Table of Contents

Ascending Stairway Modeling Live Demo: Toward Autonomous Multi-Floor Exploration

Jeffrey A. Delmerico, Jason J. Corso, David Baran, Philip David, and Julian Ryde

Abstract— **Localization and modeling of stairways by mobile robots can enable multi-floor exploration for those platforms capable of stair traversal. No system yet presented is capable of localizing a stairway on a map and estimating its properties, two functions that would enable stairways to be considered as traversable terrain in a path planning algorithm. The system we propose to demonstrate performs detection and modeling of ascending stairways while performing simultaneous localization and mapping. Our system consists of two parts: a computationally efficient detector that leverages geometric cues from depth imagery to detect sets of ascending stairs, and a stairway modeler that uses multiple detections to infer the location and parameters of a stairway that is discovered during exploration. Modeling the stairway as a whole will enable exploration of higher floors of a building by allowing the stairway to be incorporated into path planning by considering it as a portal to new frontiers. Our intended demonstration will highlight the performance of the system in accurate stairway modeling and localization by observing with a mobile robot a set of portable stairs that we will place in the demo area.**

I. Introduction

Autonomous mobile robots have traditionally been restricted to single floors of a building or outdoor areas free of abrupt elevation changes such as curbs and stairs. This restriction presents a significant limitation to real-world applications, such as whole-building mapping and rescue scenarios. Our work seeks a solution to this problem and is motivated by the rich potential of an autonomous ground robot that can climb stairs while exploring a multi-floor building. Our proposed solution to this problem is a system to detect and localize stairways in the environment during the process of exploration, and model any identified stairways in order to determine if they are traversable by the robot; an overview is presented in Fig. **??**. With a map of the environment and estimated locations and parameters of the stairways, the robot could plan a path that traverses the stairs in order to explore the frontier at other elevations that were previously inaccessible. Other systems have been proposed for related tasks—primarily a single detection triggering immediate traveral—but no existing work approaches the problem in the context of the aforementioned scenario. Autonomous multi-floor exploration is a new behavior for ground robots, and we present this work as a first step toward the realization of that capability.

Although our contribution is mainly toward semantic object perception, and might therefore be more applicable to the SPME workshop at ICRA 2012 instead, we believe that the work is best illustrated with a live demonstration—which is not offered there—and that the intended application is more in line with the higher-level semantic mapping themes of the SPMK workshop.

J. Delmerico, J. Corso, and J. Ryde are with SUNY Buffalo {jad12, jcorso, jryde}@buffalo.edu

D. Baran and P. David are with the U.S. Army Research Laboratory {david.g.baran.civ, philip.j.david4.civ}@mail.mil

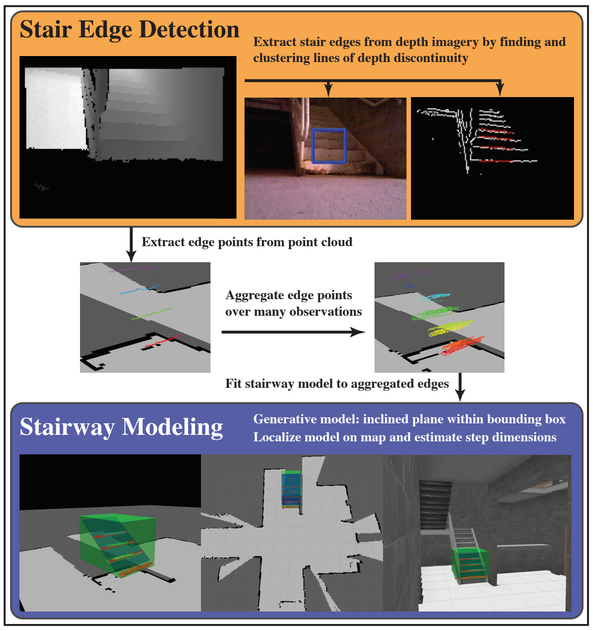

Stair Edge Detection
Extract stair edges from depth imagery by finding and clustering lines of depth discontinuity
Extract edge points from point cloud
Aggregate edge points over many observations
Fit stairway model to aggregated edges
Stairway Modeling
Generative model: inclined plane within bounding box
Localize model on map and estimate step dimensions

Fig. 1. High level workflow of the proposed system, consisting of two modules: stair edge detection and stairway modeling. Stair edges are extracted from depth imagery and collected over many observations into an aggregated point cloud. Periodically, a generative model of a stairway is fit to the aggregate cloud and its parameters re-estimated. The result is a model localized with respect to the robot's map of its environment.

II. Demonstration

We intend to demonstrate this system by showing it in live operation while deployed on a mobile robot. We plan to provide our own robot platform, as well as a set of portable demonstration stairs—several full-sized steps of our own construction—in order to facilitate the illustration of its capabilities. However, if it is possible to use any stairways within the building, we would prefer to demonstrate the system in a more natural setting. Under teleoperation, the robot will navigate around the provided demonstration area and construct a map while detecting and modeling the stairway, and localizing the model within the map. We can demonstrate the accuracy of localization and model parameter estimation, as well as the convergence of the model over multiple observations.

Task-based World Model Verification

Jos Elfring   Sjoerd van den Dries   René van de Molengraft   Maarten Steinbuch

Abstract— **In order to fulfill typical household tasks, such as fetching objects, robots need an accurate description of their environment. Maintaining such a description, called world model, requires (i) monitoring the environment for new objects and (ii) updating object attributes in the world model whenever needed. The focus in this work is on the latter of these two subtasks. We present a generic framework that can be used to keep a world model up-to-date. In addition, we present a task dependent strategy that coordinates when to update which object attributes. The task-based coordination strategy enables robots to perform *efficient* world model verification. We present a first the proof-of-concept experiment.**

I. INTRODUCTION

As domestic robots are moving towards human populated environments, they are confronted with unstructured and dynamically changing environments. In order to fulfill typical household tasks, such as fetching objects, an accurate description of the environment is indispensable. In this work, we will refer to such an environmental description as world model. We believe that a world model should at least contain information about poses of semantically labeled objects, but ideally contains information about other object attributes as well, such as the color distribution, size or shape.

Once such a world model is obtained, it is important to maintain it, *i.e.*, to keep it up-to-date. Keeping a world model up-to-date involves (i) monitoring the environment for new objects and (ii) updating the current object attributes whenever this is needed. This paper focuses on the latter of these subtasks. Especially if the number of objects in the world model becomes large, as in realistic scenarios, the maintenance of a world model can be cumbersome. In order to allow successful maintenance in these realistic scenarios, we believe various requirements have to be met.

Closing the loop from world model to perception should allow a fast and simple verification module, since the prior knowledge provided by the world model simplifies the task. If the perception system for example has to check if 'that black coat is still on the chair', the prior knowledge changes the task of 'recognizing a black coat in a camera image' to 'verifying if there still is a black blob at the location of the coat in the camera image' (Requirement 1).

Secondly, the world modeling algorithm has to be flexible regarding the object attributes. Robots should move towards semantically richer world models. This means we do not only want to be able to store or track positions of objects, but also, *e.g.*, color distributions or shapes. (Requirement 2).

All authors are with the Faculty of Mechanical Engineering, Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands {j.elfring, s.v.d.dries, m.j.g.v.d.molengraft, m.steinbuch} at tue.nl

The coordination of attribute verification must be driven by the task the robot has to perform. The world model may contain many objects with many attributes but only a subset of all objects and object attributes is relevant in the context of the task. While picking up a bottle, the position attribute of the bottle has to be very accurate whereas other object attributes are allowed more uncertainty. If a robot is waiting for a pancake to be ready it is useless to verify the location of the pancake at tens of Hertz; if the robot has to catch a ball, updating the ball's position at tens of Hertz is needed to allow the successful accomplishment of the task. Our last requirement is a strategy that both efficiently, *i.e.*, with a proper update rate, and effectively, *i.e.*, the relevant attributes only, coordinates when to update which attributes (Requirement 3).

Closing the loop from world model to perception as described in Requirement 1 is not new. In [1], adaptive appearance manifold models are learned online with the aim of improving the robustness of the detection algorithm. In [2], non-rigid objects are detected using classifiers that are updated based on the detections and [3] detects objects using classifiers that are learned based on detections. We present a generic framework that on the one hand allows accommodating these methods and on the other hand enables facilitating computationally cheap algorithms that can be used to verify attributes such as color or position.

Both in tracking and perception literature, many different attributes have been tracked such as positions and velocities [4], color distributions [5], [6] or shapes [7], hence solutions to Requirement 2 are extensively presented in existing literature. In this work, we do *not* aim at presenting new attribute trackers but rather at introducing a *generic framework* that allows accommodating any kind of tracker and a strategy that provides these trackers with input data whenever this is needed.

As stated in Requirement 3, we believe a strategy that coordinates *which* specific attributes of *which* specific objects are updated *when* is inevitable for both efficient and effective updating of the world model. Previous work based on similar ideas is scarce. In [8], [9], the methods used for state estimation are varied, *e.g.*, more robust but less accurate tracking algorithms if the conditions deteriorate and precise tracking if the conditions are good [9]. We are interested in varying coordination strategies rather then estimation methods, *i.e.*, *when* to update *what* rather then *how* to update. Furthermore, visual object search approaches [10], [11] use attention to *find* objects in a cluttered or unknown environment, *i.e.*, they focus on interesting parts of the environment only. We aim at task-based *verification* of object attributes in a, at least partially, known environment.

II. CONTRIBUTIONS

We believe that Requirements 1–3 are crucial in order to allow both efficient and effective maintenance of a world model. Current literature seems to either focus on detecting new objects or updating single attributes only. Maintaining a world model is broader than that. Efficient attribute verification is needed, which in turn requires both a generic framework that accommodates estimating various kinds of attributes and a task-based strategy that coordinates the verification. Therefore, the main contributions of this work are:

- A *generic framework* that allows both tracking and verifying any object attribute using simple perceptual routines that are informed by the world model
- A task-based coordination strategy that only updates *relevant* attributes of *relevant* objects in the world model
- Validation in a first proof-of-concept experiment

The remainder of this paper is organized as follows. In Section III the architecture of the proposed framework is explained together with the representation of objects and measurements. Then Section IV focuses on the implementation details and presents experimental results. Section V summarizes the conclusions and presents a possible direction for future research.

III. ARCHITECTURE AND REPRESENTATION

This section explains the architecture of the framework proposed in this work and shown in Figure 1. Throughout the Sections III-A–III-F, we explain the various components and their interfaces.

A. World model

The world model as it is represented in this paper contains a list of objects. Each object is represented by a collection of both continuous object attributes, *e.g.*, position, and discrete object attributes, *e.g.*, class label. The collection of object attributes describes the object state at the instance level and is represented by a probability distribution over the attribute space. One of the main tasks of the world modeling algorithm is data association: associating measurements with objects that are present in the world model already or with new objects or clutter (false positives). We use an anchoring approach [12] that incorporates multiple hypothesis tracking based data association [13]. Further details regarding the anchoring and data association algorithm are beyond the scope of this paper.

Once the data association problem is solved, the object attributes are updated based on the measurements. Typically, the trackers described in the introduction can be used for this. We use a Bayesian framework to recursively refine our state estimates in a predict–update cycle that incorporates multiple models [14] and explicitly takes uncertainties into account. Any object attribute estimator can be accommodated and further details, again, are beyond the scope of this paper.

The probabilistic models and the settings used for anchoring, tracking and data association are loaded from a knowledge base, further explained in Section III-B. The measurements used to update the attributes are generated in the perception module explained in Section III-E and are represented as explained in Section III-F.

B. Knowledge base

A knowledge base provides the tracking and data association algorithms mentioned above with the required prior knowledge, *i.e.*, models enabling prediction and probabilistic models used during association. Representing this knowledge in a separate knowledge base rather then incorporating it in the world model simplifies configuring the world modeling algorithm, *e.g.*, setting different state estimators for objects. The knowledge is represented by means of an XML-file.

C. Verification coordination

The aim of the coordination module is to control the object verification. Based on the task, a list of objects related to that task can be specified and loaded to the verification coordination module. Together with each object, a maximum allowed uncertainty for the relevant object attributes is loaded.

Propagating the object attributes in the world model to a desired point in time increases the uncertainty, updating the attribute using a measurement decreases the uncertainty. Based on the maximum allowed uncertainty and the uncertainty in the world state provided by the world model, it is decided which attributes of which objects need to be updated. Currently, the list of relevant objects and attributes per task are hand crafted. An interesting direction for future work would be to let the robot autonomously deduce this information from a common sense knowledge base.

D. Verification tasks

If the uncertainty in a predicted object attribute value is too large, an attribute verification has to be performed, as explained in Section III-C. In that case, the predicted attribute value together with both a 2D and a 3D region-of-interest (ROI) in the sensor data is sent to the perception module. The size of a ROI is based on the size of the object and the uncertainty on the predicted position. We assume that the predicted attribute value together with the ROIs enables using computationally cheap algorithms for the verification task. If this assumption is not met, the rational behind the proposed verification strategy does no longer hold and more advanced detection algorithms are required.

E. Perception

The perceptual system consists out of two parts. The first part detects object attribute values in raw sensor data provided by the robot's sensors. As a result this component detects and recognizes both new objects and objects that were detected before without using prior world model knowledge. Any algorithm can be used for this part of the perceptual task and we will not go into any details since this is not relevant in the context of this paper.

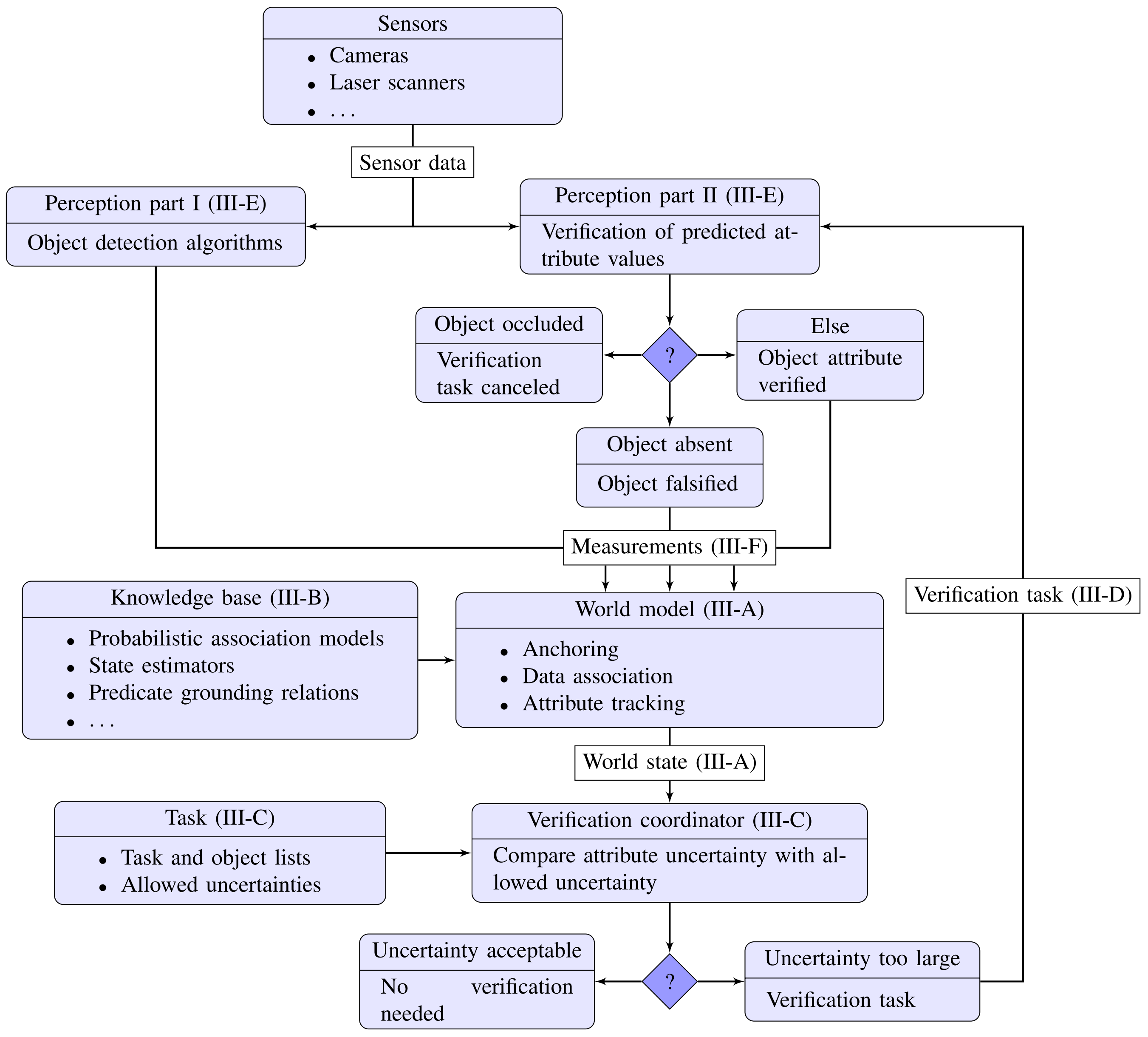


Fig. 1. Schematic overview of the proposed architecture. The numbers refer to the section explaining the components in more detail.

In the second part, the predicted object attribute values contained in the verification task are verified using a light weight algorithm. In this work, typical attributes of interest are position and color. An example of a light weight verification component is presented in Section IV-B.

Possible complicating factors in the second part of the perception module are:

1) The object is absent in the ROI in the sensor data: If an object cannot be detected in the ROI, this information is fed into the world model, since the absence of an object is valuable information in itself. The probability of the object still being present will be decreased based on such observations. We will refer to this situation as object falsification.

2) The object is occluded: Occlusions can be detected using depth information. If an object is occluded, no verification is possible and no falsification is allowed. With the emerging presence of cheap 3D sensors such as the Microsoft Kinect, we do not see the availability of depth information as a restrictive assumption.

3) Visually identical objects fall in the same ROI: In this case the ROIs will be merged and the verification problem includes a clustering problem. However, the number of clusters and reasonable initial guesses for the cluster centers are provided by the world model. If visually identical objects that are not present in the world model fall inside the ROI of an object that needs verification, our approach will result in inaccurate updates. More advanced detection algorithms, *e.g.*, the ones used in the first part of the perceptual system, are required in that case.

F. Measurements

We distinguish between three different types of measurements, represented by the three arrows entering the world model block from above in Figure 1. From left to right these three types of measurements are:

1) Non-verification measurements from part one of the perception module. These measurements can either instantiate new objects in the world model, update attributes of existing objects or be neglected as being false positives.
2) Falsification measurements from part two of the perception module. Falsification measurements will be used to remove objects from the world model.
3) Verification measurements from part two of the perception module. These measurements include verified object attributes and will *only* be used to update attributes of existing objects in the world model.

In this work we want the measurements to be represented by probability distributions. The rational is that we do not want to loose information about the uncertainty of the sensor and the detection algorithm. In case of comparing the sensor data with a set of object models, the matching score contains more information than the truncated result. Some example measurements could be:

$$z_{class} = \{cup \ : \ 0.7; \ mug \ : \ 0.3\}$$

for a discrete attribute measurement of the class label being cup with a probability of 0.7 or mug with a probability of 0.3, or:

$$z_{pos} = \mathcal{N}\left(x \mid \mu, \Sigma\right)$$

if the measured continuous position attribute can be described by a Gaussian with mean μ and covariance Σ. The Bayesian world modeling algorithm introduced in Section III-A exploits the trust the perception algorithm has in its estimates and the available knowledge regarding sensor characteristics such as noise and accuracy.

IV. EXPERIMENTS

This section presents the experiment that has been performed to validate the ideas presented in the previous sections. We start by explaining the scenario in Section IV-A. Then we explain all relevant details regarding the implementation in Section IV-B. After that, Section IV-C presents the results.

A. Scenario

In the experiment, the AMIGO robot is asked to enter a room. When AMIGO enters, he starts verifying a world model that is assumed to be available from a previous trail. Object attribute value uncertainties reduce and objects get falsified.

Then AMIGO is invited to play the game of cups. During this game, a ball is placed underneath one out of three identical cups. An operator starts shuffling the cups around and afterwards AMIGO has to tell the operator the location of the ball. The initial locations of the green cups used during the game are already verified during the first stage of the experiment hence AMIGO knows where to go. During the game, a very low position uncertainty is required to avoid mixing the identities of the visually identical cups. Other objects are permitted a higher uncertainty, since the object attributes are less relevant in the context of the game.

B. Implementation

This section elaborates all implementation details that are considered relevant for the scenario that is described in Section IV-A.

1) When to update which object attributes: The verification coordination component described in Section III-C and shown in Figure 1 takes as an input a task and the world state according to the robot's world model. To keep things well-presentable, we have defined two task only: playing-cups and room-exploration. Both tasks are defined in a single XML-file represented by the task block in Figure 1, that in addition describes the maximum allowed uncertainty for the limited set of object attributes that are considered relevant for this state. World model objects that are not defined are permitted a default uncertainty.

2) Cup position tracking: All objects, except for the three green cups, are tracked using a Kalman filter (KF) with a constant position motion model. The green cups are tracked using a KF with a constant velocity motion model. All KFs have a probability of correctly describing the position of the object. This probability increases after associating the object with verification or non-verification measurements and decreases after associating the object with falsification measurements.

3) Verifying attributes during the cups game: During the game, the positions of the green cups are re-detected using a light weight color detection algorithm:

1) Based on the color and the predicted position, an HSV color range and both a 2D and a 3D ROI are defined
2) All pixels that fall within both the ROIs and the color range are collected into a cluster
3) The center of this cluster is calculated in the 2D image
4) The 2D cluster center is transformed to a 3D position which is fed back to the world model

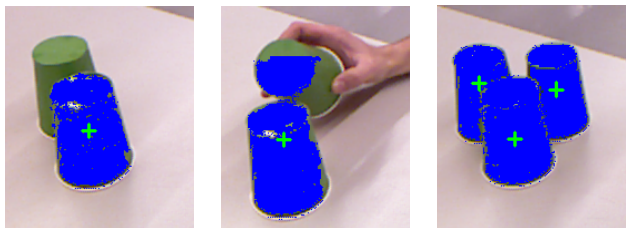
The prior information provided by the world model simplifies recognition of the cups to detecting green blobs.

During the game, the ROIs of the cups intersect regularly and as a result, pixels can be part of multiple ROIs. In that case, step two changes to:

2) Pixels that can originate from multiple objects are clustered using k-means clustering

The initial centers for the k-means clustering are the predicted 3D object locations projected onto the 2D image. As a result, the clustering converges in one or two iterations only. During the experiment, the computation time for the verification algorithm is in the order of one millisecond.

In Figure 2, three examples outputs of the verification algorithm are given. The blue pixels represent pixels that fall in the 2D and 3D ROIs and the color range of at least one cup, the green crosses represent cluster centers generated by the k-means clustering. In Figures 2(a) and 2(b) the cup on the foreground is the only object present in the world model, hence no k-means clustering is needed. The other cup is held close to disturb the verification algorithm. In Figure 2(a) the depth information still allows correct clustering. In Figure 2(b), a subset of the pixels belonging to the distracting cup on the background are grouped in the foreground cup cluster. As a result, the cluster center is off. Finally, Figure 2(c) shows how the clustering succeeds in a scenario with three cups that are present in the world model.



(a) (b) (c)

Fig. 2. Three scenarios showing the performance of the k-means clustering In (a) and (b) the world model contains one object, hence $k = 1$, and one disturbing object is placed nearby. In (c), all three cups are contained in the world model, hence $k = 3$. The blue pixels are the pixels that have to be clustered, the green crosses are the resulting cluster centers that will be fed back to the world model.

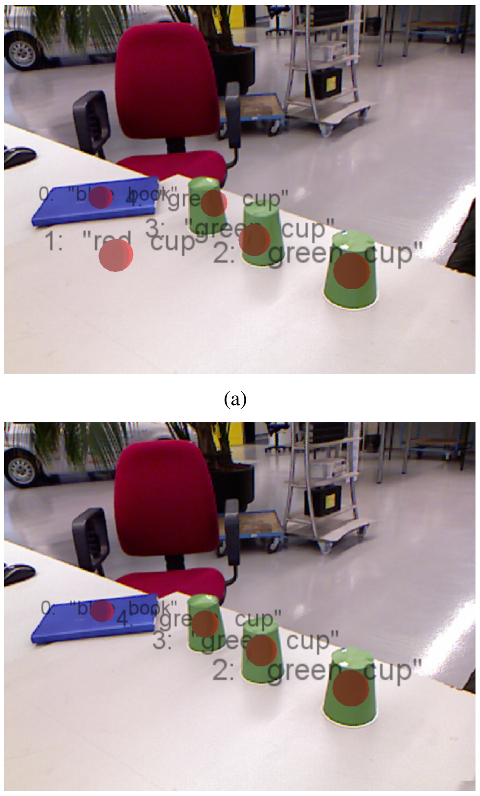
4) Object falsification: During k-means clustering, clusters might get empty. Furthermore, the ROIs might not contain pixels within the color range provided by the world model again resulting in an empty cluster. In the case of empty clusters two explanations are plausible. Either the object is not present within the ROI or the object is occluded by another object. In the experiment, the depth information is used for occlusion checking. If occlusions cannot explain the empty clusters, object falsification measurements are published to the world model.

C. Results

Figure 3(b) shows the robot's view during the experiment together with a world model overlay. The red spheres indicate the positions of the objects in the world model, the text labels show the class label, the color and a unique ID.

When AMIGO arrives at the table, the world model contains a blue book, a red cup and three green cups as shown in Figure 3(a). As can be seen in this figure, the world model is outdated, *i.e.*, some of the positions are off and the red cup is absent. After verification, the red cup is correctly falsified and the other positions are updated. The updated world model is visualized in Figure 3(b).

Figure 4 shows the variances in the x-position of the objects in the world model during the verification that was shown in Figure 3. In the first stage of the experiment, all objects on the table are allowed a variance of 0.001 m^2. The red line belongs to the red cup. Around $t = 18$, the uncertainty of the red cup is larger than allowed and a verification task is given. During the verification, the red cup is correctly falsified. The blue line shows the variance



(a)

(b)

Fig. 3. AMIGO's camera view with world mode overlay when arriving at the table in (a) and after verifying the world model in (b)

belonging to the book's x-position. During propagation of the book position, the uncertainty increases and as a result, every four or five seconds the position needs verification. Due to the verification, the position can be updated and the variance decreases to an acceptable level. The green line shows the variance of one of the green cups. Initially, the variance behaves similar to the variance of the book. However, around $t = 38$ the task switches from room-exploration to playing-cups. As a result, the allowed variance of the green cups decreases and the verification tasks are sent about 28 times a second.

The positions of the green cups during the game are shown in Figures 5 and 6. Due to the high verification frequency the robot is able to keep track of all three cups without mixing identities despite occlusions and fast cup movements.

V. CONCLUSIONS AND FUTURE WORK

In domestic robotics, robots need a world model that describes the current state of the world the robot is operating in. Maintaining a world model requires (i) monitoring the environment for new objects and (ii) verifying the states of objects that are already included in the world model. The first of these subtasks is well-studied whereas the second one is not studied as a problem per sé. This work focuses on the latter of these two subtasks.

First of all we have presented and implemented a generic framework that allows verifying any object attribute. In

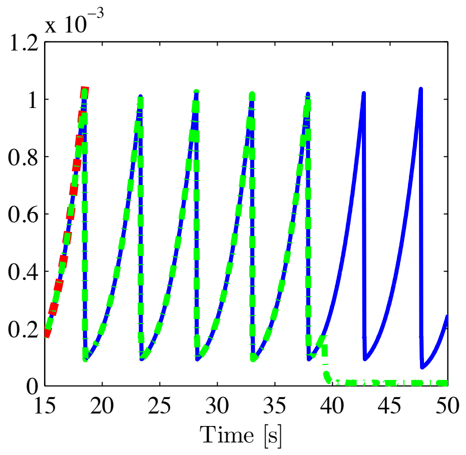
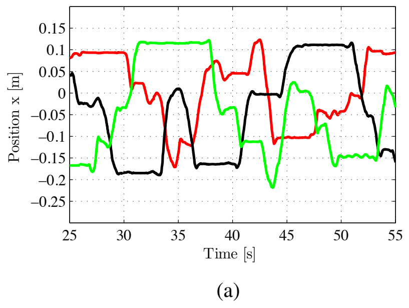


Fig. 4. Variance in the x-position for the book (black), the falsified red cup (blue) and the green cups (red).



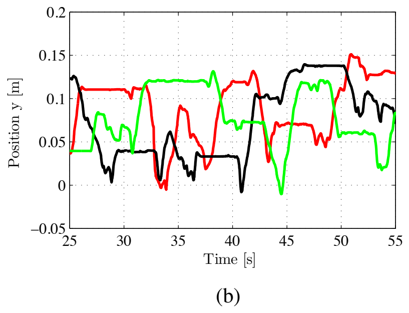


Fig. 5. Positions of the three green cups during the game as a function of time. The x-positions in (a) and the y-positions in (b).

addition, we have presented a first task-based strategy that coordinates when to update which object attributes. The task dependency has led to both efficient and effective world model verification. The results of a first proof-of-concept experiment were presented.

Future work consists of adding verification of different attributes using different verification algorithms. This could further proof the potential of our framework. In addition we will investigate the possibility of letting the robot autonomously deduce which attributes are relevant in the context of the task from a common sense knowledge base.

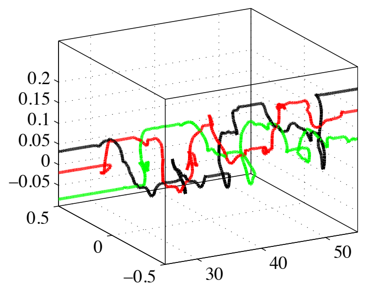


Fig. 6. The (x, y)-positions of the three green cups during the game as a function of time.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Union Seventh Framework Program FP7/2007-2013 under grant agreement no. 248942 RoboEarth.

REFERENCES

[1] J. Ding, Y. Huang, K. Huang, and T. Tan, "Robust Object Tracking via Online Learning of Adaptive Appearance Manifold", *2011 IEEE Int. Conf. on Computer Vision Workshops*, pp 1863–1869, 2011.

[2] M. Godec, P.M. Roth, and H. Bischof, "Hough-based Tracking of Non-Rigid Objects", In: *Proc. 2011 IEEE Int. Conf. on Computer Vision*, pp 81–88, 2011.

[3] Z. Kalal, J. Matas, and K. Mikolajczyk, "P-N Learning: Bootstrapping Binary Classifiers by Structural Constraints", In: *Proc. 23rd IEEE Conference on Computer Vision and Pattern Recognition*, pp 49–56, 2010.

[4] J. Elfring, M.J.G. van de Molengraft, R.J.M. Janssen, and M. Steinbuch, "Two Level World Modeling for Cooperating Robots Using a Multiple Hypotheses Filter", In: *2011 IEEE Int. Conf. on Robotics and Automation*, pp 815–820, 2011.

[5] K. Nummiaro, E. Koller-Meier, and L. van Gool, "An adaptive color-based particle filter", *Image and Vision Computing*, 21(1), pp 99–110, 2003.

[6] L. Sigal, S. Sclaroff, and V. Athitsos, "Skin Color-Based Video Segmentation under Time-Varying Illumination", *IEEE Trans on Pattern Analysis and Machine Intelligence*, 26(7), pp 862–877, 2004.

[7] T. De Laet, Rigorously Bayesian Multitarget Tracking and Localization, *Ph. D. thesis*, Katholieke Universiteit Leuven, 2010. ISBN: 978-94-6018-209-9.

[8] J.L. Crowley and F. Berard, "Multi-Modal Tracking of Faces for Video Communications", In: *1997 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, 1997*, pp 640–645, 1997.

[9] K. Toyama and G.D. Hager, "Incremental Focus of Attention for Robust Vision-Based Tracking", *Int. Journal of Computer Vision*, 35(1), pp 45–63, 1999.

[10] P. Forssén, D. Meger, K. Lai, S. Helmer, J.J. Little, D.G. Lowe, "Informed Visual Search: Combining Attention and Object Recognition", In: *2008 IEEE Int. Conf. on Robotics and Automation*, pp 935–942, 2008.

[11] K. Shubina and J.K. Tsotsos, "Visual search for an object in a 3D environment using a mobile robot", *Computer Vision and Image Understanding*, 114(5), pp 535–547, 2010.

[12] S. Coradeschi and A. Saffiotti, "An Introduction to the Anchoring Problem", *Robotics and Autonomous Systems*, 43(2–3), pp 85–96, 2003.

[13] D.B. Reid, "An Algorithm for Tracking Multiple Targets", *IEEE Transactions on Automatic Control*, AC-24(6), pp 843–854, 1979.

[14] X.R. Li and V.P. Jilkov, "Survey of Maneuvering Target Tracking. Part V: Multiple-Model Methods", *IEEE Transactions on Aerospace and Electronic Systems*, 41(4), pp 1255–1321, 2005.

Segmentation of Cluttered Scenes through Interactive Perception

Karol Hausman, Christian Bersch, Dejan Pangercic, Sarah Osentoski, Zoltan-Csaba Marton, Michael Beetz
{hausman, pangercic, marton, beetz}@cs.tum.edu,
christian.bersch@googlemail.com, sarah.osentoski@us.bosch.com

I. Introduction

For robot to perform its tasks competently, robustly and in the right context it has to understand the course of its actions and their consequences. For example, imagine the robot being tasked with the clean up of the breakfast table. The robot is confronted with a heavily cluttered scene and has to be able to tell waste, dirty, clean and valuable objects apart. The robot shall be equipped with the knowledge that will, for instance, stop it from throwing away an expensive item. Herein proposed approach elevates robot's perception skills in that it utilizes its capabilities to interact with the clutter of objects. This allows for better segmentation and finally also better object recognition by means of constraining the recognition to a region or regions of interest.

Similar to Katz et al. [1] and Bergstrom et al. [2], we propose a system that uses a robot arm to induce motions in a scene to enable effective object segmentation. Our system employs a combination of the following techniques: i) estimation of a contact point and a push direction of the robot's end effector by detecting the concave corners in the cluttered scene, ii) feature extraction using features proposed by Shi and Tomasi and tracking using optical flow, and iii) a novel clustering algorithm to segment the objects.

Segmentation of rigid objects from a video stream of objects being moved by the robot has been addressed by Fitzpatrick [3] and Kenney et al. [4]. In contrast, our arm motion is not pre-planned but adapts to the scene, we make use of the 3D data to segment the object candidates from the background and we use a novel clustering approach for the segmentation of textured objects.

Overview of the whole system is shown in Fig. 2. The system will be demostrated live during the workshop.

II. Estimation of Contact Point and Push Direction

Since most commonly encountered household items have convex outlines when observed from above, our system uses local concavities in the 2D contour of an object group as an indicator for boundaries between the objects. The robot separates objects from each other by pushing its end effector in between these boundaries.

A. Contact Points from Concave Corners

We restrict the problem of finding a contact point to the table plane. Our algorithm employs 2D-image processing techniques to select contact point candidates. The table plane is estimated from the depth-camera's point cloud data using RANSAC and separated from the object points. The remaining cloud points are projected into a virtual camera view above the table. Since the projected cloud points are sparse, we employ standard morphological operators and 2D-contour search to identify a closed region, R, corresponding to the group of objects.

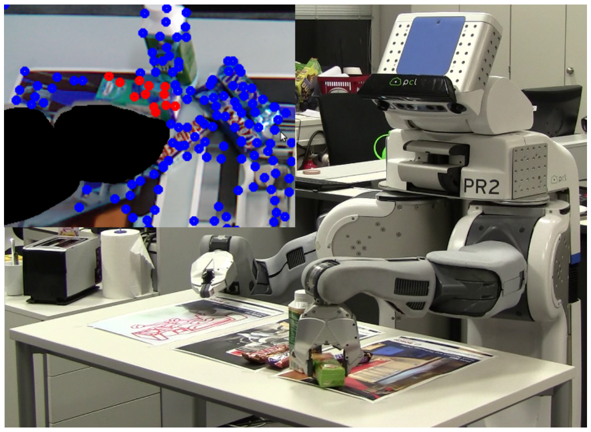


Fig. 1. Top: PR2 robot successfully picking-up the object after segmenting in it in clutter using herein proposed object segmentation algorithm.

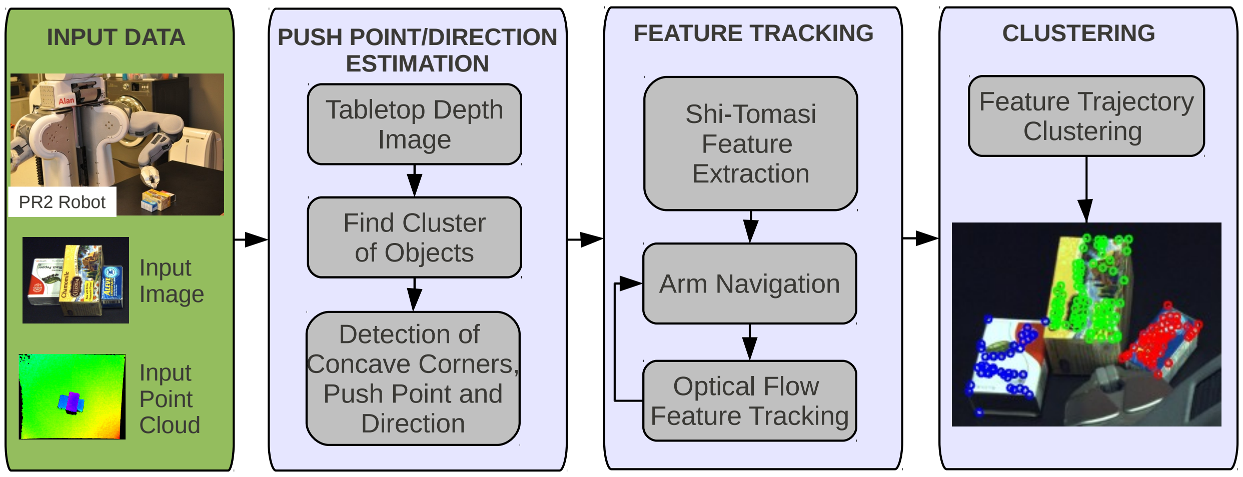
This region's outer contour is then searched for strong local directional changes by applying a corner detector and subsequently the corners that are placed at local concavities are selected.

B. Push Direction and Execution

The push direction at a corner is set to be parallel to the eigenvector corresponding to the larger eigenvalue of the Shi-Tomasi covariance matrix. Intuitively, the dominant eigenvector will align with the dominant gradient direction. However, at a corner with two similar gradient responses in two directions, the eigenvector becomes the bisector. As only corners with roughly equal eigenvalues are chosen as potential contact point candidates, the eigenvector of each contact point candidate will bisect the angles of the contour at the corner location.

III. Object Segmentation using Feature Trajectories

Once the robot′s end effector touches the objects, the resulting object motions are used to discriminate between the different items on the table. Feature points are tracked in the scene and the resulting feature point trajectories are



INPUT DATA
PR2 Robot
Input Image
Input Point Cloud
PUSH POINT/DIRECTION ESTIMATION
Tabletop Depth Image
Find Cluster of Objects
Detection of Concave Corners, Push Point and Direction
FEATURE TRACKING
Shi-Tomasi Feature Extraction
Arm Navigation
Optical Flow Feature Tracking
CLUSTERING
Feature Trajectory Clustering

Fig. 2. The system proposed in the paper consists of three main nodes: a node for estimating the initial contact point and the push direction, a node that extracts 2D-features and tracks them while it moves the robot arm in the push direction, and finally an object clustering node that assigns the tracked features to objects.

clustered. The clustering is based on the idea that features corresponding to the same objects must follow the same translations and rotations.

A. Feature Trajectory Generation using Optical Flow

We take advantage of the objects' texture properties by extracting $i = 1...N$ Shi-Tomasi features at the pixel locations $\{\mathbf{p}_{i,0}\}_{i=1}^{N}$ from the initial scene at time $t = 0$, i.e. before an interaction with the robot took place. The feature locations correspond to responses of the Shi-Tomasi feature detector. When the robot's end effector interacts with the object, a Lucas-Kanade tracker is used to compute the optical flow of the sparse feature set. Using the optical flow, each feature's position $\mathbf{p}_{i,t}$ is recorded over the image frames at time $t = 0...T$ while the robot is interacting with the objects. That is, for each successfully tracked feature i, a trajectory $S_i = \{\mathbf{p}_{i,t}\}_{t=0}^{T}$ is obtained.

B. Randomized Feature Trajectory Clustering with Rigid Motion Hypotheses

After calculating the set of all feature trajectories $\mathcal{S} \equiv \{S_i\}_{i=1}^{N}$, the goal is to partition this set such that all features belonging to the same object are assigned the same object index $c_i \in \{1,..,K\}$, where the number of objects K is not known *a priori*.

We take advantage of the rigid body property of objects and assume that each subset of the features trajectories $\mathcal{S}$ belonging to the same object k are subjected to the same sequence of rigid transformation $A_k \equiv \{\mathbf{A}_{k,t}\}_{t=0}^{T-1}$, i.e. we cluster features with respect to how well rigid transformations can explain their motions. As the objects only move on the table plane, we restrict a possible rigid transformation $\mathbf{A}$ to be composed of a 2D-rotation $\mathbf{R}$, a 2D-translation $\mathbf{t}$ and a scaling component s, i.e. $\mathbf{A} = s \cdot [\mathbf{R}|\mathbf{t}]$. The scaling component compensates for the changes in size of the projected objects in the camera image. The actual scaling

Algorithm 1: Randomized feature trajectory clustering

```
1  Input: Set of feature trajectories $\mathcal{S} \equiv \{S_i\}_{i=1}^{N}$ where
   $S_i = \{\mathbf{p}_{i,t}\}_{t=0}^{T}$
2  Output: object cluster count $K$, object cluster
   assignments $\mathbf{c} = [c_i]_{i=1}^{N}$ where $c_i \in \{1,..,K\}$
3  for $m := 1$ to $M$ do
4      $k_m := 1$, $\mathcal{S}_m := \mathcal{S}$
5      while $|\mathcal{S}_m| \geq 2$ do
6          draw 2 random trajectories $S_u, S_v \in \mathcal{S}_m$
7          generate sequence of rigid transformations:
           $A_{k_m} \equiv \{\mathbf{A}_{k_m,t}\}_{t=0}^{T-1}$ from $(S_u, S_v)$
8          for $S_j$ in $\mathcal{S}_m$ do
9              sum squared residuals w.r.t to $A_{k_m}$:
               $r_{k_m,j} := \sum_{t=0}^{T-1} \|\mathbf{p}_{j,t+1} - \mathbf{A}_{k_m,t}\mathbf{p}_{j,t}\|_2^2$
10             if $r_{k_m,j} < THRESHOLD$ then
11                 $\mathcal{S}_m := \mathcal{S}_m \setminus \{S_j\}$
12         $k_m := k_m + 1$
13     $K_m := k_m$
14     for $S_i$ in $\mathcal{S}$ do
15         Assign each trajectory to best matching rigid
           transformation sequence:
           $c^*_{m,i} := \text{argmin}_{\{1,..,k_m,..,K_m-1\}} r_{k_m,i}$, where
           $r_{k_m,i} := \sum_{t=0}^{T-1} \|\mathbf{p}_{i,t+1} - \mathbf{A}_{k_m,t}\mathbf{p}_{i,t}\|_2^2$
16 Select best overall matching set of rigid transform
   sequences: $m^* := \text{argmin}_m \sum_{k_m=1}^{K_m} \frac{\sum_i r_{k_m,i} \cdot \mathbf{1}_{[c^*_{m,i}=k_m]}}{\sum_i \mathbf{1}_{[c^*_{m,i}=k_m]}}$
17 Return: $K := K_{m^*}$, $\mathbf{c} := \left[c^*_{m^*,i}\right]_{i=1}^{N}$
```

is not linear due to the perspective view, however, the error resulting from this linearization is small as the objects are displaced only in small amounts.

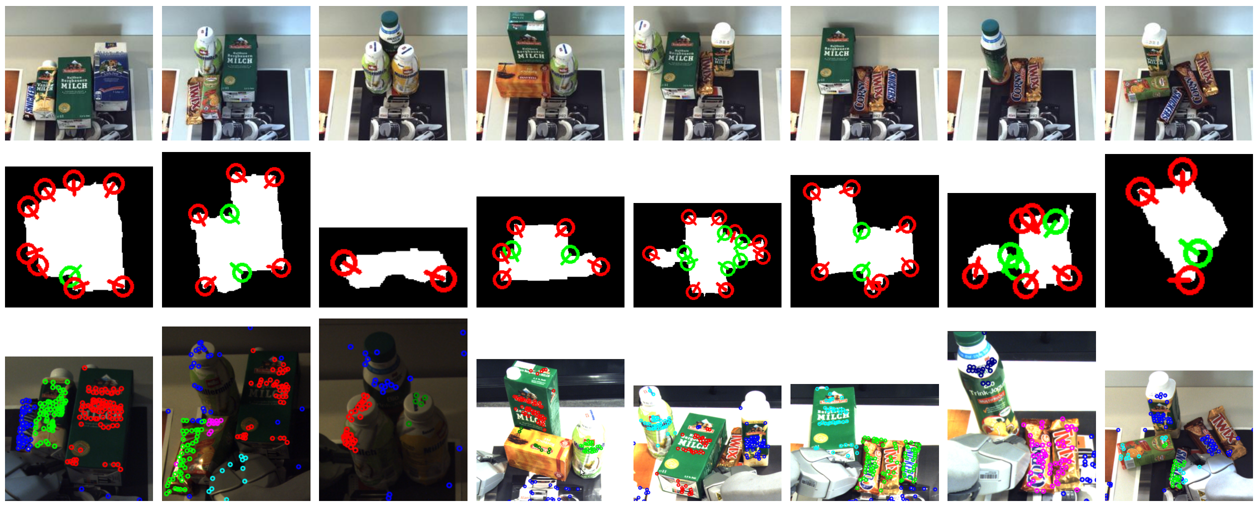


Fig. 3. Test scenes 1 to 8 from left to right. Top row: original scenes, middle row: contact point estimation, bottom row: segmentation after the first push cycle. Please note, that successfully segmented objects were removed from the scene and the contact point estimation and segmentation cycle were repeatedly executed.

The clustering algorithm we propose is outlined in Alg. 1, and combines a divisive clustering approach with RANSAC-style model hypothesis sampling. At the core of the algorithm (lines 4–12), we randomly draw 2 tracked features u,v and estimate a sequence of rigid transformations A_1 from their optical flow motions as first model hypothesis. The feature trajectories S_i that can be explained well by A_1 are considered "model inliers" and are removed from set of feature trajectories. From the remaining set, again 2 features are drawn to create a second model hypothesis A_2 and all inliers are removed. This process repeats until there are not enough features left to create a new model hypothesis. This process results in K hypotheses.

IV. Experiments

Our system was deployed on Willow Garages PR2 robot. Depth images were taken from a Kinect sensor mounted on the robots head and the PR2s built-in 5-megapixel camera was used for capturing images for feature extraction and tracking.

A. *Segmentation of Objects in Cluttered Scenes*

We evaluated our system on eight tabletop scenes with the cluttered background shown in Fig. 3. For each scene, the original setup of objects, the detected contact point candidates and push directions, and the feature clusters after the first push cycle are shown in the respective row. Across all runs using corner-based pushing 89% of all objects were segmented successfully.

The segmentation of the scenes took 1.047 seconds on average to compute, which also demonstrates that our algorithm is suitable for real world settings.

B. *Grasping*

We also ran a grasping experiment on the scene 8 (Fig.3). In this experiment, we use low-quality image from the Kinect for the segmentation and an associated point cloud for the calculation of the object pose. The accompanying video[1] is showing the above mentioned experiment.

C. *Open Source Code*

We provide the software[2] and documentation[3] as an open source. In the workshop we plan to demostrate the segmentation of textured objects using Kinect sensor and manually interaction with the objects.

V. Future Work

The results show applicability of our system for objects of various sizes, shapes and surface. Future work includes integrating our approach with other object segmentation techniques in order to account for textureless objects and to further improve the segmentation rate. We also plan to integrate an arm motion and a grasp planner which will enable the robot to perform robust grasping and deal with even more complex scenes.

References

[1] D. Katz and O. Brock, "Interactive segmentation of articulated objects in 3d," in *Workshop on Mobile Manipulation at ICRA*, 2011.

[2] N. Bergström, C. H. Ek, M. Bjrkman, and D. Kragic, "Scene understanding through interactive perception," in *In 8th International Conference on Computer Vision Systems (ICVS)*, Sophia Antipolis, September 2011.

[3] P. Fitzpatrick, "First contact: an active vision approach to segmentation," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2003.

[4] J. Kenney, T. Buckley, and O. Brock, "Interactive segmentation for manipulation in unstructured environments," in *Proceedings of the 2009 IEEE international conference on Robotics and Automation*, ser. ICRA'09, 2009.

[1]`http://youtu.be/4VVov6E3iiM`

[2]`http://ros.org/wiki/pr2_interactive_segmentation`

[3]`http://ros.org/wiki/pr2_interactive_segmentation/Tutorials`

Fast segmentation of RGB-D images for semantic scene understanding

Dirk Holz, Alexander J. B. Trevor, Michael Dixon, Suat Gedikli and Radu B. Rusu

I. Introduction

As robots move away from pre-programmed action sequences in controlled laboratory setups towards complex tasks in real-world scenarios, both the perception capabilities of these systems and their abilities to acquire and model semantic information must become more powerful. In this context, fast means for pre-processing acquired sensory information and segmenting task-relevant regions are an enabling technology and a prerequisite for avoiding longer delays in sense-plan-act cycles.

We present (on a poster and in a demo) two fast segmentation methods for RGB-D images.

II. Approximate Meshes for Pre-processing

In the first method, we exploit the organized structure of RGB-D images and apply an approximate surface reconstruction [1] by simply connecting adjacent image pixels. The resulting mesh efficiently caches local neighborhoods for further processing. Furthermore, we compute approximate surface normals directly on the mesh and apply a multi-lateral filtering step to considerably smooth the data. Using an efficient region growing implementation and different region models, we can efficiently compute plane segmentations and full polygonalizations (Fig. 1), or segment locally smooth regions and detect geometric shape primitives (Fig. 2).

III. Direct Image Segmentation

The second method also exploits the image structure by using a connected component based technique. Each pixel is compared to neighboring pixels (in a 4-connected sense) using a comparison function (similar to the region models above). Points are considered part of the same segment if the comparison function returns true. Different comparison functions can be used for different segmentation tasks, such as a plane equation comparison (the dot product between normals and range must match), euclidean distance, color, or combinations of these. These can be run in a sequence and with an optional mask, for example: tabletop objects can be detected by first detecting planar regions, then using these regions as a mask and segmenting with a euclidean distance comparison (Fig. 3). See the Point Cloud Library [2] at www.pointclouds.org for further details, documentation, and an open source implementation.

D. Holz is with the University of Bonn, Bonn, Germany. A. J. B. Trevor is with the Georgia Institute of Technology, Atlanta, Georgia, USA. M. Dixon, S. Gedikli and R. B. Rusu are with Willow Garage, Inc., California, USA. Contact: holz@ais.uni-bonn.de, atrevor@cc.gatech.edu

A demonstration video of the approach is available at http://youtube.com/watch?v=LZ8l4w3qw3E.

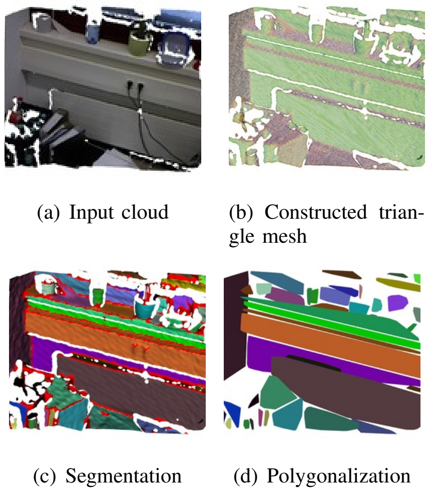

(a) Input cloud (b) Constructed triangle mesh (c) Segmentation (d) Polygonalization

Fig. 1: Approx. surface reconstruction and segmentation.

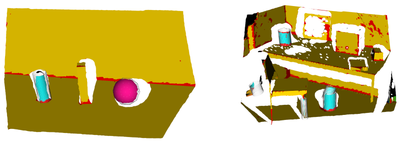


Fig. 2: Detecting geometric shape primitives: planes (yellow), cylinders (cyan) and spheres (magenta).

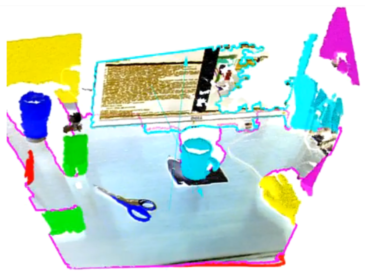


Fig. 3: Planes and connected components.

References

[1] D. Holz and S. Behnke, "Fast Range Image Segmentation and Smoothing using Approximate Surface Reconstruction and Region Growing," in *Proceedings of the International Conference on Intelligent Autonomous Systems (IAS)*, Jeju Island, Korea, 2012.

[2] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, 2011, pp. 1–4.

Semi-Autonomous Environment Mapping using GUI based Outline Operating Instructions

Yohei Kakiuchi and Atsushi Tsuda and Shunichi Nozawa and Kei Okada and Masayuki Inaba

Abstract— **There are many objects which have manipulable link in a household environment. A humanoid robot working in a household environment shared with humans needs to recognize and manipulate such objects like a refrigerator and a shelf. In order to automatically create a geometric model for recognition and manipulation, a geometric model with articulated link is constructed by a point cloud which is observed by robot equipped with a 3D vision.**

In our approach a robot and a human, using a GUI to semi-autonomously command the robot, jointly manipulate furniture entities and generate a map with geometric models with articulated link at the same time.

I. INTRODUCTION

There are many daily assistive tasks which need operations of furniture and tools with articulated link for a humanoid robot in a household environment shared with humans. In order to perform a task operating object with articulated link, a robot needs to know about a geometric model which contains a kinematic model, joint type and position.

In this paper, we propose a human supervised semantic mapping. Human indicate clues to manipulate objects with articulated link such as a drawer or a refrigerator, then a robot autonomously manipulate objects using the clues and obtain geometric models which contain information for operation such as a handle position which can operate a link, how to grasp the handle, and which direction a link move to.

II. GENERATING MAP FROM ROBOT MOTION

In order to automatically obtain environmental knowledge, robot have to acquire a geometric model and a kinematic model of object. As an example of environmental knowledge acquisition by a robot, Blodow*et al.* [1] created a map of the household environment with autonomously moving mobile robot. This map is a semantic map which contains such as the shelf position and the handle of the shelf that can be manipulated. Creating a symbol during the generating a semantic map has been determined by the number of knobs or handles and the location of each [2]. In this case, the geometric conditions were used for acquiring meaning, it is difficult to infer the meaning of the object on the map.

We propose a human supervised robot motion with obtaining object model. A operator indicates outline operating instructions for environment manipulation. A robot makes a manipulation plan based on a sensor observation of environment. The robot presented to the plan to the operator.

Y. Kakiuchi, A. Tsuda, S. Nozawa, K. Okada and M. Inaba are with Graduate School of Information Science and Technology The University of Tokyo, 7-3-1 Hongo, Bunkyo-Ku, 113-8656 Tokyo, Japan {youhei,tsuda,nozawa,k-okada,inaba}@jsk.t.u-tokyo.ac.jp

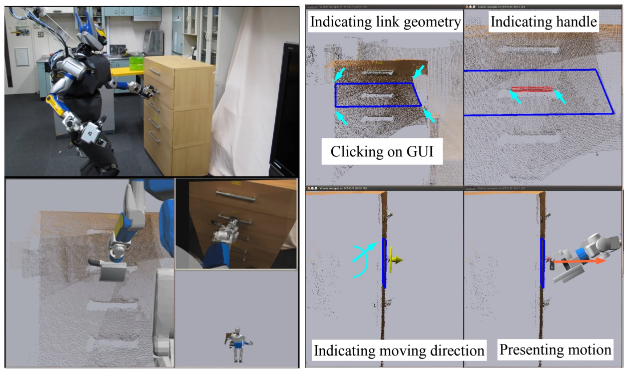


Fig. 1. Robot opened shelf (left) GUI view, blue arrow indicating clicking point (right)

The plan contains the trajectory of handle and humanoid's joint angle. The plan presented to the operator is approved or modified, the robot executes the motion in accordance with the plan. Outline the operating instructions shows place suitable for manipulation, handle position for grasping or the type of articulated mechanism. These outline operating instructions are provided through pointing points obtained from the 3D vision of robots using a GUI. A robot obtains a geometric model by observation from robot's 3D vision during robot motion.

III. CONCLUSION

A robot executing a motion and generating an environment model are performed at the same time. The former is executing the motion for environment manipulation by the human using GUI. The latter is generating a geometric model, which is necessary for an autonomous robot task, updated by the observation during the robot motion.

Human indicated just a clue for operating objects which have articulated link. From this information, robot can operate articulated object and obtain the internal structure and characteristics of the object unless the robot try to manipulate it. We created a semantic map from these information which was obtained by observing robot motion indicated through outline operating instructions.

REFERENCES

[1] N. Blodow, L. C. Goron, Z.-C. Marton, D. Pangercic, T. Ruhr, M. Tenorth, and M. Beetz, "Autonomous Semantic Mapping for Robots Performing Everyday Manipulation Tasks in Kitchen Environments," 2011, pp. 4263–4270.

[2] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz, "Towards 3d point cloud based object maps for household environments," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 927–941, 2008.

Understanding Customers: Purchase Behaviors in Ubiquitous Market

Koji Kamei[1,†] Koya Sugimoto[2,1] Hiroyuki Kidokoro[3,1] Tetsushi Ikeda[1]
Masayuki Kanbara[2,1] Kazuhiko Shinozawa[1] Takahiro Miyashita[1,3] and Norihiro Hagita[1,2,3]

***Abstract*— Shopping is a typical daily activity to be supported by Networked Robotic Services. In a retail shop, networked robots will be able to support both customers and the shop by recommendation service if they can understand the customers' interest from the observed purchase behaviors. We have constructed an experiment environment called 'Ubiquitous Market' as an instance of ubiquitous networked robot system. Two types of customers' behaviors have been extracted. One distinguishes whether a customer purchases an item on a shelf or not from the local trajectories in front of the shelf. Another estimates whether a customer visits a shelf or not from the sequences of stop-by points and then recommends items.**

Researches on network robot systems (NRS)[1] have focused on cooperation among service robots which overcomes the limitations of stand-alone robots by having robots, environment sensors, and humans communicate and cooperate through a network. Aiming at the development of life support robots that coexist with people in human living environments, many types of robots will deployed to multiple locations with ubiquitous network technology as a social infrastructure. In the project, the common functionalities are implemented as Ubiquitous Network Robot (UNR) Platform[2].

Consider an interactive system installed in a real-world shopping mall where visible robots recommend items to the customers. The system plays three roles as interaction technology: as a ubiquitous environment that understands events occurring in the real world, as a recommendation system that chooses information to be presented to the customers, and as a persuasion system in which the robots persuade the customers. Ubiquitous Market is such an environment equipped with UNR technologies that understands customers' purchase behaviors and supports their daily shopping activities in brick-and-mortar shops.

When a customer intend to purchase several items from retail shops, a sequence of activities occurs on both the customer and shop agents. The customers' behaviors are observed by networked sensors that sometimes called as "ambient intelligence." Such behaviors should be symbolized to describe shopping support services. Kanda[3] extracted "spatial primitives" and "behavioral primitives" as structured environmental information. Knowledge from marketing science and consumer psychology also provides baseline for this symbol-grounding problem. The structure of environment should be defined and understood by integrating such kind of human behaviors. The customer has to understand what he or she wants to purchase, compare corresponding items, decide whether to buy or not, and then pick it up. Shop agents have to estimate items of the customer's interests and the certainty of purchase, and then provide appropriate information. Our approach focused on the trajectories of customers observed by a set of laser range finders in retail shop environment and extracted two types of customers' behaviors.

One distinguishes whether a customer purchases an item on a shelf or not from the local trajectories in front of the shelf. Though previous works estimated the degree of a customer's interest from duration of a stop, we focused on four characteristics for each segment of local movement: such as variance of vertical distance from the shelf, minimum of the vertical distance, total distance of horizontal movement, and total angle of direction changes. The proposed method distinguishes customers who purchase items even when their stop durations were not so long. Another estimates whether a customer visits a shelf or not from the sequences of stop-by points and then recommends items from robots on the shelf [4]. Another estimates whether a customer visits a shelf or not from the sequences of stop-by points and then recommends items from robots on the shelf. The proposed method estimates a shelf that a customer possibly visits after observing more than three stop-by locations. In our experiment, the precision of the estimation was 0.48 as a baseline. Compared to this baseline, recommendation from robots made the customers' move to the estimated spot with 0.76 of probability.

In this work, we focused on two methods for recognizing customers' purchasing behaviors observed with UNR environment. In the experiments, required knowledge about items and their categories were limited so that we could organize them by ourselves. To extend the scale of the shopping experiments, the UNR Platform should provide means to obtain such knowledge from the Web as proposed in Web-enabled Robot [5].

References

[1] A. Sanfeliu et al., "Network robot systems," Robotics and Autonomous Systems, vol. 56, 2008, pp. 793-797.
[2] K. Kamei et al., "Cloud Networked Robotics," IEEE Network Magazine, vol.26, May 2012, to be published.
[3] T. Kanda et al., "A Communication Robot in a Shopping Mall," IEEE Trans. on Robotics, vol. 26, 2010, pp.897-913.
[4] H. Kidokoro et al., "You stopped by there? I recommend this: changing customer behaviors with robots," Proc. 13th Intl. Conf. Ubiquitous Computing (UbiComp2011), Beijing, 2011, pp. 569-570.
[5] M. Tenorth et al. "Web-enabled Robot," IEEE Robotics and Automation Magazine, vol. 18, Jun. 2011, pp. 58–68.

*This work was in part supported by the Japanese Ministry of Internal Affairs and Communications.

[1]ATR Intelligent Robotics and Communication Research Laboratories
[2]Dept. of Info. Science, Nara Institute of Science and Technology
[3]Graduate School of Info. Science and Technology, Osaka University
[†]K. Kamei is with ATR Intelligent Robotics and Communication Research Laboratories, 2-2-2 Hikaridai Seika-cho, Kyoto 619-0237, Japan

Elaborative Evaluation of RGB-D based Point Cloud Registration for Personal Robots

Ross Kidson, Dejan Pangercic, Darko Stanimirovic, Michael Beetz
{kidson, pangercic, beetz}@cs.tum.edu, darkos490@gmail.com

I. Introduction

The motivation for this work comes from the requirement for a robust method of generating a functional model of kitchen environments from RGB-D data. In order for mobile robots to perform useful tasks such as opening and closing of doors and drawers or retrieving objects and returning them to appropriate places a detailed 3D map is required with an accuracy to within 1cm, whereby information of the environment can be extracted and processed into a semantic and functional form for the robot [1], [2]. However in order to build up such semantic maps, it is necessary to be able to generate a reliable model of an area from sensor data without the prior knowledge of that area. An excellent candidate for recording 3D data for mapping are RGB-D cameras. These are cameras that capture RGB images as well as measure depth and associate a depth value to each image pixel, generating 3D point clouds as the output.

In this work we consider building such a 3D model using a hand-held RGB-D camera. In order to build the map from such a camera, some form of SLAM system needs to be utilized [3] which attempts to both simultaneously calculate the path the camera has taken, and create a map of the area from the data. We have qualitatively evaluated three openly available algorithms for building 3D models [3], [4], [5], the first of which was selected as the go ahead tool according to the following criteria: accuracy, computational cost, memory requirement and community support. An optimal point cloud generated by [3] is depicted in Fig. 1, top[1]. As we see in the bottom row close up views the model is far away from the precision required for the segmentation of doors and handles and thus subsequent grasping and opening. The imperfections are mostly due to a critical component of the SLAM process, the calculation of a transformation between two camera frames by aligning respective 3D point clouds. One approach to address this problem is to take SIFT features [6], projected into 3D space using depth data, and applying random consensus to obtain a transformation between two frames. This can be a fast and accurate method under certain conditions. Another approach is to use the Iterative Closest Point (ICP) [7] algorithm, utilizing for example point to plane error metrics and the Levenberg Marquardt (LM) solver to converge the point clouds. This is also effective, however is susceptible to local minima and therefore requires initialization conditions which are

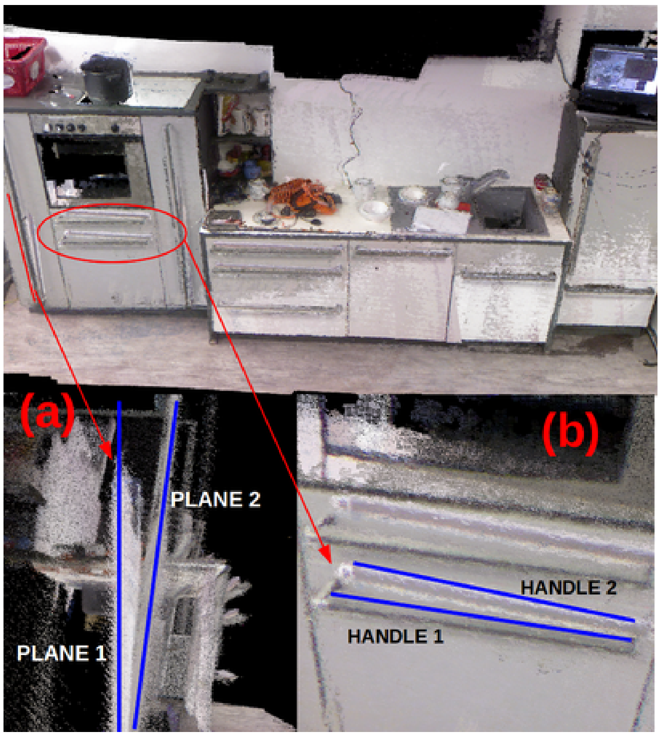


Fig. 1: Complete point cloud model of a kitchen area as created from an RGB-D SLAM system [3], illustrating imperfections. (a) Shows the model being viewed from the left side, plane 1 and plane 2 should be representing the same wall in the model and are therefore misaligned. (b) Handle 1 and Handle 2 are the same handles from different point clouds and are therefore misaligned

.

close to the optimal solution. Finally, there are algorithms that make use of both depth and visual feature data. The SLAM system in [3] uses both approaches based on the heuristically selected number of SIFT features and the system proposed by [8] uses a joint optimization method in order to combine both dense point cloud and visual feature data in one optimization step to calculate a transformation. **In this paper we performed a benchmark study with an objective to determine the behaviour of an implementation of the algorithm put forward by [8] in a number of distinct sceneries.**

II. Joint Optimization

In this section we will briefly recap the joint optimization method proposed by Henry et al. [8]. This algorithm jointly optimizes the transformation between two point clouds over

[1]Please note that the g2o global optimization technique was also used in the creation of this point cloud.

both the dense point associations and visual feature associations. SIFT feature was used to extract the latter.

The joint optimization equation is stated in Eq. 1. It jointly optimizes two distance error metrices: point-to-plane for the distance between dense point cloud correspondences and the point-to-point for the distance between visual feature correspondences. The visual feature correspondences are calculated at the start and remain the same over all iterations. $\mathbf{T}*$ is the transformation to be optimized. f_s and f_t are the visual feature points projected into 3D space for source and target point clouds respectively, and p_s and p_t are for the dense points. A_d and A_f are the number of features and dense points, and are used to normalize the two parts of the joint optimization function. w_i and w_j are weights per correspondence for feature and dense associations. They could be used to rate the reliability of the associations, which may be obtained from the algorithm used to find features or from some form of distance metric on the dense clouds. In this implementation they are all set to 1. Finally α is a weighting factor, giving either the visual feature correspondence error metric or the dense point cloud error metric more influence, whereby $\alpha = 1$ is only the visual feature error metric, $\alpha = 0$ is only dense point error metric, and anything between a mix of the both.

$$\begin{aligned}\mathbf{T}* \quad = \quad & \arg\min_{T} \Bigg[\alpha \left(\frac{1}{|A_f|} \sum_{i \in A_f} w_i \left| \mathbf{T}(f_s^i) - f_t^i \right|^2 \right) \\ & + (1-\alpha) \left(\frac{1}{|A_d|} \sum_{j \in A_d} w_j \left(\left| (\mathbf{T}(p_s^j) - p_t^j) \cdot n_t^j \right|^2 \right) \right) \Bigg] \qquad (1)\end{aligned}$$

III. Evaluation method

The algorithm was implemented in the Point Cloud Library (PCL)[1] and tested using ROS. Sets of point clouds were recorded using the Microsoft Kinect camera. A stream of clouds were recorded by moving the camera across each scene, and afterwards individual clouds were selected from this data.

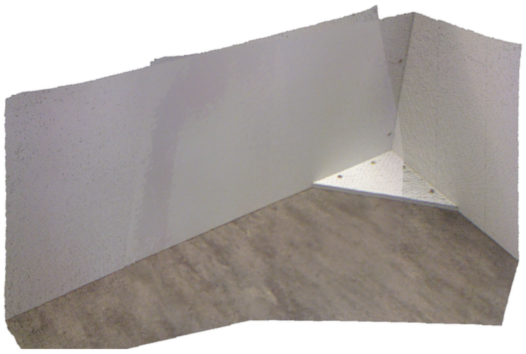
To evaluate the joint algorithm, scenes were selected based on the presence or absence of structure and texture, in order to show in which conditions the joint optimization is of benefit compared with using only dense point cloud data or only visual association data. An additional parameter was tested, the distance of the camera to the scene.

Altogether, seven scenes were taken for evaluation: the 4 shown in Fig. 2 and a typical office desk scene taken at 1m, 2m and 3m distance. For the non textured scenes, plain walls were selected which produced a small number of visual features. For the non structured scenes flat walls were recorded. For the featured and textured scene, part of the kitchen scene was taken.

[1]Staged for submission to PCL.

(a) Textured and structured

TUM
TECHNISCHE
UNIVERSITÄT
MÜNCHEN
(b) Textured and non structured


(c) Non textured and structured

(d) Non textured and non structured

Fig. 2: Examples of aligned point clouds for different scenes.

A dataset intended for testing RGB-D SLAM systems was made available [9] however this was not used for this evaluation because the data set is more aimed at testing SLAM

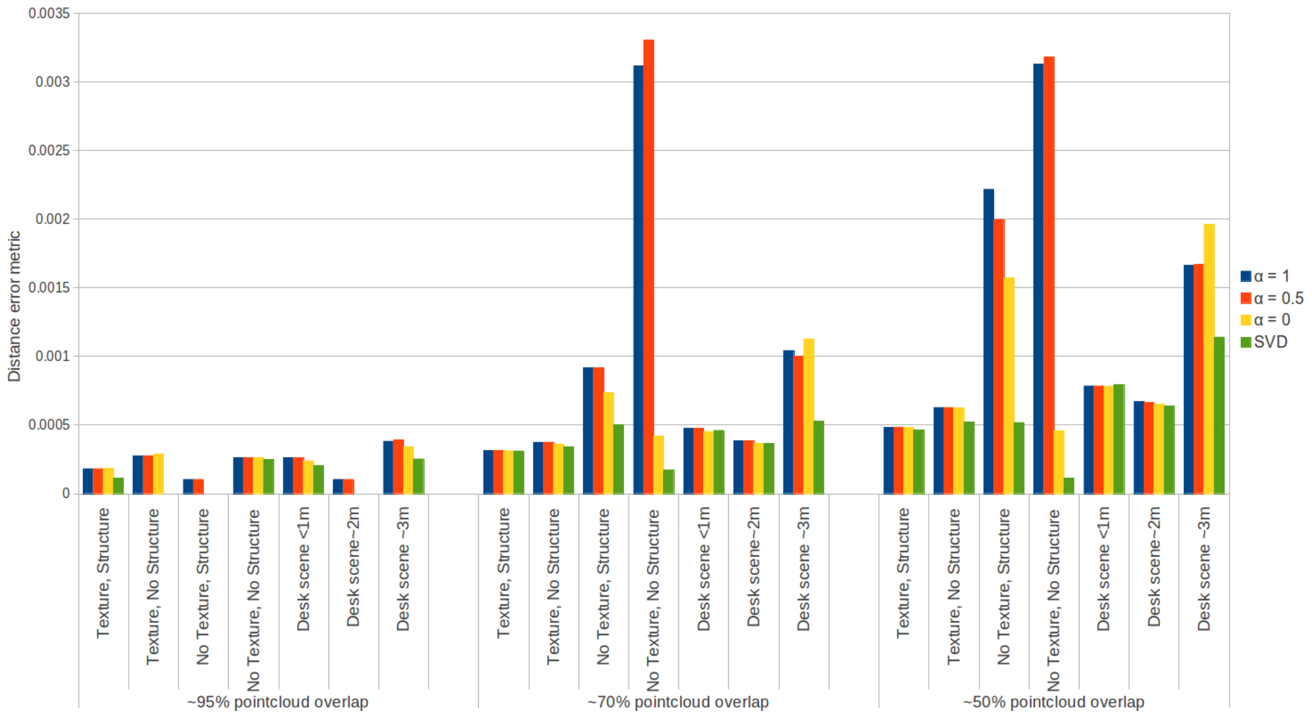


Fig. 3: Distance error metric on converged point clouds.

systems rather than point cloud alignment. In particular there are no instances of scenes with less structure or less texture.

It is also important to consider the performance not only of consecutive point clouds, but point clouds with less overlap. So for each scene, point clouds with varying overlap were selected, based on percentage of point cloud overlap. Three levels were taken approximately 95%, 70% and 50%.

In terms of the algorithm, we selected varying values of α to be $\alpha = 0.0$, $\alpha = 0.5$ and $\alpha = 1.0$, to contrast the performance of the dense point cloud error metric, the visual feature correspondence error metric, and a combination of the both. It was also an aim of this study to determine the how many visual feature points are required for accurate alignment as well as to observe the deterioration of alignment with insufficient features. Due to this, the algorithm does not automatically fall back to $\alpha = 0$ where visual features were lacking, differing from [8]. For convergence criteria the minimum change in the transformation was set to 1e-8, Euclidean fitness error was set to 0 (never converge on this criteria) and maximum number of iterations was set to 75.

In addition a standard ICP algorithm using Singular Value Decomposition as a solver was also tested. This was the standard implementation from PCL which uses point to point error metrics. This was used to contrast to the joint algorithm with α set to 0, which uses LM and point to plane error metrics, which may be susceptible to different local minima.

As described above, RANSAC is used to remove outlier feature associations, as well as to provide an initial transformation for the optimization function. However when there are a low number of features, this initial transformation can often be much worse than providing no initial transformation at all, and therefore no RANSAC initialization has been used for the non textured scenes.

We used a *distance error metric* to evaluate each test, which was calculated by first finding point correspondences using nearest neighbour search, rejecting correspondences over a distance of 1cm^2 (as to ignore non overlapping sections of the clouds) and normalizing the error over the number of correspondences. It is important to note this metric evaluates the structural alignment of clouds, and does not take matching point colour into account. Therefore flat surfaces may obtain very good results even when the textures are completely misaligned.

IV. Results

The results can be seen in Fig. 3.

For the textured scenes, it is apparent that for all different values of α, the performance was roughly the same, that is, there was no benefit from the joint optimization of both error metrics.

The non textured scenes showed a significant improvement in decreasing error when jointly optimizing with dense correspondences, or using only dense correspondences. This was expected as the dense 3D point cloud data will help improve the alignment where visual feature points are lacking. Taking the test case of no texture with structure and 50% point cloud overlap, it can be clearly seen that there is a decrease in error as α decreases, that is, more weight is given to the dense point cloud error metric.

The point cloud overlap also played an important role, whereby less overlap resulted in generally more error or in some cases no convergence (no feature, no structure). This

^{2}Accounting for Kinect's precision: `http://www.ros.org/wiki/openni_kinect/kinect_accuracy`.

is likely due to the fact that less visual feature points and less structure are available for alignment in point cloud pairs with less overlap.

Minimum number of visual features is a critical factor when incorporating distance between visual associations as an error metric. Below a certain amount of visual features, typically less than 25, it has been seen that the point clouds will never converge when using some or all feature error metric component ($\alpha > 0$). This is made very clear from no texture no structure in both 70% and 50% overlap, as $\alpha = 0.5$ has about the same error as $\alpha = 1$.

Not only is minimum number of visual features an important factor when utilizing visual feature correspondences error metric, but also the **distribution of features.** Take for instance when all features occur on a straight line (as in Fig. 2d), this allows the point clouds to essentially pivot about this line during alignment, as there is insufficient data to constrict the alignment correctly. This should be taken into account if visual features are to be automatically discarded.

For determining **RGBDSLAM suitability**, the error metric is not sufficient in itself to identify misalignment of texture, or significant misalignment of point clouds. Upon visual inspection of point cloud pairs, it was found that there was such a gross misalignment similar to that seen in Fig. 1, particularly for the untextured scenarios.

It appears that the SVD test case was superior in every scenario. However in actual fact there was often significant misalignment of texture between clouds, as the error metric did not take misalignment of texture into account. This was also true for the $\alpha = 0$ test cases.

In terms of **execution speed**, for our implementation it typically takes between 15-30 seconds on an desktop computer for the joint optimization function to converge. In nearly all cases it converges based on the minimum change of transformation criteria. Before the optimization can run the point normals also need to be calculated for the point to plane error metric, which also typically takes 30 seconds. This execution speed rules this algorithm out in its current state for realtime RGB-D SLAM operation, as for realtime operation, many point clouds need to be aligned per second. However it is still viable for an an offline processing mode, or for implementation on the GPU.

V. Conclusions and Future Work

The take home message of this experiment is two-fold. On one hand we evaluated behaviour of the state-of-the-art algorithm for the registration of RGB-D point clouds in terms of the combination of textured, non textured, structured and non-structured indoor scenes. On the other hand we showed that even in the presence of the algorithm favorable conditions, the point-based registration is not enough to acquire semantic maps that personal robots could use reliably. We therefore believe that the following research avenues shall be explored.

First, we will explore the registration based on priors. For example as the scene is being mapped, segmentation of walls, floors or fixtures will also take place, to provide more information about the scene, such as the principle alignment axes. Consecutive frames can then be registered not only based on alignment of visual associations and dense point clouds, but also based on the alignment of these segmented parts or planes. We have already begun to investigate such a solution, by extracting and using cupboard handles as the third component in the joint optimization equation. (See Fig. 4) This method shows promise, particularly addressing alignment issues illustrated in Fig. 1.

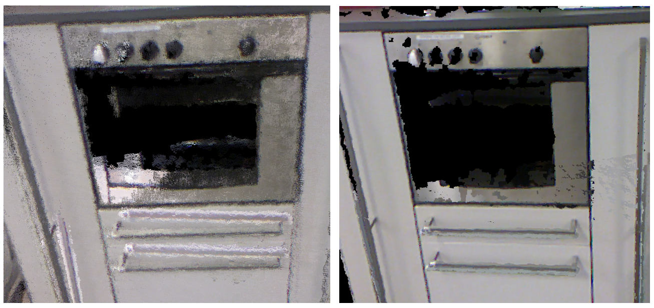

(a) Registration with RGB-D features (b) Registration by aligning segmented handles

Fig. 4

A registration method based on planes has been proposed [10], this is another line of work that we will pursue and add into the joint optimization framework. Planes are essentially difficult to consistently match but are on the other hand dominant features in the environment.

Another option to increase the accuracy of the registration is to combine the RGB-D camera with the sensors (such as accelerometers or gyroscopes) for the camera pose estimation.

References

[1] M. Beetz, M. Tenorth, D. Pangercic, and B. Pitzer, "Semantic object maps for household tasks," in *5th International Conference on Cognitive Systems (CogSys 2012)*, 2012.

[2] A. Pronobis, "Semantic mapping with mobile robots," Ph.D. dissertation, KTH Royal Institute of Technology, Stockholm, Sweden, June 2011. [Online]. Available: http://www.pronobis.pro/phd

[3] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An evaluation of the rgb-d slam system," in *ICRA*, 2012.

[4] http://robotics.ccny.cuny.edu/git/ccny-ros-pkg/ccny_vision_rgbd.git/.

[5] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon, "KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*, New York, NY, USA, 2011, pp. 559–568. [Online]. Available: http://svn.pointclouds.org/pcl/trunk/gpu/kinfu/

[6] D. Lowe., "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[7] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern analysis and Machine Intelligence*, vol. 14, pp. 91–110, February 1992.

[8] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 647–663, Apr. 2012.

[9] J. Sturm, S. Magnenat, N. Engelhard, F. Pomerleau, F. Colas, W. Burgard, D. Cremers, and R. Siegwart, "Towards a benchmark for rgb-d slam evaluation," in *Proc. of the RGB-D Workshop on Advanced Reasoning with Depth Cameras at Robotics: Science and Systems Conf. (RSS)*, Los Angeles, USA, June 2011.

[10] K. Pathak, A. Birk, N. Vaškevičius, and J. Poppinga, "Fast registration based on noisy planes with unknown correspondences for 3-d mapping," *Trans. Rob.*, vol. 26, no. 3, pp. 424–441, June 2010. [Online]. Available: http://dx.doi.org/10.1109/TRO.2010.2042989

Towards Real-Time Semantic Localization

Hyon Lim
Seoul National University, Korea
hyonlim@snu.ac.kr

Sudipta N. Sinha
Microsoft Research, Redmond, USA
sudipsin@microsoft.com

***Abstract*— We describe an efficient image-based localization system which can be used for real-time, continuous semantic localization within a known environment. Our system can precisely localize a camera in real-time from a video stream within a fairly large scene that has been reconstructed offline using structure from motion (SfM). This is achieved by interleaving a fast keypoint tracker that uses BRIEF descriptors, with a direct 2D-to-3D matching approach for recognizing 3D points in the map. Our approach does not require the construction of an explicit semantic map. Rather semantic information can be associated with the 3D points in the SfM reconstruction and can be retrieved via recognition during online localization.**

I. INTRODUCTION

In robotics, *semantic localization* refers to the task where the robot must report its location semantically with respect to objects or regions in the scene rather than reporting 6-DOF pose or position coordinates. In prior work on semantic localization using contextual maps [1], coarse location estimates could be recovered using only three states – nearby, near and far with respect to semantic landmarks in the scene. In contrast, our system aims for precise, semantic localization based on real-time 6-DOF image-based localization [2].

We represent the map with a 3D point cloud reconstruction computed using SfM, which also contains multiple DAISY feature descriptors [3,4] associated with the 3D points. By tracking keypoints in video and matching them to the 3D points, our system continuously estimates a precise pose estimate in real-time. The main idea involves interleaving a fast keypoint tracker that uses BRIEF features [5] with an efficient approach for direct 2D-to-3D matching. The 2D-to-3D matching avoids the need for online extraction of scale-invariant features. Instead, offline we construct an indexed database containing multiple DAISY descriptors per 3D point extracted at multiple scales. The key to efficiency lies in invoking DAISY descriptor extraction and matching sparingly during online localization, and in distributing this computation over a window of successive frames. Fig. 1 shows the trajectory of a camera mounted on a quadrotor micro-aerial vehicle (MAV), computed using our real-time localization system, as the MAV is flown manually[1].

Unlike our work, visual SLAM (VSLAM) systems have the flexibility of being able to localize a camera within an unknown scene [6,7]. However, semantic localization in an unknown scene can be extremely challenging. Objects must be recognized by their categories, which is very difficult to achieve even without real-time constraints [8]. Although, prebuilt maps are necessary in our method, this also provides

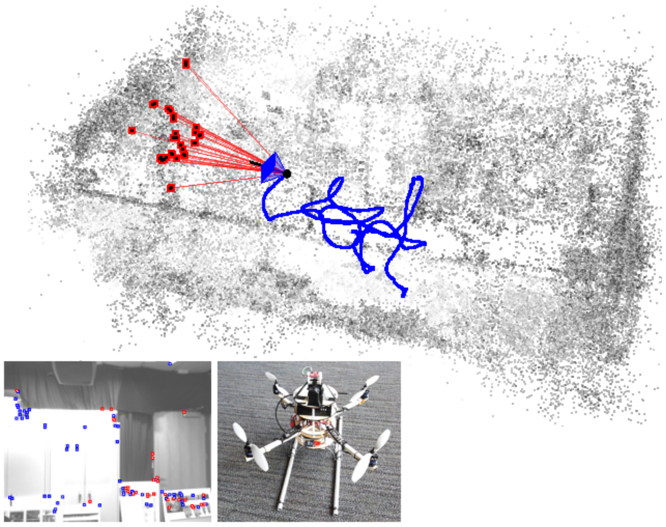


Fig. 1. The trajectory of a quadrotor micro aerial vehicle (MAV) within a 8m × 5m room computed using our method. The SfM reconstruction has 76K points. A video with the recognized landmarks are shown in red. The corresponding 3D points are shown on the map.

the underlying framework for storing detailed semantic information along with 3D points in the scene. During online localization, semantic information can be retrieved via visual recognition of 3D points in the map which are subsequently tracked in video. Our system can handle maps with an order of magnitude more 3D points than typically handled by VSLAM systems. This makes our system robust and enables both continuous localization over long durations within large scenes as well as fast relocalization whenever needed.

II. OUR METHOD

We represent the scene with a 3D reconstruction in a global coordinate frame, which is computed using SfM from an image sequence. The calibrated images are used to build a database of DAISY feature descriptors associated with the 3D points. A kd-tree index is constructed over the descriptors to support efficient approximate nearest neighbor (ANN) queries during online feature matching. Fig. 2 shows an overview of the various steps.

A. Map Construction

The map is built offline using the following steps:

- The input images are processed using SfM [9].
- The cameras are grouped into overlapping clusters.
- Keypoints and DAISY descriptors are extracted at multiple scales in the images and associated with the 3D points.

[1]See `http://goo.gl/Vp6ps` for a video of our real-time system.

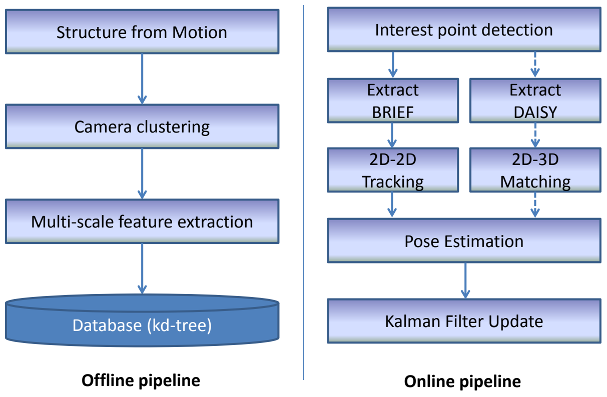


Fig. 2. Overview of the offline and online processing steps in our system. As mentioned in Section I, extraction of DAISY features and 2D-3D matching queries are not executed at every frame.

- A kd-tree is built for all the descriptors. Appropriate lookup tables are built to support efficient queries to find which image or 3D point a feature descriptor belongs to.

Semantic labels can be added to the 3D points by annotating the images with object names and bounding boxes [9]. Using the 2D-3D correspondences obtained from SfM, the labels can be easily mapped from pixels to subsets of 3D points.

B. *Place Recognition*

In large scenes, global matching can be difficult due to greater ambiguity in feature descriptors. To address this, we perform coarse place recognition to filter erroneous 2D-3D matches before 6-DOF pose estimation step. As a result, fewer RANSAC hypotheses are required during robust pose estimation, making that step more efficient. For place recognition, we cluster nearby cameras based on SfM results by solving an *overlapping view clustering* problem where cameras with many SfM points in common are grouped into the same cluster [10]. When localizing an image, the most likely camera group is selected using a simple voting scheme over the set of matching descriptors returned by the ANN query on the descriptor group.

C. *Real-time Localization*

Our algorithm aims for real-time localization over long periods and at avoiding fluctuations in the frame-rate. At its core lies a fast keypoint tracker. Keypoints (Harris corners) from one frame are tracked in the following frame by matching to candidate keypoints within a local search window in the next frame [11]. Binary feature descriptors (BRIEF) [5] are used to find the best frame-to-frame matches. This fast tracker is interleaved with an efficient approach to find which 3D points in the map correspond to the tracked keypoints. The camera pose for each frame is robustly estimated from these 2D-3D matches. For determining these matches, DAISY descriptors [3,4] must be extracted. This can be computationally expensive depending on the number of descriptors extracted and queried in the kd-tree. Our system amortizes this cost by requiring that the feature matching be performed on demand and by spreading the computation over a window of successive frames.

III. Results

A single-threaded C++ implementation of our system runs at an average frame-rate exceeding 30Hz on multiple datasets on a laptop with an Intel Core 2 Duo 2.66GHz processor running Windows 7. It is about fives times faster than the single-threaded implementation of [12], which runs at 6Hz (and at 20Hz using four cores). To test the feasibility of our method for onboard processing on a small MAV, we designed our own quadrotor vehicle mounted with the PointGrey FireflyMV camera and a FitPC2i[2] computer running Windows 7. Our algorithm runs at about 12Hz on the FitPC.

IV. Conclusions

Our real-time localization system [2] can be easily extended for semantic localization, by augmenting the 3D points in the map with semantic labels. For example, this can be done by manually inserting annotation in the 2D images used for map construction (offline SfM) and automatically transferring the labels to the 3D points in the map. For an indoor scene, the labels could refer to small objects, (e.g. books, CDs), larger objects (e.g., furniture) or a particular room in the scene.

During online localization, our system recognizes subsets of 3D points in the map using an efficient 2D-to-3D matching approach and then tracks the 3D points in video. Whatever semantic labels are stored with the tracked 3D points can be used to recognize objects or locations in the video. Additional semantic information can be inferred from the camera pose estimate and from the accurate 3D map where objects and semantic locations are precisely localized. For instance, the relative location or distances to nearby objects that are not yet visible in the camera can be predicted using this information.

References

[1] C. Yi, I. H. Suh, G. H. Lim, and B.-U. Choi, “Active-semantic localization with a single consumer-grade camera,” in *SMC*, 2009.
[2] H. Lim, S. N. Sinha, M. Cohen, and M. Uyttendaele, “Real-time image-based 6-dof localization in large-scale environments,” in *CVPR (to appear)*, June. 2012.
[3] E. Tola, V. Lepetit, and P. Fua, “A fast local descriptor for dense matching,” in *CVPR*, 2008.
[4] S. A. J. Winder, G. Hua, and M. Brown, “Picking the best DAISY,” in *CVPR*, 2009, pp. 178–185.
[5] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “BRIEF: Binary Robust Independent Elementary Features,” in *ECCV*, 2010.
[6] B. Williams, G. Klein, and I. Reid, “Real-time SLAM relocalisation,” in *ICCV*, 2007.
[7] G. Klein and D. Murray, “Parallel Tracking and Mapping for Small AR Workspaces,” in *ISMAR*, November 2007.
[8] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (voc) challenge,” *IJCV*, vol. 88, no. 2, pp. 303–338, June 2010.
[9] N. Snavely, S. M. Seitz, and R. Szeliski, “Modeling the World from Internet Photo Collections,” in *IJCV*, 2008.
[10] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski, “Towards internet-scale multi-view stereo,” in *CVPR*, 2010.
[11] D. Ta, W.-C. Chen, N. Gelfand, and K. Pulli, “SURFTrac: Efficient Tracking and Continuous Object Recognition using Local Feature Descriptors,” in *CVPR*, 2009.
[12] Z. Dong, G. F. Zhang, J. Y. Jia, and H. J. Bao, “Keyframe-based Real-time Camera Tracking,” in *ICCV*, 2009.

[2]The FitPC, which has an Intel Atom Z550 2GHz CPU, 2GB RAM and a 64GB SSD drive, weighs less than 500gms. (incl. battery) and consumes only 10W at full load.

Object Categorization through Annotated Reeb Graph and Grasping by Parts

Dario Lodi Rizzini, Jacopo Aleotti, Stefano Caselli
RIMLab - Robotics and Intelligent Machines Laboratory
Dipartimento di Ingegneria dell'Informazione, University of Parma, Italy
E-mail {*dlr,aleotti,caselli*}*@ce.unipr.it*

Recently, the proliferation of cheap and effective sensor devices for 3D perception, like depth cameras and 3D laser systems, has promoted the development of accurate and detailed object detection. The perception, representation and classification of objects is not only relevant for navigation tasks and semantic description of unstructured environments, but also has great importance for robot manipulation and grasping. Grasping action is better guided by object affordances that can be extracted from the object parts like protruding segments. Some categories of human artifacts often exhibit specific parts made for grasping, like the handles in hammers and mugs. Thus, the affordances of such objects could be detected through their categorization and semantic segmentation. Classification is often based on appearance features extracted from specific observed objects. On the other hand, categorization is the identification of the class an object belongs to and must depend on invariant attributes related to the shape and structure.

In this work, we extend the manipulation planning system presented in [1] for a robot arm with eye-in-hand laser scanner that is capable of performing 3D object reconstruction, segmentation, categorization and grasp planning on selected parts of the objects. The experimental setup consists of a Comau SMART SiX manipulator (6 dofs) equipped with a Schunk PG-70 gripper and a SICK LMS 400 laser range-finder mounted in eye-in-hand configuration on the manipulator wrist. The range measurements are gathered during the motion of the robotic arm and registered to achieve a complete point cloud. The system observes one or more objects lying on a dominant plane. Objects are detected by removing outliers, extracting the dominant plane and clustering the points above the plane. In particular, clustering is efficiently performed by exploiting the spatial and temporal coherence of scans to build a proximity graph.

Since polygonal mesh representation is required to plan motion and grasping, the object surface is reconstructed using power crust algorithm [2]. The obtained mesh is segmented by constructing its corresponding Reeb graph based on the integral geodesic function [3]. The Reed graph allows the identification of object parts which are candidate for being grasped. In our experiments, four categories have been considered: doll, jug, horse and table. Object class is recognized by matching the computed graph with the graph of each category. Figure 1 illustrates the processing steps for two instances of the horse class. Finally, the grasping and robot arm motion are planned and performed on the proper affordance for the object class.

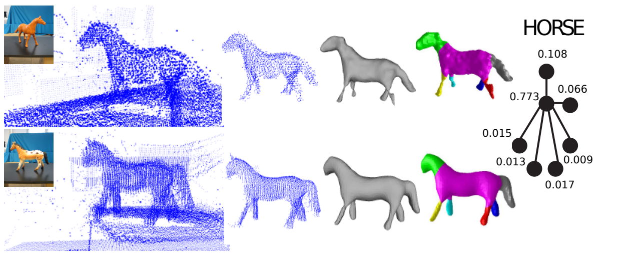


Fig. 1. From left to right the raw scan data (with a small picture of the real object), the clustered point cloud of the object, the reconstructed mesh (in gray), the colored segmented mesh and the annotated Reeb Graph for two instances of horse class (for clarity only the vertex labels are shown).

Reeb graph segmentation provides a topological decomposition of the shape of an object which is likely to identify semantic object's parts that are candidate components for being grasped. However, it could occur that two different classes have the same Reeb graph. To overcome this ambiguity we improve the categorization method proposed in [1] by using the *annotated Reeb Graph*. We propose to label each node corresponding to an object part with its normalized volume and each edge with the cosine of the angle between the principal inertial axes of the object parts connected by the edge. The prototypical Reeb graph is annotated with the average values and variance of the weigths computed on a training set. The categorization of an object is solved as a labeled graph matching problem by manipulating the weighted adjacency matrices. The permutation matrix P that maps the vertices of the graph to classify to those of each prototype graph is computed. Then, the distance between the object and a class prototype is measured by the distance between their corresponding adjacency matrices $\|A_1 - P\ A_2\ P^{-1}\|$. Thus, the proposed metric allows to discriminate between topologically similar object classes without relying on too specific geometric features.

References

[1] J. Aleotti, D. Lodi Rizzini, and S. Caselli. Object categorization and grasping by parts from range scan data. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, may 2012.

[2] N. Amenta, S. Choi, and R. K. Kolluri. The Power Crust. In *Proc. 6th ACM sym. on Solid modeling and applications*, pages 249–266, 2001.

[3] S. Berretti, A. Del Bimbo, and P. Pala. 3D Mesh decomposition using Reeb graphs. *Image and Vision Computing*, 27(10):1540–1554, 2009.

From Object Categories to Grasp Transfer Using Probabilistic Reasoning

Marianna Madry Dan Song Danica Kragic

***Abstract*— We address the problem of transferring task-specific grasp strategies from a known to novel object using information about an object category. Our system relates knowledge about: (a) several physical object properties, (b) object functionality and (c) task constrains in order to find a suitable grasp. We demonstrate that by choosing an object representation that encodes diverse objects properties and integrates information from different visual sensors our system can not only find objects that afford an assigned task, but also generate and successfully transfer task-based grasp within and across object categories.**

Perception of and interaction with objects is one of the key requirements for a robot acting in the real environment. Although excellent examples of finding and manipulating a *specific* object in a scene have been reported (Hasio et al., 2010, Welke et al. 2010) there is no system that is capable of flexibly and robustly localizing and grasping an arbitrary object that fulfills a certain functionality. Thus, executing tasks such as "**Robot, bring me something to drink from.**" or "**Robot, give me something to hammer with.**".

The aspect of function is related to that of affordances (Greeno, 1994) and has been addressed in works that learn relations between objects and actions (Fritz et al., 2006, Sahin et al., 2007). However, none of these consider the role of *task* in their model: task intention of the agent affects the type of action (grasp) to apply, i.e. not just any grasp can be applied on the object, see Fig. 1.

We present a system that allows not only to plan a suitable grasp for a given task, but also enables transfer of grasp knowledge between objects that share similar physical attributes and/or have the same functionality. To this end, we develop and evaluate an object representation that links information about an object, task and action.

The authors are with Computer Vision and Active Perception Lab, Center for Autonomous Systems, KTH-Royal Institute of Technology, Sweden, madry,dsong,danik@csc.kth.se. This work was supported by GRASP, IST-FP7-IP-215821 and Swedish Foundation for Strategic Research. We thank *Jeannette Bohg* and *Carl Henrik Ek* for their help.

The experimental setup is a scene containing multiple objects. Individual object candidates are first segmented (Björkman et al., 2010), categorized and then used as the input to a grasp generation and transfer module (Song et al., 2010). We integrate various 2D and 3D features describing object texture, color and shape to obtain a robust object representation. We demonstrate that encoding complementary object properties, not only significantly improves robustness of the categorization system, but assures relevant balance between discrimination and generalization in the representation. This allows to distinguish objects that both belong to the same functional category, but significantly differ in physical properties, and objects that afford different tasks, but are alike in color and shape.

The most suitable grasp parameters are inferred by the probabilistic grasp reasoning module based on information about an object category and assigned task. Results are illustrated by marking the grasp point probability around each object. For the *pouring* task (Fig. 2b), the likelihoods of the points around the mugs and bottle are clearly higher than for the screwdriver indicating that they are the only objects affording the task. Moreover, their likelihood maps are darker on the top, since the robot hand should not block the opening of an object. Summarizing, the proposed framework enables a robot to choose the objects that afford the assigned task while plan the grasp that satisfies the constraints posed by the task, and transfer grasp knowledge between objects that belong to the same category, even under considerable differences in appearance and physical properties.

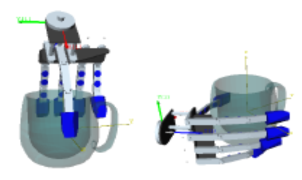

(a) *hand-over* (b) *pouring*

Fig. 1. Grasping a cup: (a) *hand-over* and (b) *pouring* task (hand should not block the opening of an object when pouring a liquid).

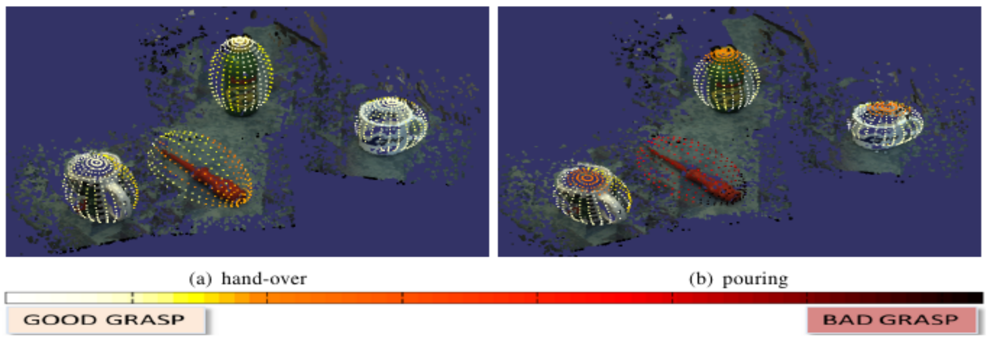


Fig. 2. Grasp hypotheses and associated probabilities for: (a) *hand-over* and (b) *pouring* task. The grasp point probability around an object is indicated by color of a point: the brighter is the point, the higher is the probability. Experimental results for a number of natural scenes and five different tasks are available on our website http://www.csc.kth.se/~madry/research/madry12icra

Towards Reliable Object Anchoring in Highly Dynamic Traffic Scenes

Ming Li, Wei Li, Jian Wang, Qingquan Li, and Andreas Nüchter

Symbol grounding is the problem of how the meaning of a symbol is to be grounded in something other than just more meaningless symbols [3]. This open problem in robotics and artificial intelligence is most challenging. The object anchoring problem is the lightweight version of symbol grounding, as it restricts the symbols to refer to objects, thus avoiding abstract concepts or attributes [2].

This demonstration presents work in progress, a novel approach for mapping highly dynamic environments, i.e., we design a system capable for mapping road traffic scenarios like given in Figure 1. Given 3D laser scans acquired at a high frame rate and no other sensor input, a 3D map is built by removing dynamic parts of the scene and estimating the ego-motion of the vehicle precisely at the same time. To achieve this, we have combined reliable and fast 3D scan matching in an ICP-like fashion with semantic perception, object tracking and recognition.

We extend the well-known ICP algorithm, available in "3DTK – The 3D Toolkit" [1] for HDL-64 laser scan data and build a system for solving the simultaneous localization and mapping problem in urban road scenarios. As the pose of the car is unknown, the geometric structure of overlapping 3D scans has to be considered for registration. However, this structure is changing, due to change of position of the other moving objects. Therefore, moving objects need firstly to be identified and be removed. We use a semantic-driven approach for solving this task of identifying dynamic objects in 3D scans. The overall system is called dynamic VeloSLAM. To be more precise, we execute the following steps:

1) First, we segment the 3D point cloud. Here a problem arises with all objects standing on the ground. For example, the feet of a human have roughly the same height value as the ground at the point he is standing on. The feet and the floor form only a crease edge, no jump edge. Thus, we use a special ground removal method and then the objects are nicely separated.
2) After the ground segmentation, one scan of point cloud was divided into many separated objects. In a typical urban environment, there is a wide variety of objects such as vehicles, pedestrians, buildings, etc. We use low-level geometric features, i.e., PCA features, for classification of the objects.
3) The interpreted objects are tracked using a Kalman filter. Each cluster is tracked separately, and the tracking is used to improve the classification described above. The multi-hypothesis approach in combination with either nearest-neighbor data association or global optimization give the algorithm its stability.
4) Dynamic VeloSLAM considers clusters, which are not matched as possible new targets, and a new tracker is initialized. If a cluster cannot be matched, we keep the hypothesis for a certain number of 3D scans, thus ensuring the possibility for reacquisition. After exceeding the threshold, the tracker is deleted.
5) If the motion detected of a cluster exceeds a threshold, we consider the object as dynamic. All dynamic objects are removed for recovering the vehicle motion and to build a map.
6) Finally, we use our 6D SLAM method as described in [4]. Its basis is a fast and reliable scan matching algorithm for ICP, a heuristic for closing loops efficiently, and a Lu/Milios-style relaxation.

To validate our approach, we collected a data set on the road from Wuhan to Huangshi. We select a roundabout in the road as main experimental scene 1. The demonstration show the results on the mentioned data set.

The research was partially supported by the National Natural Science Foundation of China (Research Fund for International Young Scientists, Program NO: 41050110437 and Research Fund for Youth Project, Program NO: 41001306) and the Sino-German Center for Research Promotion GZ 692. The authors are with the Wuhan University, China and the Jacobs University Bremen, Germany. Contact: {andreas@nuechti.de}

REFERENCES

[1] Andreas Nüchter et al. 3DTK – The 3D Toolkit. `http://threedtk.de/`, February 2012.
[2] S. Coradeschi and A. Saffiotti. An introduction to the anchoring problem. *Journal of Robotics and Autonomous Systems (JRAS)*, 43(2–3):85–96, 2003.
[3] S. Harnad. The symbol grounding problem. *Physica D*, 42:335–346, 1990.
[4] J. Sprickerhof, A. Nüchter, K. Lingemann, and J. Hertzberg. A Heuristic Loop Closing Technique for Large-Scale 6D SLAM. *AUTOMATIKA – Journal for Control, Measurement, Electronics, Computing and Communications, Special Issue with selected papers from the 4th European Conference on Mobile Robots*, 52(3), December 2011 (accepted).

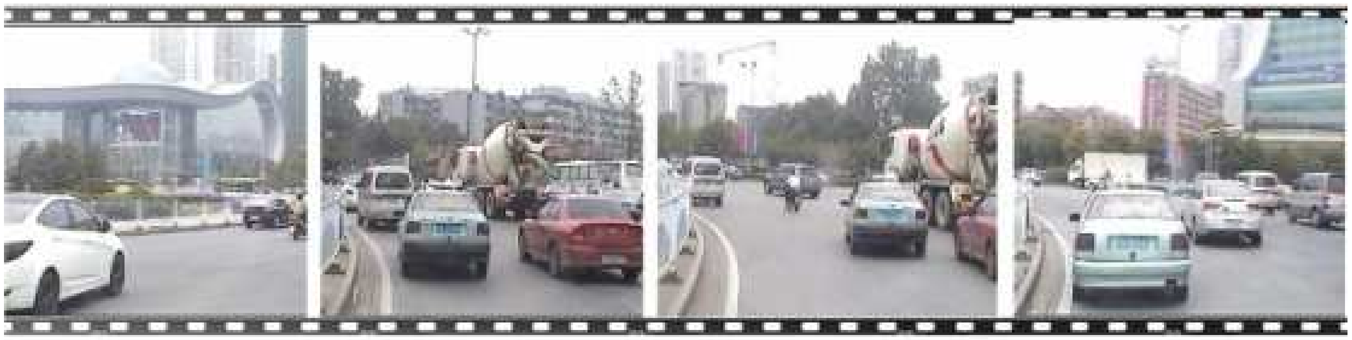


Fig. 1. Typical scenario, where we aim at precise 3D mapping. See also `http://youtu.be/bHaZpQ_5wg8`

Leveraging Semantic Context: Robot Planning with a Conditional Random Field Model

John G. Rogers III and Henrik I. Christensen

Service robots can assist humans in a variety of domestic tasks such as meal preparation, fetch and carry, and clean-up. Robots can leverage contextual cues within their environments to help them understand how to perform their duties. Context can take many forms such as that which exists between objects (a light switch is close to a door), as well as that which exists between places and objects (certain things are found in certain rooms). By leveraging place and object context, robots can be made to understand domestic environments and perform object search tasks more efficiently.

We have developed a technique described in [3] which uses a probabilistic model to represent the context of objects in places to perform an object search task in a domestic environment. We extend the idea of using *virtual objects and rooms* of [1] through the use of our probabilistic model. We leverage the context of places seen in a topological map when considering which unknown region to explore, to be the one most likely to contain the object for which the robot is searching. Each object found in a room will influence the label on the room and affect the likelihood of finding the target object.

A variety of software and hardware modules are used in our mobile robot system. Our robot is a Segway RMP200 mobile robot base with Hokuyo UTM30 as well as SICK laser scanners. The robot has a Directed Perception PTU 46-70 with an Asus Xtion Pro Live 3D camera, a Nikon D90 DSLR camera. The robot also has a Schunk gripper is also mounted on a linear actuator for picking up objects. The software is based upon the Robot Operating System (ROS) [2]. We use the GTsam nonlinear optimization library with a modular mapping framework that we have developed called OmniMapper. Room regions are segmented using a Gaussian region analysis technique described in previous work. Objects are segmented on tabletop surfaces through 3D point cloud analysis. Object recognition is performed by matching geometrically consistent SURF features against a database of known objects. If the object is not recognized in a database of known objects, then it is classified by a group of relevance vector machines trained on visual word histograms quantized from SURF features.

We introduced a probabilistic cognitive model (PCM) for place and object classification using conditional random fields in [3]. Objects and rooms are each represented as nodes in an undirected graphical model; their labels are given by multinomial distributions. There are two types of edges in this graph: between room nodes which indicate adjacency in a topological map, and between objects and rooms which indicate that object is within that room. The posterior distribution on this graphical model is approximated by loopy belief propagation. Objects which fail to be recognized by the object recognition module, but have been categorized by the object categorization module are given the specific measurement from the object classification module as their prior measurement. If the object recognition module recognizes the object, then it is clamped to the recognized class. Clamped nodes are used to select conditional probability tables in their neighbors and are skipped by the loopy belief propagation algorithm. If a new measurement is made of a previously mapped object, then the distribution with the lowest entropy is used as the measurement in the graphical model.

We have developed a planning module which uses the probabilistic model to select robot actions to perform an object search task. The PCM Planner selects actions in a Markov Decision Process framework. At each state, the robot chooses an action from the set *Move*, *Search*, *Examine*, *Fetch*. The *Move* action can be performed to transition to another room which is topologically adjacent to the one in which the robot currently inhabits. The PCM hypothesized by the planner for this action is the same as the current PCM, but with the robot in the desired adjacent room. The *Search* action is used to try to find objects in the current room using the segmentation module. The result of the *search* action in the hypothesized next state is that a new object is found by the robot. A uniform prior is placed on this new object; after the posterior is computed with LBP, the context of the current room will affect the label of the hypothesized object. The *Examine* action is selected by the planner to look at a previously segmented or categorized object and try to identify it directly with the recognition module. The hypothesized next PCM state upgrades the object categorized label distribution to a clamped, known recognition. The final action, *Fetch*, is terminal and represents the robot making its final choice; choosing this object as the solution to the object search task.

The PCM planner computes a sequence of actions leading it to the terminal *Fetch* action; however, since this relies heavily upon hypothesized results, we only execute the first action of the sequence and then re-plan based upon its result. The *Move* action sends coordinates of the center of the target room Gaussian as the destination for the motion controller. The *Search* action instructs the base controller to move to waypoints aligned with the major and minor axes of ellipsoidal regions described by the covariance matrix of the room Gaussian. While the robot is moving to these positions, objects on table surfaces are likely to be seen as they pass nearby. The *Examine* action moves the base to within 2 meters of the target object, saccades the PTU to aim the DSLR camera, and takes a high resolution image which is processed by the recognition and classification modules.

We have presented a technique for leveraging contextual cues in the form of room adjacency and object in room affinity in a Markov decision planner framework. The problem addressed is to leverage the context offered from room and object identification to find an element of an object class in an unknown indoor environment.

References

[1] A. Aydemir, M. Göbelbecker, A. Pronobis, K. Sjöö, and P. Jensfelt. Plan-based object search and exploration using semantic spatial knowledge in the real world. In *Proc. of the European Conference on Mobile Robotics (ECMR'11), Orebro, Sweden*, 2011.

[2] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng. ROS: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[3] J. G. Rogers III and H. I. Christensen. A conditional random field model for place and object classification. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

World Modeling in Robotics: Probabilistic Multiple Hypothesis Anchoring

Sjoerd van den Dries Jos Elfring René van de Molengraft Maarten Steinbuch

I. Introduction

Domestic robots are typically confronted with a complex, dynamically changing and unstructured environment. In order to adequately perform tasks such as safe navigation and object manipulation, robots need an accurate description of the world. Such an environmental description, here referred to as *world model*, is constructed based on perceived object features and must store and keep track of the objects in the world and their attributes.

Requirements of such a world modeling algorithm are: **(1)** *semantic richness:* object descriptions must be semantically annotated, *i.e.*, stored with human-readable attributes such as *position*, *color* and *class* such that human-defined tasks can be mapped to the world description, and the algorithm must allow multiple attributes to be stored and maintained simultaneously; **(2)** *probabilistic nature:* due to sensor limitations and prediction uncertainty, uncertainty must be explicitly represented and dealt with in the system by using probabilistic representations and calculations; **(3)** *data association and attribute maintenance:* measured object features must be linked to and update object descriptions, while dealing with possible ambiguities; **(4)** *exploitation of prior knowledge:* existing knowledge about the behavior of objects, *e.g.*, movement, expected locations, should be used as much as possible; **(5)** *computationally feasible and scalable:* the system must run real-time on a robot, even in cluttered scenarios with many objects.

We present Probabilistic Multiple Hypothesis Anchoring (PMHA), a world modeling algorithm that fulfills these requirements by uniquely extending perceptual anchoring ([2]) with an efficient Bayesian multiple hypothesis framework ([3], [1]) and a generalized multiple-model object tracking approach ([4]).

II. Probabilistic Multiple Hypothesis Anchoring

The PMHA algorithm works as follows:

- Measured object features are received as input from perception routines. These features are assumed to have a probabilistic representation, explicitly stating the amount of uncertainty. (*Req. 2*)
- The measured features are mapped to semantically rich structures using a predicate grounding relation known from anchoring, *e.g.*, certain hue-saturation-value color ranges are mapped to the color 'blue', relative coordinate frames are mapped to absolute coordinate frames. (*Req. 1*)
- Semantic features are associated to new objects or existing objects in the world model by generating multiple association hypotheses according to the multiple hypothesis framework. Hypothesis probabilities are calculated using a Bayesian prior-likelihood calculation. (*Req. 2, 3*)
- Within each hypothesis, object attributes are updated based on the probabilistic semantic features. The object attributes are also stored in a probabilistic way, *e.g.*, the Cartesian position is stored as a Gaussian. (*Req. 2, 3*)
- Non-measured object attributes are predicted using a multi-model approach: multiple *behavior models* may predict future states of different attributes, *e.g.*, position or color, again generating multiple hypotheses. The behavior models can be both *physics-based*, *e.g.*, a constant-velocity model, and *common-sense based*, *e.g.*, a model stating that *John comes home at 5pm.* (*Req. 4*)

To ensure computational feasibility (*Req. 5*), gating, limited hypotheses generation and hypothesis pruning are used, as well as efficient implementation of the algorithm using track trees and clustering ([3], [1]).

III. Experiments and Future Work

The potential of the algorithm has been demonstrated in several experiments, including the tracking of multiple people in a room, solving ambiguities about the identities of objects, and verification and falsification of object positions and colors based on cheap perception routines. Future work includes extending the behavior models and predicate grounding relations used, as well as combining the multiple hypothesis framework with more efficient methods for storing and retrieving object descriptions.

References

[1] S. S. Blackman. Multiple hypothesis tracking for multiple target tracking. *Ieee Aerospace And Electronic Systems Magazine*, 19(1):5–18, 2004.

[2] S. Coradeschi and A. Saffiotti. An introduction to the anchoring problem. *Robotics And Autonomous Systems*, 43:85–96, 2003.

[3] I. J. Cox and S. L. Hingorani. An efficient implementation of Reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18:138–150, February 1996.

[4] X. Rong Li and V. Jilkov. Survey of maneuvering target tracking. part v. multiple-model methods. *Aerospace and Electronic Systems, IEEE Transactions on*, 41(4):1255–1321, oct. 2005.

All authors are with the Faculty of Mechanical Engineering, Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands {s.v.d.dries, j.elfring, m.j.g.v.d.molengraft, m.steinbuch} at tue.nl.

The research leading to these results has received funding from the European Union Seventh Framework Program FP7/2007-2013 under grant agreement n^{o}248942 RoboEarth.

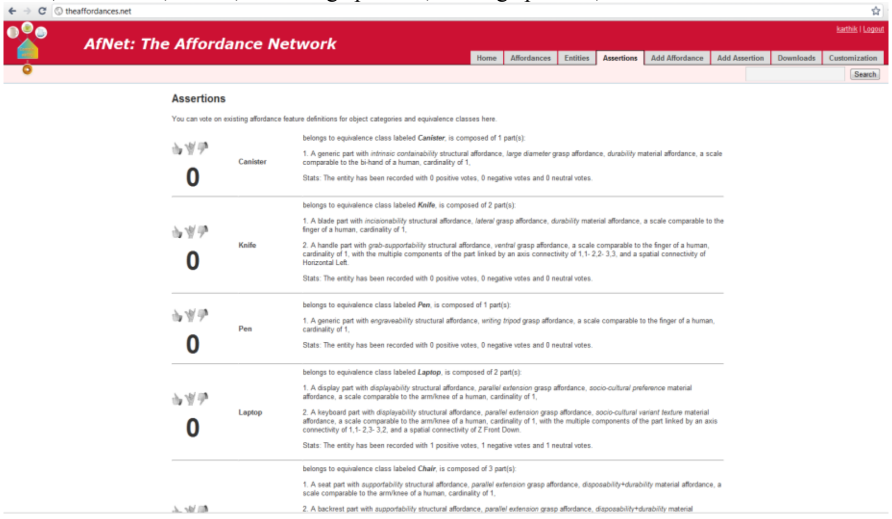
Open Affordance Feature and Entity Definition Framework - AfNet

Karthik Mahesh Varadarajan, Markus Vincze

Given the need for task based visual environment processing for domestic robots as well as object category representation in terms of robust features that are insensitive to instance specific variations in color, texture and geometry, Recognition of objects/object categories by Component Affordances (RBCA) has attained a critical role. While the use of ontologies for representation of semantic knowledge about objects based on local features (such as SIFT) or geometry (such as CAD), suffers from scalability issues, affordance features provide a convenient mechanism to circumvent the issue. AfNet – The Affordance Network (www.theaffordances.net) provides a concise set of affordance features for object/ object class description. AfNet is designed as an open framework, enabling community based addition of new affordance features as well as object/ object descriptions (termed as entity descriptions) based on these affordance features. AfNet represents the first initiative towards standardizing affordance features and rendering a community interface to semantic representation of objects using affordances.

In this paper, we describe the structure of the Web Learning Interface provided by AfNet, which can be used to teach the system new affordance features as well as define objects or classes using existing affordance features. AfNet represents each entity or conceptual equivalence class in terms of various structural (SA), material (MA), grasp (GA) affordances and topological relationships between the various components which constitute the object in question. AfNet currently provides 68 affordance types – 25 structural, 10 material and 33 grasp affordances, supporting over 250 equivalence classes. Community users can teach the system new affordance types by defining an affordance label, its level 0-4 categorization in the affordance hierarchy (see Fig 1b), along with textual definitions, geometric mapping information and examples of objects or classes that possess the affordance. In addition, users can also define assertions – i.e. users can define objects or equivalence classes in terms of a maximum of 5 labeled or generic parts, each providing single or multiple affordances (from all three categories of affordances), along with the scale of each part and its cardinality. The scales of the object parts can be defined in abstract measures with respect to the human arm: sizes comparable to finger, hand, bi-hand, arm/knee, torso, standing posture, sitting posture, non-conformal. Scales of the parts can also be specified in terms of ranges. In addition, abstract topological relationships (axial and spatial connectivity) between the object parts can also defined with respect to Part Connectivity Calculus (PCC). Given the three axes of local orientation for each part, there are 6 possible axial connectivity relationships. The spatial connectivity is defined on a 18-connectivity 3D grid, each of which is uniquely represented by a label according to PCC. In addition, the AfNet interface allows the optional storage of generic data, sample color and range/point cloud data corresponding to each object/class entity. In addition, similar to the Open Mind Indoor Common Sense (OMICS), the AfNet web interface allows for community based voting on affordance feature based entity definitions (or assertions). The voting interface presents options to the community user to provide his response with regard to the fidelity of the available representation for a list of entities which is updated in real-time (Fig 1a). The votes garnered through this process are stored in the records for each entity and is available for further processing. The online entity interface also provides a mechanism to search the database using entity labels (such as cup, table etc.). AfNet stores the assertion records in tables composited into a SQLite3 database, which can be downloaded for offline usage (on domestic robots). Alternatively, SparQL and RDF bindings are also provided, thereby enabling support for complex queries. Since AfNet defines symbol binding mechanisms for each affordance, users can mix and match visual affordance detection algorithms (pre-defined or user-defined) to enable perceptual linking of objects of interest in the given scenario with the knowledge inference schema.

As samples, AfNet, defines a pen as belonging to Equivalence Class labeled Pen, composed of 1 part(s): 1. A Generic part with Engraveability SA, Writing Tripod GA, a scale comparable to the Finger of a human, cardinality of 1. The definition for a Knife is: belongs to Equivalence Class labeled Knife, composed of 2 part(s): 1. A Blade part with Incisionability SA, Lateral GA, Durability MA, a scale comparable to the Finger of a human, cardinality of 1, 2. A Handle part with Grab-supportability SA, Ventral GA, a scale comparable to the Finger of a human, cardinality of 1, with the multiple components of the object linked by an axis connectivity of 1,1- 2,2- 3,3, and a spatial connectivity of Horizontal Left.



Assertions

You can vote on existing affordance feature definitions for object categories and equivalence classes here.

0 — Canister

belongs to equivalence class labeled *Canister*, is composed of 1 part(s):

1. A generic part with *intrinsic containability* structural affordance, *large diameter* grasp affordance, *durability* material affordance, a scale comparable to the bi-hand of a human, cardinality of 1,

Stats: The entity has been recorded with 0 positive votes, 0 negative votes and 0 neutral votes.

0 — Knife

belongs to equivalence class labeled *Knife*, is composed of 2 part(s):

1. A blade part with *incisionability* structural affordance, *lateral* grasp affordance, *durability* material affordance, a scale comparable to the finger of a human, cardinality of 1,

2. A handle part with *grab-supportability* structural affordance, *ventral* grasp affordance, a scale comparable to the finger of a human, cardinality of 1, with the multiple components of the part linked by an axis connectivity of 1,1- 2,2- 3,3, and a spatial connectivity of Horizontal Left.

Stats: The entity has been recorded with 0 positive votes, 0 negative votes and 0 neutral votes.

0 — Pen

belongs to equivalence class labeled *Pen*, is composed of 1 part(s):

1. A generic part with *engraveability* structural affordance, *writing tripod* grasp affordance, a scale comparable to the finger of a human, cardinality of 1,

Stats: The entity has been recorded with 0 positive votes, 0 negative votes and 0 neutral votes.

0 — Laptop

belongs to equivalence class labeled *Laptop*, is composed of 2 part(s):

1. A display part with *displayability* structural affordance, *parallel extension* grasp affordance, *socio-cultural preference* material affordance, a scale comparable to the arm/knee of a human, cardinality of 1,

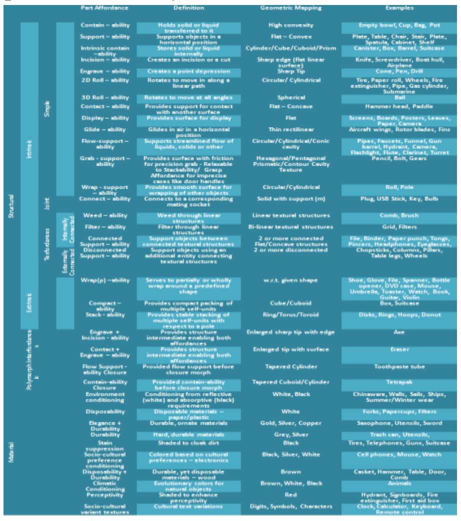
2. A keyboard part with *displayability* structural affordance, *parallel extension* grasp affordance, *socio-cultural variant texture* material affordance, a scale comparable to the arm/knee of a human, cardinality of 1, with the multiple components of the part linked by an axis connectivity of 1,1- 2,3- 3,2, and a spatial connectivity of Z Front Down.

Stats: The entity has been recorded with 1 positive votes, 1 negative votes and 1 neutral votes.

belongs to equivalence class labeled *Chair*, is composed of 3 part(s):

1. A seat part with *supportability* structural affordance, *parallel extension* grasp affordance, *disposability+durability* material affordance, a scale comparable to the arm/knee of a human, cardinality of 1,

2. A backrest part with *supportability* structural affordance, *parallel extension* grasp affordance, *disposability+durability* material



				Part Affordance	Definition	Geometric Mapping	Examples
Structural	Intrinsic	Simple		Contain – ability	Holds solid or liquid transferred to it	High convexity	Empty bowl, Cup, Bag, Pot
				Support – ability	Supports objects in a horizontal position	Flat – Convex	Plate, Table, Chair, Stair, Plate, Spatula, Cabinet, Shelf
				Intrinsic contain –ability	Stores solid or liquid internally	Cylinder/Cube/Cuboid/Prism	Canister, Box, Barrel, Suitcase
				Incision – ability	Creates an incision or a cut	Sharp edge (flat linear surface)	Knife, Screwdriver, Boat hull, Airplane
				Engrave – ability	Creates a point depression	Sharp Tip	Cone, Pen, Drill
				2D Roll – ability	Rotates to move in along a linear path	Circular/ Cylindrical	Tire, Paper roll, Wheels, Fire extinguisher, Pipe, Gas cylinder, Submarine
				3D Roll – ability	Rotates to move at all angles	Spherical	Ball
				Contact – ability	Provides support for contact with another surface	Flat – Concave	Hammer head, Paddle
				Display – ability	Provides surface for display	Flat	Screens, Boards, Posters, Leaves, Paper, Camera
				Glide – ability	Glides in air in a horizontal position	Thin rectilinear	Aircraft wings, Rotor blades, Fins
				Flow-support – ability	Supports streamlined flow of liquids, solids or other	Circular/Cylindrical/Conic cavity	Pipes, Faucets, Funnel, Gun barrel, Hydrant, Camera, Flashlight, Flute, Clarinet, Turret
				Grab - support – ability	Provides surface with friction for precision grab - Relaxable to Stackability/ Grasp Affordance for imprecise cases like door handles	Hexagonal/Pentagonal Prismatic/Contour Cavity Texture	Pencil, Bolt, Gears
				Wrap - support – ability	Provides smooth surface for wrapping of other objects	Circular/Cylindrical	Roll, Pole
		Joint		Connect – ability	Connects to a corresponding mating socket	Solid with support (m)	Plug, USB Stick, Key, Bulb
		Texfordances	Internally Connected	Weed – ability	Weed through linear structures	Linear textural structures	Comb, Brush
				Filter – ability	Filter through linear structures	Bi-linear textural structures	Grid, Filters
				Connected Support – ability	Support objects between connected textural structures	2 or more connected Flat/Concave structures	File, Binder, Paper punch, Tongs, Pincers, Headphones, Eyeglasses,
			Externally Connected	Disconnected Support – ability	Support objects using an additional entity connecting textural structures	2 or more disconnected	Chopsticks, Columns, Pillars, Table legs, Wheels
	Extrinsic			Wrap(p) –ability	Serves to partially or wholly wrap around a predefined shape	w.r.t. given shape	Shoe, Glove, File, Spanner, Bottle opener, DVD case, Mouse, Umbrella, Toaster, Watch, Book, Guitar, Violin
				Compact – ability	Provides compact packing of multiple self-units	Cube/Cuboid	Box, Suitcase
				Stack - ability	Provides stable stacking of multiple self-units with respect to a pole	Ring/Torus/Toroid	Disks, Rings, Hoops, Donut
	Polymorph Interaffordances			Engrave + Incision - ability	Provides structure intermediate enabling both affordances	Enlarged sharp tip with edge	Axe
				Contact + Engrave – ability	Provides structure intermediate enabling both affordances	Enlarged tip with surface	Eraser
				Flow Support - ability Closure	Provided flow support before closure morph	Tapered Cylinder	Toothpaste tube
				Contain-ability Closure	Provided contain-ability before closure morph	Tapered Cuboid/Cylinder	Tetrapak
Material				Environment conditioning	Conditioning from reflective (white) and absorptive (black) requirements	White, Black	Chinaware, Walls, Sails, Ships, Summer/Winter wear
				Disposability	Disposable materials – paper/plastic	White	Forks, Papercups, Filters
				Elegance + Durability	Durable, ornate materials	Gold, Silver, Copper	Saxophone, Utensils, Sword
				Durability	Hard, durable materials	Grey, Silver	Trash can, Utensils,
				Stain suppression	Shaded to cloak dirt	Black	Tires, Telephones, Guns, Suitcase
				Socio-cultural preference conditioning	Colored based on cultural preferences – electronics	Black, Silver, White	Cell phones, Mouse, Watch
				Disposability + Durability	Durable, yet disposable materials – wood	Brown	Casket, Hammer, Table, Door, Comb
				Climatic Conditioning	Evolutionary colors for natural objects	Brown, White, Black	Animals
				Perceptivity	Shaded to enhance perceptivity	Red	Hydrant, Signboards, Fire extinguisher, First aid box
				Socio-cultural variant textures	Cultural text variations	Digits, Symbols, Characters	Clock, Calculator, Keyboard, Remote control

Figure 1. Left (a) Entity voting interface in AfNet. Right (b) Partial Affordance Hierarchy

The research leading to these results has received funding from the European Community's Seventh Framework Program under grant agreement no IST-FP7-IP-215821 (GRASP).Karthik Mahesh Varadarajan and Markus Vincze are with the Vienna University of Technology, Automation and Control Institute, Gusshausstrasse 30 / E376, A-1040 Vienna, Austria (email: {kv,mv} @acin.tuwien.ac.at).

Shape based Object Class Recognition from 3D CAD Models

Walter Wohlkinger and Aitor Aldoma and Markus Vincze

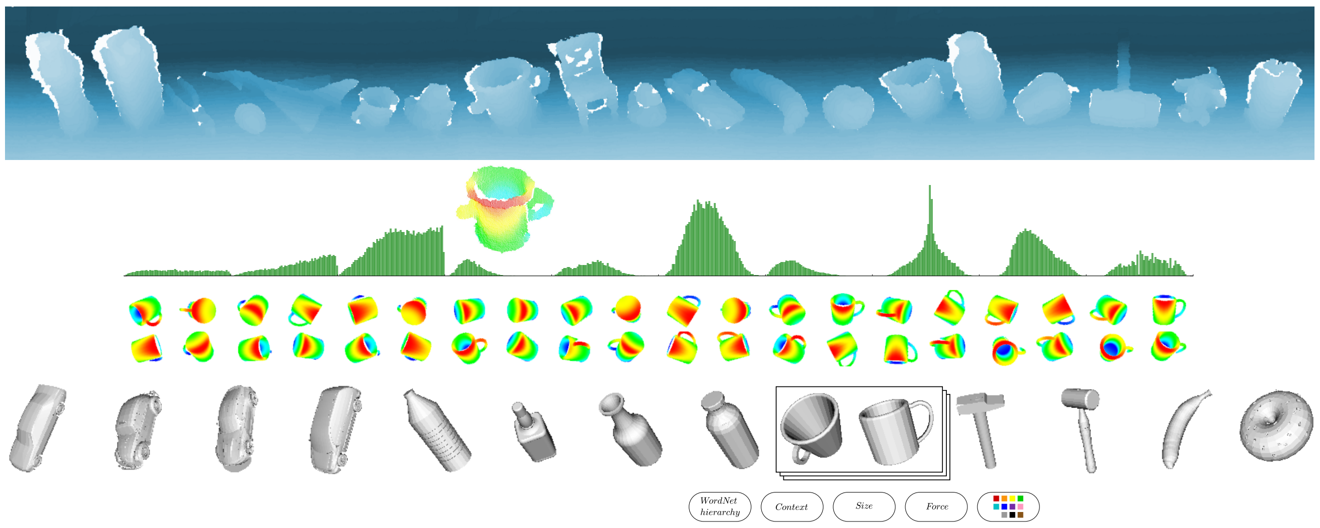


Fig. 1: The main idea of this paper is to use large collections of 3D CAD models as knowledge representations to explain the world. Hierarchically organized 3D models in large numbers build the basis for the system. Linking to WordNet and additional class specific information such as context, size, appearance and applicable force allows for semantic manipulation of object categories. A synthetic view generation process enables the matching algorithms to be trained on these synthetic models. Given objects captured by a 3D sensor as shown in the topmost row, the shape descriptors enable the matching of the sensed point clouds to synthetic views, approximately resembling the sensed object.

I. Abstract

The domestic setting with its plethora of categories and their intraclass variety demands great generalization skills from a service robot. The categories are characterized mostly by their shape ranging from low intraclass diversification as in the case of fruits and simple objects like bottles up to high intraclass variety of classes such as liquid containers, furniture, and especially toys. 3D object and object class recognition gained momentum with the arrival of low-cost RGB-D sensors and enables robotic tasks not feasible years ago. With these robots starting to tackle real-word scenarios, fast and reliable object and object class recognition is needed. Especially in robotic manipulation, where object recognition and object classification have to work from all possible viewpoints of an object, data collection for training becomes a bottleneck. Scaling object class recognition to hundreds of classes still requires extensive time and many objects for learning. To overcome the training issue, we introduce a methodology for learning 3D descriptors from synthetic CAD-models for classification of objects at first glance, where classification rates and speed are suited for robotics tasks. We provide this in 3DNet (**3d-net.org**), a free resource for object class recognition and 6DOF pose estimation from point cloud data. 3DNet provides a large-scale hierarchical CAD-model databases with increasing numbers of classes and difficulty with 10, 50, 100 and 200 object classes together with evaluation datasets that contain thousands of scenes captured with a RGB-D sensor. 3DNet further provides an open-source framework based on the Point Cloud Library (PCL) for testing new descriptors and benchmarking of state-of-the-art descriptors together with pose estimation procedures to enable robotics tasks such as search and grasping.

The proposed system only requires a 3D sensor such as a Microsoft Kinect or Asus Xtion and is able to deliver object classification results on a standard consumer notebook with 10 frames per second for scenes with objects on a flat support plane. With additional pose alignment, scale calculation and a scale invariant grasp planner, robotic grasping of categories can be tackled.

Wohlkinger, Aldoma and Vincze are with Vision4Robotics Group, Automation and Control Institute, Vienna University of Technology, Austria
`[ww,aa,mv] @ acin.tuwien.ac.at`

Mobile Manipulation Object Search using Co-occurrences and Capacity

Lawson L.S. Wong, Leslie Pack Kaelbling, and Tomás Lozano-Pérez

***Abstract*— Object search is an integral part of daily life, and in the quest for competent mobile manipulation robots it is an unavoidable problem. Previous approaches have suggested that object co-occurrence information is useful in object search. We present a novel generative model for representing both co-occurrence structure and spatial constraints, and use this to perform belief-space planning in object search. We demonstrate the model on a detailed simulation involving a PR2 robot.**

I. Introduction

Consider the following example of searching for a large mixing bowl in the kitchen when preparing a meal. There are three cupboard shelves that have been partially viewed, two containing stacks of plates, the other some dish detergent. Other objects appear to be in the back, but they are occluded, and we need to remove the objects in front to continue searching the shelves. Which shelf should we look at?

Even though the bowl has not been observed yet, it seems intuitive to us to keep looking on a shelf with plates because they have closer function to bowls. Suppose we further observe that of the two shelves with plates, one is small, the other large. If we assume a bowl, if present, is equally likely to be anywhere in the cupboard, and that both regions incur the same exploration cost, then the large region is more desirable to look at since more objects can be expected. Moreover, given that mixing bowls are usually large, we may even determine that the bowl cannot fit in the small region and eliminate that from consideration.

The above example illustrates two aspects of object search we wish to capture in our model. First, certain categories of objects tend to co-occur with each other, such as plates and bowls in the example. We will be agnostic to the causes of similarities (e.g., function, shape, color, etc.), and simply rely on co-occurrences as an indicator of similarity. Second, in the latter half of the example, we also used geometric information to choose between unseen regions. Here we understood that object types have various volume distributions, and knew that unseen regions had some remaining capacity. Our model captures both intuitions illustrated above.

Works such as [?] and [?] have previously demonstrated that co-occurrence information is useful for guiding object search. In particular, both use object co-occurrences to determine the compatability of the target object with various household locations, given observations of other objects in the vicinity. However, neither consider geometric constraints, nor are they easily extendible to do so. Moreover, manipulation is not integrated in either framework.

Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139 {lsw,lpk,tlp}@csail.mit.edu

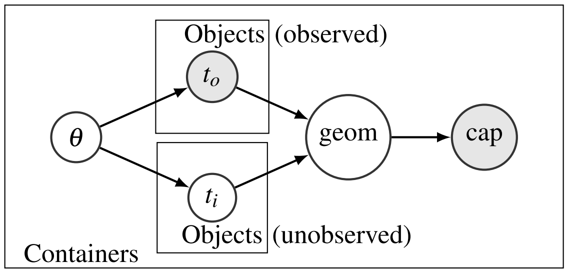


Fig. 1. Graphical representation of our model; see text below for details.

II. Model and Execution

We represent co-occurrence information and spatial constraints using the generative model shown in Figure **??**. We assume the environment is partitioned into a set of containers, each containing both seen and unseen object types. To model type similarity, we introduce the notion of a location's *composition* (θ), a latent distribution over object types, and supply a logistic-normal prior that captures type co-occurrences. The observed object types in the container then inform about the unknown ones via θ. By reasoning about the total volume that objects may occupy, we can check whether they fit within the known capacity of the container.

To find the target object, an agent needs to determine which containers likely contain the target object. This can be inferred from the above model, by determining in each container how likely the target object is one of the currently unobserved objects (if any exist). There are also costs associated with switching between containers and performing manipulation actions (e.g., moving occluding objects out of the way to reveal hidden objects). To tradeoff these costs, we use the online belief-space planner described in [?] to select next actions. The planner uses the probability that the query object type is in each container to determine whether to continue searching the current container, or to switch to another container if the current one seems unpromising.

The above object search framework was implemented in simulation with a mobile manipulator modeled on the Willow Garage PR2 robot. A sample execution can be found at: `http://youtu.be/cnWK8aBHmu0`

References

[1] T. Kollar and N. Roy, "Utilizing object-object and object-scene context when planning to find things," in *ICRA*, May 2009.
[2] M. Hanheide, C. Gretton, R. Dearden, N. Hawes, J. L. Wyatt, A. Pronobis, A. Aydemir, M. Göbelbecker, and H. Zender, "Exploiting probabilistic knowledge under uncertain sensing for efficient robot behaviour," in *IJCAI*, July 2011.
[3] L. P. Kaelbling and T. Lozano-Pérez, "Pre-image backchaining in the belief space for mobile manipulation," in *ISRR*, 2011.

Author Index

Keyword Index

Program Committee