Project Acronym:	GRASP
Project Type:	IP
Project Title:	Emergence of Cognitive Grasping through Introspection, Emulation and Surprise
Contract Number:	215821
Starting Date:	01-03-2008
Ending Date:	28-02-2012

Deliverable Number:	D29
Deliverable Title :	Surprise event demonstration on the project platforms
Type (Internal, Restricted, Public):	PU
Authors	D. Burschka, A. Morales, H. Deubel, T. Asfour
Contributing Partners	TUM, UJI, LMU, KIT

Contractual Date of Delivery to the EC:	09-02-2012
Actual Date of Delivery to the EC:	29-02-2012

Contents

Chapter 1

Executive summary

Deliverable D29 is part of WP5 -"Surprise: Detecting the Unexpected and Learning from it". According to the Technical Annex, it presents the activities in the context of

- **Task 5.2** - Evaluation of efficient methods to monitor changes in the environment that will be insensitive to sensor inaccuracies and that compensate eigen-motions/actions of the system in the environment. In collaboration with the WP4, an internal representation of the environment is generated that will define the expectations of the system. This representation goes beyond a geometric representation of the world and will define also contextual and dynamic information about the world
- **Task 5.4** - Generalisation of object form descriptions that allow to reduce the number of necessary geometric representations for a given object class. The system allows to generalize from a given basic geometry to the entire class of objects considering deformation of the geometry to match the current observation to the representation in the Atlas
- **Task 5.5** Provide a richer description of action. The research in this field focus on definition of relevant information that needs to be extracted from human actions that is of interest to plan the own actions of the robotic manipulator.

The work in this deliverable relates to the follwing fourth year milestone:

- **Milestone 10** - Linking structure, affordances, actions and tasks; evaluation of representations defined by the ontology.

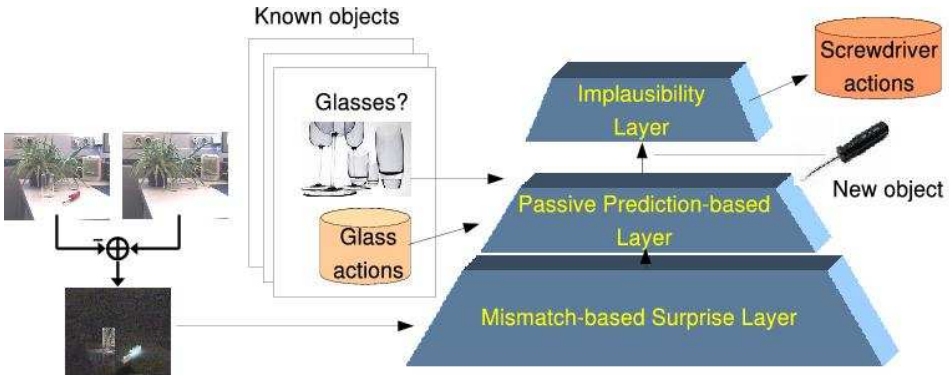


Figure 1.1: Hierarchical processing of a Surprise event (from GRASP Technical Annex)

In the past review meetings, we presented an integration of the perception required to use the surprise system on different platforms within the GRASP project. The surprise detection framework developed within the GRASP project is based on observation of objects, actions and results from defined interactions with the grasp targets. We focused the work on the following aspects in the fourth period of the project.

We worked on generalization of objects (Task 5.4) to reduce the number of reference shapes in the a-priori Atlas information. We defined action descriptions in our surprise framework that are able to represent necessary information from the observation of human actions and own active exploration performed by the system (Task 5.5). We worked on active exploration of objects to implement the learning loop from the Technical Annex, where a prediction-action-perception framework is used to refine physical properties of the object. We worked also on ways how to achieve a best possible knowledge-representation in the system to support the surprise detection in a most efficient way. The goal was to limit the number of surprise events that require the system to update its knowledge about physical properties of the object or its function in a given environment only to cases, where those actually changed. The system has to cope with the large variation in human actions keeping the number of surprise events to a necessary minimum.

We continued the work on implementation of the Surprise Event Hierarchy (Fig. 1.1) that is now used for efficient detection of points in time when an observed change in the way how an object is handled lets the system assume that the physical property of the object (e.g. its filling state) or the object function changed [5]. New in this period was the addition of Task 5.4 and Task 5.5. The generalization of object form descriptions leads to further generalization of the information stored in the Atlas information and allows to generalize between different instantiations of the same category of objects [2,3]. Fig. 1.2 shows a registration of a complex shape for a better visualization of the system performance. This processing allows to keep one representative of a shape in the Atlas and allows beside the already presented indexing function to the entries in the Atlas additionally a deformation map that can be used for grasp adaptation. The adaptation can use the grasp planning for the reference object to infer a good way to handle the deformed candidate.

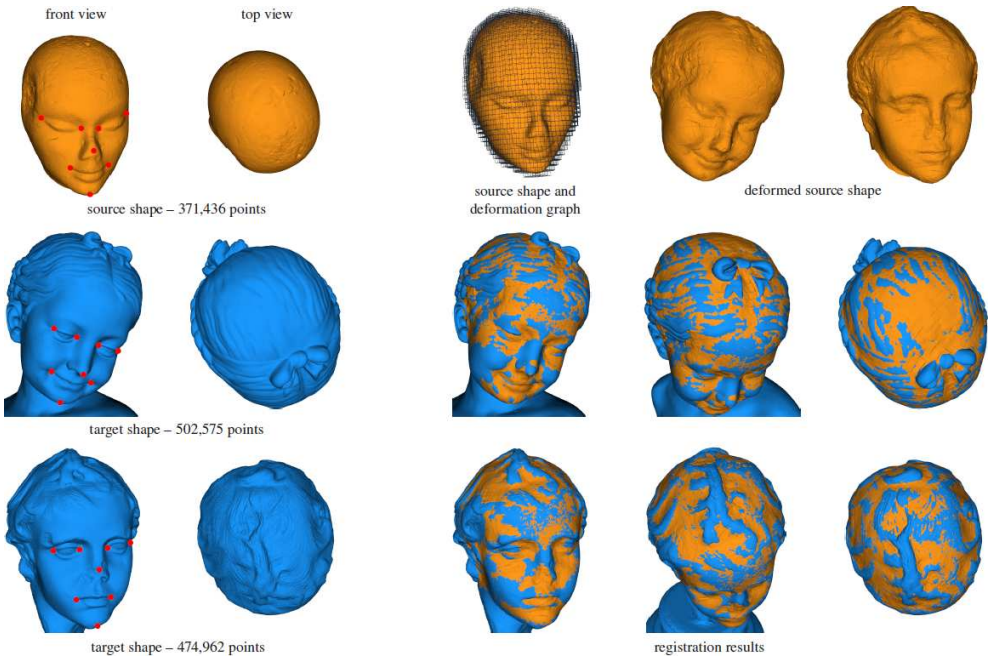


Figure 1.2: Registration of deformed shapes to a single initial reference shape (left). Each line shows registration results for the two deformations

We worked on making the information about the object richer in by observing not only the static geometric but also the dynamic properties of the object [6] that allows to estimate relevant physial attributes of the object which are not observable through simple passive observation (Fig. 1.3). We performed a planing of appropriate actions that will allow us to estimate the physical attributes of the object. The initial center of mass hypotheses is generated from the apriori assumption that the object is solid with uniform density distrubution. After striking the object at pre-planned point (based on estimated Equilibrium Planes (see above), we are able to estimate the true center of mass and the mass distribution of the object without inspecting its interior.

We refined our action parsing framework to be able to compare the already known information in the system with the current perception of the system (Fig. 1.4).

Figure 1.3: Registration of deformed shapes to a single initial reference shape (top line left). Each following line shows registration results for the two deformations

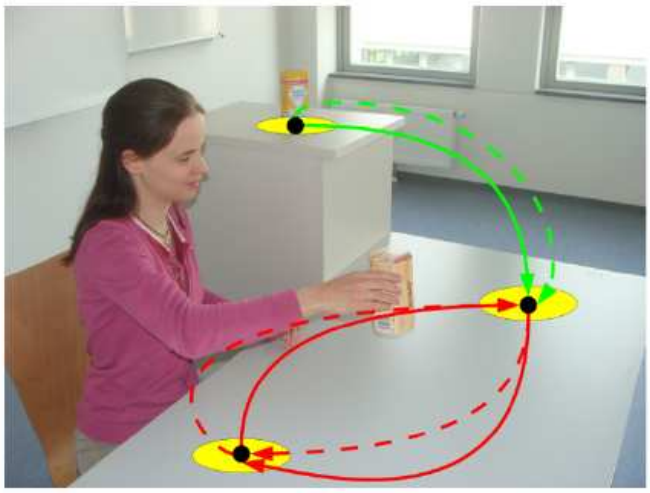


Fig. 1. The system creates an abstract map of possible manipulation actions and goals in the environment.

Figure 1.4: The system is observing the human action and tries to detect changes in actions that require a change in the Object Contrainer (physical properties of objects) or in the Functionality Map (framework representing transport relations and places of object occurance).

Chapter 2

Description of Work

We motivated our original Surprise definition already in the Technical Annex. The Surprise event is a mismatch between the prediction from a knowledge database in the system and the observation of the robot. Therefore, a knowledge database is only a source of information to predict expected behaviors in the environment and constitutes only the information source that is used in the Suprise Detection System for predictions. The Surprise System is a module processing (abstracting) the perception and extracting the necessary information about the actions that would require an update of the information in the system. Since we are not interested in pure immitation of human actions in GRASP, we needed to define what type of information do we need to derive from human actions. It is clear that we were not interested in repeating the strongly varying trajectories of human actions. We decided that the Surprise System should monitor information that is useful for object-centric information for, e.g., the grasp planner to constraint motion of the manipulated object in respect to orientation and acceleration constraints. This already presented Object Container saves object related information about

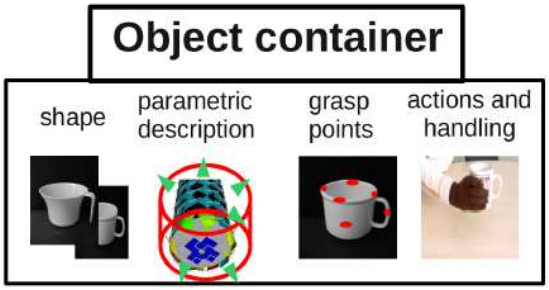


Figure 2.1: Object Container storing physical and geometric properties of the object.

We store here all information about the underlying geometry, the way how the system can grasp the object (which may have been derived from the human grasp or stored from previous interactions with the object).

Information about typical location areas and transport relations between them here we are interested in observed locations where transport actions of the object originated or ended. The actions of the human or other manipulation agents are used to fill this information source. Here a significant abstraction of the action is required, Since we are not interested in a pure immitation but are interested in the information about where to find or place down a given object while manipulating it and how the trajectory looked like during this phase. The second is important to reason about possible task constraints while performing a manipulation. An example here can be a transport motion to put away an object (arbitrary) or to place it on a very specific place (hand-over or placing on a shelf). An area where extraction of these relations is very important and where we also migrate our current system is manipulation in medical domain (minimally invasive surgery), where the surgeon may perform just an approach phase to an operation area or where we observed a highly confined motion because a specific task (like suturing) is performed.

The system needs to know here for its future actions, how constraint a specific motion is, that means how free is it to deviate from a given trajectory to optimize its transport planning.

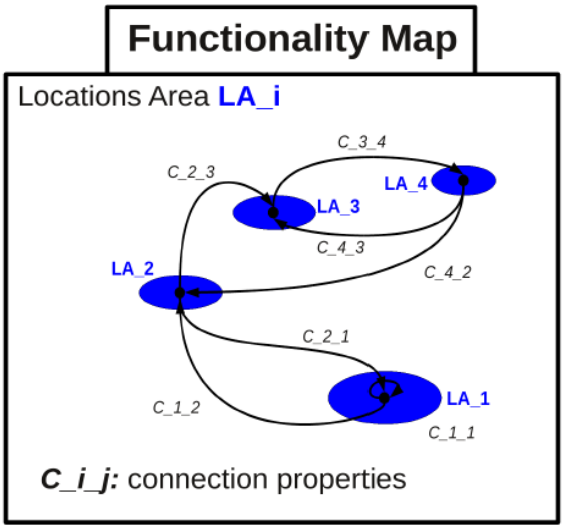


Figure 2.2: Functionality Map storing the functional relations of the transporation tasks between different areas in the world representation.

We store this information in the Functionaity Map, which stores the geometric positions of the places where an object was placed and the connection properties. The connection properties represent in our system: variation of the trajectory observed over several transport actions and type of grasp.

This information needs definitely to be refined and may be a good example of how and when to generate a surprise trigger. The system may have observed a trajectory of an object in a previous human action. This resulted in a new location (Location Area) for the object and new transport relation (action) in the environment. This could not be predicted from previous observation (first occurence) and it resulted in a surprise trigger forcing the system to update the object container (how it was moved) and the functionality map (what was the connection property). The system reasoned from the shape of the trajectory (motion profile) about the initial variation in the trajectory. Later, if the human performs a similar action, the system compares it with the stored information and if it can be explained by the knowledge stored in the system, no update of the information is required. If the human starts suddenly titling the object and a previous observation observed a constraint on the orientation of the object (in object container) then a surprise trigger is generated and the information in the Object Container will be adjusted. It appears that some physical property of the object changed resulting in a new handling modality. We see that this type of surprise is a hint for a possible change in the physical property of the object (e.g. full before now empty). Another possibility is that an object suddenly was placed at a new location, for example a fork moved around on the table before, suddenly was put in the dishwasher. This means that a new location area for an object (the dishwasher) and a new transport relation (table dishwasher) was observed. This results in a suprise trigger that updates the functionality map this time. We see that a mismatch generated in the functionality map relates to a possible change in the function of the object. Something which was originally a tool, now became a dirty object to stay with the dishwasher example.

An essential part of the Surprise Detection System is the fast processing of perception and an abstraction and reasoning from the observed information. The knowledge database is just the container which is important but just a small part of the entire system. This part of suprise detection dealt with the passive observation of human actions by the system to get an initial idea about the world. In this context, forces and physical properties of objects cannot be observed (no sensor on the human). We see that the system can reason about changes in the physical property (mismatch with Object Container information) and changes in the function of the object (mismatch between observation and the content of the Functionality Map. During each interaction of human with any object the system is monitoring the motion and comparing it with its predictions. If it matches the stored information then no surprise event is generated that means no update of knowledge is required. To address the surprise resulting from

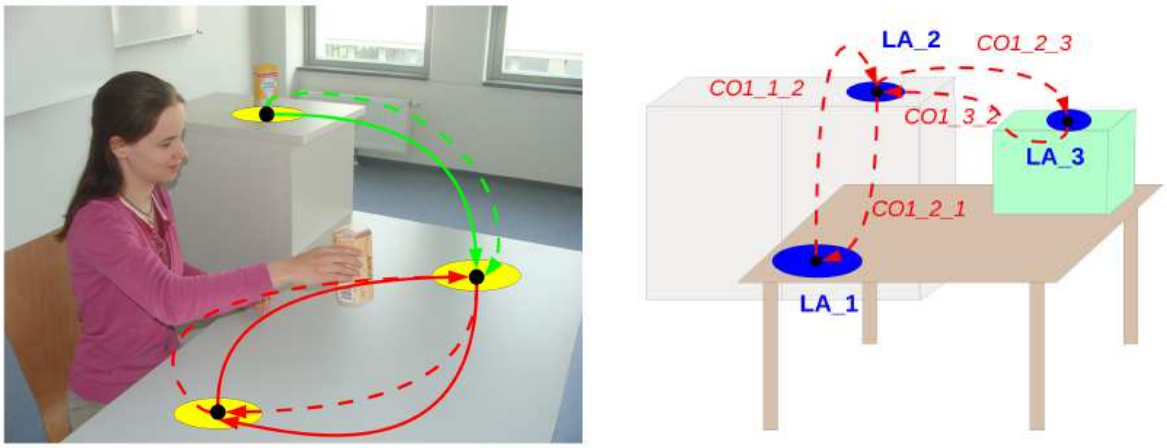


Figure 2.3: System observing human actions and creating physical (Object Container) and functional (Functionality Map) representations of knowledge.

system's own interaction with the object (experience gained in an exploration phase), we added also an additional component to the system where the true physical properties of an object (which could not be estimated from passive information) are estimated. Important examples in the object container is mass center, mass distribution of the object. We cannot guess the mass from pure observation. An active exploration is required. For this, the system initializes the Object Container with the typical assumptions about being solid, uniform object with the center of mass in the center of the object. This is the initial prediction of the system that is reason from pure observation. The system is interacting with an object (pushing) and compares the expected (predict) behavior with the actual motion of the object due to its action (act). This is our implementation of the predict-act-perceive loop. In this case, the system predicts the result of the action based on its generic assumption reasoned from passive observation. It acts on the object and observes the possible mismatches between the prediction and observation. In case of a mismatch, a suprise event is generated to update the physical properties of the object in the object container. Our current system can estimate the mass center of mass and the layered mass distribution (different density layer inside of the object) from observation of the motion induced by poking the object. This adds valuable information about the correct physical properties of the object. This knowledge helps to prevent unstable grasps resulting from, e.g., wrong estimations of the center of mass. We are also able to reason about the interior of a closed object from its responses to the external activation (see references of WP5 for details).

2.1 Parsing of Human Actions for Surprise Detection

A system for vision-based estimation of manipulation-relevant properties of objects is developed for natural scenes based on observation of human actions [5]. The system consists of an a-priori (*Atlas*) knowledge about known generic objects in the scene and classifies the scene into mission relevant objects and background geometry that is important only for collision avoidance. The system has an object-centric structure and consists of an Atlas representation and a *Working Memory*. The Working Memory stores the current knowledge about the scene, the manipulated objects and actions applied to them in the local environment. The handling properties of objects may change over time. Such a modification is triggered by a mismatch between the expectation and the observation of the human action.

We focused further on the abstract representations of the manipulation-relevant properties in a given environment [5]. The requirements on this representation have to be specified and the relevant knowledge has to be extracted from the observation in an appropriate manner. A very important aspect is, that not only the object itself (e.g., its properties or physical states) is defining the way, how it is manipulated, but also the location at which the manipulation is performed. Certain actions takes usually place at specific locations, which have certain properties. Hence, we need not only a collection of object properties, but also a map, which links locations in the environment to the specific way how objects are handled at these locations (see Fig. 2.3). It is important to notice, that we are not interested in the exact registration

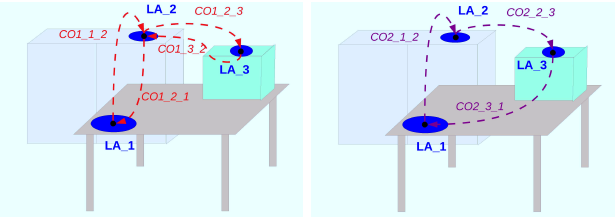


Figure 2.4: Functionality Map of the environment for two exemplary objects.

of the actions to the environment in the sense of navigation, but in an abstract representation of the functionalities in the environment. We are not using any semantic information about the environment. Furthermore, the system does not rely on any linguistic information.

The representation of the knowledge about the human actions is split into an object-centric representation, reflecting the physical properties of an object stored in an *Object Container*, and a *Functionality Map*, representing possible actions related to the environment (Fig. 2.4). While the Object Container is linked only to the object, the Functionality Map is anchored to the geometric model of the environment. This framework allows us to limit unexpected events (*surprise events*), that cannot be explained with the current knowledge, to situations, where the physical state or the function of an object changed. The system is insensitive to variations in the execution of the same action. Predictions about the current situations are based on the information stored in Object Container or the Functionality Map. Therefore, mismatches between these predictions and observations occur just at an abstract level. They signal the *right moment* to update the stored information. To built up such a system, already a small number of observations of a human performing the task is enough. Our representation contains information about the user's intention rather than a simple recording for the imitation of a trajectory.

We develop the concept of the Functionality Map and the Location Areas in [5] further. We make use of the Location Areas as representations of positions, where manipulations take typically place. The performance of different tasks requires special *dynamic properties* of a manipulator system, which depend on the *task-specific* operation areas (Location Areas) in the environment. Moreover, *efficient* dynamic properties are desirable. Each joint should just need to perform an angular speed, which has a limited magnitude and a smooth curve over time. Energy can be saved and it is possible to take more care of the hardware.

The robot should be designed in such a manner, that it is able to perform a desired task. This means first, that the robot has to reach certain positions. Second, it should be able to perform the desired manipulations there. This requires certain dynamic capabilities at these positions: The robot's end-effector should be able to move into one or more certain directions at certain speeds. The range of directions and speeds depends on the task. The task itself is usually done within a certain area in the workspace. There can be areas, where just simple tasks are performed. These tasks can even be performed by a robot with limited capabilities there. In contrast, complex manipulations can take place at other areas. Consequently, the task determines not only the areas, where manipulations are performed, but also the capabilities, which are required from the robot. A good representation of such efficient, task-specific dynamic properties is essential to reduce the dimensionality. We introduce the *maneuverability volume* as an appropriate representation. A exemplary scene is shown in Fig. 2.5.

The Functionality Map and the Location Areas have the advantage, that the manipulation properties can be extracted through the efficient observation of a human. This is user-friendly and also less time-consuming than a manual determination of a task. This is already an efficient reduction of the high number of dimensions in the observations to an abstract representation. Now, the capabilities, which are required by this abstract task, need to be integrated into the robot's design. This requires, in turn, a good representation to reduce the dimensions further. At the end, the robot's design parameters should be determined efficiently.

It is important to point out, that we are not interested in the design of a robot, which is a copy of, e.g., a human's arm, or a robot, which simply imitates the observed manipulations. Hence, the manipulated objects are in the focus. The objects and their manipulation properties are determining the manipulation. E.g., their functionality puts constraints on the manipulation. This object-centric point of view enables a design of the robot, which is independent of the human's anatomy or behavior. Just the desired manipulation capabilities themselves are important.

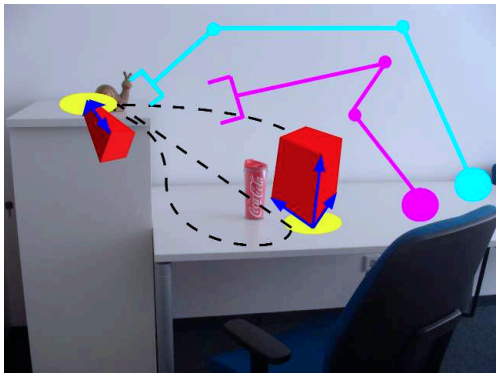


Figure 2.5: Design of a task-specific, efficient robot: Which design parameters does the robot need to achieve the desired high maneuverability volume (red) at the Location Area (yellow) on the table and a smaller one on the cupboard? Where has the robot's base to be positioned to achieve a good performance? Is, e.g., the manipulator in magenta more efficient than the cyan one?

2.2 Visual estimation of center of mass and mass distribution of objects with previously unknown internal structure

We developed a novel vision based approach for estimating physical properties of an object such as its center of mass and mass distribution. Passive observation only allows to approximate the center of mass with the centroid of the object. This special case is only true for objects that consist of one material and have unified mass distribution. However, this does not apply for most of the objects. Particularly for heavy objects, the assumption can cause undesired torques which will result in an unstable grasp.

Figure 2.6: Active striking of an object to perform system identification: mass center and mass distribution from observation

To our knowledge, the current state of the art methods for vision based graspless active estimation of physical properties of the object do not address the issue of estimating the mass distribution of the object, and they either approximate the center of mass by the centroid of the object or do not address it at all. Thus we introduce an active interaction technique with the object derived from the analogon to system identification with impulse functions. We treat the object as a black box and estimate its internal structure by analyzing the response of the object to external impulses. We use a bottom-up technique, where we make an initial assumption about the center of mass based on the external 3D geometry of the object. This is used to compute possible points of interaction between the robot and the object. The impulses are realized by striking the object at points of interaction, and the forces of the strikes are computed based on the mass of the object. We determine the center of mass from the profile of the observed angular motion of the object that is captured by a high frame-rate camera. We use the motion profiles from multiple strikes the mass of the object and its 3D geometry to compute the mass distribution. Knowledge of these properties of the object leads to more energy efficient and stable object manipulation.

For qualitative assessment some of the results from the experiments are presented in the figure below. The first column of the illustration demonstrates the results for mass distribution For the spray bottle, the region represented by yellow dots is estimated to have weight of 29.3 g, green dots 22.6g, and blue dots 5.1g. Note that here the bottle is empty and the only thing that the blue region contains withing itself is the tube for pumping out the liquid. For juice bottle, the yellow region has an approximate wight of 29.8g, the blue region representing the container part has weight of 7g, and the green region that represents the cap has a weight of 18.2g. Here the container is also empty. For the empty salt cylinder,

the estimated weights are 31g, 2.3g, 0.1g for the yellow, blue and green regions correspondingly. The second column of the illustration demonstrates the results for the center of mass estimation. Here the green dost represent the true location of the center of mass and the red dots represent the estimated location (Fig. 2.7.

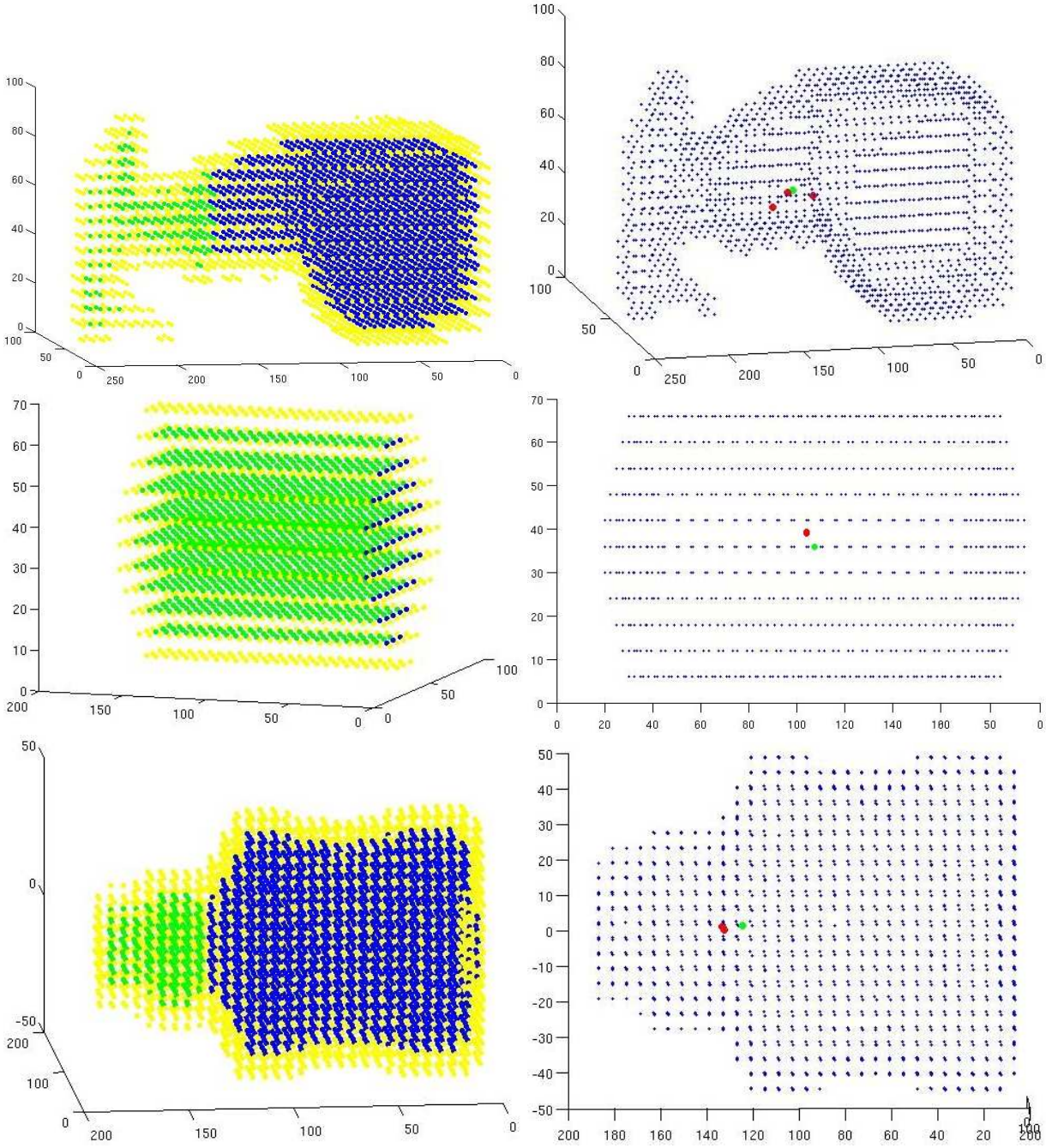
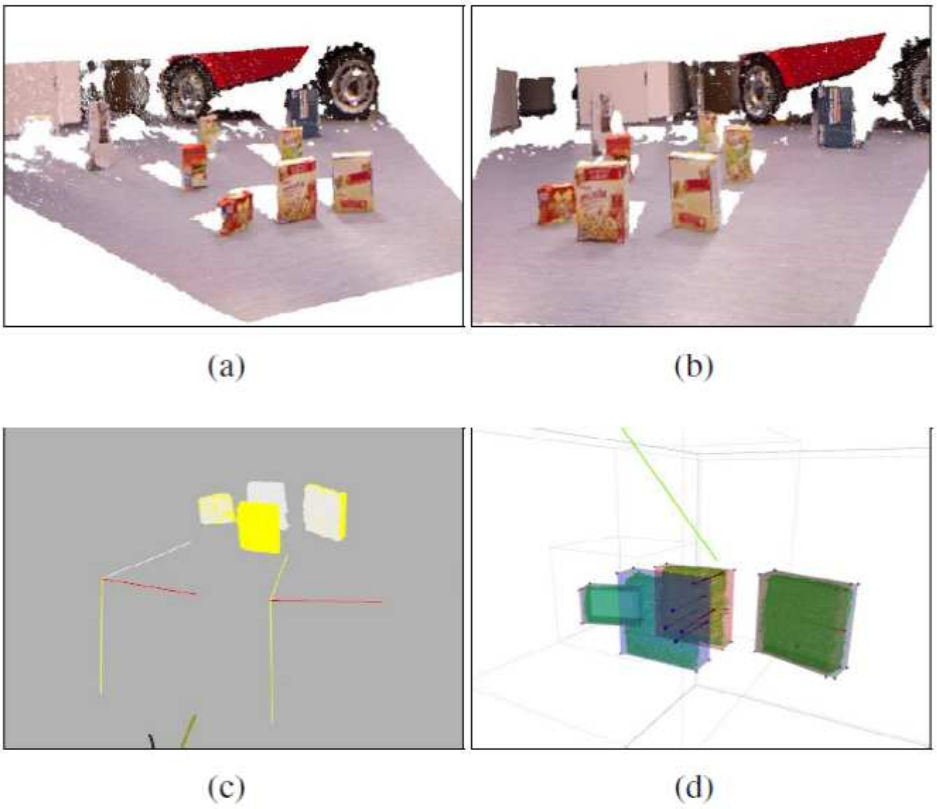


Figure 2.7: Results from the mass distribution evaluation from active exploration in the learning phase.

2.3 Visual Modeling of Independent Motion Parameters in Dynamic Scenes

Fusion of information from moving object in a scene requires robust matching of points between the images or captured data. In domestic environments it is important not only to capture the static geometry of the scene but it becomes also important to understand the motion parameters of moving objects in the scene for correct prediction of expectations. This is required, for example, for robot interaction or for motion planning involving collisions with near objects and for capturing not only the current pose of the objects but also their functions and current states.



In each loop iteration a tentative model is built from the preceived data and two types of elements are derived: those that support or fit such tentative model are called inliers and the remaining elements or outliers. The model with the largest amount of inliers is the closest approximation to the searched model and can be considered the output of the method. In most of the cases, however, the set of inliers are used again to re-compute the model using an optimization algorithm, whereas the outliers, like in any robust estimation, are commonly ruled out. In this approach we show that outliers can also contain information about the evolution of the state of a dynamic scene. Having a set of matched features, either in 2- or 3D, of a scene observed from two different poses at different times, we profit from the fact that not all the information classified as outlier is derived from noisy or mismatched data and that this information gives, in turn, patterns that can be considered as input sets of data indicating probable independent events inside the same scene. In order to detect these good outliers in this work we make use of a dual layered framework for 3D mapping that stores the mapped elements as 3D blobs representing tentative object candidates; however, the detection of these elements can be performed in other different ways. The advantage of using this framework is twofold:) in this work, the geometric layer of the framework helps to relate spatially the mapped elements with the outlier position information,) for future works, once a mapped element was detected it was moved, additional properties, like grasping points or labels like movable, can be assigned to that element and stored in the abstract layer of the framework.

Fig. 2.8 shows the matched features of poses in time k and (k + 1), both sets are displayed in the second pose image. This allows to estimate the egomotion of the system.

Outliers that are not conform with the global ego-motion estimation of the system are used to estimate

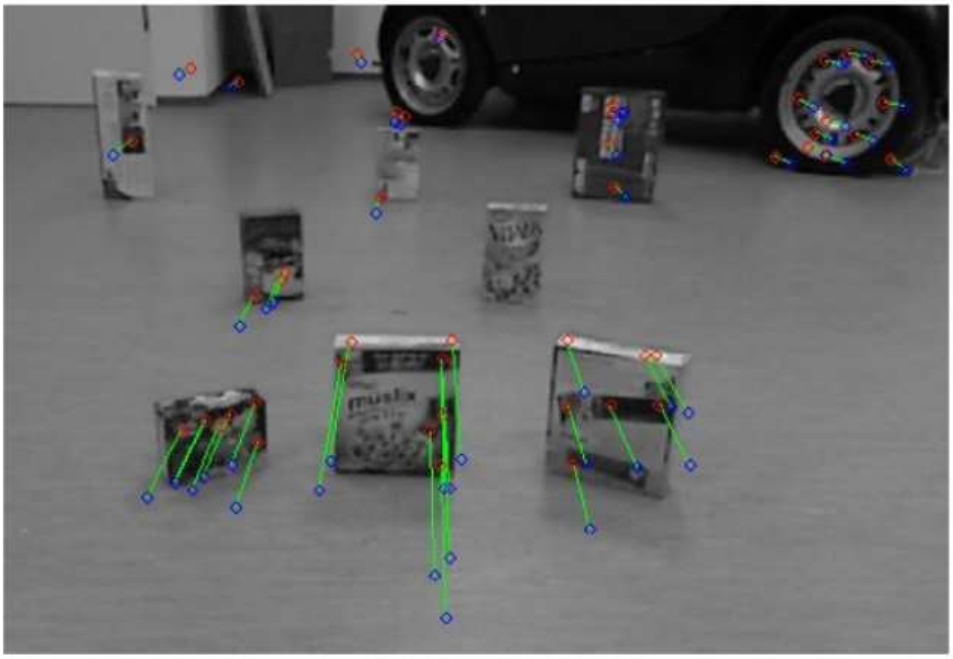


Figure 2.8: Flow of valid feature matches C2D = (I1; I2). Blue circles indicate the features of the first set and the red circles indicate the matched features of the second scene.

the indipendent motion parameters of the objects. This information is stored in the Object Container extending the available action description also to motion parameters (Fig. 2.9).

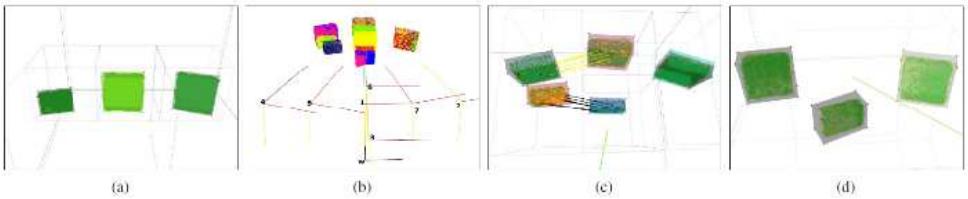


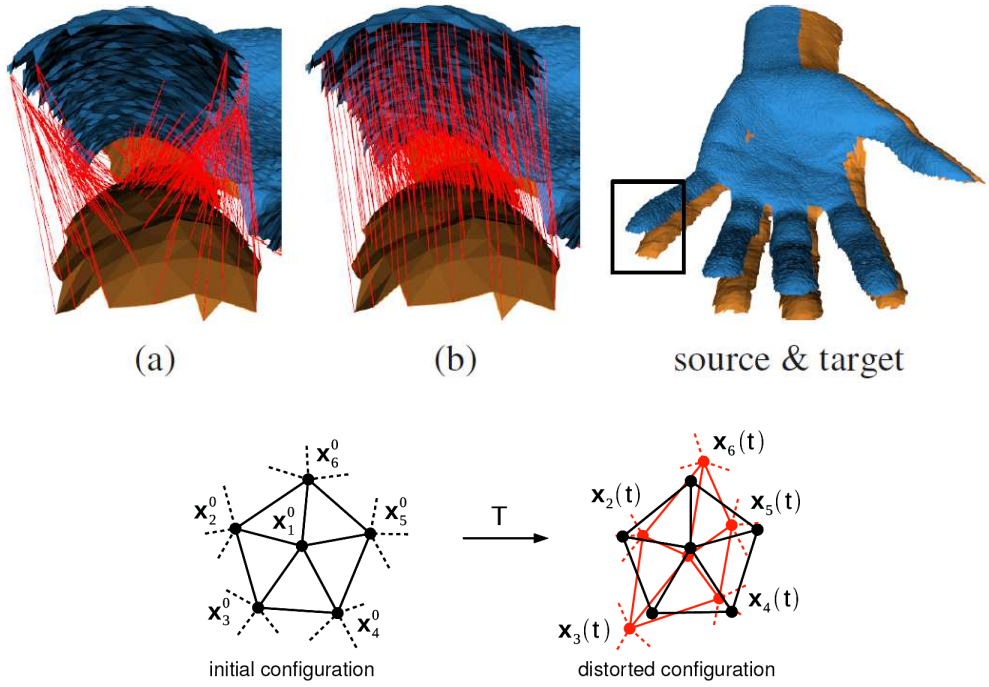
Figure 2.9: Ego-motion Estimation observing a Dynamic Scene. a) 3D blob map at the first registration is displayed, b) shows the pose frames that were reached by the robot and the sequence of static registrations enumerated by color:1-green, 2-yellow, 3-orange, 4-blue, 5-red, 6-light blue 7-pink, the blob at the right was not moved, hence, all the registrations (colors) are overlapped on it; c) motion detection of two objects between poses 4-blue and 5-red are detected in the map, the scene is observed from above, d) shows the updated map at the end of the sequence.

2.4 Deformable 3D Shape Registration

We continued the work on implementation of the Surprise Event Hierarchy (Fig. 1.1) that is now used for efficient detection of points in time when an observed change in the way how an object is handled lets the system assume that the physical property of the object (e.g. its filling state) or the object function changed [5]. New in this period was the addition of Task 5.4 and Task 5.5. The generalization of object form descriptions leads to further generalization of the information stored in the Atlas information and allows to generalize between different instantiations of the same category of objects [2,3]. Fig. 1.2 shows a registration of a complex shape for a better visualization of the system performance. This processing allows to keep one representative of a shape in the Atlas and allows beside the already presented indexing function to the entries in the Atlas additionally a deformation map that can be used for grasp adaptation. The adaptation can use the grasp planning for the reference object to infer a good way to handle the deformed candidate.

Deformable (non-rigid) shape registration is a fundamental problem in computational geometry with applications in the fields of computer vision, computer graphics, medical image processing and many others. The problem consists of finding a reasonable deformation which aligns two given 3D shapes. The rationale behind this work in the context of GRASP is to use such a method in order to fit a given 3D object model to a scene which contains an object similar, however not identical, to the given model. Based on the deformably fitted model, grasp simulation and prediction can be performed in a more precise and reliable manner.

We developed a new method for solving the deformable 3D shape registration problem. The algorithm computes shape transitions based on local similarity transforms which allows to model not only as-rigid-as-possible deformations but also local and global scale. We formulate an ordinary differential equation (ODE) which describes the transition of a source shape towards a target shape. We assume that both shapes are roughly pre-aligned. The ODE consists of two terms. The first one causes the deformation by pulling the source shape points towards corresponding points on the target shape. Initial correspondences are estimated by closest-point search and then refined by an efficient smoothing scheme. The figure below compares the correspondence estimation based on closest-point search only (a) and the one based on our smoothing scheme (b). The improvement is obvious (Fig. 2.10).



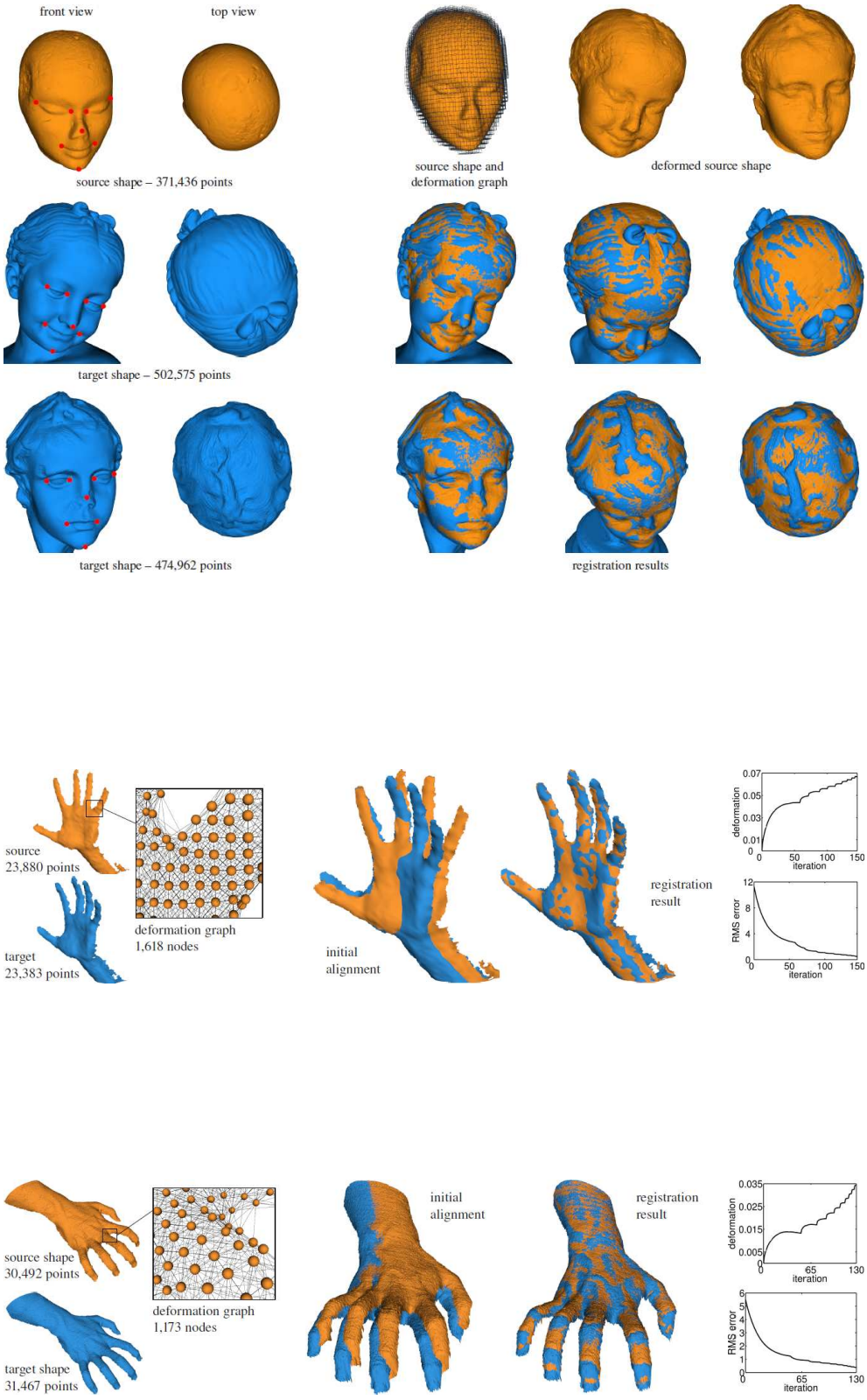


Figure 2.10: Results of newly developed registration technique applied to challenging problems in the registration. We chosen these examples to better visualize the performance of the system.

References

[1] Chavdar Papazov, Sami Haddadin, Sven Parusel, Kai Krieger, and Darius Burschka. Rigid 3D Geometry Matching for Grasping of Known Objects in Cluttered Scenes. International Journal of Robotics Research, 2012. (accepted).

[2] Chavdar Papazov and Darius Burschka. Stochastic Global Optimization for Robust Point Set Registration. Computer Vision and Image Understanding, 115, December 2011

[3] Chavdar Papazov and Darius Burschka. Deformable 3D Shape Registration Based on Local Similarity Transforms. Computer Graphics Forum, 30, 2011. (special issue SGP'11)

[4] Juan Carlos Ramirez de la Cruz, Darius Burschka. Framework for Consistent Maintenance of Geometric Data and Abstract Task-Knowledge from Range Observations. IEEE ROBIO 2011

[5] Susanne Petsch and Darius Burschka. Representation of manipulation-relevant object properties and actions for surprise-driven exploration. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1221-1227, San Francisco, California, USA, September 2011

[6] Artashes Mkhitaryan and Darius Burschka. Visual Estimation of Object Density Distribution through Observation of its Impulse Response. (submitted to RSS 2012)

Rigid 3D Geometry Matching for Grasping of Known Objects in Cluttered Scenes

Chavdar Papazov*, Sami Haddadin†, Sven Parusel†, Kai Krieger†, Darius Burschka*

*Robotics and Embedded Systems
Technische Universität München (TUM)
Boltzmannstr. 3, D-85748 Garching, Germany
{papazov, burschka}@in.tum.de

†Institute of Robotics and Mechatronics
German Aerospace Center (DLR)
P.O. Box 1116, D-82230 Wessling, Germany
{sami.haddadin, sven.parusel, kai.krieger}@dlr.de

Abstract—In this paper, we present an efficient 3D object recognition and pose estimation approach for grasping procedures in cluttered and occluded environments. In contrast to common appearance-based approaches, we rely solely on 3D geometry information. Our method is based on a robust geometric descriptor, a hashing technique and an efficient, localized RANSAC-like sampling strategy. We assume that each object is represented by a model consisting of a set of points with corresponding surface normals. Our method simultaneously recognizes multiple model instances and estimates their pose in the scene. A variety of tests shows that the proposed method performs well on noisy, cluttered and unsegmented range scans in which only small parts of the objects are visible. The main procedure of the algorithm has a linear time complexity resulting in a high recognition speed which allows a direct integration of the method into a continuous manipulation task. The experimental validation with a 7-degrees-of-freedom Cartesian impedance controlled robot shows how the method can be used for grasping objects from a complex random stack. This application demonstrates how the integration of computer vision and soft-robotics leads to a robotic system capable of acting in unstructured and occluded environments.

Fig. 1. A robot operating in a household environment.

1 Introduction

Robot manipulation tasks in non-industrial environments cannot rely on hard-coded knowledge about the scene structure. Since especially human actions modify the environment in a way which cannot be foreseen, a vision-based object recognition and localization system is very useful for providing the necessary updates of the scene knowledge. In recent years, advances in 3D geometry acquisition technology have led to a growing interest in object recognition and pose estimation techniques which operate on three-dimensional data. Furthermore, the knowledge of the 3D geometric shape and the pose of an object greatly facilitates the execution of a stable grasp. The 2D appearance of an object may not provide reliable information about its pose in space because surface texture elements may be misaligned (as it often happens to labels of household objects). Furthermore, 2D techniques have to deal with changes in viewpoint and illumination.

The 3D object recognition and pose estimation problem can loosely be defined as follows. Given a set of object models and a scene, the task is to identify the objects present in the scene and to estimate their position and orientation. The output of an object recognition and pose estimation algorithm is a list of recognized model instances each one with a corresponding transform which aligns the model instance to the scene. For the sake of simplicity, in the rest of the text, we mean by "object recognition" *both* object identification and pose estimation. Furthermore, we

discuss a special instance of the problem, given by the following assumptions.

1) Each model is a finite set of points with corresponding surface normals.
2) Each model represents a non-transparent object.
3) The scene is a range image.
4) Each transform which aligns a recognized model instance to the scene is a proper rigid transform.

Even under these assumptions the problem still remains challenging for several reasons: it is a priori not known which objects are present in the scene; usually, there are scene parts not belonging to any of the objects of interest, i.e., there is background clutter; the input is typically corrupted by noise and outliers; the objects are only partially visible due to occlusions and scan device limitations.

1.1 Contributions and Overview

This paper demonstrates how our original 3D object recognition approach presented in [Papazov and Burschka, 2010] can be used to support a manipulation task. We introduce a vision-based framework that allows a robotic manipulator to grasp objects in unstructured, dynamically changing environments.

Our object recognition approach operates directly on unsegmented point clouds provided by a range scanner. This does not require scene segmentation which may be quite time consuming. More specifically, we make the following contributions. (i) A new efficient, localized RANSAC-like sampling strategy is introduced. (ii) We use a hash table for rapid retrieval of pairs of oriented model points which are similar to a sampled pair of oriented scene points. This allows to efficiently generate object and pose hypotheses. (iii) We provide a complexity analysis of our sampling strategy and derive a formula for the number of iterations required to recognize the objects with a predefined success probability.

The proposed accelerations in our vision processing allow a seamless integration into a grasping framework, where the recognition interleaves with the actual manipulation task without causing noticeable delays in the overall process. The method shows its potential in a complex experimental use-case. We employ the DLR Lightweight Robot III (LWR-III) [Albu-Schäffer et al., 2007], which is equipped with a Cartesian impedance control method and is able to react to environment disturbances and to process faults caused by unexpected contact forces in real-time. Using the proposed 3D object recognition method, impedance control with reactive recovery strategies, and a simple grasp planner the robot quickly and robustly grasps objects from unsorted and cluttered piles. Furthermore, in case of failures, it reacts accordingly and continues the process if possible[1].

1. These experiments are enclosed as multimedia extensions (see Extensions 1 and 2 in Appendix A).

The rest of the paper is organized as follows. Previous work is reviewed in Section 2. In Section 3, we establish some notation and explain in more details two algorithms important to our work. Our 3D object recognition approach is introduced in Section 4. In Section 5, the robot and its overall control concept for robust and sensitive grasping is described. Section 6 presents experimental results. Conclusions and possible future topics of research are drawn in the final Section 7 of the paper.

2 Related Work

Object recognition is closely related to object classification/shape retrieval. However, the latter methods only compute the similarity between shapes and not an aligning transform. Furthermore, it is assumed that the input represents a single shape whereas we handle range images containing multiple objects and background clutter.

The so-called voting approaches represent one class of object recognition methods. Well-known representatives are the generalized Hough transform [Ballard, 1981], pose clustering [Stockman, 1987], geometric hashing [Lamdan and Wolfson, 1988] and tensor matching [Mian et al., 2006]. Although these methods perform well on complex scenes they tend to be costly and their integration into a real-world grasping system will lead to long delays in the overall processing loop.

Another way to tackle the problem is to model an object as an assembly of basic shapes (primitives) and to recover these shapes and their spatial relationships from an input scene. Many types of primitives can be used within this part-based framework: generalized cylinders [Binford, 1971], superquadrics [Barr, 1981], implicit polynomials [Keren et al., 1994], geometric primitives [Taylor and Kleeman, 2003], and parametric shapes [Schnabel et al., 2007]. Methods for efficient recovering of superquadrics from range data were introduced in [Solina and Bajcsy, 1990], [Dickinson et al., 1997] and [Biegelbauer et al., 2010]. However, despite their efficiency, the part-based approaches are limited to a certain shape class, namely, the one which can be described by the chosen set of primitives.

The feature-based methods belong to a further class of object recognition approaches. In a first step, point-wise correspondences between the models and the scene are computed, usually using local geometric descriptors. Next, the aligning transform is calculated based on the established correspondences. A list of geometric descriptors includes, without being nearly exhaustive, spin images [Johnson and Hebert, 1999], local feature histograms [Hetzel et al., 2001], 3D/harmonic shape context [Frome et al., 2004], intrinsic isometry invariant descriptors [Sun et al., 2009] and manifold harmonic bases [Wu et al., 2010]. All feature-based algorithms rely on the assumption that the objects to

be recognized have distinctive feature points, i.e., points with rare descriptors. However, especially for simple shapes, this assumption does not always hold and the methods degenerate to brute force search [Aiger et al., 2008].

In [Grewe and Kak, 1995], a hashing technique similar to ours was proposed. It is a learning-based method employing a multiple-attribute hash table for efficient 3D object recognition. On the positive side, attribute uncertainties are taken into account and the number of attributes as well as the size of the hash table bins are calculated automatically. However, the system cannot handle free-form objects and in the presented experimental results only objects composed of single-colored surfaces are used. Furthermore, the method relies on a segmentation to identify the planar or cylindrical surface patches the objects are made of.

A further hashing technique was proposed in [Matei et al., 2006]. Based on a hash table, a fast indexing into a collection of geometry descriptors of *single* model points is performed. In contrast, our hash table stores descriptors of *pairs* of oriented model points (called doublets). This allows to efficiently query the model doublets similar to a sampled scene doublet and it makes it very easy to compute the aligning rigid transform since it is uniquely defined by two corresponding doublets. Moreover, in [Matei et al., 2006], multiple range images are aligned to each other in order to build a more complete scene representation and a foreground/background segmentation is executed. In contrast, our method operates on a single range image and does not require segmentation. Furthermore, the test scenes used in [Matei et al., 2006] contain a single object and some background clutter.

3 Notation and Basic Algorithms

An oriented point $\mathbf{u} = (\mathbf{p}_u, \mathbf{n}_u)$ consists of a point $\mathbf{p}_u \in \mathbb{R}^3$ and a corresponding surface normal $\mathbf{n}_u \in \mathbb{R}^3$, $\|\mathbf{n}_u\| = 1$. Accordingly, an oriented point pair $(\mathbf{u}, \mathbf{v})$ is a pair of two oriented points $\mathbf{u} = (\mathbf{p}_u, \mathbf{n}_u)$ and $\mathbf{v} = (\mathbf{p}_v, \mathbf{n}_v)$.

3.1 Fast Surface Registration

In short, rigid surface registration consists of computing a rigid transform which aligns two surfaces. Assume $\mathbf{S}$ is a surface represented by a set of oriented points. According to [Winkelbach et al., 2006], for a pair of oriented points $(\mathbf{u}, \mathbf{v}) = ((\mathbf{p}_u, \mathbf{n}_u), (\mathbf{p}_v, \mathbf{n}_v)) \in \mathbf{S} \times \mathbf{S}$, a descriptor $f : \mathbf{S} \times \mathbf{S} \to \mathbb{R}^4$ is computed as

$$f(\mathbf{u}, \mathbf{v}) = \begin{pmatrix} f_1(\mathbf{u}, \mathbf{v}) \\ f_2(\mathbf{u}, \mathbf{v}) \\ f_3(\mathbf{u}, \mathbf{v}) \\ f_4(\mathbf{u}, \mathbf{v}) \end{pmatrix} = \begin{pmatrix} \|\mathbf{p}_u - \mathbf{p}_v\| \\ \angle(\mathbf{n}_u,\, \mathbf{n}_v) \\ \angle(\mathbf{n}_u,\, \mathbf{p}_v - \mathbf{p}_u) \\ \angle(\mathbf{n}_v,\, \mathbf{p}_u - \mathbf{p}_v) \end{pmatrix}, \tag{1}$$

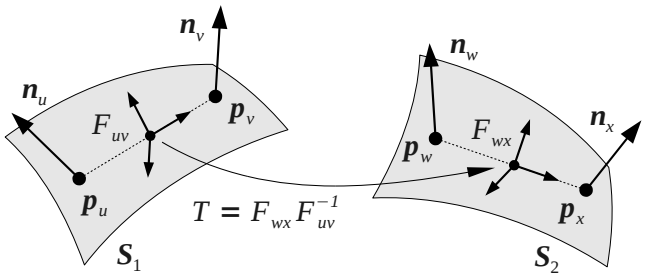


Fig. 2. Computing the rigid transform T which aligns $\mathbf{S}_1$ to $\mathbf{S}_2$ based on the local coordinate systems F_{uv} and F_{wx} of the oriented point pairs $(\mathbf{u}, \mathbf{v})$ and $(\mathbf{w}, \mathbf{x})$, respectively. See text for details on F_{uv} and F_{wx}.

where $\angle(\mathbf{a}, \mathbf{b})$ is the angle between the vectors $\mathbf{a}$ and $\mathbf{b}$. In order to register two surfaces $\mathbf{S}_1$ and $\mathbf{S}_2$, each one represented by a set of oriented points, the method proceeds as follows. It samples uniformly oriented point pairs $(\mathbf{u}, \mathbf{v}) \in \mathbf{S}_1 \times \mathbf{S}_1$ and $(\mathbf{w}, \mathbf{x}) \in \mathbf{S}_2 \times \mathbf{S}_2$ and computes and stores their descriptors $f(\mathbf{u}, \mathbf{v})$ and $f(\mathbf{w}, \mathbf{x})$ in a four-dimensional hash table. This process continues until a collision occurs, i.e., until $f(\mathbf{u}, \mathbf{v})$ and $f(\mathbf{w}, \mathbf{x})$ end up in the same hash table cell. Computing the rigid transform T which aligns $(\mathbf{u}, \mathbf{v})$ to $(\mathbf{w}, \mathbf{x})$ gives a transform hypothesis which registers $\mathbf{S}_1$ to $\mathbf{S}_2$. Fig. 2 illustrates the alignment. More formally,

$$T = F_{wx} F_{uv}^{-1} \tag{2}$$

is computed based on the pairs' local coordinate systems, each one represented by a 4×4 matrix (homogeneous coordinates) F_{uv} respectively F_{wx}. We have

$$F_{uv} = \begin{pmatrix} \frac{\mathbf{p}_{uv} \times \mathbf{n}_{uv}}{\|\mathbf{p}_{uv} \times \mathbf{n}_{uv}\|} & \frac{\mathbf{p}_{uv}}{\|\mathbf{p}_{uv}\|} & \frac{\mathbf{p}_{uv} \times \mathbf{n}_{uv} \times \mathbf{p}_{uv}}{\|\mathbf{p}_{uv} \times \mathbf{n}_{uv} \times \mathbf{p}_{uv}\|} & \frac{\mathbf{p}_u + \mathbf{p}_v}{2} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3}$$

where $\mathbf{p}_{uv} = \mathbf{p}_v - \mathbf{p}_u$ and $\mathbf{n}_{uv} = \mathbf{n}_u + \mathbf{n}_v$. F_{wx} is defined analogously by replacing the indices u and v in (3) with w and x, respectively. The transform hypothesis T generated in this way is evaluated by transforming the points of $\mathbf{S}_1$, i.e., $\mathbf{p}_i' = T\mathbf{p}_i$, $\forall \mathbf{p}_i \in \mathbf{S}_1$ and counting those $\mathbf{p}_i'$ which fall within a certain ϵ-band of $\mathbf{S}_2$.

According to [Winkelbach et al., 2006], this process of generating and evaluating hypotheses is repeated until either of the following stopping criteria is met: (i) a hypothesis is good enough, (ii) a predefined time limit is reached or (iii) all combinations are tested. Unfortunately, non of these criteria is well-grounded: the first two are ad hoc and the third one is computationally infeasible. In contrast, we derive the number of iterations required to recognize model instances with a pre-defined success probability. Furthermore, we modify this technique in a way which allows for the *simultaneous* matching of all object models to the scene.

3.2 RANSAC

RANSAC [Fischler and Bolles, 1981] is an iterative approach for model recognition. It uniformly draws minimal point sets from the scene and computes a transform which aligns the model with the minimal point set. A minimal point set is the smallest point set which uniquely determines a given type of transform. In the case of rigid transforms, it is a point triple. The score of the aligning transform is the number of transformed model points which lie within an ϵ-band of the scene. After a certain number of trials the model is considered to be recognized at the locations defined by the transforms which achieved a score higher than a predefined threshold.

The probability P_S of recognizing the model in N trials equals the complementary of N consecutive failures [Schnabel et al., 2007], i.e.,

$$P_S = 1 - (1 - P_M)^N, \tag{4}$$

where P_M is the probability of recognizing the model in a single iteration. Solving for N gives the number of trials needed to recognize the model in the scene:

$$N = \frac{\ln(1 - P_S)}{\ln(1 - P_M)}. \tag{5}$$

Note that since $P_S \approx 1$ and $P_M \approx 0$, one can safely assume that $N \geq 1$.

The RANSAC approach is conceptually simple, general and robust against outliers. Unfortunately, its direct application to the 3D object recognition problem is computationally expensive. In order to compute an aligning rigid transform, we need two corresponding point triples—one from the model and one from the scene. Assuming that the model is completely contained in the scene, the probability of drawing two such triples in a single trial is $P_M(n) = \frac{3!}{(n-2)(n-1)n}$, where n is the number of scene points. Since $P_M(n)$ is a small number we can approximate the denominator in (5) by its Taylor series $\ln(1 - P_M(n)) = -P_M(n) + O(P_M(n)^2)$ and obtain the number of trials as a function of the number of scene points:

$$N(n) \approx \frac{-\ln(1 - P_S)}{P_M(n)} = O(n^3). \tag{6}$$

In the next section of the paper, we will show that using oriented point pairs and a localized sampling strategy leads to a reduction of the time complexity from $O(n^3)$ to $O(n)$.

4 Rigid 3D Geometry Matching

Our object recognition method consists of two phases. The first one, the model preprocessing, is performed offline. It is executed only once and does not depend on the scenes in which the objects have to be recognized. The online recognition is the second phase. It is executed on the scene using the model representation computed in the offline phase. In the rest of this section, we describe both phases in detail and discuss the time complexity of our algorithm.

4.1 Model Preprocessing Phase

We assume that each object model is represented by a finite set $\mathbf{M} = \{\mathbf{u}_i = (\mathbf{p}_u, \mathbf{n}_u)_i\}_{i=1}^m$ of oriented points. For a given object model $\mathbf{M}$, we sample the pairs of oriented points $(\mathbf{u}, \mathbf{v}) = ((\mathbf{p}_u, \mathbf{n}_u), (\mathbf{p}_v, \mathbf{n}_v)) \in \mathbf{M} \times \mathbf{M}$ having $\mathbf{p}_u$ and $\mathbf{p}_v$ approximately d units apart from each other. For each such pair, the descriptor $f(\mathbf{u}, \mathbf{v}) = (f_2(\mathbf{u}, \mathbf{v}),\ f_3(\mathbf{u}, \mathbf{v}), f_4(\mathbf{u}, \mathbf{v}))$ is computed according to (1) and stored in a three-dimensional hash table. Note that f_1 is not part of the descriptor since a fixed distance d is used. In contrast to [Winkelbach et al., 2006], *not* all pairs of oriented points are considered, but only those with $\|\mathbf{p}_u - \mathbf{p}_v\| \in [d - \delta_d, d + \delta_d]$, for a given tolerance value δ_d. This has several advantages. It reduces the space complexity from $O(m^2)$ to $O(m)$, where m is the number of model points (this empirical measurement is further discussed in [Aiger et al., 2008]). Using a large d results in wide-pairs which allow a more stable computation of the aligning rigid transform than narrow-pairs do [Aiger et al., 2008]. Furthermore, a larger d leads to fewer pairs which means that computing and storing descriptors of wide-pairs results in less populated hash table cells. Thus, we will have to test fewer transform hypotheses in the online recognition phase and will save computation time.

However, the pair width d should not be too large since occlusions in real world scenes would prevent sampling a pair with points from the same object. For a typical value for d, there are still many pairs with similar descriptors which leads to hash table cells with too many entries. We avoid this overpopulation, by removing as many of the most populated cells to keep only a fraction K of the original number of pairs (in our implementation $K = 0.4$). This results in an information loss about the object shape which we take into account in the online phase of the algorithm.

In order to compute the final representation of all models $\mathbf{M}_1, \ldots, \mathbf{M}_q$, each $\mathbf{M}_i$ is processed in the way described above using the *same* hash table. In this way, a simultaneous recognition of all models is possible instead of sequentially matching each one of them to the scene. Furthermore, in order to keep track of which pair belongs to which model, every hash table cell stores the pairs in separate model-specific lists.

4.2 Online Recognition Phase

The input to the online recognition algorithm is a set $\mathcal{M} = \{\mathbf{M}_1, \ldots, \mathbf{M}_q\}$ of object models and a scene range image $\mathbf{S}$. The output is a list $\mathcal{T} = \{(\mathbf{M}_{k_1}, T_1), \ldots, (\mathbf{M}_{k_r}, T_r)\}$, where $\mathbf{M}_{k_j} \in \mathcal{M}$ is a recognized model instance and T_j is a proper rigid transform (an element of the special Euclidean group $SE(3)$) aligning $\mathbf{M}_{k_j}$ to the scene. Before turning to the details, it is advisable to read Algorithm 1 although some of the steps may not be completely clear at this point. In the rest of this section, the lines we are referring to are the lines of Algorithm 1.

input : a set $\mathcal{M} = \{\mathbf{M}_1, ..., \mathbf{M}_q\}$ of object models; a scene range image $\mathbf{S}$;
output: a list $\mathcal{T} = \{(\mathbf{M}_{k_1}, T_1), ..., (\mathbf{M}_{k_r}, T_r)\}$, with $\mathbf{M}_{k_j} \in \mathcal{M}$ and $T_j \in SE(3)$;

// 1) initialization
1 compute an octree for the scene $\mathbf{S}$ to produce a modified scene $\mathbf{S}^*$;
2 $\mathcal{T} \leftarrow \emptyset$; // an empty solution list
// 2) number of iterations
3 compute the number N of iterations;
4 **repeat** N *times*
// 3) sampling
5 sample a $\mathbf{p}_u$ uniformly from $\mathbf{S}^*$;
6 $\mathbf{L} = \{\mathbf{x} \in \mathbf{S}^* \ : \ \|\mathbf{x} - \mathbf{p}_u\| \in [d - \delta_d, d + \delta_d]\}$;
7 sample a $\mathbf{p}_v$ uniformly from $\mathbf{L}$;
// 4) normal estimation
8 estimate normals $\mathbf{n}_u$ at $\mathbf{p}_u$ and $\mathbf{n}_v$ at $\mathbf{p}_v$;
9 $(\mathbf{u}, \mathbf{v}) = ((\mathbf{p}_u, \mathbf{n}_u), (\mathbf{p}_v, \mathbf{n}_v))$;
// 5) hash table access
10 $f_{\mathbf{uv}} = (f_2(\mathbf{u}, \mathbf{v}), \ldots, f_4(\mathbf{u}, \mathbf{v}))$; // see (1)
11 access the model hash table cell at $f_{\mathbf{uv}}$ **and**
12 get its oriented model point pairs $(\mathbf{u}_j, \mathbf{v}_j)$;
// 6) generate and test
13 **foreach** $(\mathbf{u}_j, \mathbf{v}_j)$ **do**
14 get the model $\mathbf{M}$ of $(\mathbf{u}_j, \mathbf{v}_j)$;
15 compute the rigid transform T that aligns $(\mathbf{u}_j, \mathbf{v}_j)$ to $(\mathbf{u}, \mathbf{v})$; // see (2)
16 **if** μ *accepts* $(\mathbf{M}, T)$ **then**
17 $\mathcal{T} \leftarrow \mathcal{T} \cup (\mathbf{M}, T)$;
18 **end**
19 **end**
20 **end**
// 7) removing conflicting hypotheses
21 remove conflicting hypotheses from $\mathcal{T}$;

Algorithm 1: Online recognition phase.

Searching for closest points (line 8) and for points lying on a sphere around a given point (line 6) have to be performed very often in the online recognition phase. Thus, a fast execution of these operations is of great importance for the runtime of the algorithm. An efficient way to achieve this is to use an octree [de Berg et al., 2000].

Step 1) Initialization

In step 1 of the algorithm, an octree with a fixed leaf size L (the edge length of a leaf) is constructed for the input scene points. The full octree leaves (the ones containing at least one point) are voxels ordered in a regular axis-aligned 3D grid and have unique integer coordinates. Two full leaves are considered neighbors if their corresponding integer coordinates differ by not more than 1. Next, a down-sampled scene $\mathbf{S}^*$ is created by setting its points to be the centers of mass of the full octree leaves. The center of mass of a full leaf is the average of the points it contains. In this way, a one-to-one correspondence between the points in $\mathbf{S}^*$ and the full octree leaves is established. Two points in $\mathbf{S}^*$ are neighbors if the corresponding leaves are neighbors.

Step 2) Number Of Iterations

In this step, the number N of iterations is estimated such that all objects in the scene will be recognized with a certain user-defined probability. This will be explained in detail in Section 4.3.

Step 3) Sampling

As in classic RANSAC, we sample minimal sets from the scene. In our case, since we use normals, a minimal set consists of two oriented points. In contrast to RANSAC, they are *not* sampled uniformly and independently of each other. Only the first point, $\mathbf{p}_u$, is drawn uniformly from $\mathbf{S}^*$. The second one, $\mathbf{p}_v$, is drawn uniformly from the scene points in $\mathbf{S}^*$ which are approximately within a distance d from $\mathbf{p}_u$. To achieve this, we first retrieve the set $\mathbf{L}$ of all full leaves intersecting the sphere with center $\mathbf{p}_u$ and radius d, where d is the pair width used in the offline phase (see Section 4.1). This can be performed very efficiently due to the hierarchical structure of the octree. Finally, a leaf is drawn uniformly from $\mathbf{L}$ and $\mathbf{p}_v$ is set to be its center of mass.

Step 4) Normal Estimation

We estimate the normals $\mathbf{n}_u$ and $\mathbf{n}_v$ at the points $\mathbf{p}_u$ and $\mathbf{p}_v$ by performing a PCA: $\mathbf{n}_u$ and $\mathbf{n}_v$ are set to be the eigenvectors corresponding to the smallest eigenvalues of the covariance matrix of the points in the neighborhood of $\mathbf{p}_u$ and $\mathbf{p}_v$. The result of this step is the oriented scene point pair $(\mathbf{u}, \mathbf{v}) = ((\mathbf{p}_u, \mathbf{n}_u), (\mathbf{p}_v, \mathbf{n}_v))$.

Step 5) Hash Table Access

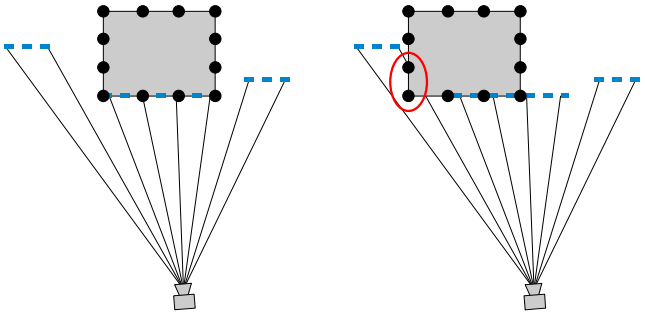
In line 10, $f_{\mathbf{uv}} = (f_2(\mathbf{u}, \mathbf{v}), f_3(\mathbf{u}, \mathbf{v}), f_4(\mathbf{u}, \mathbf{v}))$ is computed according to (1). Next, in lines 11 and 12, $f_{\mathbf{uv}}$ is used as a key to the model hash table to retrieve all model pairs $(\mathbf{u}_j, \mathbf{v}_j)$ similar to $(\mathbf{u}, \mathbf{v})$.

Step 6) Generate and Test

For each $(\mathbf{u}_j, \mathbf{v}_j)$, its model $\mathbf{M}$ is retrieved (line 14) and the rigid transform T which aligns $(\mathbf{u}_j, \mathbf{v}_j)$ to $(\mathbf{u}, \mathbf{v})$ is computed according to (2) (line 15). This results in the hypothesis that the model $\mathbf{M}$ is in the scene at the location defined by T. Finally, the hypothesis is saved in the solution list $\mathcal{T}$ if it is accepted by the acceptance function μ (line 16).

Acceptance Function

μ consists of a visibility term and a penalty term. Similar to RANSAC, the visibility term, μ_V, is proportional to the number m_V of transformed model points which fall within a certain ϵ-band of the scene. More precisely, we set $\mu_V(\mathbf{M}, T) = m_V/m$, where m



(a) $\mu_V = 4/12,\ \mu_P = 0.$ (b) $\mu_V = 3/12,\ \mu_P = 2/12.$

Fig. 3. A 2D top schematic view of the same scene (blue dashed line) with two different model hypotheses (models are shown as gray boxes). The lines of sight are shown as thin black lines originating from the scanning device. In (a), 4 (out of 12) model points match the scene and no model points are occluding scene points. In (b), 3 model points match the scene and 2 model points (marked by the ellipse) are occluding scene points. The resulting values for μ_V and μ_P are shown below the corresponding figure.

is the total number of model points. μ_V is an approximation of the visible object surface area expressed as a fraction of the total object surface area. Thus, μ_V can be interpreted as an estimation of the object visibility in the scene.

In contrast to RANSAC, our algorithm contains an additional penalty term, μ_P, which is proportional to the number of transformed model points which occlude the scene. Obviously, a correctly recognized and localized non-transparent object should not occlude any visible scene points when the scene is viewed from the viewpoint of the range scanner. In other words, if we view the scene from the perspective of the scanning device, we will not be able to see scene points lying behind the localized model since we cannot see through non-transparent surfaces. The penalty term penalizes hypotheses which violate this condition. It is computed by counting the number m_P of transformed model points which are between the projection center of the range image and a range image pixel and thus are "occluding" reconstructed scene points. We set $\mu_P(\mathbf{M}, T) = m_P/m$, where m is the number of model points.

For $(\mathbf{M}, T)$ to be accepted by μ as a valid hypothesis it has to fulfill

$$\mu_V(\mathbf{M}, T) > V \text{ and } \mu_P(\mathbf{M}, T) < P, \tag{7}$$

where $V \in [0, 1]$ is a visibility and $P \in [0, 1]$ a penalty threshold. In Fig. 3, a simple scene is shown with two different model hypotheses and the corresponding values for μ_V and μ_P.

The visibility threshold is one of the most crucial parameters in the algorithm. In the experimental part of the paper, we examine how this threshold affects the recognition and the false positives rates of our method. In the case of perfect data, the penalty threshold P should be 0. However, since we are dealing with real range images, we use $P = 0.05$.

Step 7) Removing Conflicting Hypotheses

A hypothesis $(\mathbf{M}, T)$ "explains" a subset $\mathbf{P} \subset \mathbf{S}^*$ if there are points from $T(\mathbf{M})$ lying in the octree leaves corresponding to $\mathbf{P}$. After accepting $(\mathbf{M}, T)$, the points explained by it are *not* removed from $\mathbf{S}^*$ because there could be a better hypothesis, i.e., one which explains a superset of $\mathbf{P}$. We call hypotheses conflicting if the intersection of the point sets they explain is non-empty. In other words, conflicting hypotheses transform their models such that they intersect in space.

Since the scene points explained by the accepted hypotheses are not removed from $\mathbf{S}^*$, there are many conflicting ones in the solution list $\mathcal{T}$ after the execution of the main loop (lines 4 to 20) of Algorithm 1. To filter the weak hypotheses, we construct a so-called conflict graph. Its nodes are the hypotheses in $\mathcal{T}$ and an edge is connecting two nodes if the hypotheses are conflicting ones.

To produce the final output, the solution list is filtered by performing a non-maximum suppression on the conflict graph. We borrow this technique from image processing. To perform a non-maximum suppression on a gray-scale image, the pixel under observation is set to zero (it is suppressed) if its value is not a maximum in a window placed around that pixel. In this case, the window defines the neighborhood of each pixel. In the case of our conflict graph, the neighborhood is defined by the graph structure. Using the neighborhood of each node, we perform non-maximum suppression essentially in the same way as in image processing: a node η is suppressed if there is a better one in its neighborhood, i.e., a node which explains more scene points than η.

4.3 Time Complexity

The dominating factor in the complexity of the proposed method is the number N of iterations needed to recognize all models with a predefined success probability (see the main loop of Algorithm 1, lines 4 to 20). In the following, we discuss the dependency of N on the number of scene points.

Consider a scene $\mathbf{S}^*$ consisting of $|\mathbf{S}^*| = n$ points and a model instance $\mathbf{M}$ therein consisting of $|\mathbf{M}| = m$ points. In Section 3.2 on RANSAC, we derived the number $N = \frac{\ln(1-P_S)}{\ln(1-P_M)}$ of iterations required to recognize $\mathbf{M}$ with a predefined success probability P_S, where P_M is the probability of recognizing $\mathbf{M}$ in a single iteration. Again in Section 3.2, we obtained $P_M \approx 1/n^3$ which resulted in the cubic time complexity of RANSAC. In the following, we show that our sampling strategy and the use of the model hash table lead to a significant increase of P_M and thus to a reduction of the complexity.

If $P(\mathbf{p}_u \in \mathbf{M}, \mathbf{p}_v \in \mathbf{M})$ denotes the probability that both points are sampled from $\mathbf{M}$ (lines 5 and 7 of Algorithm 1), then the probability of recognizing $\mathbf{M}$ in a single iteration is

$$P_M = KP(\mathbf{p}_u \in \mathbf{M}, \mathbf{p}_v \in \mathbf{M}), \tag{8}$$

with K being the fraction of oriented point pairs for which the descriptors are stored in the model hash table (see Section 4.1). Using conditional probability and the fact that $P(\mathbf{p}_u \in \mathbf{M}) = m/n$ we can rewrite (8) to obtain

$$P_M = (m/n)KP(\mathbf{p}_v \in \mathbf{M} | \mathbf{p}_u \in \mathbf{M}). \tag{9}$$

$P(\mathbf{p}_v \in \mathbf{M} | \mathbf{p}_u \in \mathbf{M})$ denotes the probability to sample $\mathbf{p}_v$ from $\mathbf{M}$ given that $\mathbf{p}_u \in \mathbf{M}$. Recall from Section 4.2 that $\mathbf{p}_v$ depends on $\mathbf{p}_u$ because it is sampled uniformly from the set $\mathbf{L}$ of scene points which are close to the sphere $\mathcal{S}_d(\mathbf{p}_u)$ with center $\mathbf{p}_u$ and radius d, where d is the pair width used in the offline phase. Assuming that the visible object part has an extent larger than $2d$ and that the reconstruction is not too sparse, $\mathbf{L}$ contains points from $\mathbf{M}$. In this case, $P(\mathbf{p}_v \in \mathbf{M} | \mathbf{p}_u \in \mathbf{M}) = |\mathbf{L} \cap \mathbf{M}|/|\mathbf{L}|$ is well-defined and greater than zero.

Let us discuss $C := |\mathbf{L} \cap \mathbf{M}|/|\mathbf{L}|$. It depends on the scene clutter, the number of outliers and the extent and shape of the visible object part. If all scene points originate from known objects (in particular there is no background) and if the objects are well separated then $|\mathbf{L} \cap \mathbf{M}| = |\mathbf{L}|$ since the sphere $\mathcal{S}_d(\mathbf{p}_u)$ intersects scene octree leaves containing only points from the object $\mathbf{p}_u$ belongs to. In this extreme case, $C = 1$. On the other hand, occluded scenes with many outliers can be constructed in which $\mathcal{S}_d(\mathbf{p}_u)$ intersects only objects other then the one $\mathbf{p}_u$ belongs to. This leads to $C = 0$ and simply means that the object is too occluded to be recognized.

In our implementation, we estimate C by $1/4$. This accounts for up to 75% outliers and scene clutter. Thus, we obtain for P_M as a function of n (the number of scene points)

$$P_M(n) = (m/n)KC. \tag{10}$$

Approximating the denominator $\ln(1 - P_M(n))$ in (5) by its Taylor series $-P_M(n) + O(P_M(n)^2)$ we obtain for the number of iterations

$$N(n) \approx \frac{-\ln(1 - P_S)}{P_M(n)} = \frac{-n \ln(1 - P_S)}{mKC} = O(n). \tag{11}$$

This shows that the number of iterations depends linearly on the number n of scene points. Furthermore, Eq. (11) provides means for computing the number of iterations required to recognize the model instances with the desired success probability P_S.

5 Object Manipulation

The object recognition is only one link in the overall processing chain. In order to enable the robot to perform the recognition together with the object manipulation in a loop and to recover from faulty grasps, the overall process is controlled by a hybrid state automaton. The scheme is based on the work in [Haddadin et al., 2009], [Parusel et al., 2011], [Fuchs et al., 2010] and relies on the disturbance observer introduced in [Haddadin et al., 2008]. The high-level schematic is shown in Fig. 4 and consists of the following phases:

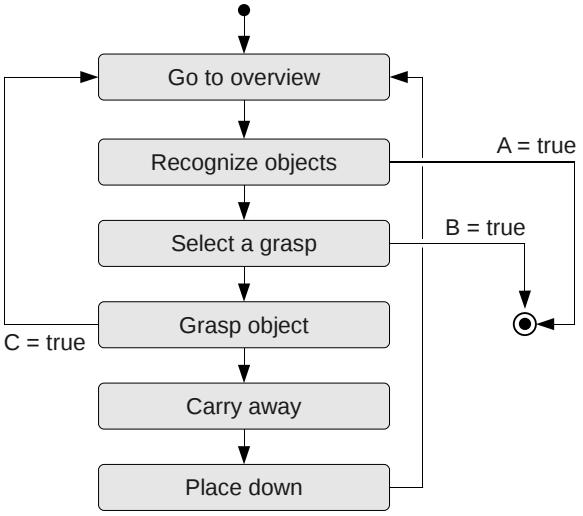


Fig. 4. Hybrid state automaton for controlling the overall object manipulation process. The logical clauses A, B and C defining the transition conditions, are the following: $A = no - object$, $B = no - grasp$ and $C = collision \vee failed - grasp$. A grasp is considered as a failure if the robot gripper completely closes.

1) Go to overview: The robot moves to an overview pose in order to avoid scene occlusion.
2) Recognize objects: The object recognition method recognizes the objects in the scene (see Section 4).
3) Select a grasp: Select an object (from the list of recognized ones) together with a suitable grasp (see Section 5.1).
4) Grasp object: The robot performs the grasp on the selected object (see Section 5.2).
5) Carry away: The robot carries the object to the place designated for it.
6) Place down: The robot softly and safely puts the object on its place (see Section 5.2).

Next, the phases 3), 4) and 6) are explained in detail. Phase 5), bringing an object to a specified location, is out of the scope of this paper and will not be discussed any further.

5.1 Grasp Selection

The first step in the manipulation chain is the selection of an object (from the list of recognized objects returned by the recognition method) together with a suitable grasp. Each object in the database is associated with a finite set of plausible grasps. A grasp

G (also called grasp frame) consists of an orientation and a position of the robot end effector relative to the object. The orientation is represented by a 3×3 rotation matrix. Its last column is a vector, called the approach vector $\mathbf{v}_{appr}$, which is aligned with the z-axis of the end effector and points towards the object.

The idea of the grasp selection is to go for the up most object, i.e., the one whose center of mass has the largest z-coordinate. This is measured according to the world coordinate system which has its z-axis, $\mathbf{z}_w$, perpendicular to the table the objects are placed on. (More details on the scene setup will be given in Section 6.3.) Next, among the grasp frames associated with the selected object, the one with the lowest

$$\mathrm{cost}(G) = w_1 \mathrm{crit}_1(G) + w_2 \mathrm{crit}_2(G), \tag{12}$$

is chosen, where

$$\mathrm{crit}_1(G) = \angle(-\mathbf{v}_{appr}, \mathbf{z}_w) \tag{13}$$

is the alignment of the end effector with respect to the z-axis of the world coordinate system and

$$\mathrm{crit}_2(G) = |\varphi_0^{wrist} - \varphi_{req}^{wrist}|, \tag{14}$$

is the absolute difference between φ_0^{wrist}, a desired (neutral) wrist orientation, and φ_{req}^{wrist}, the one required to perform the grasp. In our implementation, we set $w_1 = 3$ and $w_2 = 1$ (see (12)) which makes the end effector alignment more important than the wrist orientation. Note that (12) uses the negative of $\mathbf{v}_{appr}$ such that it shows in the same direction as $\mathbf{z}_w$. Furthermore, the selected grasp is discarded if it is "too parallel" to the table plane, i.e., if $\mathrm{crit}_1(G) \geq \varphi_z$ (we set $\varphi_z = 30°$). If this is the case, then the next lower object is inspected.

The selected grasp frame G is projected 0.1 m along the approach vector for acquiring the grasping motion.

5.2 Impedance Control and Collision Detection for Sensitive Grasping and Placing

To achieve robust and careful grasping and placing, we employ the Cartesian impedance controlled LWR-III [Albu-Schäffer et al., 2007]. Especially its soft-robotics features are crucial for such a delicate task. Since it is equipped with torque sensors in every joint, both impedance and accurate position control are possible at the same time. In order not to damage its environment and in particular the objects it is supposed to manipulate, the robot should be able to quickly detect collisions and safely react to them. The collision detection method we use was introduced and evaluated in [De Luca et al., 2006], [Haddadin et al., 2008]. It provides not only binary contact information but also an accurate estimation of the external torques. Based on this additional input, we employ a decision algorithm [Haddadin et al., 2009] which enables the robot to react in an appropriate manner to unexpected interaction forces. For example, such forces occur when the object pose estimated by the object recognition algorithm is too imprecise such that the robot collides with the object as it reaches for it. However, since the robot quickly detects the collision it is able to stop before damaging the object. After that, the robot moves to a well-defined position and restarts the recognition process (as depicted in Fig. 4).

In order to demonstrate the importance of impedance control and collision detection for a safe object manipulation, we conducted a series of object grasping and placing experiments with different impedances and with distinct collision behavior.

Fig. 5 and 6 illustrate the robot behavior while grasping and placing an object using different stiffness values and tip velocities with and without collision detection. The plots depict the z-coordinates of the desired (dashed curves) and measured (continuous curves) contact force F_{ext_z}, position z and velocity $\dot{z}$. In both figures, the left columns are obtained for a high stiffness value while the right ones for a low value. The red curves indicate a lower tip velocity (about 0.1 m/s) while the blue ones a higher velocity (about 0.8 m/s). The upper row is obtained without collision detection and reaction, while the lower one visualizes the binary collision detection signal, $hf1$, together with a respective response which obviously leads to different curves.

If collision detection and handling is activated, a contact is classified as a collision if the magnitude of the contact force exceeds a certain limit. According to the figures, a high stiffness always leads to very high contact forces which might destroy the object to be handled. In contrast, low stiffness and collision detection contribute to a significant reduction of contact forces and practically eliminate the risk of damaging the object if a planned grasp is misaligned.

Fig. 7 shows an image sequence of grasping a bottle with low stiffness. We intentionally degraded the pose estimation accuracy such that the end effector collided with the bottle. Nevertheless, because of the impedance control, the end effector was able to adjust and successfully grasped the bottle.

Extension 1 exemplary shows some of the experiments described in this section.

6 Experimental Results

In this Section, we experimentally validate the proposed geometry matching algorithm (Section 6.1), the impedance controlled grasping strategy (Section 6.2) and the grasping capabilities of the overall system (Section 6.3). The object models involved in the tests are shown in Fig. 8.

The algorithm presented in this paper is implemented in C++ and all tests were performed on a Linux PC with 4GB RAM and an Intel Xeon 2.67GHz CPU with four cores.

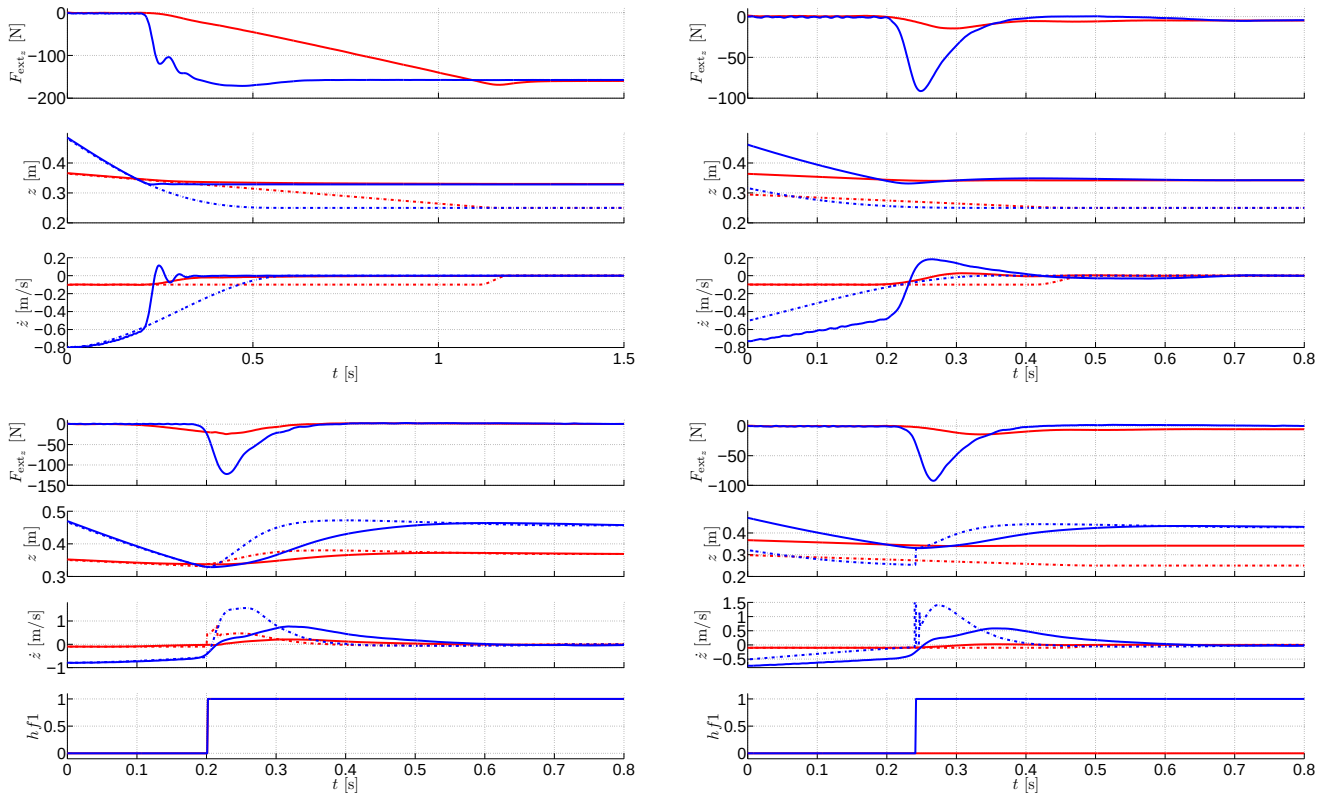


Fig. 5. Grasping behavior with high stiffness (left) and low stiffness (right) without (top) and with (bottom) collision detection and reaction.

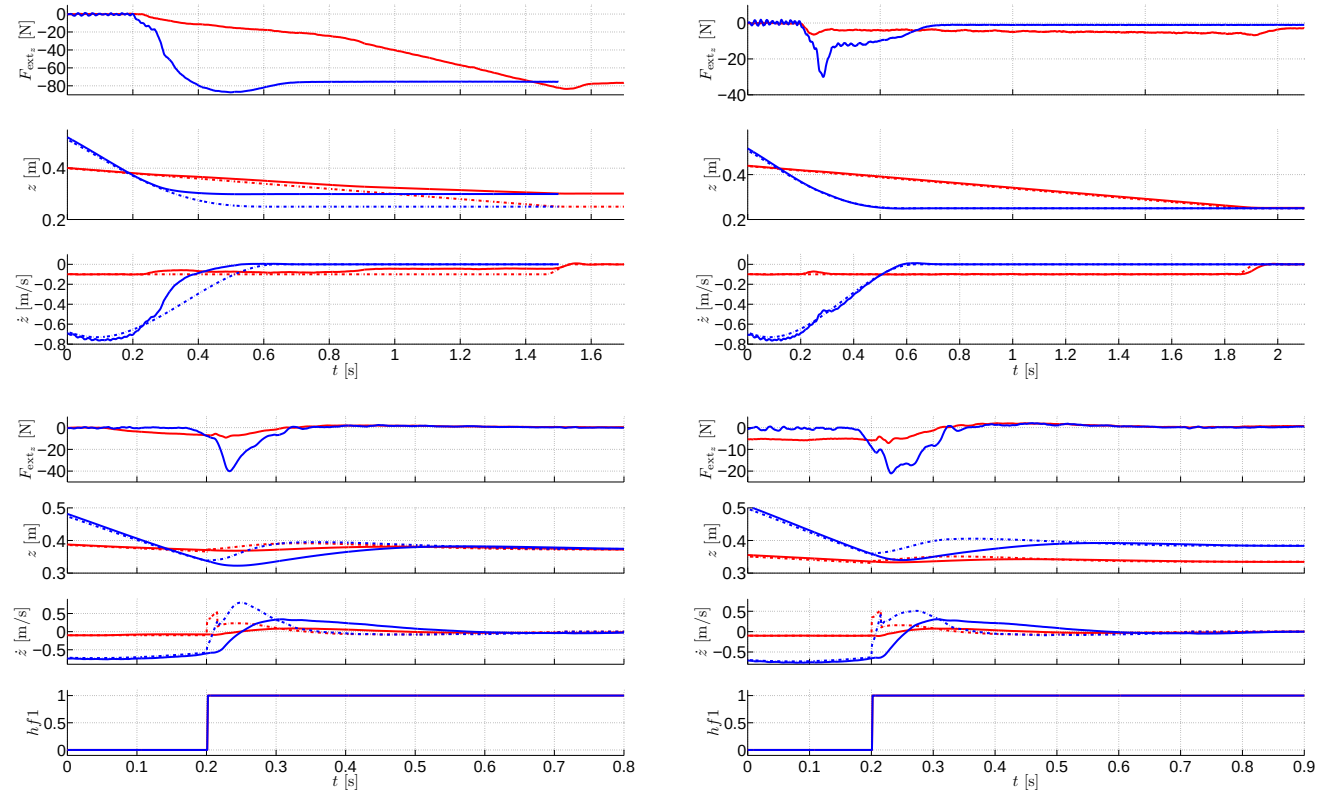


Fig. 6. Placing behavior with high stiffness (left) and low stiffness (right) without (top) and with (bottom) collision detection and reaction.

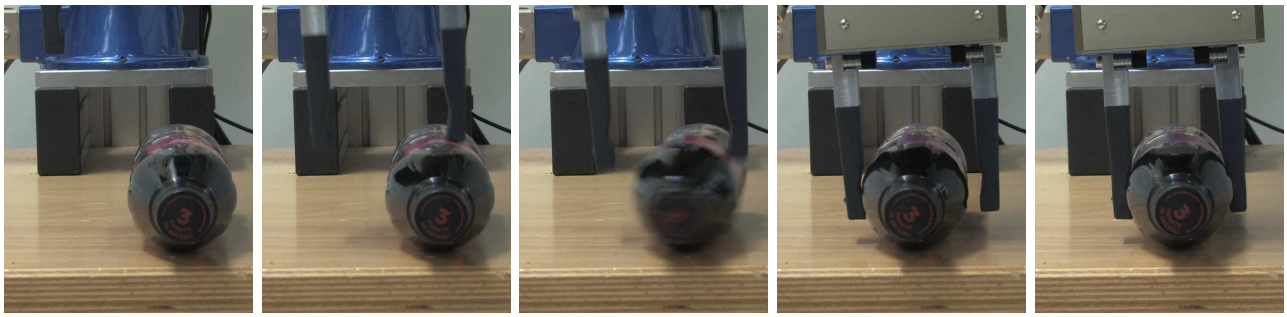


Fig. 7. Grasping behavior with low stiffness. Note that despite the imprecise object pose the robot is able to grasp the bottle.

Fig. 8. (a) The models used in the tests of the geometric matching algorithm (models provided by [Mian et al., 2006]). (b) The models involved in the grasping test scenarios (our own scans made with a low-cost light intersection-based scanning device).

6.1 Rigid 3D Geometry Matching

The scenes used in the following test cases were digitized with a Minolta VIVID 910 scanner [Konica Minolta, 2011] and provided by [Mian et al., 2006]. Examples and more information about the scenes will be given in the following subsections.

6.1.1 Matching a Single Object in Occluded Scenes

In the first test scenario, we examined how the success rate and the false positives rate of the geometry matching algorithm depend on the most important parameter, namely, the visibility threshold (introduced in Section 4.2) and the actual object occlusion in a scene. According to [Johnson and Hebert, 1999], the occlusion of an object model is given by

$$occlusion = 1 - \frac{\text{visible model surface area}}{\text{total model surface area}}. \quad (15)$$

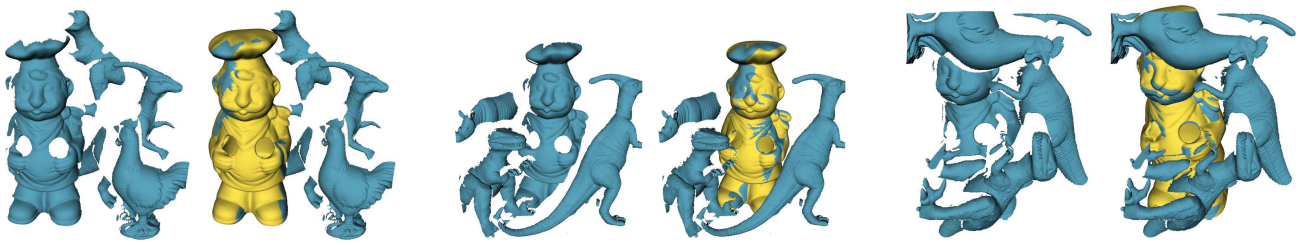
The aim of this test was to establish a value for the visibility threshold which, on the one hand, results in a high success rate even in highly occluded scenes, and on the other hand leads to as few as possible false positives. In this experiment, only the model of the Chef (Fig. 8(a)) was used for matching in three different scenes each one containing a total of three or four objects. The Chef was present in each scene at different locations and at different levels of occlusion (self-occlusion as well as occlusion caused by the other objects). The three test scenes together with typical recognition results are shown in Fig. 9.

Since the matching algorithm is a probabilistic one, we ran 100 trials on each scene and computed the recognition (success) rate and the mean number of false positives in the following way. We visually inspected the result of each trial. If object $\mathbf{A}$ (in this case only the Chef) was recognized k times ($0 \leq k \leq 100$), then the recognition rate for $\mathbf{A}$ is $k/100$. The mean number of false positives is $(k_1 + ... + k_{100})/100$, where k_i is the number of false alarms in the i-th trial.

The results of the test are summarized in Fig. 10. As expected, the visibility threshold had to fall below a certain value, namely, $1 - occlusion$ in order to achieve a positive recognition rate. More importantly, the plots suggest that the number of false positives practically does not depend on the actual level of occlusion but mainly on the visibility threshold: in all three cases it starts to grow when the visibility threshold falls below 0.15. In summary, it can be said that the method achieved a recognition rate of 1.0 in highly occluded scenes (up to 85% occlusion) at the cost of no false positives. In order to handle more occlusion the visibility threshold had to fall below 0.15 which gave rise to some false positives.

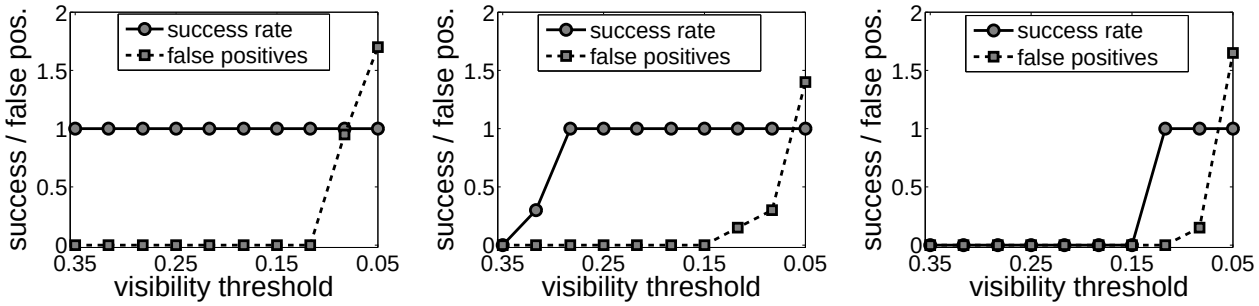
6.1.2 Matching Multiple Objects in Noisy Scenes

In this scenario, we tested the algorithm under varying noisy conditions. The four models involved are shown in Fig. 8(a) and the noise-free scene is shown in



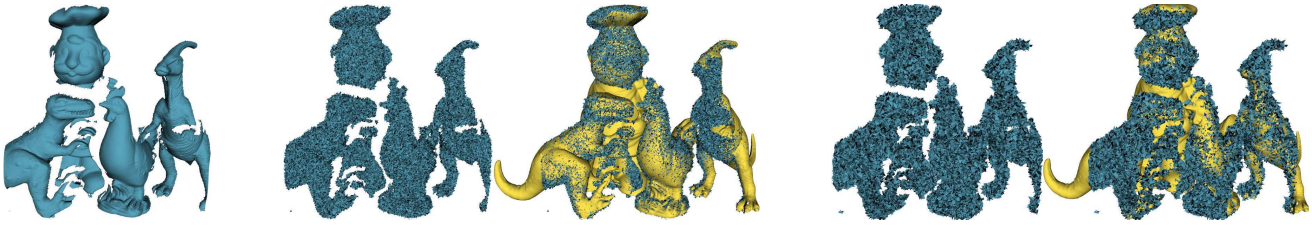
(a) 62.3% object occlusion (b) 70.4% object occlusion (c) 86.2% object occlusion

Fig. 9. The test scenes used for the Chef model matching. The level of occlusion for the Chef is indicated for each scene. On the left of each subfigure, the input scene is shown as a blue mesh, whereas on the right, the recognized Chef model is placed at the location computed by our algorithm and rendered as a yellow mesh.



(a) 62.3% object occlusion (b) 70.4% object occlusion (c) 86.2% object occlusion

Fig. 10. The success rate and the mean number of false positives as functions of the visibility threshold for three different scenes each one containing the Chef model at an occlusion level of (a) 62.3%, (b) 70.4% and (c) 86.2%.



(a) Noise-free (b) $\sigma^2 = 0.25$ (c) $\sigma^2 = 1.0$

Fig. 11. (a) Noise-free scene. (b), (c) Typical recognition results for data sets degraded by zero-mean Gaussian noise for different variance σ^2 which is given as percentage of the bounding box diagonal length of the scene. On the left side of each subfigure, only the noise-corrupted scene is shown, whereas the right side shows the scene plus the recognized models placed at the locations estimated by the matching algorithm.

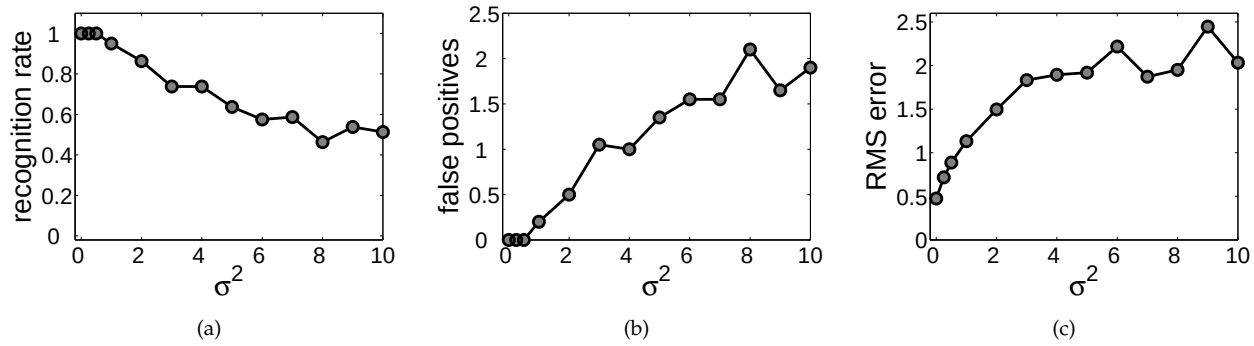
Fig. 11(a). We degraded the noise-free scene with zero-mean Gaussian noise with different variance values σ^2. Again, 100 recognition trials on each noisy scene were performed and the recognition rate, the mean number of false positives and the Root Mean Square error (RMSe) were computed as functions of σ^2.

For two point sets $\mathbf{P}$ and $\mathbf{Q}$ and a transform T the RMS error measures how close each point $\mathbf{q}_i \in \mathbf{Q}$ comes to its corresponding point $\mathbf{p}_i \in \mathbf{P}$ after transforming $\mathbf{Q}$ by T [Gelfand et al., 2005]. The smaller the error the closer the alignment between the point sets. More formally,

$$\mathrm{RMSe}(\mathbf{P}, \mathbf{Q}, T) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \|\mathbf{p}_i - T(\mathbf{q}_i)\|^2}, \qquad (16)$$

with N being the number of points. Since the ground truth location of each model in the test scene is known, the RMS error of the rigid transform computed by our method was easily calculated[2]. Note that we did *not* compute the RMS error for the transformed model and the scene but for the transformed model and the same model placed at the ground truth location. Fig. 11(b) and (c) exemplary show typical recognition results for two of the twelve noisy scenes. The results of the tests are reported in Fig. 12.

2. The ground truth rigid transform for the models is available on http://www.csse.uwa.edu.au/~ajmal/recognition.html



(a) (b) (c)

Fig. 12. (a) Recognition rate, (b) mean number of false positives and (c) RMS error as functions of the variance σ^2 of Gaussian noise. Note that the RMS error is computed for the successful trials only. Both σ^2 and the RMS error are given as percentage of the bounding box diagonal length of the scene.

6.1.3 Comparison with Spin Images and Tensor Matching

Next, we compared the recognition rate of our geometry matching algorithm with the spin images [Johnson and Hebert, 1999] and the tensor matching [Mian et al., 2006] approaches on the same 50 data sets used in [Mian et al., 2006]. This made a direct comparison possible without the need of re-implementing either of the two algorithms. The models of the four toys involved in the tests are shown in Fig. 8(a). The toys (not necessarily all four of them) are present in the scenes in different positions and orientations. Since each scene was digitized with a laser range finder from a single viewpoint the back parts of the objects were not visible. Furthermore, the toys were usually placed such that some of them occluded others which made the visible object parts even smaller. Four (out of the 50) test scenes are shown in Fig. 9 and Fig. 11(a). Again, we ran 100 recognition trials on each scene and computed the recognition rate for each object in the way described in Section 6.1.1. Since the occlusion of every object in the test scenes was known we report the recognition rate for each object as a function of its occlusion. The result of the comparison is summarized in Fig. 13(a). Note that the chef was recognized in all trials, even in the case of occlusion over 91%. The blue dots represent the recognition rate in the three chicken test scenes in which our method performed worse than the other algorithms. This was due to the fact that in these scenes only the chicken's back part was visible which contains strongly varying normals which made it difficult to compute a stable aligning transform.

Our method needed in average about 7.5 seconds for the recognition of the objects in each scene and sampled about 450 oriented point pairs per scene. For a comparison, 250 tensors, respectively, 4000 spin images per scene were used in the experiments performed in [Mian et al., 2006].

6.1.4 Runtime

We experimentally validated the linear time complexity of the matching algorithm in the number of scene points. Eleven different data sets were involved in this test case — a subset from the scenes used in the comparison test case (Section 6.1.3). Note that we did *not* take a single data set and down/up-sampled it to get the desired number of points. Instead, we chose eleven *different* scenes with varying scene extent, number of points and number of objects. This suggests that the results will hold for arbitrary scenes. We report the results of this test in Fig. 13(b).

Note that the iterations of the main loop of the matching algorithm (lines 4 to 20 of Algorithm 1) can be executed independently of each other which makes it possible to run them in parallel. This is a very important issue since parallel computing has become the dominant paradigm in computing architectures, mainly in the form of multicore processors [Asanovic et al., 2006]. In Fig. 13(c), we report the processing time as a function of the used CPU cores. Note that the parallel execution on four cores runs more than three times faster than on a single core. This indicates a great potential for further speed-up when more CPU cores become available.

6.2 "Blind" Grasping with an Impedance Controlled Robot

In contrast to Section 5.2, where the importance of impedance control and collision detection for a safe object manipulation was demonstrated, in this section, we conducted a series of grasping experiments with the aim to find a set of impedance parameters that maximizes the grasping success in the presence of simulated object pose errors. The robot altered its Cartesian translation stiffness in y-direction (denoted by $K_{t,y}$) and its rotation stiffness in x-direction (denoted by $K_{r,x}$). These directions are the lateral compliance along the gripper motion and the rotation perpendicular to this. Due to the inherent structure of the gripper, they are the significant parameters

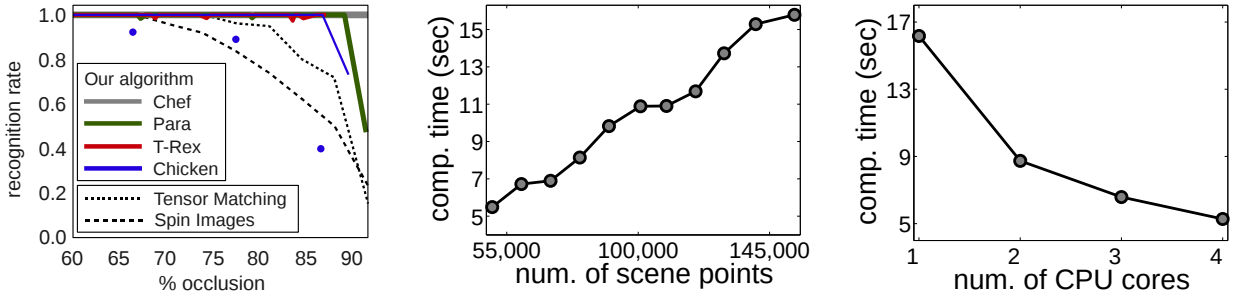


Fig. 13. (a) Comparison with spin images [Johnson and Hebert, 1999] and tensor matching [Mian et al., 2006]. The recognition rate of our algorithm for each object as a function of its occlusion is indicated by the continuous lines. The dashed lines give the recognition rate of the spin images and the tensor matching approaches on the same scenes as reported in [Mian et al., 2006]. Note that our algorithm outperforms both other methods. (b) Computation time as a function of the number of scene points for the simultaneous recognition of seven models. The line indicates a linear complexity. (c) Runtime as a function of the number of used CPU cores for the recognition of seven models in a scene consisting of around 60,000 points.

	$K_{t,y}$[N/m]	$K_{r,x}$ [Nm/rad]	success [%]
1	200	20	60
2	200	75	80
3	200	200	90
4	750	20	70
5	750	75	80
6	750	200	40
7	2000	20	50
8	2000	75	60
9	2000	200	70

TABLE 1
Grasping success with varying stiffness for a translational object pose error of 1.5 cm.

Fig. 14. The setup of the vision-based grasping experiments. The robot has grasped a green soda club bottle and is about to put it in the further red bin. The Kinect sensor can be seen in the upper right corner. In the lower left corner, the range image and the recognized models are shown (before the bottle has been taken away) from a viewpoint close to the one of the sensor.

governing the grasping process. The object involved in this test scenario was the soda club bottle (Fig. 8(b)). The object pose error was simulated by translating the bottle by 1.5 cm in a random direction parallel to the table in front of the robot. We performed ten grasping trials for each of the following stiffness configurations: "soft", "moderately stiff" and "rigid". The success rate is listed in Table 1. The optimal values (line 3) correspond to a soft (very compliant) translation and a rigid rotation behavior. A soft translation and a moderately stiff rotation (line 2) as well as a moderate stiffness in both translation and rotation (line 5) led to good success rates too.

6.3 Vision-Based Impedance Controlled Grasping

In this Section, we experimentally validate the overall vision-based impedance controlled grasping system. The models involved in these tests are shown in Fig. 8(b). We started with grasping single standing objects, moved on to object grasping from an unsorted pile and finished with a more complex task of cleaning up a table. We used the 7-degrees-of-freedom Cartesian impedance controlled DLR Lightweight robot III developed at the German Aerospace Center (DLR). It was mounted on a table

	object		
test scenario	Soda Club	Amicelli	Rusk
single standing objects	100%	95%	95%
object pile	95%	95%	90%

TABLE 2
Success rates in the grasping experiments.

Fig. 15. (Left) The single standing object grasping scenario. (Middle, Right) two input range images (top) and the recognition results (bottom) for the rusk and the Amicelli box, respectively. The points off the plane (used for matching) are shown in light blue.

and covered an area of approximately 2.5 square meters. The scene was digitized with a Kinect sensor [Kinect for Xbox 360, 2011]. Since all objects were standing on or above the table, its plane was detected in each range image (using a simple RANSAC procedure) and all points belonging to the plane or lying below were removed. The setup is shown in Fig. 14.

6.3.1 Grasping Single Standing Objects

In the first scenario, multiple grasps were performed on single standing objects (see Fig. 15). We varied the pose of the objects such that all pre-saved grasp poses were executed. A grasp trial was considered successful if the object was correctly recognized, grasped and carried to the right place (table corner for the rusk box or one of the red bins for the Amicelli box/soda club bottle). We ran ten trials for each object pose and recorded the number of successful trials. The results are summarized in the first row of Table 2. One grasp failed for the Amicelli and the rusk box, respectively. This was due to the fact that the alignment computed by the matching algorithm was too imprecise.

6.3.2 Grasping from an Object Pile

Next, the robot performed multiple grasps on a pile consisting of seven objects placed next and on top of each other. Again, we changed the positions of the items such that the robot tried all pre-saved grasp poses for each object. A grasp was considered successful if the robot picked a correctly recognized object and carried it to the right place. After an object had been taken away, we built up a new pile, i.e., the robot had to deal every time with a full pile. This experiment added some additional difficulties to both the geometry matching and the robot control algorithms. Obviously, the risk of recognition failures increased since there were more objects in the scene. Besides that, objects in a pile are in a more unstable configuration (from a statics point of view) compared to single standing ones. This made it more difficult for the impedance-based control to compensate for matching imprecision. We performed ten trials for each grasp pose and recorded the number of successful trials. The results are compiled in the second row of Table 2. As to be expected, the failure rate increased compared to the first grasping experiment.

6.3.3 Cleaning up the Table

In the last test scenario, we let the robot repeatedly perform a more complex task, namely, cleaning up the table in front of it. Seven objects were randomly placed on a pile which resulted in highly cluttered and occluded scenes. The task to the robot was to pick each object, put it away and halt when it "believes" that the table is empty. The recognition process was restarted each time an object was carried away. Thus, the robot had to deal with the changing scene and with unforeseen situations which happened during the cleanup like, e.g., an object falling off the pile. The task was accomplished if at the end each object was at the place designated for it. This time we did not consider it a failure when an object slipped out of the gripper as long as it was picked up later on and left in the right place. After each cleanup trial we built up a new pile and let the robot perform the task again. We repeated this 15 times and counted the number of successful trials. The robot achieved a success rate of 80%. Fig. 16 exemplary shows one cleanup process. More examples can be seen in Extension 2.

7 Conclusions and Future Work

In this paper, we introduced a vision-based approach for grasping of known objects. Our approach relies on a robust 3D geometry matching algorithm for the recognition and localization of multiple objects in noisy, partially reconstructed and unsegmented scenes. The matching algorithm is based on a robust geometric descriptor, a hashing technique and an efficient, localized RANSAC-like sampling strategy. We provided a theoretical complexity analysis and derived a formula for computing the number of iterations required to recognize the objects with a predefined success probability. The result of the complexity analysis, namely a linear time dependency on the number of scene points, was experimentally validated. Tests on real range data confirmed that our method performs well on complex scenes in which only small parts of the objects are visible. In a direct comparison with the spin images [Johnson and Hebert, 1999] and the tensor matching [Mian et al., 2006] approaches, our method performed better in terms of recognition rate. A further

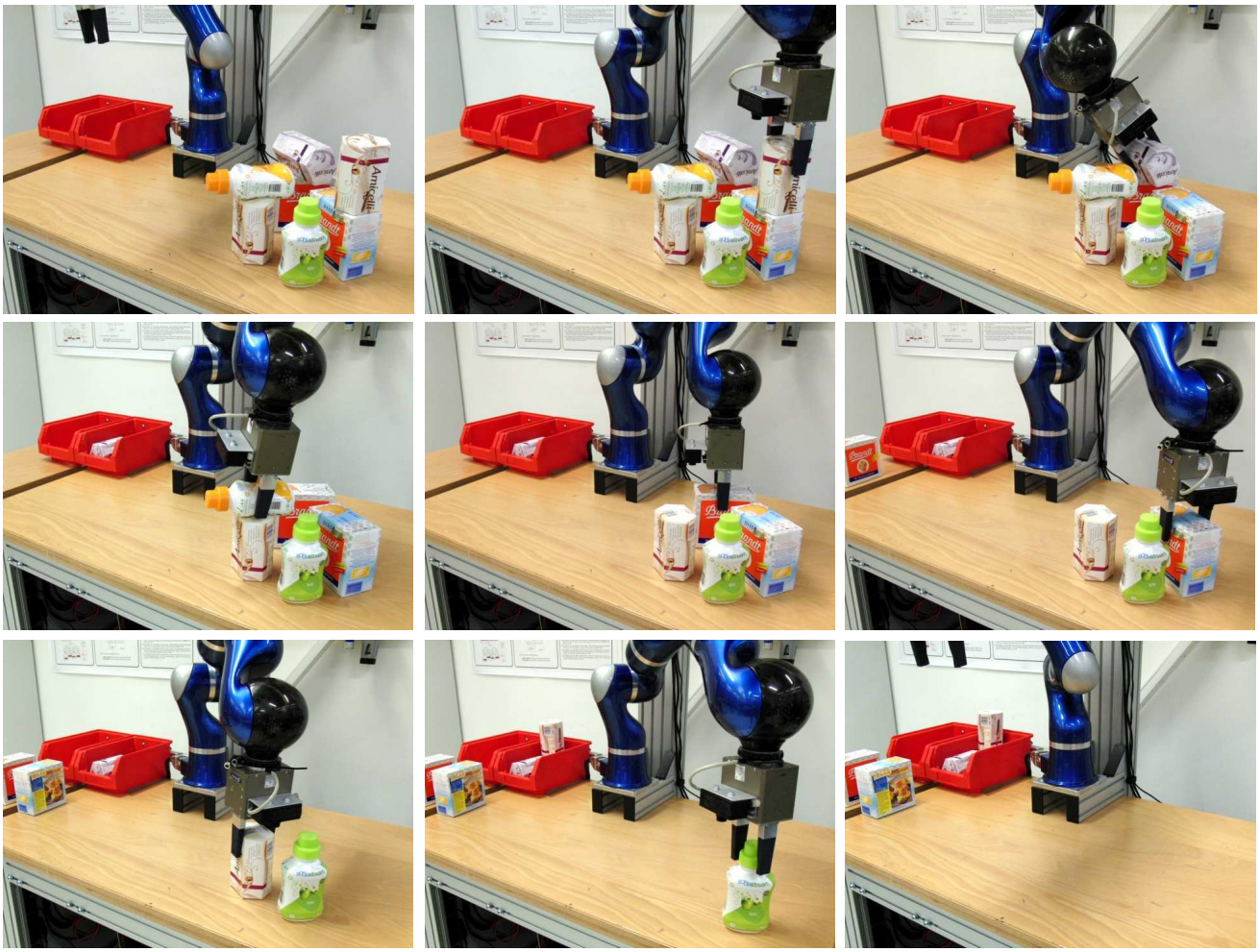


Fig. 16. (Left to right, top to bottom) A sequence of images showing the robot cleaning up the table. The first and the last image show the beginning and the end of the task, respectively. Each in-between shot shows the robot in the moment of grasping an object. Note that the first Amicelli box and all bottles are placed in a lying orientation in the red bins and are not visible from this point of view.

experimental validation with the DLR Lightweight-Robot III showed how well this new method can be exploited for grasping in unstructured and cluttered environments. The presented solution is capable of quickly recognizing and robustly grasping known objects from an unsorted pile of different everyday items. This is extremely useful for typical service robotics or industrial co-worker tasks.

One possible extension of the recognition algorithm would be to incorporate higher dimensional geometric descriptors. We expect this to lead to more uniformly spread model point pairs in the feature space and to further reduce the overpopulation of hash table cells mentioned in Section 4.1. Furthermore, in the current implementation of the recognition method, the pair width d is selected manually based on the extent of the object models stored in the hash table and on the expected degree of object occlusion in the scene. An elaborate way of automatically selecting the right value for d is the current topic of our research.

Acknowledgments

We would like to thank Tim Rokahr for his help. This work has been partially funded by the European Commission's Seventh Framework Programme as part of the projects GRASP and SAPHARI.

Appendix A: Index to Multimedia Extensions

Extension	Media Type	Description
1	Video	Impedance control and collision detection for sensitive grasping and placing.
2	Video	Cleaning up a table full of grocery items.

References

[Aiger et al., 2008] Aiger, D., Mitra, N. J., and Cohen-Or, D. (2008). 4-points Congruent Sets for Robust Pairwise Surface Registration. *ACM Trans. Graph.*, 27(3).

[Albu-Schäffer et al., 2007] Albu-Schäffer, A., Haddadin, S., Ott, C., Stemmer, A., Wimböck, T., and Hirzinger, G. (2007). The DLR lightweight robot - lightweight design and soft robotics control concepts for robots in human environments. *Industrial Robot Journal*, 34(5):376–385.

[Asanovic et al., 2006] Asanovic, K., Bodik, R., Catanzaro, B. C., Gebis, J. J., Husbands, P., Keutzer, K., Patterson, D. A., Plishker, W. L., Shalf, J., Williams, S. W., and Yelick, K. A. (2006). The Landscape of Parallel Computing Research: A View from Berkeley. Technical report, EECS Department, University of California, Berkeley.

[Ballard, 1981] Ballard, D. H. (1981). Generalizing the Hough Transform to Detect Arbitrary Shapes. *Pattern Recognition*, 13(2):111–122.

[Barr, 1981] Barr, A. (1981). Superquadrics and Angle-Preserving Transformations. *Computer Graphics and Applications*, 1(1):11 –23.

[Biegelbauer et al., 2010] Biegelbauer, G., Vincze, M., and Wohlkinger, W. (2010). Model-based 3D object detection. *Mach. Vis. Appl.*, 21(4):497–516.

[Binford, 1971] Binford, T. (1971). Visual Perception by a Computer. In *IEEE Conf. on Systems and Control*.

[de Berg et al., 2000] de Berg, M., van Kreveld, M., Overmars, M., and Schwarzkopf, O. (2000). *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2 edition.

[De Luca et al., 2006] De Luca, A., Albu-Schäffer, A., Haddadin, S., and Hirzinger, G. (2006). Collision detection and safe reaction with the DLR-III lightweight manipulator arm. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS2006), Beijing, China*, pages 1623–1630.

[Dickinson et al., 1997] Dickinson, S. J., Metaxas, D. N., and Pentland, A. (1997). The Role of Model-Based Segmentation in the Recovery of Volumetric Parts From Range Data. *IEEE TPAMI*, 19(3):259–267.

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

[Frome et al., 2004] Frome, A., Huber, D., Kolluri, R., Bülow, T., and Malik, J. (2004). Recognizing Objects in Range Data Using Regional Point Descriptors. In *ECCV*, pages 224–237.

[Fuchs et al., 2010] Fuchs, S., Haddadin, S., Parusel, S., Keller, M., and Kolb, A. (2010). Cooperative bin-picking with time-of-flight camera and impedance controlled DLR Lightweight robot III. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS2008)*, pages 4862–4867.

[Gelfand et al., 2005] Gelfand, N., Mitra, N., Guibas, L., and Pottmann, H. (2005). Robust Global Registration. In *Eurographics Symposium on Geometry Processing*, pages 197–206.

[Grewe and Kak, 1995] Grewe, L. and Kak, A. C. (1995). Interactive Learning of a Multiple-Attribute Hash Table Classifier for Fast Object Recognition. *Computer Vision and Image Understanding*, 61(3):387–416.

[Haddadin et al., 2008] Haddadin, S., Albu-Schäffer, A., Luca, A. D., and Hirzinger, G. (2008). Collision detection & reaction: A contribution to safe physical human-robot interaction. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS2008), Nice, France*, pages 3356–3363.

[Haddadin et al., 2009] Haddadin, S., Suppa, M., Fuchs, S., Bodenmüller, T., Albu-Schäffer, A., and Hirzinger, G. (2009). Towards the robotic co-worker. In *International Symposium on Robotics Research (ISRR2009), Lucerne, Switzerland*.

[Hetzel et al., 2001] Hetzel, G., Leibe, B., Levi, P., and Schiele, B. (2001). 3D Object Recognition from Range Images Using Local Feature Histograms. In *CVPR*, pages 394–399.

[Johnson and Hebert, 1999] Johnson, A. and Hebert, M. (1999). Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes. *IEEE TPAMI*, 21(5):433–449.

[Keren et al., 1994] Keren, D., Cooper, D. B., and Subrahmonia, J. (1994). Describing Complicated Objects by Implicit Polynomials. *IEEE TPAMI*, 16(1):38–53.

[Kinect for Xbox 360, 2011] Kinect for Xbox 360 (2011). http://www.xbox.com/en-US/kinect. Accessed: 20/04/2011.

[Konica Minolta, 2011] Konica Minolta (2011). Minolta VIVID 910. http://www.konicaminolta.com/instruments/products/3d/non-contact/vivid910/index.html. Accessed: 18/09/2011.

[Lamdan and Wolfson, 1988] Lamdan, Y. and Wolfson, H. (1988). Geometric Hashing: A General and Efficient Model-Based Recognition Scheme. In *ICCV*, pages 238–249.

[Matei et al., 2006] Matei, B., Shan, Y., Sawhney, H. S., Tan, Y., Kumar, R., Huber, D. F., and Hebert, M. (2006). Rapid Object Indexing Using Locality Sensitive Hashing and Joint 3D-Signature Space Estimation. *IEEE TPAMI*, 28(7):1111–1126.

[Mian et al., 2006] Mian, A. S., Bennamoun, M., and Owens, R. A. (2006). Three-Dimensional Model-Based Object Recognition and Segmentation in Cluttered Scenes. *IEEE TPAMI*, 28(10):1584–1601.

[Papazov and Burschka, 2010] Papazov, C. and Burschka, D. (2010). An Efficient RANSAC for 3D Object Recognition in Noisy and Occluded Scenes. In *Asian Conference on Computer Vision (ACCV'10)*, pages 135–148.

[Parusel et al., 2011] Parusel, S., Haddadin, S., and Albu-Schäffer, A. (2011). Modular state-based behavior control for safe human-robot interaction: A lightweight control architecture for a lightweight robot. In *IEEE Int. Conf. on Robotics and Automation (IROS2011), Shanghai, China*.

[Schnabel et al., 2007] Schnabel, R., Wahl, R., and Klein, R. (2007). Efficient RANSAC for Point-Cloud Shape Detection. *Comput. Graph. Forum*, 26(2):214–226.

[Solina and Bajcsy, 1990] Solina, F. and Bajcsy, R. (1990). Recovery of Parametric Models from Range Images: The Case for Superquadrics with Global Deformations. *IEEE TPAMI*, 12(2):131–147.

[Stockman, 1987] Stockman, G. (1987). Object Recognition and Localization via Pose Clustering. *Computer Vision, Graphics, and Image Processing*, 40(3):361–387.

[Sun et al., 2009] Sun, J., Ovsjanikov, M., and Guibas, L. J. (2009). A Concise and Provably Informative Multi-Scale Signature Based on Heat Diffusion. *Comput. Graph. Forum*, 28(5):1383–1392.

[Taylor and Kleeman, 2003] Taylor, G. and Kleeman, L. (2003). Robust Range Data Segmentation using Geometric Primitives for Robotic Applications. In *SIP*, pages 467–472.

[Winkelbach et al., 2006] Winkelbach, S., Molkenstruck, S., and Wahl, F. M. (2006). Low-Cost Laser Range Scanner and Fast Surface Registration Approach. In *Proceedings of the 28th DAGM Symposium on Pattern Recognition*, pages 718–728.

[Wu et al., 2010] Wu, H.-Y., Zha, H., Luo, T., Wang, X., and Ma, S. (2010). Global and Local Isometry-Invariant Descriptor for 3D Shape Comparison and Partial Matching. In *CVPR*, pages 438–445.

Computer Vision and Image Understanding 115 (2011) 1598–1609

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

Stochastic global optimization for robust point set registration

Chavdar Papazov *, Darius Burschka

Department of Computer Science, Technische Universität München, Boltzmannstr. 3, 85748 Garching, Germany

ARTICLE INFO

Article history:
Available online 23 July 2011

Keywords:
Rigid registration
Robust cost function
Stochastic global optimization
Generalized BSP tree
Hierarchical decomposition of $SO(3)$
Uniform sampling from spherical boxes

ABSTRACT

In this paper, we propose a new algorithm for pairwise rigid point set registration with unknown point correspondences. The main properties of our method are noise robustness, outlier resistance and global optimal alignment. The problem of registering two point clouds is converted to a minimization of a nonlinear cost function. We propose a new cost function based on an inverse distance kernel that significantly reduces the impact of noise and outliers. In order to achieve a global optimal registration without the need of any initial alignment, we develop a new stochastic approach for global minimization. It is an adaptive sampling method which uses a generalized BSP tree and allows for minimizing nonlinear scalar fields over complex shaped search spaces like, e.g., the space of rotations. We introduce a new technique for a hierarchical decomposition of the rotation space in disjoint equally sized parts called spherical boxes. Furthermore, a procedure for uniform point sampling from spherical boxes is presented. Tests on a variety of point sets show that the proposed registration method performs very well on noisy, outlier corrupted and incomplete data. For comparison, we report how two state-of-the-art registration algorithms perform on the same data sets.

1. Introduction and related work

Point set registration is a fundamental problem in computational geometry with applications in the fields of computer vision, computer graphics, image processing and many others. The problem can be formulated as follows. Given two finite point sets $\mathbf{M} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subset \mathbb{R}^3$ and $\mathbf{D} = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\} \subset \mathbb{R}^3$ find a mapping $T : \mathbb{R}^3 \to \mathbb{R}^3$ such that the point set $T(\mathbf{D}) = \{T(\mathbf{y}_1), \ldots, T(\mathbf{y}_n)\}$ is optimally aligned in some sense to $\mathbf{M}$. $\mathbf{M}$ is referred to as the model point set (or just the model) and $\mathbf{D}$ is termed the data point set. Points from $\mathbf{M}$ and $\mathbf{D}$ are called model points and data points, respectively.

If T is a rigid transform, i.e., $T(\mathbf{x}) = R\mathbf{x} + \mathbf{t}$ for a rotation matrix R and a translation vector $\mathbf{t}$, we have to solve a rigid point set registration problem. This special case is of major importance for the tasks of object recognition, tracking, localization and mapping, and object modeling, just to name a few. The problem is especially hard when no initial pose estimation is available, the point sets are noisy, corrupted by outliers and incomplete and no correspondences between the points of the input sets are known. In Fig. 1, a model and a data set are shown before and after rigid registration.

1.1. Rigid point set registration

One class of rigid point set registration approaches consists of methods designed to solve the initial pose estimation problem.[1] These methods compute a (more or less) coarse alignment between the point sets without making any assumptions about their initial position and orientation in space. Classic initial pose estimators are the generalized Hough transform [2], geometric hashing [3] and pose clustering [4]. These algorithms are guaranteed to find the optimal alignment between the input point sets. However, because of their high computational cost and/or high memory requirements, these methods are only applicable to small data sets.

Johnson et al. introduced in their work [5] local geometric descriptors, called spin images, and used them for pose estimation and object recognition. The presented results are impressive, but no tests with noisy or outlier corrupted data were performed. Gelfand et al. [6] developed a local descriptor which performs well under artificially created noisy conditions, but still, defining robust local descriptors in the presence of significant noise and a large amount of outliers remains a difficult task.

A more recent approach to the initial pose estimation problem is the robust 4PCS algorithm introduced by Aiger et al. [7]. It is an efficient randomized generate-and-test approach. It selects an appropriate quadruple $\mathbf{B}$ (called a basis) of nearly coplanar points from the model set $\mathbf{M}$ and computes the optimal rigid transform

* Corresponding author.
E-mail addresses: papazov@in.tum.de (C. Papazov), burschka@in.tum.de (D. Burschka).

[1] Pose = position (translation) + orientation (rotation).

1077-3142/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.cviu.2011.05.008

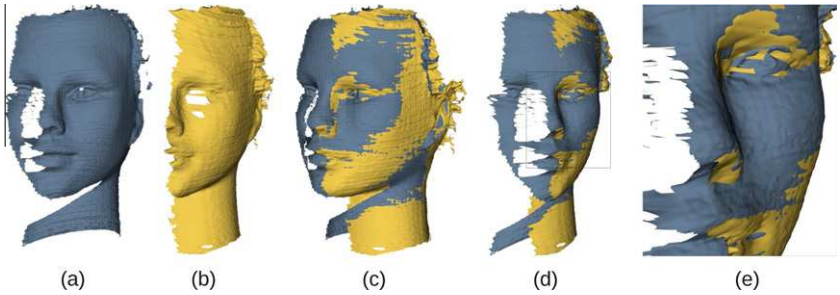


Fig. 1. Pairwise rigid point set registration obtained with our method. The input point sets, model and data, are shown in (a) and (b), respectively. Although rendered as meshes no surface information (like, e.g., normals) is used for the registration. Note that the scans are noisy and only partially overlapping. (c) and (d) Our registration result (shown from two different viewpoints) obtained without noise filtering, local ICP refinement [1] or any assumptions about the initial pose of the input scans. (e) A closer view of the part marked by the rectangle in (d). Observe the high quality of the alignment.

between **B** and each of the potential bases in the data set **D** and chooses the best one. In order to achieve high probability for success, the procedure is repeated several times for different bases $\mathbf{B} \subset \mathbf{M}$. Note, however, that the rigid transform, found by the algorithm, is optimal only for the two bases (i.e., for eight points). In contrast to this, the rigid transform we compute is optimal for all points of the input sets and thus we expect to achieve higher accuracy than the 4PCS algorithm. This is further validated in the experimental results in Section 5 of this paper.

Since the accuracy of the pose computed by the above mentioned methods is insufficient for many applications, an additional pose refinement step needs to be performed. The pose refining algorithms represent another class of registration approaches. The most popular one is the Iterative Closest Point (ICP) algorithm. Since its introduction by Chen and Medioni [8], and Besl and McKay [1], a variety of improvements has been proposed in the literature. A good summary as well as results in acceleration of ICP algorithms have been given by Rusinkiewicz and Levoy [9]. A major drawback of ICP and all its variants is that they assume a good initial guess for the pose of the data point set (with respect to the model). This pose is improved in an iterative fashion until an optimal rigid transform is found. The quality of the solution heavily depends on the initial guess. Furthermore, the methods compared by Rusinkiewicz and Levoy [9] use local surface features like surface normals which cannot be computed very reliably in the presence of noise.

Recently, a variety of registration algorithms based on robust statistics has been proposed. Granger and Pennec [10] formulated the rigid point set registration as a general maximum likelihood estimation problem which they solved using expectation maximization principles. Tsin and Kanade [11] introduced the kernel correlation approach as an extension of the well-known 2D image correlation technique to point sets. The model and data sets are represented by a collection of kernel functions each one centered at a model/data point. If each point in the model set has a close counterpart in the data set the kernel correlation value is large. Thus the registration problem is converted to the maximization of the kernel correlation of the input point sets. An extension of this approach through a Gaussian mixture model was proposed by Jian and Vemuri [12]. Instead of using one-to-one correspondences between the points of the input sets, the above cited methods work with multiple, weighted correspondences. Although this significantly widens the basin of convergence the resulting computational cost limits the applicability of the algorithms to small point sets only [13].

A further class of rigid registration methods is based on particle filtering. Ma and Ellis [14] pioneered the use of the unscented particle filter for registration of surfaces in the context of computer-assisted surgery. A major limitation of the method is its running time: it takes 1.5 s for a data set consisting of 15 points. Moreover, outlier robustness was not addressed by the authors. Further interesting approaches from this class are the algorithm of Moghari and Abolmaesumi [15] which is based on the unscented Kalman filter and the point set registration method via particle filtering and stochastic dynamics introduced by Sandhu et al. [16]. Although these algorithms have a band of convergence significantly wider than the one of local optimizers, they still depend on the initial alignment of the point sets.

1.2. Optimization-based point set registration

Solving the registration problem by minimizing a cost function with a general-purpose optimizer has already been introduced in the literature. Depending on the choice of either a global or a local optimization procedure the corresponding registration approach belongs to the class of initial pose estimators or pose refining methods, respectively.

Breuel [17] used a deterministic branch-and-bound method to globally maximize a quality measure which counts the number of data points a given rigid transform brings within an ϵ-neighborhood of some model point. Although this method always finds the global optimal solution its computational cost seems to be very high since only planar rigid transforms (with three degrees of freedom) were considered.

Olsson et al. [18] also used a deterministic branch-and-bound algorithm to globally minimize the sum of squared distances between corresponding entities (points, lines or planes) in **M** and **D**. This method is guaranteed to find the global optimal solution, however, at a high computational cost: a problem consisting of 10 point-to-plane, 4 point-to-line and 4 point-to-point correspondences is solved in about 10 s. Furthermore, when applied in the case of point set registration, the correspondences between the points have to be known in advance which is seldom the case in a real world setting.

Another deterministic solver based on Lipschitz global optimization theory was introduced by Li and Hartley [19]. On the positive side, the method does not assume any known correspondences across the point sets and it always solves the problem in a globally optimal way. Unfortunately, the algorithm is very costly (about 18 minutes for input sets consisting of 200 points each) and it is based on some unrealistic assumptions: (i) the model and data sets have exactly the same number of points, (ii) there are no outliers and (iii) there is no missing data, i.e., there is a 100% overlap between model and data.

Mitra et al. [20], Pottmann et al. [21] and Fitzgibbon [22] also formulated the registration problem as a minimization of a geometric cost function. For its minimization, however, a local

optimization method is used. This results in the already mentioned strong dependence on a good initial transform estimation.

1.3. Stochastic optimization

Stochastic optimization has received considerable attention in the literature over the last three decades. Much of the work has been devoted to the theory and applications of simulated annealing (SA in the following) as a minimization technique [23–25]. A comprehensive overview of this field is given in [26]. A major property of SA algorithms is their "willingness" to explore regions around points in the search space at which the objective function takes values greater than the current minimum [27]. This is what makes SA algorithms able to escape from local minima and makes them suitable for global minimization. A known drawback of SA algorithms is the fact that they waste a lot of iterations in generating candidate points, evaluating the objective function at these points, and finally rejecting them [26]. In order to reduce the number of rejections, Bilbro and Snyder [28] select candidate points from "promising" regions of the search space, i.e., from regions in which the objective function is likely to have low values. They achieve this by adapting a k-d tree to the objective function each time a new candidate point is accepted. If, however, the current point is rejected, the tree remains unchanged. This is a considerable waste of computation time since the information gained by the (expensive) evaluation of the objective function is not used. In contrast to this, our algorithm adapts a generalized BSP tree at every iteration and thus uses all the information collected during the minimization. Furthermore, the use of a generalized BSP tree allows for a minimization over complex shaped spaces and not only over rectangular regions as in the case of [28].

1.4. Contributions and overview

Our registration algorithm aims to robustly solve the initial pose estimation problem in the case of noisy, outlier corrupted and incomplete point sets with unknown correspondences between the points. Our main contributions are (i) a noise and outlier resistant cost function, (ii) a stochastic approach for its global minimization, (iii) a technique for a hierarchical rotation space decomposition in disjoint parts of equal volume and (iv) a procedure for uniform sampling from spherical boxes. The work presented here is a significant extension of the concept introduced in the conference paper [29].

The rest of the paper is organized as follows. In Section 2, we define the task of aligning two point sets as a nonlinear minimization problem and define our cost function. In Section 3, a stochastic approach for global minimization is presented. In Section 4, we motivate the choice of the rotation space parametrization we use in combination with our minimization approach and introduce a technique for a hierarchical rotation space decomposition. Furthermore, a procedure for uniform sampling from spherical boxes is described. Section 5 presents experimental results obtained with our registration algorithm as well as comparisons with two state-of-the-art registration methods. The paper ends with some conclusions in Section 6.

2. Registration as a minimization problem

Consider, we are given a model point set $\mathbf{M} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subset \mathbb{R}^3$ and a data point set $\mathbf{D} = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\} \subset \mathbb{R}^3$. Suppose, we have a continuous function $S : \mathbb{R}^3 \to \mathbb{R}$, called the model scalar field, which attains small values at the model points $\mathbf{x}_j$, $j \in \{1, \ldots, m\}$ and increases with increasing distance between the evaluation point and the closest model point. Our aim is to find a rigid transform $T : \mathbb{R}^3 \to \mathbb{R}^3$ of the form $T(\mathbf{x}) = R\mathbf{x} + \mathbf{t}$ for a rotation matrix R and a translation vector $\mathbf{t} \in \mathbb{R}^3$ such that the functional

$$\mathcal{F}(T) = \sum_{i=1}^{n} S(T(\mathbf{y}_i)), \quad \mathbf{y}_i \in \mathbf{D} \tag{1}$$

is minimized. This definition of $\mathcal{F}$ is based on the following idea common for most registration algorithms: we seek a rigid transform that brings the data points as close as possible to the model points.

2.1. Definition of the model scalar field

Given the model point set $\mathbf{M} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, we want our model scalar field $S : \mathbb{R}^3 \to \mathbb{R}$ to attain its minimal value at the model points, i.e.,

$$S(\mathbf{x}_j) = s_{\min} \in \mathbb{R}, \quad \forall \mathbf{x}_j \in \mathbf{M}, \tag{2}$$

and to attain greater values for all other points in $\mathbb{R}^3$, i.e.,

$$S(\mathbf{x}) > s_{\min}, \quad \forall \mathbf{x} \in \mathbb{R}^3 \setminus \mathbf{M}. \tag{3}$$

Define

$$d_{\mathbf{M}}(\mathbf{x}) = \min_{\mathbf{x}_j \in \mathbf{M}} \|\mathbf{x} - \mathbf{x}_j\| \tag{4}$$

to be the distance between a point $\mathbf{x} \in \mathbb{R}^3$ and the set $\mathbf{M}$, where $\|\cdot\|$ is the Euclidean norm in $\mathbb{R}^n$. If we set

$$S(\mathbf{x}) = d_{\mathbf{M}}(\mathbf{x}), \tag{5}$$

we get an unsigned distance field which is implicitly used by ICP [1]. It is obvious that this choice for S fulfills both criteria (2) and (3).

Mitra et al. [20] and Pottmann et al. [21] considered in their work more sophisticated scalar fields. They assumed that the model point set $\mathbf{M}$ consists of points sampled from an underlying surface Φ. The scalar field S at a point $\mathbf{x} \in \mathbb{R}^3$ is defined to be the squared distance from $\mathbf{x}$ to Φ. In this context, S is called the squared distance function to the surface Φ. We refer to [20] for details on computing the squared distance function and its approximation for point sets.

The version of S given in (5) and the ones used by Mitra et al. [20] and Pottmann et al. [21] are essentially distance fields. This means that $S(\mathbf{x})$ approaches infinity as the point $\mathbf{x}$ gets infinitely far from the point set. This has the practical consequence that a registration technique which minimizes a cost function based on an unbounded scalar field will be sensitive to outliers in the data set. This is because data points lying far away from the model point set will have great impact on the sum in (1) and thus will prevent the minimization algorithm from converging towards the right alignment. A similar problem arises in the case of model and data sets with low overlap. In this case, there will be a lot of data points which have no corresponding model points and vice versa. The distance between such a data point and the closest model point will be large and thus will deteriorate the sum in (1). A simple way to overcome this is just to exclude data points which are too far away from the model set. However, this strategy introduces discontinuities in the cost function which cause a problem for many optimization methods.

Fitzgibbon presented in his work [22] a more convenient way to alleviate these difficulties which does not lead to a discontinuous cost function. He proposed to use either of the following two robust kernels:

$$S(\mathbf{x}) = \log\left(1 + \frac{(d_{\mathbf{M}}(\mathbf{x}))^2}{\sigma}\right) \text{(Lorentzian kernel)} \quad \text{or} \tag{6}$$

$$S(\mathbf{x}) = \begin{cases} (d_{\mathbf{M}}(\mathbf{x}))^2 & \text{if } d_{\mathbf{M}}(\mathbf{x}) < \sigma \\ 2\sigma d_{\mathbf{M}}(\mathbf{x}) - \sigma^2 & \text{otherwise} \end{cases} \text{(Huber kernel)}. \tag{7}$$

However, we still have $\lim_{d_{\mathbf{M}}(\mathbf{x})\to\infty} S(\mathbf{x}) = \infty$ for both kernels as in the case of (5). Thus a cost function based on (6) or (7) will still be sensitive to outliers. We further validate this in the experimental results presented in Section 5 of the paper.

To avoid this sensitivity, we propose to use a bounded scalar field satisfying (2) and (3) and having the additional property

$$\lim_{d_{\mathbf{M}}(\mathbf{x})\to\infty} S(\mathbf{x}) = 0. \tag{8}$$

We set

$$S(\mathbf{x}) = -\varphi(d_{\mathbf{M}}(\mathbf{x})), \tag{9}$$

where $\varphi : \mathbb{R}^* \to \mathbb{R}^*$, for $\mathbb{R}^* = \{x \in \mathbb{R} : x \geqslant 0\}$, is a strictly monotonically decreasing continuous function with

$$\max_{x\in\mathbb{R}^*} \varphi(x) = \varphi(0) \quad \text{and} \tag{10}$$

$$\lim_{x\to\infty} \varphi(x) = 0. \tag{11}$$

In our implementation, we use an inverse distance kernel of the form

$$\varphi(x) = \frac{1}{1+\alpha x^2}, \quad \alpha > 0 \tag{12}$$

because it is computationally efficient to evaluate and can be controlled by a single parameter α (see Fig. 2a). This results in the following model scalar field:

$$S_\alpha^{\mathbf{M}}(\mathbf{x}) = -\frac{1}{1+\alpha(d_{\mathbf{M}}(\mathbf{x}))^2}, \quad \alpha > 0. \tag{13}$$

It is easy to see that (2), (3) and (8) hold. Different values for α in (13) lead to different scalar fields. The greater the value the faster $S_\alpha^{\mathbf{M}}(\mathbf{x})$ convergences to zero as $d_{\mathbf{M}}(\mathbf{x}) \to \infty$ (see Fig. 2b). In Section 2.2, we will discuss how to choose a suitable value for α and why this particular form of $S_\alpha^{\mathbf{M}}(\mathbf{x})$ leads to an outlier robust cost function.

2.2. Cost function definition

The group of all rigid transforms in $\mathbb{R}^3$ is called the special Euclidean group and is denoted by $SE(3)$. At the beginning of Section 2, we formulated the rigid point set registration problem as a functional minimization problem over $SE(3)$. Using a parametrization of $SE(3)$, the functional $\mathcal{F}$ in (1) can be converted to a real-valued scalar field $F : \mathbb{R}^6 \to \mathbb{R}$ of the form

$$F(\varphi,\psi,\theta,x,y,z) = \sum_{i=1}^{n} S_\alpha^{\mathbf{M}}(R_{\varphi,\psi,\theta}\mathbf{y}_i + (x,y,z)), \tag{14}$$

where $\mathbf{y}_1, \ldots, \mathbf{y}_n$ are the data points, $S_\alpha^{\mathbf{M}}$ is the model scalar field defined in (13), $R_{\varphi,\psi,\theta}$ is a rotation matrix parametrized by φ, ψ, θ and $(x,y,z) \in \mathbb{R}^3$ is a translation vector. In order to achieve good optimization performance, it is very important to choose the right parametrization of the rotation group. We employ an axis-angle based parametrization which is especially well suited for our branch and "stochastic bound" minimization method. Furthermore, we introduce a new technique for a hierarchical decomposition of the rotation space in spherical boxes and describe a procedure for uniform sampling from them. Since the advantages of these techniques are best seen in the context of our minimization algorithm we postpone the detailed discussion to Section 4 after the introduction of the minimization method in Section 3.

A global minimizer $\mathbf{x}^* \in \mathbb{R}^6$ of F defines a rigid transform that brings the data points as close as possible to the model points. What makes the proposed cost function robust to outliers is the fact that outlier data points have a marginal contribution to the sum in (14) depending on α. More precisely, given a positive real number d, we can compute a value for α such that $\left|S_\alpha^{\mathbf{M}}(\mathbf{x})\right|$ is less than an arbitrary $\delta > 0$, if $d_{\mathbf{M}}(\mathbf{x}) > d$ holds. In this way, the contribution of an outlier point to the sum in (14) can be made arbitrary close to zero and F will behave like an outlier rejector. However, too large values for α will lead to the rejection of data points which do not have exact counterparts in a sparsely sampled model set, but still are not outliers. In our implementation we set

$$d = \frac{1}{4}\min\{bbox_x(\mathbf{M}), bbox_y(\mathbf{M}), bbox_z(\mathbf{M})\}, \tag{15}$$

$$\delta = 0.1, \tag{16}$$

where $bbox(\mathbf{M})$ denotes the bounding box of the model point set and $bbox_s(\mathbf{M})$, $s \in \{x,y,z\}$ is the extent of the bounding box along the x, y or z axis. Using the absolute value of the right side of (13) and solving for α yields

$$\alpha = \frac{1-\delta}{\delta d^2}. \tag{17}$$

The cost function given in (14) is nonconvex and has multiple local minima over the search space (see [19] where this is experimentally verified for a similar cost function). Using a local optimization procedure—common for many registration methods—will lead in most cases to a local minimizer of F and thus will not give the best alignment between model and data. To avoid this, we employ a new stochastic approach for global minimization described in the next Section.

3. Stochastic adaptive search for global minimization

Our stochastic minimization approach is inspired by the simulated annealing (SA) method of Bilbro and Snyder [28]. The main difference between their work and a typical SA algorithm is the way how the minimizer candidates are generated. As we already mentioned in Section 1.3, SA algorithms are known to waste many iterations in sampling candidate points from the search space, evaluating the cost function at these points and finally rejecting them [26]. In order to reduce the number of rejections, Bilbro

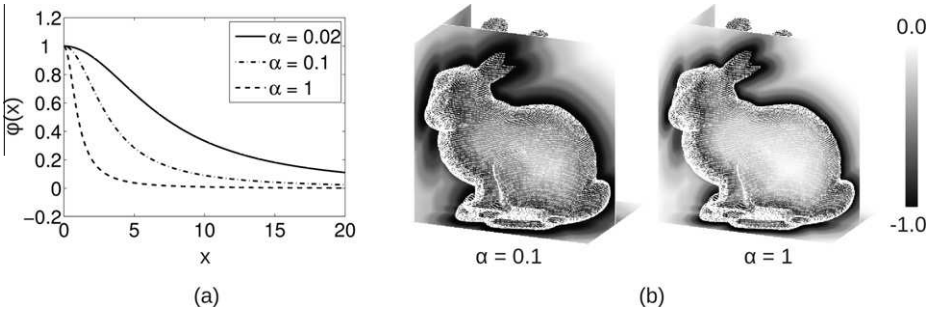


Fig. 2. (a) The inverse distance kernel (defined in (12)) for three different α values. (b) The model scalar field $S_\alpha^{\mathbf{M}}(\mathbf{x})$ (defined in (13)) based on the inverse distance kernel from (a) for $\alpha = 0.1$ and $\alpha = 1$. In this example, the Stanford bunny is used as the model set. $S_\alpha^{\mathbf{M}}(\mathbf{x})$ is visualized by evaluating it at a number of points lying on the three planes and mapping the scalar values to gray levels.

and Snyder [28] sampled the points from a distribution which is modified iteratively during the minimization such that its modes are built around minimizers of the cost function. They achieved this by building a k-d tree and sampling the candidates from those leaves of the tree which cover "promising" regions of the search space, i.e., regions in which the cost function is likely to attain low values. Although this leads to fewer candidate rejections and thus saves computation time the method in [28] still has two drawbacks. First, the candidate points are sampled directly from the tree leaves which are n-dimensional boxes of the form $[a_1,b_1]\times\cdots\times[a_n,b_n]$, where $[a_i,b_i]\subset\mathbb{R}$ is a closed interval. This strategy is based on the implicit assumption that the search space can be covered efficiently by such boxes. This, however, is not the case if we have a more complex shaped space, e.g., the space of rotations (see Section 4). Second, the k-d tree used in [28] is updated only if the generated candidate is accepted. In the case of a rejection, the tree remains unchanged. This is a waste of computation time since the information gained by the expensive cost function evaluation is not used.

We account for the first drawback by formulating our minimization algorithm using a more general spatial data structure, namely, a generalized binary space partitioning tree (we will call it a G-BSP tree in the following). As opposed to the classic BSP trees (see, e.g., [30]), we do not require that the subspaces represented by the tree nodes are convex sets. Thus we can minimize efficiently over more complex shaped search spaces like, e.g., the space of rotations (see Section 4). To avoid the second drawback, i.e., to use all the information gained by the cost function evaluation, we update the tree at every iteration—even in the cases of bad minimizer candidates. This apparently minor modification leads to a rather different algorithm (than [28]) and enables a faster rejection of the regions in which the cost function is likely to have high (i.e., poor) values and thus speeds up the convergence.

3.1. Generalized BSP trees

A binary space partitioning tree (BSP tree) is a spatial data structure which decomposes the real space $\mathbb{R}^n$ in a hierarchical manner. At each subdivision stage, the space is subdivided by a (hyper)plane in two disjoint parts of arbitrary size. Thus the resulting decomposition consists of arbitrarily shaped convex polygons [30]. Each node of the tree has exactly two or zero child nodes. A node with zero children is called a leaf. If we drop the assumption that the space subdivision is performed by planes we get a generalized BSP tree (G-BSP tree). This results in a decomposition made up of subspaces of arbitrary shape.

3.2. Problem definition

Given a set $\mathbf{X}$ (called the search space) and a function $f:\mathbf{X}\to\mathbb{R}$ our aim is to find a global minimizer of f, i.e., an $\mathbf{x}^*\in\mathbf{X}$ such that

$$f(\mathbf{x}^*)\leqslant f(\mathbf{x})\quad\forall\mathbf{x}\in\mathbf{X}. \tag{18}$$

The following assumptions about $\mathbf{X}$ should hold:

- $\mathbf{X}\subset\mathbb{R}^n$ is a bounded set of positive volume (Lebesgue measure in $\mathbb{R}^n$).
- There is an algorithm of acceptable complexity which can build a G-BSP tree for $\mathbf{X}$ such that each two subsets of $\mathbf{X}$ at the same level of the tree are of equal volume (have the same Lebesgue measure in $\mathbb{R}^n$).
- $\mathbf{X}$ is simple enough for sampling algorithms of acceptable complexity to be able to sample uniformly from the G-BSP tree nodes, i.e., from the subsets of $\mathbf{X}$ represented in the G-BSP tree.

Furthermore, the cost function f is required to be bounded and defined at each $\mathbf{x}\in\mathbf{X}$.

3.3. Overall algorithm description

We use a G-BSP tree to represent the n-dimensional search space $\mathbf{X}$. The root η_0^0 is at the 0th level of the tree and represents the whole space $\mathbf{X}_0=\mathbf{X}$. η_0^0 has two children, η_{00}^1 and η_{01}^1, which are at the next level. They represent the subsets $\mathbf{X}_{00}$ and $\mathbf{X}_{01}$, respectively, which are disjoint, have equal volume and their union equals $\mathbf{X}_0$. In general, a node η_s^k (where $k\geqslant 0$ and s is a binary string of length $k+1$) is at the kth level of the tree and has two children, η_{s0}^{k+1} and η_{s1}^{k+1}, which are at the next, $(k+1)$th, level. The volume of η_s^k is $1/2^k$ of the volume of $\mathbf{X}$. This concept is easily visualized in the case $n=2$ and $\mathbf{X}$ and its subsets being rectangles (see Fig. 3a).

During the minimization, the G-BSP tree is built in an iterative fashion beginning at the root. The algorithm adds more resolution to promising regions in the search space, i.e., the tree is built with greater detail in the vicinity of points in $\mathbf{X}$ at which the objective function attains low values. The overall procedure can be outlined as follows:

1. Initialize the tree (see Section 3.4) and set an iteration counter $j=0$.
2. Select a "promising" leaf according to a probabilistic selection scheme (see Section 3.5).
3. Expand the tree by bisecting the selected leaf. This results in the creation of two new child nodes. Evaluate the objective function at a point which is uniformly sampled from the subset of one of the two children (see Section 3.6).
4. If a stopping criterion is not met, increment the iteration counter j and go to step 2, otherwise terminate the algorithm (see Section 3.7).

3.4. Initializing the tree

For every tree node η_s^k the following items are stored: (i) a set $\mathbf{X}_s\subset\mathbf{X}$ and (ii) a pair $(\mathbf{x}_s,f(\mathbf{x}_s))$ consisting of a point $\mathbf{x}_s$, uniformly sampled from $\mathbf{X}_s$, and the corresponding function value $f(\mathbf{x}_s)$. The tree is initialized by storing the whole search space $\mathbf{X}$ and a pair $(\mathbf{x}_0,f(\mathbf{x}_0))$ in the root.

3.5. Selecting a leaf

At every iteration, the search for a global minimizer begins at the root and proceeds down the tree until a leaf is reached. In order to reach a leaf, we have to choose a concrete path from the root down to this leaf. At each node, we have to decide whether to take its left or right child as the next station. This decision is made probabilistically. For every node, two numbers p_0, $p_1\in(0,1)$ are computed such that $p_0+p_1=1$. Arriving at a node, we choose to descend via either its left or right child with probability p_0 or p_1, respectively. We make these left/right decisions until we reach a leaf.

Computing the probabilities p_0 and p_1. The idea is to compute the probabilities in a way such that the "better" child, i.e., the one with the lower function value, has greater chance to be selected. We compute p_0 and p_1 for each node η_s^k based on the function values associated with its children η_{s0}^{k+1} and η_{s1}^{k+1}. Let f_{s0} and f_{s1} be the function values associated with η_{s0}^{k+1} and η_{s1}^{k+1}, respectively. The following criterion should be fulfilled:

$$f_{s0}<f_{s1}\quad\Longleftrightarrow\quad p_0>p_1. \tag{19}$$

If $f_{s0}<f_{s1}$ we set

$$p_0=(t+1)/(1+2t),\quad p_1=t/(1+2t), \tag{20}$$

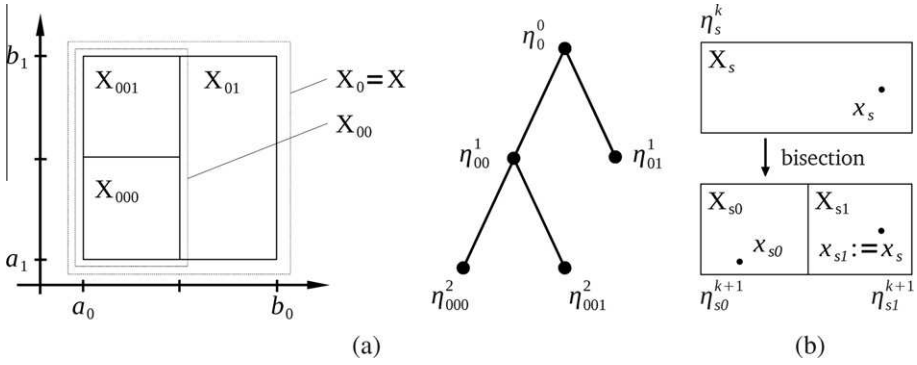


Fig. 3. (a) An example of a two-dimensional G-BSP tree and a rectangular search space **X**. In this case, the G-BSP tree is a two-dimensional k-d tree. (b) Expanding the leaf η_s^k. In this example, after the bisection of η_s^k, the point $\mathbf{x}_s$ lies in the box $\mathbf{X}_{s1}$, hence η_{s1}^{k+1} adopts the pair $(\mathbf{x}_s, f(\mathbf{x}_s))$ from η_s^k. For the other child, η_{s0}^{k+1}, a point $\mathbf{x}_{s0}$ is sampled uniformly from $\mathbf{X}_{s0}$ and the objective function is evaluated at that point.

for a parameter $t \geqslant 0$. For $t \to \infty$ we get $p_0 = p_1 = \frac{1}{2}$ and our minimization algorithm becomes a pure random search. Setting $t = 0$ results in $p_0 = 1$ and $p_1 = 0$ and makes the algorithm deterministically choosing the "better" child of every node which leads to the exclusion of a large portion of the search space and in most cases prevents the algorithm from finding a global minimizer. For $f_{s1} < f_{s0}$ we set

$$p_0 = t/(1+2t), \quad p_1 = (t+1)/(1+2t). \tag{21}$$

Updating the probabilities. From the discussion above it becomes evident that t should be chosen from the interval $(0, \infty)$. For our algorithm the parameter t plays a similar role as the temperature parameter for a simulated annealing algorithm [23] so we will refer to t as temperature as well. Like in simulated annealing, the search begins at a high temperature level (large t) such that the algorithm samples the search space quite uniformly. The temperature is decreased gradually during the minimization process so that promising regions of the search space are explored in greater detail. More precisely, we update t according to the following cooling schedule:

$$t = t_{\max} \exp(-vj), \tag{22}$$

where $j \in \mathbb{N}$ is the current iteration number, $t_{\max} > 0$ is the temperature at the beginning of the search (for $j = 0$) and $v > 0$ is the cooling speed which determines how fast the temperature decreases.

3.6. Expanding the tree

After reaching a leaf η_s^k, the set $\mathbf{X}_s$ associated with it gets bisected in two disjoint subsets $\mathbf{X}_{s0}$ and $\mathbf{X}_{s1}$ of equal volume. The corresponding child nodes are η_{s0}^{k+1} and η_{s1}^{k+1}, respectively. In this way, we add more resolution in this part of the search space. Next, we evaluate the new children, i.e., we assign to the left and right one a pair $(\mathbf{x}_{s0}, f(\mathbf{x}_{s0}))$ and $(\mathbf{x}_{s1}, f(\mathbf{x}_{s1}))$, respectively.

Note that the parent of η_{s0}^{k+1} and η_{s1}^{k+1}, namely, the node η_s^k, stores a pair $(\mathbf{x}_s, f(\mathbf{x}_s))$. Since $\mathbf{X}_s = \mathbf{X}_{s0} \cup \mathbf{X}_{s1}$ and $\mathbf{X}_{s0} \cap \mathbf{X}_{s1} = \emptyset$ it follows that $\mathbf{x}_s$ is contained either in $\mathbf{X}_{s0}$ or in $\mathbf{X}_{s1}$. Thus we set

$$(\mathbf{x}_{s0}, f(\mathbf{x}_{s0})) = (\mathbf{x}_s, f(\mathbf{x}_s)) \quad \text{if } \mathbf{x}_s \in \mathbf{X}_{s0} \quad \text{or} \tag{23}$$

$$(\mathbf{x}_{s1}, f(\mathbf{x}_{s1})) = (\mathbf{x}_s, f(\mathbf{x}_s)) \quad \text{if } \mathbf{x}_s \in \mathbf{X}_{s1}. \tag{24}$$

To compute the other pair, we sample a point uniformly from the appropriate remaining set ($\mathbf{X}_{s0}$ or $\mathbf{X}_{s1}$) and evaluate the function at this point (see Fig. 3b for the case $n = 2$ and **X** and its subsets being rectangles).

Updating the tree. During the search we want to compute the random paths from the root down to a certain leaf such that promising regions—leaves with low function values—are visited more often than non-promising ones. Thus, after evaluating a new created leaf, we propagate its (possibly very low) function value as close as possible to the root. This is done by the following updating procedure. Suppose that the parent point $\mathbf{x}_s$ is contained in the set $\mathbf{X}_{s1}$ belonging to the new created child η_{s1}^{k+1}. Therefore, we randomly generate $\mathbf{x}_{s0} \in \mathbf{X}_{s0}$, compute $f(\mathbf{x}_{s0})$ and assign the pair $(\mathbf{x}_{s0}, f(\mathbf{x}_{s0}))$ to the child η_{s0}^{k+1}. Updating the tree consists of ascending from η_{s0}^{k+1} (via its ancestors) to the root and comparing at every parent node η_u^j the function value $f(\mathbf{x}_{s0})$ with the function value of η_u^j, i.e., with $f(\mathbf{x}_u)$. If $f(\mathbf{x}_{s0}) < f(\mathbf{x}_u)$ we update the current node by setting $(\mathbf{x}_u, f(\mathbf{x}_u)) = (\mathbf{x}_{s0}, f(\mathbf{x}_{s0}))$ and proceed to the parent of η_u^j. The updating procedure terminates if we reach the root or no improvement for the current node is possible.

Note that if $f(\mathbf{x}_{s0})$ is the lowest function value found so far, it will be propagated to the root, otherwise it will be propagated only to a certain level $l \in \{1, \ldots, k+1\}$. This means, that every node contains the minimum function value (and the point at which f takes this value) found in the subset associated with this node. Since the root represents the whole search space, it contains the point we are interested in, namely, the point at which f takes the lowest value found up to the current iteration.

3.7. Stopping rule

We break the search if the following two criteria are fulfilled. (i) The leaf η_s^k selected in the current iteration has a volume which is smaller than a user predefined value $\delta_v > 0$. (ii) The absolute difference between the minimal function value found so far and the function value computed in the current iteration is less than a user specified $\delta_f > 0$.

The first condition accounts for the desired precision of the solution and the second one assures that the algorithm makes no significant progress any more.

3.8. Remark

We want to emphasize that it is very important that each two nodes at the same tree level are of equal volume. Note that the points are uniformly sampled within the tree nodes (see Section 3.2). In this case, if two differently sized nodes at the same tree level are selected equally often, the part of the search space represented by the smaller node will be sampled more densely than the other part. Thus, the algorithm will possibly prefer parts of the search space only because the G-BSP tree is constructed in a particular way and not because of the cost function.

4. Processing in the space of rigid transforms

As already mentioned in Section 2.2, the choice of a parametrization of $SE(3)$ (the group of rigid transforms) is an important issue since different parametrizations lead to different optimization

performance. We decompose $SE(3)$ into a translational and a rotational part. While parametrizing translations is straightforward special care is needed when dealing with rotations since the geometry of the rotation space is more complex than the geometry of $\mathbb{R}^3$. In the following, we concentrate on the rotation space.

In view of our branch and "stochastic bound" minimization method, three specific problems have to be solved. (i) We need to parametrize rotations. (ii) We have to hierarchically decompose the rotation space in disjoint parts of equal volume. In other words, a G-BSP tree has to be computed in which the nodes are representing equally sized parts of the rotation space. (iii) We need to sample points (i.e., rotations) uniformly from each leaf of the G-BSP tree. These issues are discussed separately in the next three subsections.

4.1. Parametrizing rotations

There are many ways how to parametrize 3D rotations. Discussing all of them is far beyond the scope of this paper. An excellent introduction to this topic is included in the books by Kanatani [31] and Watt and Watt [32] in the context of computer vision and computer graphics, respectively. The set of all 3×3 rotation matrices is a group (under matrix multiplication) which is referred to as $SO(3)$. A parametrization of $SO(3)$ is a mapping R: $\mathbf{U} \rightarrow SO(3)$, where $\mathbf{U}$ is a subset of $\mathbb{R}^3$ since every rotation has three degrees of freedom.

Parametrizing rotation matrices using Euler angles is probably the most widely used technique which is, however, inefficient in conjunction with our minimization method. This is due to the fact that Euler angles are a redundant representation of rotations. In order to represent all elements in $SO(3)$ the following range, $\mathbf{E}$, for the three Euler angles is needed: $\mathbf{E} = [0, 2\pi) \times [0, 2\pi) \times [0, \pi]$. However, the corresponding parametrization R: $\mathbf{E} \rightarrow SO(3)$, which is given in [31], is not one-to-one. There are infinitely many combinations of Euler angles (within the range $\mathbf{E}$) which lead to the same rotation matrix (see [32]). A minimization method like ours which considers the whole search space will waste computation time exploring regions in $\mathbf{E}$ which should be completely ignored because they do not lead to "new" rotation matrices. The same applies to deterministic branch-and-bound methods (see, e.g., [33]).

In order to avoid this difficulty, we employ a redundant-free rotation space parametrization based on the axis-angle representation of $SO(3)$. According to Euler's theorem (see [31]), each rotation in $\mathbb{R}^3$ can be represented by an axis specified by a unit vector $\mathbf{n}$ and an angle θ of rotation around it. $\mathbf{n}$ can itself be parametrized using spherical coordinates φ and ψ:

$$\mathbf{n} = (\sin(\psi)\cos(\varphi), \quad \sin(\psi)\sin(\varphi), \quad \cos(\psi)). \tag{25}$$

Fig. 4a visualizes this concept. In order to represent all rotation matrices, we need to consider the following range for the spherical coordinates (φ, ψ) and the rotation angle θ:

$$(\varphi, \psi, \theta) \in [0, 2\pi) \times [0, \pi] \times [0, \pi) = \mathbf{A}. \tag{26}$$

The parametrization R: $\mathbf{A} \rightarrow SO(3)$, which can be found in [31], is a one-to-one mapping between $\mathbf{A}$ and $SO(3)$.

4.2. Hierarchical decomposition of the rotation space

According to the axis-angle representation and to (26), it is possible to express the set of rotations by the open ball in $\mathbb{R}^3$ with radius π which we will denote by $\mathbf{B}^3(\pi)$ (see Fig. 4b). Thus a straightforward way to decompose the rotation space is to enclose $\mathbf{B}^3(\pi)$ in the cube $\mathbf{C}^3(\pi) = [-\pi, \pi]^3$ and to divide $\mathbf{C}^3(\pi)$ into smaller cubes by simply bisecting the x, y or z axis. Hartley and Kahl [33] used this technique in conjunction with a deterministic branch-and-bound minimization method to estimate the essential matrix and to solve the relative camera pose problem. However, if combined with our minimization algorithm, this technique leads to two problems. First, the sub-cubes of $\mathbf{C}^3(\pi)$ which do not lie within $\mathbf{B}^3(\pi)$ have to be ignored since the rotations they represent are included in other cubes within $\mathbf{B}^3(\pi)$. This gives rise to nodes in the corresponding G-BSP tree which have only one "legal" child. Second, the sub-cubes of $\mathbf{C}^3(\pi)$ which are partially intersecting $\mathbf{B}^3(\pi)$ represent a smaller region of the rotation space than sub-cubes at the same tree level which are fully enclosed in $\mathbf{B}^3(\pi)$. Thus the minimization algorithm will prefer rotations which are close to the boundary of $\mathbf{B}^3(\pi)$.

We solve these two problems by changing the shape of the building blocks of the decomposition. Since we are dealing with a three-dimensional ball the most natural shape is the shape of a spherical box (see Fig. 4b). In ball coordinates, we define a spherical box $\mathbf{S}^3$ to be a point set of the form

$$\mathbf{S}^3 = \{(\varphi, \psi, \theta) : (\varphi, \psi, \theta) \in [\varphi_1, \varphi_2) \times [\psi_1, \psi_2) \times [\theta_1, \theta_2)\}, \tag{27}$$

where $[\varphi_1, \varphi_2) \times [\psi_1, \psi_2)$ is the range of the spherical coordinates and $[\theta_1, \theta_2)$ limits the distance of the points to the origin. Decomposing the rotation space means to hierarchically subdivide $\mathbf{B}^3(\pi)$ into disjoint spherical boxes of equal volume (see Fig. 5). Note that the volume of $\mathbf{S}^3$ is given by

$$vol_{\mathbf{S}^3}(\varphi_1, \varphi_2, \psi_1, \psi_2, \theta_1, \theta_2) = \int_{\varphi_1}^{\varphi_2} \int_{\psi_1}^{\psi_2} \int_{\theta_1}^{\theta_2} \theta^2 \sin\psi \, d\theta \, d\psi \, d\varphi \tag{28}$$

$$= (\varphi_2 - \varphi_1)(\cos\psi_1 - \cos\psi_2)\frac{\theta_2^3 - \theta_1^3}{3}. \tag{29}$$

Our aim is to consecutively cut $\mathbf{S}^3$ along the φ, ψ or θ axis such that the resulting pieces have the same volume. Since $vol_{\mathbf{S}^3}$ depends in a different way from each of the ball coordinates φ, ψ and θ we get a different rule for cutting along each axis. We are looking for

$$\varphi \in (\varphi_1, \varphi_2), \quad \psi \in (\psi_1, \psi_2), \quad \theta \in (\theta_1, \theta_2) \tag{30}$$

such that

$$vol_{\mathbf{S}^3}(\varphi_1, \varphi) = vol_{\mathbf{S}^3}(\varphi, \varphi_2), \tag{31}$$

$$vol_{\mathbf{S}^3}(\psi_1, \psi) = vol_{\mathbf{S}^3}(\psi, \psi_2), \tag{32}$$

$$vol_{\mathbf{S}^3}(\theta_1, \theta) = vol_{\mathbf{S}^3}(\theta, \theta_2), \tag{33}$$

where, for the sake of clarity, $vol_{\mathbf{S}^3}$ is expressed as a function of two variables only, namely, the ones defining the interval which is currently cut. Using (29) to solve the Eqs. (31)–(33) leads to

$$\varphi = \frac{\varphi_1 + \varphi_2}{2}, \quad \psi = \arccos\left(\frac{\cos\psi_1 + \cos\psi_2}{2}\right),$$

$$\theta = \sqrt[3]{\frac{\theta_1^3 + \theta_2^3}{2}}. \tag{34}$$

Thus we fully specified how to hierarchically decompose the space of rotations in disjoint equally sized parts such that a G-BSP tree can be built. Furthermore, the shape of the parts is optimally tailored to our minimization algorithm.

4.3. Uniform sampling from spherical boxes

Our method for sampling points uniformly from a spherical box is grounded on the following basic result from Statistics called the inverse probability integral transform. Since it is proved in many textbooks (like, e.g., in [34]) we state it here without a proof.

Theorem 1. *Let F be a cumulative distribution function (c.d.f.) on $\mathbb{R}$ and let U be a random variable uniformly distributed in $[0, 1]$. Then the random variable $X = F(U)^{-1}$ has c.d.f. F.*

Based on this result we perform the uniform sampling from a spherical box $\mathbf{S}^3 = [\varphi_1, \varphi_2) \times [\psi_1, \psi_2) \times [\theta_1, \theta_2)$ in three steps:

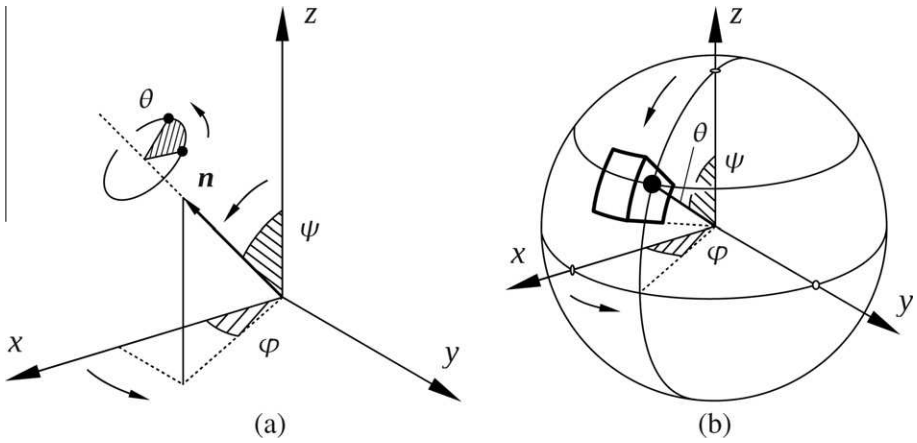


Fig. 4. (a) The axis-angle based parametrization of $SO(3)$. The two bold dots in the figure represent a point before and after rotation by the angle θ around the axis defined by the unit vector $\mathbf{n}$, which is itself parametrized using spherical coordinates (φ,ψ). (b) The rotation space represented as the open ball in $\mathbb{R}^3$ with radius π. The spherical coordinates (φ,ψ) of the point (shown as a bold dot) define the rotation axis and the distance to the origin gives the angle of rotation θ. The bold lines depict a spherical box.

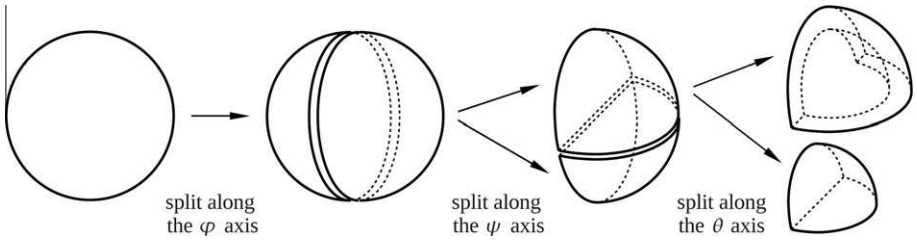


Fig. 5. Decomposing the rotation space (represented as $\mathbf{B}^3(\pi)$) into spherical boxes of equal volume. In this example, only one spherical box at each splitting step is further decomposed.

1. Sample a φ uniformly from $[\varphi_1,\varphi_2)$.
2. Sample a ψ from $[\psi_1,\psi_2)$ according to a c.d.f. F_2 such that the point in $\mathbb{R}^3$ with spherical coordinates (φ,ψ) is uniformly distributed on the spherical patch $\mathbf{S}^2 = [\varphi_1,\varphi_2)\times[\psi_1,\psi_2)$.
3. Sample a θ from $[\theta_1,\theta_2)$ according to a c.d.f. F_3 such that the point in $\mathbb{R}^3$ with ball coordinates (φ,ψ,θ) is uniformly distributed in the spherical box $\mathbf{S}^3$.

Step 1 is easy to perform. In step 2, we need to compute the area of a spherical patch (of the unit 2-sphere) as a function of an interval $[\varphi_1,\varphi_2)\times[\psi_1,\psi_2)$:

$$area_{\mathbf{S}^2}(\varphi_1,\varphi_2,\psi_1,\psi_2) = \int_{\varphi_1}^{\varphi_2}\int_{\psi_1}^{\psi_2} \sin\psi\, d\psi\, d\varphi \tag{35}$$

$$= (\varphi_2-\varphi_1)(\cos\psi_1-\cos\psi_2). \tag{36}$$

Thus the c.d.f. we need in step 2 is given by

$$F_2(\psi) = \frac{area_{\mathbf{S}^2}(\varphi_1,\varphi_2,\psi_1,\psi)}{area_{\mathbf{S}^2}(\varphi_1,\varphi_2,\psi_1,\psi_2)} \tag{37}$$

$$= \frac{\cos\psi_1-\cos\psi}{\cos\psi_1-\cos\psi_2}, \tag{38}$$

Analogously, we see that the c.d.f. in step 3 is given by

$$F_3(\theta) = \frac{vol_{\mathbf{S}^3}(\varphi_1,\varphi_2,\psi_1,\psi_2,\theta_1,\theta)}{vol_{\mathbf{S}^3}(\varphi_1,\varphi_2,\psi_1,\psi_2,\theta_1,\theta_2)} \tag{39}$$

$$= \frac{\theta^3-\theta_1^3}{\theta_2^3-\theta_1^3}, \tag{40}$$

where (40) follows from (29). Note that both F_2 and F_3 can easily be inverted and we can use Theorem 1 to sample according to F_2 and F_3 and hence uniformly from the spherical box $\mathbf{S}^3$.

4.4. Computing the search space and the G-BSP tree

Now since all details regarding the parametrization and decomposition of $SO(3)$ and the sampling from spherical boxes are given, we define the search space $\mathbf{X}$ and specify how to build the corresponding G-BSP tree. We set

$$\mathbf{X} = \mathbf{A}\times bbox(\mathbf{M}), \tag{41}$$

where $\mathbf{A}$ is, according to (26), the domain of the axis-angle based parametrization of $SO(3)$ and $bbox(\mathbf{M})$ (the bounding box of the model $\mathbf{M}$) represents the translational part of the search space. Since $bbox(\mathbf{M})$ is a rectangular box of the form $[x_1,x_2]\times[y_1,y_2]\times[z_1,z_2]\subset\mathbb{R}^3$ it can easily be broken up into smaller boxes of the same size by simply bisecting it along the x, y or z axis.

The root η_0^0 of the G-BSP tree represents the whole set $\mathbf{X}$. The child nodes of the root, namely, η_{00}^1 and η_{01}^1, represent the subsets $\mathbf{X}_0$ and $\mathbf{X}_1$, respectively, resulting from cutting the 0th interval of $\mathbf{X}$—which is $[0,2\pi)$ in (26)—using the rule $(34)_1$. In general, a node η_s^k (where $k\geqslant 0$ and s is a binary string of length $k+1$) is at the kth level of the tree, represents a subset $\mathbf{X}_s$ of the 6D search space and has two children, η_{s0}^k and η_{s1}^k. The child nodes represent the sets $\mathbf{X}_{s0}$ and $\mathbf{X}_{s1}$, respectively, which are computed by cutting the (k mod 6)th interval of $\mathbf{X}_s$ according to (34) if $0\leqslant k \bmod 6\leqslant 2$ (rotational part) or by dividing it in the middle if $3\leqslant k \bmod 6\leqslant 5$ (translational part).

5. Experimental results

In this Section, we test our registration method on a variety of point sets. All tests presented in the paper are performed on a laptop with a 3GHz CPU and 4GB RAM running a Linux operating

Table 1
The parameter values used in all experiments in this paper. The value of δ_v equals the volume of a spherical box with side one degree times the volume of a box with sides equal to one percent of the sides of the bounding box of the model point set.

	Parameter	Defined in	Value
Cost function	d	Eq. (15)	1/4 (min bbox side(**M**))
	δ	Eq. (16)	0.1
Cooling schedule	t_{max}	Eq. (22)	50.0
	v	Eq. (22)	0.00008
Stopping rule	δ_v	Section 3.7	1° rot. and 1% transl.
	δ_f	Section 3.7	0.1

system. The algorithm is implemented in C++. The parameter values used in all experiments presented here are given in Table 1.

Since our method is a probabilistic one, it computes each time a (slightly) different result. In order to make a statistical meaningful statement about its performance, we run 100 registration trials for each pair of inputs and report the mean performance values. We measure the success rate and the accuracy under varying amount of noise and outliers in the input sets. The success rate gives the percentage of registration trials in which a transform which is close to the global optimal one is found. The accuracy is measured using the RMS error (see [6]). The type of noise added to some of the model and data sets is Gaussian and the outliers are simulated by drawing points from a uniform distribution within the bounding box of the corresponding point set. We report the number of outliers as percentage of the original number of points and not as percentage of the points in the corrupted set. For example, 100% means that there are so many outliers in the corrupted point set as there are points in the outlier-free set. We did it so because the results in [7], which we use for comparison, are reported in this way.

We also measure the number of cost function evaluations and the computation time for varying cooling speed v (defined in (22)). We analyze the robustness of our method using two different kernels in the cost function. Furthermore, we report how two state-of-the-art registration approaches perform on the same point

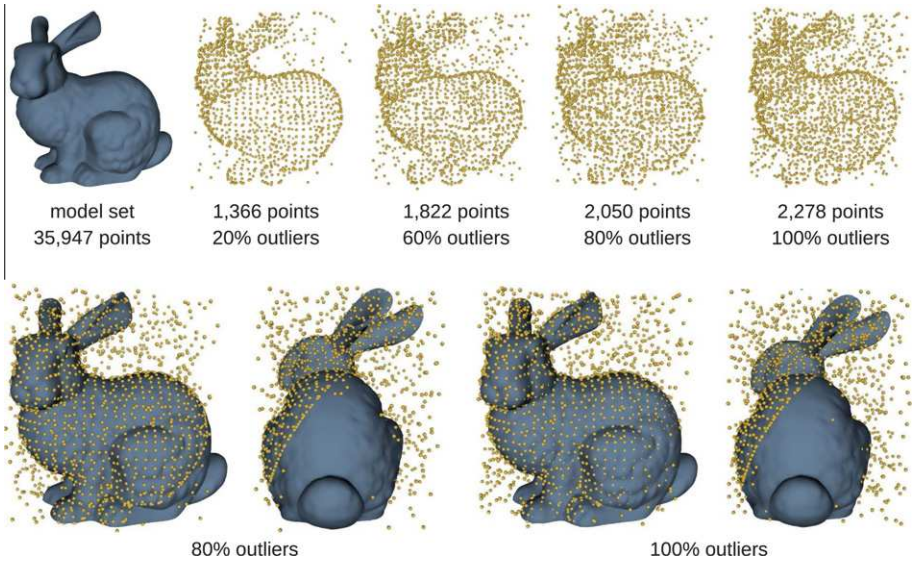


Fig. 6. (Top row) The model set is shown as a blue mesh (note that only the mesh vertices are used for the registration). The outlier corrupted data sets are rendered as yellow point clouds. The size of each point set and the number of outliers as percentage of the original number of input points are shown below each figure. Further note that the data sets are incomplete and sparsely sampled compared to the model. (Bottom row) Typical registration results obtained with our algorithm using the model scalar field (13) based on the inverse distance kernel (12). Observe the high quality of the alignment even in the presence of a significant amount of outliers. A registration trial took between 9 and 17 s (depending on the number of points). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

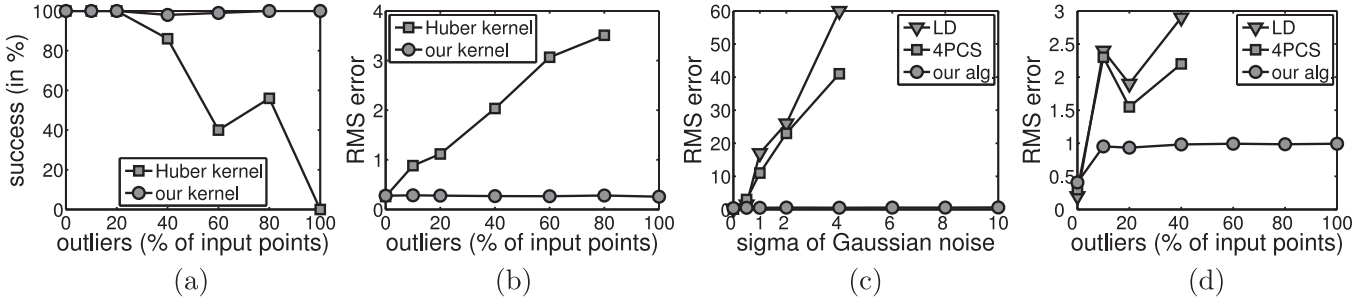


Fig. 7. (a) The success rate as a function of the percentage of outliers in the data sets shown in Fig. 6. The success rate of the registration is shown when using the inverse distance kernel (12) (our kernel) and the Huber kernel (7). Note that our kernel leads to an almost constant success rate of 100% even in the presence of a very large amount of outliers whereas at the level of 100% outliers the registration completely fails if the Huber kernel is used. (b) The RMS error between the ground truth pose for each data set and the estimated pose is shown as a function of the percentage of outliers. Only the successful trials are used for computing the RMS error. Note that our kernel leads to much more precise registration results which are almost independent of the amount of outliers. (c) and (d) We compare our method with the robust 4PCS algorithm [7] and a local descriptor based approach (LD). A combination of a spin-image based descriptor and integral invariants are used as local descriptors (see [7]). Note that the graphs corresponding to LD and 4PCS end by $\sigma = 4.0$ and 40% outliers. This is because the authors in [7] did not test their methods on point sets with more noise or outliers whereas we did. Observe that our algorithm is quite insensitive to noise and outliers and it outperforms both other methods. The alignment error is measured using the RMS error between the model and the data after registration. One unit corresponds to 1% of the bounding box diagonal length of the model set.

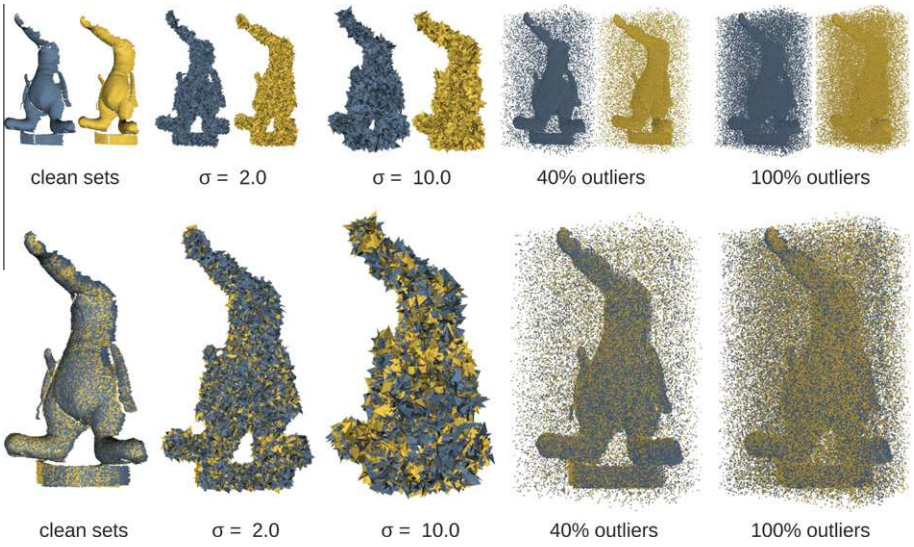


Fig. 8. Registration of partially overlapping noisy and outlier corrupted point sets. The models are shown in blue whereas the data sets in yellow. (Top row) Partial scans of the Coati model degraded by noise or outliers. The σ of the Gaussian noise or the amount of outliers as percentage of the original number of input points is indicated below each figure. One σ unit equals 1% of the bounding box diagonal length of the corresponding point set. (Bottom row) Typical registration results computed with our algorithm. 5,000 (randomly sampled) points from each point set are used for the registration. The results are obtained without any noise or outlier removal, ICP refinement [1] or assumptions about the initial pose of the point sets. Each registration trial took about 33 s. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

#points	40	60	80	100	150	200
time(sec) b &b[19]	10.2	48.6	107.5	156.2	722.1	1103.2
time(sec) our alg.	2.0	2.1	2.3	2.4	2.8	3.1

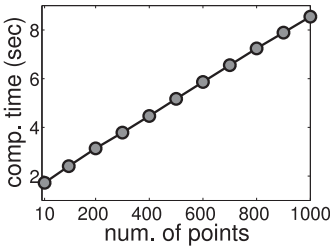


Fig. 9. (Left) Computation time comparison between our algorithm and the box-and-ball (b&b) registration algorithm of Li and Hartley [19] which is based on global deterministic Lipschitz optimization theory. The processing time is given in seconds. In the case of 200 input points, our algorithm outperforms [19] by three orders of magnitude. (Right) Runtime of our algorithm as a function of the number of input points. The figure clearly indicates a linear time complexity. Model and data used in this test case are downsampled copies of the outlier-free version of the data set shown in the top row of Fig. 6. In all tests, our method achieved a success rate of 100%.

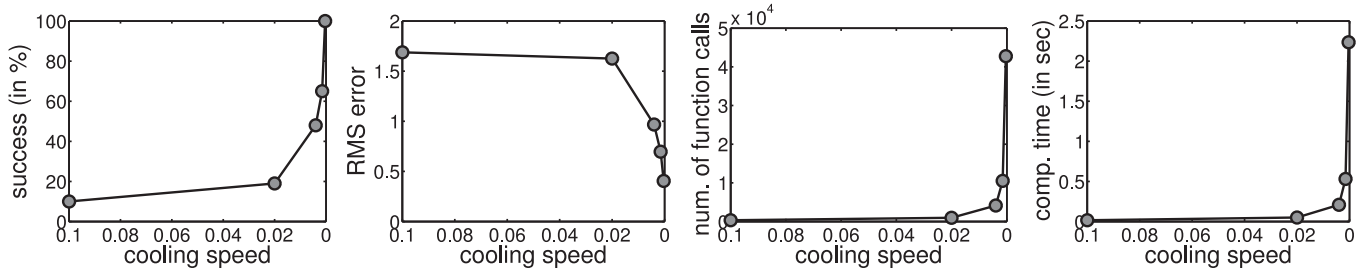


Fig. 10. From left to right: success rate, RMS error, number of cost function evaluations and computation time of our registration algorithm as a function of the cooling speed v (defined in (22)). Model and data used in this test case consist of 100 points randomly sampled from the outlier-free version of the data set shown in the top row of Fig. 6. One RMS error unit equals 1% of the bounding box diagonal length of the point set.

sets and compare the runtime of our algorithm with the one of a deterministic branch-and-bound method. In the following, we describe each test scenario in detail.

First, the success rate and the accuracy of our method are tested with two different kernels, namely, the inverse distance kernel (12) used in our cost function and the Huber kernel (7) used in [22]. The point sets used in this test together with some typical registration results are shown in Fig. 6. Note that outliers are added only to the data set and it is a subset of the model. This case occurs in real world scenarios in which one has a complete (relatively clean) model of an object and wants to align it to a low quality data set which only partially represents the object (due to visibility issues like, e.g., occlusion and scene clutter). As already mentioned in Section 2.1, we expect a registration method which minimizes a cost function based on the (unbounded) Huber kernel to have difficulties with outlier corrupted data sets. This is confirmed by the results of this test case which are summarized in the Fig. 7a and b.

In the second test case, we align two partially overlapping parts of the Coati model under varying conditions. This time, noise and outliers are added to both the model and the data set. This situation occurs in practice when building a complete object model out of multiple partially overlapping scans. We compare our

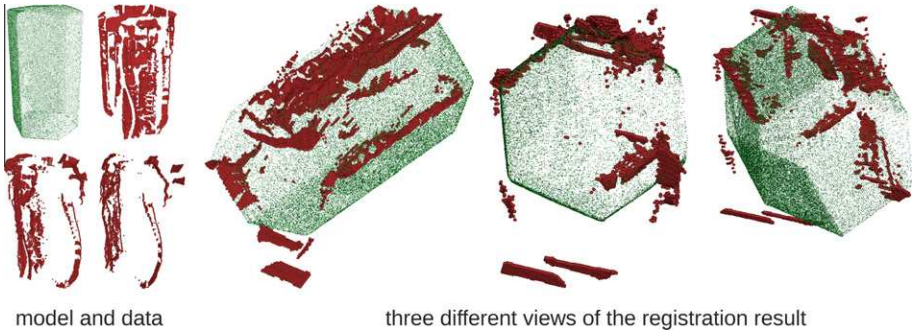


Fig. 11. (Left) The complete model of a box (shown in green; 236,089 points) and three views of the very low quality data set (shown in red; 5000 out of 9623 points were randomly sampled and used for the registration). The data was obtained with a correlation based stereo algorithm under poor lighting conditions. (Right) Our method robustly achieved the right alignment in 10 out of 10 trials. Each registration trial took about 30 s. The high amount of noise and outliers which almost completely destroy the shape of the object makes this a challenging example. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

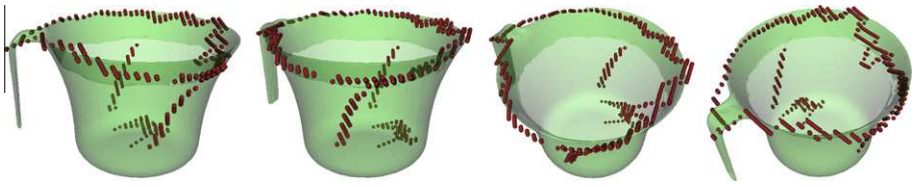


Fig. 12. Registration result in the case of a noisy and very sparsely reconstructed data set (shown by the red "curve") and a complete noise-free model (transparent green mesh). Note that in this case the state-of-the-art integral volume descriptor (used in [6]) will fail since the curve which represents the data set does not enclose a volume in $\mathbb{R}^3$. Local descriptors which use surface normals like, e.g., spin images [5] will fail as well since in general the normal of a curve which lies on a surface does not match the surface normal. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

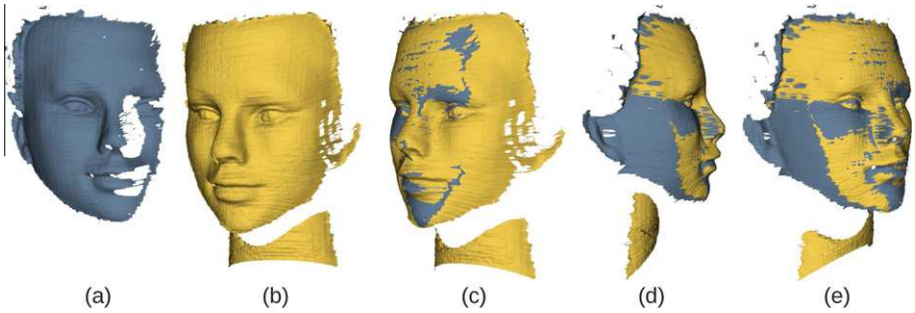


Fig. 13. Registration of noisy point sets with low overlap. Although rendered as meshes only points are used for the registration. Note that the input scans, (a) and (b), represent different parts of the face and the model set, shown in (a), contains no parts of the neck. (c)–(e) A typical registration result obtained with our method shown from three different viewpoints.

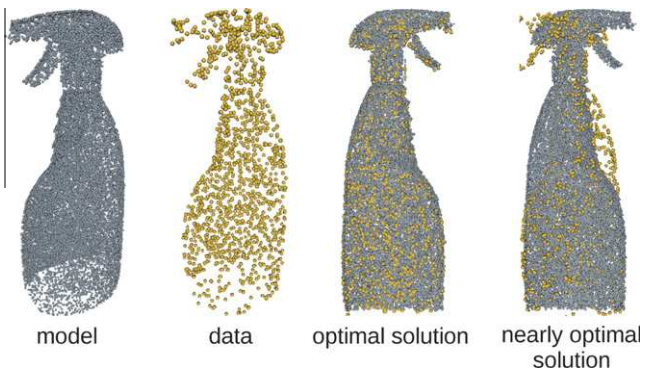


Fig. 14. Point sets leading to a cost function which has two almost equally low minima. The nearly optimal solution differs from the optimal one by a rotation of the data set by 180° about the axis which corresponds to the upright orientation of the bottle.

results with the ones reported in [7] which are obtained with the robust 4PCS algorithm and a state-of-the-art local descriptor based approach. We perform the tests on the same point sets which are used in [7]. This allows for a precise comparison without the need of re-implementing neither of the two algorithms. The model and data sets together with some typical registration results obtained with our method are shown in Fig. 8. In the Fig. 7c and d, we plot our results together with the ones reported in [7].

In the third test scenario, we measure the computation time of our algorithm and compare it with the one of the deterministic registration method of Li and Hartley [19]. Since we run the tests on a similar (i.e., *not* more powerful) hardware as the one used in [19] an accurate comparison is possible. The results are summarized in Fig. 9.

Next, we measure the performance of our method for varying cooling speed v defined in (22). We report the results in Fig. 10. Our algorithm achieves a success rate of 100% and an RMS error below 0.5 for less than 2.5 s (for point sets consisting of 100 points).

Finally, we demonstrate the ability of our method to deal with partially overlapping and very sparsely sampled point sets corrupted by noise and outliers which are not artificially generated but originate in scan device imprecision. In Fig. 11, we show that our method successfully computes the right registration even in the case of an extremely degraded data set which represents only a subset of the model. Fig. 12 illustrates the stability of our algorithm when dealing with very sparsely sampled data sets. Figs. 1 and 13 show typical registration results for partially overlapping points sets.

Note that our registration method could lead to incorrect results for a class of shapes for which several almost equally good alignments exist and the registration ambiguity can be dissolved by small scale features only. An example of such a shape is a large cup with a small handle. In this case, the corresponding point sets lead to a cost function with several local minima which are almost as "good" as the global one (see Fig. 14).

6. Conclusions

We introduced a new technique for pairwise rigid registration of point sets. Our method is based on a noise robust and outlier resistant cost function which itself is based on an inverse distance kernel. One of the main messages of the paper is that a registration method which minimizes an objective function based on an unbounded kernel will be sensitive to outliers in the point sets. This was fully validated by comparisons between our kernel and the Huber kernel which were presented in the experimental part of the paper.

A further property of our algorithm is that it does not rely on any initial estimation of the globally optimal rigid transform. This was achieved by employing a new stochastic algorithm for global optimization. In order to minimize efficiently over complex shaped search spaces like the space of rotations we generalized the BSP trees and introduced a new technique for hierarchical rotation space decomposition. Furthermore, we derived a new procedure for uniform point sampling from spherical boxes.

Tests on a variety of point sets showed that the proposed method is insensitive to noise and outliers and can cope very well with sparsely sampled and incomplete data sets. Comparisons showed that our algorithm is by three orders of magnitude faster than a deterministic branch-and-bound method and that it outperforms a recently proposed generate-and-test approach and a state-of-the-art local descriptor based method in terms of accuracy and robustness.

Acknowledgments

This work has been funded by the European Commission's Seventh Framework Programme as part of the Project GRASP (IST-FP7-IP-215821).

References

[1] P. Besl, N. McKay, A method for registration of 3-D shapes, IEEE Transactions on PAMI 14 (1992).
[2] Y. Hecker, R. Bolle, On geometric hashing and the generalized hough transform, IEEE Transactions on Systems, Man, and Cybernetics 24 (1994).
[3] H. Wolfson, I. Rigoutsos, Geometric hashing: an overview, IEEE Computational Science & Engineering 4 (1997) 10–21.
[4] G. Stockman, Object recognition and localization via pose clustering, Computer Vision, Graphics, and Image Processing 40 (1987) 361–387.
[5] A. Johnson, M. Hebert, Using spin images for efficient object recognition in cluttered 3D scenes, IEEE Transactions on PAMI 21 (1999) 433–449.
[6] N. Gelfand, N. Mitra, L. Guibas, H. Pottmann, Robust global registration, Eurographics Symposium on Geometry Processing (2005) 197–206.
[7] D. Aiger, N. Mitra, D. Cohen-Or, 4-Points congruent sets for robust pairwise surface registration, ACM Transactions on Graphics 27 (2008).
[8] Y. Chen, G. Medioni, Object modeling by registration of multiple range images, in: Proceedings of IEEE International Conference on Robotics and Automation, vol. 3, 1991, pp. 2724–2729.
[9] S. Rusinkiewicz, M. Levoy, Efficient variants of the ICP algorithm, 3DIM, 2001, pp. 145–152.
[10] S. Granger, X. Pennec, Multi-scale EM-ICP: a fast and robust approach for surface registration, in: Proceedings of ECCV, 2002, pp. 418–432.
[11] Y. Tsin, T. Kanade, A correlation-based approach to robust point set registration, in: Proceedings of ECCV, 2004, pp. 558–569.
[12] B. Jian, B.C. Vemuri, A robust algorithm for point set registration using mixture of gaussians, in: Proceedings of ICCV, 2005, pp. 1246–1251.
[13] T. Tamaki, M. Abe, B. Raytchev, K. Kaneda, Softassign and EM-ICP on GPU, in: Proceedings of the 2nd Workshop on Ultra Performance and Dependable Acceleration Systems (UPDAS), 2010.
[14] B. Ma, R.E. Ellis, Surface-based registration with a particle filter, in: Proceedings of MICCAI, 2004, pp. 566–573.
[15] M.H. Moghari, P. Abolmaesumi, Point-based rigid-body registration using an unscented Kalman filter, IEEE Transactions on Medical Imaging 26 (2007) 1708–1728.
[16] R. Sandhu, S. Dambreville, A. Tannenbaum, Point set registration via particle filtering and stochastic dynamics, IEEE Transactions on PAMI 32 (2010) 1459–1473.
[17] T.M. Breuel, Implementation techniques for geometric branch-and-bound matching methods, Computer Vision and Image Understanding 90 (2003) 258–294.
[18] C. Olsson, F. Kahl, M. Oskarsson, Branch-and-bound methods for Euclidean registration problems, IEEE Transactions on PAMI 31 (2009) 783–794.
[19] H. Li, R.I. Hartley, The 3D–3D registration problem revisited, in: Proceedings of ICCV, 2007, pp. 1–8.
[20] N. Mitra, N. Gelfand, H. Pottmann, L. Guibas, Registration of point cloud data from a geometric optimization perspective, in: Symposium on Geometry Processing, 2004, pp. 23–32.
[21] H. Pottmann, Q.-X. Huang, Y.-L. Yang, S.-M. Hu, Geometry and convergence analysis of algorithms for registration of 3D shapes, International Journal of Computer Vision 67 (2006) 277–296.
[22] A.W. Fitzgibbon, Robust registration of 2D and 3D point sets, Image Vision Computing 21 (2003) 1145–1153.
[23] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller, Equation of state calculations by fast computing machines, The Journal of Chemical Physics 21 (1953) 1087–1092.
[24] V. Cerny, Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm, Journal of Optimization Theory and Applications 45 (1985) 41–51.
[25] S. Kirkpatrick, C. Gelatt, M. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.
[26] P. Pardalos, E. Romeijn (Eds.), Handbook of Global Optimization 2, Nonconvex Optimization and Its Applications, Kluwer Academic Publishers, 2002.
[27] D. Bulger, G. Wood, Hesitant adaptive search for global optimisation, Mathematical Programming 81 (1998) 89–102.
[28] G. Bilbro, W. Snyder, Optimization of functions with many minima, IEEE Transactions on Systems, Man, and Cybernetics 21 (1991) 840–849.
[29] C. Papazov, D. Burschka, Stochastic optimization for rigid point set registration, in: Proceedings of ISVC, 2009, pp. 1043–1054.
[30] H. Samet, The Design and Analysis of Spatial Data Structures, Addison-Wesley, 1990.
[31] K. Kanatani, Group-Theoretical Methods in Image Understanding, Springer Series in Information Sciences, Springer, 1990.
[32] A. Watt, M. Watt, Advanced Animation and Rendering Techniques, Addison-Wesley, 1992.
[33] R.I. Hartley, F. Kahl, Global optimization through rotation space search, International Journal of Computer Vision 82 (2009) 64–79.
[34] N. Madras, Lectures on Monte Carlo Methods, American Mathematical Society, 2002.

Eurographics Symposium on Geometry Processing 2011
Mario Botsch and Scott Schaefer
(Guest Editors)

Volume 30 (2011), Number 5

Deformable 3D Shape Registration Based on Local Similarity Transforms

C. Papazov and D. Burschka

Technische Universität München (TUM), Robotics and Embedded Systems, Germany

Abstract

In this paper, a new method for deformable 3D shape registration is proposed. The algorithm computes shape transitions based on local similarity transforms which allows to model not only as-rigid-as-possible deformations but also local and global scale. We formulate an ordinary differential equation (ODE) which describes the transition of a source shape towards a target shape. We assume that both shapes are roughly pre-aligned (e.g., frames of a motion sequence). The ODE consists of two terms. The first one causes the deformation by pulling the source shape points towards corresponding points on the target shape. Initial correspondences are estimated by closest-point search and then refined by an efficient smoothing scheme. The second term regularizes the deformation by drawing the points towards locally defined rest positions. These are given by the optimal similarity transform which matches the initial (undeformed) neighborhood of a source point to its current (deformed) neighborhood. The proposed ODE allows for a very efficient explicit numerical integration. This avoids the repeated solution of large linear systems usually done when solving the registration problem within general-purpose non-linear optimization frameworks. We experimentally validate the proposed method on a variety of real data and perform a comparison with several state-of-the-art approaches.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—

1. Introduction

Deformable (non-rigid) shape registration is a fundamental problem in computational geometry with applications in the fields of computer vision, computer graphics, medical image processing and many others. In recent years, 3D geometry acquisition techniques have been developed which allow to capture the surface of deforming objects in real time [WLG07]. In order to analyze the motion of the object it is important to register subsequent scans and/or to register a complete geometric model to the scans. Since the object is undergoing a non-rigid motion, rigid registration algorithms [CM91, BM92, GMGP05, PB09] can not be used adequately in this setting.

The problem of deformable shape registration can loosely be defined as follows. Given a source shape $\mathcal{S}$ and a target shape $\mathcal{T}$ find a "reasonable" deformation F that brings $\mathcal{S}$ "close" to $\mathcal{T}$. In this paper, we assume that $\mathcal{S}$ and $\mathcal{T}$ are consisting of a finite set of points with an underlying neighborhood structure. Examples include range images, meshes and volumetric grids, just to name a few. In this case, the deformation we are looking for is a mapping $F : \mathcal{S} \rightarrow \mathbb{R}^3$. To choose a reasonable one from the space of all mappings, we have to impose some constraints on the deformation. This is called regularization of F. We use a regularizer that pulls each source shape point $\mathbf{x}_k$ towards its rest position given by the optimal similarity transform which matches the initial (undeformed) neighborhood of $\mathbf{x}_k$ to its current (deformed) neighborhood. This is a generalization of the object deformation technique presented in [MHTG05, RJ07, SOG08], where local rigid shape matching is used. Employing similarity transforms instead of rigid ones allows to model as-rigid-as-possible shape deformations plus local scale. Note that the topic of the papers [MHTG05, RJ07, SOG08] is the generation of physically plausible animations. To the best of our knowledge, this is the first paper which exploits rigid and similarity-based shape matching as regularizers in the context of deformable 3D shape registration.

In order to deform the source shape such that it comes closer to the target shape $\mathcal{T}$, each source point moves towards a corresponding point on $\mathcal{T}$. We use closest-point

search to establish preliminary correspondences which are further refined by a simple but effective vector field smoothing procedure.

Considering the regularizer and the correspondence field, we introduce a system of ordinary differential equations (ODEs) which describes the non-rigid motion of the source shape. The iterative solution of the ODEs yields a trajectory for each source point from its initial position to its end position on the target shape. In the case of incomplete data, the points move according to the regularizer and fill in missing regions in a reasonable way. Our method computes a dense correspondence between source and target. Since the initial correspondence estimation is based on closest-point computations we assume that both shapes are roughly pre-aligned. This assumption holds in a variety of situations like, e.g., in the case of scanning a deforming object at high frame rates such that the inter-frame displacements are small.

The rest of the paper is organized as follows. After reviewing previous work in Section 2, we describe our algorithm in Section 3. Important implementation issues are discussed in Section 4. Section 5 presents experimental results. Conclusions are drawn in the final Section 6 of the paper.

2. Related Work

There is a large variety of deformable registration algorithms each one having its advantages and drawbacks. In this paper, our main criterion to judge the methods is the processing time they require.

One class of deformable registration approaches consists of the feature-based methods. Several papers [WAS10, WZL*10, BK10, RBBK10] proposed to use local invariant geometric descriptors to compute a one-to-one mapping between corresponding features on the input shapes. However, detecting feature points and establishing the correspondence can be problematic especially in the presence of noise and missing data. Furthermore, many shapes do not have distinctive features which gives rise to many ambiguous correspondences and the matching algorithm degenerates to a brute force search [AMCO08].

A different strategy is to transform the shapes to a canonical representation in a suitable space in which the correspondence problem is easier to solve. Several papers [EK03, BBK06, WSBA07, WSB09] proposed to compute isometry-invariant embeddings of the original shapes in a low-dimensional Euclidean space and to establish the correspondence using rigid registration algorithms. These methods, however, tend to be costly and, moreover, fail for incomplete data (caused by surface holes, partial views, *etc.*).

The methods cited so far solve the correspondence problem even in the presence of significant deformations and without making any assumptions about an initial alignment of the shapes. However, the deformations are restricted to isometries (an exception is [BK10] which can handle an additional global or local scaling). Furthermore, the actual warp between the shapes has to be computed in an additional step using the established correspondences as constraints [MHTG05, RJ07, BPWG07, SOG08]. In contrast to this, our method is not restricted to a particular family of transformations and it efficiently computes both a dense correspondence and the warp between the shapes.

There is a variety of registration algorithms specialized to articulated shapes. [ACP03] presented a framework for deformable marker-based fitting of a high-resolution template to 3D scans of different humans in the same pose. In [ASK*05] a deformable model was learned that is able to synthesize realistic muscle deformations based on the pose of an articulated human skeleton. Both methods can be used for human shape completion as well. Further shape completion algorithms which use deformable registration modules were presented in [KS05, PMG*05]. In [CZ08], a fully automatic approach for articulated shape registration was proposed. The registration problem is converted to a discrete labeling problem and solved via graph cuts. However, this seems to be very costly since the authors report processing times of more than an hour for shapes consisting of not more than 12,000 points.

A further class of non-rigid registration algorithms consists of iterative solvers. They deform the source shape in an iterative fashion until an "optimal" alignment to the target shape is achieved. Many methods in this class are extensions of the classic ICP algorithm [CM91, BM92]. In [IGL03], a non-rigid registration technique was introduced which decomposes the input scans in a coarse-to-fine hierarchical manner in overlapping rigid pieces which are aligned separately. However, the resulting deformation is not continuous which can lead to artifacts in the overlapping regions. Furthermore, the procedure has a quadratic time complexity in the number of pieces. In [BR04], the discontinuity issue was resolved by incorporating a global thin-plate splines warp which guarantees the smoothness of the solution. A generalization of this method to the simultaneous matching of multiple scans was proposed in [BR07].

Instead of assuming a one-to-one correspondence between the shape points, one-to-many relaxations can be used in order to enlarge the basin of convergence and thus to increase robustness against imprecise initializations. Significant contributions along these lines are the softassign and deterministic annealing technique [CR03] and the coherent point drift algorithm [MS10]. A statistical registration approach without explicitly establishing point-to-point correspondences was proposed in [TK04]. The input point sets are modeled as probability distributions and a distance measure between them is minimized over the transform space. Recently, a Gaussian mixture models-based approach [JV11] was proposed which can be seen as a generalization of [CR03, TK04, MS10]. However, these algorithms compute

registration results which are not as precise as ours and are much slower than our method (see the experimental comparisons in Section 5).

A deformable ICP extension was introduced in [ARV07]. The authors formulated a cost function, similar to the one used in [ACP03], which measures the cost of a given non-rigid alignment between the shapes. The deformation is modeled using one affine 3×4 transformation matrix per shape point. This gives rise to a cost function of $12m$ variables, where m is the number of points in the source shape. In order to solve this highly over-determined system, a stiffness term (a regularizer) is introduced. It penalizes differences between the transformation matrices of neighboring points.

A similar strategy was proposed in [SAY*09]. The authors iteratively minimized an error measure which is based on an elastic convolution between the difference of corresponding points in the shapes. This is the 3D surface patch analog to 2D template matching commonly used in image processing. In [LSP08], the deformation is also modeled using one affine transformation matrix per point. The cost function comprises four energy terms and depends on $15m+6$ variables. The authors minimized it with the Levenberg-Marquardt algorithm.

Note that the iterative methods cited above model the deformation in a very redundant way: many more degrees of freedom (DoFs) are introduced than needed to describe an arbitrary motion of a system of m points in $\mathbb{R}^3$. This results in high-dimensional and computationally heavy optimization problems. In contrast to this, our approach is based on a system of ODEs with $3m$ unknowns, which is the least number necessary to model a general motion of a system with $3m$ DoFs. Furthermore, since the proposed ODEs system allows for a very efficient explicit integration we avoid to repeatedly solve large linear systems which is usually done during the minimization of non-linear cost functions. Thus, our method is efficient in terms of both computational complexity and memory.

3. Method Description

Before we describe our deformable shape registration algorithm in detail let us first introduce some notation used in the paper. Consider the source shape $\mathcal{S} = (\mathbf{V}_S, \mathbf{E}_S)$, where $\mathbf{V}_S = \{\mathbf{x}_1^0, \ldots, \mathbf{x}_m^0\} \subset \mathbb{R}^3$ is the set of initial positions of the points $\mathbf{x}_1, \ldots, \mathbf{x}_m$ and $\mathbf{E}_S = \{\mathbf{N}_1, \ldots, \mathbf{N}_m\}$ is the neighborhood structure. Each $\mathbf{N}_k$ contains the indices (including k) of the neighbors of $\mathbf{x}_k$. The position of $\mathbf{x}_k$ at a time t is given by a function $t \mapsto \mathbf{x}_k(t)$, with initial value $\mathbf{x}_k(0) = \mathbf{x}_k^0$. Let $\mathbf{X}_k(t)$ denote the position of $\mathbf{x}_k$ and its neighbors at a time t, i.e., $\mathbf{X}_k(t) = (\mathbf{x}_{k_1}(t), \ldots, \mathbf{x}_{k_{N_k}}(t))$, where $\{k_1, \ldots, k_{N_k}\} = \mathbf{N}_k$.

Analogously, the target shape is a pair $\mathcal{T} = (\mathbf{V}_T, \mathbf{E}_T)$ with $\mathbf{V}_T = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\} \subset \mathbb{R}^3$ being the set of points and $\mathbf{E}_T = \{\mathbf{M}_1, \ldots, \mathbf{M}_n\}$ being the neighborhood structure. Note that since $\mathcal{T}$ does not deform we use $\mathbf{y}_1, \ldots, \mathbf{y}_n$ to denote both the target points and their positions in $\mathbb{R}^3$.

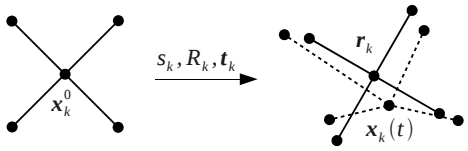


Figure 1: *The initial (undeformed) neighborhood $\mathbf{X}_k(0)$ (shown on the left) is matched to the current (deformed) neighborhood $\mathbf{X}_k(t)$ (indicated by the dashed lines on the right) using the similarity transform which minimizes the sum of squared distances between the corresponding points $\mathbf{x}_i^0 \leftrightarrow \mathbf{x}_i(t)$. The rest position $\mathbf{r}_k$ for $\mathbf{x}_k(t)$ is the transformed point $\mathbf{x}_k^0$.*

3.1. Shape Matching-Based Regularization

The regularizer pulls each point $\mathbf{x}_k$ towards its rest position $\mathbf{r}_k$ which is computed in the following way:

$$\mathbf{r}_k = s_k R_k \mathbf{x}_k^0 + \mathbf{t}_k. \tag{1}$$

$s_k \in \mathbb{R}$ is the scale factor, $R_k \in SO(3)$ is the rotation matrix and $\mathbf{t}_k \in \mathbb{R}^3$ is the translation vector which optimally match the initial (undeformed) neighborhood $\mathbf{X}_k(0)$ of $\mathbf{x}_k$ to its current (deformed) neighborhood $\mathbf{X}_k(t)$. More formally,

$$(s_k, R_k, \mathbf{t}_k) = \underset{s,R,\mathbf{t}}{\operatorname{argmin}} \sum_{i \in \mathbf{N}_k} \|(sR\mathbf{x}_i^0 + \mathbf{t}) - \mathbf{x}_i(t)\|^2. \tag{2}$$

Note that s_k, R_k and $\mathbf{t}_k$ depend on the current positions of the neighbors of $\mathbf{x}_k$. This means that s_k, R_k, $\mathbf{t}_k$ and the rest position $\mathbf{r}_k$ are functions of $\mathbf{X}_k(t)$. To stress this, when necessary, we write $\mathbf{r}_k(\mathbf{X}_k(t))$. Fig. 1 illustrates the idea of the regularization based on similarity shape matching.

The minimization problem (2) is called the absolute orientation problem and is often encountered in different fields as part of different computational problems [BM92, MHTG05, MS09]. Our solution is based on [MS09]. First, a linear deformation matrix A_k is computed:

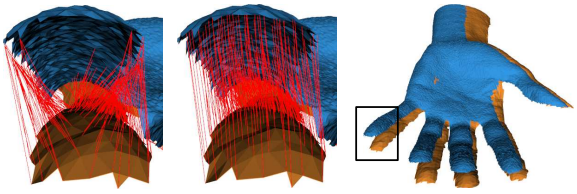
$$A_k = \sum_{i \in \mathbf{N}_k} (\mathbf{x}_i(t) - \mathbf{c}_k(t))(\mathbf{x}_i^0 - \mathbf{c}_k^0)^T, \tag{3}$$

where the center of mass of the initial and the deformed neighborhood of $\mathbf{x}_k$ are denoted by $\mathbf{c}_k^0$ and $\mathbf{c}_k(t)$, respectively. Next, the optimal rotation matrix R_k is extracted from A_k using its singular value decomposition $A_k = U\Sigma V^T$ in the following way:

$$R_k = UCV^T, \quad C = \operatorname{diag}(1, \ldots, 1, \det(UV^T)), \tag{4}$$

where the diagonal matrix C assures that R_k is a rotation and not a reflection. The scale factor is given by

$$s_k = \sqrt{\frac{\sum_{i \in \mathbf{N}_k} \|\mathbf{x}_i(t) - \mathbf{c}_k(t)\|^2}{\sum_{i \in \mathbf{N}_k} \|\mathbf{x}_i^0 - \mathbf{c}_k^0\|^2}} \tag{5}$$



(a) (b) source & target

Figure 2: *The correspondence field computed with a closest-point search (a) and with our smoothing procedure (b) for the input shapes on the right. The black rectangle on the right marks the part of the little finger magnified in (a) and (b). Obviously, our procedure estimates the correspondences much more accurate.*

and the translation vector is computed as $\mathbf{t}_k = s_k R_k \mathbf{c}_k^0 - \mathbf{c}_k(t)$.

Using Eq. (1) with $s_k = 1$ results in a rigid shape matching-based regularizer which is well-suited to model as-rigid-as-possible deformations. Computing the scale according to (5) allows us to include a local scale.

The rigid shape matching regularizer was first introduced in [MHTG05] for the generation of physically plausible animations of deforming objects. Further improvements, again, for animating deformations, were proposed in [RJ07, SOG08]. To the best of our knowledge, there is no paper which exploits the rigid shape matching ($s_k = 1$) or similarity-based (s_k according to (5)) regularization in the context of deformable 3D shape registration.

3.2. Correspondence Estimation

In this subsection, we introduce a correspondence field $\mathbf{d} : \mathcal{S} \rightarrow \mathbb{R}^3$ which defines a pointwise correspondence between the source shape $\mathcal{S}$ and the target shape $\mathcal{T}$. As already discussed in Section 2, there is a substantial amount of work in the field of non-rigid correspondence estimation [BK10, RBBK10, WZL*10, WAS10, ZWW*10]. These methods solve the problem without making any assumptions about the initial alignment of the shapes but, unfortunately, tend to be costly. In contrast to this, we exploit the fact that consecutive scans of deforming objects exhibit small inter-frame displacements and design a simple but effective correspondence establishment procedure.

Perhaps the most common way of doing this (see [BM92, ARV07, LSP08]) is to compute the vector connecting each source point $\mathbf{x}_k$ to its closest point on $\mathcal{T}$:

$$\mathbf{c}_{\mathcal{T}}(\mathbf{x}_k) = \underset{\mathbf{y}_i \in \mathcal{T}}{\operatorname{argmin}} \|\mathbf{x}_k - \mathbf{y}_i\| - \mathbf{x}_k. \tag{6}$$

This, however, leads to many wrong correspondences, even for shapes which are not very far from each other (see Fig. 2(a)). In this paper, we use $\mathbf{c}_{\mathcal{T}}$ only as a starting point of a simple and effective smoothing technique which significantly improves the correspondence estimation. In many cases (like the one shown in Fig. 2(a)), $\mathbf{c}_{\mathcal{T}}$ is quite irregular in the sense that it varies too strong when evaluated at neighboring points on the source shape. This should not be so since scanning a deforming object at high frame rates produces a smoothly varying surface which, in turn, results in a smooth correspondence field between consecutive frames. This is the reason why we expect a smoothing of $\mathbf{c}_{\mathcal{T}}$ to improve the correspondence estimation between the shapes.

More precisely, we use $\mathbf{c}_{\mathcal{T}}$ as a starting point for a local optimization procedure which returns a displacement field $\mathbf{d} : \mathcal{S} \rightarrow \mathbb{R}^3$ which minimizes the smoothness term

$$E_s(\mathbf{d}) = \sum_{k=1}^{m} \left\| \mathbf{d}(\mathbf{x}_k) - \overline{\mathbf{d}(\mathbf{x}_k)} \right\|^2, \tag{7}$$

where m is the number of source shape points and $\overline{\mathbf{d}(\mathbf{x}_k)} = \frac{1}{N_k} \sum_{i \in \mathbf{N}_k} \mathbf{d}(\mathbf{x}_i)$ is the mean value of $\mathbf{d}$ over the neighborhood of $\mathbf{x}_k$. Furthermore, E_s has to be minimized under the constraint

$$\mathbf{x}_k + \mathbf{d}(\mathbf{x}_k) \in \mathcal{T}, \quad \forall \mathbf{x}_k \in \mathcal{S}. \tag{8}$$

Essentially, (7) penalizes displacement fields which vary at neighboring source points and (8) assures that the solution is indeed a correspondence field, i.e., that $\mathbf{d}$ brings each $\mathbf{x}_k$ to a point on the target shape and not to some arbitrary position in $\mathbb{R}^3$. This means that $\mathbf{d}$ has the form $\mathbf{d}(\mathbf{x}_k) = \mathbf{y}_m - \mathbf{x}_k$ for some $\mathbf{y}_m \in \mathcal{T}$ and we have a discrete optimization problem. Having this in mind, we construct the following optimization procedure:

1. Initialize $\mathbf{d}_0 := \mathbf{c}_{\mathcal{T}}$ and $j := 0$.
2. Compute $\mathbf{d}_{j+1}$

[for each $\mathbf{x}_k \in \mathcal{S}$ compute $\mathbf{d}_{j+1}(\mathbf{x}_k)$]

   a. Get the target point $\mathbf{y}_l$ which corresponds to $\mathbf{x}_k$, i.e., $\mathbf{y}_l = \mathbf{x}_k + \mathbf{d}_j(\mathbf{x}_k)$.
   b. Among the neighbors of $\mathbf{y}_l$, choose the target point $\mathbf{y}_m \in \mathcal{T}$ with minimal $\|(\mathbf{y}_m - \mathbf{x}_k) - \overline{\mathbf{d}_j(\mathbf{x}_k)}\|^2$.
   c. Set $\mathbf{d}_{j+1}(\mathbf{x}_k) := \mathbf{y}_m - \mathbf{x}_k$.

[end for]

3. If $E_s(\mathbf{d}_{j+1}) < E_s(\mathbf{d}_j)$ increment j and go to step 2. Otherwise terminate the procedure.

The only parameter of this algorithm is the radius of the neighborhood of the target point $\mathbf{y}_l$ in step 2.b.. The bigger the radius the more global the search, i.e., the greater the chance to overcome local minima in the landscape of the smoothness term. Of course, this comes at the cost of a higher computational load. The algorithm always converges since $\mathbf{y}_l$ is part of its own neighborhood and after a certain number of iterations no further minimization of the terms in step 2.b. is possible which results in $E_s(\mathbf{d}_{j+1}) = E_s(\mathbf{d}_j)$.

Fig. 2(b), shows a correspondence field computed with our smoothing procedure. The improvement compared to the closest-point field $\mathbf{c}_{\mathcal{T}}$ is obvious.

3.3. The ODEs System and its Integration

In contrast to [MHTG05, RJ07, BPWG07], we are not interested in creating physically plausible animations of deforming objects. Thus, our approach is not based on a physical model like, e.g., Newton's second law used in [MHTG05, RJ07, BPWG07].

At each time instance every source shape point should move according to both the regularizer and the correspondence field. In other words, the velocity of each $\mathbf{x}_k$ is a linear combination of the vector pointing to its rest position, namely, $\mathbf{r}_k - \mathbf{x}_k$ and the vector given by the correspondence field evaluated at $\mathbf{x}_k$ which is $\mathbf{d}(\mathbf{x}_k)$. More precisely, we set the velocity to be a convex combination of $\mathbf{r}_k - \mathbf{x}_k$ and $\mathbf{d}(\mathbf{x}_k)$. Writing this down in a formal way, leads to the following system of m ordinary differential equations (with m being the number of source shape points):

$$\dot{\mathbf{x}}_k(t) = \alpha(\mathbf{r}_k(\mathbf{X}_k(t)) - \mathbf{x}_k(t)) + (1-\alpha)\mathbf{d}(\mathbf{x}_k(t)), \quad (9)$$

$$\mathbf{x}_k(0) = \mathbf{x}_k^0. \quad (10)$$

Using a convex combination instead of a general linear combination has the advantage of introducing only one parameter $\alpha \in [0,1]$ which can be interpreted as the stiffness of the source shape: the greater the value the more rigid the motion of the shape. (9) together with (10) define an initial value problem which we solve numerically using the following integration scheme:

$$\mathbf{x}_k^{n+1} = \mathbf{x}_k^n + \alpha(\mathbf{r}_k^n - \mathbf{x}_k^n) + (1-\alpha)\mathbf{d}(\mathbf{x}_k^n), \quad (11)$$

where n is the iteration number. Since this is an explicit method (Euler's method with step size 1) its stability is a-priori not guaranteed. If the step size is chosen too large, explicit integration schemes can overshoot the equilibrium of the system by an amount which increases in each iteration and finally leads to an "explosion" [MHTG05]. In our case, however, this does not happen. Recall from Section 3.2 that for each source point $\mathbf{x}_k^n$ the correspondence field has the form $\mathbf{d}(\mathbf{x}_k^n) = \mathbf{y}_m - \mathbf{x}_k^n$ for some target point $\mathbf{y}_m$. Using this we can rewrite (11) to get

$$\mathbf{x}_k^{n+1} = \alpha\mathbf{r}_k^n + (1-\alpha)\mathbf{y}_m. \quad (12)$$

This means that the new point $\mathbf{x}_k^{n+1}$ lies on the straight line between $\mathbf{r}_k^n$ and $\mathbf{y}_m$ and thus can not overshoot the equilibrium since it lies on this line as well. Using a step size larger than 1 leads to an $\mathbf{x}_k^{n+1}$ which overshoots the line (and thus the equilibrium) and can lead to instability.

3.4. The Overall Registration Procedure

The overall deformable registration algorithm works as follows. We start the integration of the ODEs system using the numerical scheme described in Section 3.3 with a high stiffness value $\alpha = 0.95$. In each iteration, the rest positions and the correspondences are recomputed as described in Sections 3.1 and 3.2, respectively, using the updated positions of the source shape points. This is repeated until convergence which is detected by checking whether (11) has reached a fix point. The stiffness parameter is then lowered by 0.05 and the integration continues. The registration terminates when the stiffness falls below 0.5.

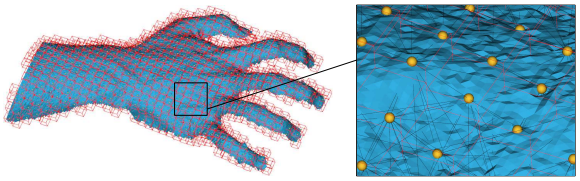


Figure 3: *A deformation graph computed for a range scan of a hand. The magnified part on the right shows the nodes as yellow spheres. Neighboring nodes are connected with black lines.*

Note that this procedure does not cope well with missing data since the correspondence field guides *each* source point $\mathbf{x}_k$ towards a target point no matter if it is the one which semantically corresponds to $\mathbf{x}_k$ or not. What happens to such a source point in the course of the registration is that its neighborhood gets too distorted, especially for lower stiffness values. We detect these points using a simple deformation measure and let them move only according to the regularizer. We use

$$D(n,k) = \frac{1}{N_k - 1} \sum_{\substack{i \in \mathbf{N}_k \\ i \neq k}} \frac{|\|\mathbf{x}_i^n - \mathbf{x}_k^n\| - l_{ik}|}{l_{ik}}, \quad (13)$$

as the measure of deformation of the source point $\mathbf{x}_k$ in the n-th iteration of the registration algorithm. (This is essentially the definition of strain in mass-spring systems [WOR10].) Recall that $\mathbf{N}_k$ is the set of indices (including k) of the neighbors of $\mathbf{x}_k$ and $N_k = |\mathbf{N}_k|$. l_{ik} denotes the distance between $\mathbf{x}_i$ and $\mathbf{x}_k$ in the undeformed source shape, i.e., $l_{ik} = \|\mathbf{x}_i^0 - \mathbf{x}_k^0\|$. Note that $D(n,k)$ is dimensionless meaning that it does not depend on the units in which the shapes are saved. If $D(n,k)$ exceeds a certain fixed threshold the neighborhood of $\mathbf{x}_k$ is considered too distorted and from the n-th iteration on the point moves only according to the regularizer without being directly attracted to the target shape. In our implementation, we set this threshold to 0.2.

4. Implementation Issues

Note that the registration algorithm introduced in the last section is applicable to all shapes which are represented by a finite set of points plus an underlying neighborhood structure. In this Section, we will briefly discuss two important special cases, namely, range scans and triangular meshes.

A range scan is a 2D image in the xy-plane which stores a depth value along the z-direction [LSP08]. The range scans used in the experimental part of the paper contain

around 30,000 points. This introduces a significant computational load and leads to an unpractical registration algorithm. Moreover, many deformations of practical interest like, e.g., articulated motion have much fewer degrees of freedom and can be described in a more efficient way. This is the reason why we decouple the complexity of the registration algorithm from the geometric complexity by using a so-called deformation graph [SSP07]. It consists of nodes connected by undirected edges. We exploit the regularity of the range scans and cover the xy-plane with non-overlapping three-dimensional boxes of a certain fixed size. The z-coordinate of each box is chosen to be median of the z-coordinates of all shape points being covered by this box. Then for each box the shape point closest to its center is defined to be a node in the deformation graph. Boxes having integer xy-coordinates which do not differ by more than 2 and having z-coordinates which do not differ by a certain amount (we use 30mm) are considered to be neighbors. Fig. 3 illustrates this concept.

If the source shape is represented by a triangular mesh, we use a different strategy. In this case, the deformation graph is an octree of fixed leaf size. If, furthermore, the mesh represents a closed surface, we add all leaves to the octree which are contained within the surface. In this way, a solid 3D lattice is created which results in more stable deformations [BPWG07].

After the deformation graph has been built it is used for the registration instead of the original source shape. However, since we are interested in deforming the shape itself, the deformation computed for the graph has to be transferred to the shape. This is done using thin plate spline (TPS) interpolation of the vector field defined by $\mathbf{g}_i - \mathbf{g}_i^0$, where $\mathbf{g}_i$ is the position of the i-th graph node after the registration and $\mathbf{g}_i^0$ is its initial position. Each source shape point is then transformed using the computed TPS. We chose this interpolation scheme since it produces high-quality vector fields and is easy to compute for scattered data.

5. Experimental Results

In this Section, the proposed registration algorithm is experimentally validated on a variety of real data sets. All tests are performed on a laptop with a 3GHz CPU, 4GB RAM and a Linux operating system. The method is implemented in C++.

Qualitative Tests First, we show two qualitative test results using a doll head model as the source shape and a head model of a girl and a boy as the target shapes. The shapes are represented as closed triangular meshes. Since they were not pre-aligned, we preformed a manual rigid registration based on 8 landmarks. Even after this user intervention the test scenario remains challenging because the models are representing different "subjects" and there is a significant scale difference between the shapes. Figure 4 shows the input data sets and the registration results from several view points.

Range Scan Pairs Next, we run our method on pairs of range scans representing the same object in different poses. In order to evaluate the method quantitatively, we measure the source shape deformation and the RMS error between source and target and plot them versus the iteration number. The source shape deformation is defined using (13) as

$$D(n,\mathcal{S}) = \frac{1}{m}\sum_{k=1}^{m} D(n,k) \tag{14}$$

and the RMS error between $\mathcal{S}$ and $\mathcal{T}$ is given by

$$\mathrm{RMS}(n,\mathcal{S},\mathcal{T}) = \sqrt{\frac{1}{m}\sum_{k=1}^{m} \|\mathbf{c}_{\mathcal{T}}(\mathbf{x}_k^n)\|^2}, \tag{15}$$

where n is the iteration number, m is the number of points in $\mathcal{S}$ and $\mathbf{c}_{\mathcal{T}}(\mathbf{x}_k^n)$ computes the vector connecting $\mathbf{x}_k^n$ to its closest point in $\mathcal{T}$ (see (6)).

Figures 5 to 9 show the data sets used in the test. The scans are shown as they were captured by the scanning devices without any additional alignment. These configurations are used as starting point for our registration algorithm. Fig. 5 shows a range scan pair which is part of a sequence representing a slowly closing hand. The inter-frame displacement is small and mainly caused by the bending finders. The registration computed with our method together with the deformation measure and the RMS error are shown on the right side of the figure.

Fig. 6 shows a further example of a closing hand. This time, there is a larger bending deformation plus an additional global translation. As is to be expected from the initial configuration of the scans the RMS error at the beginning of the registration is bigger. Furthermore, since the fingers bend more than in the last example the amount of deformation required to register the scans is larger. This is confirmed by the plots on the right side of the figure.

Fig. 7 demonstrates the ability of our algorithm to deal with incomplete data. Note that there are many holes in both scans caused by self-occlusion and scan device imperfection. Our method successfully registers the scans even in areas of low overlap as the magnified parts of the figure show.

Fig. 8 shows registration of an articulated object, namely, a bending arm. Note that there is a significant deformation between the scans. This is a challenging example for feature-based methods since the scans are smooth and lack distinctive features. Our method successfully recovers the deformation as depicted in the figure.

In Fig 9, a further example of a moving hand is shown. Additionally to the local deformation caused by the closing fingers this example contains a significant global rotation.

Range Scan Sequence Next, we test our method on a sequence of range scans representing a closing hand. The first frame of the sequence is used as the source shape and is sequentially registered to the other frames. The result for the

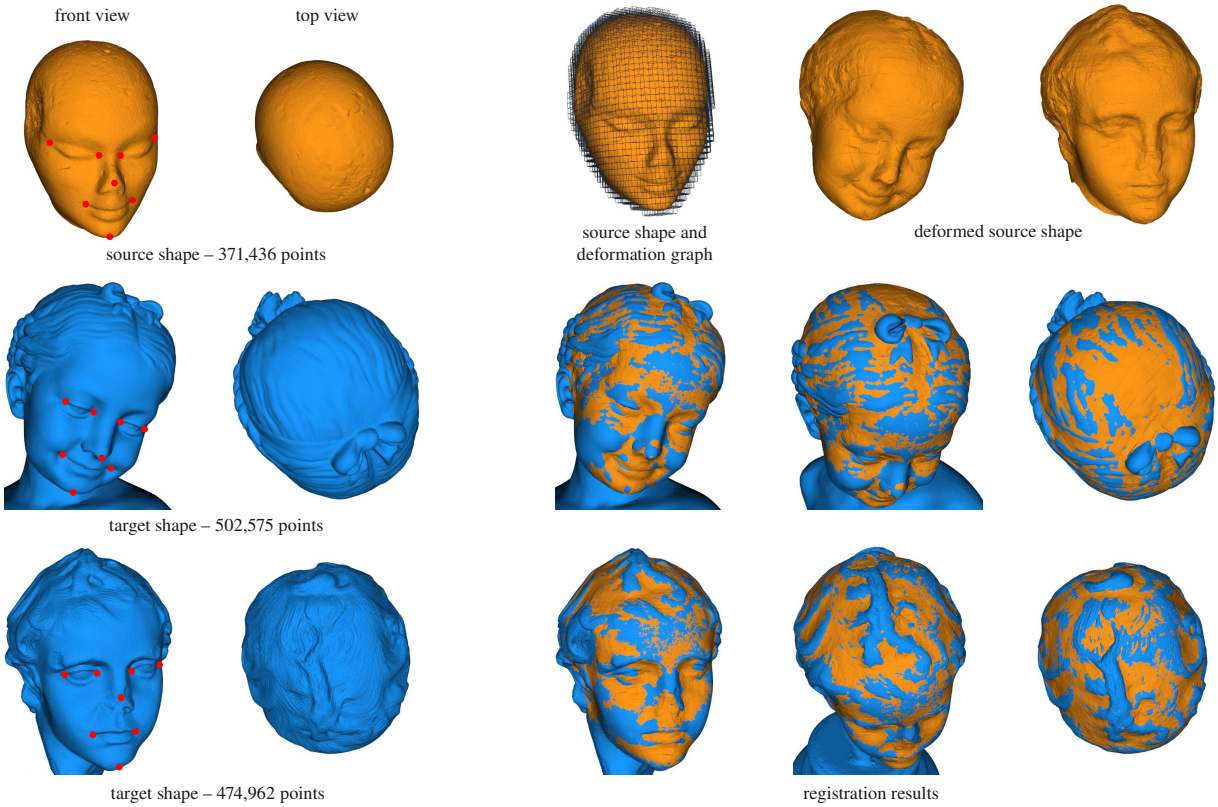


Figure 4: *Registering a doll head (upper left) to a head of a girl and a boy (lower left). The landmarks used for these registration tests are shown as red dots. Note that there is a significant difference in scale between the source and the target shapes. Our algorithm successfully performs the registration as shown on the right. The models were downloaded from the AIM@SHAPE Repository – http://shapes.aim-at-shape.net/*

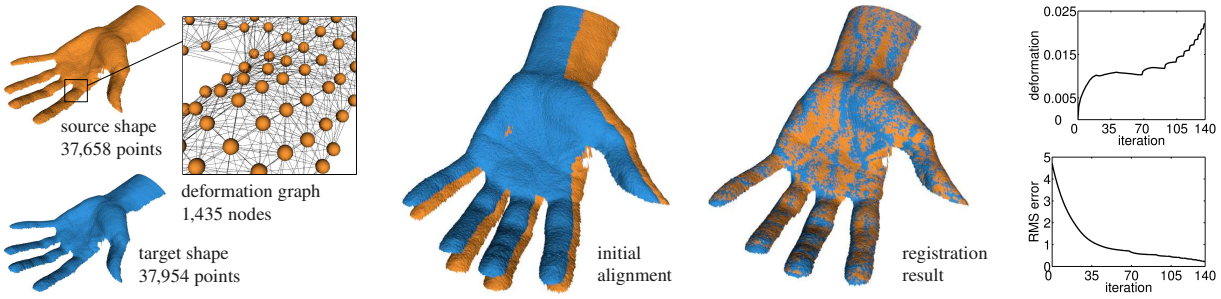


Figure 5: *Registering two range images representing the front part of the same hand in two different poses. The data sets were obtained with a 3D geometry scanner [WLG07] and are publicly available on the authors webpage.*

current frame is used as initialization for the next one. The sequence consists of 100 frames. Fig. 10 exemplary shows some frames and the corresponding registration results.

Comparison Finally, we compare the performance of our method (both registration quality and runtime) with the performance of several state-of-the-art non-rigid registration algorithms: the softassign + deterministic annealing (SDA) approach [CR03], the kernel correlation-based (KC) method [TK04], the coherent point drift (CPD) algorithm [MS10] and the Gaussian mixture models-based (GMM) algorithm [JV11]. Note that the KC [TK04] and the GMM [JV11] methods can perform the registration us-

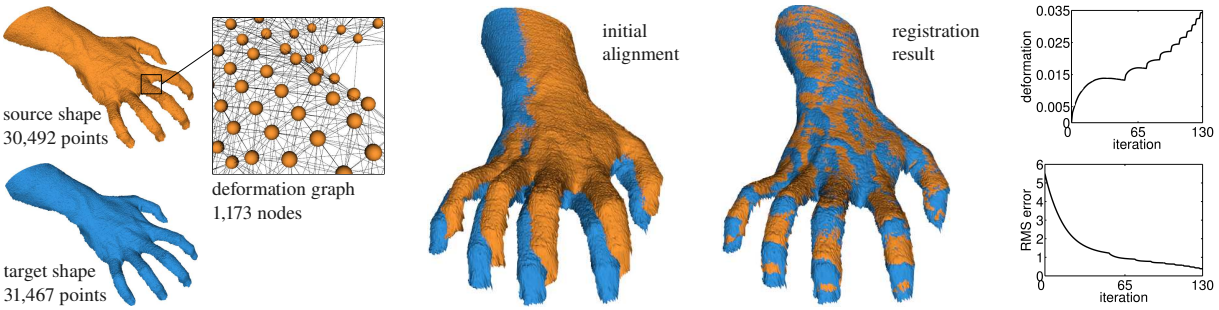


Figure 6: *Range scans representing the back part of the same hand in two different poses. The poses differ not only by the local bending deformation of the fingers but also by a global translation. The data sets were obtained with a 3D geometry scanner [WLG07] and used in [SAY*09].*

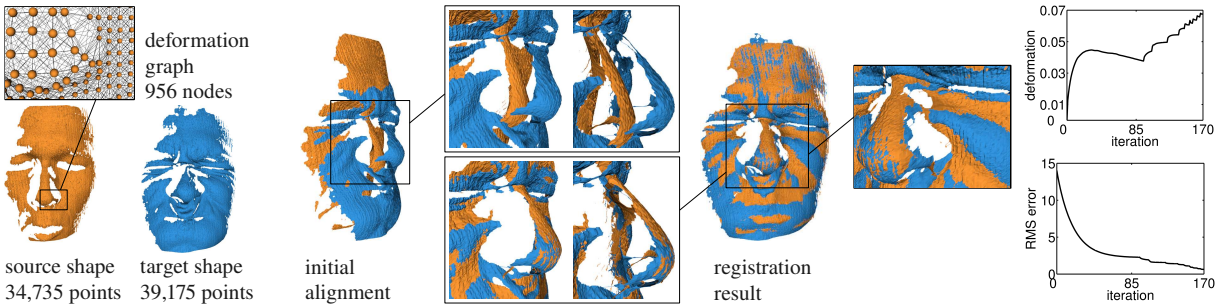


Figure 7: *Registering two facial expressions. Note that the scans are noisy and incomplete. Our methods correctly aligns the shapes even in areas of low overlap.*

ing two different deformation models, namely, thin plate splines (TPS) and Gaussian radial basis functions (GRBF). This effectively results in six different registration methods which we will denote as follows: SDA+TPS is the abbreviation for [CR03], CPD+GRBF stands for [MS10], KC+TPS, KC+GRBF denote [TK04], and GMM+TPS, GMM+GRBF stand for [JV11], depending on which deformation model is employed.

We use the implementation of the above mentioned methods, publicly available on http://gmmreg.googlecode.com, and run them on the same hardware and with the same data sets as our algorithm. The quality of the registration computed by the algorithms is compared using the source shape deformation measure (14) and the RMS error (15). Table 1 shows the results of the quality comparison. Note that our algorithm outperforms the others since it produces a lower RMS error for virtually the same source shape deformation. The results of the runtime comparison are summarized in table 2. Our algorithm clearly outperforms all six methods with the difference in processing time being up to two orders of magnitude.

	data set				
method	Fig. 5	Fig. 6	Fig. 7	Fig. 8	Fig. 9
GMM+TPS	497	315	273	332	453
GMM+GRBF	361	244	237	269	267
SDA+TPS	1004	621	501	642	1033
CPD+GRBF	4389	2443	1368	2269	5952
KC+TPS	491	314	273	331	451
KC+GRBF	361	243	237	267	267
our alg.	14	13	11	18	21

Table 2: *Computation time (in seconds) taken by our algorithm and six state-of-the-art approaches for the registration of the scans presented in the paper. Our method clearly outperforms the others.*

6. Conclusions

In this paper, we proposed an efficient algorithm for deformable registration of 3D shapes. We focused on modeling as-rigid-as-possible shape deformations augmented with local scale. In contrast to many recent methods, our approach is not formulated within a general-purpose optimization framework. The minimization of high-dimensional, nonlinear cost functions is computationally very demanding

	data set									
method	Fig. 5		Fig. 6		Fig. 7		Fig. 8		Fig. 9	
GMM+TPS	0.04	0.6	0.05	0.8	0.10	1.4	0.08	1.4	0.06	2.1
GMM+GRBF	0.03	0.8	0.03	0.9	0.07	2.4	0.05	2.2	0.06	2.5
SDA+TPS	0.09	0.7	0.09	0.7	0.11	1.2	0.08	1.3	0.08	0.9
CPD+GRBF	0.30	1.4	0.30	1.2	0.20	1.5	0.30	3.6	0.20	1.7
KC+TPS	0.04	0.6	0.05	0.8	0.10	1.3	0.08	1.4	0.06	1.8
KC+GRBF	0.03	0.8	0.03	0.9	0.09	2.2	0.06	2.2	0.07	2.4
our alg.	0.02	0.2	0.03	0.4	0.07	0.6	0.04	0.7	0.07	0.5

Table 1: *Comparing the quality of the registration computed by our algorithm and six state-of-the-art approaches for the scans presented in the paper. The first number in each table cell gives the source shape deformation (Eq. (14)) and the second one gives the RMS error in millimeters (Eq. (15)). Our method provides the most precise alignment for a low shape deformation.*

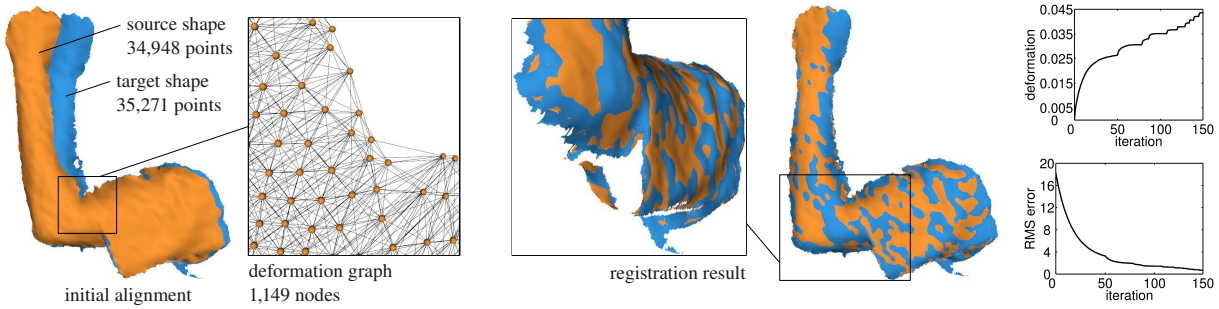


Figure 8: *Registration of a bending arm.*

since it involves the repeated solution of large linear systems. Instead, we rely on a simple and effective numerical integration scheme and solve an ODEs system which models the non-rigid motion of a source shape towards a target shape. The ODE we proposed is based on a correspondence field and a regularization term. Preliminary correspondences are estimated with a closest-point search and further refined with an efficient smoothing procedure. The regularizer is a generalization of the rigid shape matching technique recently developed in the context of physically plausible deformation modeling. We experimentally validated our method on a variety of real range scans and demonstrated that it performs well on noisy and incomplete data. Finally, an experimental comparison to six state-of-the-art approaches showed that the proposed algorithm outperforms them in terms of both registration quality and processing time.

Acknowledgments This work has been funded by the European Commission's Seventh Framework Programme as part of the project GRASP (IST-FP7-IP-215821).

References

[ACP03] ALLEN B., CURLESS B., POPOVIC Z.: The Space of Human Body Shapes: Reconstruction and Parameterization from Range Scans. *ACM TOG 22*, 3 (2003).

[AMCO08] AIGER D., MITRA N. J., COHEN-OR D.: 4-Points Congruent Sets for Robust Pairwise Surface Registration. In *ACM SIGGRAPH* (2008).

[ARV07] AMBERG B., ROMDHANI S., VETTER T.: Optimal Step Nonrigid ICP Algorithms for Surface Registration. In *CVPR* (2007).

[ASK*05] ANGUELOV D., SRINIVASAN P., KOLLER D., THRUN S., RODGERS J., DAVIS J.: SCAPE: Shape Completion and Animation of People. *ACM TOG 24*, 3 (2005).

[BBK06] BRONSTEIN A. M., BRONSTEIN M. M., KIMMEL R.: Generalized Multidimensional Scaling: A Framework for Isometry-Invariant Partial Surface Matching. *PNAS 103*, 5 (2006).

[BK10] BRONSTEIN M. M., KOKKINOS I.: Scale-Invariant Heat Kernel Signatures for Non-Rigid Shape Recognition. In *CVPR* (2010).

[BM92] BESL P., MCKAY N.: A Method for Registration of 3-D Shapes. *IEEE TPAMI 14*, 2 (1992).

[BPWG07] BOTSCH M., PAULY M., WICKE M., GROSS M. H.: Adaptive Space Deformations Based on Rigid Cells. *Comput. Graph. Forum 26*, 3 (2007).

[BR04] BROWN B., RUSINKIEWICZ S.: Non-Rigid Range-Scan Alignment Using Thin-Plate Splines. In *3DPVT* (2004).

[BR07] BROWN B., RUSINKIEWICZ S.: Global Non-Rigid Alignment of 3-D Scans. *ACM TOG 26*, 3 (2007).

[CM91] CHEN Y., MEDIONI G.: Object Modeling by Registration of Multiple Range Images. In *ICRA* (1991).

[CR03] CHUI H., RANGARAJAN A.: A New Point Matching Algorithm for Non-Rigid Registration. *CVIU 89*, 2-3 (2003).

[CZ08] CHANG W., ZWICKER M.: Automatic Registration for Articulated Shapes. *Comput. Graph. Forum 27*, 5 (2008).

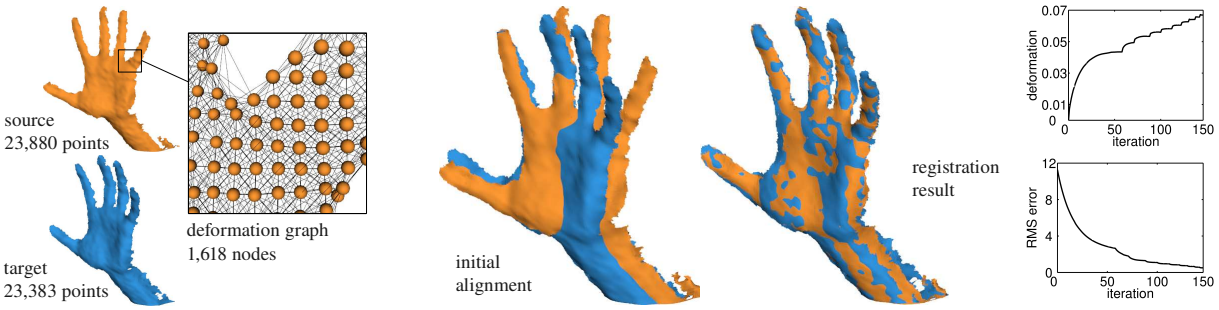


Figure 9: *Registering a closing and rotating hand.*

Figure 10: *Frame 0 is registered sequentially to the other frames which are part of a range image sequence of a closing hand. The sequence was obtained with a 3D geometry scanner [WLG07] and can be downloaded from the webpage of the authors.*

[EK03] ELAD A., KIMMEL R.: On Bending Invariant Signatures for Surfaces. *IEEE TPAMI 25*, 10 (2003).

[GMGP05] GELFAND N., MITRA N., GUIBAS L., POTTMANN H.: Robust Global Registration. In *Eurographics Symposium on Geometry Processing* (2005).

[IGL03] IKEMOTO L., GELFAND N., LEVOY M.: A Hierarchical Method for Aligning Warped Meshes. In *3DIM* (2003).

[JV11] JIAN B., VEMURI B.: Robust Point Set Registration Using Gaussian Mixture Models. *IEEE TPAMI* (2011).

[KS05] KRAEVOY V., SHEFFER A.: Template-Based Mesh Completion. In *Symposium on Geometry Processing* (2005).

[LSP08] LI H., SUMNER R. W., PAULY M.: Global Correspondence Optimization for Non-Rigid Registration of Depth Scans. *Comput. Graph. Forum 27*, 5 (2008).

[MHTG05] MÜLLER M., HEIDELBERGER B., TESCHNER M., GROSS M. H.: Meshless deformations based on shape matching. *ACM TOG 24*, 3 (2005).

[MS09] MYRONENKO A., SONG X. B.: On the Closed-Form Solution of the Rotation Matrix Arising in Computer Vision Problems. *CoRR abs/0904.1613* (2009).

[MS10] MYRONENKO A., SONG X. B.: Point Set Registration: Coherent Point Drift. *IEEE TPAMI 32*, 12 (2010).

[PB09] PAPAZOV C., BURSCHKA D.: Stochastic Optimization for Rigid Point Set Registration. In *ISVC* (2009).

[PMG*05] PAULY M., MITRA N. J., GIESEN J., GROSS M. H., GUIBAS L. J.: Example-Based 3D Scan Completion. In *Symposium on Geometry Processing* (2005).

[RBBK10] RAVIV D., BRONSTEIN M. M., BRONSTEIN A. M., KIMMEL R.: Volumetric Heat Kernel Signatures. In *3DOR* (2010).

[RJ07] RIVERS A. R., JAMES D. L.: FastLSM: Fast Lattice Shape Matching for Robust Real-Time Deformation. *ACM TOG 26*, 3 (2007).

[SAY*09] SAGAWA R., AKASAKA K., YAGI Y., HAMER H., VAN GOOL L.: Elastic Convolved ICP for the Registration of Deformable Objects. In *3DIM* (2009).

[SOG08] STEINEMANN D., OTADUY M. A., GROSS M.: Fast Adaptive Shape Matching Deformations. In *SCA* (2008).

[SSP07] SUMNER R. W., SCHMID J., PAULY M.: Embedded Deformation for Shape Manipulation. *ACM TOG 26*, 3 (2007).

[TK04] TSIN Y., KANADE T.: A Correlation-Based Approach to Robust Point Set Registration. In *ECCV* (2004).

[WAS10] WUHRER S., AZOUZ Z. B., SHU C.: Posture Invariant Surface Description and Feature Extraction. In *CVPR* (2010).

[WLG07] WEISE T., LEIBE B., GOOL L. J. V.: Fast 3D Scanning with Automatic Motion Compensation. In *CVPR* (2007).

[WOR10] WANG H., O'BRIEN J., RAMAMOORTHI R.: Multi-Resolution Isotropic Strain Limiting. *ACM TOG 29*, 6 (2010).

[WSB09] WUHRER S., SHU C., BOSE P.: Posture Invariant Correspondence Of Triangular Meshes In Shape Space. In *3DIM* (2009).

[WSBA07] WUHRER S., SHU C., BOSE P., AZOUZ Z. B.: Posture Invariant Correspondence of Incomplete Triangular Manifolds. *International Journal of Shape Modeling 13*, 2 (2007).

[WZL*10] WU H.-Y., ZHA H., LUO T., WANG X.-L., MA S.: Global and Local Isometry-Invariant Descriptor for 3D Shape Comparison and Partial Matching. In *CVPR* (2010).

[ZWW*10] ZENG Y., WANG C., WANG Y., GU X., SAMARAS D., PARAGIOS N.: Dense Non-Rigid Surface Registration Using High-Order Graph Matching. In *CVPR* (2010).

Framework for Consistent Maintenance of Geometric Data and Abstract Task-Knowledge from Range Observations

Juan Carlos Ramirez and Darius Burschka

***Abstract*— We present a framework for on-line exploration of object attributes from range data designed to include the cognitive aspects for surprise detection. In this framework we introduce a layered representation of the environment that couples the pure geometric 3D representation of the world to the abstract knowledge about the structures in the scene. This knowledge in the higher layer represents a-priori known, task-relevant information about structures in the world like mass, handling properties and grasping points being examples in the case of a manipulation task. The coupling of abstract knowledge to the geometry in the dual layered structure of our map helps to ensure consistency of the representation.**

The focus of the paper is on data association of dense 3D points from range sensors. We introduce a z-buffered re-projection method as a way to filter outlier information in sensor readings and present our technique for fusion based on the uncertainties in the map representation and the current observation. In contrast to common registration methods, our approach does not store the data as one rigid model but as a set of independent point clusters (foreground) embedded in a 3D point cloud of the supporting structure (background). This allows us to cope with dynamic changes in the world. We register the incoming data not rigidly to the entire map but we update independently the pose of single objects represented in our hierarchical model. The fusion approach combines a local-heuristic with a global-robust procedure and the correspondence search cost of $\mathcal{O}(nm)$ is reduced to a set of m sub-searches with linear cost each. The benefit of the re-projection is twofold: it helps speeding up the point matching search by ordering the 3D data according to the manner they might have been captured and making the matching process robust by filtering out outliers and occluded object parts. We present the theoretical framework and we validate our approach on range data from a binocular stereo setup.

I. INTRODUCTION

As with humans, when it comes to interacting effectively with its environment, the first step for a robot is to determine *what is where*, the next step would imply to determine *how*. These steps involve the recognition and localization of the objects a robot can interact with and based on these assumptions and a given task determine the strategies of interaction to be carried out. Furthermore, mission and path planning for mobile systems require a knowledge about the geometric structure of the world. This information can be provided a-priori from CAD models or explored with the sensors of the robot. There exist systems like KARTO from SRI [1], which stores the world as 2D maps to be able to navigate in them without any additional knowledge. On the other hand, manipulation tasks usually deal with single objects with, most of the time, known geometries. An exploration of the environment for manipulation is a challenging tasks that we aim to support with our approach. Our framework has the goal to support manipulation tasks in addition to traditional mobile exploration, therefore, a 3D representation of the world is required. An on-line exploration allows to update the map representation to the most current state. In a typical configuration, we assume that three-dimensional data along with the estimated position of the sensor in the world frame is passed to the map. In Fig. 1 we present an overview of our layered framework. From a range image of a scene we build first a geometric representation by means of an octree, the higher layer in the map is represented by a tree of blobs or blobtree, which contains a set of 3D blobs as tentative object candidates with the capacity to incorporate abstract knowledge or properties of each blob.

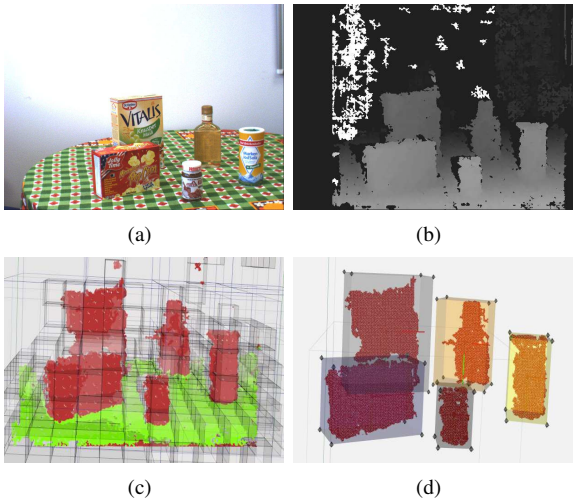


Fig. 1. Data abstraction in the dual layered framework. (a) The scene image, (b) its corresponding range image. (c) Octree, geometric layer representation, (d) tree of blobs, or *blobtree*, representation of the scene.

A. Motivation

The recent development in the area of range measuring devices allows a cheap perception of the three-dimensional information from the environment. While the sensors become cheaper with steady increasing robustness, the reconstructed

This work was supported by a DAAD-Conacyt Interchange Program under grant code number A/06/13408 and also partially supported by the European Communitys Seventh Framework Programme FP7/2007-2013 under grant agreement 215821 (GRASP project).

Juan Carlos Ramirez and Darius Burschka are with the Machine Vision and Perception Group, Department of Informatics, Technische Universitaet Muenchen. 85748 Garching b. Muenchen, Germany ramirezd@in.tum.de, burschka@cs.tum.edu

[1] http://kartorobotics.com

data provides only very limited information about the actual independent objects in the scene and their properties. Since video-based devices provide only a limited field of view and due to occlusions in the scene, the fusion of the sensor data from consecutive views becomes a very challenging task. The still poor sensor accuracy requires a continuous refinement of the position of the reconstructed objects in the map, especially for distant objects. Partial occlusion of the objects and lack of knowledge about the rigid structures in the environment can lead to distortion of the fused information, where object pose becomes partially corrected while the occluded parts remain untouched. Additionally, the current development in the scene perception requires not only a modeling of the static scene structure but requires additional parameters like motion parameters, physical properties to be maintained in the map. We address this problem in our framework, where we introduce a two-layer representation of the environment, which stores in addition to the pure geometry information also additional information about compact connected components observed as separated entities in the world. The abstract layer allows also to store additional information which is not important for the map fusion but which may be useful for the applications using this information, like manipulation or navigation systems. Because we are primarily concerned on resolving *what is where* as the first step for a intelligently interaction, in this work we focus basically on the detection and isolation of 3D blobs as potential heterogeneous objects candidates.

In the next section we discuss the theory and praxis related to this work. In section III our approach is described and its fundamental parts are delineated, and tested in section IV. The analysis and validation are addressed in section V.

II. Theoretical Background and Related Work

The main objective of a Data Fusion System (DFSys) is to combine and integrate data from different (types of) sensors or multiple observations from the same sensor in order to understand the phenomenon under observation and to increase the confidence of a observation when several sensor readings confirm that observation or state; Under a DFSys architecture [1], two basic functional blocks are the data alignment and the data association steps. The former consists in transforming the data received from the observations into a common spatial and temporal reference frame, it comprises basically coordinate and time transformations. The latter is concerned with correlating the multiple observations and determine whether a group of those belong to a new or the same event or target. Roughly speaking, given two noisy data sets, the Data Association (DA) problem is defined as that of finding for each point in one of the sets the appropriate corresponding matching point in the other set. DA problems arise mainly in registration systems in which new observations have to be integrated to previous ones. Two examples of theses systems are the tracking of objects or features [2] and the Simultaneous Localization and Mapping (SLAM) [3] [4] dilemma. Some authors have tackled the SLAM DA as a probabilistic issue [5] [6] and made used of family of filters to deal with the uncertainty in the prediction-correction model of the Kalman filter [7] [8]. There exists another branch of techniques to treat the uncertainty in DA that do not rely on the Bayesian filtering scheme but instead on statistical and geometric analysis like the use of Mahalanobis distance metric (MD) [9] and the Nearest Neighbor algorithm. In the context of 3D scene modeling with stereo cameras, the work described in [10] is based on stereo-SIFT features for plane extraction, object recognition and scene registration. Although in their work they deal with reconstructed stereo points and scene registration they do not consider the uncertainties in the stereo reconstruction and the spatial distribution of the points in order to identify heterogeneous objects; the approach is based on recognizing objects with already known object models in a database. After the objects in the scene are recognized and replaced by their corresponding models, an octree is used to detect possible obstacles before interaction. Since its introduction by Chen and Medioni [11], and Besl and McKay [12], the Iterative Closest Point (ICP) algorithm has been the most popular method for rigid point registration and a variety of improvements has been proposed in the literature. A good summary as well as results in acceleration of ICP algorithms have been given by Rusinkiewicz and Levoy [13]. A major drawback of ICP and all its variants is that they assume a good initial guess for the pose of the data point set (with respect to the model). This pose is improved in an iterative fashion until an optimal rigid transformation is found. The quality of the solution heavily depends on the initial guess. Furthermore, the methods compared in [13] use local surface features like surface normals which cannot be computed very reliably in the presence of noise. Another drawback of ICP is that insufficient results are obtained in the presence of noisy points and outliers. Although in our approach a pre-filtering procedure is performed, the presence of noisy points still remain throughout the scene.

III. Approach

A general overview of the system can be seen in Fig. 2. The scheme shows the data flow and the relations between the principal components and procedures that are explained in this section.

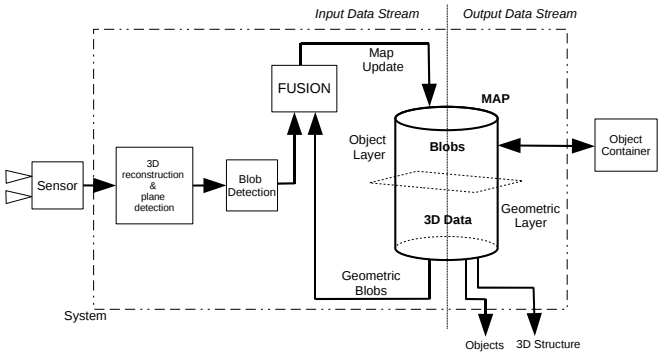


Fig. 2. Principal components of our approach.

A. Object Container

The Object Container (OC) is the model in which the properties of the potential object candidates are collected, Fig 3. Adding new properties to this model means the acquisition of new knowledge about an actor in the scene. The acquisition of these properties can arrive from different sources: a-priori properties, or learned, extracted properties, acquired by observation, experience or interaction [14]. In this approach, the OC is represented by a 3D blob that contains information about the spatial structure of an object candidate like dimensions and orientation.

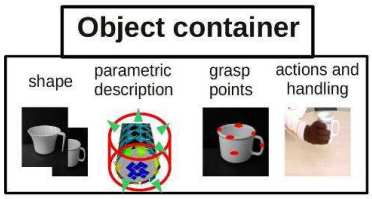


Fig. 3. The Object Container (OC) collects the a-priori known information and new extracted properties about an actor or object in a scene.

B. 3D Stereo Reconstruction and Plane Detection

With a calibrated stereo camera system we infer the depth of 3D point by computing the disparity value $d = u_l - u_r$ between two corresponding imaged points (u_l, v_l) and (u_r, v_r) lying in the left and right camera image plane respectively. Given a point in the space $P_i = (X_i, Y_i, Z_i)$, its estimated depth is given by $z = (b/d) \cdot f$, where f is the focal length of the cameras (with similar internal parameters) and b is the baseline. Any point $p_i = (\bar{x}_i, \bar{y}_i, d_i)$ in disparity space has an associated reconstructed 3D point $\bar{p}_i = (\bar{x}_i, \bar{y}_i, \bar{z}_i, \Lambda_i)$ where $\Lambda_i = diag(\sigma_x^2, \sigma_y^2, \sigma_z^2)$ is the covariance matrix representing the uncertainty of the 3D reconstruction in each axis:

$$\begin{aligned} \sigma_z^2 &= \tfrac{b \cdot f}{p_x \cdot d} \cdot \tfrac{\Delta d}{(d - \Delta d)} \\ \sigma_x^2 &= \tfrac{\Delta d \cdot p_x}{f} \cdot \bar{z} + \tfrac{(u + \Delta d) \cdot p_x}{f} \cdot \sigma_z^2 \\ \sigma_y^2 &= \tfrac{\Delta d \cdot p_x}{f} \cdot \bar{z} + \tfrac{(v + \Delta d) \cdot p_x}{f} \cdot \sigma_z^2 \end{aligned} \quad (1)$$

where p_x is the pixel dimension and Δd is the disparity error. The first step after the 3D reconstruction is the detection of the supporting plane. This is done by applying the random sample consensus (RANSAC) [15] procedure. In this work only the supporting plane is considered due to its immediate spatial proximity to the entities of interest, see Fig.1(c).

C. Octree and Blob Detection

After plane detection, the 3D reconstruction is stored in an octree. This data structure allows to spatially order three-dimensional data in a hierarchical and recursive fashion by partitioning the space in octants or voxels. The octree represents the low layer of abstraction in our framework because the information stored in it still constitutes a simple group of raw points without any spatial relation connecting any of them. This relation is setup by searching and identifying the connected components (leaves) inside the octree. We perform this search by traversing the octree according to the Depth-First Search (DFS) algorithm. The octree resolution, i.e. the size of the tree's leaves, has to be modified in order to adapt the search to more dense scenes in which we obtain more data points within reduced spatial regions, see Fig4.

Fig. 4. Blob detection in two scenes with different spatial observation densities. First row shows the scene images and the corresponding segmentations are showed below them.

D. Fusion

Typically the output of registration methods, after minimizing the distance of entity correspondences between two sets, is a rigid spatial correction that is applied and affects to all entities in such sets uniformly. In stereo vision, the 3D reconstruction of entities does not present an uniform error distribution and applying such global correction might hinder a proper registration in some regions while improving it in others. In this framework the newly captured 3D data is associated to our map not as rigid entity but as a group of individual clusters, blobs. The fusion between new sensor readings and the map is performed in a blob-pair-wise manner considering the local measurement errors presented in that region and the association of their corresponding geometric entities. One of the main advantages of this dual-layered framework is that of updating/correcting the state of an individual blob by propagating to all its points any modification in its geometric structure and abstract knowledge attributes even in the presence of occlusion or partial visibility. In our approach two blobs are fused only if their points are associated, and this association occurs whenever the rules and steps explained later in this section are carried out.

1) Blobtree and Blob Management: The blobtree is an octree-like structure that represents in our framework the higher level layer of knowledge of the scene. Unlike an octree, in which only the leaves store information, the blobtree nodes are also able to store data. A 3D blob is assigned to the smallest node that can contain it. This means that larger blobs are placed in higher level nodes inside the tree structure. Before a blob is inserted, it is encapsulated in a rectangular box whose dimensions and orientations are computed through the Principal Component Analysis (PCA) procedure (Fig 5). Additional to storing, the blobtree helps in

the process of detection of overlappings between incoming blobs with the stored ones as well as in the blob management during the fusion process. Furthermore, the blobtree represents our final map whenever it is updated, this is, when there are no more blob fusions to be performed. Basically, our map $\mathbf{M}(k) = \{\mathbf{B}_j(k)\}$ at time k, is defined as the set of blobs $\{\mathbf{B}_j(k)\}$, where each blob $\mathbf{B}_j = \{p_i, P_i, CV_i\}$, is the set of captured points along with their spatial uncertainty and confidence value CV (Sec. III-D.3). The pseudo-code of the blob intersection search is outlined in Algorithm 1 and the blob fusion procedure in Algorithm 2. In Algorithm 1.5 the

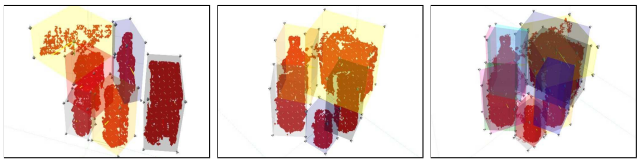


Fig. 5. Two 3D-blob groups of the same scene captured at different times: (left) $\mathbf{B}_j(k)$, (middle) $\mathbf{B}_i(k+1)$, (right) intersection observed from pose in $(k+1)$.

Algorithm 1 Blob Intersection Search

Require: Map $\mathbf{M}(k) = \{\{\mathbf{B}_j(k)\} \cup \{\mathbf{B}_i(k+1)\}\}$, with $j = 1, 2, \ldots, J$ and $i = 1, 2, \ldots, I$
Ensure: List of intersection containers $\Psi(k) = \{\psi_b\}$ where $\psi_b = \{\mathbf{B}_b(k) \cap \mathbf{B}_m(k+1)\}$ with $b \leq J$ and $m \leq I$

$\Psi(k) \leftarrow \emptyset$
for all $\mathbf{B}_j(k) \in \mathbf{M}(k)$ **do**
  $\psi_j \leftarrow \emptyset$ ▷ $\mathbf{B}_j(k)$'s intersection container
  for all $\mathbf{B}_i(k+1) \in \mathbf{M}(k)$ **do**
    if $\mathbf{B}_j(k) \cap \mathbf{B}_i(k+1)$ **then**
      $\psi_j \leftarrow \psi_j \cup (\mathbf{B}_j(k) \cap \mathbf{B}_i(k+1))$
    end if
  end for
  if $\psi_j \neq \emptyset$ **then**
    $\Psi(k) \leftarrow \Psi(k) \cup \psi_j$
  end if
end for
return $\Psi(k)$

condition refers to the intersection of 3D geometric bodies, in our particular case, to the collision of rectangular solids. For this kind of plane-faced geometries, it is sufficient to detect a contact of an edge on one object with a face of the other [16]. In Algorithm 2.5, however, the condition makes reference to the fusion of blobs through the association of their points. This condition and the call DOASSOC() are treated in the next subsection. The call UPDATELIST() is needed to update the references of the blobs as they are fused.

2) Z-buffered Re-projection and Data Association: By re-projection we mean the emulation of a camera screen containing only the visible points of the two blobs to be fused. To determine what point is visible when more than one is reprojected to the same pixel we use the z-buffer of each point and infer its depth, the point closest to the

Algorithm 2 Blob Fusion

Require: Map $\mathbf{M}(k)$ and List $\Psi(k)$
Ensure: Map $\mathbf{M}(k+1)$

$\Gamma \leftarrow \emptyset$ ▷ updated list of blobs
for all $\psi_b \in \Psi$ **do**
  $\mathbf{B}_b(k+1) \leftarrow$ EXTRACTFROMMAP($\mathbf{B}_b(k)$)
  for all $\mathbf{B}_m(k+1) \in \psi_b$ **do**
    if $\mathbf{B}_b(k+1) \cap \mathbf{B}_m(k+1)$ **then**
      $\mathbf{B}_b(k+1) \leftarrow$ DOASSOC($\mathbf{B}_b(k+1), \mathbf{B}_m(k+1)$)
    end if
  end for
  $\Gamma \leftarrow \Gamma \cup \mathbf{B}_b(k+1)$
  UPDATELIST($\Psi(k)$)
  UPDATELIST($\Gamma(k)$)
end for
$\mathbf{M}(k+1) \equiv \mathbf{M}(k) \leftarrow \mathbf{M}(k) \cup \Gamma$
return $\mathbf{M}(k+1)$ ▷ Map updated

camera is selected. In Fig. 6 two different re-projection screens are showed between two blobs captured from two different poses. In this association phase we imply that the observation-to-observation association is based on positional information that has been determined previously to some degree of certainty by an unbiased sensing process. At time $k+1$, a measurement $z_i(k+1)$ of a point $p_i(k+1)$ with covariance matrix $R_i(k+1)$, is normally distributed around its estimated value or state $\hat{z}_j(k+1|k)$ of $p_j(k+1|k) = T_k^{k+1} p_j(k)$ with covariance $P_j(k+1|k) = H_j P_j(k) H_j^T$, where T_k^{k+1} represents a rigid transformation to the current pose and H represents the measurement model matrix; the innovation or observation residual is defined as:

$$v_{ij}(k+1) = z_i(k+1) - \hat{z}_j(k+1|k) \tag{2}$$

the quantification of similarity between the observations is given by the Mahalanobis distance χ^2:

$$\chi^2_{ij} = v_{ij} S_{ij}^{-1} v_{ij}^T < \chi^2_\alpha \tag{3}$$

where

$$S_{ij} = P_j(k+1|k) + R_i(k+1) \tag{4}$$

is the innovation covariance. In order to find the correspondences between an observation $z_i(k+1)$ from a data set $\mathbf{S}_1 = \{p_{1,i}\}$ with $i = 1, 2, 3, \ldots, n$ and prediction state $\hat{z}_j(k+1|k)$ from a data set $\mathbf{S}_2 = \{p_{2,j}\}$ with $j = 1, 2, 3, \ldots, m$, that fulfill Eq.3, we re-project both sets to an emulated camera screen with camera model $C(k+1)$ in the current pose $T(k+1)$

$$(u_i, v_i) = C(k+1) z_i(k+1) \tag{5}$$

$$(u_j, v_j) = C(k+1) \hat{z}_j(k+1|k) \tag{6}$$

By the re-projection we have bucketed [17] the data points in screen pixels (u, v) and confined the matching search to a small pixel neighborhood around the predicted state represented by (u_j, v_j) and corresponding to the point $p_j(k+1|k)$. The matching search cost is reduced from $\mathcal{O}(nm)$ to $\mathcal{O}(km)$

where k is the number of neighbor pixels. Considering bucketing as a re-quantization loss of information can occur when more than one point is reprojected to the same pixel. For the case in which we re-project the current-pose observed points, (Eq. 5), the loss is negligible. For the case of Eq. 6, we examine the z-buffer of each point in order to determine which point is visible and which ones are occluded. We consider only the visible points of each set for association as we are interested in the restoration of what can constitute the shell or chassis of the actors in the scene. The validation

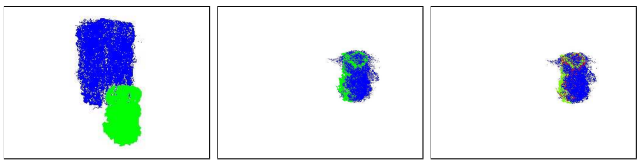


Fig. 6. Re-projection of objects of the same scene from two different poses. (Left) The objects are not intersected, (middle) they intersect, (right) they are fused (red points).

of correspondences by Eq.(3) constitutes the local-heuristic part of the process. After this, a new set of corrected, fused points are obtained:

$$p_l(k+1|k+1) = p_j(k+1|k) + K(k+1)v_{ij}(k+1) \quad (7)$$

with the gain K defined by:

$$K(k+1) = P_j(k+1|k)H_j S_{ij}^{-1} \quad (8)$$

and the covariance propagation

$$P_l(k+1|k+1) = P_j(k+1|k) - K(k+1)H_j P_j(k+1|k) \quad (9)$$

We obtain a set $\mathbf{L} = \{(\mathbf{S}'_1, \mathbf{S}'_2)_l\}$ of L matching-ordered subsets $(\mathbf{S}'_1, \mathbf{S}'_2)$, where $\mathbf{S}'_1 \subset \mathbf{S}_1$ and $\mathbf{S}'_2 \subset \mathbf{S}_2$ and a set of fused points $\mathbf{F} = \{f_l \equiv p_l(k+1|k+1)\}$ with $l = 1, 2, 3, \ldots, L$. As a global matching validation we apply RANSAC to the matched subsets in order to determine the two rigid transformations [18] that relate with the smallest error each of them to the corrected point set

$$\text{RANSAC}(\Sigma_i^2 = \sum_{s \in \mathbf{S}'_i, f \in \mathbf{F}} \|f - (\mathbf{R}_i s + \mathbf{T}_i)\|^2) \quad (10)$$

for $i = 1, 2$.

The application of these rigid transformations to the entire blob sets $\mathbf{S}_1, \mathbf{S}_2$, can also be considered as a blob state update propagation in a static environment since they modify spatially the blob data points to the current state.

3) Map Maintenance: The map maintenance is based on the degree of credibility or confidence value assigned to each point during the association process. With each re-projected point to the current camera frame we can state that it is inside the current scope and with its z-buffer we can infer whether it is visible or not. This leads to the next analysis based on the current observed point $p_i(k+1)$, for simplicity $p_{1,i}$; similar situations would result based on the predicted point $p_j(k+1|k)$, here $p_{2,j}$.

$p_{1,i}$ is visible and ...

- it has to be visible, i.e. there exists a corresponding point $p_{2,j}$ predicting the existence of $p_{1,i}$ to be matched. This constitutes a ***matching***. The confidence value of the fused point is increased.
- it has not to be visible, i.e. there is not a corresponding predicting point: $p_{2,j}$ is occluded or out of scope. $p_{1,i}$ is a new (noisy) point or outlier. This is an ***unpredicted observation***. $p_{1,i}$'s confidence value is initialized.

$p_{1,i}$ is not visible and ...

- it has to be visible, i.e. there is a corresponding prediction $p_{2,j}$ anticipating a new observation $p_{1,i}$ in its region: $p_j(k)$ was probably a noisy point or outlier. This is an ***unobserved prediction***. $p_{2,j}$'s confidence value is decreased.
- it has not to be visible. This affects to the rest of the points in- or outside the scope. Their confidence values remain the same.

The CV updating is based on the number of times a observed point is supported $so(k)$ or unsupported $uo(k)$ up through time k; In dynamic environments, this same procedure applied at blob level will be the base for an object-removal/addition detection (surprise detection), and with rates for learning α and forgetting β the rule to updating the CV will be: $CV_i(k) = 1 - e^{-(so_i(k)/\alpha - uo_i(k)/\beta)}$, as proposed in [19].

IV. Experiments and Results

Our stereo camera rig constitutes two Guppy F-080C IRF cameras. For the evaluation of our framework we took a sequence of observations at different positions and distances of the scene in Fig. 7(a). In Fig. 7(i) a view of the final blobtree is showed and in Fig. 7(j) shows the final map in which the blobs have been valuated by assigning confidence values to each point. Each point color in the blobs has an integer confidence value from 0 to 7 corresponding to the colors: white, yellow, light brown, orange, green, violet, purple and red, respectively. Two final valuated blobs are shown in Fig 8.

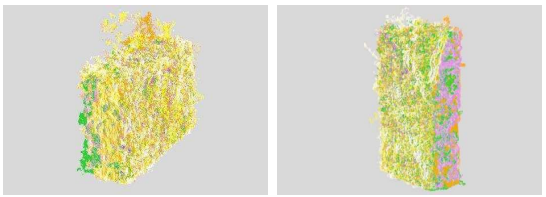


Fig. 8. Two blobs of the final map with assigned confidence values. Left, the pop corn box and right, the cereal box.

In a first experiment we are concern in evaluating how precise the points with confidence value assignments describe the actual size of a mapped object. We present the results for the two boxes of the scene presented in Fig. 7 which are the objects whose blob point readings undergo the widest range of error observations due to the variations in the proximity of the objects to the cameras as the cameras moved around the scene. In Tables I and II we summarize the

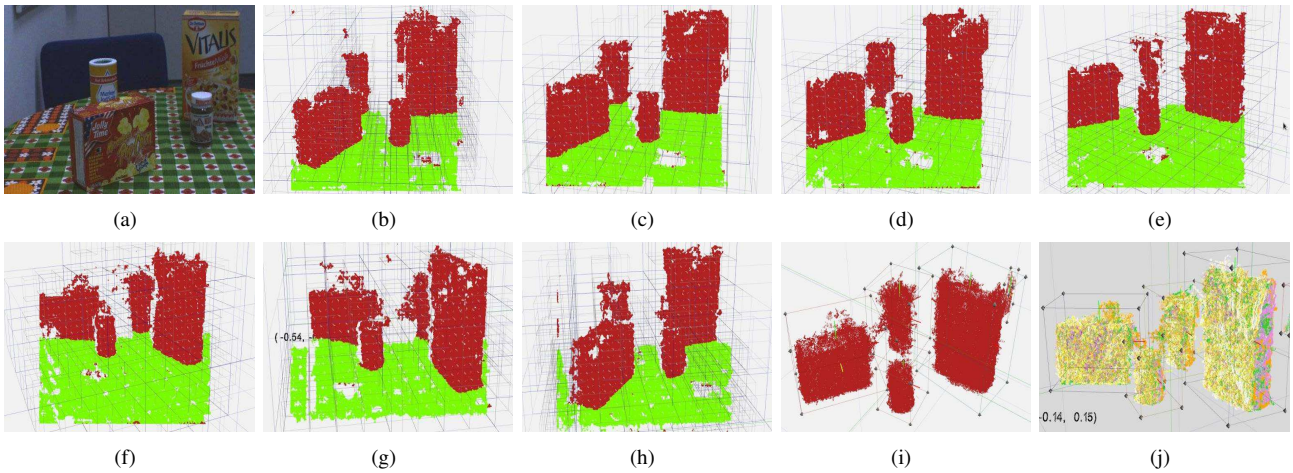


Fig. 7. Sequence of the captured scenes. (a) The scene, (b-h) the captured scenes in the geometric layer, (i) blobtree after the fusion of all scenes, and (j) valuated blobs in the final map.

results obtained with the valuated points of the cereal box blob and pop corn box blob respectively. The tables show by confidence value assignation the percentage of points that belong to that value, the measured size of the box these points define, and the root mean square deviation as a measure of error fitting between the valuated blob points with its corresponding actual-size box model yielded by the Iterative Closest Point (ICP) algorithm. Visually this fitting can be observed in the pictures of Fig.9. Similar results were obtained for the second box.

TABLE I

RESULTS OF THE CONFIDENCE VALUE (CV) ASSIGNMENTS TO THE CEREAL BOX BLOB POINTS AFTER A SERIES OF MAP UPDATES.

Cereal Box, actual size = 13.6 x 5.3 x 21.2 cm				
CV	Points(%)	Measured Size	RMS Error	Figure
7	0.52	13.67x4.83x20.23	0.000081	Fig.9(a)
6	2.17	15.36x8.21x22.21	0.000098	Fig.9(b)
5	12.7	16.01x12.13x22.43	0.000092	Fig.9(c)
4	56.53	18.36x11.38x23.16	0.004374	Fig.9(d)
3	12.92	18.98x13.14x23.12	0.005539	Fig.9(e)
2	6.41	18.37x10.06x24.29	0.005946	Fig.9(f)
1	4.29	18.26x10.38x24.38	0.006908	Fig.9(g)
0	4.46	19.1x9.19x24.45	0.005781	Fig.9(h)

TABLE II

RESULTS OF THE CONFIDENCE VALUE (CV) ASSIGNMENTS TO THE POP CORN BOX BLOB POINTS AFTER A SERIES OF MAP UPDATES.

Pop Corn Box, actual size = 16 x 5.75 x 11.7 cm			
CV	Points(%)	Measured Size	RMS Error
7	1.19	15.23x7.697x10.75	0.002548
6	3.61	17.69x8.91x11.01	0.002792
5	10.09	18.61x11.15x13.45	0.003715
4	54.46	20.09x12.29x15.90	0.004805
3	14.41	21.34x12.58x15.24	0.007764
2	8.9	19.45x11.76x14.60	0.006681
1	4.77	19.28x12.41x12.40	0.003378
0	2.55	18.90x11.26x12.45	0.003580

In a second experiment we want to observe the updating and the propagation of a blob state under partial occlusion

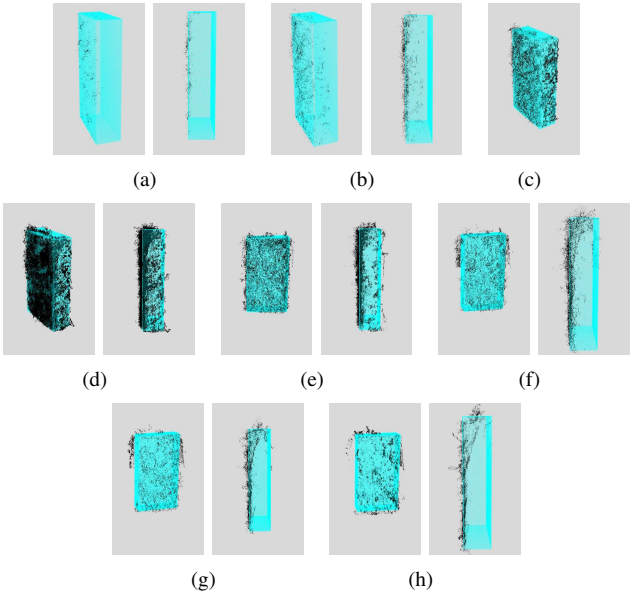


Fig. 9. Cereal box valuated points (black points) with different confidence values and the cereal model box (blue points) fitted by ICP .

but now in a context of a dynamic environment. For this, we took a sequence of measurements of an object before and after being occluded, the partially occluded object is slightly rotated after the occlusion, Fig. 10. The small updated rotation of the occluded blob points can be better observed by fitting its corresponding object model.

Finally, we present a first tentative output of our framework. Fig. 11 shows a 3D object-candidate detection screen of the scene in Fig. 1(a). The framework is able to segment the 3D scene by first subtracting the supporting plane and then clustering and characterizing the potential objects for further processing.

V. CONCLUSIONS

We have presented a dual layered framework for scene mapping aimed at covering not only the geometrical aspects

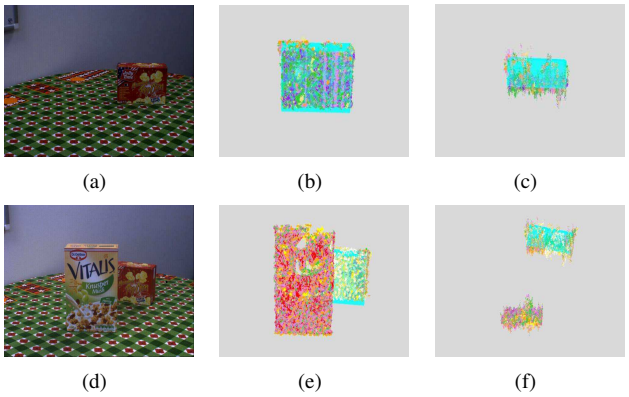


Fig. 10. Update and propagation of the state of a blob under partial visibility. (a) The pop corn box before the occlusion, (b) and (c) show the zoomed-in images of the valuated pop corn blob fitted with its object model before and after the occlusion from camera and bird's eye perspective respectively, (d) the pop corn box is occluded and rotated, (e) mapped blobs of the scene and (f) the rotation is better perceived by observing the fitted model from above.

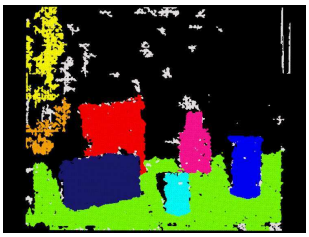


Fig. 11. 3D object-candidate detection screen: Labeling of the 3D information using the representation of the map observed from the camera perspective.

of the captured data but also being able to incorporate the abstract knowledge of the potential object-candidates represented by 3D blobs. Our approach decomposes each sensor observation into a set of independent blobs and registers two consecutive sensor readings in a blob-pair-wise fusion. This allows flexibility in the registrations as opposed to most rigid registration methods between entire images. With the re-projection we are able to obtain a linear cost in the matching search by looking for potential matching points in a reduced neighbor of pixels in the re-projected screen. With the results of Tables I and II we can see that the points with higher degrees of credibility are more precise and define better the surface of the objects. Criteria for stopping the updating and fusion in a certain blob have not been outlined yet. One criterion might be to stop such process as soon as a blob can be substituted for a well-fitted object model taken from a local data base, or for a basic geometric body. With the second experiment we observed that the new information added to a state of blob, like a change in the object pose, is also propagated to all its points even when the blob is partially visible. This fact is important not only when working in dynamic scenes but also in static ones where the state propagation of an object is updated as more partial views of the object are captured and fused. An immediate improvement to our approach is the addition of a procedure to split a blob in two or more blobs. In scenes in which objects are very close to each other, and due to the inherent noise in readings and in 3D stereo camera reconstruction, there always exists the possibility that two or more objects are captured as one single blob. The presented work constitutes our basic functional framework that describes the principal mechanisms for 3D mapping building and update, which are to be tested and improved in more different, complex and larger scenes.

REFERENCES

[1] A. Steinberg, "Data fusion system engineering," in *Information Fusion, 2000. FUSION 2000. Proceedings of the Third International Conference on*, vol. 1, July 2000, pp. MOD5/3 – MOD510 vol.1.

[2] Y. Bar-Shalom and T. E. Fortmann, "Tracking and data association." *New York: Academic Press*, 1988.

[3] A. Nuchter, K. Lingemann, J. Hertzberg, and H. Surmann, "6d slam with approximate data association," in *Advanced Robotics, 2005. ICAR '05. Proceedings., 12th International Conference on*, july 2005, pp. 242 –249.

[4] H. Durrant-whyte and T. Bailey, "Simultaneous localisation and mapping (slam): Part i the essential algorithms. robotics and automation magazine," *IEEE Robotics and Automation Magazine*, vol. 2, 2006.

[5] B. Habtemariam, R. Tharmarasa, T. Kirubarajan, D. Grimmett, and C. Wakayama, "Multiple detection probabilistic data association filter for multistatic target tracking," in *Information Fusion (FUSION), 2011 Proceedings of the 14th International Conference on*, july 2011, pp. 1 –6.

[6] I. J. Cox, "Probabilistic data association for dynamic world modeling: A multiple hypotesis approach." *Fifth Int. Conf. on Advanced Robotics*, vol. 10, no. 1, pp. 1287–1294, 1991.

[7] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960. [Online]. Available: http://www.cs.unc.edu/~welch/kalman/media/pdf/Kalman1960.pdf

[8] Y. Bar-Shalom, F. Daum, and J. Huang, "The probabilistic data association filter: Estimation in the presence of measurement uncertanty." *IEEE Control Systems Magazine*, vol. 29, no. 6, pp. 82–100, 2009.

[9] J. M. M. Montiel and L. Montano, "Efficient validation of matching hypoteses using mahalanobis distance." *Eng. Applicat. Artif. Intell*, vol. 11, no. 3, pp. 439–448, 1998.

[10] S. Lee, D. Jang, E. Kim, S. Hong, and J. Han, "A real-time 3d workspace modeling with stereo camera," in *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, Aug. 2005, pp. 2140 – 2147.

[11] Y. Chen and G. Medioni, "Object Modeling by Registration of Multiple Range Images," *Robotics and Automation, Proceedings., IEEE International Conference on*, vol. 3, pp. 2724–2729, 1991.

[12] P. Besl and N. McKay, "A Method for Registration of 3-D Shapes," *IEEE Trans. PAMI*, vol. 14, no. 2, 1992.

[13] S. Rusinkiewicz and M. Levoy, "Efficient Variants of the ICP Algorithm," *3DIM*, pp. 145–152, 2001.

[14] S. Petsch and D. Burschka, "Estimation of spatio-temporal object properties for manipulation tasks from observation of humans," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, pp. 192 –198.

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

[16] J. W. Boyse, "Interference detection among solids and surfaces," *Commun. ACM*, vol. 22, no. 1, pp. 3–9, 1979.

[17] Z. Zhuang and O. D. Faugeras, "Three-dimensional motion computation and object segmentation in a long sequence of stereo frames." *Intern. J. Comput*, pp. 211–241, 1992.

[18] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-d point sets," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 9, no. 5, pp. 698–700, September 1987.

[19] J. J. Leonard, H. F. Durrant-Whyte, and I. J. Cox, "Dynamic map building for an autonomous mobile robot." *In Proc. IEEE Int. Workshop on Intelligent Robots and Systems.*, pp. 286–298, 1990.

2011 IEEE/RSJ International Conference on
Intelligent Robots and Systems
September 25-30, 2011. San Francisco, CA, USA

Representation of Manipulation-Relevant Object Properties and Actions for Surprise-Driven Exploration

Susanne Petsch and Darius Burschka

***Abstract*— We propose a framework for the sensor-based estimation of manipulation-relevant object properties and the abstraction of known actions in a learning setup from the observation of humans. The descriptors consists of an object-centric representation of manipulation constraints and a scene-specific action graph. The graph spans between the typical places, where objects are placed. This framework allows to abstract the strongly varying actions of a human operator and to monitor unexpected new actions, that require a modification of the knowledge stored in the system. The usage of an abstract, object-centric structure enables not only the application of knowledge in the same situation, but also the transfer to similar environments. Furthermore, the information can be derived from different sensing modalities.**

The proposed system builds up the representation of manipulation-relevant properties and actions. The properties, which are directly related to the object, are stored in the *Object Container*. The *Functionality Map* links the actions with the typical action areas in the environment. We present experimental results on real human actions, showing the quality of the results, that can be obtained with our system.

I. Motivation

A robot should be able to learn unsupervised through the observation of human actions in its environment. Unfortunately, humans do not follow exact trajectories, while performing repetitive manipulation tasks. The system needs to be able to abstract the manipulation actions, in order to focus only on information, which is necessary to accomplish a manipulation or to cooperate with a human in a given environment. A mismatch between the expectation of the robot as an observer system and a current human action should occur only in situations, in which the change appears to be a result of a changed function or physical property of the object. We will call such a mismatch a *surprise* event in the following. A surprise event triggers the refinement or the modification of the stored information. Important examples are the following: Motion constraints are suddenly changed in the object transport phase (e.g., a cup carried always upright is now tilted arbitrarily); an object is suddenly placed on an unexpected place, e.g., a cup on the floor. These observations are usually an indication, that the physical properties of the object (e.g., the level of the liquid in the object) or their function (not a drinking cup, but a dirty dish) changed. This needs to be considered in the internal representation of the manipulation system.

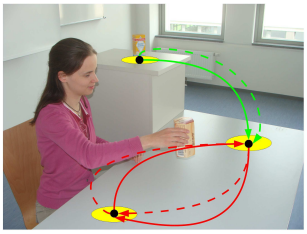


Fig. 1. The system creates an abstract map of possible manipulation actions and goals in the environment.

Our aim is to define a model, that allows to map different physical properties of the object to modifications in the handling properties. The model should efficiently abstract known actions applied to a given object, in order to be able to correctly predict the often strongly varying transport trajectories and goals. This second property of the system allows to reason about changes in the function of a specific object in a given environment. For example, a tool is not used for its specific purpose anymore, but just put away. The representation of the object specific properties and actions needs to be independent of exact Cartesian motion.

A-priori knowledge about an object class is stored in an *Atlas*, introduced in [1]. It contains already known properties of the objects as well as a-priori knowledge about the handling alternatives. Different handling properties might occur for the same object depending on the context. The correct alternative has to be selected based on the observation. This handling property may change over time. Such a modification is triggered by a mismatch between the expectation and the observation of the human action. The abstract knowledge from the Atlas can be mapped into the current context and stored in the *Working Memory*. The Working Memory is the explored representation of the current scene, where the a-priori (Atlas) knowledge gets registered to provide additional information about the structures observed by the system. The Atlas information gets mapped onto representations in the working memory, that describe the handling properties of objects and the observed functional relations between the typical places, where these objects can be found. In this paper, we focus on these representations of the manipulation-relevant properties in a given environment. The requirements on this representation have to be specified and the relevant knowledge has to be extracted from the observation in an appropriate manner. A very important aspect is, that not

This work was supported by the European Communitys Seventh Framework Programme FP7/2007-2013 under grant agreement 215821 (GRASP project)

Susanne Petsch and Darius Burschka are with the Machine Vision and Perception Group, Department of Informatics, Technische Universität München. 85748 Garching, Germany
{petsch|burschka}@in.tum.de

only the object itself (e.g., its properties or physical states) is defining the way, how it is manipulated, but also the location at which the manipulation is performed. Certain actions takes usually place at specific locations, which have certain properties. For example, washing the dishes is usually done in the sink and not on a flat table without any water source around. The conclusion is, that we need not only a collection of object properties, but also a map, which links locations in the environment to the specific way how objects are handled at these locations (see Fig. 1). It is important to notice, that we are not interested in the exact registration of the actions to the environment in the sense of navigation, but in an abstract representation of the functionalities in the environment. We are not using any semantic information about the environment. Furthermore, the system does not rely on any linguistic information.

The representation of the knowledge about the human actions is split into an object-centric representation, reflecting the physical properties of an object stored in an *Object Container*, and a *Functionality Map*, representing possible actions related to the environment. While the *Object Container* is linked only to the object, the *Functionality Map* is anchored to the geometric model of the environment. This framework allows us to limit unexpected events (surprise events), that cannot be explained with the current knowledge, to situations, where the physical state or the function of an object changed. The system is insensitive to variations in the execution of the same action. Predictions about the current situations are based on the information stored in Object Container or the Functionality Map. Therefore, mismatches between these predictions and observations occur just at an abstract level. They signal the *right moment* to update the stored information.

It is important to consider, that the robot might face different types of input like, e.g., vision data. This can be useful in a household environment. Further examples include procedural descriptions, which provide knowledge in topological form (e.g., the steps of a surgical procedure). Here, the trajectories of the human motions are neither used for workflow applications nor for planning of actions. We focus on the object-centric constraints and the object's function in the environment. Our system does not rely on a trajectory in a certain representation, like x,y,z-coordinates, or on a certain colored texture of an object. The properties acquired in our system have to be sufficient for a manipulation task. At the same time, they have to be generic and extractable from different sources.

The goal of this paper is the presentation of a system, which provides the manipulation-relevant knowledge in a way, such that the described requirements can be met.

The paper is structured as follows: our approach is presented in the next section. First, the manipulation-relevant object properties and the Functionality Map of the environment are described, followed by the presentation of the knowledge extraction. The results of the experiments with real human actions are described in Section III. We end with conclusions and future work.

A. Related Work

An extensive work exists in the field of imitation learning. In [2], HMMs are used for imitation learning of arm movements in manipulation tasks for humanoid robots, in order to achieve a human-like reproduction of the motions. It is important to point out the difference between our approach and non-object-centric approaches. Such approaches are, for example, imitation learning of motor skills or imitation of movements with Dynamic Movement Primitives (DMPs), which encode the trajectories themselves directly [3], [4]. This paper does not aim to encode the trajectories with, e.g., DMPs or using the model in [5]. Calinon *et al.* use imitation learning, in order to learn control strategies [6]. Approaches related to Reinforcement Learning [7] use the observations of humans as reward [8], [9] in the context of imitation learning. In contrast to theses approaches, our aim goes beyond simple imitation. We want to generalize the observation to cope with variations in repetitive human actions.

The intention in imitation tasks is addressed by Jansen and Belpaeme [10]. They train their agent in a grid with blocks in a computer simulation. In contrast, we deal with more complex, real-world environments and our system needs much less training instances than the one presented in [10]. A real-world example of capturing the user's intention about sequential task constraints is presented in [11]. Their system reasons about the existence of sequential dependencies of operations.

The object's motion has to be analyzed, in order to achieve a further understanding of the object's functionality. The effects on a manipulated object (position, orientation) are taken into account in [12]. The difference to our approach is, that we obtain information about the manipulation properties of the object and, furthermore, to the objects functionality in the environment. In [13], function from motion is analyzed for "primitive motions", which are translations or rotations relative to the main axes of primitive objects. Our approach goes further to more and more general manipulation-relevant object properties. In [14], functional roles of objects, like "pour out", have been explicitly introduced. These roles do not refer to the object's properties, which are directly observable during manipulation.

It should be to pointed out, that we are not interested in the reconstruction or the analysis of the environment, like [15]. We split the information in pure grasp related object-centric information and the information for trajectory planning represented by the Functionality Map. The relative/absolute position of objects to each other have been used for the consideration of the environment in manipulation properties in [15]. In [16], a perceptual space (for the color and shape object properties) and a situation space (for the displacement of the objects in the scene) are introduced. In contrast, the object properties in our paper are beyond the pure visual appearance of the object, since we are interested in the manipulation-relevant object properties (like motion constraints, and known linkages between geometric static occurrences of the object).

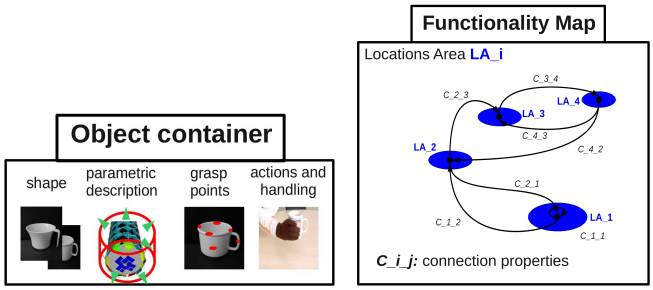


Fig. 2. Object Container and Functionality Map. The Object Container stores the object properties. The Functionality Map is an abstract representation of the manipulation-relevant operating areas in the environment.

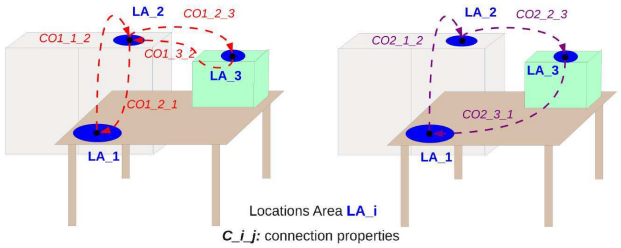


Fig. 3. Functionality Map of the environment for two exemplary objects.

II. Approach

An overview of the system is given in Fig. 2. It illustrates the *Object Container* with the object properties, and the representation of the actions in the *Functionality Map*. The map is represented by a graph, which encodes *Location Areas* and their connections between them.

A. Manipulation-Relevant Object Properties

Since we are interested in a general knowledge of the object properties, we do not want to list the x,y,z-coordinates of the recorded trajectory points, but the abstract handling properties important for grasp planning. The properties, we consider as important, are the variation of orientation, the maximal allowed acceleration, the grasp type allowing a successful grasp with a given manipulator, the mass and the center of mass. Some of these properties are not observable with a pure vision or tracking system. Therefore, the already described information database Atlas [1], which contains the "experience" (*a-priori information*), is used to provide initial information. The other properties need to be extracted, using, for example, a vision system.

The handling properties themselves are constraints, which limit the handling possibilities of the object in a certain situation. Our object-centric representation has the advantage, that the representation of a constraint does not rely on a specific Cartesian position in space. For a specific object, the internal properties, like fragile or liquid content, constraint merely the velocity and acceleration parameters independent of the position in the environment.

B. Functionality Map of the Environment

The Functionality Map provides information about possible trajectories in the environment. The first component of the Functionality Map are the Location Areas. These areas are the locations in the 3D space, where a manipulation sequence can start or end. We define explicitly location *areas* and not single points, since an object is usually placed in a certain area and not on one certain point in space. A Location Area does not necessarily imply, that the manipulated object is standing on a surface. A hand-over step (e.g., changing hands) can also establish a Location Area, which is, therefore, not connected to a surface, but to an area in space.

The connections between different Location Areas are the next component of the Functionality Map. A connection exists, if an action has been performed directly between both areas without visiting another one in between. It is important to consider, that a connection is directed. A connection itself stores the different manipulation properties of the actions, which are performed on this connection. The properties depend on different factors. The first factor are the objects themselves. The other factor are the different grasp alternatives, that can occur for each object. Two exemplary instantiations of a Functionality Map can be seen in Fig. 3.

The properties, which are stored in the Functionality Map, are the following:

- **pushed object vs. lifted object** - An object can be manipulated by lifting or by pushing it. A pushed object needs just to be pushed in the desired direction, whereas lifting an object requires an entire grasp planning (including knowledge about the object's weight).
- **arbitrary movement vs. constrained trajectory** - The trajectory between two Location Areas has either an arbitrary shape or it is the result of a constrained motion. A constrained motion connects the Location Areas in a direct manner, avoiding deviations. In contrast, an arbitrary movement is unconstrained.
- **connection relevance** - The connection relevance shows the probability of a connection property, based on the observed actions.
- **velocity constraints during pick-up** - The three phases defining an action introduced in [1] are used: the pick-up, the transportation and the placement phase. The maximal speed during the pick-up phase is stored as velocity constraint in the Functionality Map. It is an indicator for the difficulty to pick up the object.
- **grasp taxonomy** - The grasp type is mainly important for the pick-up and placement phase of the manipulation and not part of this paper. The grasp taxonomy we consider for the system is summarized in [17].
- **grasp approach vector** - The grasp approach vector is, similarly to the grasp type, mainly important for the pick-up and placement phase of the manipulation and is not part of this paper. The grasp direction is the direction, from which the object is grasped in the object-centric point of view.

The assignment of the properties to the Object Container or the Functionality Map depends on the type of the property. A property, which is related to the function in the environment, is assigned to the Functionality Map. For example, the velocity constraint during the pick-up is part of the Functionality Map, since the possible velocity constraint depends on the environment of the object (e.g., obstacles). In contrast, a property, which is directly related to the object and its state, is a part of the Object Container. An example for such a property is the maximal allowed acceleration for an object in a certain state (e.g., no high acceleration for a filled cup).

C. Acquisition of Knowledge

The presented Object Container and Functionality Map need to be filled with information. For example, a scene can be observed with a system, which provides a 6-DoF trajectory of the manipulated objects.

1) Object Container: The properties for the Object Container, which we want to consider in this paper, are the maximal acceleration value and the variation of observed orientation of the object during the manipulation.

a) Maximal Acceleration: The maximal acceleration value is approximated by the difference of two consecutive velocity samples, which, in turn, are computed as the difference of two consecutive samples.

b) Orientation: The observed orientation is determined from the given 6-DoF trajectory. Just the orientation change relative to the gravitational vector is of interest for the constraints in the manipulation task. The aim is to distinguish a motion with rotation from a motion without tilting. We use Hidden Markov Models (HMMs) [18] for the classification because of their ability of generalization. They are statistical classifiers, which use an observation sequence for the estimation of the underlying state-sequence. Moreover, they take into account knowledge of the past (previous state) in the sequential input. Discrete HMMs with $\lambda = (A, B, \Pi)$ are chosen. They comprise a transition probability matrix A, an observation symbol probability distribution matrix B and an initial state distribution $\mathit{\Pi}$.

First, the preprocessing takes place, until a codebook of the rotation information is built. An overlapping window of 400 ms with a 200 ms overlap (according to [19]) is applied on the sequential input. For each window, the angles between the axes of the current coordinate frame and the coordinate frame at the beginning of the manipulation are measured. Depending on the object and the way of recording its trajectory, different angular variations can occur for different objects. A relative amount of change is needed for each object. Therefore, the angles are normalized for each object with its maximum angle, occurring in all movements of the object. After this preprocessing, the collected data of the rotation information is clustered with the K-means algorithm [20] independent of its time of occurrence. The result is a 64 symbol rotation information codebook.

Then two HMMs (each with 10 states) are built, in order to classify the motions as ones which contain a rotation (λ_r) or as rotation-free ones (λ_{noR}). The transition and emission probabilities for each model are calculated with the maximum likelihood estimation, using the labeled training sequences.

For evaluation, the system receives test sequences, which are preprocessed as described above. The k-nearest-neighbors-method (knn) is used for the assignment to the corresponding symbols in each codebook. To evaluate the classification performance of the trained HMMs, the maximum log likelihood $log\ P(o_{test}|\lambda_i)$ of a given model λ_i is computed for each test sequence with observations o_{test} similar to [21]:

$$\lambda_r^* = \arg\max[\log\ P(o_{test}|\lambda_{noR}), \log\ P(o_{test}|\lambda_r)]\ . \quad (1)$$

2) Functionality Map:

a) Location Areas: The possible Location Areas have to be determined first. Therefore, the available trajectories are split up in single sequences. A sequence starts as soon as the human grasps the object. The end of the sequence is reached, when the hand and the object are not in contact any more. The collected 3D-points of the start and end positions are clustered. The resulting cluster-centers are the centers of the Location Areas. It is possible, that the system detects two Location Areas, which coincide in fact, but appear randomly as two. If these Location Areas are close to each other and have the same connection properties, they can be fused.

b) Connection Properties: The next step is the determination of the connections between the detected Location Areas and the corresponding properties of the connections for each object. The properties, we are using in this paper, are the distinction of a pushed vs. a lifted object, an arbitrary movement vs. a movement with a constrained trajectory, the velocity constraints during the pick-up phase and the connection relevance of a movement property on a certain connection. If possible, the grasp type of the manipulation is determined.

Pushed Object vs. Lifted Object: An object is pushed, if it is in contact with its background during the whole manipulation. In order to check this contact, the distance between the object and the supporting surface is measured along the normal vector of the surface (see [1] for the computation).

Arbitrary Movement vs. Movement with Constrained Trajectory: A Principle Component Analysis PCA (with rescaling) is performed for the distinction of an arbitrary movement and a movement with a constrained trajectory. The PCA is done on a 4.8 s window with a 2.4 s overlap, the resulting principal components are normalized. We check for arbitrary motion, where the motion has no major direction component, but the movement is relatively large in all directions. Therefore, we are especially interested in the third (and smallest) component of PCA, since it shows the variation of motion. If this component has a high amplitude, then all three principal components have relatively high amplitudes. In this case, the movement is large in all directions and it is an arbitrary movement. We define the amplitude of the third component as "high", if it meets one of two requirements. The first one

is a comparison with the main direction of motion (= the first principal component): If the magnitude of the first and the third component are relatively "close" to each other, there is hardly any main direction of the movement and the movement is arbitrary. "Close" means the following: The component of the smallest movement is multiplied with a factor (multiplication factor arbitrary-movement). This factor is the maximal ratio of the first and the third principal component among all arbitrary movements. It determines, how many times the largest movement is maximally allowed to be larger than the smallest movement to classify it still as an arbitrary motion. The second requirement for a "high" amplitude of the third component is occurring, when the this component is higher than a threshold (arbitrary-movement-threshold). The arbitrary-movement-threshold has to be chosen in the magnitude of the third principal component of the arbitrary movements. If all the described criteria are not met, the direction of the smallest motion is not high and the movement is a movement with a constrained trajectory.

Velocity constraints during the pick-up: The pick-up phase is defined manually with 50 samples from the starting position. The speed is computed for two consecutive samples.

Connection relevance: The connection relevance can easily be determined by dividing the number of occurrences of a certain movement property on a connection by the number of all movements on this connection.

Grasp Type: In the current implementation, grasp type is determined by manual labeling.

III. Results

The proposed system is tested on sequences (seq.) of real human actions. First, we test our system on external tracking data (subsection III-A). This data provides directly the 6-DoF trajectory of the tracked markers, which are placed on top of the manipulated objects (obj.). The system is also evaluated on stereo data directly from the manipulation system (subsection III-B).

A. Basic Results on Tracking Data

The tracking data is recorded with a marker-based IR tracking system[1] at 50 Hz. The data is preprocessed first: Each sample, which does not differ from the consecutive one, is deleted. The remaining motion corresponds to an at least minimal movement. Furthermore, the trajectories are smoothed with a 1.4 s moving-average-window, in order to eliminate high-frequency noise, which can especially occur at the beginning of the movement. After this basic preprocessing, the sequences vary between 4.3 s and 17.72 s, the average is 7.45 s (example in Fig. 4, *left*).

The sequences are recorded with four different objects: a milk carton, a spoon, a cup and a vase. The cup is grasped twice: at the handle and at the cylindrical part from the side. This leads to five "different" objects for the test. For each object, there are 18 different actions of a person, shown in Table I. The implementation is done in Matlab (Statistics Toolbox: PCA, HMM, K-mean algorithm. Bioinformatics Toolbox: knn-classification.).

[1]Advanced Realtime Tracking system. Advanced Realtime Tracking GmbH, *url:* http://www.ar-tracking.de/ .

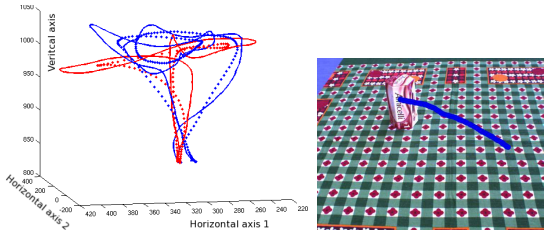


Fig. 4. *Left*: Arbitrary movements of the tracking data (in mm) for the cup in red (without rotation) and in blue (with rotation) (line = original movements, dotted line = result of the basic preprocessing). *Right*: Trajectory of a pushed object (seq. 1, vision data).

TABLE I

Left: Description of the seq. (Tracking data). *Right*: Further description of the four constrained trajectory (seq. 1-4, 5-8, 9-12, 13-16). Positions: *table-start* = pos. on the table, *box* = pos. on a box on the table, *corner* = corner of the table.

Seq	Movement	Rotation	Seq	Start Pos.	End Pos.
1-2	constr.: line		1	table-start	box
3-4	constr.: pushed		2	box	table-start
5-8	constr.: curve		3	table-start	corner
9-12	constr.: line	x	4	corner	table-start
13-16	constr.: curve	x			
17	arbitrary				
18	arbitrary	x			

The parameters and the initial values for the knowledge extraction (see Section II-C) are set as follows. The knn-assignment of a new value to a cluster in the rotation information codebook is done with k=3. The multiplication factor arbitrary-movement for the third component is 15, and the arbitrary-movement-threshold is 0.06. The height difference to the table is measured along its vertical axis for the distinction pushed vs. lifted object. The object is pushed, if its height difference to the table is not changing ($\pm 5\ mm$). The maximal acceleration is computed for a window of 8 ms. Furthermore, the initialization of the cluster for the build-up of the rotation information codebook is set. Otherwise, the results of the clustering are not always deterministic, even though they look mostly very similar. The initialization values are chosen between 0 and 1, since the input values are the normalized changes of the angles.

1) Object Container: A leave-one-out cross validation is made for the rotation classification. The results show, that 42 of 45 of the motions without tilting are correctly labeled, and 30 of 45 motions with titling are correctly classified (see Table II).

The final result of the Object Container can be seen in Table III. Acceleration classes are introduced, in order to make the Object Container more generic. The number of acceleration classes is set to three for illustration. Each class represents an approximately equal sized part of the achieved acceleration values. Table III shows the number of observations per acceleration class and the rotation-classification.

TABLE II

STATISTICAL RESULTS OF THE CLASSIFICATIONS.

Property: (T = Tracking data, V = Vision data)	Accuracy	True positive rate	True negative rate
Rotation (T)	80.0%	66.7%	93.3%
Pushed Object (T)	98.9%	100.0%	87.5%
Arbitrary Traj. (T)	94.4%	80.0%	96.3%
Rotation (V)	77.5%	93.8%	66.7%
Pushed Object (V)	95.0%	87.5%	96.9%
Arbitrary Traj. (V)	80.6%	100.0%	78.6%

TABLE III

RESULT: OBJECT CONTAINER. (R = MOTION WITH ROTATION).

Acceleration class	1		2		3	
Objects Tracking	no R	R	no R	R	no R	R
Milk	3	3	6	3	3	0
Spoon	3	1	3	4	4	3
Cup-handle	2	4	3	8	1	0
Cup	5	2	8	1	2	0
Vase	10	4	4	0	0	0
Objects Vision	no R	R	no R	R	no R	R
Object 1	1	1	0	1	2	5
Object 2	3	0	0	0	4	3
Object 3	2	1	1	1	1	4
Object 4	0	2	2	2	1	3

2) Functionality Map: All three used location areas have been correctly identified. A leave-one-out cross validation is done for the classification of the (non-)arbitrary movements (at first without the distinction of a pushed or lifted object). 8 of 10 arbitrary movements are correctly labeled, and 77 of 80 movements with a constrained trajectory are correctly classified. This shows, that the system performs definitely better than guessing. One has to consider for the true positive rate (see Table II), that there are just 10 arbitrary movements among all 90 sequences, leading to a significant influence of every mislabeled arbitrary movement.

The classification of the pushed vs. the lifted object is successful for all sequences except one spoon-sequence.

The results of the Functionality Maps show, that the system is able to handle some misclassifications, since the correct high connection relevance is gained for all objects except for the cup-handle. The best (= completely correct) results are achieved for the cup and the milk (Fig. 5, *left*). The worst result is the Functionality Map of the cup-handle, since it contains the highest number of misclassifications (two misclassifications) among all Functionality Maps. The misclassified arbitrary movements (red self-loop LA_2) and the misclassified movements with a constrained trajectory (magenta connection from LA_1 to LA_2) are drawn in Fig. 5 (*top, right*). The Functionality Maps of the other two objects have just one misclassification.

B. Results from a Vision System

The vision data is recorded with a Firewire Marlin FO46C camera at 30 Hz and an image size of 640x480 pixel (width x height). The trajectories are acquired as described

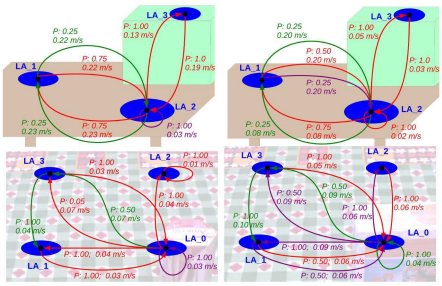


Fig. 5. *Top*: Functionality Maps of the tracking data (*left*: milk, *right*: cup-handle). *Bottom*: Functionality Maps of the vision data (*left*: obj. 1, *right*: obj. 2). Red arrow = constrained trajectory, green arrow = pushed obj., magenta arrow = arbitrary movement. P = probability.

TABLE IV

SEQUENCE PROPERTIES - VISION SYSTEM. THE START AND END POSITIONS ARE THE BOTTOM RIGHT (BR), THE BOTTOM LEFT (BL), THE TOP RIGHT (TR) AND THE TOP LEFT (TL) OF A TABLE.

Seq.:	Movement	Rotation	Start Pos.	End Pos.
1, 11, 21, 31	constr.: push		br	tl
2, 12, 22, 32	constr.: push		tl	bl
3, 13, 23, 33	constr.: curve		bl	br
4, 14, 24, 34	arbitrary		br	br
5, 15, 25, 35	constr.: curve		br	tl
6, 16, 26, 36	constr.: curve		tl	br
7, 17, 27, 37	constr.: curve	x	br	tr
8, 18, 28, 38	constr.: curve	x	tr	br
9, 19, 29, 39	constr.: curve	x	br	bl
10, 20, 30, 40	constr.: curve	x	bl	br

in [1]. The C++ Implementation of Hidden Markov Model by Dekang Lin [2] is (slightly modified) used for the implementation of HMMs. The PCA, the K-means algorithm and the knn-classification are done with OpenCV. Ten sequences are recorded with each of the used four objects. The properties of the sequences are listed in Table IV. An example is shown in Fig. 4 (*right*).

A basic preprocessing is performed (minimal movement $> 0.01/sample$, 140 sample moving-average-window) similarly to the tracking data. Furthermore, the first and last 20 samples are cut of, in order to deal with the arbitrary motions at the beginning and at the end of the sequences. The initialization and threshold values are set as for the experiment with the tracking data, except for the arbitrary-movement-threshold (0.06 for the vision data) and the window for the acceleration-computation (16 ms).

1) Object Container: The occurrence of (non-) rotation is correctly identified for 31 of 40 sequences (Table II). Eight sequences are mislabeled as sequences with rotations. All of them show, that one or both horizontal angles vary during the manipulation. The variations are not as strong as for most of the sequences with rotation, but it is still visible. Table III shows the final result of the Object Container.

2) Functionality Map: The Location Areas themselves are successfully determined. The assignment is successful

[2]Copyright (C) 2003 Dekang Lin, lindek@cs.ualberta.ca, *url:* http://webdocs.cs.ualberta.ca/ lindek/hmm.htm .

for 77 of 80 positions (96.3%). The misclassifications occur for the end positions of sequence 8, 21 and 33. These misclassifications are mainly caused by the z-components (the depth) of the end positions, which are closer to other Location Areas.

As the statistical measures in Table II show, the result of the distinction between a pushed object and an object, which is lifted for the movement, is remarkable. There is just one sequence mislabeled as pushed object, and one sequence mislabeled as raised object. The performance of the classification as arbitrary movement or as movement with a constrained trajectory achieves a true positive rate of 100.0%. Consequently, no arbitrary movement is mislabeled as non-arbitrary movement. Six sequences are misclassified as arbitrary movements instead of movements with constrained trajectory. These movements contain small parts with an arbitrary shape. The kind of grasp is analyzed according to [17]. All used grasps are power grasps with an abducted position of the thumb.

The Functionality Maps of object 1 and 4 have just one wrong assignment of an end location each, everything else is correct (see object 1 in Fig. 5, *left*). The Functionality Map of object 2 suffers mainly from misclassifications as arbitrary movements (see Fig. 5, *right*). One movement of object 3 can be seen a outlier, since its connection property, as well as the assignment of its end location, are wrong. Besides one further misclassified connection property, the Functionality Map of object 3 is correct.

IV. Conclusions and Future Work

The proposed system is developed for abstract representation of manipulation-relevant knowledge about objects. This system aims to monitor object properties and function in a given environment. The experiments on external tracking and vision data show, that the system can derive the knowledge from different sources. The presented system allows an efficient monitoring scheme for the detection of unexpected (surprising) event, that require an update of the information in the internal representation. The proposed framework allows to deal with the strong variations in actions performed by a human operator, reducing the number of false positive surprise events to a minimum. The proposed descriptors, consisting of an Object Container and a Functionality Map spanning typical object locations in a graph, allow a close monitoring of changes in a physical state of the object and its function in the environment.

The results from the vision system show, that a single trajectory is not enough to avoid a misclassification. However, an observation of multiple actions along a given edge of the Functionality Map allows a robust estimation. Our next goal is to focus more on unknown situations and environments. They provide new information to the system.

References

[1] S. Petsch and D. Burschka, "Estimation of Spatio-Temporal Object Properties for Manipulation Tasks from Observation of Humans," in *IEEE International Conference on Robotics and Automation*, Anchorage, USA, 2010, pp. 192–198.

[2] T. Asfour, P. Azad, F. Gyarfas, and R. Dillmann, "Imitation learning of dual-arm manipulation tasks in humanoid robots," *International Journal of Humanoid Robotics*, vol. 5, no. 2, pp. 183–202, 2008.

[3] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement Imitation with Nonlinear Dynamical Systems in Humanoid Robots," in *IEEE International Conference on Robotics and Automation*, Washington, DC, USA, 2002, pp. 1398–1403.

[4] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and Generalization of Motor Skills by Learning from Demonstration," in *IEEE International Conference on Robotics and Automation*, Kobe, Japan, 2009, pp. 763–768.

[5] K.Ogawara, J.Takamatsu, K.Kimura, and K.Ikeuchi, "Generation of a task model by intergrating multiple observations of human demonstrations," in *Proceedings of the IEEE Intl. Conf. on Robotics and Automation (ICRA '02)*, May 2002, pp. 1545–1550.

[6] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, and A. Billard, "Learning and Reproduction Gestures by Imitation," *IEEE Robotics and Automation Magazine*, vol. 17, pp. 44 – 54, 2010.

[7] R. S. S. und A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[8] D. Verma and R. P. N. Rao, "Imitation Learning Using Graphical Models," in *ECML 2007*, J. N. K. et al., Ed., vol. 4701. Lecture Notes in Artificial Intelligence, Springer-Verlag Berlin Heidelberg, 2007, pp. 757–764.

[9] G. Bombini, N. D. Mauro, T. M. A. Basile, S. Ferilli, and F. Esposito, "Relational Learning by Imitation," in *KES-AMSTA 2009*, A. H. et al., Ed., vol. 5559. Lecture Notes in Artificial Intelligence, Springer-Verlag Berlin Heidelberg, 2009, pp. 273–282.

[10] B. Jansen and T. Belpaeme, "A Model for Inferring the Intention in Imitation Tasks," in *The 15th IEEE International Symposium on Robot and Human Interactive Communication, RO-MAN'06*, 2006, pp. 238–243.

[11] M. Pardowitz, S. Knoop, R. Dillmann, and R. D. Zöllner, "Incremental Learning of Tasks From User Demonstrations, Past Experiences, and Vocal Comments ," *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, vol. 37, no. 2, pp. 322–332, April 2007.

[12] V. Krüger, D. L. Herzog, S. Baby, A. Ude, and D. Kragic, "Learning actions from observations," *IEEE Robotics and Automation Magazine*, pp. 30–43, June 2010.

[13] Z. Duric, J. A. Fayman, and E. Rivlin, "Function from Motion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 6, pp. 579–591, June 1996.

[14] R. D. Zöllner and R. Dillmann, "Using multiple probabilistic hypothesis for programming one and two hand manipulation by demonstration," in *IEEE International Conference on Intelligent Robots and Systems*, Las Vegas, Nevada, USA, 2003, pp. 2926–2931.

[15] M. Mitani, M. Takaya, A. Kojima, and K. Fukunaga, "Environment Recognition Based on Analysis of Human Actions for Mobile Robot ," in *The 18th International Conference on Pattern Recognition (IEEE)*, 2006, pp. 782–786.

[16] A. Chella, H. Dindo, and I. Infantino, "Learning high-level tasks through imitation," in *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 3648–3654.

[17] T. Feix, R. Pawlik, H.-B. Schmiedmayer, J. Romero, and D. Kragic, "The generation of a comprehensive grasp taxonomy," in *Robotics, Science and Systems Conference: Workshop on Understanding the Human Hand for Advancing Robotic Manipulation, Poster Presentation*, June 2009.

[18] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[19] M. Kawato, "Trajectory formation in arm movements: Minimization principles and procedures," in *Advances in Motor Learning and Control, ser. Human Kinetics, H. N. Zelaznik, Ed. Human Kinetics Publishers, Chanpaign Illinois*, 1996, pp. 225–259.

[20] J. Hartigan and M. Wong, "A k-means clustering algorithm," *Applied Statistics*, vol. 8, no. 1, pp. 100–108, 1979.

[21] C. Reiley and G. Hager, "Task versus subtask surgical skill evaluation of robotic minimally invasive surgery," in *Medical Image Computing and Computer-Assisted Intervention -MICCAI 2009*, 2009, pp. 435–442.