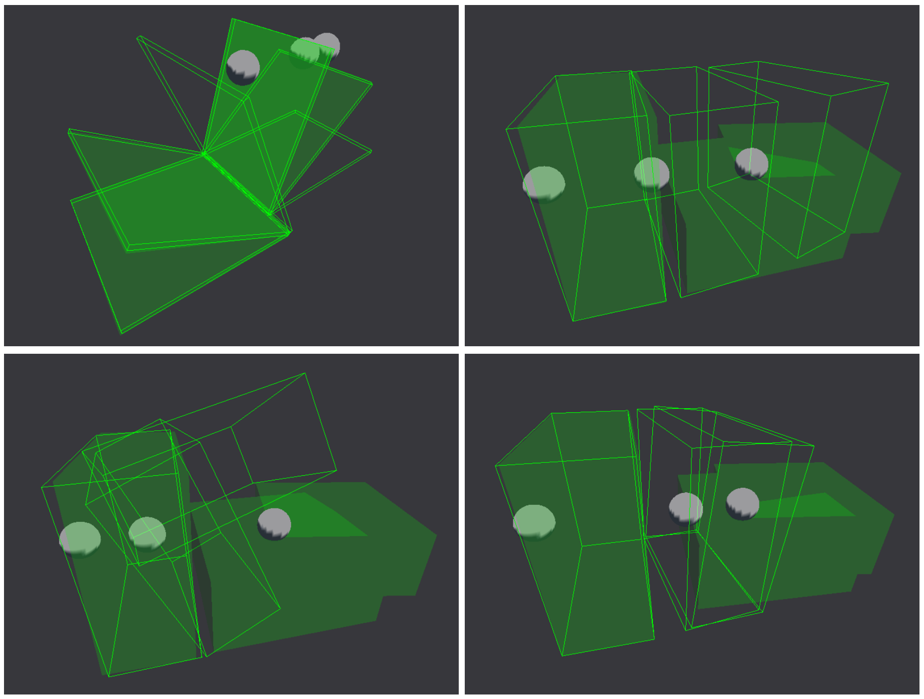


Figure 18: Experiment 4 and 5. An example of shape variation and tipping/toppling error (top left). The multiple expert method trained on a polyflap predicts a simple forward movement of a box (top right). In contrast, the single expert method (bottom left) and the regression method (bottom right) constantly violate impenetrability constraints. The ground-truth and the predicted poses are shown as solid and wire-frame shapes respectively.

In **Experiment 5**, the system is trained on a polyflap, but in testing it is required to predict the motions of a box shaped object. This is a severe test of “extrapolative” generalization.

Similarly as in Experiment 3 the multiple expert method predicts a simple forward movement of a box without violating physics (Figure 18). In contrast the single expert and the regression methods constantly violate physics although they predict some sort of forward movement. As in Experiment 2 the regression method frequently manages to predict qualitative character of a movement such as rotation or toppling.

5.3.2 Interpolative generalization to novel shapes

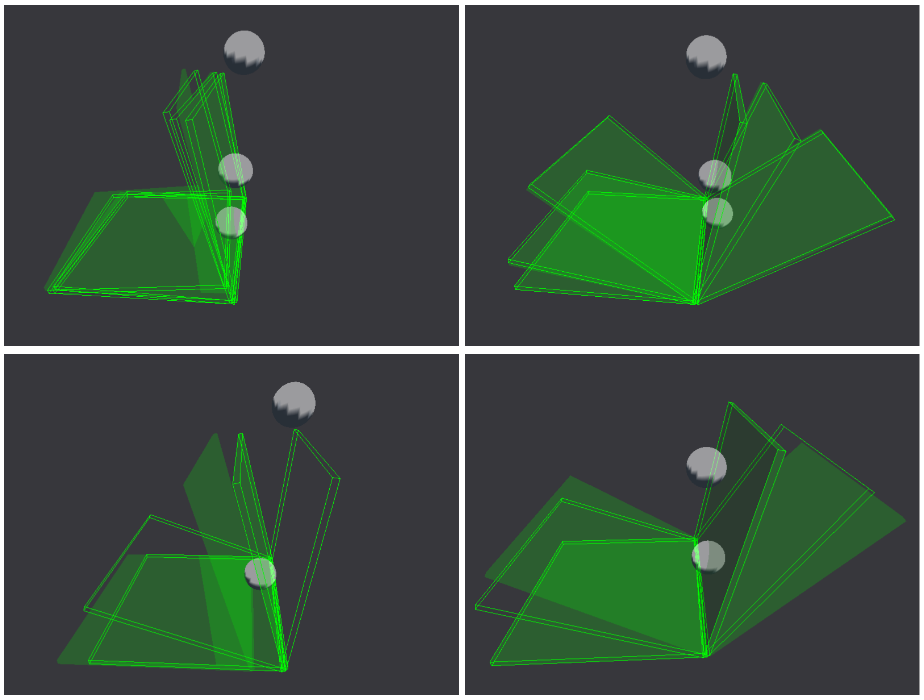
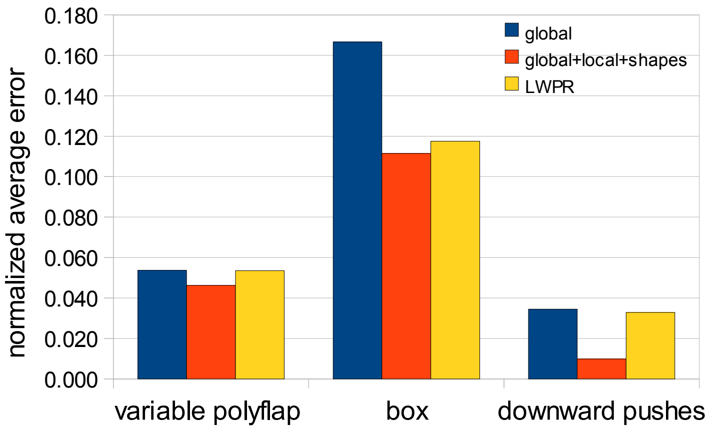


Figure 19: Experiment 6 reveals limitations of the single expert and the regression methods which fail to predict the motion of a polyflap when subjected to a downward push (bottom panel). Both methods constantly predict the same motion of a polyflap. The multiple expert method can cope well with this kind of shape variations (top panel). The ground-truth and the predicted poses are shown as solid and wire-frame shapes respectively.

In **Experiment 6**, all training and testing data involve polyflaps constructed from two square flanges. Shape variation consists in varying the angle at which the two square flanges are connected along a common edge (see Figure 19). The shapes used for training are different from those encountered by the predictors during testing, however we consider this a form of “interpolative” generalization task, in that the test and



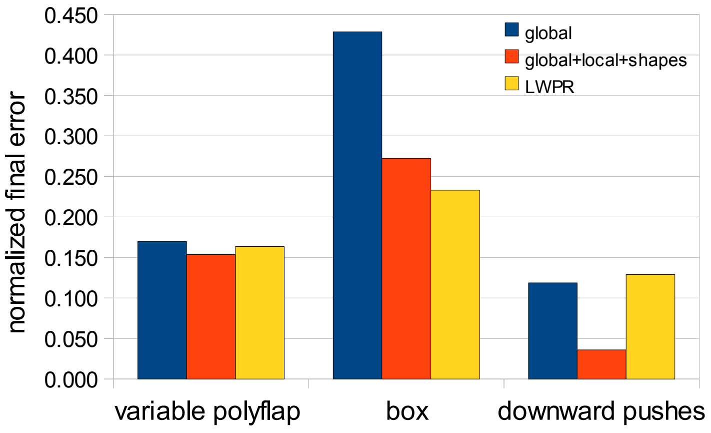


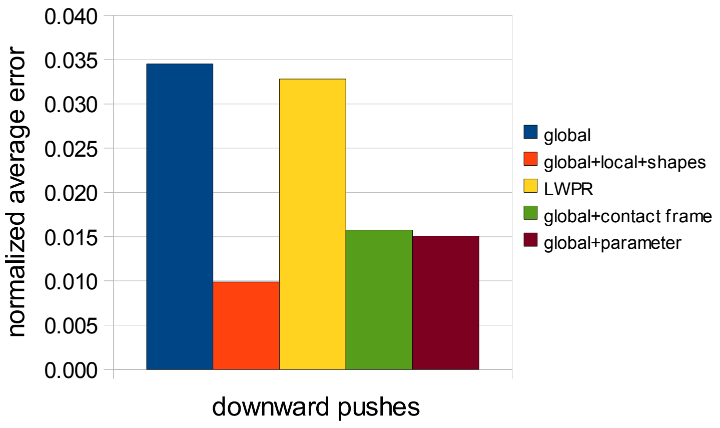
Figure 20: Experiment 4, 5 and 6. Shape generalization errors for a variable polyflap (Experiment 4), a box (Experiment 5) and downward pushes (Experiment 6).

training shapes are qualitatively similar and the range of test shapes can be considered to be spanned by the range of training examples.

This experiment is informative, in that it reveals limitations of the regression method as well as the single expert method. Since the regression technique does not encode information about the object shape variability, it is difficult for it to generalize in situations where small changes in shape can cause significant and qualitative changes in the resulting motion, even when the robotic push is the same. For example, see Figure 19, when subjected to a downwards push, the angle at which the polyflap flanges are joined, determines whether it will be pushed towards the left or towards the right.

In contrast to regression, the product of experts techniques cope much better with this kind of shape generalization.

5.3.3 Incorporating additional shape information into the global expert



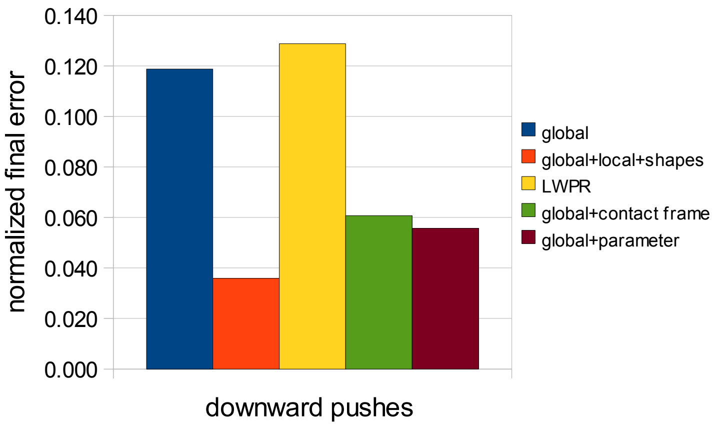


Figure 21: Experiment 7. Prediction errors with two additional single expert methods which incorporate information about a finger-polyflap contact (global + contact frame) and the angle between two flanges of a polyflap (global + parameter).

As we could see in Experiment 6, the single expert method failed because the corresponding conditional density function 6 is not capable to determine whether a polyflap is pushed towards the left or towards the right (Figure 19). We extended Experiment 6 by two predicting methods to test if introducing additional information about contacts and about shape variability can improve prediction performance of the single expert method (see Section 4.3).

The global conditional density function has been modified by incorporating a finger-object contact frame (Equation 13) and information about the angle between two flanges of a polyflap (Equation 14). In both cases the single expert method was able to determine two push cases. As it is shown in Figure 21, the multiple expert method is still a winner mostly because the additional experts manage to incorporate other constraints of the moving polyflap.

In **Experiment 7** the upright flange of a polyflap is randomly shifted along the bottom flange (Figure 23). Because the global frame (B) of a polyflap is attached to the upright flange, the single expert and the regression methods alone have no chance to determine e.g. whether a polyflap should be pushed forward or to be toppled. Not surprisingly, Figure 23 shows that a finger-object contact frame does not improve predictions. However incorporating information about the upright flange shift significantly improves performance over the single expert method. The basic multiple expert method also performs very well preventing in most cases from impossible polyflap movements, despite occasional "qualitative" mistakes (see Figure 22). The multiple expert method with a flange shift information is the best performer in this experiment (Figure 23).

5.4 Experiments with a real robot

In the following experiments, we demonstrate the learned predictors on a real robotic manipulator. A 5-axis Katana robotic manipulator [13] is equipped with a single rigid finger (Figure 26, Figure 27 and Figure 28). It applies pushes to a metal polyflap on a table-top, and the resulting motions are tracked by a model-based vision system [14]. At the time of the experiment, the positioning accuracy of the arm was order +/- 2mm, and the polyflap has been manufactured to a similar level of precision.

The experimental setup (i.e. type and variability of push actions, size and pose of the polyflap) roughly corresponds to the setup of Experiment 1.

5.4.1 Real experimental learning data

In **Experiment 8** we have trained the system on 9, 90 and 900 pushes of a real robot. We evaluated the performance of the multiple and the single expert methods and the regression method. Figure 24 shows that the average and final prediction error decreases with increased number of trials used in learning for all tested prediction methods. All density-based methods performed reasonably well learned with even so little as 9 example pushes. The multiple expert method performs particularly well with 90 learning trials where local experts successfully prevent from violating impenetrability constraints as it is frequently the case with the other tested methods. However, the multiple expert method does not significantly improve its performance with 900 learning trials. Also, none of the tested method achieves the level of performance from Experiment 1. One of the reasons for this is that the vision tracker was not able to provide a sequence of tracked object poses of pose-independent quality. We found that tipping and toppling movements were particularly difficult to track.

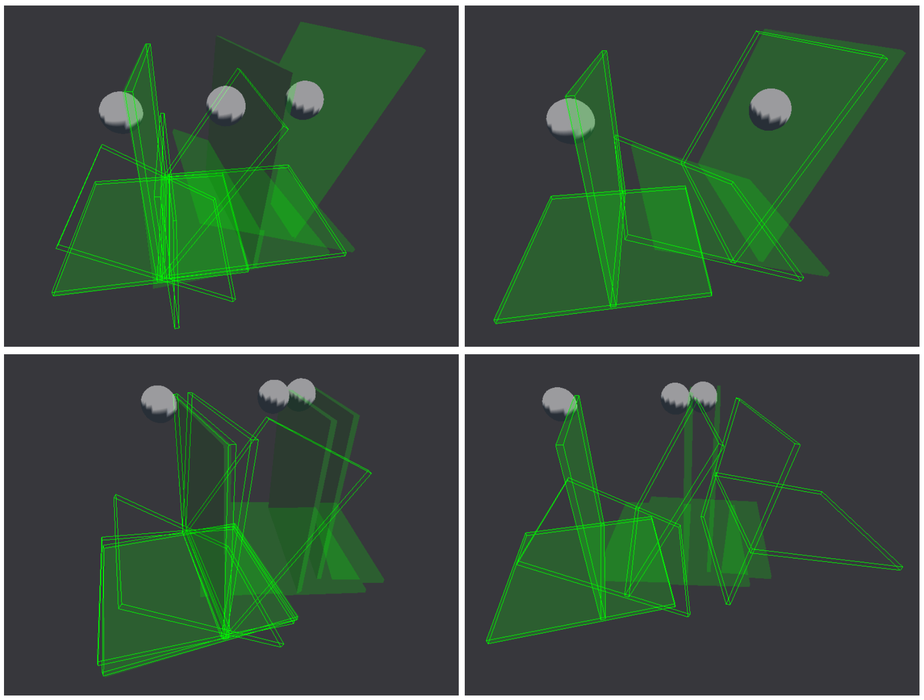
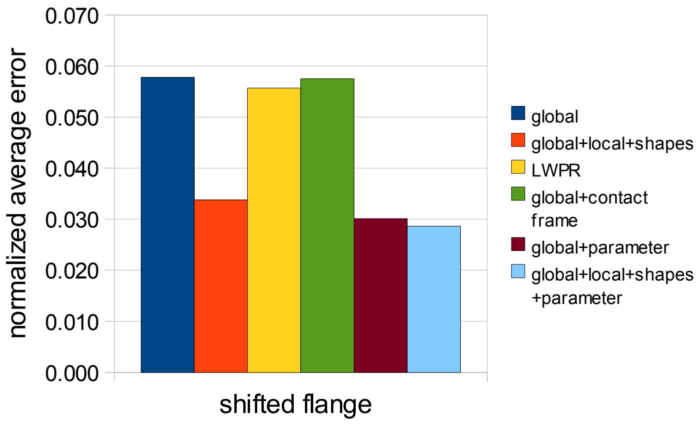


Figure 22: Experiment 7. The basic single expert method frequently generates physically impossible movements (left panel). However, the basic multiple expert method performs very well preventing from impossible polyflap movements, despite occasional "qualitative" mistakes (right panel).

5.4.2 Virtual experimental learning data

In **Experiment 9**, we have trained the system on 900 simulated robot pushes which correspond to those from Experiment 8. Parameters of a simulation were tuned by hand to match the real system. Then we attempted to predict the motion of the real polyflap when it is pushed by the real robot. Figure 26 and Figure 27 show two distinct qualitative behaviours where predictions are convincingly accurate. Figure 28 shows that the predicted behaviour is physically plausible and qualitatively correct, however a quantitative error has resulted from learning on simulated data rather than real test data: while the system has correctly learned to predict the motion of the polyflap within the physics simulator, the physics simulator does not accurately model the frictional interactions of the real world, leading to a discrepancy between the predicted and real polyflap motions in this case.

Figure 25 presents a comparison of prediction methods learned in a real experiment (Experiment 8) and in a virtual experiment (Experiment 9), all with 900 learning trials. Clearly, predictors trained in a simulation are unable to match predictors trained in a real experiment despite some problems with the polyflap pose tracking.



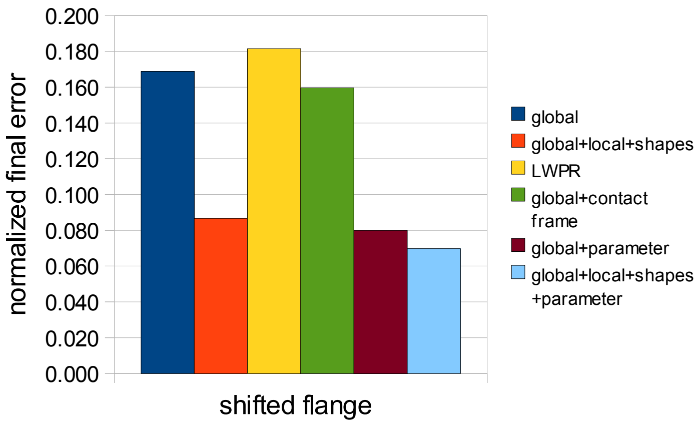
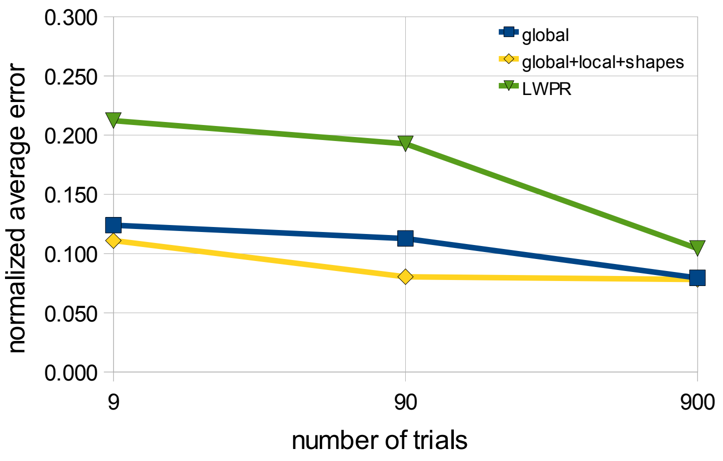


Figure 23: Experiment 7. Prediction errors with three additional methods.

6 Conclusions

We have presented an array of methods for learning to predict the motions of objects during robotic manipulation tasks, that are able to encode physics information without explicitly representing physics knowledge. We show that various geometric relations between parts of objects can be represented as statistically independent *shape/contact experts* - distributions, when used in *products of experts* allow us to generalize over shape and applied actions, as well as to effectively learn in high dimensional spaces.

We have also presented an alternative prediction system based on a state of the art regression method. The regression method and the single expert method show some



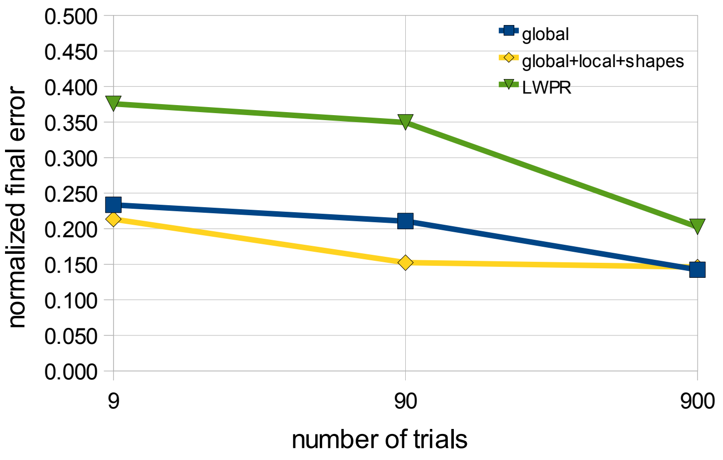
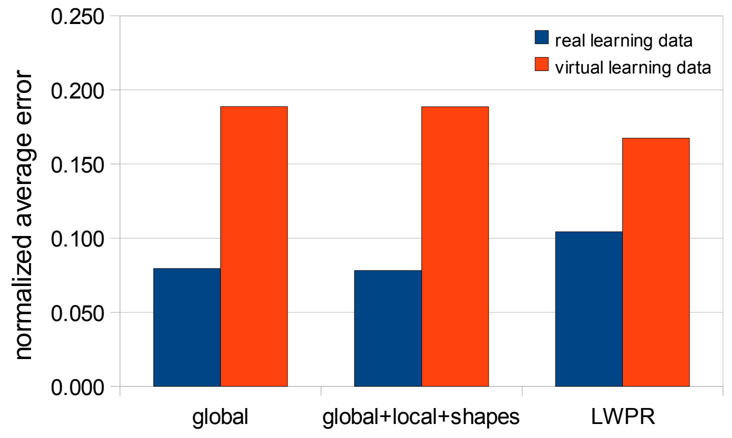


Figure 24: Experiment 8 with a real robot. Decrease in average (top) and final (bottom) prediction errors with increased number of trials used in learning, for three different prediction method.

advantage only when there is low variation in shape and action, but its performance deteriorates where shape variation occurs that substantially affects object behaviour. In contrast, the multiple experts approaches extrapolate comparatively well in such situations.

Furthermore, we have shown that the introduced prediction methods can be successfully used to learn to predict from real experiments with a small number of examples. All methods trained in a real experiment provide substantially better predictions than methods trained in a simulation. The multiple experts method shows the best per-



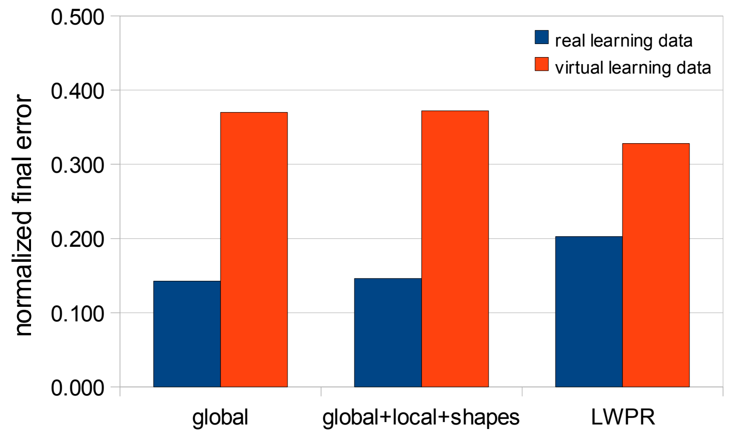


Figure 25: Experiment 8 and 9. Predictors trained in a simulation are unable to match predictors trained in the real experiment.

formance of all other methods, especially when trained with a small number of example pushes.

Future work will investigate:

- Product of experts can incorporate observations during prediction using recursive Bayesian filter.

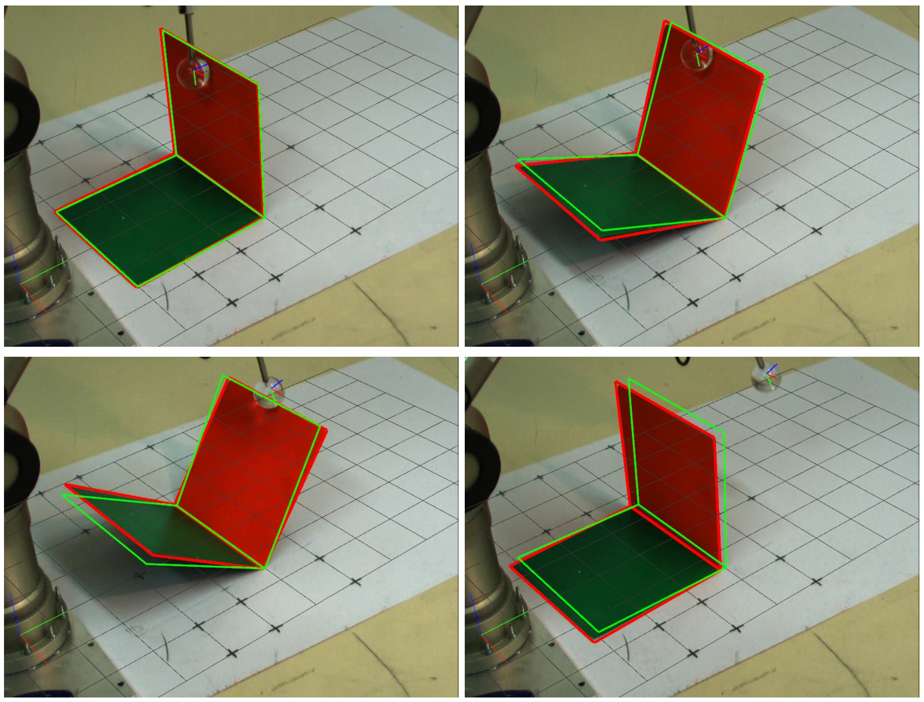


Figure 26: Experiment 8 and 9. Prediction (green wire frame) for a real experiment (red wire frame denotes visual tracking). The polyflap tips forward before rocking back to its starting position. Predictions were generated by the multiple expert method.

- Autonomous selection and construction of experts could allow to select and also construct experts in an autonomous way to provide the best prediction results.
- Shape invariant experts constructed using information about contacts and the centre of mass of an object can be common for a large set of objects.
- Learning to predict in compliant manipulation and grasping which involves motor torques rather than a kinematic type of actions.
- Applying products of experts in inverse models.
- Combining forward and inverse models in modular motor learning.

7 Acknowledgements

We gratefully acknowledge the support for this work of the EC funded FP7 project CogX, number ICT-215181-CogX.

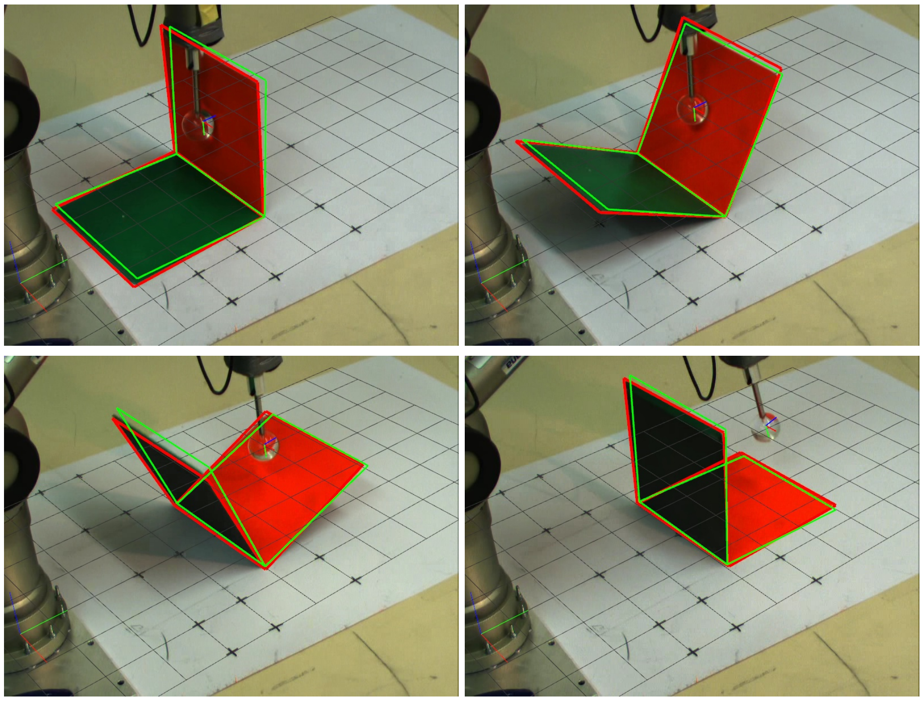


Figure 27: Experiment 8 and 9. Prediction (green wire frame) for a real experiment (red wire frame denotes visual tracking). The polyflap tips forward and then topples over. Predictions were generated by the multiple expert method.

References

[1] M. T. Mason, "Mechanics and planning of manipulator pushing operations," *The International Journal of Robotics Research*, vol. 5, no. 3, p. 53, 1986.

[2] M. A. Peshkin and A. C. Sanderson, "The motion of a pushed, sliding workpiece," *IEEE Journal on Robotics and Automation*, vol. 4, pp. 569–598, 1988.

[3] K. Lynch, "The mechanics of fine manipulation by pushing," in *IEEE International Conference on Robotics and Automation*, pp. 2269–2276, 1992.

[4] D. J. Cappelleri, J. Fink, B. Mukundakrishnan, V. Kumar, and J. C. Trinkle, "Designing open-loop plans for planar micro-manipulation," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 637–642, 2006.

[5] P. Fitzpatrick, G. Metta, L. Natale, S. Rao, and G. Sandini, "Learning about objects through action-initial steps towards artificial cognition," in *IEEE International Conference on Robotics and Automation, 2003. Proceedings. ICRA'03*, vol. 3, 2003.

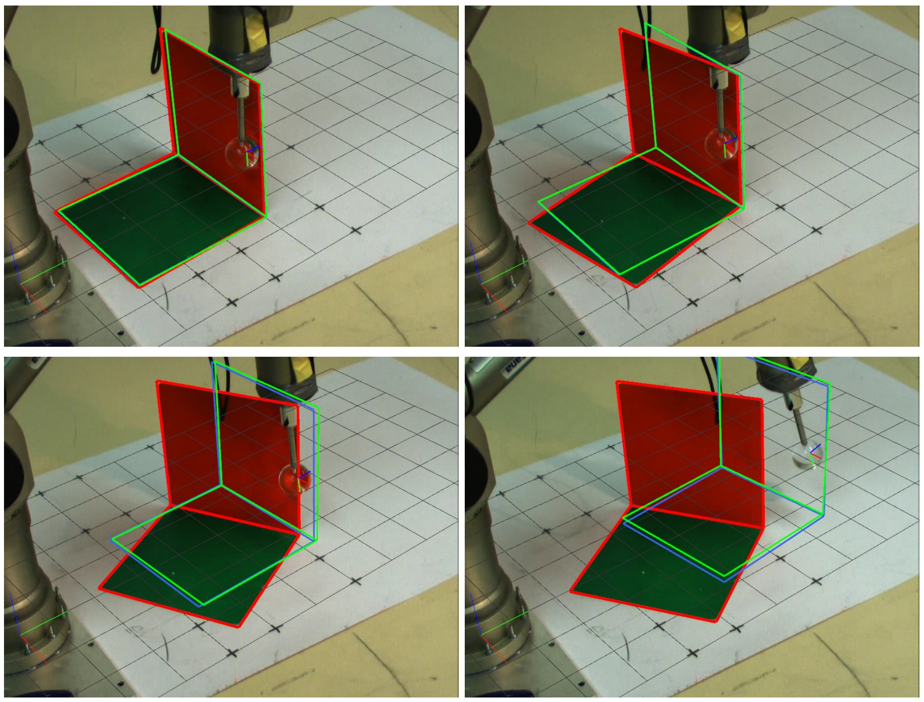


Figure 28: Experiment 8 and 9. Prediction (green wire frame) for a real experiment (red wire frame denotes visual tracking). Blue wire frame denotes prediction of a physics simulator. Note that the learned statistical predictor has correctly learned to replicate the physics simulator data that it was trained on, but fails to properly predict the real object motion because the physics simulator training data does not always correspond well with reality. Predictions were generated by the multiple expert method.

[6] L. Paletta, G. Fritz, F. Kintzler, J. Irran, and G. Dorffner, "Learning to perceive affordances in a framework of developmental embodied cognition," in *IEEE 6th International Conference on Development and Learning, 2007. ICDL 2007*, pp. 110–115, 2007.

[7] B. Ridge, D. Skocaj, and A. Leonardis, "Towards learning basic object affordances from object properties," in *Proceedings of the 2008 International Conference on Cognitive Systems*, (Karlsruhe, Germany), 2008.

[8] M. Kopicki, J. Wyatt, and R. Stolkin, "Prediction learning in robotic pushing manipulation," in *Advanced Robotics, 2009. ICAR 2009. International Conference on*, pp. 1–6, 2009.

[9] D. W. Scott and S. R. Sain, *"Multi-Dimensional Density Estimation"*, pp. 229–263. Elsevier, 2004.

[10] R. Storn and K. Price, "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[11] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., 2006.

[12] N. PhysX, "Physics simulation for developers." http://developer.nvidia.com/object/physx.html, 2009.

[13] Neuronics AG, "Katana user manual and technical description." http://www.neuronics.ch, 2004.

[14] T. Mörwald, M. Zillich, and M. Vincze, "Edge tracking of textured objects with a recursive particle filter," in *Proceedings of the Graphicon 2009*, (Moscow, Russia), 2009.

[15] S. Vijayakumar, A. D'souza, and S. Schaal, "Incremental online learning in high dimensions," *Neural Computation*, vol. 17, no. 12, pp. 2602–2634, 2005.

[16] S. Klanke, S. Vijayakumar, and S. Schaal, "A library for locally weighted projection regression," *The Journal of Machine Learning Research*, vol. 9, pp. 623–626, 2008.

Experiment	**Prediction method**	E_{av}	E_f
1 (1000 trials)	single expert (global)	0.0203	0.0638
	multiple experts (global+local)	0.0226	0.0712
	multiple experts (global+local+shapes)	0.0227	0.0716
	regression (LWPR)	0.0426	0.1326
2	single expert (global)	0.0836	0.2816
	multiple experts (global+local+shapes)	0.0783	0.2266
	regression (LWPR)	0.0607	0.1893
3	single expert (global)	0.1503	0.3669
	multiple experts (global+local+shapes)	0.0141	0.0393
	regression (LWPR)	0.1392	0.3600
4	single expert (global)	0.0537	0.1696
	multiple experts (global+local+shapes)	0.0463	0.1537
	regression (LWPR)	0.0536	0.1636
5	single expert (global)	0.1666	0.4286
	multiple experts (global+local+shapes)	0.1114	0.2722
	regression (LWPR)	0.1175	0.2332
6	single expert (global)	0.0345	0.1188
	multiple experts (global+local+shapes)	0.0098	0.0359
	regression (LWPR)	0.0328	0.1287
	single expert (global+contact frame)	0.0157	0.0607
	single expert (global+parameter)	0.0151	0.0558
7	single expert (global)	0.0578	0.1688
	multiple experts (global+local+shapes)	0.0338	0.0867
	regression (LWPR)	0.0557	0.1815
	single expert (global+contact frame)	0.0575	0.1596
	single expert (global+parameter)	0.0301	0.0799
	multiple experts (global+local+shapes+parameter)	0.0286	0.0697
8 (900 trials)	single expert (global)	0.0794	0.1424
	multiple experts (global+local+shapes)	0.0781	0.1460
	regression (LWPR)	0.1043	0.2026
9	single expert (global)	0.1888	0.3700
	multiple experts (global+local+shapes)	0.1887	0.3721
	regression (LWPR)	0.1675	0.3281

Table 1: Comparative performance of tested prediction methods.

Predicting workpiece motions under pushing manipulations using the principle of minimum energy

Marek Kopicki[1], Rustam Stolkin[1], Sebastian Zurek[1], Thomas Mörwald[2], Jeremy Wyatt[1]
[1]School of Computer Science, University of Birmingham, UK
[2]Automation and Control Institute, Vienna University of Technology, AT

Abstract— **We are investigating the problem of predicting how objects behave under manipulative actions. In particular, we wish to predict the workpiece motions which will result from simple pushing manipulations by a single robotic fingertip. Such interactions are themselves fundamental components of multi-fingered grasping and other complex interactions. Physics simulators can be used to do this, but they model many kinds of object interactions poorly, being dependent on detailed scene descriptions and parameters, which in practice are often difficult to tune. Additionally, we have previously investigated ways of *learning* to predict, by employing density estimation techniques to learn, from many example pushes, a probabilistic mapping between applied pushing motions and resulting workpiece motions. In contrast, this paper presents an alternative approach to prediction, which does not rely on learning but infers the likelihood of possible workpiece motions by using the simple physics principle of minimum energy. This approach is advantageous in situations where insufficient prior knowledge is available for training our learned predictors. In such situations, possible strategies include either training learned predictors on unrealistic simulation data, or making use of the simple physics approach which requires no training. We show that the second of these strategies performs significantly better, and approaches the performance of learned predictors are trained on observations of real object motions.**

I. Introduction

Pushing operations are encountered frequently in robotics, but have received relatively little attention in the research community. Push manipulations are interesting and challenging in that (especially in 3D problems) they provide a large number of unstable positions. They are also important in that push contacts are fundamental to more complex tasks such as grasping [1]. When a two fingered gripper or a multi-fingered hand approaches a grasp configuration, uncertainty (object geometry and pose subject to sensing accuracy, fingertip pose subject to robot accuracy) means that one finger will typically contact the workpiece before the others, resulting a in a single finger pushing phase before a stable grasp is achieved. Furthermore, any grasp is achieved as a combination of pushing forces from the grasping fingers, and in-hand dexterous manipulation motions are essentially the (non-linear) superposition of the effects on a workpiece of pushing motions due to each of the contacting fingers.

Our previous work [2][3] has presented and compared several algorithms which can learn to predict the motions of a rigid object that result from an applied robotic pushing action. These algorithms do not rely on any understanding or encoding of Newtonian mechanics, but can be trained in simple online experiments in which a robot arm applies random pushes to objects of interest and extracts the resulting motions using a vision system. Properties of objects, and their interactions, are learned as distributions. Distributions are important, firstly because they cope with uncertainty of many kinds, and secondly because they enable the opinions of multiple "expert" predictors to be meaningfully combined by a simple product of densities.

This paper presents an alternative approach, in which simple physics principles are used to infer the likelihood of candidate rigid body motions, without the need for learning. This approach is useful, in that it can provide information about the motions of new objects, without having to learn on prior training data for those objects. Furthermore, by expressing the minimum energy principle in terms of a Boltzmann distribution, this simple physics approach can produce, not only a single prediction, but a probability distribution over possible future motions of a workpiece. This means that the opinion of the simple physics predictor can be usefully combined with the opinions of learned density estimators (see above) using the same product of densities scheme. Powerful capabilities for generalization to new objects can now result, by using the simple physics predictor to make overall predictions about gross body motion, while combining with the predictions of learned *local* predictors which have been trained on information about the motions of small parts or surface patches which are common to many objects.

An advantage of the simple physics approach, based on the minimum energy principle, is that it can be applied to previously unencountered objects of arbitrary geometry, and can make relatively robust predictions without exact knowledge of many physical parameters in the scene. In contrast, conventional physics simulation software (e.g. NVIDIA PhysX) might also be applied to these prediction problems, however such techniques are very sensitive to uncertainty in workpiece and scene geometry, and are also dependent on a large number of physical parameters (e.g. frictional constants) which must be very precisely tuned if accurate predictions are to result. In practice, it can be prohibitively difficult or even impossible to tune the parameters of conventional physics simulators such that their predictions match the observed motions of real objects [4]. Furthermore, such simulators make only a single prediction about the future pose of the workpiece, and cannot output a probability distribution over a space of candidate motions. This means that there is no elegant way to combine such physical predictions with our learned predictor techniques, in which we find it useful to combine the opinions of multiple experts as a product of densities.

The simple physics approach does not generally perform as well as a combination of learned expert predictors which have been trained on *real observations of real objects*. However, the advantage of the simple physics predictor is that it can be used to enhance predictions in situations where insufficient prior knowledge or training data are available for training learned predictors. In such situations our alternative options are: firstly train a combination of learned experts on synthetic training data from simulation environments which do not correspond well to the real world; or secondly, replace the "gross body motion" expert (in the product of experts) with an untrained expert based on simple physics. In this paper we show how the second option, making use of simple physics, significantly outperforms the first option, and can bring the performance of a system (equipped with no prior observations of a new object) closer to the ideal situation, in which a combination of learned experts has been trained on a large body of observations of that object.

The paper proceeds as follows. Section II provides an essential overview of our previous work. We first explain how the motions of the workpiece and pushing fingertip are described by coordinate frames and rigid body transformations between these frames, and show how predictors can be learned from many examples of the rigid body transformations that result from applied pushes. We further explain how objects and their motions can be decomposed, by using several different coordinate frames to encode information about the relative motions of small parts or surface patches of objects. We show how the motions of each of these parts can be learned by multiple *local experts*, and how the opinions of these experts can be meaningfully combined as a product of probability densities.

Section III presents the main focus of this paper, which is an additional or alternative approach to predicting the motions of manipulated objects by making use of basic physics principles. We first describe the principle of minimum energy, and then show how future workpiece poses can be computed as those which minimise the work that was done in reaching them. We further show how a Boltzmann distribution can be used to assign probabilities to a distribution over a space of multiple candidate workpiece motions, and how this description in terms of a distribution enables the opinion of the simple physics predictor to be conveniently combined with the opinions of learned predictors as a product of densities.

Section IV presents experimental test results in which a series of pushes are applied to objects and both learned and physics based predictors are tasked with predicting the resulting motions. Performance is evaluated through a combination of virtual experiments in a physics simulator, and real experiments with a 5-axis arm equipped with a simple, rigid finger, and a vision system which can capture the motions of pushed objects.

For a detailed review of the robotic pushing manipulation literature, and a more detailed exposition of our previous work on learning push predictors via density estimation, see [3].

II. Prediction learning

A. Representations

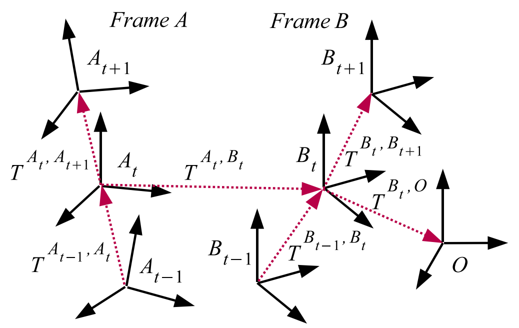


Fig. 1. A system consisting of two interacting bodies with frames A and B in some constant environment with frame O can be described by six rigid body transformations T^{A_t,B_t}, $T^{B_t,O}$, T^{A_{t-1},A_t}, $T^{A_t,A_{t+1}}$, T^{B_{t-1},B_t}, and $T^{B_t,B_{t+1}}$.

Consider three reference frames A, B and O in a 3-dimensional Cartesian space (see Figure 1). While frame O is fixed, A and B change in time and are observed at discrete time steps $..., t-1, t, t+1, ...$ every non-zero Δt. A frame X at time step t is denoted by X^t, a rigid body transformation between a frame X and a frame Y is denoted by $T^{X,Y}$.

From classical mechanics we know that in order to predict a state of a body, it is sufficient to know its mass, velocity and a net force applied to the body. We do not assume any knowledge of the mass and applied forces, however the transformations of a body, with attached frame B, over two time steps T^{B_{t-1},B_t} and $T^{B_t,B_{t+1}}$ encode its acceleration - the effect of the applied net force. Therefore, if the net force and the body mass are constant, the transformations T^{B_{t-1},B_t} and $T^{B_t,B_{t+1}}$ provide a complete description of the state of a body at time step t in absence of other bodies. A triple of transformations $T^{B_t,O}$, T^{B_{t-1},B_t} and $T^{B_t,B_{t+1}}$ provide a complete description of a state of a body in some fixed frame of reference O which accounts for a constant or stationary environment. Similarly, transformations $T^{A_t,O}$, T^{A_{t-1},A_t} and $T^{A_t,A_{t+1}}$ provide such a description for some other body with frame A.

The state of a system consisting of three bodies with frames A and B in some constant environment with frame O can be described by the six transformations as it is shown in Figure 1, where $T^{A_t,O}$ has been replaced by a relative transformation T^{A_t,B_t}. The transformation $T^{B_t,O}$ can be omitted, if the environment does not affect the motion of the bodies or it is explicitly modelled by one of them.

The prediction problem can be stated as: given we know or observe the starting states and the motion of the pusher, $T^{A_t,A_{t+1}}$, predict the resulting motion of the object, $T^{B_t,B_{t+1}}$. This is a problem of finding a function:

$$f : T^{A_t,B_t}, T^{B_t,O}, T^{A_{t-1},A_t}, T^{B_{t-1},B_t}, T^{A_t,A_{t+1}} \rightarrow T^{B_t,B_{t+1}} \tag{1}$$

Function 1 is capable of encoding all possible effects of interactions between rigid bodies A and B, providing their physical properties and applied net forces are constant in time. Furthermore, it can be learned purely from observations for some fixed time delta Δt.

In many robotic operations, manipulations are slow, we can assume quasi-static conditions, and it is often possible to ignore all frames at time $t-1$. This conveniently reduces the dimensionality of the problem, giving:

$$f : T^{A_t,B_t}, T^{B_t,O}, T^{A_t,A_{t+1}} \rightarrow T^{B_t,B_{t+1}} \tag{2}$$

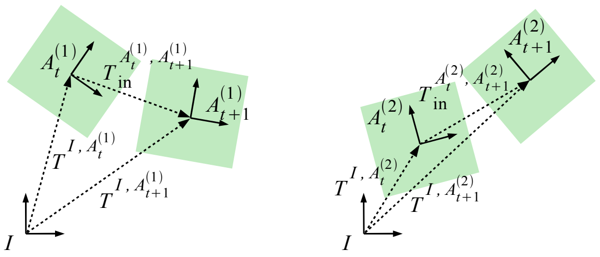


Fig. 2. In the above two scenes a pose change between time step t and $t+1$ as observed in instantaneous object body frame $A^{(1)}$ and the same object in another instantaneous body frame $A^{(2)}$ given inertial frame I are both the same. However because transformations $T^{I,A^{(1)}}$ and $T^{I,A^{(2)}}$ are different, the corresponding transformations in the inertial frame are also different, i.e. $T_{in}^{A_t^{(1)},A_{t+1}^{(1)}} \neq T_{in}^{A_t^{(2)},A_{t+1}^{(2)}}$.

We expect that the behaviour of interacting bodies represented by rigid body transformations as in Figure 1 shares some statistical similarities *independently* on their global poses with respect to some current inertial frame I [3]. Instead of using inertial frame-dependent transformation $T_{in}^{A_t,A_{t+1}}$, one can represent object transformations as observed in the object body frame (see Figure 2). Body frame transformation $T_{body}^{A_t,A_{t+1}}$ is obtained by moving instantaneous frame A, so that at time t it overlaps with inertial frame I. Given some instantaneous object frame A_t at time t, transformation $T_{in}^{A_t,A_{t+1}}$ and because $T^{I,A_{t+1}} = T_{in}^{A_t,A_{t+1}} T^{I,A_t} = T^{I,A_t} T_{body}^{A_t,A_{t+1}}$, one can obtain transformation $T_{body}^{A_t,A_{t+1}}$ in the body frame as follows:

$$T_{body}^{A_t,A_{t+1}} = (T^{I,A_t})^{-1} T_{in}^{A_t,A_{t+1}} T^{I,A_t} \tag{3}$$

Similarly from a given transformation in body frame, instantaneous object frame A_t at t and using Equation 3, one can obtain expression for transformation $T_{in}^{A_t,A_{t+1}}$ in the inertial frame

$$T_{in}^{A_t,A_{t+1}} = T^{I,A_t} T_{body}^{A_t,A_{t+1}} (T^{I,A_t})^{-1} \tag{4}$$

In further discussion we will keep subscripts in while dropping subscripts $body$ assuming that all transformations $T^{X,Y}$ are transformations in the body frame X obtained from $T^{X,Y} \equiv T_{body}^{X,Y} = (T^{I,X})^{-1} T_{in}^{X,Y} T^{I,X}$.

B. Learning global and local experts as density estimation

Prediction learning with using Functions 1 or 2 is limited with respect to changes in shape [3]. Consider two objects lying on a table top. Figure 3 shows two situations that are identical except for the shape of object A. It is clear that the same transformation of A's position will lead to different motions for object B in each case. How can we encode the way in which the shapes of A and B alter the way they behave? We use a product of several densities to approximate the density over the rigid body transformation given in the function 2.

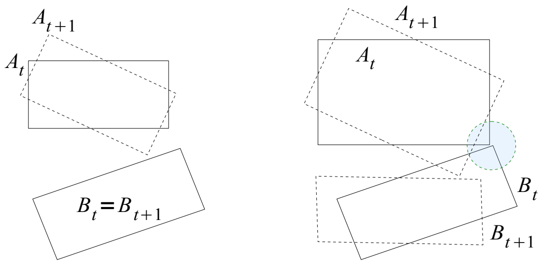


Fig. 3. Two scenes, each with two objects on a table top, viewed from above. Between the two scenes only the shape of A is different. Yet when A moves the resulting transformation $T^{B_t,B_{t+1}}$ will be quite different. This shows that our predictors must take some aspect of the shape of A and B into account.

In the simplest case one can approximate two densities, conditioned on local and global information respectively [3]. We define the global information to be the information about changes of the pose of the whole object. The local information is specified by changes of the pose of the surfaces of A and B at the contact point, or the point of closest proximity, between the object and the finger. We model this local shape as a pair of planar surface patches, of limited extent (see Figure 4).

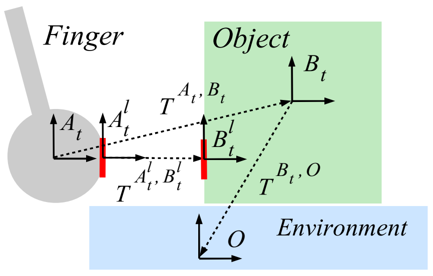


Fig. 4. 2D projection at time t of a robotic finger with global frame A_t, an object with global frame B_t, and a ground plane with constant global frame O. Local frames A_t^l and B_t^l describe the local shape of the finger and an object at their point of closest proximity.

Consider a 2D projection at time t of a robotic finger with global frame A_t, an object with global frame B_t, and a ground plane with constant global frame O (Figure 4). Similarly, local frames A_l^t and B_t^l describe local shapes belonging to a finger and an object. We define a global conditional density function as [3]:

$$p_{global}(T^{B_t,B_{t+1}} | T^{A_t,A_{t+1}}, T^{A_t,B_t}, T^{B_t,O}) \tag{5}$$

and similarly a local conditional density function as:

$$p_{local}(T^{B^l_t,B^l_{t+1}}|T^{A^l_t,A^l_{t+1}},T^{A^l_t,B^l_t}) \quad (6$$

To predict the rigid body transformation of an object whe
it is in contact with others we are faced with how to represen
the constraints on motion provided by the contacts. We do thi
using a product of experts [3]. The experts represent by densit
estimation which rigid body transforms are (in)feasible fo
each frame of reference. In the product, only transformation
which are feasible in both frames will have high probability

The only problem is to find relations between transfor
mations in the body frame of the local shapes and th
corresponding transformations in the inertial frames. For a particular situation shown in Figure 4 from object rigidity and using Equation 3 we have:

$$T^{A^l_t,A^l_{t+1}} = (T^{I,A^l_t})^{-1}T_{in}^{A_t,A_{t+1}}T^{I,A^l_t} \quad (7a)$$

$$T^{B^l_t,B^l_{t+1}} = (T^{I,B^l_t})^{-1}T_{in}^{B_t,B_{t+1}}T^{I,B^l_t} \quad (7b)$$

where I is the inertial frame. $T^{A^l_t,B^l_t}$ can be determined directly from the shape frame:

$$T^{A^l_t,B^l_t} = (T^{I,A^l_t})^{-1}T_{in}^{A^l_t,B^l_t}T^{I,A^l_t} \quad (8)$$

For the finger-object scenario a prediction problem can then be defined as finding that transformation $T_{in}^{B_t,B_{t+1}}$ in the inertial frame which maximizes the product of the two conditional densities (experts) 5 and 6:

$$\max_{T_{in}^{B_t,B_{t+1}}} p_{local}((T^{I,B^l_t})^{-1}T_{in}^{B_t,B_{t+1}}T^{I,B^l_t}|T^{A_t,A_{t+1}},T^{A^l_t,B^l_t})\times$$
$$p_{global}((T^{I,B_t})^{-1}T_{in}^{B_t,B_{t+1}}T^{I,B_t}|T^{A_t,A_{t+1}},T^{A_t,B_t},T^{B_t,O}) \quad (9)$$

Starting with some initial state of the finger T^{A_0} and the object T^{B_0}, and knowing a trajectory of the finger $A_1,\dots A_N$ over T time steps, one can now predict a whole trajectory of an object $B_1,\dots B_N$ by sequentially solving a problem of maximization of the product 9.

C. Incorporating information from additional experts

In addition to learning how an object moves in response to a push, it is desirable if we can also incorporate learned information about the inherent tendencies of parts of an object to move in various directions with respect to the environment or any other objects, but regardless of whether it is being pushed or not. This additional information may help when predicting the motions of previously unseen objects, because it provides some prior knowledge about what kinds of motions are possible and which are not.

We can incorporate this additional information by attaching an arbitrary number of additional coordinate frames B^{sn_t} to various parts of the object. We then learn densities for the future motions of each of these frames, conditioned only on their relative pose $T^{E^{S^k_t},B^{S^k_t}}$ with respect to a corresponding

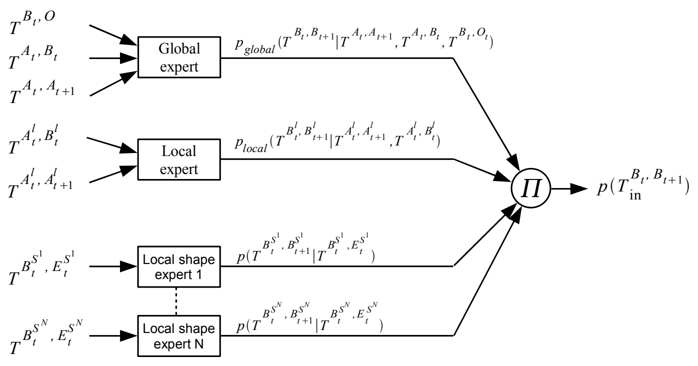

$T^{B_t,O}$
T^{A_t,B_t}
$T^{A_t,A_{t+1}}$
Global expert
$p_{global}(T^{B_t,B_{t+1}}|T^{A_t,A_{t+1}},T^{A_t,B_t},T^{B_t,O_t})$
$T^{A^l_t,B^l_t}$
$T^{A^l_t,A^l_{t+1}}$
Local expert
$p_{local}(T^{B^l_t,B^l_{t+1}}|T^{A^l_t,A^l_{t+1}},T^{A^l_t,B^l_t})$
Π
$p(T_{in}^{B_t,B_{t+1}})$
$T^{B^{S^1}_t},E^{S^1}_t$
Local shape expert 1
$p(T^{B^{S^1}_t,B^{S^1}_{t+1}}|T^{B^{S^1}_t,E^{S^1}_t})$
$T^{B^{S^N}_t},E^{S^N}_t$
Local shape expert N
$p(T^{B^{S^N}_t,B^{S^N}_{t+1}}|T^{B^{S^N}_t,E^{S^N}_t})$

Fig. 5. Inputs and outputs of learned prediction system. The 2-expert approach can be extended to include opinions from multiple local shape experts represented by coordinate frames S^N.

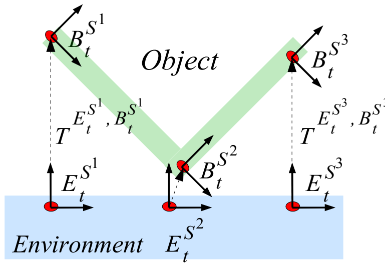


Fig. 6. Co-ordinate frames can be attached to an arbitrary number of local shapes, and local experts can be learned for each of these frames, predicting a distribution of how the frame may move next, given where it is at the present time step.

pose $E^{S^k_t}$ of a patch on a ground plane at the present time step, ignoring any information about the motions of the pushing finger. For the k-th such frame, we estimate *the local contact conditional density*:

$$p(T^{B^{S^k_t},B^{S^k_{t+1}}}|T^{E^{S^k_t},B^{S^k_t}}) \quad (10)$$

which represent probability density over possible rigid body transformations in the body frame of the k-th local contact. The subsequent motion of the object in the inertial frame can now be predicted as:

$$\max_{T_{in}^{B_t,B_{t+1}}} p_{local}((T^{I,B^l_t})^{-1}T_{in}^{B_t,B_{t+1}}T^{I,B^l_t}|T^{A_t,A_{t+1}},T^{A^l_t,B^l_t})\times$$
$$p_{global}((T^{I,B_t})^{-1}T_{in}^{B_t,B_{t+1}}T^{I,B_t}|T^{A_t,A_{t+1}},T^{A_t,B_t},T^{B_t,O})\times$$
$$\prod_{k=1\dots N} p((T^{I,B^{S^k_t}})^{-1}T_{in}^{B_t,B_{t+1}}T^{I,B^{S^k_t}}|T^{E^{S^k_t},B^{S^k_t}}) \quad (11)$$

All joint and conditional densities are approximated by a variant of kernel density method with Gaussian kernels described in details in [2]. For simplicity, the density product 11 is maximised using the differential evolution optimization algorithm [6].

III. Simplified physics approach

A. Principle of minimum energy

The previous section presented a set of methods for learning to predict the behaviour of objects in simple robotic manipulation tasks. The methods incorporate information about objects' shapes and other physical properties in terms of distributions. The local distributions encode information about the behaviour of objects' local shape parts during interactions and can be shared among many objects. However, the global distribution is unique to a particular object or object category, therefore the generalization capabilities of such global distributions are limited, in particular with respect to objects of different shapes.

A simplified physics approach is an alternative method for predicting the motion of an object subjected to pushing action. The approach relies on the *principle of minimum energy* known from thermodynamics as a consequence the *second law of thermodynamics* applied to *closed systems*. The principle of minimum energy states that the total energy of a closed system decreases and reaches a local minimum value at equilibrium, where a closed system is a system with fixed entropy and other parameters such as volume or mass, but which can exchange energy with other connected systems [5].

A system consisting of a robot, an object and a ground plane can also be considered as a closed system. From the principle of minimum energy we know that the total energy of our system must reach a local minimum for a given amount of work introduced to the system. Each movement of a robotic finger, if it touches an object, produces some amount of work, which in the prediction scenario is unknown because the corresponding movement of the object is unknown. However, this movement can be computed by searching for such movements which minimize the produced amount of work, given known physical properties of the system.

A simplified physics approach uses a very simple model of physical interactions, which can be split into the physical phenomena and the corresponding work done by moving objects as follows:

1) *Mass* via work done by accelerating a given object.
2) *Gravity force* via work done while moving in a given potential field.
3) *Friction* via work done by two objects in contact moving in tangential direction. It is the simplest case of Coulomb's law of sliding friction with dynamic friction only.
4) *Restitution* via work done by two objects in contact moving in directions normal to the contacting surfaces.

B. Finding a trajectory at equilibrium

The simplified physics approach represents the object body by a set of N "volumetric" particles v_t^i with index i at discrete time step t randomly generated at time step $t = 0$ and then rigidly attached to the object throughout all prediction time steps (see Figure 7). Trajectory of an object is approximated by a sequence of rigid body transformations q which are found

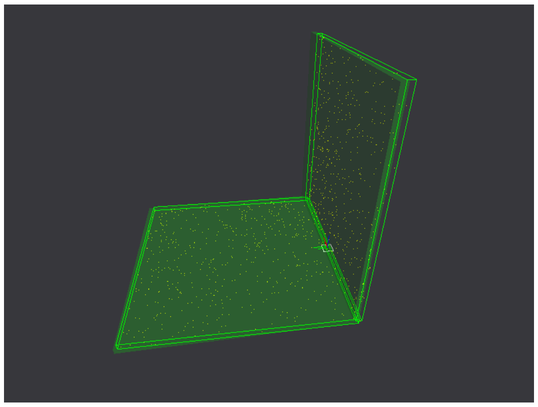


Fig. 7. A set of "volumetric" particles (yellow dots) representing the object body (green solid shape).

by solving a problem of minimizing the energy function $E(q)$ at each time step $t = 2, \dots, T$:

$$\min_q E(q,t) \tag{12}$$

Energy function $E(q)$ consists of four work type-specific functions which correspond to four ways of producing work as described in the previous section:

$$E(q,t) = E^a(q,t) + E_i^g(q,t) + E_i^f(q,t) + E_i^r(q,t) \tag{13}$$

where each work function computes work during movement generated by q as follows[1]:

$$E^a(q,t) = C_a \| \sum_{i=1}^{N} (qv_{t-1}^i - 2v_{t-1}^i + v_{t-2}^i) \| \tag{14}$$

$$E_i^g(q,t) = -C_g \sum_{i=1}^{N} G \cdot (qv_{t-1}^i - v_{t-1}^i) \tag{15}$$

$$E_i^f(q,t) = C_f \sum_{i \in V_f} \| qv_{t-1}^i - v_{t-1}^i \| \tag{16}$$

$$E_i^r(q,t) = C_r \sum_{i \in V_r} d_i(qv_{t-1}^i) \tag{17}$$

where $C_* \in \mathbb{R}^+$ are work type-specific constants, $G \in \mathbb{R}^3$ is the gravity vector, V_f is an index set of all particles which are in contact with the ground plane, V_r is an index set of all particles which penetrate a robotic finger or the ground plane with the corresponding penetration depth d_i.

Transformation q which minimizes $E(q)$ can be computed using e.g. a differential evolution optimization algorithm [6].

C. Probability density over trajectories

Energies $E(q)$ can be transformed into a probability density function over possible transformations q by using a Boltzmann distribution [5]:

[1] Work functions are only a crude approximation of real physical phenomena and do not even preserve physical units.

$$p_{Boltzmann}(E(q)) = \frac{\exp\left(-\frac{E(q)}{kT}\right)}{Z(T)} \tag{18}$$

where k is Boltzmann constant and T is temperature. $Z(T)$ is a partition function (a normalization constant) which for a given temperature can be computed from:

$$Z(T) = \sum_q \exp\left(-\frac{E(q)}{kT}\right) \tag{19}$$

Because a basic prediction scenario requires computation of only the most likely trajectory, normalization constant $Z(T)$ need not to be estimated and can be assumed a non-zero constant.

$p_{Boltzmann}(E(q))$ can be used as an approximation of the global conditional density function given by Equation 5 and it can be combined in a product with other experts as was discussed in the previous section. The global conditional density function can be replaced with $p_{Boltzmann}(E(q))$ so that Equation 11 now becomes:

$$\begin{aligned}\max_{T_{in}^{B_t,B_{t+1}}} & p_{Boltzmann}(E(T_{in}^{B_t,B_{t+1}}))\times \\ & p_{local}((T^{I,B_t^l})^{-1}T_{in}^{B_t,B_{t+1}}T^{I,B_t^l}|T^{A_t,A_{t+1}},T^{A_t^l,B_t^l})\times \\ & \prod_{k=1\dots N} p((T^{I,B^{S_t^k}})^{-1}T_{in}^{B_t,B_{t+1}}T^{I,B^{S_t^k}}|T^{E^{S_t^k},B^{S_t^k}})\end{aligned} \tag{20}$$

where symbol T stands for a rigid body transformation. The predicted object motion is a transformation $T_{in}^{B_t,B_{t+1}}$ which maximises the value of the above product.

$p_{Boltzmann}(E(q))$ depends on several constants which have to be estimated for a particular system, but crucially it also depends on temperature T. When temperature $T \to \infty$, $p_{Boltzmann}(E(q)) \to 1$ for any transformation q, consequently $p_{Boltzmann}(E(q))$ has no influence on a result of the maximization procedure 20. On the other hand, when temperature $T \to 0$, $p_{Boltzmann}(E(q))$ becomes very rugged, likely with a single peak only, so that the other factors in the product 20 have no impact on the maximization result.

IV. Results

A. Experimental setup

We have tested the introduced prediction algorithms in simulation experiments using PhysX physics engine [7], and in real experiments using 5-axis Katana robotic manipulator [8] equipped with a single rigid finger. We capture the motion of an object using a vision tracking system [9].

Multiple experimental trials were performed, in which a robotic arm equipped with a finger performs a random pushing movement of length approximately 25 cm towards an object placed at a random initial pose (Figure 8). In each experiment data samples are stored over a series of such random trials. Each trial lasts 10 seconds, while data samples are stored every $1/15$th of a second.

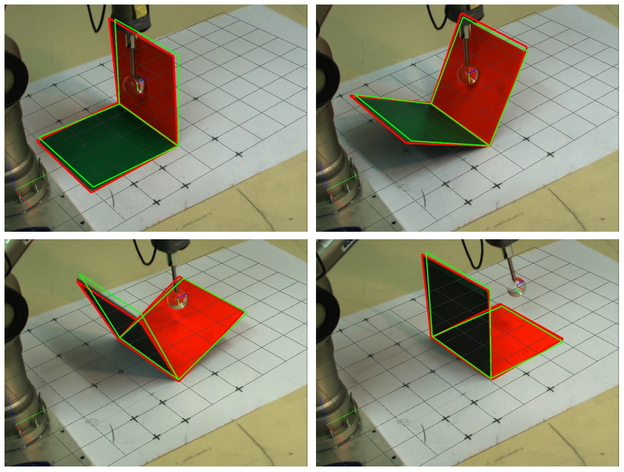


Fig. 8. A 5-DOF robotic arm equipped with a finger performs forward movements towards an object. Object behaviour varies depending on the initial object pose and finger trajectory. An example image sequence shows toppling behaviour. Orange wire frame denotes output of the vision based tracking system. Green wire frame shows predictions made by a simplified physics - note that the entire motion sequence is predicted before the physical push is initiated, without any recursive correction from visual feedback during the push execution.

B. Performance measure

In all experiments, we take the output of the tracked pose of a real object to be ground-truth, and compare it against predictions forecast by the simplified physics approach (Section III) or by the learned approaches (Section II). Prediction performance is evaluated as follows.

At any particular time step, t, a large number, N, of randomly chosen points $p_n^{1,t}$, where $n = 1 \dots N$, are rigidly attached to an object at the ground-truth pose, and the corresponding points $p_n^{2,t}$ to an object at the predicted pose. At time step t, an average error E_t can now be defined as the mean of displacements between points on the object at the predicted pose and points on the object at the ground-truth pose:

$$E_t = \frac{1}{N}\sum_{n=1\dots N}|p_n^{2,t} - p_n^{1,t}| \tag{21}$$

Note that for each robotic push action, we predict approximately 150 consecutive steps into the future, with no recursive filtering or corrector steps, hence it is expected that errors will grow with range from the initial object pose. We therefore find it more meaningful to normalize all errors with respect to an "average range", R_t, of the object from its starting position, defined as:

$$R_t = \frac{1}{N}\sum_{n=1\dots N}|p_n^{1,t} - p_n^{1,0}| \tag{22}$$

For a test data set, consisting of K robotic pushes, each of which breaks down into many consecutive predictions over T time steps, we can now define an *normalized average error*:

$$E_{av} = \frac{1}{K}\sum_{k=1\dots K}\frac{1}{T}\sum_{t=1\dots T}\frac{E_t}{R_t} \tag{23}$$

For each set of test data, we also report an *normalized final error*, E_f which represents the typical discrepancy between prediction and ground truth that has accumulated by the end of each full robotic push:

$$E_f = \frac{1}{K} \sum_{k=1 \ldots K} \frac{|p_n^{2,T} - p_n^{1,T}|}{R_T} \tag{24}$$

We performed 10-fold cross-validation where at the beginning of each experiment all the trials are randomly partitioned into 10 subsets. Prediction was then subsequently performed (10 times) on each single subset, while learning (only for learned approaches) was always performed on the remaining 9 subsets of these trials. All the results were then averaged to produce a single estimation.

C. *Performance of a simplified physics approach*

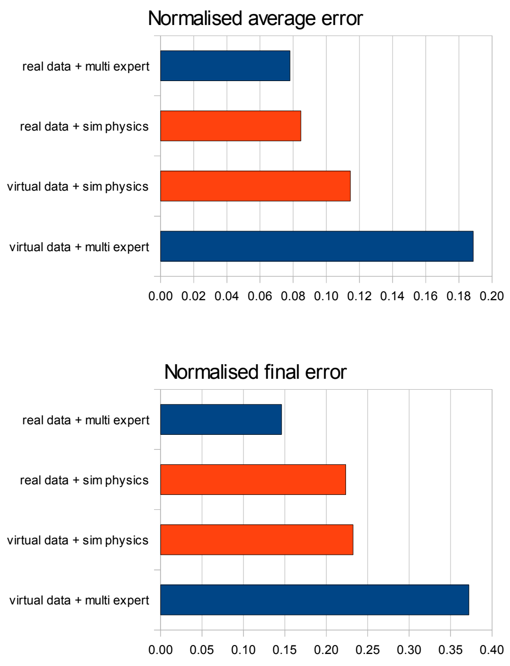


Fig. 9. If only simulation data is available for training of experts, then incorporating a simple physics predictor improves performance, approaching that of the ideal situation in which experts are trained on real observations of real objects. These charts compare the performance of a combination of learned experts which have been trained only on erroneous synthetic data, with and without the incorporation of an additional expert based on simple physics. In each case, performance has been assessed by then attempting to make predictions about motions of real objects which were not seen during training. For comparison, we also show data for the ideal situation in which experts have been exposed to examples of the real object during training.

Here we are interested in seeing how well our prediction systems can do, when no examples of captured real object motion are available for training, and instead we must rely on synthetic training data generated by simulation environments which do not correspond well with the real world. In all experiments, the various prediction approaches are tested by trying to make predictions about real objects being pushed by a 5-axis robot arm, and the predicted motions are compared against those captured by a vision system. We first train a combination of learned experts on synthetic push sequences, and then test by trying to predict the real motions of real objects being pushed by the robot. The resulting errors are shown in the bottom bar (virtual data + multi expert) in the charts of Figure 9. We next show that, by replacing one of the learned experts (the global or "gross body motion" expert) with an untrained simple physics predictor, we can significantly improve the predictions made about real objects. This result is represented by the second bar from the bottom in each chart (virtual data + sim physics). For comparison, we show the ideal situation (real data + multi expert) where a combination of learned experts has been trained on a large number of observed trajectories of real objects subject to real robot pushes.

If the reader compares the top bars of Figure 9 (real data + multi expert) against the second from top (real data + sim physics), it will be noted that simplified physics *does not perform as well* as a combination of purely learned predictors in cases where plenty of real-world observations are available for training. The advantage of the simplified physics contribution is that, in situations where prior experience of a real object is limited (e.g. when a robot encounters a new object), then a simplified physics contribution can improve on the performance of learned predictors that are merely trained on synthesized data from erroneous physics simulators.

Figure 8 and 10 shows some examples of successful predictions made by simplified physics. The toppling behaviour from Figure 8 is also correctly predicted by NVIDIA PhysX [7] game physics simulator. However PhysX struggles to provide correct predictions of sliding motion which involves large amount of rotation as it is visible in Figure 11. Similar rotational movements are reasonably well predicted by our simplified physics approach as it is shown in Figure 10.

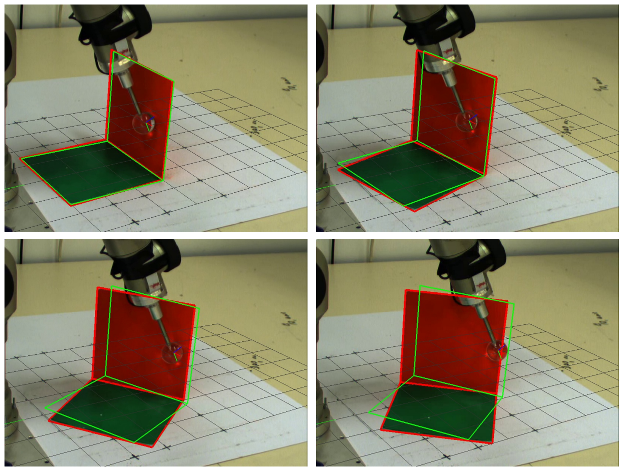


Fig. 10. An example image sequence shows sliding behaviour with large amount of rotational movement. Orange wire frame denotes output of the vision based tracking system. Green wire frame shows predictions made by simplified physics predictor.

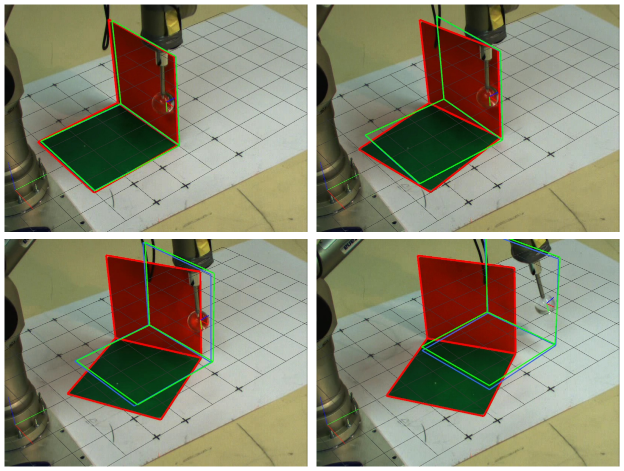


Fig. 11. An example image sequence shows sliding behaviour with large amount of rotational movement. Orange wire frame denotes output of the vision based tracking system. Green wire frame shows erroneous predictions made by predictor trained on virtual data provided by NVIDIA PhysX.

V. Conclusions

We have developed a number of methods for predicting the motions of manipulated rigid bodies, and we have also developed ways of combining these methods as a product of experts. Conventional physics simulators are often inadequate for making useful predictions about the interactions of real objects, and in many cases we find that learned predictors, trained on multiple example motions, perform much better. Unfortunately, it is not possible to use learned predictors for objects for which no prior training data is available, for example when a robot encounters a new object that it has not seen before. In such circumstances, one could attempt to train predictors on synthetic data, generated by a physics simulator, but the performance will be poor because such data is a poor representation of reality. this paper has shown that, in such situations, substantial advantage can be gained by incorporating (into the combination of experts) an expert that is not trained, but which infers the likelihoods of workpiece motions by applying the simple physics principle of minimum energy.

A useful property of the simple physics predictor is that it does not merely predict a single future object pose. Instead, by expressing the minimum energy principle in terms of a Boltzmann distribution, it is possible to predict an entire probability distribution over the space of possible candidate object motions. This is useful because it enables the opinions of the simple physics predictor to be combined with those of learned predictors via a simple product of densities approach. It is also useful in that, by supplying probabilities for candidate object poses, this can be used in predictor-corrector type recursive estimation problems such as vision-based tracking of manipulated objects using particle filters.

In the present work, predictions are made in advance for an entire 10 second push sequence, before the push is made, without any corrector or update steps from sensory inputs or recursive filtering. Ongoing work is exploring the use of these predictions as part of a predictor-corrector recursive estimation system for online visual tracking of manipulated objects. Since visual tracking data is necessary for training learned predictors, and the learned predictors may be useful for enhancing tracking, a bootstrapping problem is suggested for which the simple physics approach of this paper may prove a useful ingredient - the simple physics predictor may be used to enhance tracking, until sufficient data can be acquired to train a superior set of learned predictors.

VI. Acknowledgments

We gratefully acknowledge support for this research from EU-FP7-IST grants Nos 248273 (GeRT) and 215181 (CogX).

References

[1] M. T. Mason, *Mechanics of robotic manipulation*. MIT press, 2001.

[2] M. Kopicki, J. Wyatt, and R. Stolkin, “Prediction learning in robotic pushing manipulation,” in *Advanced Robotics, 2009. ICAR 2009. International Conference on*, pp. 1–6, 2009.

[3] M. Kopicki, *Prediction learning in robotic manipulation*. PhD thesis, University of Birmingham, 2010.

[4] D. J. Duff, J. Wyatt, and R. Stolkin, “Motion estimation using physical simulation,” in *IEEE International Conference on Robotics and Automation*, (Alaska), IEEE, May 2010. To be published.

[5] L. D. Landau and E. M. Lifshitz, *Statistical Physics, Part 1*, vol. 5 of *Course of Theoretical Physics*. Elsevier Butterworth-Heinemann, third ed., 1980.

[6] R. Storn and K. Price, “Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces,” *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[7] N. PhysX, “Physics simulation for developers.” http://developer.nvidia.com/object/physx.html, 2009.

[8] Neuronics AG, “Katana user manual and technical description.” http://www.neuronics.ch, 2004.

[9] T. Mörwald, M. Zillich, and M. Vincze, “Edge tracking of textured objects with a recursive particle filter,” in *Proceedings of the Graphicon 2009*, (Moscow, Russia), 2009.

Motion estimation using physical simulation

Damien Jade Duff, Jeremy Wyatt, and Rustam Stolkin
{D.J.Duff, J.L.Wyatt, R.Stolkin}@cs.bham.ac.uk

Abstract— **We consider the task of monocular visual motion estimation from video image sequences. We hypothesise that performance on the task can be improved by incorporating an understanding of physically likely and feasible object dynamics. We test this hypothesis by incorporating a physical simulator into a least-squares estimation procedure. We initialise a full trajectory estimate using RANSAC followed by gradient descent refinement. We present results for 2D image sequences consisting of single ambiguous, visible or occluded balls, as well as results for 3D computer-generated sequences of objects in free-flight with added long-tailed noise and outliers. Results suggest that restricting the estimation to allow only motions that are feasible according to the physics simulator can produce marked improvement when the observed object motion is within the limits of the physics simulator and its world model. Conversely, merely penalising deviations from feasible physical dynamics produces a consistent but incremental improvement over more common dynamics models.**

I. Introduction

In the field of cognitive robotics, the problem of vision is intricately tangled with other cognitive problems, such as control, interaction and manipulation. Practical solutions to these problems arguably need to cut away from a primarily image-centered approach and need to use frameworks suited to a broader class of problems.

In our attempt to broaden the approach we tackle the classic motion estimation problem, but expand on it incrementally. Specifically, we build software that attempts to estimate dynamic parameters of moving objects from image sequences, in addition to pose, kinematic or image parameters. This software also takes into account the impact of physical dynamics on estimated parameters by making use of physical simulation. We make the prediction that incorporating physical simulation in the motion estimation procedure will improve the accuracy of motion estimation over videos consisting of long sequences of images, particularly in the presence of occlusion, distractors, poor image quality or multiple objects.

In the remainder of this section we discuss some related work and describe our own approach. The following two sections discuss motion estimation and physical dynamics in detail, as applicable to this work. Finally we present some results on 3D motion estimation of a rigid objects from simulated sequences of noisy poses and 2D motion estimation of a ball from colour histograms in real-world videos.

A. Further Motivation

While our approach is primarily aimed as an iteration on conventional robotic solutions, the direction that we are driving towards has some justification in the abilities that humans display. Specifically, in humans non-visual information is important in visual tasks; very early in life objects start to hold implications outside the visual and sensory array. If we are to approximate these human capabilities we need to find ways of making judgements about objects and their location that makes use of expectations about the way that objects behave.

More practically, good quality motion estimation and tracking of single or multiple objects in the presence of occlusion, distractors, glare, and blur is important in a number of areas. Robotic manipulation is one such area, where a robot's interaction with passive objects can lead those objects to move in complicated ways, tumbling or falling for instance. An understanding of how objects move has potential to improve the robot's estimation of object pose. Moreover, there are synergies to be had between control, planning, and visual estimation [1]. The kind of physics-based motion estimation we are investigating will also find application in trajectory estimation of balls in analysis of sports footage, object tracking in virtual reality, and motion capture.

B. Related Work

Elsewhere, work is being done on employing a 3D physical dynamics while tracking human movements [2], [3], [4]. Our work differs from that work primarily in that we are tackling the passive object problem, focussing rather on scenarios in which objects display non-intentional rather than intentional behaviour, and where full object trajectories are considered. However there is a great deal of continuity between that work and this.

The computer graphics community is interested in a number of related problems; motion capture and motion synthesis being the two most related (e.g. Popovic et al. 2000 [5]). Motion estimation from image sequences has applications in motion capture, and potentially uses a similar framework to constrained motion synthesis. A recent example of this kind of synthesis can be see in in the work of Bhat et al., 2002 [6] who estimate the motion of free-flight objects from silhouettes.

There is also a large literature on general motion estimation in long image sequences [7], [8], [9], [10], [11], often posed simultaneously with the problem of structure estimation. Of particular relevance is work that attempts to track objects from image sequences by considering them as deformable 3D objects undergoing local forces calculated from images [12], and earlier work that uses motion capture information to guide a simulation of arbitrary deformable 3D objects [13]. In our work we consider the motion estimation rather than tracking problem, and we fit to image sequences rather than motion capture data.

C. Approach

Within this broader project, in order to motivate our work, we make the specific prediction that it is possible to use prior knowledge about physical dynamics to improve vision tasks within existing task domains, particularly in motion estimation, by exploiting restrictions on the range of feasible motions and by giving weight to more feasible motions. Indeed, it seems sensible to suggest that a strong and accurate physics model will allow extrapolation and interpolation where data from observations is missing or misleading.

We first test this on computer generated data by automatically generating noisy three-dimensional trajectories of rigid objects in free-flight, and of bouncing. We use a least-squares estimation procedure that incorporates knowledge of free-flight and simple collision dynamics to reconstruct the generated trajectory from the input set of noisy poses.

Secondly, we evaluate the use of physical dynamics on real two-dimensional image sequences of a ball or balls moving through a scene or stationary in it. We use a colour histogram matching procedure to generate ball location hypotheses at each frame and best-fit a trajectory to these hypotheses, again using a least-squares cost function.

Before we present the results of these experiments we discuss our our estimation framework and inclusion of physics in more detail.

II. Motion Estimation

The problem of visual motion estimation is related to, but not the same as, the problem of visual tracking. Visual tracking is the problem of maintaining an estimate of instantaneous object location over time - the estimated instantaneous location can be hand-initialised or the tracker can try and try to obtain its own lock. On the other hand, visual motion estimation is the problem of reconstructing a whole trajectory from an image sequence. Clearly the problems are related and solutions to one can be adapted as solutions to the other.

Here we tackle the motion estimation problem, however, because its statement is simpler and it allows us to easily incorporate object dynamics in an explicit fashion. In object tracking, object dynamics are often incorporated implicitly; for instance by making the assumption that an object will not move far between frames, as with mean-shift object trackers where the search for the object in each image frame is initialised from the best location found in the previous frame [14], [15], [16]. In this paper we call this assumption, when made explicitly, "constant displacement dynamics" and use it as a base-case that we compare our new physics-based methods to. Our other base-case is the "constant velocity dynamics" in which the velocity of objects is assumed not to deviate significantly from frame to frame. Both dynamics are usable as a part of any recursive estimation procedure, such as Kalman filter and particle filter variants.

If we specify a forward dynamics function and assume that deviations from the displacement and velocity predictions of this function at each time step are drawn from a normal distribution then we can assign a sum of squares cost to any trajectory, based on that dynamics. In the case of translational dynamics we get:

$$\begin{array}{c} dyncost(\{T_t, T'_t\}) = \\ K_1 \sum_{t=t_i}^{t_f} \left\| dyn_d(T_t, T'_t) - T_{t+1} \right\|^2 \ + \\ K_2 \sum_{t=t_i}^{t_f} \left\| dyn_v(T_t, T'_t) - T'_{t+1} \right\|^2 \end{array}$$

In the above equation $dyncost$ is represents the sum of squares error in dynamics cost over a whole trajectory, $\{T_t, T'_t\}$ (T_t being the pose coordinates of the object at time t and T'_t being the first derivative of this - i.e. the velocity). dyn_d and dyn_v are functions that represent the forward dynamics of an object, by specifying, in the first case, a predicted displacement at timestep $t+1$ given the object state at time t, and, in the second case, a predicted velocity. Rotational dynamics can be specified in a similar way.

The constant displacement forward dynamics therefore has $dyn_d(T_t, T'_t) = T_t$ and $K_2 = 0$. i.e. the displacement is not expected to change, and the velocity is not considered.

Similarly, a constant velocity forward dynamics (assuming that each time step is one unit of time) has $dyn_d(T_t, T'_t) = T_t + T'_t$ and $dyn_v(T_t, T'_t) = T'$.

Given this framework, any forward dynamics can be incorporated.

The dynamics cost is mirrored by the observation cost, which is, again, a sum of squares cost based on the deviation between predicted observations and actual observations:

$$obscost(\{T_t, T'_t\}) = \sum_{t=t_i}^{t_f} \left\| obs(T_t) - T_{t+1} \right\|^2$$

The obs function, given above, is any function that predicts an observation from an object's location. In the case of our experiments below, obs simply provides location of an object given its estimated location position at any time.

Given a cost function made up of the sum of observation and dynamics costs, we can attempt to find a trajectory $\{T_t, T'_t\}$ that minimizes it - i.e. we attempt to find the least squares in the cost terms. This can be carried out by any general optimisation procedure. We employ the RANSAC algorithm [17] to select inliers and initialise solutions, and we use a gradient descent using finite differences and line search to subsequently refine the solution.

Note that in the RANSAC procedure, an accurate dynamics model is required to instantiate a full trajectory from a small number of observations. Without this, it is necessary to either use large numbers of samples of observations (somewhat defeating the purpose of RANSAC) or to frequently risk failing to find a sufficiently accurate trajectory. The constant displacement and constant velocity dynamics models described above do need to make this tradeoff.

III. Incorporating Physical dynamics

Within the above cost-minimisation framework it is a simple extension to add more sophisticated physical dynamics by providing a more sophisticated dyn function. We adapt the commercial PhysX physics engine to this purpose, by supplying it with a rough model of the environment and of the object to be estimated. We call this dynamics a "locally parametrised collision dynamics". It is called "local" because a dynamics cost is calculated locally between pairs of points and summed together to provide a trajectory cost, as discussed above.

We also implement a slightly different "globally parametrised collision dynamics". If we allow that no deviation in the trajectory from the dynamics model is possible then we may parametrise an object's entire trajectory in terms of the instantaneous motion parameters of the object at only one time-point during that trajectory, since the dynamics model deterministically specifies the motion parameters of the remaining time-points. In this case, the cost function to be minimised is simply the observation cost $obscost$, and the parameters to be estimated are the velocity and pose parameters at a single point during the trajectory. We call this approach a "global" model because the model allows the entire trajectory to be parametrised in terms of the motion parameters of a single time-point.

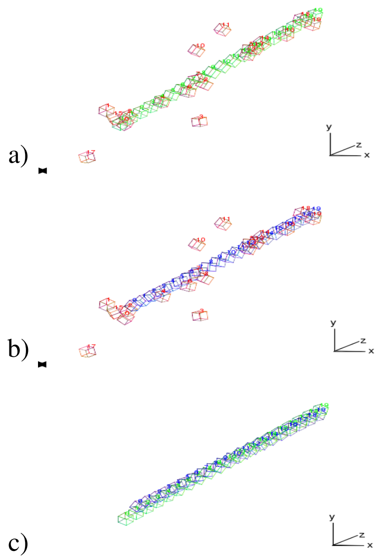
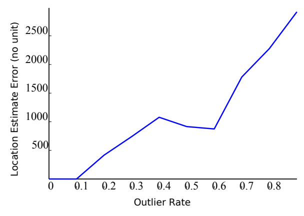


Fig. 1. A single run of the 3D free-flight trajectory estimation algorithm (RANSAC + refinement) on a rigid object in free-flight. The axes are without unit. (a) Green: The veridical (simulated) trajectory. Red: The same trajectory with added Cauchy noise ($\gamma = 0.5$) and outliers ($r = 0.1$). (b) Red: The noisy trajectory again. Blue: The trajectory fit to it using RANSAC. (c) Green: Again, the veridical (simulated) trajectory. Blue: Again, the trajectory reconstructed by RANSAC.

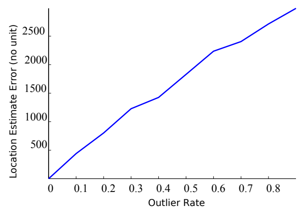
IV. Experiments

A. Free-flight 3D Rigid Object Motion Estimation in Simulation

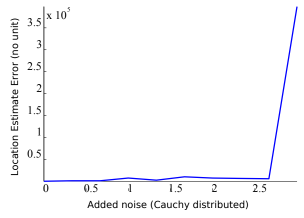
Our first experiment was intended to determine the workability of the proposed method.



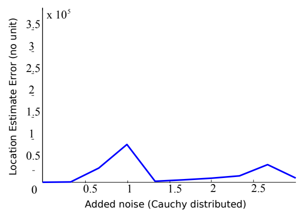
(a) RANSAC



(b) Refinement



(c) RANSAC



(d) Refinement

Fig. 2. Results of applying free-flight 3D motion estimation algorithms to automatically generated noisy data (50 trajectories are generated and fitted to). The error has no fixed unit. (a) Total pose RMS error in estimated trajectory found by the RANSAC motion estimation algorithm, as outlier rate increases. (b) Total pose RMS error in estimated trajectory found by the refinement-only motion estimation algorithm, plotted against increasing outlier rate. (c) Total pose RMS error of the RANSAC motion estimation algorithm with increasing long-tailed noise. (d) Total pose RMS error in the refinement-only motion estimation algorithm with increasing long-tailed noise.

1) Method: We assume in this experiment that the algorithm takes as input 3D pose estimates at each time step, as supplied by some arbitrary pose-estimation routine. We test our algorithm by testing it on trajectories consisting of sequences of poses generated by first simulating a trajectory and then adding some noise. Noise is added in the form of long-tailed noise from a Cauchy distribution, which has a single spread parameter γ, as well as added outliers parametrised by the probability of any pose being an outlier, p. We generate trajectories consisting of rigid objects in free-flight, under the influence of gravity, or bouncing in a simple fashion. This experiment was implemented in matlab and the simple physics models used were programmed by hand. A generated free-flight trajectory, and the same trajectory with added noise can be seen in fig. 1(a).

In this experiment, the dynamics model used in estimation is the same as that used to generate data initially. As a result, we only need to make use of the "globally parametrised dynamics model" mentioned above, since we know that the dynamics models used are sufficient to model the data.

2) Results: The output of a sample estimation run can be seen in fig. 1 (b-c), where output of the estimation algorithm is compared against the noisy data that is input into the algorithm (b), and is also compared to the ground-truth trajectory before noise was added (c).

Fig. 2 shows the result of applying the estimation algorithm in the presence of varying amounts of noise, and with or without the RANSAC initalisation step. These results show that RANSAC is robust to higher levels of noise, which is expected.

Later experiments with bouncing objects suggest that the

RANSAC step was even more necessary under the more complex physical dynamics since the search space forms at least one local minima for each cardinal number of bounces.

3) Discussion: The assumption of the existence of a pose-estimation routine that can provide 3D pose estimates is clearly not universally valid, though it does allow us to construct our algorithm in a feature-agnostic way. However, the observation cost function can in theory be based on any arbitrary least-squares likelihood cost.

When calculating the distance between a predicted pose and an observed pose, we need some way of comparing poses that takes into account both translational and rotational error. We motivate this calculation by making the assumption that we are interested in the integral sum of error distance across all points in the object. By integrating analytically, we obtain the pose error cost $|V|t_e^2 + 2(1 - cos\theta_e)\omega_e^T I\omega_e$, where t_e is the translational error, θ_e, ω_e comprise the rotational error in angle-axis form,$|V|$ is the volume, and I is the inertial matrix of the object, calculated assuming a constant unit density.

B. Ball Motion Estimation from Histogram Matches in 2D

Having ascertained the practicability of the proposed method it is necessary to apply it to a real-world problem. We apply it to the problem of tracking a ball as it moves through a scene in a small number of video image sequences, some frames of which can be seen in fig. 3.

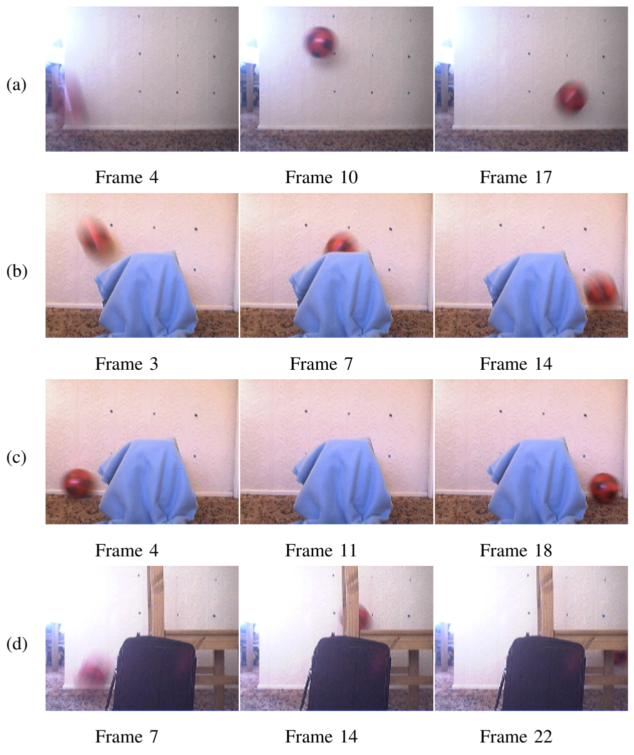


Fig. 3. Image sequences used to test 2D motion estimation. (a) A ball bounces across the field of view with glare, blur. (b) A ball bounces across the field of view, occluded by a stationary object mid-sequence. (c) A ball rolls across the field of view, again occluded by a stationary object. (d) A ball bounces across the field of view, with blur, glare, partial and full occlusion, and in the presence of a strong distractor.

1) Method: We model the ball using a normalised colour histogram, and for each image frame we determine the points in the image that correspond to image areas that best match the ball according to the Bhattacharyya distance measure. This measure is in wide use in the literature for tracking from colour histograms [16], [14], [15]. We then use these points as observations and generate an observation cost function from them. Observation cost terms and RANSAC inlier thresholds for each observation are weighted according to histogram match score.

We examine the effect on the quality of estimation of each of a handful of different dynamics functions. Standing in for the implicit and explicit dynamics found in the tracking literature, we have a constant displacement dynamics function and a constant velocity dynamics function. We also have a no-dynamics case where the best observation in each frame is taken as-is; in object tracking this would be equivalent to tracking-by-detection.

To demonstrate our experimental approach we have two cases with dynamics derived from a physics engine (in which we model the floor and as many balls as are in the scene). These cases are the locally parametrised collision dynamics and globally parametrised collision dynamics.

Note that in the case of globally parametrised dynamics it is required that any time-point can be used to parametrise a trajectory, since the RANSAC procedure will sample observations at arbitrary time-points. As such, we need to be able to physically simulate backwards in time from such a time-point. We achieve this by inerting object velocities and coefficient of restitution and running the simulator forward, however we note that physical parameters such as static and dynamic friction, and linear damping, are not invertible in the physics simulator. This means that, during RANSAC using collision-based dynamics, and while doing gradient descent on the globally parametrised cost function, the estimator can acquire trajectories that it would not normally be capable of simulating. As discussed in section II, however, any locally parametrised dynamics cost only requires a forward dynamics function.

Note that because we use normalised histograms to detect the ball in the video images, the black bag in the video in fig. 3(d) provides a distractor while the ball is occluded. As an artifact of the fact that a ball location is observed at every frame in the image sequence, a mild distractor effect also occurs whenever the ball is not sufficiently visible.

2) Results: Fig. 4 gives the estimation performance of each of the five dynamics models. The constant displacement and constant velocity dynamics fail very badly in many cases because they are unable to select good inliers. As such we also compare the performance of the refinement component of each algorithm by initialising them all with the RANSAC routine employing collision dynamics - see fig. 5. Fig. 6 contains characteristic examples illustrating the behaviour of constant displacement, constant velocity, and collision dynamics on each video sequence.

The two collision-based dynamics perform better on every video except the video where the ball rolls behind an

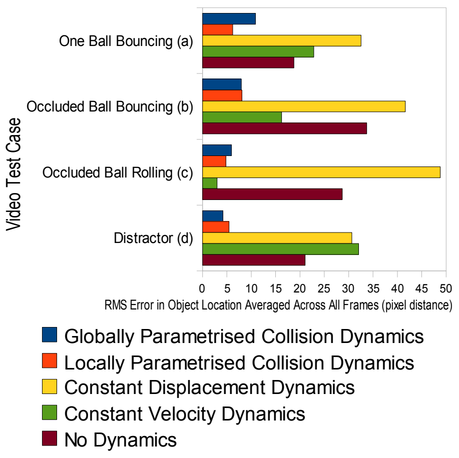


Fig. 4. Initialisation and refinement performance of 2D motion estimation from colour histogram algorithms: Performance measured in RMS error in pixel distance from labelled object location, averaged over all frames of each of five algorithms on each of four image sequences. Shorter bars indicate better performance and lower error. In these results, each dynamics model is used both in the initialisation and refinement phases. (a-d) refer to the same image sequences as shown in fig. 3.

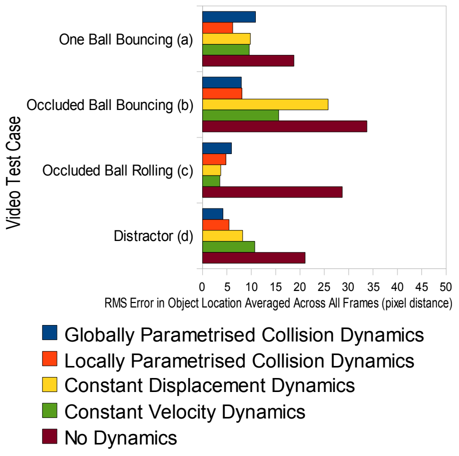


Fig. 5. Refinement only performance of 2D motion estimation from colour histogram algorithms: Performance measured in RMS error in pixel distance from labelled object location, averaged over all frames of each of five algorithms on each of four image sequences. Shorter bars indicate better performance and lower error. In these results, the collision-based dynamics is used select an initial set of inliers and to initialise the refinement procedure which then uses each of the dynamics models listed in the legend. (a-d) refer to the same image sequences as shown in fig. 3.

occluder and out the other side again, (c). In this video the constant velocity dynamics performs better. In such a situation the constant velocity dynamics is a sufficient model of object behaviour. The collision-based dynamics fare worse because the ground plane is slightly tilted with respect to the camera while the provided dynamics model assumes that the ground plane is perfectly parallel with the image x-axis.

Note also that the locally parametrised collision dynamics does better than the globally parameterised collision dynamics in those videos where observation data is useful in correcting the subtle mismatches between the dynamics model and the behaviour of objects in the real-world (a & c). Where there are lots of distractors and occlusion (d), however, the global collision dynamics is able to compensate more for the bad data than local collision dynamics can.

In video (a) we see that the constant displacement dynamics makes a characteristic error of estimating the ball position near to observed positions nearby in time, and the constant velocity dynamics estimates the ball position to be travelling through the floor. The collision dynamics is able to compensate somewhat by giving credence to the hypothesis that the ball may have bounced. Video (b) shows both non-collision dynamics unable to benefit from hypotheses involving the bouncing of the ball behind the obstacle. Video (c) and (d.i) show the constant velocity dynamics succeeding but constant displacement dynamics unable to initialise a trajectory that moves a long distance between observations. Video (d.ii) shows constant velocity dynamics unable to initialise a trajectory that involves the ball changing direction while occluded.

Additional experiments suggest that the performance of the algorithm is much more sensitive to the RANSAC inlier threshold parameter, than to tunable parameters in the physics model such as coefficient of restitution.

3) Discussion: The data confirm our hypothesis that a more sophisticated dynamics model, particularly involving collisions and gravity, is able to substantially improve motion estimation, particularly when observation data is absent or misleading. However, this improvement is contingent on a good match between model and world.

V. Summary and future work

A. Summary

We present a motion estimation framework that allows the incorporation of arbitrary dynamics models. We demonstrate that the use of a dynamics as provided by an off-the-shelf physics simulator is able to improve the accuracy of motion estimation when the model and the world match well.

B. Future Work

There are open questions as to which parameters of a dynamics model the accuracy of estimation is sensitive to, which simplifications may be made with little impact on performance, and which model aspects are already impacting accuracy by deviating from real-world behaviour. We would like to test the proposal that allowing simulatenous refinement of parameters of the physics engine, local surface parameters, even shape parameters, will lead to a better fit and a more accurate estimation procedure. It might be possible to use the provided physics model to bootstrap the learning of a more generic dynamics model or one more like those used by humans[19].

We are currently working on image sequences involving two interacting identical balls, again in 2D, much like in the work of Chang, et al. 2005 [18]. This work involves the necessary addition of layers in the Z direction and a data association framework. We predict that collision dynamics will improve performance in a number of scenarios, for instance, when an object collides with another and loses

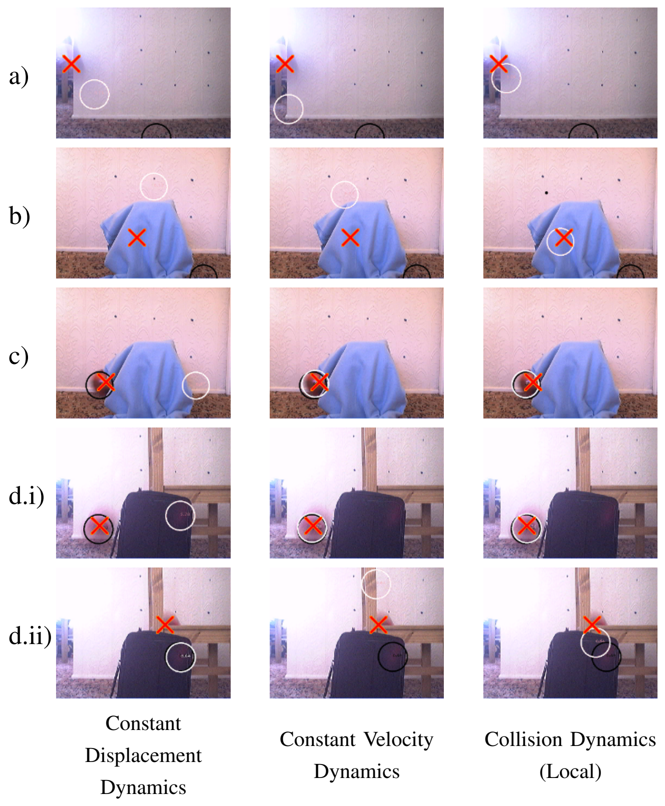

a)
b)
c)
d.i)
d.ii)
Constant Displacement Dynamics
Constant Velocity Dynamics
Collision Dynamics (Local)

Fig. 6. Some examples of success and failure of 2D motion estimation from colour histogram algorithms. Three dynamics models are illustrated: constant displacement, constant velocity, and collision dynamics based on local parametrisation. Each row shows one time-point during a test image sequence, and each column corresponds to a dynamics model as applied to that image sequence. The white circle represents the estimated ball location as determined using the dynamics model; the black circle represents the estimated ball location as determined in the absence of any dynamics model; the red cross represents the pre-labelled (by hand) ball location. The first two rows are from examples pre-initialised using collision dynamics while the remaining three rows use the same dynamics model for initialisation as for refinement. (a) One bouncing ball, frame 3. (b) Ball bouncing while occluded, frame 5. (c) Ball rolling while occluded, frame 6 (here it can be seen that constant velocity dynamics performs best). (d.i) Ball bouncing behind distractor, frame 7 (in this frame constant velocity dynamics performs almost as well as the collision-based dynamics but constant displacement dynamics has failed because of the distractor). (d.ii) This is the same case at a later frame: ball bouncing behind distractor, frame 15 (by this point in the sequence the constant velocity dynamics has also failed).

energy as a consequence, thereby disambiguating the nature of the interaction.

There is also room for incorporating a more intelligent way of handling observations, such as using the colour histogram match score directly in the observation function, thereby filling the gap currently filled by intelligent dynamics by better observations.

Of course, it would be useful to extend this work to 3D and to arbitrary objects. Our next effort will be in the area of robotic manipulation where we will adapt a recursive filter such as a particle filter or an unscented Kalman filter to use the physics engine in its probabilistic forward dynamics model. Particular concerns are the nonlinearity of the physics model and potentially high number of dimensions when considering velocity state parameters and multiple objects.

VI. Acknowledgments

We gratefully acknowledge the contribution of the EU FP6 IST Cognitive Systems Integrated Project (CoSy), FP6-004250-IP, the EU FP7 IST Project CogX FP7-IST215181, the University of Birmingham School of Computer Science and the UK Overseas Research Students Awards Scheme. Many thanks to Zeyn Saigol for his careful reading.

References

[1] D. Kragic, A. T. Miller, and P. K. Allen, "Real-time tracking meets online grasp planning," in *Proceedings IEEE International Conference on Robotics and Automation*, Seoul, Republic of Korea, 2001, pp. 2460–2465.

[2] C. Wren and A. Pentland, "Dynamic models of human motion," in *Proceedings IEEE International Conference on Automatic Face and Gesture Recognition*, 1998, pp. 22–27.

[3] M. Vondrak, L. Sigal, and O. C. Jenkins, "Physical simulation for probabilistic motion tracking," in *Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 0. Los Alamitos, CA, USA: IEEE Computer Society, 2008, pp. 1–8.

[4] C. Sminchisescu and B. Triggs, "Estimating articulated human motion with covariance scaled sampling," *International Journal of Robotics Research*, vol. 22, no. 6, pp. 371–393, 2003.

[5] J. Popovic, S. M. Seitz, M. Erdmann, Z. Popovic, and A. Witkin, "Interactive manipulation of rigid body simulations," in *SIGGRAPH*, 2000, pp. 209–218.

[6] K. S. Bhat, S. M. Seitz, J. Popovic, and P. K. Khosla, "Computing the physical parameters of Rigid-Body motion from video," in *Proceedings of the European Conference on Computer Vision: Lecture Notes in Computer Science*, vol. 2350, 2002, pp. 551–565.

[7] G. J. Young and R. Chellappa, "3-D motion estimation using a sequence of noisy stereo images: Models, estimation, and uniqueness results," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, no. 8, pp. 735–759, 1990.

[8] G. Young, R. Chellappa, and T. Wu, "Monocular motion estimation using a long sequence of noisy images," in *Proceedings International Conference on Acoustics, Speech, and Signal Processing*, 1991, pp. 2437–2440.

[9] X. Hu and N. Ahuja, "Estimating motion of constant acceleration from image sequences," in *Proceedings IAPR International Conference on Pattern Recognition, Conference A: Computer Vision and Applications*, 1992, pp. 655–659.

[10] ——, "Long image sequence motion analysis using polynomial motion models," in *IAPR Workshop on Machine Vision Applications*, 1992, pp. 109–114.

[11] T. J. Broida and R. Chellappa, "Estimating the kinematics and structure of a rigid object from a sequence of monocular images," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 13, no. 6, pp. 497–513, 1991.

[12] M. Chan, D. Metaxas, and S. Dickinson, "Physics-Based tracking of 3D objects in 2D image sequences," in *Proceedings International Conference on Pattern Recognition*, 1994, pp. 432–436.

[13] D. Metaxas and D. Terzopoulos, "Shape and nonrigid motion estimation through physics-based synthesis," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 15, no. 6, pp. 580–591, 1993.

[14] G. R. Bradski, "Computer vision face tracking for use in a perceptual user interface," *Intel Technology Journal*, vol. Q2, 1998.

[15] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 25, no. 5, pp. 564–577, 2003.

[16] A. Naeem, S. Mills, and T. Pridmore, "Structured combination of particle filter and kernel Mean-Shift tracking," in *International Conference Image and Vision Computing New Zealand*, vol. 21, 2006.

[17] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, pp. 381–395, 1981.

[18] C. Chang, R. Ansari, and A. Khokhar, "Multiple object tracking with kernel particle filter," in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[19] R. Grzeszczuk, D. Terzopoulos, and G. Hinton, "NeuroAnimator: fast neural network emulation and control of Physics-Based models," in *Computer Graphics Proceedings, Annual Conference Series*. Orlando, Florida: ACM SIGGRAPH, 1998, pp. 9–20.

Representations for Object Grasping and Learning from Experience

Oscar J. Rubio, Kai Huebner, and Danica Kragic

***Abstract*— We study two important problems in the area of robot grasping: i) the methodology and representations for grasp selection on known and unknown objects, and ii) learning from experience for grasping of similar objects. The core part of the paper is the study of different representations necessary for implementing grasping tasks on objects of different complexity. We show how to select a grasp satisfying force-closure, taking into account the parameters of the robot hand and collision-free paths. Our implementation takes also into account efficient computation at different levels of the system regarding representation, description and grasp hypotheses generation.**

I. INTRODUCTION

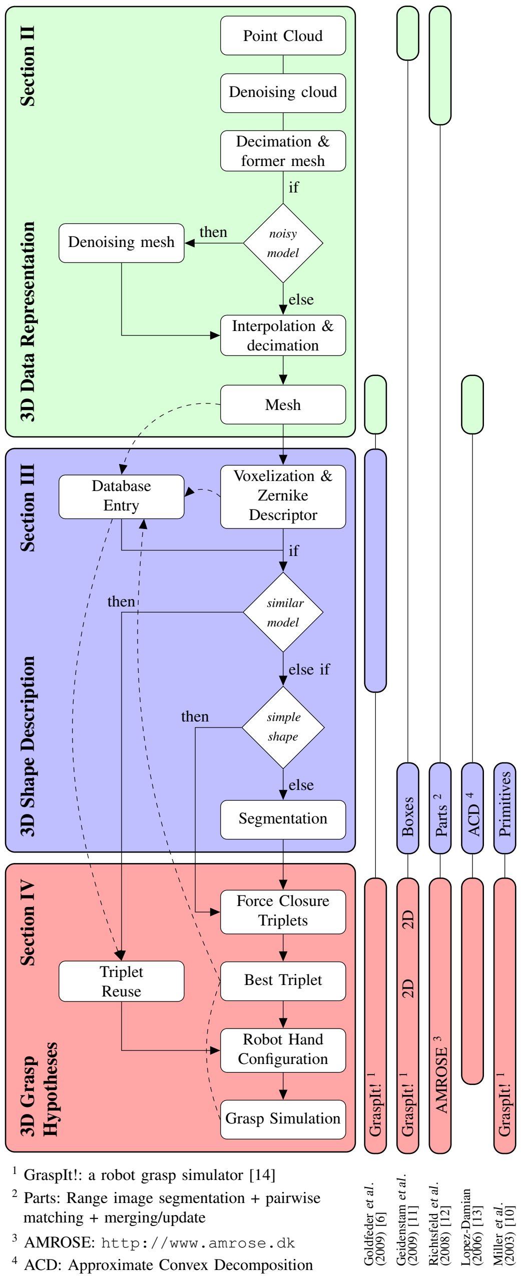
A robot grasping cycle involves data representation, shape description and grasp hypotheses generation, Fig. 1. Most of the recent work on robotic grasping relies on 3D data although there are approaches producing grasp hypotheses using 2D image features, e.g. [1]. A range of state-of-the-art methods synthesize 3D object shapes from point clouds by using superquadrics [2] or other shape primitives such as boxes [3]. Assuming that an arbitrary point cloud has to be approximated, a single primitive is obviously not enough for many objects. The more complex the shape is, the more primitives have to be used to represent its individual parts. Multiple methods approach this problem by a variety of segmentation methods, [2], [3], [4].

A major issue is that for *unknown objects*, grasps need to be evaluated from data a robot can extract on-line. This is a difficult problem due to the (i) high dimensionality of the problem, (ii) incomplete and uncertain information about the environment and the objects to be grasped, and (iii) lack of generalizable measures of quality for grasp planning, i.e. (*"What is a good grasp?"*).

Many systems rely on object recognition and/or shape registration. This requires a database of objects or shapes, as for example in [5], or objects combined with grasps, [6]. To approach the recognition problem, an object has to be described using a shape descriptor meeting some desirable requirements: it should be primarily able to cope with real-time and real-world requirements: it should be compact, invariant under transformations, and fast to calculate. An example are the Zernike descriptors, [7].

Overall, there has been a lot of work on grasp planning on different levels: path planning [8], planning on 3D mesh models [9] and databases [6], planning on shape primitives [10], [5]. A general problem is that for any kind of single-view 3D sensor system, a generated point cloud of an object or a scene is not complete. Grasp selection may also benefit from assesing the shape complexity. If complexity is high, it is worthy to segment the object into graspable parts. The segmentation of a model into its parts is also necessary for task-constrained grasping of simple objects.

This work was supported by EU through the project CogX, IST-FP6-IP-027657; GRASP, IST-FP7-IP-215821 and Swedish Foundation for Strategic Research. The authors are with the Computer Vision and Active Perception Lab, Centre for Autonomous Systems, School of Computer Science and Communication, KTH, Stockholm, Sweden. {ojrm,khubner,danik}@kth.se.



[1] GraspIt!: a robot grasp simulator [14]
[2] Parts: Range image segmentation + pairwise matching + merging/update
[3] AMROSE: http://www.amrose.dk
[4] ACD: Approximate Convex Decomposition

Fig. 1. Structure of our approach, and related approaches' foci.

However, the decision if an effort in terms of a segmentation or part-based decomposition is in fact worthy is often not treated in the literature. Rarely, a set of differently granulated decision criteria and representations is evaluated to see if some partitioning of the object in fact is reasonable or not.

In this paper, we present a system for robot grasp selection that copes with known and unknown objects where the focus is not to find the most stable grasp, but a grasp that is force-closure and feasible, i.e. collision-free and constrained by the parameters of the robotic hand, given available sensor data.

II. 3D Data Representation

In our work, the input to the system is a sparse or dense 3D point cloud. We then use a mesh representation to interpolate and assign volume to the data. In the following subsections, we describe: denoising, fast point cloud decimation and interpolation, mesh building, convex hull computation and convexity estimation.

A. Denoising

Before mesh building, we perform noise removal using the ANN library [15]: we first calculate d_i as the sum of distances to the 10 nearest neighbors for each point i. We then obtain the number of points n_{iso} which hold $d_i > 3 \cdot mean(d)$ and tag them as isolated vertices to calculate the ratio $r = n_{iso}/n \in [0,1]$. If $r < 0.99$, we perform noise removal. After building a mesh (Section II-C), we delete faces with longest edges, since residual noisy points will be linked by long edges to the surface of the model. For this reason, the perimeter of all the faces of the mesh will be calculated and those ones whose perimeter is more than twice as long as the average will be deleted. Consequently, if there were groups of noisy faces, they will be isolated and removed from the model. This may cause cracks and holes in the mesh, therefore it is advisable to perform an interpolate-decimate step.

B. Fast Point Cloud Decimation and Interpolation

Obtaining a mesh from a point cloud is a non-trivial task: dense data results in high-quality meshes, but needs more time to be processed. A sparse set of points may be processed fast, but can result in incomplete or erroneous meshes.

1) Decimation: We limit the input point cloud O to a minimum number of points n, and decimate large point clouds by using a filter, preserving constant density in the output point cloud. This increases the accuracy of the axis extraction and benefits the grasp search as well: since all the faces in the mesh representation will be similar sized, all the areas of the object have similar probability of becoming grasp points. The filtering is performed by removing neighbors of each point using kd-tree-search, [15] in a radius

$$rad = \sqrt{surfaceArea(convexHull(O))/(k \cdot n)}, \qquad (1)$$

where n is the number of points to keep and the factor $k = 2.43$ was found empirically. We allow a tolerance of 5% to the requested number of output points. In case of dense point clouds, a random reduction is performed first to speed up the decimation.

2) Interpolation: The above procedure ensures a good result by decimating the most redundant points in the cloud but it does not create new points in sparse areas. We approach those cases taking advantage of the mesh representation by (i) iteratively subdividing all the triangles of the mesh, until the number of vertices exceeds a number n, (ii) iteratively subdividing only those triangles t_i with $perimeter(t_i) > 1.2 \cdot mean(perimeter(t))$, and (iii) decimating the resulting cloud to n points. To subdivide a triangle into four similar sized triangles, we use triangle edge bisection. In the evaluation section, we will motivate our choice to set $n = 2000$.

C. Mesh Building

The mesh building process enables the subsequent voxelization and the grasp search including optional segmentation as well. Being more than a pure requirement, meshing is important since the quality of the mesh greatly affects the estimated quality of the final grasp. We use a PowerCrust-based algorithm [16] to acquire a tightly closed *triangular* mesh for each point cloud. We stress the importance of performing the manifold extraction, i.e. deleting badly oriented triangles and ensuring that all remaining triangles are roughly parallel to the surface.

An inappropriate meshing will cause omitting of existing surfaces (false negatives) or adding non-existing surfaces (false positives) in the grasp search. However, both problems can be treated: the first can be detected by checking the percentage of points from the point cloud included in the mesh; in our approach, we reject meshes with a confidence value below 90%, leaving a margin of 10% for small details and/or outliers. The second problem can be avoided or at least minimized after simple post-processing.

D. Convex Hull Computation and Convexity Estimation

We estimate the minimum convex hull enclosing the cloud and its volume using an implementation of the Quickhull algorithm [17], which has shown itself as the most efficient. Based on the mesh, we can measure the convexity of our model. If the object is convex, the volume of the object mesh and the volume of its convex hull are equal. Moreover, the more concave an object O is, the lower the convexity ratio

$$conv = volume(O)/volume(convHull(O)) \in [0,1]. \qquad (2)$$

We exploit this ratio to evaluate the complexity of an object: the simpler the object shape, the closer its convexity to 1.

III. 3D Shape Segmentation

The segmentation divides complex shapes into simpler, independently graspable parts. We consider a body to be complex if its convexity measure (2) is lower than 0.85. Different object segmentation methods have been proposed in the literature: spectral clustering [18], minimum volume bounding box (MVBB) decomposition [3], hierarchical segmentation based on primitives [19]. However, most of these are either time-consuming or not suitable for integrating with grasp selection. We develop a new algorithm for segmentation, fulfilling the requirements to be grasp-oriented, fast, simple and robust. We use core extraction as a starting point, add a system to carry out several cuts in the mesh using a criterion to find out which one segments best. Although this strategy is similar to MVBB decomposition, there are two main differences: first, we use convex hulls instead of boxes for the decomposition in order to get higher flexibility; and second, our starting point is the hull enclosing the core instead of a box surrounding the whole object.

Our algorithm works as follows:

1) Center-of-Mass Extraction: Assuming that the density in an object O is constant, we estimate the volume integrals by tetrahedrons generated from the mesh representation. Based on the tetrahedrons, we approximate the center of mass of O. First, the center point of each tetrahedron t_i is computed from the ith mesh triangle $\Delta_i = (\mathbf{v}_1(i), \mathbf{v}_2(i), \mathbf{v}_3(i))$ by

$$\mathbf{c}(t_i) = (\mathbf{v}_1(i) + \mathbf{v}_2(i) + \mathbf{v}_3(i) + \mathbf{p})/4 \tag{3}$$

where $\mathbf{p}$ is a random point inside the model. Secondly, we calculate the each tetrahedron's signed volume

$$V(t_i) = (\mathbf{v}_1(i) - \mathbf{p}) \cdot ((\mathbf{v}_2(i) - \mathbf{p}) \times (\mathbf{v}_3(i) - \mathbf{p}))/6 \tag{4}$$

before averaging the distances using the signed volume of each tetrahedron to approximate the object's center of mass

$$\mathbf{c}(O) = \sum_{i=0}^{n} V(t_i)\mathbf{c}(t_i) / \sum_{i=0}^{n} V(t_i) \tag{5}$$

Note that for (4), we have to assure that all triangles in the mesh are defined with the same orientation, clockwise or counter-clockwise. The simple steps result in a good approximation even with a very reduced number of input points, avoiding overestimation of denser areas in the cloud.

2) Core Extraction via Spherical Mirroring: Spherical mirroring aims at reversing the situation, in such a way that vertices of the core become external and easily extractable. To achieve this, all vertices of the mesh are mirrored on a minimal bounding sphere. Thereby, vertices of the core component are identified as residing on the convex hull of the mirrored vertices. With $\mathbf{c}(O)$ the center of the sphere, the mirrored vertices $\mathbf{v}'$ are given by

$$\mathbf{v}' = \mathbf{v} + 2\tfrac{r - d(\mathbf{v})}{d(\mathbf{v})}(\mathbf{v} - \mathbf{c}(O)) \text{ , with} \tag{6}$$

$$d(\mathbf{v}) = \|\mathbf{v} - \mathbf{c}(O)\| \text{ and } r = \max_{\mathbf{v}} d(\mathbf{v}) \text{ .} \tag{7}$$

An example for core extraction is shown in Fig. 2 (a)-(b).

3) Cut Trials: We associate core points with their corresponding triangles in the mesh. Then, we apply an algorithm based on the connectivity filter of the graphical library VTK to cut the mesh into parts. We found that the size of the core enclosing the hull is usually not optimal to break the mesh in a suitable way, thus the process is repeated scaling the size of the hull by several scales within a range between 1 and 2. The best segmentation is defined by the scale minimizing the sum of convex hull volumes over all parts. Some models segmented using this method are presented in Fig. 2 (c).

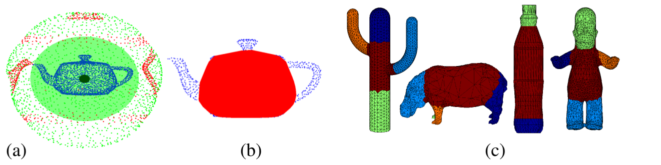


Fig. 2. Core segmentation by spherical mirroring. (a) original point cloud, center of mass, mirroring sphere (tightly enclosing the teapot) and mirrored points. (b) Core (solid) and points outside (dotted). (c) Some examples.

4) Cut Refinement: After segmentation, the different parts are studied and catalogued as *graspable* or *non-graspable*. We define two requirements for *graspable* parts as (i) being larger than 1/100 of the size of the whole object, and (ii) having some curvature, since flat regions are nou suitable for grasping. The non-graspable parts in a segmentation are iteratively merged with their closest neighbor.

IV. 3D Grasp Hypotheses Generation

Obtaining a 3D grasp hypothesis for a given object is the primary purpose of a grasping system. The two major policies to acquire are either (i) the search for and selection of best candidate for a new object, or (ii) the adaptation of a learned grasp on a similar or familiar object.

A. Generation of Force-Closure Triplets

The search for force-closure property is aimed at collecting a list of triplet candidates, where each triplet representing the the fingertip positions of a three-fingered robotic hand should reach on the object's surface to result in a stable grasp. We base our search on the method described in [20], adapting the idea from four-fingered hands to three-fingered hands. We acquire a set of N contact triplets using Algorithm 1.

We highlight that the probability of selecting a grasp with stability higher than the average human grasp quality is very

Algorithm 1: Triplet-from-Mesh Computation.

```
input : Set of triangle mesh $M = \{(\Delta_1, ..., \Delta_m\}$, their outwards
        normals $N(\Delta_i)$ and the friction coefficient $\mu$.
output: Set of triplets $\{T_1(M), ..., T_N(M)\}$
begin
    for $n \leftarrow 1$ to $N$ do
        Choose random $\Delta_r$ and center as 1. contact point:            finger 1
        $\mathbf{f}_1 \leftarrow (\mathbf{v}_1(r) + \mathbf{v}_2(r) + \mathbf{v}_3(r))/3 : r = rand(1, m)$
        Sample a ray $\mathbf{r_1}$ departing from $\mathbf{f}_1$ and deviating            finger 2
        from the negative surface normal of $\Delta_r$, using the
        friction cone angle as standard deviation:
        $\mathbf{r_1} \leftarrow rotate(-N(\Delta_r), \alpha)$ :
        $\alpha = normrnd(0, \tan^{-1}(\mu))$
        Find the intersections of $\mathbf{r_1}$ with the mesh $M$:
        $\{\mathbf{i}_1, ..., \mathbf{i}_l\} \leftarrow intersect(\mathbf{r_1}, M)$
        If there is more than one such point, we choose one at
        which the surface is penetrated outwards:
        $\mathbf{f}_2 \leftarrow$ if $(l = 1)$ then $\mathbf{i}_1$ else $chooseOutwards(\mathbf{i}_1, ..., \mathbf{i}_l)$
        Sample a ray $\mathbf{r_2}$ perpendicular to the line            thumb
        given by $\mathbf{f}_1, \mathbf{f}_2$ with origin at the middle point:
        $\mathbf{r_2} \leftarrow\perp (\mathbf{f}_1\mathbf{f}_2)$
        Find the intersections of $\mathbf{r_2}$ with the mesh $M$:
        $\{\mathbf{i}_1, ..., \mathbf{i}_l\} \leftarrow intersect(\mathbf{r_2}, M)$
        If there is more than one such point, we choose one at
        which the surface is penetrated outwards:
        $\mathbf{f}_0 \leftarrow$ if $(l = 1)$ then $\mathbf{i}_1$ else $chooseOutwards(\mathbf{i}_1, ..., \mathbf{i}_l)$
        $T_n(M) \leftarrow (\mathbf{f}_0, \mathbf{f}_1, \mathbf{f}_2)$            triplet
```

high after $N = 100$ randomly generated grasps, from [21]. Since that study was done with few objects corresponding to primitives shapes, thus different from our input data, we extend our search to $N = 400$ grasps and stop at the first sample exceeding a quality threshold. If none of the grasps exceeds this threshold, the best one is taken. We choose the number of triplets to be generated on each part to be proportional to its relative size.

Taking advantage of the mesh representation which provides faces in a mesh and their corresponding normals, it is possible to evaluate the quality of a grasp created from a given triplet. Since contact points and their normals are thus given, we can easily approximate the friction cones, estimate the convex hull of the Grasp Wrench Space and obtain a quality measure.

B. Reuse of Triplets

When dealing with similar shaped objects, we want to reuse the stored grasp hypotheses. This is realized by finding the affine transformation that resizes and reorients the original model M to match the current object O, allowing for the same with its corresponding triplet. The information we need is the original triplet, $T(M) = (\mathbf{f}_{M,0}, \mathbf{f}_{M,1}, \mathbf{f}_{M,2})$, the center-of-mass ($\mathbf{c}$), the volume ($V$) and the main axes of inertia ($\mathbf{A}$) of both M and O. Since we obtain high quality meshes after interpolation-decimation, Principal Component Analysis (PCA) is suitable for the latter.

C. Hand Configuration

The next step is to translate a triplet of contact points $(\mathbf{f}_0, \mathbf{f}_1, \mathbf{f}_2)$ into a robotic hand configuration, which in our case is the Barrett hand [22]. The hand has 10 degrees of freedom: 6 for the pose of the wrist, 1 for the spread angle of the fingers and 3 for the proximal joint angles. The choice of the triplet points according to the method described in IV-A permits obtaining an optimal configuration of the Barrett hand after Algorithm 2.

Algorithm 2: Triplet-to-Barrett Hand Adaptation.

```
input : Triplet $T = (\mathbf{f}_0, \mathbf{f}_1, \mathbf{f}_2)$.
output: Hand configuration $(\mathbf{p}, \mathbf{o})$ (pose), $(\Theta, e)$ (spread, extension).
begin
    Compute the normal vector of $T$:
    $\mathbf{n}_T \leftarrow (\mathbf{f}_1 - \mathbf{f}_0) \times (\mathbf{f}_2 - \mathbf{f}_0)$.
    After finding the circle $(\mathbf{c}, r)$ passing through $\mathbf{f}_0, \mathbf{f}_1, \mathbf{f}_2$ on the
    plane described by $T$, acquire the hand configuration:
    $(\Theta, e) \leftarrow (0.5 \cdot \measuredangle\, \mathbf{f}_1\mathbf{c}\mathbf{f}_2, r)$.
    There are two possible configurations:
    $(p, o)_1 \leftarrow (\mathbf{c} + (-0.953 \cdot e + 128.8) \cdot \mathbf{n}_T, -\mathbf{n}_T)$.
    $(p, o)_2 \leftarrow (\mathbf{c} - (-0.953 \cdot e + 128.8) \cdot \mathbf{n}_T, \mathbf{n}_T)$.
    Choose the one holding that the palm is further from the model
    (and outside):
    $(\mathbf{p}, \mathbf{o}) \leftarrow (p, o)_i$ :
    $dist(p_i, model) > dist(p_j, model),\ dist(p_i, model) > 0$.
```

Note that for the Barrett hand, a change in a proximal link implies a change in the corresponding distal link as well. Therefore, we use the linear estimation from [11] to compute $\mathbf{p}$, causing little loss in precision compared to calculating the actual inverse kinematics for the hand.

V. System Evaluation

As a suitable dataset to evaluate the algorithms described in this paper, we apply all models from the Princeton Shape Benchmark (PSB) [23]. Similar to [6], we rescale all 1,815 point-clouds and consider them as graspable toys to get a complete overview of the system's performance. To provide a reference for the efficiency, all processes are performed on a 2GHz dual-core processor laptop, running Ubuntu 9.04.

A. Point Cloud Interpolation-Decimation

The time used in the reduction of large point clouds clearly depends on the number of points n that we want to keep (see Fig. 3 left). The higher n, the more selective we delete and the lower the number of points we remove in each step. We found only one algorithm, the k-means by Huang [24], to result in similar performance. However, the time k-means needs to reduce to more than 1000 points is over 100 s. The time spent in the interpolation by using the mesh representation is negligible ($\ll 0.1s$) as it is done by simple triangle edge bisection.

B. Mesh Building

Also in Fig. 3 left, the time spent to mesh point clouds of different sizes is plotted. We found out that decimated point clouds with more than 3000 points do not result in better depicted meshes, thus Fig. 3 left only presents the range from 600 to 3000 points. As point cloud reduction and meshing take longer time as the number of points increases, we determine the minimum necessary size ensuring good meshing. By considering that the confidence of well meshed models does not improve when increasing the number of points, we empirically found 2000 points to be an optimum value for the PSB object models. Meeting this value in Fig. 3 center, we can infer that reduction and meshing of a point cloud can be done in around 1.7 s.

C. Shape Descriptors

A comparative study of descriptors, e.g. Fourier descriptors, curvature scale space descriptors, Zernike and grid descriptors, [25], encourages the use of Zernike moments. We evaluate the Zernike descriptors through an object classification experiment on the PSB. Following [7], we use precision-recall diagrams (see Fig. 3 right) to find a good combination (N_{voxels}, $O_{Zernike}$) $\in \mathbb{N} \times \mathbb{N}$ for retrieving the shape of a model. We use the diagrams to measure the ability of different combinations to separate sets of objects belonging to a shape class given only one object from a class. The quality measure we use for each class C is the integral of the normalized precision-recall diagram averaged over the members of C, P_o^C. The better the descriptors represent the models of a class, the closer this value to 1.

We selected 6 different classes from the PSB. The purpose of each experiment was to separate 2 or 3 sets of models composed of between 20 and 50 elements (each set), using 4 different voxelizations $N_{voxels} \in \{48, 64, 128, 256\}$ and 21 different Zernike orders $O_{Zernike} \in \{5, ..., 25\}$.

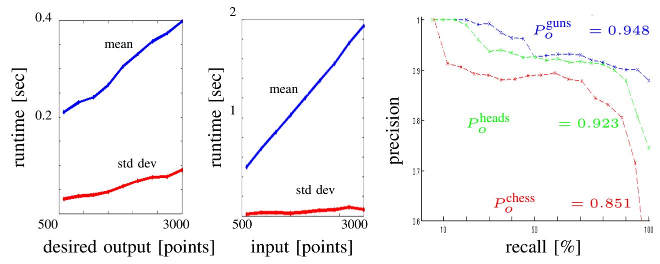


Fig. 3. Performance of point cloud filter (Left) and mesh building (Center). Right: Precision-recall diagram of three classes (guns, heads, chess) from the PSB. In this example, we use $N_{voxels} = 256$, $O_{Zernike} = 25$.

The experiments revealed a good behavior of the Zernike orders, obtaining an average P_o^C value around 0.9. The results suggest that complex shapes are better retrieved when using detailed voxelizations, $N_{voxels} = 256$, while simple objects got better results with simpler voxelizations, e.g. $N_{voxels} \in \{48, 64\}$. Nevertheless, the loss in P_o^C for making the opposite choice is less than 0.05. The optimal range of Zernike order was observed as $O_{Zernike} \in \{10, 11, 12\}$. Since it is thus not possible to get the optimum for both simple and complex shapes at the same time, and detailed voxelizations and high Zernike orders demand more calculations, it was decided to use $N_{voxels} = 64$ and $O_{Zernike} = 12$ (49 coefficients). The time spent to perform both processes with this configuration is 0.5 s.

D. Segmentation

The segmentation is the slowest process in our system. Though the mesh is calculated before-hand and core extraction is performed only once, the decomposition is repeated several times to find a variety of segmentations and choose the best one. We consider that testing 10 different scales of the core is sufficient to get a good segmentation in most of the cases. Obviously, including more scales could lead to a better result, but at the cost of computation time.

The lower the number of points to be handled, the faster the segmentation. This puts more emphasis on the cloud decimation described in II-B, limiting the time spent not only for meshing, but even more for segmentation. We found that the average number of segmented parts over the PSB in two test cases (a: <2000 points, b: ≥2000 points) is similar (a: 2.61, b: 3.34). Nevertheless, the convexity gain (the merit value corresponding to the difference between the convexity measures (2) of the original model and its decomposition) is much higher in the case of large point clouds (a: 0.12, b: 0.26). We associate this observation to the fact that simple objects are represented by lower number of points in the PSB database. Thus, the improvement reached through the segmentation in these models is lower. However, there is a clear difference in computation time (a: 2.93s, b: 6.38s).

E. Grasp Stability Estimation

Four main definitions of grasp quality are implemented according to [14]. Out of those we take a measure of grasp quality (eps_{L1}) that implies we are using a pessimistic criterion and considering one unit force distributed over all grasp points. We assume the friction coefficient μ to be 1, corresponding to a rubber coated hand grasping an object made of metal, glass, plastic or wood.

The percentage of force-closure among all 400 grasps found on each object is typically in the range of 75-80%. The average time for generating a triplet, evaluating its grasp quality and configure the Barrett Hand is 1, 2.8 and 0.2 milliseconds, respectively. This results in 1.6 s in the worst case, when we evaluate all candidate triplets. The eps_{L1} value of the best grasp is typically within the range of 0.12 to 0.16. Our acceptance threshold is set to 0.15.

VI. Using Experience in an Object-Grasp Database

An important aspect of our our work is to enable decisions if some process on the object data is reasonable or not. For example, if the object is convex, there is no need for segmentation. Our approach is based on a database in which each entry for an object O is composed of three fields:

$D(O)$,	a description of an object O; we will use its Zernike descriptor (Sec. V-C, the volume of its mesh, its main axes and center-of-mass (Sec. IV-B).
$T(O)$,	a triplet (Sec. IV-A) leading to a grasp on O.
$Q(O)$,	a measure of grasp quality (Sec. IV-A and V-E) connected to the triplet $T(O)$.

Given an object description $D(O)$, the system can search for similar models in the database using the shape descriptor. If a similar model O^* is found, the stored triplet $T(O^*)$ can be adapted to the new object size and orientation. Otherwise, a collection of triplets $\{T(O)\}$ will be created according to the morphology of O and the first $T(O)$ with quality $Q(O)$ exceeding an acceptance threshold will be selected.

Finally, a corresponding entry will be added into the local database. Optionally, the configuration of the robotic (Barrett) hand is computed to align the fingertips with the triplet. As the number of grasp requests grows, the local database becomes more complete and the likelihood of finding similar models stored increases. Since re-using a triplet takes less time than searching for force-closure-feasible triplets on the object surface, the average response time is reduced. We note that our database is hand-independent, since we do not store hand configurations, but triplets of contact points. The time spent to find a similar object in the database and adapt its triplet is negligible ($< 0.1\ s$).

In Tab. I, we show the average runtime of the major processes, considering 3 basic cases: 1) known objects, 2) unknown, simple objects, and 3) unknown, complex objects.

TABLE I

Average runtime of the different subtasks on PSB models.

Input model:	[sec]	case 1)	case 2)	case 3)
Denoising & Interp-Decim.	2.95+N	●	●	●
Meshing	1.3	●	●	●
Voxel. & Zernike	0.45	●	●	●
Segmentation	6.4			●
Triplet Search	1.6		●	●
Total [sec]		4.7+N	6.3+N	12.7+N

with $N = 10^{-6}$ * number of points [sec].

The input cloud is firstly randomly decimated to 10000 points (if it exceeds this size) in order to limit N. The table does not include processes with runtime $\ll 0.1\ s$: search in the database, adding entries or reuse of a triplet. As it can be seen, there is a big difference in timing between cases 1) and 2) with respect to case 3).

VII. Discussion and Conclusion

We have proposed a grasping framework capable of dealing with known and unknown objects considering the acquisition of a good 3D point cloud, the choice of an appropriate 3D shape representation and the management of the experience as the key aspects. The creation and update of a grasp database in order to gain experience is an important part of the system. The idea of building a grasp database is not new: the Columbia Grasp Database (CGDB) [6] is a most recent and attractive repository of grasps over a set of 3D models. Nevertheless, there are three main difference between the CGDB and the grasp database presented here:

Extension: While the CGDB covers a set of 7,256 models (the 1,814 models from the Princeton Shape Benchmark [23] cloned at four different scales), the database proposed here is constructed according to local experience. Only the models that have been processed and grasped by the system will be included in the database. Obviously, when searching for similar models, this results in a much shorter response time, noticeable saving in memory space, and links to strategies of active learning (and forgetting).

Specific nature: When an object is grasped once, it is likely that the same object or a similar one will be requested to be grasped again (e.g. a book). The first time a *specific* grasp will be generated for that specific object. The next times this object will be re-grasped using the same grasp again, and those objects which are similar will be grasped by adapting the original triplet. In the worst case, if an adapted grasp does not work on a new object, a new *specific* grasp will be generated for that object and a corresponding new entry will be added to the database.

Independence of object size: The grasp is not directly reused, the original triplet is *adapted* to the size (and to the orientation) of the new object instead. Next, the grasp is generated from the adapted triplet.

Regarding future work, there are several ideas to be explored. In this paper, we took into account the constraints given by the hand and the objects, but left out the constraints given by the task (e.g. hand-over, pouring, tool use); this could be included in the grasp hypothesis search. Secondly, in the presented framework we considered that the objects were pre-segmented from the scene. Our current work in stereo based segmentation will be integrated with the system for better point-cloud generation. Then, a path planning algorithm could be added to avoid collisions with the obstacles when approaching the object.

Acknowledgments

This work was supported by EU IST-FP6-IP-027657 PACO-PLUS, EU IST-FP7-IP GRASP and Swedish Foundation for Strategic Research (SSF).

References

[1] J. Bohg and D. Kragic, "Learning Grasping Points with Shape Context," *Robotics and Autonomous Systems*, 2009, in Press.

[2] C. Goldfeder, P. K. Allen, C. Lackner, and R. Pelossof, "Grasp Planning Via Decomposition Trees," in *IEEE International Conference on Robotics and Automation*, 2007, pp. 4679–4684.

[3] K. Huebner, S. Ruthotto, and D. Kragic, "Minimum Volume Bounding Box Decomposition for Shape Approximation in Robot Grasping," in *IEEE Int'l Conf. on Robotics and Automation*, 2008, pp. 1628–1633.

[4] L. Chevalier, F. Jaillet, and A. Baskurt, "Segmentation and Superquadric Modeling of 3D Objects," *Journal of Winter School of Computer Graphics, WSCG'03*, 2003.

[5] K. Huebner, K. Welke, M. Przybylski, N. Vahrenkamp, T. Asfour, D. Kragic, and R. Dillmann, "Grasping Known Objects with Humanoid Robots: A Box-based Approach," in *International Conference on Advanced Robotics*, 2009.

[6] C. Goldfeder, M. Ciocarlie, H. Dang, and P. K. Allen, "The Columbia Grasp Database," in *IEEE International Conference on Robotics and Automation*, 2009, pp. 3343–3349.

[7] M. Novotni and R. Klein, "3D Zernike Descriptors for Content Based Shape Retrieval," in *ACM Symposium on Solid and Physical Modeling*, 2003, pp. 216–225.

[8] R. Diankov, S. Srinivasa, D. Ferguson, and J. Kuffner, "Manipulation Planning with Caging Grasps," in *IEEE International Conference on Humanoid Robots*, 2008, pp. 285–292.

[9] S. El-Khoury and A. Sahbani, "On Computing Robust N-Finger Force-Closure Grasps of 3D Objects," in *IEEE International Conference on Robotics and Automation*, 2009, pp. 2480–2486.

[10] A. T. Miller, S. Knoop, H. I. Christensen, and P. K. Allen, "Automatic Grasp Planning Using Shape Primitives," in *IEEE International Conference on Robotics and Automation*, 2003, pp. 1824–1829.

[11] S. Geidenstam, K. Huebner, D. Banksell, and D. Kragic, "Learning of 2D Grasping Strategies from Box-based 3D Object Approximations," in *Proceedings of Robotics: Science and Systems*, 2009, pp. 9–16.

[12] M. Richtsfeld and M. Zillich, "Grasping Unknown Objects Based on 2 1/2 D Range Data," in *IEEE International Conference on Automation Science and Engineering*, 2008, pp. 691–696.

[13] E. Lopez-Damian, "Grasp Planning for Object Manipulation by an Autonomous Robot," Ph.D. dissertation, Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS, 2006.

[14] A. Miller and P. Allen, "Graspit! A Versatile Simulator for Robotic Grasping," *Robotics and Automation*, vol. 11 (4), pp. 110–122, 2004.

[15] S. Ary, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, "Algorithm for Approximate Nearest Neighbor Searching Fixed Dimensions," *Journal of the ACM*, vol. 45, no. 6, pp. 891–923, 1998.

[16] L. Giaccari, "MyRobustCrust – Surface Reconstruction from Scattered Point Clouds," **[URL]** http://www.mathworks.com/matlabcentral/fileexchange/22185, January 2010.

[17] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The Quickhull Algorithm for Convex Hulls," in *ACM Transactions on Mathematical Software*, vol. 22, 1995, pp. 469–483.

[18] R. Liu and H. Zhang, "Segmentation of 3D Meshes through Spectral Clustering," in *Computer Graphics and Applications, 12th Pacific Conference*, 2004, pp. 298–305.

[19] M. Attene, B. Falcidieno, and M. Spagnuolo, "Hierarchical Mesh Segmentation based on Fitting Primitives," *The Visual Computer*, vol. 22, no. 3, pp. 181–193, 2006.

[20] C. Borst, M. Fischer, and G. Hirzinger, "A Fast and Robust Grasp Planner for Arbitrary 3D objects," in *IEEE International Conference on Robotics and Automation*, vol. 3, 1999, pp. 1890–1896.

[21] ——, "Grasping the Dice by Dicing the Grasp," in *IEEE/RSJ Int'l Conference on Intelligent Robots and Systems*, 2003, pp. 3692–3697.

[22] W. T. Townsend, "The BarrettHand Grasper – Programmably Flexible Part Handling and Assembly," *Industrial Robot: An International Journal*, vol. 27, no. 3, pp. 181–188, 2000.

[23] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser, "The Princeton Shape Benchmark," in *International Conference on Shape Modeling and Applications*, 2004, pp. 167–178.

[24] Z. Huang, "Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values," *Data Mining and Knowledge Discovery*, vol. 2, no. 3, pp. 283–304, 1998.

[25] D. Zhang and G. Lu, "Content-Based Shape Retrieval Using Different Shape Descriptors: A Comparative Study," in *IEEE International Conference on Multimedia and Expo*, 2001, pp. 1139–1142.