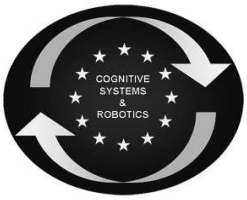
# SAPHARI

SAFE AND AUTONOMOUS PHYSICAL HUMAN-AWARE ROBOT INTERACTION


**Project funded by the European Community's 7th Framework Programme (FP7-ICT-2011-7)**
**Grant Agreement ICT–287513**

## Deliverable D4.3.1

## *Tracking of Human Motions and Object Interaction*

| | |
|---|---|
| **Deliverable due date:** 30 September 2013 | **Actual submission date:** 30 September 2013 |
| **Start date of project:** 1 November 2011 | **Duration:** 48 months |
| **Lead beneficiary:** UNIHB | **Revision:** 1.0 |

| **Nature:** R | **Dissemination level:** PU |
|---|---|
| R = Report<br>P = Prototype<br>D = Demonstrator<br>O = Other | PU = Public<br>PP = Restricted to other programme participants (including the Commission Services)<br>RE = Restricted to a group specified by the consortium (including the Commission Services)<br>CO = Confidential, only for members of the consortium (including the Commission Services) |

# Executive Summary

Many robot co-worker, assistant, and companion applications require the robots to perceive their human partners and to interpret the actions of humans in the respective task and environment context. This deliverable will explain our work on the system under development for task T4.3 of work package 4. Task T4.3 is to design and implement novel and powerful methods for the observation of human actions/movements and object interactions in the context of repetitive joint human-robot manipulation tasks. This deliverable presents the result of integrating several human and object perception algorithms to a perception system called RoboSherlock. The work was mainly performed by researchers from the Institute for Artificial Intelligence at the University of Bremen.

To provide the necessary perceptual capabilities we worked on two categories of results:

- a novel framework tracking human motions and object interaction that implements human activity perception as an unstructured information processing task; and
- special purpose algorithms for the detection of humans and the estimation and tracking of their poses.

SAPHARI's achievements in detecting and tracking of humans and objects are documented by the following four publications one of which published in the proceedings of BMVC (2013) and three publications currently submitted for review.

- Beetz, M., Blodow, N., Balint-Benczedi, F., Marton, Z.-C., Nyga, D., Seidel, F. and Kerl, C. (2013). RoboSherlock: Unstructured Information Processing for Robot Perception. Submitted for review to International Journal of Robotics Research.
- Stommel, M., Edelkamp, S., Wiedemeyer, T., and Beetz, M. (2013). Fractal Approximate Nearest Neighbour Search in Log-Log Time. British Machine Vision Conference (BMVC), Bristol.
- Stommel, M., Beetz, M. and Xu, W. L. (2013). Inpainting of Missing Values in the Kinect Sensor's Depth Maps Based on Background Estimates. Submitted for review to Sensors.
- Stommel, M., Beetz, M. and Xu, W. L. (2013). Model-Free Detection, Encoding, Retrieval and Visualisation of Human Poses from Kinect Data. Submitted for review to IEEE/ASMA Transactions on Mechatronics.

# Table of contents

# 1 Introduction

Many robot co-worker, assistant, and companion applications require the robots to perceive their human partners and to interpret the actions of humans in the respective task and environment context. This deliverable will explain our work on the system under development for task T4.3 of work package 4. Task T4.3 is to design and implement novel and powerful methods for the observation of human actions/movements and object interactions in the context of repetitive joint human-robot manipulation tasks. This deliverable presents the result of integrating several human and object perception algorithms to a perception system called RoboSherlock. The work was mainly performed by researchers from the Institute for Artificial Intelligence at the University of Bremen.

To provide the necessary perceptual capabilities we worked on two categories of results:

- a novel framework tracking human motions and object interaction that implements human activity perception as an unstructured information processing task; and
- special purpose algorithms for the detection of humans and the estimation and tracking of their poses.

SAPHARI's achievements in detecting and tracking of humans and objects are documented by the following four publications one of which published in the proceedings of BMVC (2013) and three publications currently submitted for review.

- Beetz, M., Blodow, N., Balint-Benczedi, F., Marton, Z.-C., Nyga, D., Seidel, F. and Kerl, C. (2013). RoboSherlock: Unstructured Information Processing for Robot Perception. Submitted for review to International Journal of Robotics Research.
- Stommel, M., Edelkamp, S., Wiedemeyer, T., and Beetz, M. (2013). Fractal Approximate Nearest Neighbour Search in Log-Log Time. British Machine Vision Conference (BMVC), Bristol.
- Stommel, M., Beetz, M. and Xu, W. L. (2013). Inpainting of Missing Values in the Kinect Sensor's Depth Maps Based on Background Estimates. Submitted for review to Sensors.
- Stommel, M., Beetz, M. and Xu, W. L. (2013). Model-Free Detection, Encoding, Retrieval and Visualisation of Human Poses from Kinect Data. Submitted for review to IEEE/ASMA Transactions on Mechatronics.

# 2 Tracking of human motions and object interaction as unstructured information processing

The tracking of human motion and object interaction requires the combination of several specialized perception algorithms within one system. Especially in robotics, where vision must provide task relevant information about the environment and the objects in it to various planning related modules.

In order to handle this problem, we introduce RoboSherlock a common framework for cognitive perception, based on the principle of unstructured information management (UIM). For SAPHARI we have worked on a framework for perceiving and interpreting human activities, which is depicted in the figures 1-3.

Unstructured information is a term coined for web-scale learning systems and refers to data where the form of the data does not reflect its semantic meaning. Considering RGBD images, one can say that, in perception systems, unstructured information is processed, since arrays of color information and corresponding depth measurements do not reflect that the image represents scenes, objects, humans and relations between them. The basic idea of unstructured information processing is to detect information pieces that have a deeper semantic structure and to infer this structure. In robot perception these pieces are often objects and object configurations and the structure of these pieces including their texture, color, shape, category, pose, function, parts, possibly bar codes, content, etc.

In RoboSherlock data pieces that are assumed to have more structure are called Subjects of Analysis (SofAs). Typically they are hypotheses of objects and object configurations. SofAs and groups of SofAs are represented in a Common Analysis Structure (CAS), which contain the data pieces and additional annotations for the SofAs. The annotations are computed by annotators, software modules that look at the data pieces and existing annotations and add new annotations or revise existing ones. Annotators are computational experts. For example, if the robot hypothesizes an object, annotators might take the respective point cloud data and try to fit a geometric primitive to it. Another annotator might try to find a bar code and look up the object in a product database. Yet another annotator might take the picture of the SofA send it to Google goggles in order to annotate the CAS with web pages that contain similar images. Other annotators might take the previous annotations and might add hypotheses about the object categories and semantic relations. Again these annotations are hypotheses that are to be tested and ranked by other annotators.

In an UIM system large volumes of unstructured data are analyzed in order to discover knowledge relevant for the end user (e.g the robotic agent). Unstructured information from the robotic process or online sources is processed by Collection Processing Engines (CPEs) described in a CPE Descriptor. A CPE may contain several Analysis Engines (AE), where the flow of the data through different Annotators is defined using Flow Controllers. AEs are either primitive (consisting of a single processing block, expert) or aggregate (multiple primitive engines orchestrated by a Flow Controller).

Figure 1 shows the pipeline in RoboSherlock as used for SAPHARI. The figure includes examplary results for each AE. Sequences of RGBD images are used as input for the pipeline. At the beginning the depth image is preprocessed in order to filter out known structures from the depth measurements in order to reduce the computational effort and at the same time, missing depth measurements are estimated, to gain more accurate measurements. The so updated CAS is used as input for the subsequent AEs. These are used for human and object tracking. The examplary results show perceived bodyparts and objects which are marked by a colored overlay. As a last step the CAS is “consumed”. There are several ways to do so. Informations about objects can be stored in a MongoDB, a Knowledge Base, or returned as a response to a request that triggered the object perception AE.

Figure 2 shows the result for an analysed scene. The rectangles show annotations for two objects.

RoboSherlock discovered that there is a “box” of “Vitalis Crunchy with Honey” on top of the kitchen countertop. The product information were gathered by sending an image of the region of interest to Google Goggles. Actually there is “Vitalis Knusper Schoko” on the countertop, but additionally to the product information Googles provided an OCR result which sais “VITALIS Scholo”. The misdescription is due to the low resolution of the current kinect sensor. Nevertheless one could try to enhance this result with even more annotations. On the right side one can see the annotations for the second object. Correctly detected as being on top of the countertop, having a round shape and white as dominant color.

At last, figure 3 shows the pipeline including all AEs used for SAPHARI. The green arrows define the flow from AE to AE, so that is understandable which annotator depends on the output of which annotator.

In [1], as attached to this deliverable, we introduce several object detection and tracking algorithms integrated in RoboSherlock and discuss the idea of UIM in more detail.

Figure 1: Overview of the pipeline of AEs used within the context of SAPHARI. The collection reader at the very beginning, the pre-processing AE and subsequent the tracking AEs, running in parallel. They forward informations to the CASConsumer which (a) stores informations in a mongo database, (b) in a knowledgebase, or/and (c) returns information as a reponse to a request. Perceived bodyparts and objects are marked by a colored overlay.


Figure 2: An analysed scene. The original image is overlayed with colored regions representing the recognized objects and bodyparts. The rectangles on the left and right exemplary show annotated properties for two of the recognized objects.

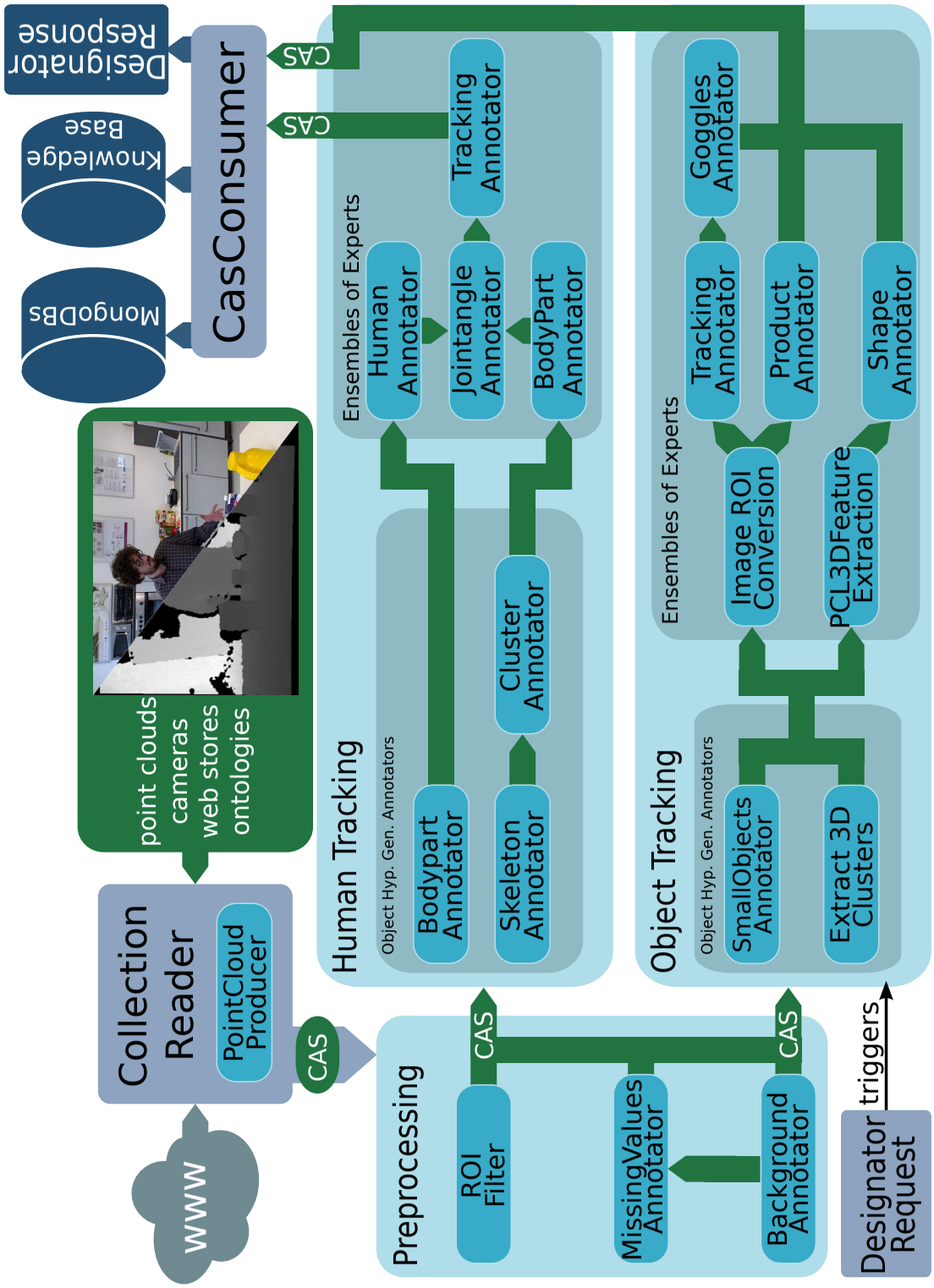



Figure 3: Detailed overview of the annotators used within each AE. Green arrows are marking the flow from AE to AE and from annotator to annotator.

# 3 Special purpose algorithms for perceiving human activity

## 3.1 Inpainting of missing values in depth images

Since we decided to use RGB-D sensors, like the Microsoft Kinect, we have to deal with missing values in the recorded depth data. The Kinect sensor records depth data using structured infrared light. For surfaces that do not reflect infrared, like glass, no depth measurements can be obtained. This can lead to problems in any kind of perception algorithms that heavily relies on high quality measurements. These aspects and a way to estimate those missing values are explained in [3] as attached to this deliverable.

## 3.2 Preprocessing for robust gesture recognition

Another challenging aspect of human and object detection is to segment the image into meaningful regions of interest. Segmenting the entire image, might result into too many objects that have to be analysed. Hence, one can use apriori knowledge in order to remove unecessary image regions. For example if the robot is used within a known environment one can use a model of the environment, to remove known regions from the image to analyse. Same counts for the robot itself. For example if there is the robotarm visible in the image, one can use the robot model to remove the corresponding region from the image.

We therefore preprocess the depth image in order to incorporate as much of the system's knowledge of the environment as possible. Since the tracking RGBD camera is calibrated with respect to the robot system, we can render the robot model in a virtual view from the same vantage point as the depth image (c.f. Figure 4). The measured and virtual depth views correspond pixel-perfect, which allows filtering out all points of the environment and the robot of which we know they are unnecessary for analysing the scene. The performance of this filtering step is critical, since it needs to work even if the robot arm is moving at high speeds, and cannot introduce large latencies for the tracker. This is achieved by loading the measured depth image into a texture on the GPU, which the virtual renderer has access to. A fragment shader is used to compare each depth value of the virtual view with the corresponding measured depth value to perform filtering.

The whole process takes approximately 1ms on a standard GPU, and can filter the known environment and the robot arm even when moving at maximum speed. The filtered depth image is used as input to the object and human tracking components.

## 3.3 Model-Free detection of human poses from rgbd images

For human-robot-interaction it is necessary to be able to perceive both fullbody and upper body human poses. In order to improve the limited scope of many model-based methods, in [4] we propose a spatio-temporal segmentation of keypoints provided by a skeletonisation of the depth contours.

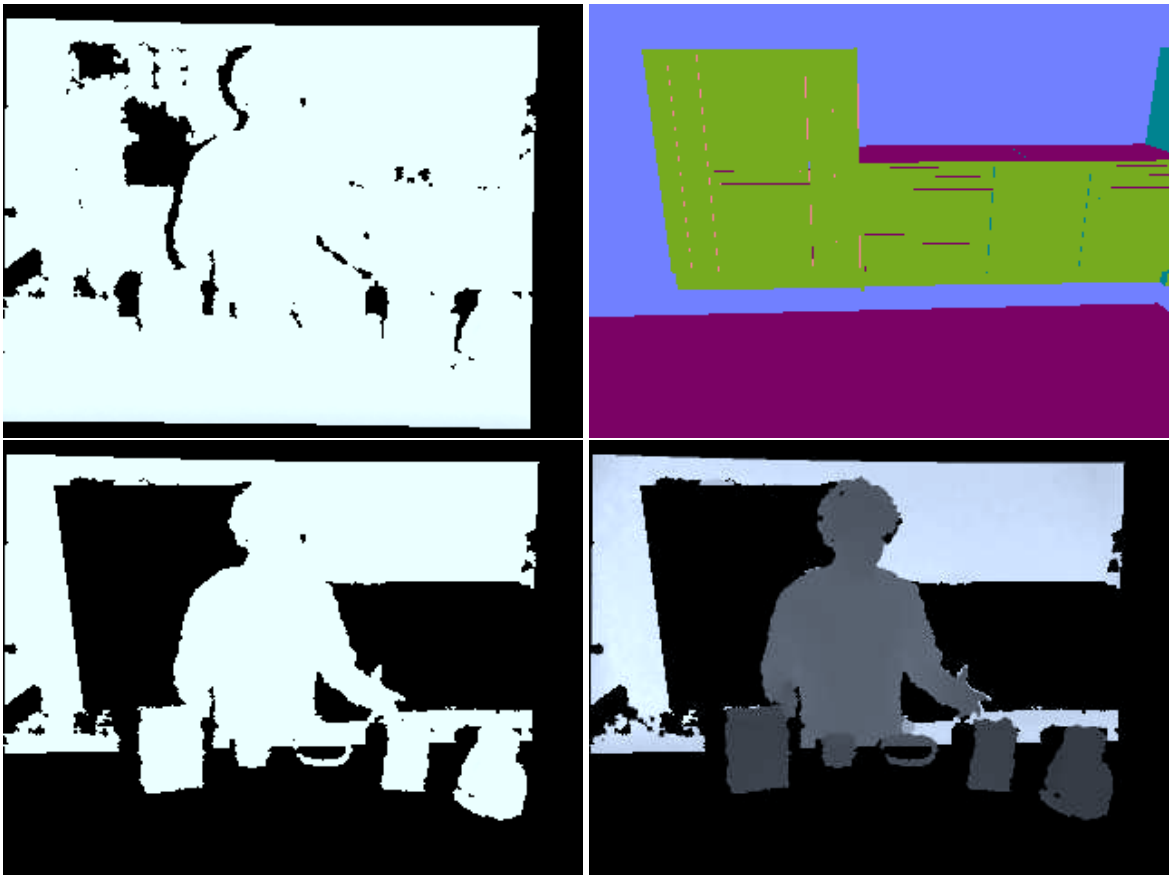


Figure 4: Filtering known structure (e.g. kitchen furniture) from the depth image. The original depth image (top left, binarized for convenience) is compared with virtual depth image (top right) to filter out irrelevant regions (additional black regions, bottom left). The image on the left bottom is used to mask the original depth image, resulting into the image shown on the right bottom

### *3.4 Clustering of human poses*

When trying to analyse human motions, one faces the problem of how to cluster similar poses in huge datasets. An intuitive approach is to use k-nearest-neighbour-search in order to get the k-most similar poses to a query. Unfortunately is a very time consuming task to perform a complete neighbour search. Hence, we present an approach to perform an approximate nearest neighbour search in log-log time in [2], as attached to this deliverable.

# 4 Results

In [1] we investigate the potential of a perception system built on top of the unstructured information management paradigm and propose several perception algorithms to detect and track objects of daily use. Additionally to those algorithms we integrated the model-free human tracking approach of [4] and the approach presented in [5]. This combination of two different “experts” supports the

"ensemble-of-experts" paradigm.

Figure 1 shows a blackbox overview of the pipeline of AEs used for this deliverable. For easier understanding we included a representative image for each AE. There is a collection reader, which currently only reads data from the kinect sensor and an AE that does image preprocessing for the subsequent AEs. The subsequent AEs might run in parallel and perform

- object detection and tracking and
- human detection and tracking.

The input of each run is a RGBD-image. At the very beginning the original depth image (shown on top the preprocessing AE) is preprocessed to fit our needs, which means that we filter out known regions that are not necessary to understand the scene. The result is shown at the bottom of the AE. The preprocessed depth image and the rgb image are used as input for the following AEs. At the end all relevant information about an analysed scene are postprocessed by a CAS consumer. Postprocessing means, (a) storing informations in a mongo database, (b) in a knowledgebase, or/and (c) returning information as a reponse to a request. The latter is done in order to improve the realtime capability of the system, which means, that object detection is only performed after being triggered by a subsystem that needs object informations. Hence, object detection has not to be performed for every single frame.

Figure 2 shows an example of an analysed scene using RoboSherlock. The colored regions represent detected objects and bodyparts and on the left and right we show the annotated properties for two of the recognized objects. For the cereals Google Googles provides very useful informations, like OCR results and product informations. Additionally to that Goggles provides URLs that could also be parsed to gather even more informations. Other annotators give information about the relative position of the object in the environment ("on top of" and "/iai_kitchen/counter_top_island_link"), the primitive shape ("box") and the dominant color ("red").

To achieve realtime for human detection we integrated the human detector presented in [5], which uses a CUDA GPU implementation and runs at approximately 15 fps. In contrast to that, in [4] we present an easier approach based on skeletons and a feature based clustering to detect people. Combining those approaches can improve the accuracy of detecting people. In a next step we are going to apply a particle filter in order to improve the realtime capability, since that could reduce the number of necessary detections of people.

Within SAPHARI, we look at robots operating closely together with humans, and this brings safety considerations also for the perception systems. The perception system showed in this deliverable helps the robot be aware of his environment and to know where a human is located relative to its own position or what the human might do next. This information will allow the high-level control systems to control the robot in a safe way.

# RoboSherlock: Unstructured Information Processing for Robot Perception

Michael Beetz, Nico Blodow, Ferenc Balint-Benczedi,
Zoltan-Csaba Marton, Daniel Nyga, Florian Seidel, Christian Kerl [¶]

September 26, 2013

**Abstract**

A pressing question when designing intelligent autonomous systems is how to integrate the various subsystems concerned with complementary tasks. More specifically, robotic vision must provide task relevant information about the environment and the objects in it to various planning related modules. In most implementations of the traditional Perception-Cognition-Action paradigm these tasks are treated as quasi-independent modules that function as black boxes for each other. Often these subsystems are running in completely different frameworks, with a thin communication interface or middle-ware between them. While each subproblem poses specific requirements that can make fusing them more challenging, it is our view that at least perception can benefit tremendously from a tight collaboration with cognition.

In this article we present a common framework for cognitive perception, based on the principle of unstructured information management (UIM). UIM has proven itself to be a powerful paradigm for scaling intelligent information and question answering systems towards real-world complexity. Complexity in UIM is handled by identifying (or hypothesizing) pieces of structured information in unstructured documents, by applying ensembles of experts for annotating information pieces, and by testing and integrating these isolated annotations into a comprehensive interpretation of the document. This is enabled by a common type structure that defines the semantics behind the annotations and allows for seamless linking with knowledge-bases and intelligently selecting the necessary processing steps based on the observed data.

We will describe RoboSherlock, an open source software framework for unstructured information processing in robot perception and sketch a feasibility study of a perception system built on top of the framework that indicates the potential of the paradigm for real-world scene perception.

[*]M. Beetz, F. Balint-Benczedi are with the Institute for Artificial Intelligence part of TZI, University of Bremen, Germany, {beetz,balintbe}@tzi.de

[†]N. Blodow and D. Nyga are with the Intelligent Autonomous Systems Group, Technische Univerität München, {blodow,nyga}@cs.tum.de

[‡]Z-Cs. Marton is at the Institute of Robotics and Mechatronics German Aerospace Center (DLR), Germany zoltan.marton@dlr.de

[§]F. Seidel is with the Computer Science Department, Technische Univerität München, Germany seidel.florian@googlemail.com

[¶]C. Kerl is with the Computer Vision Group, Technische Univerität München, Germany christian.kerl@in.tum.de

# 1 Introduction

Consider a household robot assistant, a mobile personal robot that is to perform household chores such as setting the table, loading dishwashers, and cleaning up. Such a robot has to find the objects it is to fetch such as my coffee mug or a breakfast bowl. The objects are typically in particular places, in drawers, on tables, or in the refridgerator. The robot also has to look where the objects should be grasped and look for the proper places to put them. In many cases objects are only partially described and the robot needs to perceptually examine them to get additional information needed for manipulating the objects such as their accurate pose or a geometric reconstruction to parameterize the grasp appropriately. *From the perspective of the control system all these perceptual tasks can be phrased as queries that the perception system is to answer and the perception system can be viewed as a question answering system.*

These are some of the simpler queries, but robot perception systems might be required to answer much more complex queries. [1] consider the task of cooperative table setting in which they combine perception and first-order probabilistic reasoning to enable the robots to bring the *missing* items. Or, Tenorth et. al. [2] enables robots to competently handle objects by perceptually decomposing objects in their parts and reasoning about the parts. A number of researchers have equipped robots with means to perceive object affordances as reviewed in [3]. *Answering such queries require competences that go beyond perception alone. They require the perception system to employ knowledge-based reasoning to answer queries competently.* A particular challenge is that the set of queries is open-ended and therefore the robot has to be able to competently answer new queries.

In contrast to these requirements robot perception today is mainly tackled through individual algorithms that solve specific perception tasks under specific conditions. Currently, we have several leading edge methods for detecting and localizing known objects based on previously learned object models in perceived scenes: MOPED [4] can detect and localize textured objects, BLORT [5] can reliably detect cuboid and cylindric objects, and LINEMOD [6], which returns positions but not the poses of the objects it detects. There are many methods for training and learning classification systems for object categorization, e.g. [7, 8]. There are also several model-generating methods, for example for object reconstruction [9]. In addition, there are services such as Google Goggles, which a robot can send images to and receive web pages with similar images from which textual information can be extracted, and so on. Many other special purpose methods such as door handle detectors [10, 11], scene gist classifiers [12], etc. are available. Collections of such algorithms are made available as software libraries such as ECTO[1], PCL [13], OpenCV [14] and the STAIR Vision Library [15].

Clearly, libraries of individual algorithms using common data structures and homogeneous application programmers' interfaces do not suffice for realizing the perceptual capabilities needed by autonomous manipulation robots acting in real-world environments. They need *perception systems* that can synergetically combine available perception methods to meet the information requirements of manipulation robots and scale towards open environ-

[1] `http://plasmodic.github.com/ecto/`

ments. Such systems have to reason about which methods to apply in order to accomplish a given perception task. They have to bridge the gap between the output of algorithms and the information that a robot requests. They also have to combine results of individual algorithms into a coherent and plausible answer for the query. The investigation of perception systems that have these kinds of functionality has received surprisingly little attention so far.

Another aspect of robot perception that deserves to be highlighted is the mismatch between the structure of data (images as arrays of pixels) and the hierarchical, abstract, and relational information that is depicted by the images. Extracting relational information from unstructured data is a research challenge on its own. *unstructured information management* [16] deals with these issues in the context of web-scale information systems.

Our conclusion from our previous considerations is that we propose that autonomous manipulation robots should be equipped with *perception systems that can answer open sets of queries of the robot control systems using ensembles of experts that encapsulate leading-edge perception algorithms that solve specific perception problems.* We propose that the system should start with hypothesizing substructures in the low-level data that are meaningful, that is the ones that reflect depicted objects and relational structures. The system then runs the experts methods on the substructures they are competent for and the expert annotate the respective structures with the information they can compute. The individual annotations are considered as hypotheses and are tested, combined, and ranked by other experts to produce the most appropriate answer to the original query.

Query answering systems that operate this way already exist and have had impressive success in scaling towards open real-world problems using noisy, incomplete, ambiguous, and even inconsistent real-world information. Perhaps, the most promising example is Watson [17]. Watson is a question answering system that has won the US quiz show *jeopardy!* against the champions of the show demonstrating an unprecedented breadth of knowledge and the crucial ability to correctly judge it's own competence in answering a particular questions. Key success factors in Watson were unstructured information processing, hypothesis generation, testing, and ranking, and the use of annotation and ensembles of experts. We believe that the same concepts is capable of helping to scale robot perception systems towards real-world tasks.

In this article we transfer the ideas of *unstructured information management* to the problem domain of perception for autonomous manipulation robots. To this end, we take the UIMA, an opensource middleware software framework and extend the framework to be applicable to robot perception and perception-guided manipulation. The framework provides data structures and representations for semantic unit hypotheses and annotations for hypotheses, mechanisms for the context-directed application of expert methods, mechanisms for testing, filtering, evaluating, and ranking answers and combining information from different sources. Extensions needed for robot perception include mechanisms for incorporating unstructured information processing into perception-guided control and a datatype system for robot perception. Using the framework we can build perception systems that can extract more comprehensive information from perceived scenes and can combine information from different methods more coherently.

The main contributions of this article are the following ones:

- The design and implementation of a framework based on the premise of treating perception as an Unstructured Information Management task. This includes ensemble-of-experts methods, infrastructure for the orchestration and execution of these methods, and well-defined, open and extensible object descriptions.
- We combine knowledge processing and perception in an organized, flexible, reusable and extensible manner. In particular, we propose two key embodied symbolic reasoning mechanisms:
  - *object identity resolution*, which is a first-order probabilistic reasoning technique that computes the probabilty that two descriptions of perceived objects refer to the same object. The underlying reasoning model takes probabilities of moving objects and that objects might look different under different conditions into account.
  - *learning and reasoning about first-order probabilistic models of objects, their perceptual features, and their co-occurrence in scenes.* This reasoning techniques boosts perception results by exploiting background knowledge such as very few objects with a logo of a particular brand are round, milk tetrapaks usually occur in scenes taken from the refridgerator, etc.

The remainder of the article is organized as follows. In Section 2 we present the state of the art and relevant related work, followed by an overview of our approach in Section 3. Details about implementation and key parts of the system are presented in Sections 4 to 7. Finally in Section 8 we present the experiments conducted and the obtained results.

# 2 Related Work

Existing perception systems usually consider the case where a database of trained object is used to match it with sensor data. Even more, many systems focus on individual algorithms that only work on objects with specific characteristics, e.g. point features for 3D opaque objects [18], visual keypoint descriptor based systems like MOPED [4] for textured or [19] for translucent objects. There are as of now no methods to perceive "everything". While the detection and accuracy of individual algorithms are continually improved it will still be very hard – if possible at all – to achieve the recall and accuracy levels needed in manipulation tasks.

However, a large number of methods exist that can handle some of the subproblems of perception reasonably well. Many of these methods are complementary and could be combined to boost performance. In the area of visual perception for cognitive systems the *ensemble of experts* approach has received surprisingly little attention, though.

Jain *et al.* [20] provide a taxonomy for ensemble methods, categorizing them based on:

- the architecture type (which can be parallel, cascading, gated, hierarchical or a combination of these);
- whether or not the ensemble is trainable;

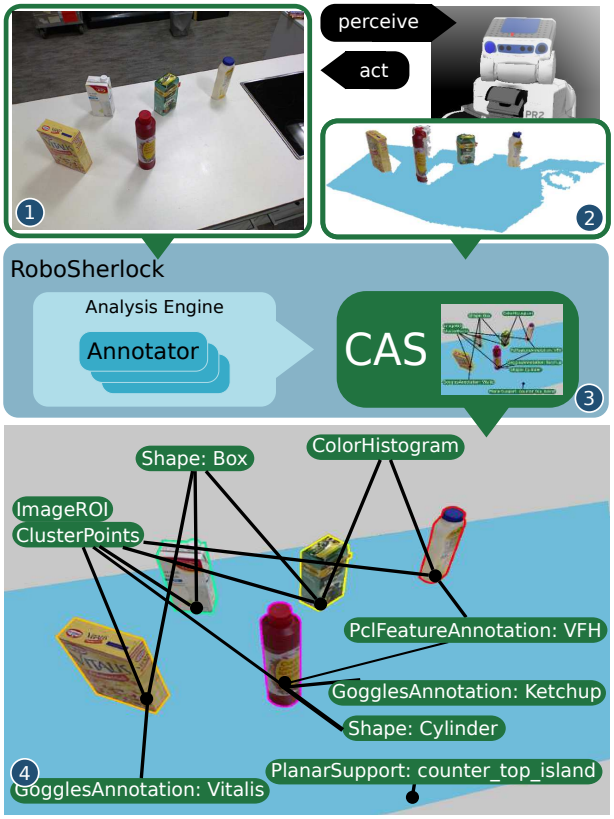



Figure 1: Perception as Unstructured Information Management: Common Analysis Structure (CAS) holds the original high-res camera (1) and depth (2) images, which act as input documents for our RoboSherlock system (3). The CAS (4) also contains information concerning known structure (such as the 3D semantic map, light blue), and filtered regions (gray), as well as Annotations referencing various parts of the document (green labels).

- according to the level of information the members of the ensemble produce about their classification decisions;
- whether it is adaptive, which means that the ensemble weights the contribution of a member to the decision based on the input pattern.

In another machine learning application domain, the NetFlix Prize, as individual teams started to join efforts, ensemble learning became a popular, and very successful approach [21]. We expect a similar development in the perception field as well, as combining various approaches with complementary strengths can improve over the individual performance of the methods [22, 23] and make it generalize better [24].

There are a lot of works oriented at creating perception libraries which are a collection of task specific algorithms, e.g. ECTO[2], PCL [13], OpenCV [14] and the STAIR Vision

[2] `http://plasmodic.github.com/ecto/`

Library [15]. The perception tasks that such a robot has to accomplish go substantially beyond what is supported by current perception libraries and frameworks. Frameworks, mostly based on middle-ware like ROS[3], such as SMATCH [25]) or REIN [26] have targeted the ease of program development but the problems of boosting perception performance through more powerful method combination has received surprisingly little attention.

An early example of a robotic perception system was described by Okada *et al.* [27], where a particle filter based integration of multiple detectors and views was achieved. The probabilistic fusion of different results corresponds to a simple rule ensemble, i.e. one that is not trainable. Similar methods have been employed for semantic mapping approaches [28, 29], but in this work we propose to use a more flexible strategy. The UIM architecture allows both for the incorporation of the ensemble learning framework from [8] and allows for steering the processing workflow in order to support active classification [30].

RoboSherlock was created by extending the pluggable Apache Unstructured Information Management Architecture open source project [4], that supports development in Java and C++ (Perl, Python and TCL via SWIG[5]). The UIM Architecture (UIMA) incorporates the work of the Open Advancement of Question Answering Initiative [31], and it is designed with a distributed architecture, multiple processing modules working together, and self-adapting strategies for data interpretation in mind. By providing such functionalities for a perception system, we can facilitate more advanced robotic capabilities [32].

# 3 Overview

In this paper we propose RoboSherlock an architecture to build scalable robot perception systems.

RoboSherlock considers point clouds and camera images as being unstructured information. Unstructured information is a term coined for web-scale learning systems and refers to data where the form of the data does not reflect its semantic meaning. For example, point clouds are arrays of depth measurements together with color information. The data does not reflect that the point clouds represent scenes, objects, object parts, and relations between them. The basic idea of unstructured information processing is to detect information pieces that have a deeper semantic structure and to infer this structure. In robot perception these pieces are often objects and object configurations and the structure of these pieces include their texture, color, shape, category, pose, function, parts, possibly bar codes, content, etc.

In RoboSherlock data pieces that are assumed to have more structure are called Subjects of Analysis (SofAs). Typically they are hypotheses of objects and object configurations. SofAs and groups of SofAs are represented in a Common Analysis Structure (CAS), which contain the data pieces and additional annotation for the SofAs. The annotations are computed by *annotators*, software modules that look at the data pieces and existing annotations and add new annotations or revise existing ones. Annotators are computational experts. For example, if the robot hypothesizes an object, annotators might take the respective point

[3] `http://ros.org`

[4] `http://uima.apache.org/`

[5] `http://www.swig.org/`

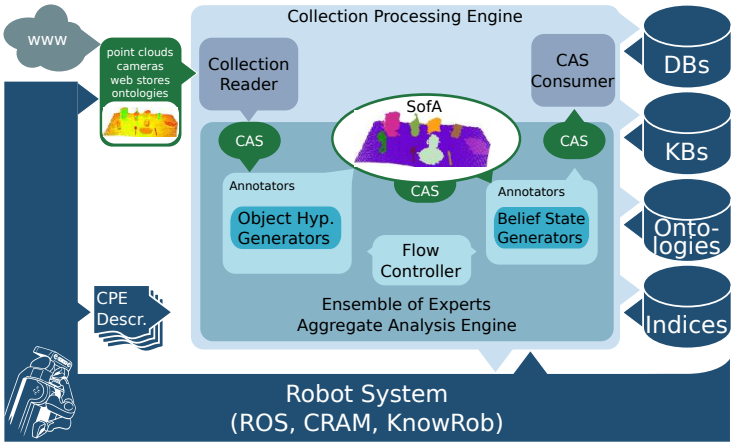



Figure 2: Schematic overview of our UIMA-based object processing framework.

cloud data and try to fit a geometric primitive to it. Another annotator might try to find a bar code and look up the object in a product database. Yet another annotator might take the picture of the SofA send it to Google goggles in order to annotate the CAS with web pages that contain similar images.

Other annotators might take the previous annotations and might add hypotheses about the object categories and semantic relations. Again these annotations are hypotheses that are to be tested and ranked by other annotators.

The design of the RoboSherlock toolbox facilitates and supports unstructured information processing, hypothesis generation, testing, and ranking, and the use of annotation and ensembles of experts.

For a demonstration let us consider the example scenario from Fig. 1. At an initial phase the CAS only contains the scene, this being our SofA (bottom row of Fig. 3). While being processed (Fig. 3), the CAS gets filled with hypotheses and beliefs about the scene through annotations. In the considered scene, first a hypotheses is generated about the region of interest. Afterwards this gets refined by creating object hypotheses. and even more information is gathered about these objects, in the end generating a semantically rich collection of annotations.

## 3.1 Analysis Engines in RoboSherlock

In accordance to Watson, sense making from raw perception data is organized in programmer-specified analysis engines. For specific query types analysis engines specify the expert methods that are to be employed, the order in which they are activated and the data flows during the analysis process.

In an UIM system large volumes of unstructured data are analyzed in order to discover knowledge relevant for the end user (e.g the robotic agent). Unstructured information from the robotic process or online sources is processed by Collection Processing Engines (CPEs) described in a CPE Descriptor. A CPE may contain several Analysis Engines (AE), where

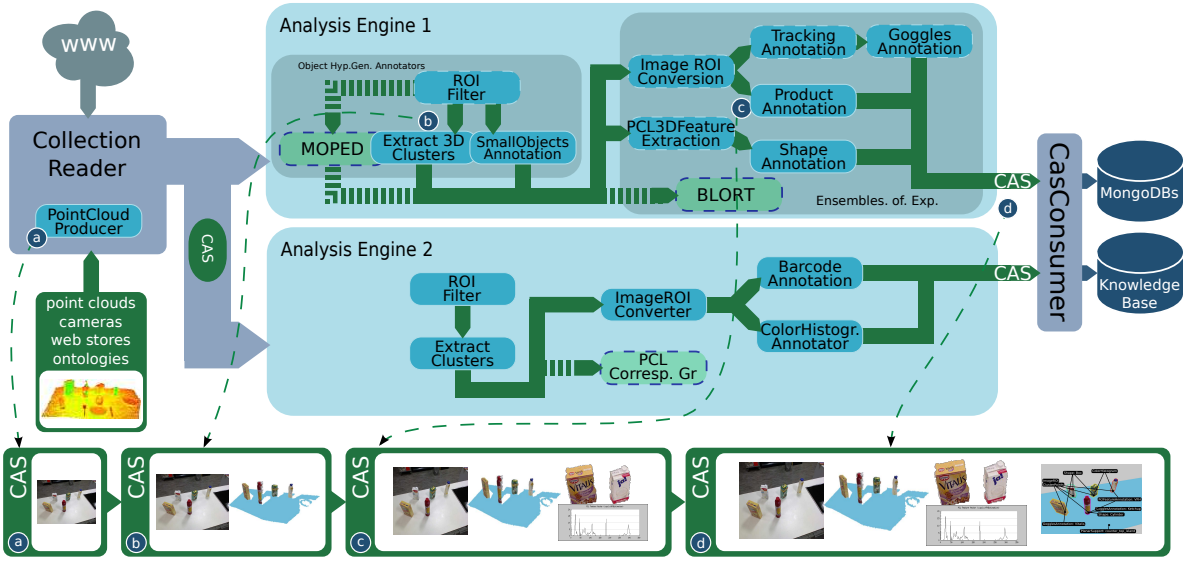



Figure 3: Example of a pipeline execution in RoboSherlock showing how the content of the CAS changes over time and the main parts of the system: CAS producer, Analysis Engines, Cas Consumers and Annotators. Note: the locations of MOPED [4] and Blort [5] are marked with different color because at the moment they represent a possible integration

the flow of the data through different *Annotators* is defined using Flow Controllers. AEs are either primitive (consisting of a single processing block, expert) or aggregate (multiple primitive engines orchestrated by a Flow Controller). An example of how the execution of an AE can look like is shown in Fig. 3. Here the CPE contains two AEs, inside which several run sequentially or in parallel in order to annotate the data, at the end of execution data from the CAS being stored in a data or knowledge base. The annotators connected with dashed lines represent potential integration points for other perception *experts* that we could make use of: e.g. MOPED [4] could be very useful for generating object hypotheses, while Blort [5] and correspondence grouping from PCL are good examples of what an expert that refines the hypotheses should be.

## 3.2 Integrating Perception Capabilities into RoboSherlock

Integrating a legacy perception component into RoboSherlock requires three steps: First, declaring the data structures of the perception system in the RoboSherlock type hierarchy. Second, wrapping perception routines as RoboSherlock hypothesis generators and annotators. Third, specifying the control flow for complex perception tasks in terms of RoboSherlock pipelines. RoboSherlock already provides type hierarchies and wrappers for PCL and ROS data structures and modules. The control flow specified in RoboSherlock pipelines are then applications on top that can offer very flexible and task adaptive processing for semantically rich problem formulations.

In our scenario (like the one in Fig. 3), a typical Collection Processing Engine can be sketched as follows: A CAS is created that holds the Kinect sensor data and interesting information about the robot (e.g. localization data). The CAS holds one or more Subjects of Analysis (SofA), which can be annotated in Index Repositories for convenient search, retrieval and structuring. It can be thought of as a mixture of message passing and black board systems. These annotations usually reference a subset of the SofA, such as a region in text or a mask in an image. [6]Annotators – experts – can filter the contents to obtain the data relevant for their tasks, create new SofAs, annotations and index repositories. Annotators create hypotheses about objects (cf. Sec. 4.1). Potential objects are then annotated, creating rich feature representations (cf. Sec. 4.2) that get analyzed in total by entity resolution modules (cf. Sec. 5)

In the end, the resulting belief states are collected in a data base for convenient visualization, reuse in later processing iterations, and to serve as data collection for different research scenarios (cf. Sec. 6).

This hierarchy of execution modules enables easy combination of results from different experts and seamless integration with knowledge bases, enabling the perception system to fulfill the functionalities mentioned in Section 1. Another important aspect is that storing the results of experts as annotations in the CAS during execution we have easy access to all information resulting from the different perception algorithms.

## 3.3 Principles of RoboSherlock

**Principle 1:** Consider **perception as a problem of making sense of sensor data**. Unstructured information is defined as information without an underlying a-priori data model or which cannot be easily expressed in a relational table. While it has usually been associated with textual data we apply it to sensor data. An RGB-D scan can be regarded as a large set of raw data points and low-level features which are a function of latent underlying structure in the real world, such as furniture or other objects.

**Principle 2: Annotation of object hypotheses through ensembles of experts.** Annotations are a key part of the overall system. The idea behind having annotations is that the results of every expert that is run on a certain scene are stored with respect to the original scene, allowing this way the cross-examination of the results from different experts.

**Principle 3: Information fusion and hypothesis selection** Results obtained from complementary or competing methods need to be merged and filtered in order to arrive at a final decision. This is essential for combining strengths and mitigating weaknesses, thus improving performance over the individual methods.

The mechanism enabling this is self-analysis of the system's competences and abilities. This is made possible by attaching confidences and other quality measures to both processing modules (Annotators) and their statements (Annotations) by collecting respective statistics. These can also be used in algorithm selection by reasoning over available computing resources and expected returns [30]. Examples include the synergistic effects of drawing on multiple

[6]In contrast, data flow in e.g. ROS is very rigid, and message channel ("topic") connectivity encodes semantic meaning of data. Addition or modification of information to messages can be cumbersome.

heterogeneous object descriptions, but also the fusion of different segmentations.

# 4 Object Hypothesis Generation and Annotation

The main perceptual task that we are using RoboSherlock for so far is the detection of objects and examining properties of objects in order to answer queries about the objects and the scenes they are part of. In this section we will describe analysis engines and their components that we have designed for these perceptual tasks.

When dealing with object recognition, reconstruction, classification or similar object-centric tasks, segmentation is an integral part, both in importance and in representation. Pixel-level post-processing usually requires image masks, 3D point cloud based feature descriptors require 3D regions or point cloud subsets (index vectors) to operate on, whereas higher-level knowledge reasoning on objects is mostly concerned on where something is in terms of some coordinate system.

## 4.1 Hypothesis Generation

We propose to treat segmentation slightly more abstractly by dealing with *object hypotheses* on a higher level, which are subclassed for the different instantiations of segmentation methods. This way, we combine different algorithms in a consistent manner, e.g.:

- an attention mechanism that detects points of interest in pixel coordinates creates a region of interest (point and extents) in the camera's *tf* frame.
- image segmentation algorithms (e.g. color-based) can generate masks or region maps, referencing the respective image.
- point cloud segmentation relying on supporting planar structures can generate index vectors.
- even uncommon object hypothesis generators such as a skeleton tracker which observes a "put down" event could just generate a point in 3D space to be analyzed as a potential region of interest.

Note that most of these types can be converted between each other relatively trivially, e.g. projecting a point cluster from a point cloud into a camera image, or transforming an image region to a grasping pose in robot-local coordinates. In a well-integrated platform such as the PR2, this can even be done at a later point in time from another vantage point to some extent (e.g. through the *tf* library). This allows us to retrieve the camera image region of interest corresponding to a 3D point cluster, enabling the combination of image analysis annotations and point cloud processing.

We use a 3D semantic map of our environment [33] which we render off screen from the sensor point of view. The resulting virtual depth image is compared with the real one, and the known "background" segmented on a per-pixel level. This also allows us to augment the

description about an object cluster with a semantically meaningful object location (e.g. "on top of **sink_counter_link**").

This way, it becomes possible for the designer of the component engine descriptors to formulate regions of interest in the environment to be connected to different processing pipelines: e.g. "object detection should be performed on all tables, counter tops and shelves" or "ensure empty space in front of the dishwasher" before opening it. This is done analogously to previously published methods such as table top object segmentation, with an additional prior on the locations of supporting structures. The result of our segmentation engine is a timestamped **Scene** object, as described above. As this results in point cloud clusters, we augment each cluster with the corresponding image regions from the robot's cameras, thus obtaining a higher-resolution image of the object that is suitable for image-based cluster annotation.

## 4.2 Object Annotation

One can think of these multivariate descriptions of clusters as more or less independent object hypotheses which can be annotated with various feature descriptions in parallel. In our case, we have explored and implemented the following annotations and algorithms.

**GenericFeatureFromNormalsEstimator** is an annotator which can process any point cluster (with estimated normals) and compute any subtype of `pcl::Feature`, depending on parametrization. This utilizes the largest collection of open source 3D feature computation methods, including VFH, CVFH, FPFH, Spin images, RIFT, SHOT etc.

**SACModelAnnotator** is also a wrapper around PCLs' RANSAC based model fitting. It can fit the models defined in PCL (planes, lines, circles etc.) to a given point cluster.

Another annotator is based on Google Goggles[7], which is a web service and smart phone app allowing the analysis of an image, which generates a highly structured list of matches including product descriptions, bar codes, logo/brand recognition, OCR text recognition or a list of similar images. Note that for each Goggles reply, there is an abundance of additional information, such as URLs to web stores, price ranges, translations etc.

To compensate the latency of online Goggles queries, we continue processing the cluster while the query is issued asynchronously and make use of tracking capabilities (cf. Sec. 5) to attach the result to the corresponding object *track* when it is done. This is usually not much of an issue, since during the round-trip time of around 2 seconds, most scenes remain mostly unchanged.

In order to reduce network bandwidth, the Goggles annotator contains a cache that uses perceptual hashing on the cluster image to prevent continuously resending very similar image regions. This principle of *composition* of change detection with any algorithm could be applied to a large number of annotators and should in future work be factored out as a general capability.

An important annotator for the entity resolution is the **SimilarityAnnotator**. This module, when used, defines a similarity measure between the object hypotheses.

[7] `http://www.google.com/mobile/goggles/#label`

In general it is not sufficient to rely on just one feature, because e.g. a purely geometry-based object description like a shape classification fails to incorporate color or texture properties and might therefore misjudge the actual similarity. We incorporated a relatively elementary approach based on combining multiple feature dimensions using the geometric mean. This problem,not being trivial deserves a more thorough investigation in the future.

Let a Scene $\mathcal{S}$ be a set of object observations: $\mathcal{S} = \{o_i | i \in [0 \dots n]\}$. Each object observation $o_i$ in turn is a tuple of a set of object hypothesis descriptions (e.g. image regions, point cloud clusters) and a set of Annotations: $o_i = \langle \mathcal{H}, \mathcal{A} \rangle$, where $\mathcal{H} = \{h_i\}$ and $\mathcal{A} = \{a_i\}$.

Every $a_i \in \mathcal{A}$ has a certain data type $t(A)$. For Annotations $a_j, a_k$ of same type $T = T(a_j) = T(a_k)$, we define a similarity function $sim_T : T \times T \rightarrow [0..1]$. The result of such a function *sim* represents the degree to which the two given clusters appear similar, given the respective data representation.

Since different Annotation types encode arbitrary information about the cluster, the similarity function needs to act as a bridge between raw or processed sensor data, which usually contain geometrically, visually or semantically meaningful values, and the probabilistic degree of belief to which we consider two clusters to appear similar. For many Annotations, one can define an actual distance metric, e.g. color hue distance, cylinder length difference, etc. For all types $T$ where such a distance metric $d_T$ is possible, we offer two convenience functions $g$ and $l$ that are designed to transform the co-domain of $d_T$ to be in the range [0..1]:

$$g(x) = e^{-\left(\frac{x}{2\sigma}\right)^2}, \tag{1}$$

$$l(x) = 2/(2 - (1 + \exp(-x))), \tag{2}$$

where $g$ is a zero-mean Gaussian function normalized to have a maximum of 1 at $x = \mu = 0$, and $l$ is a function that decays approximately linearly at first and approaches 0 asymptotically. Annotation similarity functions have access to these functions in their implementation.

We understand $g(d_T(a_j, a_k))$ and $l(d_T(a_j, a_k))$ to be a "percentage" of how close the feature distance is to the optimum at $d_T(a_j, a_k) = 0$. The variance $\sigma^2$ needs to be estimated for each Annotation type.

For some, this variance can be easily justified, e.g. the variance of the differences in position of two cluster observations taken from the same objects, but from different points of view, is expected to be in the same order of magnitude as the localization covariance of the robot itself and the sensor noise. This is to say that any error in localization directly affects a cluster position. However, for other feature spaces, e.g. color hue, this variance is not strictly meaningful, and must be adjusted as a user parameter or learned in order to account for the special environment properties, such as lighting conditions, or desired color sensitivity.

There are also annotators, which wrap around existing perception frameworks, namely for BLORT, Moped and Linemod.

### 4.3 Classification Framework

Another open source component we integrated is the classification framework described in [8], in the form of the **FeatureClassificationAnnotator** capable of examining all object hypotheses that have a certain **PclFeatureAnnotation**. It can be parametrized (e.g. to process all clusters with VFH features) and creates a **ClassificationAnnotation**. As of now, this annotator provides object-level classification using global point cloud features, but the framework supports any type of feature (and bag-of-features), so we will use it for *local* feature classification as well.

The **ClassificationAnnotation** type specifies the employed feature and classifier and the resulting label, in addition to a list of class confidences and/or class accuracies ($\langle label, probability \rangle$ pairs) that classifiers can report, and are useful for ensemble learning [8, 23].

Note that this annotator can be used to process past data, which is interesting for time consuming operations like post-processing large amounts of data collected over extended periods to complete models for the universe of objects present, or to create and test classifiers and ensemble of experts methods.

### 4.4 WWW Knowledge Sources

To obtain the necessary "common sense" knowledge needed for performing complex tasks, using rich information from online sources is a promising approach [34]. There are already huge databases online that hold robotic perception relevant information, like the Trimble (Google) 3D Warehouse, that can be used as training data for detecting similar objects in real sensor data acquired by the robot [29], and even enhanced by real, environment specific data using *domain adaptation* [35]. Such links to online structured object descriptions, knowledge bases and web stores allows the creation of more complete object descriptions and to reason on more than what is visible (backsides, storage temperature, weight, etc.) [36]. By combining perception and knowledge reasoning to improve knowledge gathered by the robotic agent, it is possible to treat perception as an integrated capability of the autonomous agent, capable of answering complex queries about its perceived environment [1].

In [36], we report on the acquisition of highly structured product data bases including images, textual descriptions and information pertaining to e.g. perishability and ingredients by preprocessing web sites of online retailers such as e.g. GermanDeli.com. This information is parsed and stored offline in a database for ready access by any annotator. We train appearance (SIFT) features from the product images for image based recognition of objects, and can perform textual searches for product descriptions mentioning e.g. "Nestlé", "milk" or "Kellog's", after corresponding logo/brand results from the Goggles annotator.

## 5 Tracking/Entity Resolution

We want to be able to know where things are. And where they were. And on top of that, we also want a reconstruction and object description and like to know what data we have

collected about them in the past. In order to achieve this we make use of tracking algorithms and a for more challenging situations a method for entity resolution, as it will be detailed in the .

Currently, the following, independent tracking systems are in place: In its simplest form, a *position based tracker* attempts to relate current observations of objects to past observations, ignoring appearance. 3D appearance is used in a *shape based tracker* in the form of VFH features, and image appearance in a tracker based on color histograms. These components assign tracker-local IDs to clusters and share a common Annotation type, **TrackingAnnotation**, which expresses the tracker's belief about the cluster trajectory and identity. A separate *tracking arbiter* uses these annotations in a voting scheme to attempt to find an explanation consistent with all the evidence.

These can deal with most frame-to-frame changes of static and moderately dynamic scenes. However, there are situations in which these tracking methods are too simple in nature and thus contradict each other. In these cases, we employ a more time-consuming inference step similar to [37] based on Markov Logic Networks (MLNs). This method can make use of arbitrary similarity metrics computed between observations to compute a probability distribution of which current clusters are the *same* as which previous clusters, while being able to deal with hard cases such as indiscernible objects (e.g. 12 equal cups) or partial visibility of the scene. It also maintains a belief of which previous observations of objects could still be valid, even if they are currently out of sight. Using statistical relational models, the system can consider all the observations and annotations simultaneously, obtaining a globally consistent posterior distribution over the associations.

In [37], we presented an MLN model we used to solve the problem of object identity resolution. The model was based on a notion of *objects* and *clusters*, and modeled most predicates directly or indirectly involving a predicate *is*. This predicate encodes the assignments of clusters to objects, and requires a database of objects, or put differently, an a priori known universe of objects.

For the scenario presented in this work, we remodeled the MLN in order to eliminate this limitation and be able to tackle the problem of object identity resolution even without a known universe of objects. This means that predicates that were modeled around *is* needed to be changed to rely on *similarity measures* only in order to estimate a probability of two clusters being the same object. In our framework these similarity measures are obtained using the similarity annotator described in Section 4.2.

Table 1 shows the concrete MLN model we use to solve the problem of object identity resolution. As entity types, the model considers object hypotheses pertaining to objects placed in a scene, as well as abstract entities representing *explanations* for pairs of object observations taken at different points in time. As such, we implicitly mode the time domain in a heavily discretized way, as we differentiate only between the most recent point in time (at which all entities $n \in newCluster$ were observed) and points in time that preceded it (these past observations are all entities $o \in oldCluster$).

As a scene containing new object observations arrives in our Annotator, we interpret the scene to populate the domain *newCluster* and create the evidence predicates (marked "e" in Table 1): *similar*, *proximity* and *outOfView*. We then apply the model to update our beliefs which we extract by analyzing the query predicates (marked "q"): *is*, *appear*, *disappear* and

| e/q | *predicate declarations* | *domain declarations* |
|---|---|---|
| e | similar(oldCluster, newCluster) | oldCluster = {O1, O2,...} |
| e | proximity(oldCluster, newCluster) | newCluster = {N1, N2,...} |
| e | outOfView(oldCluster) | |
| q | is(oldCluster, newCluster, explanations!) | explanations = {*move*, *stay*, *newview*, *not* } |
| q | disappear(oldCluster) | |
| q | appear(newCluster) | |
| q | hidden(oldCluster) | |

| $i$ | $F_i$ |
|---|---|
| 1 | (is(a, c, e1) $\wedge$ is(b, c, e2) $\wedge \neg$ (e1=*not*) $\wedge \neg$ (e2=*not*)) $\rightarrow$ a=b. |
| 2 | (is(c, a, e1) $\wedge$ is(c, b, e2) $\wedge \neg$ (e1=*not*) $\wedge \neg$ (e2=*not*)) $\rightarrow$ a=b. |
| 3 | is(o, n, *stay*) $\leftrightarrow$ (similar(o, n) $\wedge$ proximity(o, n)). |
| 4 | is(o, n, *move*) $\leftrightarrow$ (similar(o, n) $\wedge \neg$ proximity(o, n)). |
| 5 | is(o, n, *newview*) $\leftrightarrow$ ($\neg$ similar(o, n) $\wedge$ proximity(o, n)). |
| 6 | outOfView(o) $\rightarrow \neg$ ($\exists$ n (is(o, n, *stay*) $\vee$ is(o, n, *newview*))) $\wedge$ (is(o, n, *not*) $\vee$ is(o, n, *move*)). |
| 7 | appear(n) $\leftrightarrow \neg$ ($\exists$ o (is(o, n, *stay*) $\vee$ is(o, n, *move*) $\vee$ is(o, n, *newview*))). |
| 8 | disappear(o) $\leftrightarrow \neg$ ($\exists$ n (is(o, n, *stay*) $\vee$ is(o, n, *move*) $\vee$ is(o, n, *newview*))) $\wedge \neg$ outOfView(o). |
| 9 | hidden(o) $\leftrightarrow \neg$ ($\exists$ n (is(o, n, *stay*) $\vee$ is(o, n, *move*) $\vee$ is(o, n, *newview*))) $\wedge$ outOfView(o). |

Table 1: Markov logic network for object identity resolution. Free variables in formulas are implicitly assumed to be universally quantified. Formulas 1 and 2 are hard formulas and essentially have an infinitely large weight, which, in practice, is substituted by a sufficiently large real number. The domain *explanations* is fixed across all instantiations of the model, which is why we declare it explicitly. The domains *oldCluster* and *newCluster* change for every instantiation, depending on the number of objects observed in a scene. The predicate declarations indicate the domains to which the predicates are applicable. The predicate declaration for *is(oldCluster,newCluster,explanations!)* contains an argument suffixed by an exclamation mark, which means this predicate is declared as functional, i.e. for each pair of old and new clusters, there must be exactly one explanation for which the predicate is to hold in any possible world.

*hidden*.

Markov logic networks are particularly well-suited for the representation of such a model, as they allow us to specify the various hard constraints that any valid entity-observation association must satisfy in first-order logic. At the same time, probabilistic rules (which either increase or decrease the likelihood of associations) can be expressed as soft constraints. Any beliefs computed by the model are guaranteed to be probabilistically sound and globally consistent with respect to the constraints that were specified.

# 6 Information Fusion, Storage and Reuse

In the previous sections, we described how the raw sensory input data is being transformed into semantically more meaningful information by application of annotators. These annotators can be thought of feature generators, transforming low-level observations into more abstract, symbolic representations. As an example, consider the *PclFeatureAnnotator*, transforming a 3D point cloud into a symbolic descriptor such as *Box* or *Round*, or a *ColorAnnotator* yielding a symbolic description of what colors can be found in the object under consideration, such as *Red* and *Blue*. However, since most of the annotators producing those object hypotheses are applied independently of each other, their outputs are not guaranteed to be globaly consistent and they typically do not take into account object interactions in the current scene. Thus, for a perception system like RoboSherlock, the ultimate goal is to come up with a final ensemble decision on object identities/hypotheses.

To this end, we apply state-of-the-art methods from the field of statistical relational learning (SRL), a subfield of the machine learning discipline that has emerged and gained a lot of attraction in the recent couple of years. In those SRL models, we can capture complex object interactions, represent and reason about object properties, their attributes and the relations that hold between them. Most notably, the ultimate strength of SRL models is their capability of allowing for reasoning about *all* observations simultaneously, taking into account interactions between objects and thus achieving a posterior belief that is guaranteed to be globally consistent. Maintaining a joint probability distribution over observations, their class labels and the robot's current context or belief state has several advantages over a discriminative model that seeks to just discern a set of objects given the observations and gives raise to numerous possibilities of how information from different sources can be combined in order to improve the performance of the overall system. From a probabilistic point of view, inference in the former can formulated as solving for

$$\arg\max_{l} P\left(object(o, l) \mid shape(o, Box), \ldots\right) \tag{3}$$

where *object* is the predicate assigning a class label to some observation $o$ and *shape* etc. is the set of observations we obtained from the sensory modules. Inference in a joint probability distribution, however, can not only be employed to reason about what kinds of objects are present in the current scene, but also, e.g. to compute what observations can be *expected* in the current or even in variations of the current scene:

$$P\left(object(o, l), shape(o, Box), \ldots\right), \tag{4}$$

which can be used to formulate arbitrary queries. As an example, consider a breakfast scene where a box of cereals is located on the table, but the robots current view angle does not allow a texture annotator to extract informative features. A distribution of the form of (4) can be used in order obtain a belief about what additional features from e.g. Google goggles can be expected if the object under consideration was actually a box of cereals:

$$\arg\max_l P\left(\mathit{goggles\_logo}(o, l) \mid \mathit{shape}(o, \mathit{Box}), \mathit{object}(o, \mathit{Cereals}, \mathit{scene}(\mathit{Breakfast}))\right) = \text{“Dr Oetker”}$$

In a next step, the goggles annotator can be specifically applied to the scene in order to check if the “Dr Oetker” brand logo can actually be recognized on the box. We argue that the opportunities offered by a joint distribution over the system's belief state go far beyond what can be done with traditional discriminative approaches for object identification.

In particular, we employ Markov Logic Networks (MLN) [38] a powerful knowledge representation formalism combining the expressive power of First-Order Logic (FOL) and the ability to deal with uncertainty of probabilistic graphical models. As opposed to most traditional machine learning approaches, learning and reasoning in MLNs is not restricted to a feature vector of fixed length, but is rather performed on whole databases of entities and relations. More formally, an MLN consists of a set of formulas $F$ in First-Order Logic and a real-valued weight $w_i$ attached to each of those formulas $F_i$. The probability distribution over the set of possible worlds $\mathcal{X}$ represented by the MLN is defined as follows:

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_i w_i n_i(x)\right), \tag{5}$$

where $x$ is a complete truth assignment to all predicate groundings $X$ (i.e. one possible world), $n_i(x)$ is the number of true groundings of formula $F_i$ in $x$, and $Z$ is normalization constant.

From a logical point of view, the outputs of the feature annotators in RoboSherlock can be regarded as tables in a relational database and thus naturally correspond to predicates in FOL, and the segmented clusters represent the domain of discourse of entities we wish apply probabilistic, logical reasoning to. Furthermore, we can think of the final class label, i.e. the object identity we wish to predict, as an additional predicate. As an example, consider a scene of two objects $c_1$ and $c_2$, where the ensemble of experts have identified $c_1$ being a yellow-ish box with a “Dr Oetker” brand logo on it, and $c_2$ being a round, blue thing. We can capture such a scene in a relational database as follows:

$$
\begin{aligned}
&\mathit{shape}(c_1, \mathit{Box})\\
&\mathit{color}(c_1, \mathit{Yellow})\\
&\mathit{goggles\_logo}(c_1, \text{“}\mathit{Dr\ Oetker}\text{”})\\
&\mathit{color}(c_2, \mathit{Blue})\\
&\mathit{shape}(c_2, \mathit{Round})\\
&\\
&\mathit{object\_type}(c_1, \mathit{Vitalis\_Cereals})\\
&\mathit{object\_type}(c_2, \mathit{Bowl}),
\end{aligned}
$$

where we have manually added information about object classes in the *object_type* predicate. In MLNs, it is straight-forward to create a model putting object attributes into relation with their class labels, since they provide a simple, declarative template language for generating probabilistic models. If we assume, for instance, that we can conclude from the shape of an object to its type, a set of weighted formulas such as

$$
\begin{aligned}
w_1 = \log(0.66) \qquad & \forall x.\ \mathit{shape}(x, \mathit{Round}) \wedge \mathit{object\_type}(x, \mathit{Bowl})\\
w_2 = \log(0.33) \qquad & \forall x.\ \mathit{shape}(x, \mathit{Box}) \wedge \mathit{object\_type}(x, \mathit{Bowl})
\end{aligned}
$$

can be added to the model, which naturally represent the rules “everything is a round bowl” and “everything is a box-shaped bowl”. In this work, we pick up the convention that constant symbols in FOL are written uppercase whereas variables are written lowercase. Of course, the above rules do not hold for most of the entities we encounter in the real world and, in fact, they can be considered as mutually exclusive. However, according to (5), the probability distribution defined by this MLN indicates that any world in which we encounter a round bowl is twice as likely as a world in which we find a box-shaped bowl. Following this manner, we add such abstract, coarse “rules of thumb” modeling connections between the ensemble experts and the final ensemble decision. The weight parameters of the resulting MLN can be learned in a supervised learning fashion. In the next chapter, we will describe the model we are using in RoboSherlock in more detail.

We think that applying a statistical relational model such as Markov Logic Networks is superior to other approaches for coming to a final ensemble decision for several reasons, a couple of which shall be depicted in the following:

1. **Simultaneous classification of an arbitrary number of objects.** As mentioned above, MLNs are able to simultaneously take into account any arbitrary but finite number of objects for classification. This an important feature for a perception system like RoboSherlock, since it captures interaction between objects in a scene. If a classification system is aware of the probability of jointly encountering two objects of particular types, this can tremendously boost the classification accuracy. Encountering milk and cornflakes together on a table, for instance, is much more likely than finding cornflakes and ketchup.

2. **Confidence-rated output.** Compensation for annotator noise

3. **Full joint probability/generative models.** If there still is uncertainty about an object's identity after the classification process, the MLN's joint probability distribution over input data can be exploited to collect additional information. If the system, for instance, has a rather weak belief that some object is a box of cereals, the joint probability over feature inputs might suggest that there should be some logo of a cereal brand on it. In a second trial, the system could then run the goggles annotator (maybe on retaken images of the box) in order to explicitly search for the logo and hence actively collecting stronger evidence.

4. **Ease of extension.** integration of additional context information, which do not come from the perceptual system, such as task information from an upper knowledge base can be easily integrated into the MLN. Looking onto a breakfast table, it much more likely to encounter cereals, milk and a bowl than looking into a drawer.

An important step besides object hypothesis generation and annotation, which structure and enrich raw perceptual data, is to aggregate and *filter* the information contained in these annotations. This leads to better decisions about object identity, pose and class and can also provide consensus based confidence estimates. The benefit of combining different cues for the object perception task was already shown in [8], and the system presented there is integrated into the architecture described here.

As mentioned before, all UIMA types can automatically be stored in a MongoDB database, and we have implemented a CAS Consumer that takes every **Scene**, its **Cluster**s and their *Annotations* and inserts it as a timestamped, hierarchically structured observation (cf. Fig. 4). This serves multiple purposes: (1) storage as a memory mechanism for later evaluation, including problem formulations such as modeling object life cycles or common storage locations, (2) a knowledge source for feedback into the next iteration of scene processing, and (3) a backend for a convenient visualization web page. We plan to extend the functionality of this web page in the future with mechanisms to provide manual ground truth data, data labeling and annotation corrections. This is invaluable for training machine learning systems semi-automatically through the help of the tracking mechanisms, or as a general debugging tool.

Reconstructed information (completed, registered CAD models, object position life cycles etc.) can be reused as priors in future processing as well. Additionally, task adaptation requires to limit detectors to the ones that provide relevant information for the task at hand [39], and take into account e.g. intended object use (pouring, stowage, disposal, etc.) for selecting the right approach. Similarly, (task, domain, location) context can shape object database selection, generate relevant search regions, and select segmentation strategy, etc. as in [40].

The orchestration of the analysis process by flow controllers can be guided by annotations made in previous steps of the process. It can thus adjust the flow based on the task query but also as a response to environmental conditions such as object arrangement or lighting.

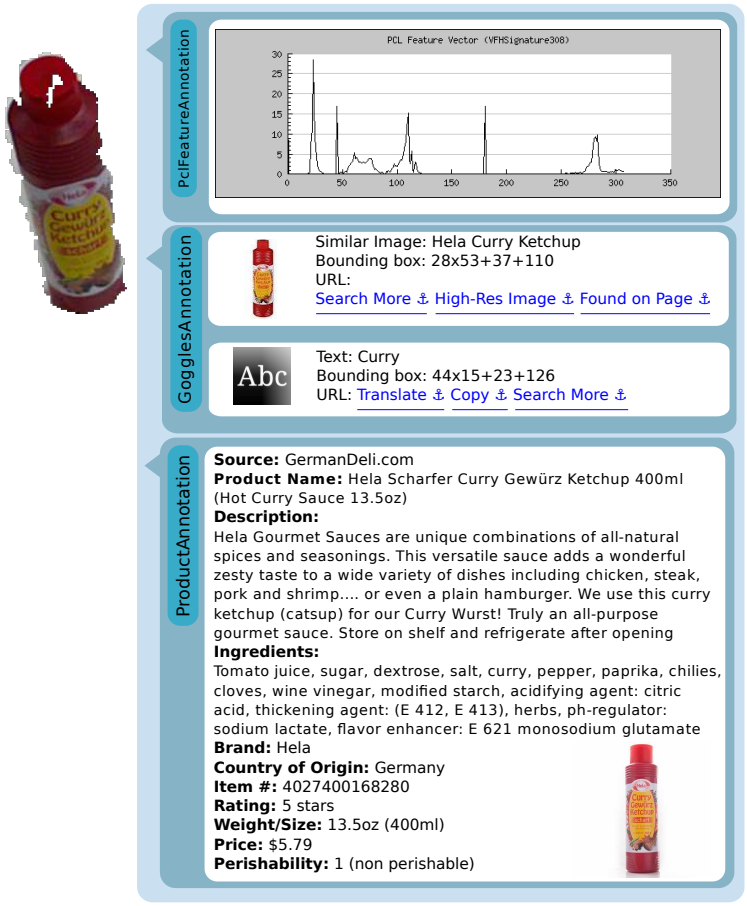



Figure 4: Results of RoboSherlock for an observation of a ketchup bottle.

# 7 Object Perception Type System

The type system is a fundamental component in our system and in UIM systems in general, since it allows independent development of components while enabling very heterogeneous data representation. It consists of the following types:

- basic (raw) data types
- object hypotheses
- object centric annotations

To be able to represent all information sent via ROS topics in our system, we provide an automatic converter that translates all available ROS message types into UIMA types. This includes core functionality such as transformation/kinematic chains (*tf*) or sensor data such as camera images and point clouds. Likewise, all PCL point types from are supported out of the box, as well as all OpenCV image types. For database storage, we opted to use

MongoDB, and created automatic conversion mechanism so all current and future UIMA types are available for "serialization" to Mongo objects.

The type system for storing object hypotheses consists of **ClusterPoints**, **ImageROI**, **TFLinkSphereROI**, **TFLinkBBoxROI** where all are subtypes of **ObjectHypothesis**, and can be used to describe a region of interest in either point cloud data, image data or in any defined *tf* coordinate system (cf. Sec. 7).

Object centric annotations are the types that store results from the different experts that analyze object hypotheses. We introduce some of the more important types that are used more frequently: **LocationAnnotation** is a supertype to define locations in 3D, in an occupancy, hierarchical or semantic map; **PclFeatureAnnotation** can hold any feature representation available in PCL; **PclSACAnnotation** can hold the model of a previously fitted SAC model **GogglesAnnotation** models the structure of responses from Google Goggles; **ClassificationAnnotation** is a flexible representation of classification results (cf. Sec. 4.3); **TrackingAnnotation** (cf. Sec. 5) holds information concerning frame-to-frame or scene-to-scene associations between object observations; **ProductAnnotation** is a particularly semantic piece of evidence, allowing the representation of ingredient lists, prices, or concepts such as perishability or heat sensitivity retrieved from e.g. online web stores; **BarcodeAnnotation** can store bar codes of various types for matching with product registries;

A **Cluster** is defined as a region (one or more **ObjectHypothesis** representations, e.g. an image region or a point cluster) linked with various **Annotations**. The **Scene** type contains a hierarchy of **Clusters**, the robot state and a **Timestamp** and as such represents a snapshot of a spatial arrangement of objects at a given time.

# 8 Experiments and Results

Since the contributions are neither individual algorithms nor a monolithic system, but a framework, and since it covers a considerably wider scope than previous work, it is hard to quantitatively assess the quality of the proposed approach. We therefore chose to evaluate key parts of the framework separately. We start by showing an illustrative example of an annotated cluster, followed by a demonstration of a key aspect of the given framework: richness of object information that the robot can generate *autonomously* with our exemplary application. We show, the power of the entity resolution in subsection 8.3, ending our evaluation with a preliminary fusion of the results from several experts in subsection 8.4.

## 8.1 Illustrative Example

Let us demonstrate the principles underlying RoboSherlock using an illustrative example. The robot is looking at the kitchen counter and acquiring an RGB-D scan of the scene with its Kinect sensor (cf. Fig. 1)

Fig. 5 shows an example of a point cluster of a cereal carton. A feature annotator creates a **PclFeatureAnnotation**, which is used by the classification framework to create a "box" **ClassificationAnnotation** with confidence 88.9%.

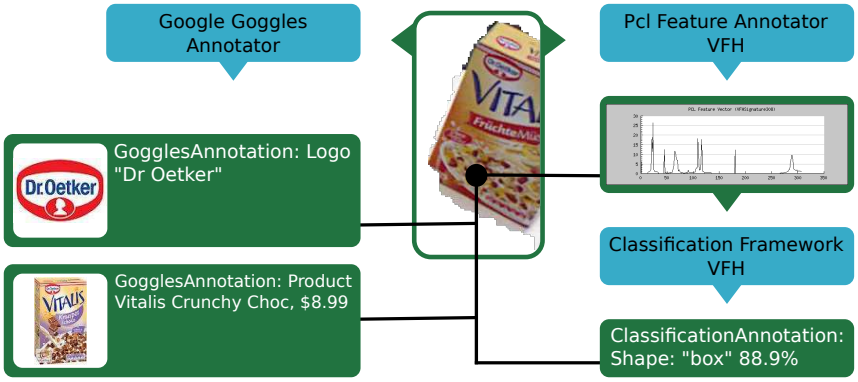



Figure 5: Annotators (blue) take an artifact, in this case a cluster of a cereal box, from the CAS. They create Annotations (green), which reference the artifact and can be (re-)used in different Annotators. The example shows the VFH [41]feature vector, a shape classification, as well as two Goggles results.

On the left hand side, the Goggles annotator uses the corresponding image region from the camera, uploading it for analysis by Google Goggles (cf. Sec. 4.2). The resulting **GogglesAnnotation** contains interesting information such as a brand logo and a product link.

Combining these annotations makes the construction of powerful mechanisms possible: RoboSherlock can extend these data-driven object descriptions with descriptions of object properties, such as volume, more specific names, prices, ingredients, product images, or even online search queries for additional resource retrieval, e.g.: i) text recognition (OCR) on a bottle label can be linked to ingredients list, container volume or price from an online store (c.f. Sec 4.2);ii) bar codes can be used in publicly available product registries; iii) the concept of e.g. perishability of milk can be grounded in the observation of a specific brand of milk through the use of ontologies (e.g. OWL-DL [36]).

## 8.2 Richness of Annotation

As mentioned in the introduction of this Section we resort to demonstrate the key aspect of the given framework: richness of object information.

To give a specific example, Fig. 4 depicts the contents of different Annotations that were obtained for the ketchup bottle in a single snapshot: A **PclFeatureAnnotation** of type VFH, a **GogglesAnnotation** with two results, and a **ProductAnnotation** with a positive database match from GermanDeli.com. Some annotations, e.g. **ColorHistogram** and **TrackingAnnotation** have been dropped for simplicity.

The Goggles results are of two types: *Similar Image*, along with the name “Hela Curry Ketchup”, and three URLs, pointing to a Google search page, a high resolution image, and the page on which the image has been found. The second hit is OCR *Text* detection, “Curry”, with links offering e.g. a translation. It is evident that these two strings by themselves are very informative and could be used in a variety of contexts, e.g. matching a user voice

| Annotation Type | # occurrence |
|---|---|
| *Scene Annotations* | |
| Scene | 103 |
| Logging Information | 103 |
| Location Annotation | 103 |
| Cluster Similarities Matrix | 103 |
| Cluster Distances Matrix | 103 |
| *Annotations for all Clusters* | |
| Cluster | 587 |
| Reference Cluster Points | 587 |
| Tracking Annotation | 587 |
| Standalone Image ROI | 587 |
| Image Mask | 587 |
| Perceptual Hash | 587 |
| Point Cloud | 587 |
| Point Indices | 587 |
| PCL Features | 587 |
| Shape Classification | 587 |
| Hue Saturation Histogram | 587 |
| *Annotations for selected Clusters* | |
| Named WWW Link | 766 |
| Goggles Annotation | 269 |
| Logo | 66 |
| Similar Image + Desc. | 62 |
| User Submitted Result + Desc. | 56 |
| Text Recognition | 48 |
| Product | 37 |

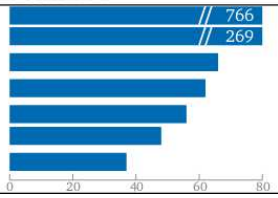



Figure 6: Histograms of object annotations

command to "pass the ketchup" or to narrow down a list of matches of similar products based on flavor.

Based on the **ImageROI** of the cluster, a match has been found in the database of GermanDeli products, and the resulting **ProductAnnotation** contains a list of key-value pairs, some generic (e.g. source and product name), some specific to the source (e.g. ingredients and perishability). From this, it can be deduced that e.g. the bottle is not stored in the fridge, and it gives rise to the idea of the robot ordering a new bottle if so instructed. A robot deployed in a clinic might point out the sugar content before handing it to a diabetic patient.

In order to demonstrate the rich information gathering capabilities of RoboSherlock, we set up 103 scenes (like the ones in Fig. 7) in our lab kitchen environment, on top of two counters, containing 587 object observations. Note that we are not trying to evaluate our object hypotheses generations, thus objects were arranged in the scenes at a distance large enough, without occlusions, that they are easily separable using a basic 3D segmentation and Euclidean clustering.

We used textured (e.g. cereal) and untextured (e.g. bowl) objects, ones with protruding 3D shapes and ones without (e.g. cutlery which in 2D data blended into the table plane). Combining hypotheses from multiple complementary segmentation methods (3D table top segmentation and color segmentation) yields better object clusters than the methods for themselves, and while we do not get every annotation for each object, we consistently get several (multi-modal) descriptions for each view of an object. A subset of the set of objects that have been used for the experiment is shown in Fig. 7.

The range of objects varied from small items such as a bottle cap and cutlery to relatively


Figure 7: Example scenes and clusters of some of the objects used in the experiments.

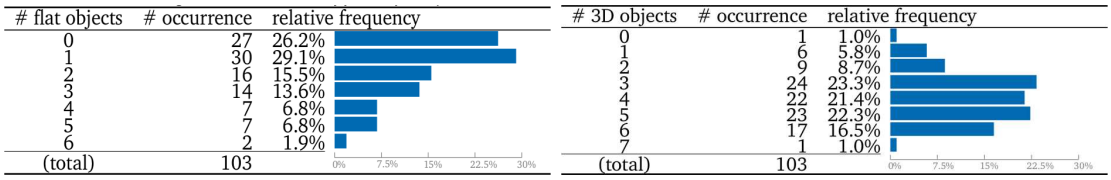



Figure 8: Histograms of flat and 3d objects in the scene

large items, e.g. a pancake maker or cereal containers. Several objects were rather flat (e.g. a book and a magazine, cutlery), others were three-dimensional (e.g. bottles or boxes), and there were textured (e.g. food products, drink containers) as well as non-textured objects (e.g. plates and cups). All in all, except for transparent objects, we attempted to create a comprehensive set of objects that occur in regular household environments.

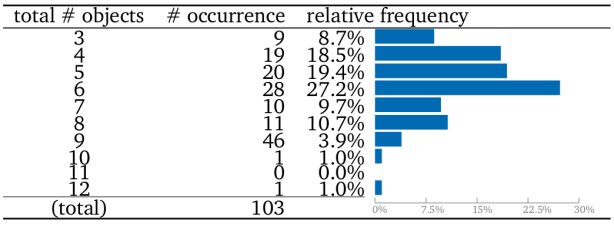



Figure 9: Histograms of objects found per scene

In Fig. 6 we show, the occurrence of several annotations that resulted during the experiment. Some experts annotate all of the clusters found during execution (e.g. Shape Classification), while other like the GogglesAnnotation are there only for certain clusters. Even though the scope of this work not being how we fuse these informations from different experts but offering a framework where this is made possible, one can easily see how certain annotations can complement each other: e.g. in the case of an erroneous shape classification we can rely on annotations that come from web sources or other experts in order to create a more accurate model of the object at hand.

Histograms showing the distribution of objects over the test data are shown in Fig. 8 and 9. Although it is a very simplistic example the aforementioned histograms show how the two experts complement each other and come up with better object hypotheses. Having better segmentation algorithms (e.g. ones that deal with cluttered scenes) would furthermore improve the capabilities of the system.

| | URDF-RegionFilter | PointCloud-ClusterExtractor | SmallObject-Annotator | GenericFeature-FromNormalsEstimator | Cluster-Tracker | ClusterGoggles-Annotator | TOTAL |
|---|---|---|---|---|---|---|---|
| Mean | 48.5 | 1513.8 | 741.4 | 265.6 | 101.4 | 4945.7 | 7616.5 |
| STD | 6.0 | 850.3 | 166.0 | 91.5 | 36.7 | 1446.0 | 2023.9 |

Table 2: Average and standard deviation of the recorded runtimes (in milliseconds) for the different processing steps.

The total processing time and the ones for the individual modules is shown in Table 2. The online query made to Google Goggles takes the longest time, but as these are issued asynchronously it is not so problematic. Extracting clusters from the input frame was the second slowest step, but it has lots of potential for improvement. Annotating and tracking the objects was performed quite efficiently, and overall the runtimes were fast enough, if no immediate **GogglesAnnotation** is required.

## 8.3 Entity Resolultion

As described in 5, a tracking mechanism is necessary in order to find corresponding object hypotheses from different time frames, detect objects that disappeared or reappeared in the scene or realize that a certain hypotheses is novel. In order to assess the validity, performance and robustness of our approach, we performed a series of experiments. In order to be able to illuminate various aspects with statistical significance, we implemented a synthetic test case that simulated a number of tables where actions were performed to put, remove, and move objects around, as well as change their appearance to simulate different partial views of an object. A virtual sensor with limited view frustum was used to generate noisy measurements of a subset of the objects, which was used to infer the actions that were performed as well as the the belief state of all objects in the environment.

We used 4 tables of 1 square meter area each and a sensor that is placed at a randomly changing position in front of any one table that can see up to 1 meter in width. We used a set of 15 objects of random color.

For each object in the sensor's view, we generated an observation by adding Gaussian noise with a standard deviation of $\sigma_1 \in \{0.025, 0.05, 0.075, 0.1\}$ per color channel to the color vector, and we used the same principle (and the same additive Gaussian noise) in two dimensions to simulate a measurement for the true position.

The computation of *similar* and *proximity* is then performed per pair of old and new measurements by computing the euclidean distances between them and applying $g$ with a standard deviation of $\sigma = 0.25$:

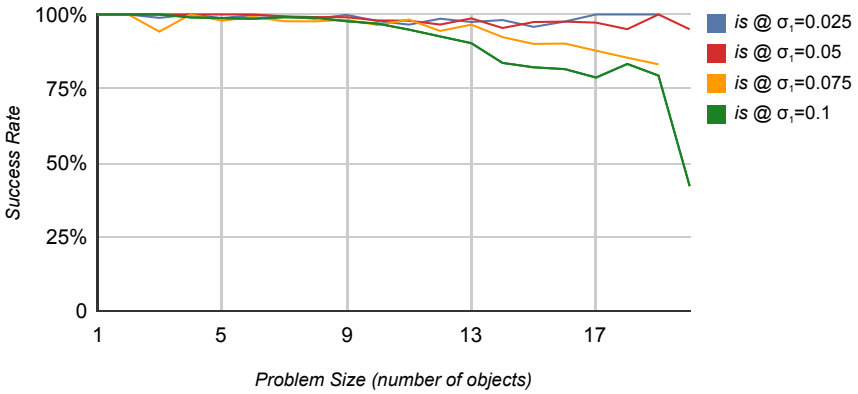
$$similar(o, n) = g(c(o) - c(n)) \tag{6}$$

$$proximity(o, n) = g(p(o) - p(n)). \tag{7}$$

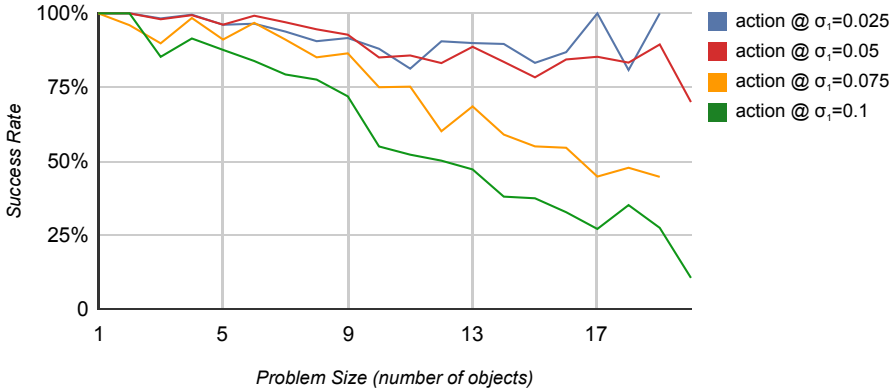
We evaluated the success rate of the object identity resolution method depending on noise level and the problem size (number of objects involved) in the inference step. This is shown in Figure 10c, where you can see that the cluster association (subfigure a) has a very high success rate throughout the entire domain of problem sizes, up until the highest level of noise we evaluated. It is clear that as the number of objects increase, the chance of mistaking objects for one another rises. The correct explanation for what *happened* to a certain object (subfigure b) is more susceptible to noise, and fails faster for larger problem sizes. This is due to the fact that the various explanations are directly modeled after their respective combination of evidence predicate configurations.

Surprisingly, our method is not that sensitive to the number of actions that were performed before observing the scene and inferring what happened. In Figure 10, we show the

(a) Success rate of correct cluster association (mode of *is*).




(b) Success rate of correct action explanation (*stay*, *move*, *newview*, *appear*, *hidden*, *disappear*).




(c) Success rates of inference results over problem size (number of objects involved in inference), shown for increasing levels of noise ($\sigma_1$).

same success rates, graphed over the number of actions, and it can be seen that the cluster association problem is not much affected by performing 5 actions between observations. For moderate noise levels, even the action explanation can mostly deal with this issue. At high noise levels, performance begins to drop to the point where only 3 or 4 out of 5 performed actions were inferred correctly.

Example results for one of the scenarios from the experiment described in Section 8.2 are shown in Fig. 11c. Shown in each sub figure is a visualization of *similar* and *proximity* predicates (left,center) and the inference results showing the mode of the distribution over *is* as well as the predicates *appear* (A), *hidden* (H) *disappear* (D). New clusters are shown on the top, old clusters along the left side. In subfigure *a* there are two identical plates and one of them was moved, while in subfigure *b* one of the objects was replaces with another one. In both cases the entity resolution system successfully identifies the action that took place.

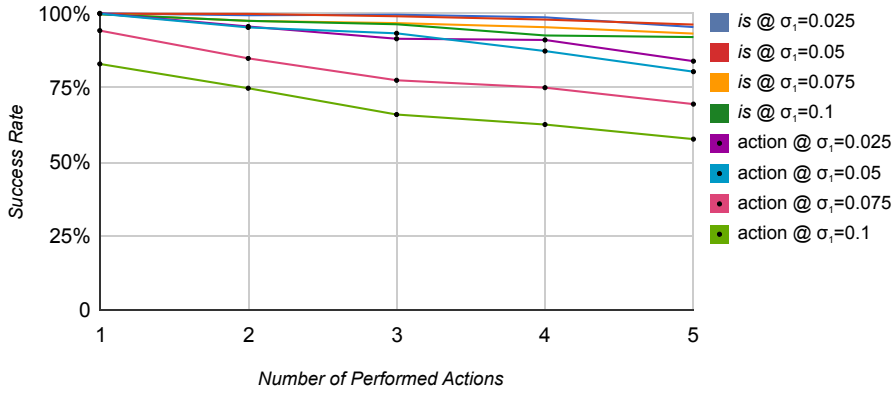
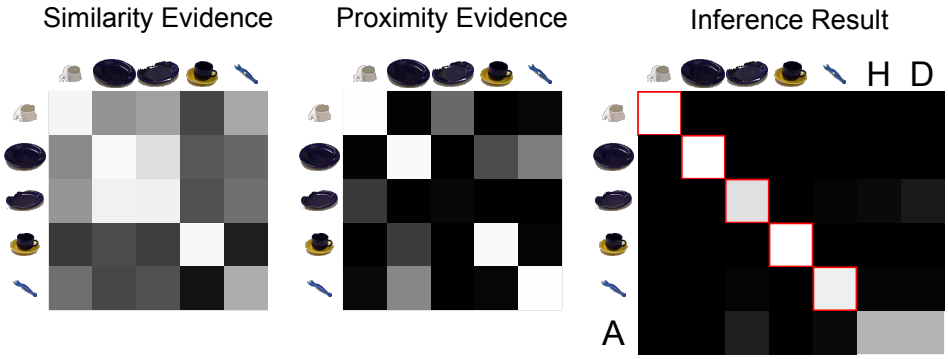
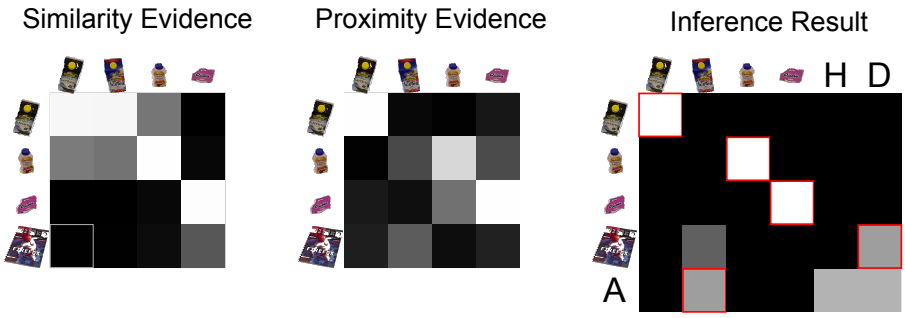



Figure 10: Success rates of inference results of cluster association and action explanation as a function of the number of actions performed. Errors are plotted separately for different noise levels.

(a) There are two identical blue plates, one of which has been moved. The association is still inferred correctly.




(b) The magazine disappeared, the ice tea bottle appeared, the remaining object stayed.




(c) Exemplary results showing various effects of the object identity resolution for a single time step each

## 8.4 Information Fusion

We conducted experiments on a data set of 50 realistic scenes, each comprising about 3-8 objects. For each scene, we had the class label for each cluster in the scene as well the context information manually annotated. For the MLN we used for obtaining a final ensemble decision of experts, we used the following predicates, which correspond to the annotator outputs in the RoboSherlock system:

- *shape*(*cluster*, *shape*). Determines the shape of a cluster found in the scene. Possible values for *shape* are $dom(shape) = \{Round, Flat, Box\}$.
- *color*(*cluster*, *color*). Determines the colors found in the RGB image of a cluster. Possible values for *color* are $dom(color) = \{Red, Yellow, Blue, Green, White, Black\}$.
- *goggles_Text*(*cluster*, *text*). Output from the Google goggles text annotator. Can be an arbitrary string.
- *goggles_Logo*(*cluster*, *company*). Output from the Google goggles brand logo annotator. Returns company names, such as "Dr Oetker" or "Kellogs".
- *goggles*(*cluster*, *product*)
- *goggles_Contact*(*cluster*, *contact*)

Two additional predicates have been used for specifying knowledge about the current context the perceptual task is conducted in and for assigning a class label to each of the clusters in the scene at hand:

- *scene*(*scene*). Determines the current context in which the perceptual task is being performed. possible contexts are $dom(scene) = \{Breakfast, Cooking, Drawer, Fridge\}$
- *object*(*cluster*, *object!*) Assigns an class label to a cluster. In our experiments, we distinguished 25 different kinds of objects (see also Table 3). The "!" operator specifies that this predicates is to be treated as a functional constraint, i.e. for every cluster $c \in dom(cluster)$, there must be true exactly one ground atom.

The following Markov Logic Network has been designed in order to model correlations between annotator outputs (i.e. the object properties) and the object classes:

$$
\begin{aligned}
w_1 &\quad \forall c, s, col, obj.\ shape(c, +s) \wedge color(?c, +col) \wedge object(?c, +obj) \\
w_2 &\quad \forall c, comp, obj.\ goggles\_Logo(c, +comp) \wedge object(c, +obj) \\
w_3 &\quad \forall c, text, obj.\ goggles\_Text(c, +text) \wedge object(c, +obj) \\
w_4 &\quad \forall c, prod, obj.\ goggles(c, +prod) \wedge object(c, +obj) \\
w_5 &\quad \forall c, contact, obj.\ goggles\_Contact(c, +contact) \wedge object(c, +obj) \\
w_6 &\quad \forall s, obj.\ scene(+s) \wedge object(c, +obj) \\
w_7 &\quad \forall c_1, c_2, t_1, t_2.\ object(c_1, +t_1) \wedge object(c_2, +t_2) \wedge c_1 \neq c_2,
\end{aligned}
$$

where the "+" operator specifies that the respective formula will be expanded to one individual formula for every value in the respective domain. The weights have been determined by supervised learning of the manually labeled data using pseudo-log-likelihood learning. Since learning and inference in MLNs is a computationally expensive task, we had to restrict our experiments to using only 50% of the scenes we recorded. Table 3 shows the confusion matrix of a 5-fold cross validation on this data set.

| Prediction/Ground Truth | Coffe_pitcher | Coffee_pitcher | Kellogs_corn_flakes | Pfanner_ice_tea | Pfanner_orange_juice | Pringles_chips | Vitalis_cereal | bowl | coffee | cup | fork | hot_plate | ja_oil | ketchup | knife | milk | nutella | pancake_mix | plate | pot | salt | spatula | spoon | sugar | tea | toaster |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Coffe_pitcher | **0** | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| Coffee_pitcher | 0 | **0** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Kellogs_corn_flakes | 0 | 0 | **1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Pfanner_ice_tea | 0 | 0 | 0 | **0** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Pfanner_orange_juice | 0 | 0 | 0 | 0 | **2** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Pringles_chips | 0 | 0 | 0 | 0 | 0 | **0** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 1 | 0 | 0 | 0 |
| Vitalis_cereal | 0 | 0 | 0 | 0 | 0 | 0 | **2** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| bowl | 0 | 0 | 0 | 0 | 1 | 0 | 0 | **1** | 1 | 2 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| coffee | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **0** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| cup | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 2 | 1 | **3** | 2 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| fork | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 2 | **1** | 0 | 0 | 0 | 0 | 2 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| hot_plate | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ja_oil | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | **1** | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| ketchup | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **4** | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| knife | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | **3** | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| milk | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 1 | **4** | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| nutella | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | **0** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| pancake_mix | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | **2** | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| plate | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | **5** | 0 | 0 | 5 | 4 | 0 | 0 | 0 |
| pot | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **2** | 0 | 0 | 0 | 0 | 0 | 0 |
| salt | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | **0** | 0 | 0 | 0 | 0 | 1 |
| spatula | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | **1** | 0 | 0 | 0 | 0 |
| spoon | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | **1** | 0 | 0 | 0 |
| sugar | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | **1** | 0 | 1 |
| tea | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **0** | 0 |
| toaster | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | **1** |

Table 3: Confusion matrix resulting from 5-fold cross validation on 50% of the scenes by applying the MLN for Information Fusion.

# 9 Conclusion and Future Work

This paper proposes RoboSherlock, a software framework for implementing perception systems that can scale towards real-world perception task complexity as defined above.

We know of no other robot perception system aiming at providing perceptual information about objects of daily use that are as rich as the ones computed by the RoboSherlock application sketched above. It can provide robots with the evidence required to answer queries such as: where is $X$? where has $X$ been? and what was on the table this morning? More interesting are queries such as "is there milk on the table?" (mapping of general "milk" concept to a specific brand) or does $X$ go into the fridge?.

Yet, these capabilities are only the tip of the iceberg. The UIMA framework offers sophisticated methods for combining results with confidence. A number of open source tools exist in the UIMA software libraries enabling extraction and use of knowledge from the world-wide web. The machine learning infrastructure within the UIMA framework is particularly sophisticated and promising, impressively demonstrated with the Watson [17] system winning the *jeopardy!* challenge. Investigating the intriguing prospect of whether the impact of these methods on robot perception is equally strong is on our agenda for future research.

# Acknowledgements

This work is supported in part by the EU FP7 projects *RoboHow* (grant number 288533) *ACAT*(grant number 600578) and *SAPHARI* grant number 287513.

# References

[1] D. Pangercic, M. Tenorth, D. Jain, M. Beetz, Combining Perception and Knowledge Processing for Everyday Manipulation, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Taipei, Taiwan, 2010, pp. 1065–1071.

[2] M. Tenorth, S. Profanter, F. Balint-Benczedi, M. Beetz, Decomposing cad models of objects of daily use and reasoning about their functional parts, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Tokyo Big Sight, Japan, 2013, accepted for publication.

[3] D. Kragic, M. Vincze, Vision for robotics, Found. Trends Robot 1 (1) (2009) 1–78. doi:http://dx.doi.org/10.1561/2300000001.

[4] A. Collet Romea, M. Martinez Torres, S. Srinivasa, The MOPED framework: Object recognition and pose estimation for manipulation, International Journal of Robotics Research 30 (10) (2011) 1284 – 1306.

[5] T. Mörwald, J. Prankl, A. Richtsfeld, M. Zillich, M. Vincze, Blort- the blocks world robotic vision toolbox, in: "Best Practice Algorithms in 3D Perception and Modeling for Mobile Manipulation Workshop" - CD (in conjunction with the IEEE ICRA 2010), 2010.

[6] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, , N. Navab, Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes.

[7] K. Lai, L. Bo, X. Ren, D. Fox, Sparse distance learning for object recognition combining rgb and depth information, in: Proc. of Int. Conf. on Robotics and Automation (ICRA), 2011.

[8] Z.-C. Marton, F. Seidel, F. Balint-Benczedi, M. Beetz, Ensembles of Strong Learners for Multi-cue Classification, Pattern Recognition Letters (PRL), Special Issue on Scene Understandings and Behaviours AnalysisIn press.

[9] K. Duncan, S. Sarkar, R. Alqasemi, R. Dubey, Multi-scale superquadric fitting for efficient shape and pose recovery of unknown objects., in: IEEE International Conference on Robotics and Automation (ICRA), 2013.

[10] R. B. Rusu, W. Meeussen, S. Chitta, M. Beetz, Laser-based Perception for Door and Handle Identification, in: Proceedings of the International Conference on Advanced Robotics (ICAR), Munich, 2009, best Paper Award.

[11] T. Rühr, J. Sturm, D. Pangercic, D. Cremers, M. Beetz, A generalized framework for opening doors and drawers in kitchen environments, in: IEEE International Conference on Robotics and Automation (ICRA), St. Paul, MN, USA, 2012.

[12] A. Oliva, A. Torralba, Modeling the shape of the scene: A holistic representation of the spatial envelope, International Journal of Computer Vision 42 (2001) 145–175.

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

[14] G. Bradski, The OpenCV Library, Dr. Dobb's Journal of Software Tools.

[15] S. Gould, O. Russakovsky, I. Goodfellow, P. Baumstarck, A. Y. Ng, D. Koller, The stair vision library (v2.4), http://ai.stanford.edu/ sgould/svl (2010).

[16] D. Ferrucci, A. Lally, UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment, Natural Language Engineering 10 (3-4) (2004) 327–348.

[17] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager, N. Schlaefer, C. Welty, Building Watson: An overview of the DeepQA project, AI Magazine 31 (3) (2010) 59–79.
URL `http://www.aaai.org/ojs/index.php/aimagazine/article/view/2303`

[18] A. Aldoma, Z.-C. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. B. Rusu, S. Gedikli, M. Vincze, Tutorial: Point Cloud Library – Three-Dimensional Object Recognition and 6 DoF Pose Estimation, Robotics & Automation Magazine 19 (3) (2012) 80–91.

[19] I. Lysenkov, V. Eruhimov, G. Bradski, Recognition and Pose Estimation of Rigid Transparent Objects with a Kinect Sensor, in: Proc. of Robotics: Science and Systems, Sydney, Australia, 2012.

[20] A. K. Jain, R. P. W. Duin, J. Mao, Statistical pattern recognition: A review, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (1) (2000) 4–37.

[21] J. Sill, G. Takács, L. Mackey, D. Lin, Feature-weighted linear stacking, CoRR abs/0911.0460.

[22] M. Sewell, Ensemble learning, available online. (2007).

[23] K. M. Ting, I. H. Witten, Stacked generalization: when does it work?, in: Procs. International Joint Conference on Artificial Intelligence, Morgan Kaufmann, 1997, pp. 866–871.

[24] L. Lam, C. Y. Suen, Optimal combinations of pattern classifiers, Pattern Recognition Letters 16 (9) (1995) 945–954.

[25] J. Bohren, R. B. Rusu, E. G. Jones, E. Marder-Eppstein, C. Pantofaru, M. Wise, L. Mosenlechner, W. Meeussen, S. Holzer, Towards Autonomous Robotic Butlers: Lessons Learned with the PR2, in: ICRA, Shanghai, China, 2011.

[26] M. Muja, R. B. Rusu, G. Bradski, D. Lowe, REIN - A Fast, Robust, Scalable REcognition INfrastructure, in: ICRA, Shanghai, China, 2011.

[27] K. Okada, M. Kojima, S. Tokutsu, T. Maki, Y. Mori, M. Inaba, Multi-cue 3D object recognition in knowledge-based vision-guided humanoid robot system, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). (2007) 3217–3222.

[28] A. Pronobis, O. Martínez Mozos, B. Caputo, P. Jensfelt, Multi-modal Semantic Place Classification, The International Journal of Robotics Research 29 (2-3) (2010) 298–320. doi:10.1177/0278364909356483.

[29] O. M. Mozos, Z. C. Marton, M. Beetz, Furniture Models Learned from the WWW – Using Web Catalogs to Locate and Categorize Unknown Furniture Pieces in 3D Laser Scans, Robotics & Automation Magazine 18 (2) (2011) 22–32.

[30] T. Gao, D. Koller, Active classification based on value of classifier, in: Adv. in Neural Info. Proc. Systems (NIPS), 2011.

[31] D. Ferrucci, E. Nyberg, J. Allan, K. Barker, E. Brown, J. Chu-Carroll, A. Ciccolo, P. Duboue, J. Fan, D. Gondek, E. Hovy, B. Katz, A. Lally, M. McCord, P. Morarescu, B. Murdock, B. Porter, J. Prager, T. Strzalkowski, C. Welty, W. Zadrozny, Towards the Open Advancement of Question Answering Systems, Tech. Rep. RC24789, IBM Research Report (2009).

[32] I. Horswill, Integrating vision and natural language without central models, in: In Proceedings of the AAAI Fall Symposium on Embodied Language and Action, 1995.

[33] D. Pangercic, M. Tenorth, B. Pitzer, M. Beetz, Semantic object maps for robotic housework - representation, acquisition and use, in: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vilamoura, Portugal, 2012.

[34] M. Waibel, M. Beetz, R. D'Andrea, R. Janssen, M. Tenorth, J. Civera, J. Elfring, D. Gálvez-López, K. Häussermann, J. Montiel, A. Perzylo, B. Schießle, O. Zweigle, R. van de Molengraft, RoboEarth - A World Wide Web for Robots, Robotics & Automation Magazine 18 (2) (2011) 69–82.

[35] K. Lai, D. Fox, Object recognition in 3d point clouds using web data and domain adaptation, The International Journal of Robotics Research 29 (8) (2010) 1019–1037.

[36] M. Tenorth, U. Klank, D. Pangercic, M. Beetz, Web-enabled Robots – Robots that Use the Web as an Information Resource, Robotics & Automation Magazine 18 (2) (2011) 58–68.

[37] N. Blodow, D. Jain, Z.-C. Marton, M. Beetz, Perception and Probabilistic Anchoring for Dynamic World State Logging, in: 10th IEEE-RAS International Conference on Humanoid Robots, Nashville, TN, USA, 2010, pp. 160–166.

[38] M. Richardson, P. Domingos, Markov Logic Networks, Machine Learning 62 (1-2) (2006) 107–136. doi:http://dx.doi.org/10.1007/s10994-006-5833-1.

[39] Z. C. Marton, D. Pangercic, N. Blodow, M. Beetz, Combined 2D-3D Categorization and Classification for Multimodal Perception Systems, The International Journal of Robotics Research 30 (11) (2011) 1378–1402.

[40] A. Aydemir, K. Sjöö, J. Folkesson, A. Pronobis, P. Jensfelt, Search in the real world: Active visual object search based on spatial relations, in: Robotics and Automation (ICRA), 2011 IEEE International Conference on, IEEE, 2011, pp. 2818–2824.

[41] R. B. Rusu, G. Bradski, R. Thibaux, J. Hsu, Fast 3d recognition and pose using the viewpoint feature histogram, in: Proceedings of the 23rd IEEE/RSJ IROS, Taipei, Taiwan, 2010.

# Inpainting of Missing Values in the Kinect Sensor's Depth Maps Based on Background Estimates

M. Stommel, M. Beetz and W. L. Xu

***Abstract*—Low-cost game controllers such as Microsoft's Kinect sensor provide dense, real-time depth measurements of indoor environments at high framerate. The sensor is based on the principle of active stereo using structured infrared light. For surfaces that distract infrared light, no measurements can be obtained. It is important to replace missing values early on, to avoid a slow subsequent conditional evaluations or the propagation of errors into neighbouring regions. To solve this problem we present an inpainting method that adds missing values based on background estimates of the unoccluded scene. It is therefore not necessary to hypothesise missing regions based on similarity to other image regions. The procedure also avoids a blurring between foreground and background. By adapting the method to the specific properties of the Kinect (and comparable) cameras, we were able to keep the complexity of the algorithm low, so high speed can be achieved.**

***Index Terms*—Kinect, missing values, real-time, stereo, depth map, background subtraction.**

## I. Introduction

RECENT advances in sensors and computers have inspired the application of computer systems in increasingly complex tasks and situations, often containing aspects of human-computer interaction. Optical sensors are attractive for such scenarios because they allow for a a contactless, non-intrusive sensing of the environment.

Microsoft's kinect sensor has probably been the sensor with the highest influence on the robotics and mechatronics communities in the last three years. Originally proposed as a game controller, many researchers have explored additional applications in healthcare [1], robotics [2], [3], video surveillance [4], or the facilitation of communication [5]. It is primarily the low price of the device that motivates researchers to explore the kinect device in applications [4] that have formerly been solved even by non-optical sensors [6], [7], although alternative depth sensors with comparable performance exist [8], [9].

### A. Principle of Operation of the Kinect Sensor

The kinect sensor can shortly be characterised as a stereo system consisting of an infrared projector and an infrared camera at a baseline of 7.5cm. Colour images are provided by an additional camera. The projector illuminates the scene by a quasi-random speckle pattern generated by an IR laser with 830nm wavelength and a set of diffraction gratings. The pattern is known and used to establish stereo correspondence between projector and camera. The speckle pattern resolves most ambiguities found in passive stereo systems, thus providing dense depth maps even for large homogeneous surfaces. Specular surfaces like glass windows deflect infrared light causing missing values in comparatively large areas. Light absorbing materials such as black cloth can also cause gaps. In our experiments, we noticed holes produced by the following objects: the windows, posters behind glass, the glass front of an oven, a pair of black jeans, a checkerboard pattern used for camera calibration, computer screens made of plastic, a glass window in a door, an empty plastic bottle. Depth discontinuities cause additional gaps because either the projected or the reflected light is obscured.

Based on an inspection of hardware and system output, Konolige and Mihelich [10] conjecture that a $9 \times 9$ or $9 \times 7$ pixel correlation window in an infrared image downsampled to $640 \times 480$ pixels is used to establish stereo correspondence. The correlation seems to be measured over 64 [10] or 88 [11] neighbouring pixels. An unknown sub-pixel interpolation algorithm increases the resolution to $1/8$ pixel. The depth is transformed to metric distance values as output. The image resolution is typically $640 \times 480$ at 30Hz, which is comparable to a time-of-flight camera. In contrast, a mechanically tiltable laser scanner has a frame rate in the orders of a magnitude of 1Hz. The selection of other frame rates and sizes is primarily limited by the maximum throughput of the USB 2.0 port. The Asus Xtion camera which is built according to the same reference design by PrimeSense also provides images of $640 \times 480$ pixels size, although in our experiments the actual resolution was only $320 \times 240$ pixels. The registration of the infrared camera with the colour camera causes missing values at the border of the depth map. The vertical and horizontal field of view is $43^\circ \times 57^\circ$, which is comparable to a time-of-flight camera, whereas laser scanners achieve up to $180^\circ$.

### B. Accuracy of the Sensor

The biggest influence on the accuracy of the sensor is noise in the disparity measurements. Since the difference between two consecutive disparity values corresponds to a quadratic term with respect to depth, the resolution decreases quadratically with increasing depth [12]. In most experiments with the Kinect sensor, noise is assumed to be additive zero-mean Gaussian [12], [13]. The quantisation steps of the depth values increase with distance according to the noise properties. The step width varies between 0.65mm at 50cm distance [14], 7cm at 5m distance [12], and 685mm at 15.7m distance [14]. Different sources report different ranges of operation depending on the resolution required for a particular application. In the *Kinect for Windows*, a hardcoded threshold limits the

M. Stommel and W. L. Xu are with the Department of Mechanical Engineering, University of Auckland, New Zealand, e-mail: m.stommel@auckland.ac.nz. M. Beetz is with the Institute for Artificial Intelligence, University of Bremen, Germany. This work is supported by the European Community, within the FP7 ICT-287513 SAPHARI project.

output to 40–300cm. Additional to the error of individual devices, Boehm [15] find a span of up to 20mm in the errors measured at distances between 0.7m and 1.2m over multiple devices from the same series. Qualitatively, devices of different brand are often regarded as comparable [8]. Interference of the emitted light pattern with objects from the scene [16] and complex structures in residual errors [8], [14] limit the effectiveness of traditional noise filters. Laser scanners are often chosen as ground truth for Kinect depth measurements because of their clearly superior accuracy [8], [13].

### C. Inpainting of Missing Values

Missing values need to be corrected because continuous checks for validity impair efficient real-time programming. Untreated, they tend to spread over the whole image plane within a few filter applications.

Inpainting methods aim at the reconstruction of damaged or missing image areas by propagating the surrounding image structure in the isophotes direction into the gap [17]. Structural inpainting methods use diffusion filters or global optimisation to find an intensity function of minimum complexity and error at the boundary of the gap. Gradient based methods lead to a good fit with respect to the isophotes, but produce a noticeable blurring in textured areas and large gaps [17], [18].

Textural inpainting approaches aim at reconstructing non-smooth image regions. For thin stripe-shaped regions, missing texture can be learned from the global image statistics and represented by exponential family distribution [19]. Large gaps can be inpainted by exemplar-based approaches, where the image area is filled by one or multiple patches taken from another image region. Suitable patches can be taken from images other than the inpainted one [20]. Correspondences can be found by comparing landmarks and determining a suitable transformation for the patch [21]. The more influential approach is however to inpaint image patches with similar local texture [22]. Like in structural approaches, the inpainted texture is aligned along the isophotes at the border of a gap. The identification of matching texture often requires CPU-intensive block-matching strategies.

Colour image based inpainting methods are not generally transferable to depth maps [23] because of the lack of texture and the different roles of foreground and background. Small gaps can be closed by local operators [9]. In multi-camera setups, occluded surfaces can be added if visible in another camera [24], [25]. Because of sensor interference, this is not possible for the Kinect sensor. A particular difficulty for inpainting are gaps over depth discontinuities. In order to accurately determine the outline of the foreground object, depth has been jointly evaluated with colour [23]. For small gaps and under the assumption of piece-wise smoothness and constancy, depth values from the border of the gap can be propagated to edge detections in the colour image. For large gaps, these detections are however often ambiguous. For view synthesis and other situations where the contour cannot be clearly determined from the colour image, exemplars of detected background areas with matching depth and colour structure can be painted in [26]. Such methods are however computationally costly.

### D. Methods for Background Estimation

In order to solve the problem of estimating the correct contour of foreground objects over large areas of missing values (as caused e.g. by the window panes), we propose to use background estimates of the unoccluded scene.

Background estimation is traditionally performed by filtering out transient objects from a video sequence by applying different kinds of temporal bandpasses [27]. Pixel values are usually modelled by a single Gaussian [28] or a mixture of Gaussians [29] (MOG). The method is particularly suited for relatively controlled scenarios where speed is more important than accuracy [30], [31]. Despite the different roles of depth and colour, MOG has also been applied to depth maps with a certain success [9]. A related fast approach uses a codebook of background values per pixel [32] thereby decoupling pixels from using a common filter speed.

Many refinements of the classical methods aim at the development of the statistical foundation [33]. A kernel density estimation allows for a more accurate modelling of colour distributions [34]. New findings in subspace learning and compressed sensing allow for a training of the background model from sparse samples [35], [36]. Another direction of research focusses on new features like highly sensitive local gradient descriptors [37] or detectors for surface convexity [38] to detect camouflaged objects.

Since colour does only provide limited geometrical information, colour has been fused with depth information. By concatenating colour and depth to a 4d-tupel, Gaussian mixtures can be used [39]. The different roles of depth and colour, and in particular the noise behaviour of passive stereo result in a weighting of the algorithms towards colour information [39].

Motivated by the higher robustness of the Kinect sensor, this study sets out to analyse a primarily depth based method. A related approach has been proposed by Gordon et al. [40] but not been applied because of sensor limitations at that time. The direct accessibility of the relevant information will be used to simplify and speed up the estimation process. Additional information about colour and surface normals will be used to resolve remaining ambiguities.

### E. Outline of the Proposed Method

In order to achieve a real-time inpainting of missing values, we combine a low complexity inpainting method with a depth based background estimator. Since background estimation algorithms are used in many mechatronics and robotics applications, the approach may not neccessarily require additional computational cost. The algorithm starts with a non-linear mapping of the depth map to a disparity map in order to normalise noise. A pyramid based inpainting method provides a fast guess of a preliminary, dense disparity map. The disparity map is updated by background estimates as soon as they are available. The background estimates are updated based on the disparity map.

In the following, we will first describe the sensor characteristics as they appeared in our experiments. We will then introduce the basic filters needed in our inpainting algorithm. After that, we will introduce a mode-maximising background

estimator. The technique is designed especially for the dense depth maps as provided by the kinect sensor. Due to the sensor characteristics of traditional stereo, the method has not been applicable before. We will then present the inpainting method and give experimental results.

## II. Sensor Characteristics and Basic Algorithms

In order to adapt our filters to the data, we repeated some of the above-mentioned experiments on the noise behaviour of the kinect sensors. In parts, the practical measurements seem to deviate strongly from the simulations and experiments [12], [13] described in the first sections.

### A. Normalisation of the Standard Deviation

To validate the quadratic dependency between noise and depth [12], we recorded 2500 frames of the office scene shown in Fig. 1a by the Asus Xtion. For each pixel position that provided valid depth measurement over more than 1250 frames, we computed mean and standard deviation. Figure 1b shows that the standard deviation non-linearly increases with depth although less smooth than the ideal result. A high number of sharp spikes indicates that some depth values have a considerably higher standard deviation than others. As Fig. 1c shows, the spikes can not be explained by a lack of data. Since the exact quadratic curve has been found for measurements of plane surfaces perpendicular to the optical axis, we conclude that the more natural scene has a complex influence on the measurements.

Figure 2a shows the distribution of the depth values along a fixed scanline for 2000 frames of a scene. The nonlinear increase of the quantisation steps with depth is clearly visible. The size of the quantisation steps compensates for the increasing variance at higher depth. Fig. 2b indicates that for a given pixel position, the depth is approximately normal distributed around a mean. Deviations from the Gaussian can be seen as relatively clear side maxima. They might result from reflections of the infrared signal within the scene.

We used the Matlab `nlinfit` function to find a good polynomial approximation of the quantisation levels produced by the Asus Xtion. The found mapping is

$$norm\_disp = a_1 - a_2 \cdot (a_3 \cdot depth^2 + a_4 \cdot depth)^{-1} \quad (1)$$

with $a_1 = 828.6404$, $a_2 = 3.1992 \cdot 10^5$, $a_3 = -1.93115 \cdot 10^{-9}$, and $a_4 = 0.9352$. The mapping transfers depth values back to the so called *normalised disparity* obtained as an intermediate result after the stereo matching [10]. Ideally, the mapping establishes a constant variance over the range of depth values. This is an important property for fast thresholding techniques in real-time programming. Considering the strong effect of the scene itself on the measurements, a high number of outliers from the ideal Gaussian distribution must be expected. For efficient use, the mapping is stored in a look-up table.

### B. Pyramid-Based Inpainting

A pyramid-based inpainting method is used to provide a fast estimate of missing values in the unoccluded scene.

**Algorithm 1:** Selective $3 \times 3$-median filter

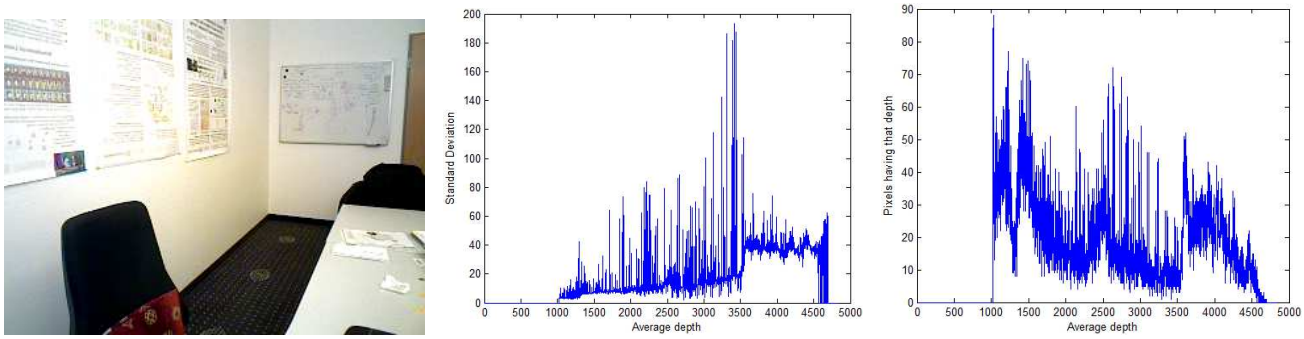
**input** : $d(x,y)$// Depth map with gaps
**output**: $\tilde{d}(x,y)$// Median filtered depth map
**for** *all* $x, y$ **do**
  $W = \left\{ d(\tilde{x},\tilde{y}) : \left\| \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} - \begin{pmatrix} x \\ y \end{pmatrix} \right\|_1 \leq 1 \wedge d(\tilde{x},\tilde{y}) \neq \square \right\}$;
  **if** $W = \emptyset$ **then**
    $\tilde{d}(x,y) = \square$;
  **else**
    $\tilde{d}(x,y) =$ median $W$;
  **end**
**end**

For reasons of speed, we decided against exemplar based or diffusion based methods. Instead, we decided for a median filter because it is very fast, interpolates between the borders of a gap, but does not blur over edges. It is therefore appropriate to filter disturbances caused by glass panes and thin areas of occlusion. Complex interferences with foreground objecs will be addressed later. In order to filter large gaps without using large (and therefore slow) filter windows, we combine the local median filter with an image pyramid.

We build the image pyramid by iteratively smoothing and subsampling the image. For smoothing, we use a selective $3 \times 3$-median filter. The filter considers local windows of $3 \times 3$ pixels and picks the median among all valid depth values. After every smoothing, a selective 50 percent downsampling is performed. If in the downsampling multiple depth values are mapped to the same destination coordinate, the median of the values is computed. The term *selective* means that missing values will not be considered in the computation of the median or the downsampling. The downsampled result is again smoothed and subsampled if it is wider and higher than one pixel. The original depth map is considered as the lowest level of the pyramid. The subsampled depth maps form the higher levels. For accuracy, Alg. 1 shows the pseudo-code for the selective median filter.

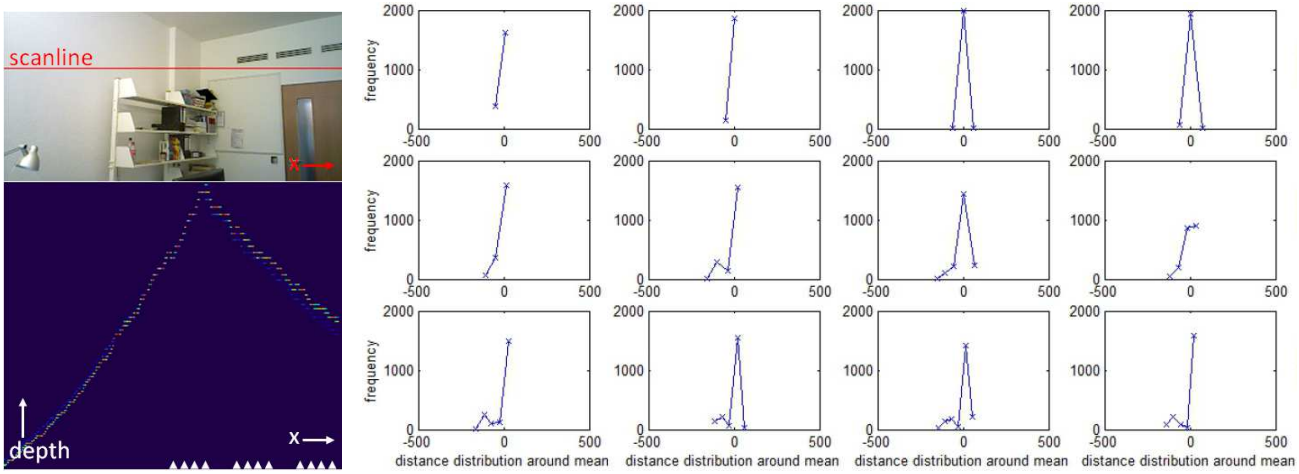
In order to replace missing values, the pyramid based inpainting algorithm scans through the base level of the pyramid and tests for missing depth values. If a missing value is detected, the coordinate is iteratively mapped to the next higher pyramid level until a valid result is found. For clarity, Alg. 2 gives the pseudo-code of the procedure.

The smoothing and downsampling of an image of width $w$ and height $h$ by the proposed selective filters can be done in time $O(wh)$ because the filter size is fixed. The number of pyramid levels is $\log \min w, h$. Since each pyramid level has four times less pixels than the adjacent lower level, the sum of all pixels in the pyramid is determined by a geometric series with limit $4wh/3$. The pyramid based inpainting requires a walk through the depth map and in the worst case the ascent to the top of the pyramid for every pixel. The search and replacement of missing values therefore has runtime of $O(wh)$ in the best case and $O(wh \log \min w, h)$ in the worst case. From the pseudo-code is becomes clear, that the procedure




(a) Test image (b) Standard deviation (c) Histogram of the depth values

Fig. 1: Variation of depth values around per pixel means averaged over all pixels of the same depth




(a) Histogram of depth along a scanline (b) Depth histograms on the scanline at columns marked by △ (centred to mean)

Fig. 2: Histograms of depth values along a scanline and at single pixels on the scanline

Fig. 3: Frames 0, 23, 45, 68, and 89 of a video sequence recorded in an indoor environment.

(a) Unprocessed depth map (b) Gradient magnitude (c) Gradient direction

Fig. 4: Depth map with large gaps (taken from the video sequence shown in Fig. 3) and gradient of the inpainted depth map

**Algorithm 2:** Pyramid based inpainting of missing values

**input** : $d(x,y)$// Depth map with gaps
**output**: $d(x,y)$// Depth map without gaps
Create image pyramid $pyr(x,y,level) : \mathbb{N}^3 \rightarrow \mathbb{N}$ with $top$ levels;
**for** *all* $x,y : d(x,y) = \square$ // where $\square$ is a gap **do**
  $level = 0$;
  $\tilde{x} = x$;
  $\tilde{y} = y$;
  **while** $level < top \wedge pyr(\tilde{x},\tilde{y},level) = \emptyset$ **do**
    $level = 1 + level$;
    $\tilde{x} = \tilde{x}/2$;
    $\tilde{y} = \tilde{y}/2$;
  **end**
  $d(x,y) = pyr(\tilde{x},\tilde{y},level)$;
**end**

also has small constant factors to the asymptotic run-time. The method therefore seems lightweight enough for real-time applications.

### C. The Three-Sliding-Bins Estimator

The three-sliding-bins estimator is a little technical trick we employed to determine the mode of a series of measured angles. The estimator is a histogram of three equally spaced bins. Let $b_i$ be the position of bin $i$ and $h_i$ the number of values assigned to bin $b_i$. The position of a *reference bin* $b_0$ is defined as a running average of all values assigned to it. The other bins are localised at relative spacings of $\pm 2\pi/3s$. Initially, $b_0$ can be set randomly, and $h_i = 0$. Incoming values are assigned to the nearest bin thereby increasing the respective frequency $h$. If a value is assigned to the reference bin, then the running average $b_0$ is updated. The other bins are recomputed so that the relative spacing is maintained. If a value is assigned to a bin different from the reference bin, then the bins are not updated. The reference bin $b_0$ usually converges quickly against the most frequent angle in the series.

## III. Inpainting Using Background Estimates

The role of the background estimation is to guide the inpainting process in areas where a foreground object occludes a background surface that does not allow for depth measurements. This prevents a blurring between foreground and background and resolves ambiguities of combined colour and depth based inpainting methods in large gaps.

### A. Mode Maximisation

Per pixel information is provided by the camera as a 4-tupel $(r,g,b,d)$ of red, green, blue, and depth. The pyramid based inpainting method described in Sec. II-B is used to fill holes. We then apply the non-linear mapping of metric depth values to the indices of their quantisation level according to Eq. 1. Based on the observation $p_t = (r_t, g_t, b_t, d_t)$ of a pixel at time $t$, we update the background estimate $\hat{p}_t = (\hat{r}_t, \hat{g}_t, \hat{b}_t, \hat{d}_t)$ from the background estimate $\hat{p}_{t-1}$ of the previous frame $t-1$ as the running average

$$(\hat{r}_t, \hat{g}_t, \hat{b}_t, \hat{d}_t) = \tau(\hat{r}_{t-1}, \hat{g}_{t-1}, \hat{b}_{t-1}, \hat{d}_{t-1}) + (1-\tau)(r_t, g_t, b_t, d_t) \quad (2)$$

if the new depth is greater than the background estimate minus a hysteresis parameter $\zeta$, i.e. if

$$d_t > \hat{d}_{t-1} - \zeta, \quad (3)$$

and if the depth at the pixel is currently increasing, i.e. if

$$d_t > d_{t-1}. \quad (4)$$

Parameter $\tau \in 0..1$ is a mixing constant we set to 20%. The first if-condition (Eq. 3) asserts that we are finding the largest mode, i.e. that the background estimate is maximally far away. Threshold $\zeta$ can only be used because we already have converted the depth values back to the normalised disparity according to Eq. 1. The second condition (Eq. 4) saves a number of unnecessary and often wrong updates. If no condition holds, the background estimate is left unchanged. The hysteresis parameter must be chosen in a way that the mode of the depth distribution is greater than $\hat{d} - \zeta$. Otherwise, the running average would miss the mode and converge to an outlier. As shown in Sec. II-A, outliers are a frequent phenomenon under realistic conditions. To prevent that singular outliers move $\hat{d} - \zeta$ away from the mode, an upper limit of $\hat{d}_{t-1} + \zeta - 1$ is imposed on the updated estimate. The background estimator may therefore need multiple frames to adapt to a newly discovered distant background. The delay is not problematic because it only occurs the first time (afterwards the background is known). It would be worse to have an increasing number false pixels that cannot be corrected over time.

If $d_t > \hat{d}_{t-1} - \zeta$, we also update a one-dimensional estimate $\hat{\phi}_t$ of the surface normal of the background. This adds sensitivity for static, rotated foreground objects to the method. The direction $\phi_t$ of the surface normal is computed as direction of the gradient of the depth map

$$\phi = \arctan(\partial d/\partial y, \partial d/\partial x). \quad (5)$$

Figures 4c and 4b show the gradient direction and magnitude.

The estimate $\hat{\phi}_t$ is the mode of a histogram over $\phi$ with three sliding bins at the angles $\hat{\phi} - 2\pi/3$, $\hat{\phi}$, and $\hat{\phi} + 2\pi/3$ (cf. Sec. II-C). For every pixel coordinate, a separate histogram ist stored and updated over all frames. The histogram with three sliding bins can be seen as a simple and fast approximation of a Gaussian mixture with three means. The bins other than the mode are designed to absorb noise and foreground.

Foreground segmentation is done by subtracting the background estimate from the current input image, resulting in a foreground weighting

$$w_t = ||p_t - \hat{p}_t||_1 + \beta \min(|\phi_t - \hat{\phi}_t|, 2\pi - |\phi_t - \hat{\phi}_t|). \quad (6)$$

Because of its speed, the Manhattan distance is used in the equation. The min-expression combines clockwise and counterclockwise distances in angles. The constant $\beta$ weights the influence of the surface normal and is set to $360/2\pi$ in our case. A pixel is detected as foreground if $w_t > \eta$. Over a number of training videos, the threshold was optimised to $\eta = 200$.

**Algorithm 3:** Inpainting using background estimates

Let $\hat{d}(x, y)$ be the depth of the background;
**for** *the video sequence* **do**
  record depth map $d$;
  $d_1 =$ Alg. 2 applied to $d$;
  apply the mapping of Eq. 1 to $d_1$;
  **for** *all* $x, y : d(x, y) = \square$ **do**
    $d(x, y) = \max(d_1(x, y), \hat{d}(x, y))$;
  **end**
  compute the gradient $\nabla d$;
  update $\hat{d}$ from $d$ and $\nabla d$ using the algorithm outlined in Sec. III-A;
**end**

### *B. Inpainting Using Background Estimates*

The background estimate of the depth map (Sec. III-A) is combined with the pyramid based inpainting method (Sec. II-B) by a maximum operation. Algorithm 3 gives the detailed sequence of the proposed sub-procedures. The method avoid overly high depth values because of the median filter in the pyramid based method. Too small values are avoided by the maximum principle. Outliers do not permanently damage the filter result.

The non-linear mapping can be done in linear time depending on the number of pixels by using a lookup table. The computation of the gradient is linear, too. The background estimator is a point operator that loops over all pixels. For every pixel, only running averages and conditional statements are executed. This can be done in constant time per pixel. The time complexity of the background estimator is thus linear in the number of pixels. All intermediate results can be stored in a vector of fixed size per pixel. No dynamically growing data structures are needed. The inpainting method therefore has a linear space complexity with respect to the number of pixels.

## IV. Experimental Results

To test the background estimator, we recorded a video sequence using a Microsoft Kinect sensor. The recorded scene is difficult in several ways. First, the windows and the big aluminium panel in the corner of the room distract the infrared light, so the depth values must be guessed in large parts of the image. Significant brightness changes are caused by people occluding the specular area of the floor. The movements of the people require a quick response of the filter. With up to 7.5m, the depth of the scene exceeds typical applications of the Kinect by a factor of two. This requires a robust noise handling. For comparison, we applied the GMG and MOG2 algorithms and parameterisations from the well known OpenCV library both to the colour image and - as proposed by Han et al. [9] - to the depth map. For evaluation, we computed the pixel-wise precision, recall, accuracy, and f-measure against a manually annotated groundtruth.

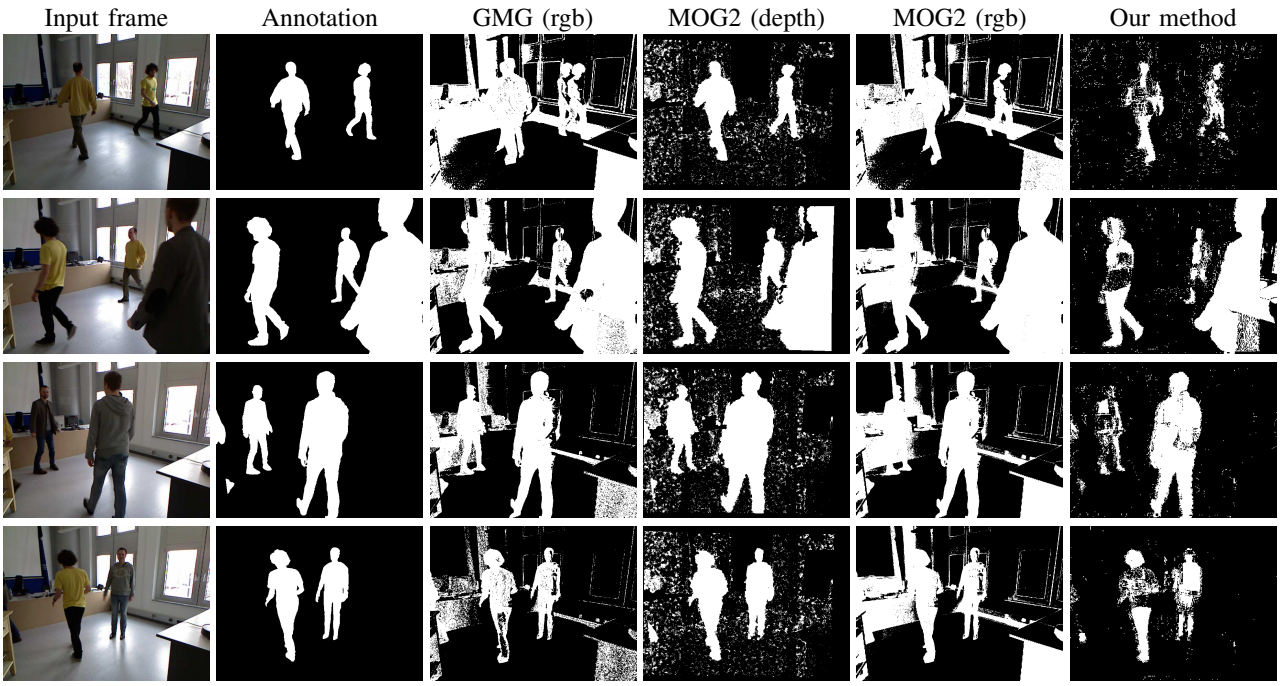
Figure 5 shows the results. GMG is left out because it did not work on the depth map. We obtained clearly the best results for the proposed method. The purely colour-based methods often mistook furniture for foreground. MOG2 on the depth map produces qualitatively similar results to our method. MOG2 is however less accurate and produces more false positives.

Figure 6 shows the advantage of inpainting using background estimates for a practical example. The original depth map for the example can be seen in Fig. 4a. The video sequence suffers from large areas of missing values at the windows and the glass front of the oven. Other problematic areas are the depth discontinuity at the kitchen unit on the left as well as several smaller spots scattered over the scene. As Fig. shows, the pyramid based method is able to correct most errors satisfactorily. However, the largest area of missing values, i.e. the window area, is spuriously assigned to the person in the foreground. As a consequence, a subsequent filter would not be able to detect the correct outline of the person. Using background estimates for inpainting resulted in the depth map shown in Fig. . The change in colour stems from the non-linear mapping. The window area has been assigned correctly to the background. As a consequence the contour of the person in the foreground becomes clear and can be retrieved by subsequent image analysis.

As an alternative to the median filtering in the pyramid based preliminary inpainting, we also tested a variant using a binomial filter. The method works exactly as the proposed method except that in the selective filter and the selective downsampling a binomial filter is used instead of a median filter. The method has minimal advantages in speed. As apparent from Fig. , the outcome is similar to the result of the median filter. Generally, the median filter has the advantage that it does not blend depth values over depth discontinuities. However, there are only few situations where one filter clearly outperforms another.

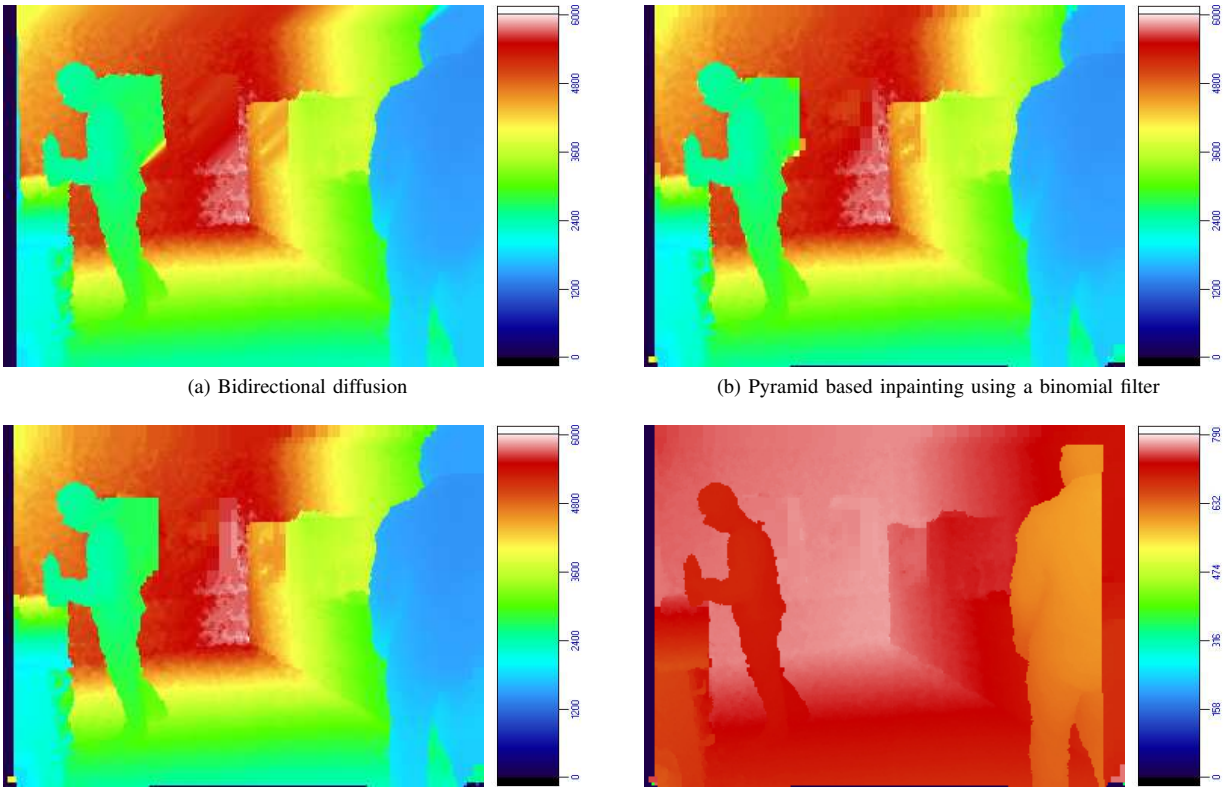
Secondly, we tested a bilinear diffusion of the depth values (Fig. 6a). In the filter chain, the diffusion filter was used instead of the pyramid based ones. The filter is computationally extremely slim because it only continues the depth values from one side of the gap to the inside by averaging with existing values from the lower left corner. The filter results however suffer from stripe like structures, ramps, and discontinuities. A slight advantage in the avoidance of contouring effects is balanced by the noticeable higher robustness of the pyramid based median filter. In summary, the results do not encourage to replace the pyramid based median filter by another filter of comparable complexity. We also do not expect qualitative improvements for the use of structural inpainting methods because many shortcomings in the preliminary inpainting are already corrected by using the background estimate.

A partly open question is if the proposed method is applicable to the depth maps of time-of-flight sensors. They are announced to be used in a new series of Kinect cameras. Because of the fundamentally different measurement procedure, it will not be appropriate to discretise the depth values according to the non-linear functions outlined in the introduction and our measurements. Instead, the accuracy of time-of-flight sensors depends strongly on the radial distance from the optical axis. Because of its high noise tolerance, the proposed method might be transferable with small modifications.




| | | | | |
|---|---|---|---|---|
| Precision | 0.52 | 0.58 | 0.46 | 0.80 |
| Recall | 0.91 | 0.84 | 0.94 | 0.75 |
| Accuracy | 0.83 | 0.87 | 0.80 | 0.93 |
| F-Measure | 0.63 | 0.68 | 0.60 | 0.77 |

Fig. 5: Test results



(a) Bidirectional diffusion

(b) Pyramid based inpainting using a binomial filter

(c) Pyramid based inpainting using a median filter

(d) Inpainting using background estimates

Fig. 6: Comparison of different inpainting methods based on local filtering

## V. Conclusion

In this study, we investigated a method for the inpainting of missing values in the depth maps obtained from Kinect and related sensors. By using background estimates of the unoccluded scene, large gaps could be inpainted without blurring between foreground and background. The high number of outliers found in practical measurements is handled in real-time by robust statistics, a limited Gaussian model, and a sliding bin estimator. The method is shown to have constant time and space complexity per pixel. Experiments on video data including complex movements of foreground objects and large areas of missing values demonstrate the effectiveness of the method, also in comparison to other methods. The experiments support a primarily depth-based procedure in background estimation. The directness of the approach allows for fast computations at lower complexity compared to colour-based approaches.

## References


[1] X. Ning and G. Guo, "Assessing spinal loading using the kinect depth sensor: A feasibility study," *Sensors Journal, IEEE*, vol. 13, no. 4, pp. 1139–1140, 2013.

[2] M. Draelos, N. Deshpande, and E. Grant, "The kinect up close: Adaptations for short-range imaging," in *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2012 IEEE Conference on*, 2012, pp. 251–256.

[3] E. Machida, M. Cao, T. Murao, and H. Hashimoto, "Human motion tracking of mobile robot with kinect 3d sensor," in *SICE Annual Conference (SICE), 2012 Proceedings of*, 2012, pp. 2207–2211.

[4] Z. Zhang, W. Liu, V. Metsis, and V. Athitsos, "A viewpoint-independent statistical method for fall detection," in *Pattern Recognition (ICPR), 2012 21st International Conference on*, 2012, pp. 3626–3630.

[5] C. Sun, T. Zhang, B.-K. Bao, C. Xu, and T. Mei, "Discriminative exemplar coding for sign language recognition with kinect," *Cybernetics, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2013.

[6] T. Shany, S. Redmond, M. Narayanan, and N. Lovell, "Sensors-based wearable systems for monitoring of human movement and falls," *Sensors Journal, IEEE*, vol. 12, no. 3, pp. 658–670, 2012.

[7] C.-F. Lai, S.-Y. Chang, H.-C. Chao, and Y.-M. Huang, "Detection of cognitive injured body region using multiple triaxial accelerometers for elderly falling," *Sensors Journal, IEEE*, vol. 11, no. 3, pp. 763–770, 2011.

[8] B. Moller, I. Balslev, and N. Kruger, "An automatic evaluation procedure for 3-d scanners in robotics applications," *Sensors Journal, IEEE*, vol. 13, no. 2, pp. 870–878, 2013.

[9] J. Han, L. Shao, D. Xu, and J. Shotton, "Enhanced computer vision with microsoft kinect sensor: A review," *Cybernetics, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2013.

[10] K. Konolige and P. Mihelich, "Kinect calibration: Technical," 2012, accessed July 15th, 2013. [Online]. Available: http://www.ros.org/wiki/kinect_calibration/technical

[11] C. D. Mutto, P. Zanuttigh, and G. Cortelazzo, *Time-of-Flight Cameras and Microsoft Kinect*. Springer, 2013.

[12] K. Khoshelham and S. O. Elberink, "Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications," *Sensors*, vol. 12, pp. 1437–1454, 2012.

[13] J. Chow and D. Lichti, "Photogrammetric bundle adjustment with self-calibration of the primesense 3d camera technology: Microsoft kinect," *Access, IEEE*, vol. PP, no. 99, pp. 1–1, 2013.

[14] J. Smisek, M. Jancosek, and T. Pajdla, "3D with Kinect," in *Workshop on Consumer Depth Cameras for Computer Vision*, 2011.

[15] J. Boehm, "Natural user interface sensors for human body measurement," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XXXIX-B3, pp. 531–536, 2012.

[16] K. Essmaeel, L. Gallo, E. Damiani, G. De Pietro, and A. Dipanda, "Temporal denoising of kinect depth data," in *Int. Conf. on Signal Image Technology and Internet Based Systems (SITIS)*, 2012, pp. 47–52.

[17] V. C. C. B. M. Bertalmio, G. Sapiro, "Image inpainting," in *ACM SIGGRAPH*, 2000, pp. 417–424.

[18] Z. Tauber, Z.-N. Li, and M. Drew, "Review and preview: Disocclusion by inpainting for image-based rendering," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 37, no. 4, pp. 527–540, 2007.

[19] A. Levin, A. Zomet, and Y. Weiss, "Learning How to Inpaint from Global Image Statistics," in *International Conference on Computer Vision (ICCV)*, 2003, pp. 305–312.

[20] S. Mirkamali and P. Nagabhushan, "Depth-wise image inpainting," in *Pattern Recognition (ICPR), 2012 21st International Conference on*, 2012, pp. 141–144.

[21] S. H. Kang, T. F. Chan, and S. Soatto, "Landmark based Inpainting from Multiple Views," UCLA Math CAM, Tech. Rep. TR-CAM 02-11, 2002.

[22] A. Criminsi, P. Perez, and K. Toyama, "Object Removal by Exemplar-Based Inpainting," in *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2003, pp. 721–728.

[23] D. Miao, J. Fu, Y. Lu, S. Li, and C. W. Chen, "Texture-assisted kinect depth inpainting," in *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*, 2012, pp. 604–607.

[24] L. Wang, H. Jin, R. Yang, and M. Gong, "Stereoscopic inpainting: Joint color and depth completion from stereo images," in *Computer Vision and Pattern Recognition (CVPR)*, 2008, pp. 1–8.

[25] R. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *IEEE International Symposium on Mixed and Augmented Reality*, 2011.

[26] I. Daribo and H. Saito, "A novel inpainting-based layered depth video for 3dtv," *Broadcasting, IEEE Transactions on*, vol. 57, no. 2, pp. 533–541, 2011.

[27] M. Piccardi, "Background subtraction techniques: a review," in *IEEE International Conference on Systems, Man and Cybernetics*, 2004, pp. 3099–3104.

[28] C. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland, "Pfinder: real-time tracking of the human body," *IEEE TPAMI*, vol. 19, no. 7, pp. 780–785, 1997.

[29] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *International Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 1999, pp. 246–252.

[30] S.-C. S. Cheung and C. Kamath, "Robust techniques for background subtraction in urban traffic video," in *Proceedings of Video Communications and Image Processing*. SPIE Electronic Imaging, 2004, pp. 881–892.

[31] S. Herrero and J. Bescos, "Background subtraction techniques: Systematic evaluation and comparative analysis," in *Advanced Concepts for Intelligent Vision Systems*, ser. Lecture Notes in Computer Science, J. Blanc-Talon, W. Philips, D. Popescu, and P. Scheunders, Eds. Springer Berlin Heidelberg, 2009, vol. 5807, pp. 33–42.

[32] K. Kim, T. H. Chalidabhongse, D. Harwood, and L. Davis, "Real-time foregroundbackground segmentation using codebook model," *Real-Time Imaging. Special Issue on Video Object Processing*, vol. 11, no. 3, pp. 172–185, 2005.

[33] T. Bouwmans, F. E. Baf, and B. Vachon, "Background Modeling using Mixture of Gaussians for Foreground Detection - A Survey," *Recent Patents on Computer Science*, vol. 1, no. 3, pp. 219–237, 2008.

[34] A. Elgammal, D. Harwood, and L. S. Davis, "Non-parametric model for background subtraction," in *European Conference on Computer Vision*, 2000, pp. 751–767.

[35] V. Cevher, A. C. Sankaranarayanan, M. F. Duarte, D. Reddy, R. G. Baraniuk, and R. Chellappa, "Compressive Sensing for Background Subtraction," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2008, pp. 155–168.

[36] J. He, L. Balzano, and A. Szlam, "Incremental gradient on the grassmannian for online foreground and background separation in subsampled video," in *Computer Vision and Pattern Recognition (CVPR)*, 2012.

[37] S. Noh and M. Jeon, "A New Framework for Background Subtraction Using Multiple Cues," in *Asian Conference on Computer Vision (ACCV)*, 2012, pp. 493–506.

[38] A. Tankus and Y. Yeshurun, "Convexity-Based Visual Camouflage Breaking," *Computer Vision and Image Understanding*, vol. 82, no. 3, pp. 208–237, 2001.

[39] M. Harville, G. Gordon, and J. Woodfill, "Foreground segmentation using adaptive mixture models in color and depth," in *IEEE Workshop on Detection and Recognition of Events in Video*, 2001, pp. 3–11.

[40] G. Gordon, T. Darrell, M. Harville, and J. Woodfill, "Background estimation and removal based on range and color," in *Computer Vision and Pattern Recognition (CVPR)*, 1999, pp. 2459–2464.

# Model-Free Detection, Encoding, Retrieval and Visualisation of Human Poses from Kinect Data

M. Stommel, M. Beetz and W. L. Xu

***Abstract*—The recognition of humans in Kinect camera data is a crucial problem in many mechatronics applications with human-computer interaction. In order to improve the limited scope of many model-based methods, we propose a spatio-temporal segmentation of keypoints provided by a skeletonisation of the depth contours. A vector-shaped pose descriptor allows for the retrieval of similar poses and the combination with many machine learning libraries. A visualisation method based on the Hilbert curve provides valuable insight in the detected poses.**

***Index Terms*—Kinect, real-time, human body tracking, pose estimation.**

## I. Introduction

FOR many mechatronics applications it is fundamentally important to detect the human body from video data and estimate its pose. The automatic recognition of human poses allows for a technical system to assist humans in complex situations and cooperate with them in a meaningful and appropriate way. This could lead to what has been termed as *human oriented machinery* by Schweitzer [1]. Present examples of mechatronic systems that include the recognition of persons are patient monitoring and evaluation [2], [3], [4], man-machine interfaces [5], [6], [7], a robotic room [8], and rehabilitation systems [9].

The key to such flexible systems is situation awareness with respect to people that are present in a given scene. Situation awareness, as it is understood in this study, comprises the

1) detection of the people present in a scene,
2) extraction of the poses,
3) encoding of a pose by a feature vector,
4) ability to compare feature vectors to each other,
5) mapping of proximate poses to proximate feature encodings,
6) and ability to visualise the vector space.

The first to steps provide the necessary input data for the system. The level of detail at which poses are extracted is naturally application dependent. This study aims at the extraction of torso, head, and limb positions. The reason is that this is the level of detail provided by the current real-time depth sensors, and in particular Microsoft's Kinect sensor [10]. The recognition of smaller body parts like e.g. the hands requires the fusion of different sensor modalities [11] or close-up recordings [12], possibly using modified hardware [13].

M. Stommel and W. L. Xu are with the Department of Mechanical Engineering, University of Auckland, New Zealand, e-mail: m.stommel@auckland.ac.nz

M. Beetz is with the Institute for Artificial Intelligence, University of Bremen, Germany

This work is supported by the European Community, within the FP7 ICT-287513 SAPHARI project.

The encoding of the pose can be a simplified version of the extracted pose data. The pose should be encoded in a way that facilitates later steps. The encoding does not necessarily need to represent all extracted information. It also does not need to be illustrative or related to a kinematic model. Instead, it is important that the encoding preserves pose similarities and that it allows for mathematically convenient distance measurements. The preservation of pose similarities is important because it allows for the comparison of unknown poses to known poses and hence the classification of unknown poses. However, small errors can be tolerated. The preservation may for example be valid locally only. Modern machine learning methods like support vector machines are also able to map disconnected regions of a feature space to a common class label. A representation in the form of a vector allows for an easy integration with existing machine learning tools.

The possibility to visualise the feature space is important for the design of application software. It allows for the designer to develop a deeper understanding of the application domain and the data layout. This allows for an adequate system layout without using too simple or overly complex methods.

Additional constraints arise from the application context. Since real-time capabilities are required for many mechatronics applications, we also require them in this study.

Situation awareness as outlined here has two benefits: In simple mechatronics applications, the encoded pose can be used as an input to a control system. In complex applications, the encoded pose can be used to switch between multiple pose-specific processes. It could for example be possible to switch between different kinematic or action models depending on the perceived pose. Simple models could be used for simple poses, saving CPU time for the analysis of more difficult poses or the execution of other processes.

In this paper, situation awareness is solved by constructing a feature space for appearances of human poses:

- First, a combination of different filters is used to detect persons in a video and separate them from background. We use RGBD-cameras to record the input images.
- The next step is to perform a skeletonisation of the foreground, which reduces the search space to the medial axes of body and limbs. This is an important advantage over generic keypoint detectors which are usually based on local image structure only.
- Then we perform a spatio-temporal segmentation that provides the trajectories of individual persons in a scene. Compared to model based methods, the spatio-temporal segmentation has the advantage of being computationally slim while providing a similar result. At the same time, it is much more generic because it is not restricted to the

modelled scene type. The method is appearance based and requires only a small set of predefined rules. Since no training is needed, this brings a massive saving in CPU time (compared e.g. to the 1 day training time on a cluster with 1000 cores mentioned by Shotton et al. [14]).
- We compute an appearance based feature descriptor encoding the pose of every single person. We show that the feature descriptor preserves pose similarity and allows for a clustering and sampling of the space of pose appearances. This allows for situation awareness in scenarios with human-computer interaction.
- Finally, we visualise the feature space by linearising and aligning it along a Hilbert curve.

## II. Related Work

The recognition and tracking of persons has been addressed in different ways depending on the sensor configuation and area of application. For image retrieval, people must be recognised from single pictures. Approaches dealing with such data are typically contour based [15] or part pased [16]. Since geometrical information is only indirectly accessible, such methods require complex machine learning algorithms to be viable in limited tasks. For robotics applications such approaches are usually not fast enough and lack robustness.

In video surveillance, hardware cost is often critical. As a consequence, people must be detected and tracked from sequences of colour images. The use of dynamical information simplifies foreground/background segmentation, especially in the processing of sports videos [17]. Repetitive movements can be learned from spatio-temporal volumes [18] or sequences of limb configurations [19].

By using multiple cameras or depth sensors, it is possible to measure the 3D geometry directly [20]. The popular approach of marker-based motion capturing [21] is undesirable in most mechatronics applications because it requires very strong manipulations of the recorded scene. Instead, most approaches use a 3D model that represents configurations of body parts or a 3D mesh of the body surface. The use of a model limits the range of possible interpretations of the input data, which has a strong noise suppressing effect. Model-based approaches use global or mixed global/local optimisation to estimate the 3d-pose of a person. Global methods try to find the model parameterisation that best explains the observation. As a consequence, a high correspondence between model and data can be achieved [22]. Since mesh models have many degrees of freedom, global optimisation is computationally expensive.

For real-time applications, simplified stick models have been developed [23], [24], [14] which are adapted to feature based representations of the input images. Stick models represent the kinematics of the human body in a simplified way. Features used for adapting the model can be body part detections [14], skeletons [24] or Reeb graphs [25]. The use of intermediate feature representations reduces the search space compared to a purely global optimisation. Noise in the parameter estimation can be reduced by incrementally updating the results of the last video frames. However, often a manual initialisation of the model must be given.

The scope of a model is often limited [22]. Detailed 3D-meshes as well as silhouette based models often cannot adequately model clothing.

Most estimators assume particular scene properties like full body observations [14] or upper body observations [26]. In order to further constrain the scene interpretation, assumptions about the scene context, physics and actors are made [27].

The variety of models, each one with relatively small scope, exhibits a certain discrepancy between kinematic model and observation. The variety of possible appearances is usually not addressed. This study therefore aims at incorporating a higher level of awareness of the observed situation. To this end, we do not represent human poses by the parameters of a model with limited scope but by a pose sensitive descriptor of the appearance of a person.

Most of the existing pose descriptors are based on joint angles of a stick-model, extracted for example by the skeleton tracker of the Kinect SDK [28], [29], [30]. The scope of such descriptors is therefore limited to the scope of the underlying model. Lu et al. [31] present an appearance based solution by applying the shape context [32] to silhouettes of people. The descriptor is used to model action segments. In contrast to the work of Lu et al., our descriptor is based on the medial axes instead of the silhouette. This reduces the sensitivity to contour variations produces e.g. by clothing. Our descriptor also distinguishes angles of limbs, so it is more selective for differences in pose. Our approach differs also in the desired use: We demonstrate that the descriptor preserves pose similarity, how the feature space can be visualised, and that it encodes both important key poses as well as the recording situation. As a consequence, we establish situation awareness with respect to the pose of a person and the way it is depicted in the image, independently of the limited scope of a kinematic or surface model.

## III. Proposed Method

In a bottom-up procedure, the input images obtained from a Kinect camera are transformed into increasingly abstract representations: At first, we filter out noise and defects in the sensor data. The input images are then condensed to a keypoint based representation that focusses on aspects relevant to the recognition of persons. A segmentation algorithm detects single persons. The pose of a person is encoded by a feature descriptor that allows for visually plausible comparisons of pose and appearance. A visualisation based on the Hilbert curve is proposed to analyse the space of pose appearances.

### A. Image Filtering and Keypoint Detection

The Kinect sensor provides a colour image and a depth map of a scene. The sensor illuminates the scene by a known, static, locally unique speckle pattern. The depth map is computed according to the principle of stereoscopy as the inverse proportion of the disparity of the pattern as it appears to a built-in infrared camera [33]. The camera performs a non-uniform quantisation of the depth values according to a quadratic increase of the noise level [34]. As a consequence of the active illumination, the camera does not provide depth

measurements for surfaces that distract or absorb infrared light, e.g. window panes, computer monitors, or occasionally black cloth.

Missing values in the depth map are inpainted as the local medians in a pyramid image of the existing depth values. The method is real-time capable, scale-invariant, and works for most situations except for overlappings with foreground objects. In order to prevent in such situations, the scene background is estimated from the depth maps over time. The depth map is then corrected by using the maximum of the local median and the depth of the background.

The inpainted depth map is then transformed into a disparity map by applying the inverse of the quantisation function. Since the exact camera intrinsics are not relevant to our problem, our 'disparity' map just gives the index of the quantisation level of the original depth map. The mapping is done efficiently using a lookup table. The important property of the disparity map is that the noise level is uniform over depth.

In order to reduce the search space, we condense the input images (ca. $10^6$ pixels) to a comparatively small set of keypoints (ca. $10^3$). To this end, we first extract the contours of the foreground objects. by thresholding the gradient magnitude of the disparity map. The method is robust because persons stand out clearly in disparity and we have a uniform noise level over depth. Since isolated contour pixels disturb the following keypoint detector, we remove all connected components with less than 10 pixels from the contour. As a keypoint detector, we use a 2D skeletonisation [35] of the extracted contour pixels. Compared to corner based keypoint detectors (e.g. SIFT [36]), the method has the advantage that keypoints tend to be located on the medial axes of the limbs and body, which is a good starting point for pose extraction. A related approach is to use Reeb graphs [25]. They are however sensitive to the alignment of an elevation function. 3D approaches like Reeb graphs and 3D skeletons are also computationally costly, which makes them unattractive for real-time applications [24] Compared to depth invariant approaches [37], the method has the advantage that the number of keypoints created for the background is low.

To further reduce the number of keypoints in background areas, we discard all keypoints where the proportion of nearby foreground pixels to nearby background pixels is less than 20%. A pixel is considered 'nearby' if it is closer than the nearest contour pixel.

### *B. Scene Segmentation and Detection of Persons*

The keypoint extraction simplifies the detection of persons to a 2D clustering problem in the x-depth plane. Remaining difficulties are the consistency of cluster labels over time, the handling of elongated clusters (e.g. created by stretching out an arm), the determination of the number of clusters, and the absorption of noise into separated clusters.

In order to deal with these problems, we developed an algorithm that combines techniques from $k$-means, mean-shift, and minimum distance clustering.

The clustering algorithm stores the state of the current segmentation $s = \{k_{max}, k, r, M, A, L_\mu\}$ and updates it for every new frame. The values $k_{max}$ and $k$ are the maximum and the current number of means. The maximum number of means is a user definable parameter of the algorithm which we set to $k_{max} = 15$. Too small values reduce the ability of the algorithm to absorb noise. Large values slow down the algorithm. Parameter $k$ is set automatically by the algorithm. The variable $r$ denotes a radius around a mean (here $r = 50$cm). The state contains $k_{max}$ means $M = \{\mu_0, \mu_1, \ldots, \mu_{k_{max}-1}\}$ of which only the first $k$ are used.

**Algorithm 1:** The sub-procedure *increase_k* determines a set of means that covers a set of feature descriptors

**input** : State $s$ of the segmentation
Set $F$ of feature descriptors
**output**: State $s$ with additional means
bubblesort $k$ means $M$ by age $A$;
let cluster size $c = (c_0, \ldots, c_{k_{max}}) = 0$;
**for** *all* $f \in F$ **do**
  `// Search for mean` $\mu_j$ `near` $f$
  let $\tilde{M} = \arg\max_\mu \{a(\mu_i) : \mu_i \in M \wedge \|f - \mu\| < r \wedge i < k\}$;
  **if** $\tilde{M} \neq \emptyset$ **then**
    pick first $\mu_j \in \tilde{M}$ ;
    let $c_j = 1 + c_j$;
  **else**
    `// Increase` $k$ `if no mean matches`
    **if** $k < k_{max}$ **then**
      let $\mu_k = f$;
      let $a(\mu_k) = 0$;
      let $c_k = 1$;
      let $k = k + 1$;
    **end**
  **end**
**end**

Every mean has a segment label $l(\mu_i)$ and an 'age' $a(\mu_i)$. The state contains the vector of labels $L_\mu(l(\mu_0), \ldots, l(\mu_{k_{max}-1}))^\top$ and the vector of ages $A = (a(\mu_0), \ldots a(\mu_{k_{max}-1}))^\top$.

The clustering algorithm assigns a data points to a means. By $\mu(f)$ we denote the index of the mean assigned to a feature descriptor $f$. The function $\mu(F)$ represents the indices $\mu(f)$ of all feature descriptors $f \in F = \{f_0, f_1, \ldots, f_n\}$. The set $M$ contains therefore all means $\mu_i$ where $i = \mu(f_j)$ and $f_j \in F$ plus those means that have not been assigned to any descriptor in $F$.

The overall structure of the algorithm is as follows: At first, state $s$ is initialised. Then for every new video frame, the features $F$ are extracted. By calling the sub-procedure *increase_k*$(s, F)$, the number of clusters is determined. The cluster positions are refined by three iterations of the sub-procedures *local_neighbourhood*$(s, F)$, *mean_shift*$(s, F)$, and *reduce_k*$(s, F)$. The final assignments of features to clusters are determined by a nearest neighbour search.

The sub-procedure *increase_k* (see Alg. 1) adapts $k$ by performing a minimum distance clustering. At first the means are sorted by age. We use bubblesort because the number of means is small ($k_{max} = 15$). By setting the variable $c$ to zero,

**Algorithm 2:** The sub-procedure *local_neighbourhood* assigns data points to randomised, neighbouring means within radius $r$

**input** : State $s$ of the segmentation
Set $F$ of feature descriptors
**output**: Assignment $\mu(f)$ of means to descriptors
Compute $n_p$ permutations $P_g(h)$ of $h = 0, 1, \ldots, k-1$ where $g = 0, \ldots, n_p - 1$;
**for** *all* $f_i \in F$ **do**
  $\mu(f_i) = \min\{j : \|\mu_{P_{i \bmod n_p}(j)} - f_i\| < r\}$;
**end**

we mark all means as unused. We then assign every descriptor $f$ to the first mean within a distance $r$. This is done in order of age, so old and large clusters are systematically preferred. A matching mean is marked as being used. If the descriptor is not close to any mean, the algorithm checks if $k$ can be incremented. If all means are already in use, then $f$ is not assigned to any mean. The result of the sub-procedure is a full coverage of the descriptors by circular disks with radius $r$ around the means as far as it is possible with the given $k_{max}$.

The positions of the means are still very inaccurate. A small number of mean-shift iterations are conducted to move the means in the density maxima of the feature set. The sub-procedure *local_neighbourhood* (Alg. 2) assigns means to close feature descriptors, i.e. it computes $\mu(f)$. In the mean-shift step, the means are updated over the respective feature descriptors. Only one mean is assigned to one descriptor. The ambiguity of overlapping means (with respect to their radius) is resolved by randomisation. A set of four random permutations of means is precomputed in the beginning of the sub-procedure. A feature descriptor $f_i$ is assigned to the first mean found to have a distance from $f_i$ smaller than $r$. The means are evaluated in the order given by permutation $i$ modulo four.

The sub-procedure *mean_shift* recomputes the means as the arithmetic average

$$\mu_j = \frac{1}{c_j} \sum_{i:\mu(f_i)=j} f_i \tag{1}$$

of all feature descriptors assigned to that mean. The means are normalised by the cluster size $c_j = \|\{i : \mu(f_i) = j\}\|$. The age $a(\mu_j)$ is increased by the value $1 + \sqrt{c_j}$. The number 1 asserts a minimum increase. The root function serves as an approximate measure of the cluster diameter. The age has therefore the meaning of a 'diameter over time'.

As a result of the mean shift, multiple means can move to the same density maximum. The sub-procedure *reduce_k* (Alg. 3) removes such redundant means and lowers the value of $k$. To this end, the procedure test all means in decreasing order of their index. Means that have not been assigned to any feature descriptors are deleted by sorting them to the end of the list and reducing $k$. Non-empty means are tested for overlapping with other means. The overlap is detected by comparing the distance between the mean by a threshold $r_2$ which we set to 70% of $r$. The cluster with lower age is either

**Algorithm 3:** The sub-procedure *reduce_k* identifies redundant means and reduces $k$ accordingly

**input** : State $s$ of the segmentation
Radius $r_2 < r$
**output**: State $s$ with less means
**for** $i = k-1, k-2, \ldots, 0$ **do**
  **if** $c_i = 0$ **then**
    `// Delete empty cluster`
    $k = k - 1$;
    swap means $i$ and $k$;
  **else**
    `// Check overlap with other clusters`
    **for** *all* $j < i$ *where* $\|\mu_i - \mu_j\| < r_2 \wedge c_j \neq 0$ **do**
      **if** $a(\mu_i) < a(\mu_j)$ **then**
        `// Delete cluster i now`
        $k = k - 1$;
        swap means $i$ and $k$;
      **else**
        `// Delete cluster j later`
        set $c_j = 0$;
      **end**
    **end**
  **end**
**end**

deleted or marked for deletion. The usage of a second radius $r_2$ introduces a certain tolerance for small or close objects. Otherwise, small objects would be merged to overlapping nearby clusters with higher age.

The function used for swapping means is also used in the bubblesort algorithm (Alg. 1). It does not only swap the means but also the corresponding age and segment label. The segments therefore have a consistent label over multiple frames, although the indices of the means are permanently changed.

### C. Encoding, Comparison, and Visualisation of Poses

A histogram-based approach is chosen to encode the extracted segments as a feature descriptor. A regular $9 \times 9$-grid is imposed over the bounding box around the keypoints of a segments. For every grid cell, a four-bin histogram of the branch orientation of the skeleton is computed. A mild Gaussian smoothing is applied to increase the generalisability over small pose variations. The descriptor is the concatenation of all histograms. Figure 1 shows an example. The normalised correlation coefficient is used to compare the histograms.

Compared to graph-matching algorithms on skeletons, the approach is easier and more noise tolerant. It is not necessary to extract the endpoints of the skeleton, correct loops, close gaps, or consider dependencies between structure and feature values. Instead, the vector-shaped format of the feature descriptor makes it easy to combine the method with modern kernel based classifiers. Such methods easily map separate regions of a feature space to a common class label. It is

therefore not necessary to manually resolve ambiguities in the graph structure.

The space of pose descriptors is not Euclidean because the triangle equation is not satisfied. There must also be 'forbidden' zones in the feature space because many bins of the histograms are mutually exclusive. In order to visualise the space of pose descriptors, we apply Kruskal's minimum spanning tree algorithm to the data. The minimum spanning tree is then traversed in preorder. The resulting linearisation of the feature space is aligned along a Hilbert curve [38] which has the property of arranging clusters in the linearisation in compact 2D regions. The chosen procedure has the advantage that it does not rely on a metric space, averages, medoids, or a preset number of clusters.

## IV. Experimental Results

We conducted practical experiments in a kitchen environment. Our sample videos showed different actors performing manipulations with kitchen objects, e.g. placing objects on the cabinets, passing objects from one person to another, or wiping the table.

### A. Spatio-Temporal Segmentation

The spatio-temporal segmentation was able to detect people, track them over time, and automatically adapt the number of clusters. Figure 2 shows some results.

In the first example, the person is tracked while being handed over a kettle from a person outside the image. A new segment for the kettle and arm is created during the action. Additional clusters absorb spurious keypoints in the background of the right image border. The cluster labels remain stable during the handover task in spite of the touching segments. Since the method automatically creates new clusters for noisy detections, it can handle significant amounts of noise. Noise clusters are easy to recognise by a small size and age. The second example shows that the segment labels assigned to the two persons remain stable despite the large overlap in the image plane and interaction with foreground objects. An additional cluster is created for the newly discovered foreground objects (the ketchup bottle is detected by the background estimator after it has been moved for the first time). Little inaccuracies in the plot result from using a rectangle based drawing algorithm.

The limits of the method are people standing very close together and disruptions of the temporal trajectory.

We also compared the method to the graph-based segmentation by Felzenszwalb and Huttenlocher [39] but the computation time of that method was in the order of magnitude of up to one hour for a single frame.

### B. Encoding and Retrieval of the Pose

To evaluate the distance measure, we recorded 200 poses from a video sequence showing two people working in a kitchen. Near duplicate poses were detected and removed by comparing the CRC32 hash codes of the feature descriptors. For every pose, the distances to the other poses were computed and sorted by similarity in decreasing order. Figure 3 gives an impression of the results. The left image in a row shows the reference pose. The numbers give the normalised correlation coefficient scaled to the interval $[0\ 100]$. Due to the limited space, only poses with highest correlation and uncorrelated poses with a minimum correlation coefficent of zero (corresponds to a value of 50 in the figure) are given. Anticorrelated poses and poses with weak correlation are left out. The first rows show the result fore relatively complete skeletons. The last row shows the poses retrieved for a noise sample consisting of only one keypoint.

The retrieved poses show a high visual correspondence with the reference pose for normalised correlation coefficients between $0.5$ (i.e. $75$ in the figure) and $1.0$ (i.e. $100$ in the figure). With decreasing numerical correlation also the visual correspondence gradually decreases. The highest ranked search results are visually homogenous with only rare visual outliers. A consistent anticorrelation consists between the most clearly outlined skeletons and the most noisy detections. Areas of corresponding pose in feature space are thus not contaminated by spurious detections. From the results it is also visible that for correct retrieval it is not necessary that the skeleton exactly follows the medial axes of limbs and body. The method tolerates additional and missing loops and branches in the sense that they reflect a consistent behaviour of the keypoint detector.

### C. Visualisation of the Feature Space

Figures 4 and 5 visualise the feature space for 256 and 4096 poses sampled from a set of videos of kitchen scenes. The course of the Hilbert curve is indicated by a thin red line.

The figures show a clear grouping of homogeneous poses in compact 2D regions. The upper right corner of Fig. 4 shows a knee-shot of the 'Psi'-pose (known from some calibration methods) with slightly higher arms. The cluster below shows a full body side view of a person leaning over a cabinet. This pose is problematic for many model-based methods because it is atypical for gaming applications and because of self-occlusion. Below this cluster, there is a group of Psi-poses with horizontal arm posture. The lower right corner shows a cluster of similar noisy detections. Many more poses can be discovered in the figure. Fig. 5 is of course too small to recognise any concrete pose. However, the clustering is also visible for the 4096 poses. The right figure shows the overall structure of the minimum spanning tree. The tree consists of small subtrees plotted on the left side of the graph. The subtrees are connected by the path of maximum length in the graph. The nodes along the path of maximum length appear as good prototypes for distinctive poses. It must be taken into account that the size of the subtrees is related to the frequency of a pose in a certain application.

In summary, the method is able to produce a very clear and intuitive 2d-representation of the originally 324-dimensional feature space.

## V. Conclusion

The recognition of humans is crucial to a number of mechatronics applications. This paper presented a generic appearance

based solution for automatic detection and pose estimation. The method has solved the problem of overspecialisation to a specific scene configuration that can be seen in many model-based approaches. It does not need training and is real-time capable. Our experiments have shown that the proposed spatio-temporal segmentation is able to detect persons and track them over time. The method adapts automatically to the number of people and tolerates noise and occlusion to a large degree. The proposed pose descriptor allows for a visually convincing retrieval of similar poses. This is for example important to connect the system to a data base with further information. This paper also presented a visualisation method based on a clustering and a Hilbert curve that maps the 324-dimensional space of pose descriptors to a compact 2d-representation while preserving pose similarities.

## REFERENCES


[1] G. Schweitzer, "Mechatronics for the design of human-oriented machines," *Mechatronics, IEEE/ASME Transactions on*, vol. 1, no. 2, pp. 120–126, 1996.

[2] Z. Zhang, W. Liu, V. Metsis, and V. Athitsos, "A viewpoint-independent statistical method for fall detection," in *Pattern Recognition (ICPR), 2012 21st International Conference on*, 2012, pp. 3626–3630.

[3] C.-F. Lai, S.-Y. Chang, H.-C. Chao, and Y.-M. Huang, "Detection of cognitive injured body region using multiple triaxial accelerometers for elderly falling," *Sensors Journal, IEEE*, vol. 11, no. 3, pp. 763–770, 2011.

[4] T. Shany, S. Redmond, M. Narayanan, and N. Lovell, "Sensors-based wearable systems for monitoring of human movement and falls," *Sensors Journal, IEEE*, vol. 12, no. 3, pp. 658–670, 2012.

[5] S. Chiaverini, B. Siciliano, and L. Villani, "A survey of robot interaction control schemes with experimental comparison," *Mechatronics, IEEE/ASME Transactions on*, vol. 4, no. 3, pp. 273–285, 1999.

[6] R. Gassert, R. Moser, E. Burdet, and H. Bleuler, "MRI/fMRI-compatible robotic system with force feedback for interaction with human motion," *Mechatronics, IEEE/ASME Transactions on*, vol. 11, no. 2, pp. 216–224, 2006.

[7] E.-H. Kim, K.-H. Hyun, S.-H. Kim, and Y.-K. Kwak, "Improved emotion recognition with a novel speaker-independent feature," *Mechatronics, IEEE/ASME Transactions on*, vol. 14, no. 3, pp. 317–325, 2009.

[8] T. Sato, T. Harada, and T. Mori, "Environment-type robot system "roboticroom" featured by behavior media, behavior contents, and behavior adaptation," *Mechatronics, IEEE/ASME Transactions on*, vol. 9, no. 3, pp. 529–534, 2004.

[9] K. D. Nguyen, I.-M. Chen, Z. Luo, S. H. Yeo, and H.-L. Duh, "A wearable sensing system for tracking and monitoring of functional arm movement," *Mechatronics, IEEE/ASME Transactions on*, vol. 16, no. 2, pp. 213–220, 2011.

[10] S. M. Choi, J. chan Jeong, and J. Chang, "Dense 3d depth map with doe pattern," in *Ubiquitous Robots and Ambient Intelligence (URAI), 2012 9th International Conference on*, 2012, pp. 34–37.

[11] T. Hongyong and Y. Youling, "Finger tracking and gesture recognition with kinect," in *Computer and Information Technology (CIT), 2012 IEEE 12th International Conference on*, 2012, pp. 214–218.

[12] I. Oikonomidis, N. Kyriazis, and A. Argyros, "Efficient model-based 3d tracking of hand articulations using kinect," in *British Machine Vision Conference (BMVC)*, 2011.

[13] M. Draelos, N. Deshpande, and E. Grant, "The kinect up close: Adaptations for short-range imaging," in *Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2012, pp. 251–256.

[14] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 1297–1304.

[15] I. Giosan and S. Nedevschi, "Building Pedestrian Contour Hierarchies for Improving Detection in Traffic Scenes," in *Int. Conf. on Computer Vision and Graphics (ICCVG)*. Springer, 2008, pp. 154–163.

[16] G. Mori, X. Ren, A. A. Efros, and J. Malik, "Recovering Human Body Configurations: Combining Segmentation and Recognition," in *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004, pp. 326–333.

[17] J. Liu, P. Carr, R. T. Collins, and Y. Liu, "Tracking Sports Players with Context-Conditioned Motion Models," in *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[18] Y. Iwashita and M. Petrou, "Person identification from spatio-temporal volumes," in *IEEE International Conference on Image and Vision Computing New Zealand (IVCNZ)*, 2008.

[19] N.-G. Cho, A. Yuille, and S.-W. Lee, "Self-Occlusion Robust 3D Human Pose Tracking from Monocular Image Sequence," in *International Conference on Systems, Man, and Cybernetics*, 2012, pp. 254–257.

[20] C. Wu and H. Aghajan, "Real-time human pose estimation: A case study in algorithm design for smart camera networks," *Proceedings of the IEEE*, vol. 96, no. 10, pp. 1715–1732, 2008.

[21] Y. Wang and G. Qian, "Robust human pose recognition using unlabelled markers," in *Applications of Computer Vision, 2008. WACV 2008. IEEE Workshop on*, 2008, pp. 1–7.

[22] J. Gall, B. Rosenhahn, T. Brox, and H.-P. Seidel, "Optimization and Filtering for Human Motion Capture," *International Journal of Computer Vision*, vol. 887, pp. 75–92, 2010.

[23] K. Buys, C. Cagniart, A. Baksheev, T. D. Laet, J. D. Schutter, and C. Pantofaru, "An adaptable system for rgb-d based human body detection and pose estimation," *accepted for Journal of Visual Communication and Image Representation*, 2013.

[24] D. Moschini and A. Fusiello, "Tracking human motion with multiple cameras using an articulated model," in *Int. Conf. on Computer Vision/Computer Graphics CollaborationTechniques (MIRAGE)*. Berlin, Heidelberg: Springer, 2009, pp. 1–12.

[25] W. Areevijit and P. Kanongchaiyosy, "Reeb Graph Based Partial Shape Retrieval For Non-Rigid 3D Object," in *ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry (VRCAI)*, 2011, pp. 573–576.

[26] H. P. Jain and A. Subramanian, "Real-Time Upper-Body Human Pose Estimation Using a Depth Camera," in *Int. Conf. on Computer Vision / Computer Graphics Collaboration Techniques and Applications (MIRAGE)*. Springer, 2011, pp. 227–238.

[27] N. Kyriazis and A. Argyros, "Physically Plausible 3D Scene Tracking: The Single Actor Hypothesis," in *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[28] I. Theodorakopoulos, D. Kastaniotis, G. Economou, and S. Fotopoulos, "Pose-based human action recognition via sparse representation in dissimilarity space," *Journal of Visual Communication and Image Representation*, 2013.

[29] M. Raptis, D. Kirovski, and H. Hoppes, "Real-time classification of dance gestures from skeleton animation," in *ACM SIGGRAPH/Eurographics Symp. on Computer Animation*, 2011, pp. 976–990.

[30] F. Ofli, R. Chaudhry, G. Kurillo, R. Vidal, and R. Bajcsy, "Sequence of the most informative joints (smij): A new representation for human skeletal action recognition," in *Computer Vision and Pattern Recognition Workshop (CVPRW)*. IEEE Computer Society, 2012, pp. 8–13.

[31] G. Lu, M. Kudo, and J. Toyama, "Robust human pose estimation from corrupted images with partial occlusions and noise pollutions," in *Granular Computing (GrC), 2011 IEEE International Conference on*, 2011, pp. 433–438.

[32] S. Belongie, J. Malik, and J. Puzicha, "Shape Context: A new descriptor for shape matching and object recognition," in *Neural Information Processing Systems (NIPS)*, 2000, pp. 831–837.

[33] K. Konolige and P. Mihelich, "Kinect calibration: Technical," 2012, accessed July 15th, 2013. [Online]. Available: http://www.ros.org/wiki/kinect_calibration/technical

[34] K. Khoshelham and S. O. Elberink, "Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications," *Sensors*, vol. 12, pp. 1437–1454, 2012.

[35] M. Stommel, "Zur Erkennung verformbarer Objekte anhand ihrer Teile," Ph.D. dissertation, Fachbereich Informatik und Elektrotechnik, Universität Siegen, 2010.

[36] D. G. Lowe, "Object Recognition from Local Scale-Invariant Features," in *Int. Conf. on Computer Vision (ICCV)*, 1999, pp. 1150–1157.

[37] D. Weikersdorfer, D. Gossow, and M. Beetz, "Depth-Adaptive Superpixels," in *International Conference on Pattern Recognition (ICPR)*, 2012.

[38] D. Hilbert, "Über die stetige Abbildung einer Linie auf ein Flächenstück," *Mathematische Annalen*, vol. 38, pp. 459–460, 1891.

[39] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient Graph-Based Image Segmentation," *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167–181, 2004.

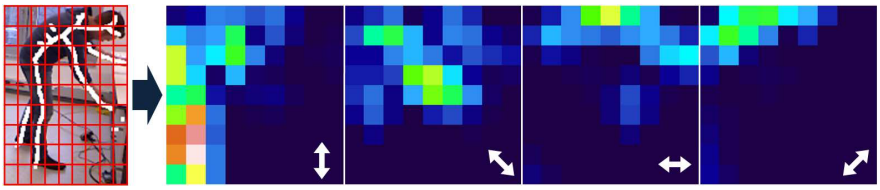


Fig. 1: The bounding box of a skeleton is subdivided by a regular $9 \times 9$-grid. The orientation of the branches of the skeleton is computed and written to four $9 \times 9$-histograms. Each $9 \times 9$-histogram represents one direction (indicated by an arrow). As a result, the first histogram shows only the legs, the second histogram the arm, the third histogram the shoulder and head, and the fourth histogram the back.

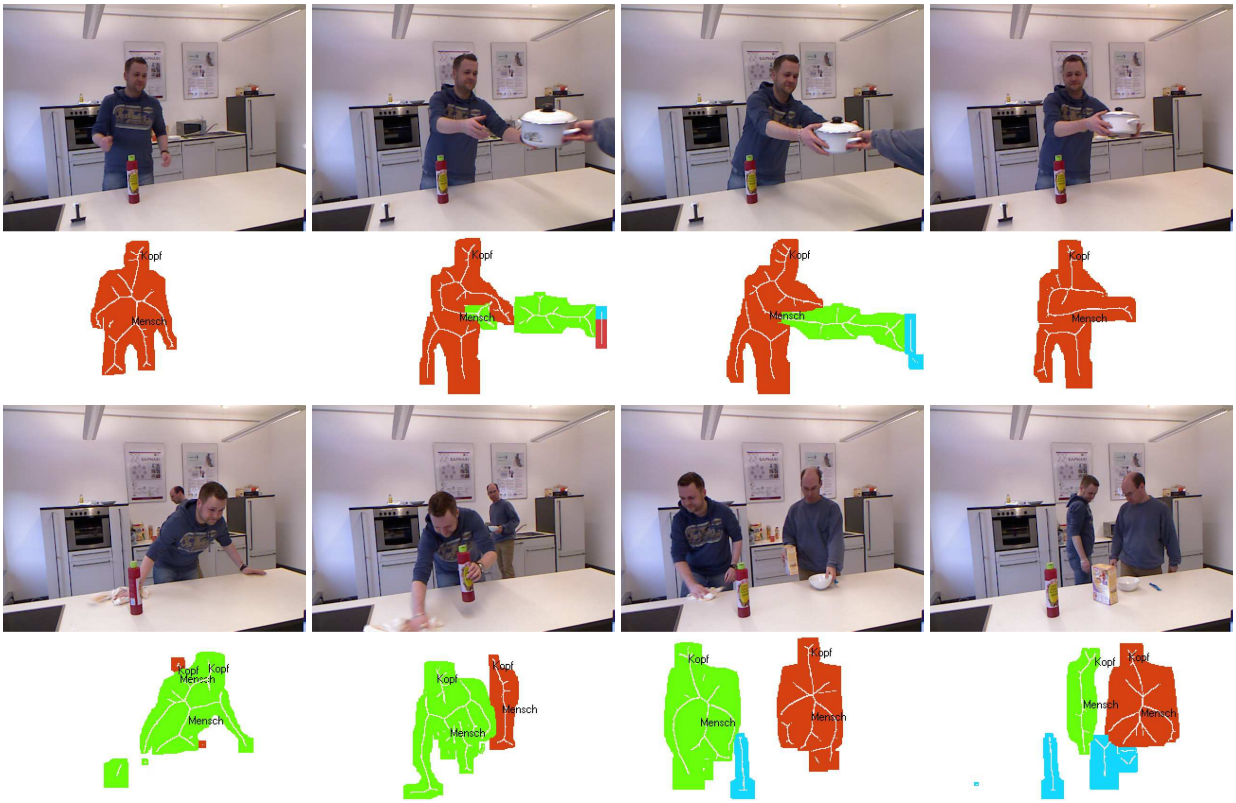



Fig. 2: Spatio-temporal segmentation of two kitchen scenes.

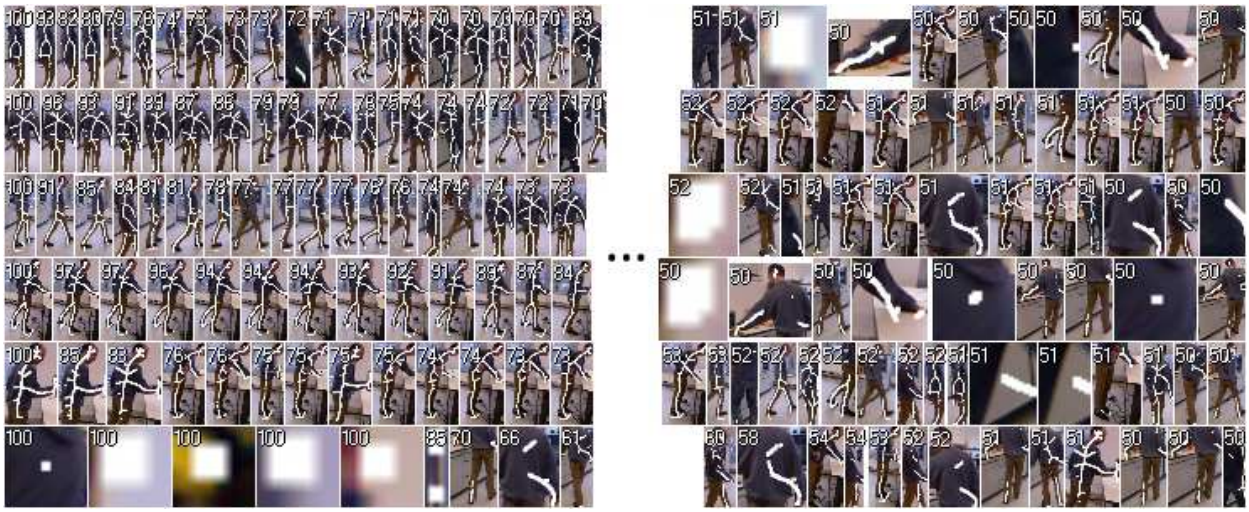
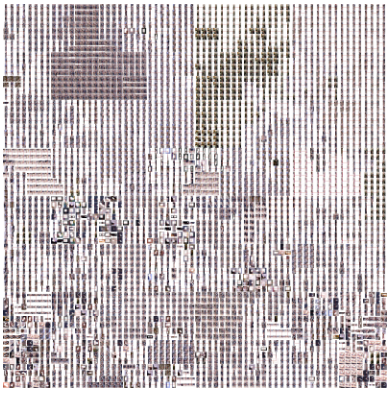
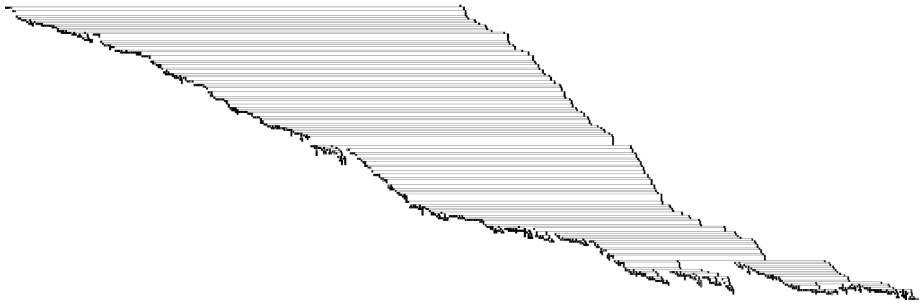


Fig. 3: Skeletons ordered by correlation coefficient.

Fig. 4: Clustering of 256 skeletons by a minimum spanning tree algorithm. The red line indicates a linearisation of the tree by a preorder traversal. The line is drawn as a Hilbert curve for a compact survey of the whole data.



a) Overview of 4096 skeletons



b) Minimum spanning tree

Fig. 5: Clustering of 4096 skeletons shown as a mapping of a minimum spanning tree to a Hilbert curve (a) and as the tree itself. Due to the number of skeletons and the small size, only the global properties of the clustering are visible.

# Fractal Approximate Nearest Neighbour Search in Log-Log Time

Martin Stommel
mstommel@tzi.de
Stefan Edelkamp
edelkamp@tzi.de
Thiemo Wiedemeyer
wiedemeyer@informatik.uni-bremen.de
Michael Beetz
beetz@cs.uni-bremen.de

Institute for Artificial Intelligence
University of Bremen
Am Fallturm 1
28359 Bremen
Germany

**Abstract**

The power of fractal computation has been mainly exploited for image compression and halftoning. Here, we consider it for finding a fast approximate solution for the fundamental problem of nearest neighbour computation in the image plane. Traditional solutions use Delaunay triangulations or hierarchies (for the case of optimal solutions) or kd-trees for approximate ones. In contrast, we use a space-filling Hilbert curve which allows us to reduce the problem from 2D to 1D. The Hilbert curve has already been used to optimise high-dimensional nearest neighbour queries in the context of data base systems. In this paper, we propose a simplified solution that fits better to the computer vision context. We show that our algorithms solve two particular nearest neighbor problems efficiently. We provide practical results on the accuracy of the method and show that it is significantly faster than a kd-tree.

## 1 Introduction

Nearest neighbour searches in the image plane are among the most frequent problems in a variety of computer vision and image processing tasks. They can be used to replace missing values in image filtering, or to group close objects in image segmentation, or to access neighbouring points of interest in feature extraction. In image filtering, the filter result is often only computed for a sparse set of key points. This is either the case if the processing of the whole image would take too much time or if only a small set of pixels is suitable for processing (e.g. because of the aperture problem). The missing filter output must then be interpolated between the nearest keypoints. In image segmentation, nearest neighbour searches allow for a (possibly recursive) combination of close points to more complex objects, which leads to a fine-to-coarse decomposition of the image [28, 30]. In particular for object recognition, the concept of spatial proximity seems to be of fundamental importance, with a clear effect on the image statistics [29]. Figure 1 illustrates examples of common nearest neighbour problems.


a) Interpolation between sparse keypoints b) Line feature extraction

Figure 1: Nearest neighbour problems in the image plane. a) Mesh of SIFT keypoints used to detect the movements of cows in a stable. Since there are only measurements for the nodes of the mesh, pixels within the cells must be assigned to the nearest node. As a second nearest neighbour problem, the cells of the mesh can be constructed by finding two neighbouring nodes for every node. Missing measurements can then be interpolated between the corners of a cell. b) Estimation of the orientation of the edge through point B. The edge is represented by keypoints A–D. The orientation is computed by constructing the line segment of B to the approximate nearest neighbour D instead of the exact nearest neighbour A.

Traditional solutions to such nearest neighbour problems are a pixel-wise search within adaptive or fixed-size image windows, or the use of Delaunay triangulations and kd-trees. Search windows are unattractive because many irrelevant pixels must be visited. Fast results can only be achieved by using small windows or sub-sampling, often with a loss of accuracy. And although fast approximate results would often be preferred over accurate but slow computations, fixed window sizes may simply not suit the problem well. Balanced trees are more attractive because of their logarithmic run-time for a nearest neighbour search. The construction of the tree however introduces additional overhead, in the case of video sequences even repeatedly.

In this paper, we propose a fractal approach to achieve an approximate solution. The basic idea is to map the image plane to a one-dimensional space filling curve, the Hilbert curve, and perform the nearest neighbour search there. The Hilbert curve is known to keep the original 2D-relationship to a certain degree. As a result, an approximate nearest neighbour can be found by searching the nearest neighbour (in other words the successor or predecessor) in a linear list. This can be done in one step or in log-log time depending on the implementation. Since the mapping is the same for every image (assuming a fixed size), there is no repeated overhead for video sequences. The theoretical and experimental results in this paper show that the proposed method is quite powerful in a computer vision context. Surprisingly, the use of space filling curves is largely unknown in this domain. Rare counterexamples include the application in halftone methods [26], image compression [22], and edge detection [19]. Aside from simplifications of the exact method [10], the contribution of this paper consists in the transfer of the method from database systems to computer vision.

## 2 Related Work and Problem Definition

The *nearest neighbour problem* is usually stated independently of the application as returning the point $p \in S, S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ that minimises the Euclidean distance $||p - q||_2$

to a query point $q = (x,y)$. The simple solution of a linear scan comprises a comparison of $q$ to all elements of $S$, which is too time-consuming for most applications, especially those with real-time requirements.

In image processing and computer vision the search space can often be reduced if the nearest neighbour is known to lie on an edge or another detectable feature. Contour tracers [15, 25] can then lead to fast results. The drawback of such methods is that many special cases must be considered and contours may be broken because of noise. A full search for the continuation of an edge after a gap is slow and complicated. A noise-tolerant but still slow solution could be to use active contours [18], where the shape of a contour is optimised with respect to energy functionals describing the adaptedness to the image and the shape complexity. Contour tracing does not benefit from information of neighbour searches of adjacent query points, which makes the method slow for dense sets of query points.

If the nearest neighbour $p \in S$ must be found for every coordinate $q \in I$ of an image $I = \{(0,0),(0,1),(0,2),\ldots,(W,H)\}$ of width $W$ and height $H$, we obtain the *all nearest neighbours problem*. This problem occurs frequently in modern saliency based approaches, where only robustly detectable image regions are processed (exemplarily [20]). Filter results for subsequent steps (e.g. object movements) are only computed for keypoints placed in those regions. In order to assign the filter results to single pixels, it might be necessary to find the nearest keypoint. The problem can be solved by a distance transform [27]. The method exploits that images consist of connected discrete pixels. The result can then be computed in two passes through the image [14, 27], and hence in linear time. The coordinates of the nearest neighbours can be recorded in the same process. The method does however not generalise to the *k-nearest neighbour problem*, where $k$ neighbours $p_1,\ldots,p_k$ of a query point $q$ must be found with distance $||p_i - q||_2 \leq ||p_j - q||_2, i \leq k, j > k$.

Graph-based methods [4, 8, 12] are also often used for nearest neighbour searches in computer vision, although these methods do not benefit from any image specific structuring of the data (e.g. contours). Approximate solutions can be computed by recursively subdividing the search space into smaller cells by using a kd-tree [3, 4, 21]. The approximately nearest neighbour in a bounded uniform distribution is found in $O(\lg n)$ by searching a list of cells ordered by proximity to the query point. Exact solutions can be computed by using Delaunay hierarchies [8, 12]. One recent result to the nearest neighbor problem are full Delaunay hierarchies [6]. The main difference between an ordinary Delaunay triangulation is to additionally record all edges that have been used during the incremental construction. Using a randomized incremental construction of the Delaunay triangulation, the approach uses $O(n \lg n)$ expected time for building the data structure with an expected number of $O(n)$ edges. The expected number of nodes traversed for finding the nearest neighbor by a simple greedy algorithm yield an expected query time of $O(\lg n)$. Different to many other approaches, once the graph is constructed, an additional search structure is not needed.

To simplify high-dimensional nearest neighbour queries, locality sensitive hashing has been proposed [13, 23, 31]. The idea is to map the input data on the unit hypercube in a way that preserves the original distance relationships to a certain degree, even if the Hamming distance is used to compare the mapped vectors. The methods have been shown to allow for fast vector comparisons, predominantly for SIFT-related data sets. For low-dimensional data, such methods are too coarse: The cited approaches would divide the 2D-plane into at most four areas.

Space filling curves have originally been introduced to demonstrate the existance of a point-to-point mapping of the real valued unit square to a continuous curve [24]. The Hilbert curve [17], a variant of the original Peano curve [24], is defined as the recursive subdivi-

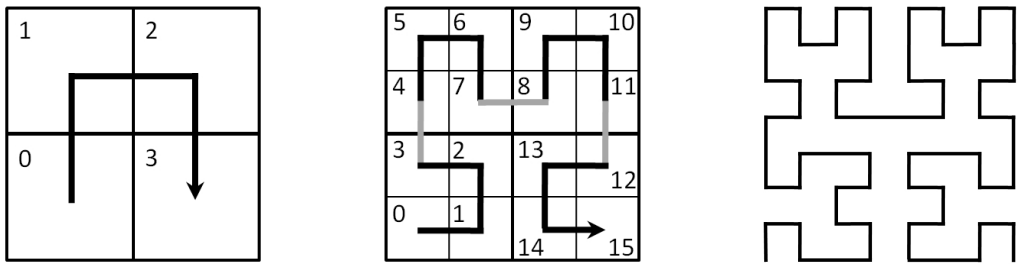



Figure 2: Hilbert curve for a $2 \times 2$ (on the left), $4 \times 4$ (middle) and $8 \times 8$ image (on the right). Pixels are counted along the curve. The second curve (and higher resolutions) can be computed (recursively) from the first one by replacing each corner by a rotated and reflected version of the basic U-shaped pattern.

sion of a square into four sub-squares with a U-shaped ordering imposed on the sub-squares (cf. Fig. 2). In the case of discrete images, the recursion stops after a certain number of iterations $R$. This results in a subdivision of the unit square into $W = H = 2^R$ rows and columns. There are several algorithms that can be used to compute the complete mapping in linear time (in terms of the number of pixels) [9, 16, 17]. For a specific image coordinate, the corresponding index of a point on the Hilbert curve (the *Hilbert index*) requires the recursive descent over $R$ levels and a number of bit operations on each level, hence the complexity is $O(\lg W)$. The mapping preserves 2D-distance relationships in the sense that neighbouring points on the Hilbert curve are neighbouring in the image, too. Larger distances are preserved by the quad-tree-like recursive subdivision of the image. This has motivated to use the Hilbert curve to answer exact [10, 11] and approximate [32] nearest neighbour queries. To compensate for errors in the mapping, the Hilbert indices of adacent image coordinates must be found. This can be solved by reducing the problem to the easier Peano curve [10] or by evaluating multiple mappings [32]. Most methods consider high-dimensional input data. The generalisation of the Hilbert curve to more than two dimensions is however not as straightforward as it seems. Alber and Niedermeier [1] show that already in 3D there are 1536 structurally different curves with Hilbert property, which might be a potential for further optimisation. Aside from nearest neighbour computations, the Hilbert curve is also used to visualise high-dimensional data sets [2].

We observed that in many computer vision applications, a high accuracy of the nearest neighbour search is not needed. Our method therefore differs from the above mentioned ones in that we do not attempt to correct the inaccuracies in the mapping of 2D to 1D-distances. As a result, our method is able to find close points at low computational cost.

# 3 Proposed Method

Our method answers two types of nearest neighbour problems in three steps each: At first, the mapping between 2D and 1D-coordinates must be computed. For the all nearest neighbour problem, the set of keypoints $S$ must be written as an array. The nearest neighbour assignment can then be done in two passes throught the array. For the k-nearest neighbour problem, the set of keypoints $S$ must be stored in a priority queue. The neighbours of a query point $q$ can then be found by using the successor function of the queue.

A Hilbert curve of recursive depth $R$ subdivides the unit square into $2^R$ rows and columns. The size of the sub-squares is $2^{-R} \times 2^{-R}$. We map an image with unit square sized pixels (i.e. integer coordinates) to the unit cube by applying the scale factor $2^{-R}$. To make sure that

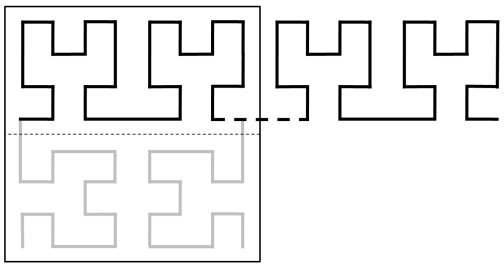


Figure 3: For elongated rectangular images it is possible to use only the upper half of the Hilbert curve or to stitch together multiple Hilbert curves.

an image with $W$ columns and $H$ rows fits into the unit cube, the recursive depth must be $R = \lceil \log_2 \max(W,H) \rceil$. We obtain $C$ Hilbert indices range from 0 to $2^{2R}-1$. More recursions are redundant. With this scaling, we have a correspondence between the Hilbert indices of the 1D-curve, the sub-squares of the unit-cube, and the pixels of an image. Rectangular images and images whose side length is not a power of two do not cover the unit square completely. An image of size $640 \times 480$ would be mapped to a $1024 \times 1024$ grid in the unit cube. In this case, only 30% of the Hilbert indices correspond to pixels in the image. Since the algorithm for the all nearest neighbour computation requires a loop over all Hilbert indices, it might be advantageous to cut the Hilbert curve to the the upper half of the unit square, or to string together multiple Hilbert curves (Figure 3) in order to reduce the overhead. In the rest of the paper, however, this optimisation is not used. The result of the first step is a mapping $M(x,y) : \mathbb{N}^2 \to \mathbb{N}$ of image coordinates $(x,y)$ to Hilbert indices, as well as the partial inverse mapping $M^{-1}(i) : \mathbb{N} \to \mathbb{N}^2$ of a Hilbert coordinate to the image coordinate $(x,y)$. The mapping must be computed only once for a fixed image size. For the whole image this can be done in linear time with respect to the number of pixels. We use arrays to store the mapping.

The mapping is used to solve two types of problems. In the all nearest neighbour computation we find the approximately closest keypoint $p \in S$ for all pixels $q \in I$.

**Theorem 1.** *The all approximately nearest neighbour problem can be solved in precomputation time $O(C+n)$ and query time $O(1)$. It uses $O(C)$ space.*

*Proof.* Constant query time is achieved by computing a lookup table $T$ of size $O(C)$ with the results. The indices of the lookup table are the Hilbert indices. At first, we mark the $n$ keypoints $S$ in the table, which is $O(n)$. In one forward pass, we compute for all indices $i$ the distance $i-j$ to the nearest keypoint with lower Hilbert index $j$. To this end, we need to check if index $i$ of the lookup table corresponds to a keypoint, i.e. if $M^{-1}(i) \in S$. In that case, we set the distance to zero. We also set it to zero if there is no preceding keypoint. Otherwise we increment the distance of the preceding table entry by one. The check and the distance increment can be done in constant time, so the general complexity of the forward pass is $O(C)$. In a backward pass, we compute the distance to the nearest keypoint with higher Hilbert index. Then we assign each index of the Hilbert curve to the nearest keypoint index from the forward and backward pass. The complexity of the backward pass is again $O(C)$. The total complexity is therefore $O(n+C)$. □

The second problem is the approximately k-nearest neighbour problem.

**Theorem 2.** *The approximately k-nearest neighbours can be found in precomputation time $O(n \lg\lg C)$ and query time $O(\lg\lg n + k)$. The space requirement is $O(C \lg\lg C)$.*

*Proof.* We build a priority queue $Q$ where all $n$ keypoints $S$ are ordered by their Hilbert index $M(p), p \in S$. Using the precomputed arrays for $M$, the Hilbert index of a key point can be found in constant time. Inserting an element in a priority queue takes constant time plus an overhead of $O(\lg\lg C)$ for locating the right position in the queue using the successor function [7]. The complexity for inserting $n$ elements is therefore $O(n\lg\lg C)$. By linking all adjacent elements of the queue (in $O(n)$), we can find the successor and predecessor of an existing element in constant time. The priority queue needs to be computed only once for a given set of keypoints. The precomputation time is therefore $O(n\lg\lg C)$.

For a query point $q$, the approximately nearest neighbour can be the successor of $M(q)$ in $Q$ or its direct predecessor. The successor $M(p_s)$ can be found in $O(\lg\lg C)$. Its direct predecessor $M(p_p)$ is found in constant time using direct links. The comparison of the Hilbert indices ($|M(q)-M(p_s)|$ and $|M(q)-M(p_p)|$) and the nomination of an approximately nearest neighbour is also constant time. The remaining $k-1$ neighbours can be found by reading out the directly linked $(k-1)/2$ predecessors and $(k-1)/2$ successors, which are single step operations. The $k$ nearest neighbours can therefore be found in time $O(\lg\lg C+k)$ once the queue is constructed. The space requirement is $O(C\lg\lg C)$ [7]. □

# 4 Experimental Results

We first measured how well 1D-neighbourhoods on the Hilbert curve correspond to 2D-neighbourhoods in an image and vice versa. To this end, we created a $256\times256$ image together with its Hilbert curve. We sampled a first list of 50 000 pairs of points on the Hilbert curve with a maximum difference of 10 in their Hilbert indices. For every pair we then computed the corresponding 2D-Euclidean distance in the image. As Fig. 4 (left) shows, close points in 1D are close in 2D as well. On average, the 2D-distance is the square root of the 1D distance. Median and mean are close together, so there are no serious outliers to the rule. We then sampled a second list of 50 000 pairs of image points with a Euclidean distance less than 10 and measured the corresponding 1D-distance of their Hilbert indices. Figure 4 (right) shows that close points in 2D are not necessarily close in 1D. The 1D-distances are not normally distributed and the mean deviates from the median because of far outliers (the maximum 1D-distance was about 54 000 for every 2D-distance). The relatively unaffected median of only up to 200 shows that the number of outliers is however below the 50% breaking point of the median. For our nearest neighbour search this means that the approximated result will be close to the query point in 2D if it is close on the Hilbert curve in 1D. The distance in 1D will be small because it is minimised by our algorithm. On the other hand, our method will sometimes miss the exact nearest neighbour because small 2D-distances do not necessarily translate to small 1D-distances. But because of the good 1D-to-2D-correspondence, the result will still be good.

In order to measure the accuracy of the method, we solved the all nearest neighbour problem for a small image and varying numbers of keypoints first using the proposed approximation and secondly using an exact method. It turns out that our approximation yields the same nearest neighbour as the exact method in about 50% of the queries (Fig. 5). Even in the case where the proposed method produces different results, the approximated neighbour is close (as shown above). As Fig. 6 shows, the proposed method produces a compact partitioning of the image plane. It is roughly comparable to the Voronoi diagram of the exact solution.

We benchmarked the computation time of our method for the all nearest neighbour prob-

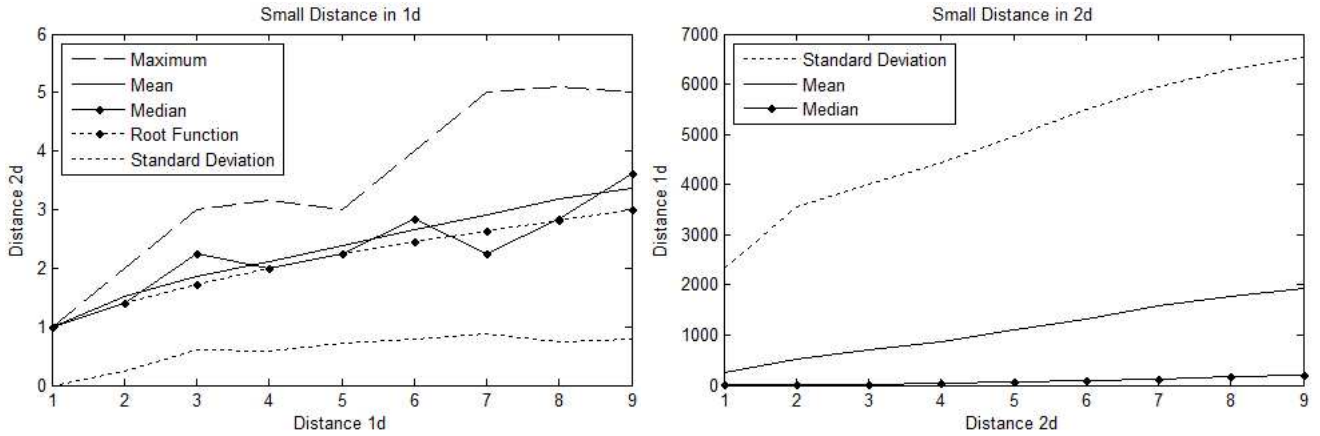



Figure 4: Relationship between corresponding 1D and 2D-distances for randomly selected pairs of close points (distance < 10) once in 1D (left diagram) and then in 2D (on the right). The left diagram shows that the approximated nearest neighbour is close to the query point in 2D if it is close in 1D (which it is expectedly).

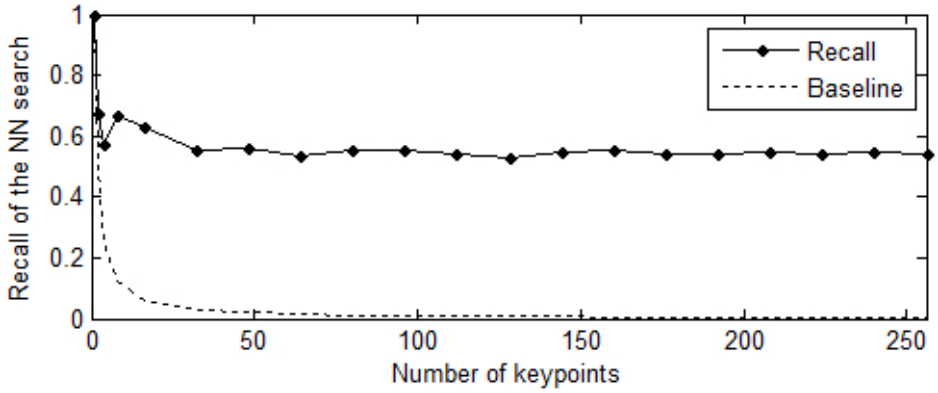



Figure 5: Probability of finding the exact nearest neighbour by applying the proposed *approximative* method to a $256 \times 256$ image. For a cursory reader we remark that it is quite improbable to find the nearest neighbour by chance (baseline curve).

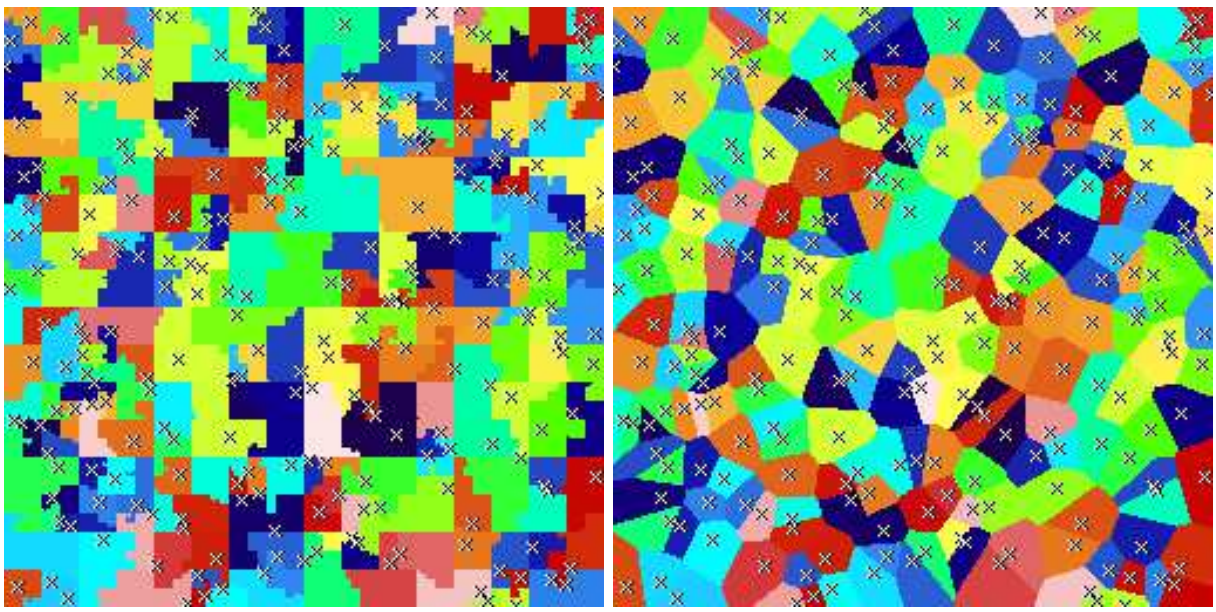


Figure 6: All nearest neighbour assingment using the proposed approximative method (left image) and an exact search (right image) for 240 keypoints (marked by crosses). The resulting cells are coloured randomly but consistent over both diagrams.

Figure 7: Using a Hilbert curve to display the clustering of a high dimensional data set: The data set consists of images of skeletons of persons in Kinect images. Each skeleton is described by a 3D-histogram over the x and y image coordinate and the angle of a branch of the skeleton. The angle was found using the proposed nearest neighbour method to vectorise the branches (similar to the method outlined in Fig. 1 a). The skeletons were clustered by computing the minimum spanning tree in a fully connected graph weighted by the cross correlation between the histograms. A preorder traversal of the tree yields a 1D-visualisation of the data. The Hilbert curve was used to obtain a more compact and space effective representation.

lem against a kd-tree [5] which provides an exact solution in $O(\lg n)$ query time. Our input data is an image with $1920 \times 1200$ query points and 4753 keypoints provided by a feature extractor. The software was run on an Intel Core-i7 2670QM processor in a single-threaded implementation. We measured a precomputation time of 1.9ms to build the kd-tree, and 536.7ms to find all nearest neighbours (averaged over 100 runs each). For the proposed method, we measured 195ms for initialisation and 29.8ms to find all nearest neighbours

(again averaged over 100 runs). This is a speed-up factor of 18. For a smaller image of $1280 \times 800$ pixels and 4694 keypoints, building the kd-tree took again 1.9ms, whereas the nearest neighbour computation took 236.3ms. Since the recursive depth $R$ is equal for both images, we achieved similar results for the proposed method: 190ms to initialise the method, and 26.2ms for the all nearest neighbour search. The speed-up factor is 9 here.

As a last result, we present an example of using the Hilbert curve to visualise a high-dimensional data set (Fig. 7). The curve was used both to determine the global arrangement of the figure, and to compute one of the underlying features.

# 5 Conclusion

Our method uses the Hilbert curve to compute fast approximate solutions of the all nearest neighbour problem and the k-nearest neighbour problem in the image plane. Our method has a precomputation time of $O(n)$ for adapting to a fixed image size. Depending on the problem, an additional precomputation time of $O(C+n)$ (all nearest neighbours) or $O(n \lg\lg C)$ (k-nearest neighbours) is needed to adapt to a certain set of key points. The query time is then $O(1)$ or $O(\lg\lg C + k)$, respectively. This is an advantage over a balanced tree (with runtime $\lg n$) if $n > \lg C$ (the latter being a small number for common image formats). Our experiments show that our method yields a compact and visually meaningful approximation of the Voronoi diagram in the image plane, which is sufficient for many applications. For 50% of the queries, our method yields the exact result. In a practical example of finding all nearest neighbours of a set of keypoints, our method was 9–18 times faster than a kd-tree.

# Acknowledgement

This work has been funded by the German Science Foundation (DFG) as part of the project MeMoMan2 and by the European Community within the FP7 ICT-287513 SAPHARI project.

# References

[1] J. Alber and R. Niedermeier. On Multidimensional Curves with Hilbert Property. *Theory of Computing Systems*, 33(4):295–312, 2000.

[2] S. Anders. Visualisation of genomic data with the Hilbert curve. *Bioinformatics*, 25: 1231–1235, 2009.

[3] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, January 2008. ISSN 0001-0782.

[4] S. Arya and H.-Y. A. Fu. Expected-case complexity of approximate nearest neighbor searching. In *Symposium on Discrete Algorithms*, pages 379–388, 2000.

[5] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, September 1975. ISSN 0001-0782.

[6] M. Birn, M. Holtgrewe, P. Sanders, and J. Singler. Simple and Fast Nearest Neighbor Search. In *ALENEX*, pages 43–54, 2010.

[7] P. Emde Boas, R. Kaas, and R. Zijlstra. Design and implementation of an efficient priority queue. *Mathematical systems theory*, 10(1):99–127, 1976.

[8] J.-D. Boissonnat and M. Teillaud. The hierarchical representation of objects: The Delaunay tree. In *ACM Symposium on Computational Geometry*, pages 260–268, 1986.

[9] A.R. Butz. Alternative algorithm for hilbert's space-filling curve. *Computers, IEEE Transactions on*, C-20(4):424–426, 1971.

[10] H.-L. Chen and Y.-I. Chang. Neighbor-finding based on space-filling curves. *Inf. Syst.*, 30(3):205–226, May 2005. ISSN 0306-4379.

[11] H.-L. Chen and Y.-I. Chang. All-nearest-neighbors finding based on the Hilbert curve. *Expert Systems with Applications*, 38(6):7462 – 7475, 2011. ISSN 0957-4174.

[12] O. Devillers. The Delaunay Hierarchy. *Internat. J. Found. Comput. Sci.*, 13:163–180, 2002.

[13] S. Edelkamp and M. Stommel. The Bitvector Machine: A Fast and Robust Machine Learning Algorithm for Non-linear Problems. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, pages 175–190. Springer, 2012.

[14] R. Fabbri, L. Da F. Costa, J. C. Torelli, and O. M. Bruno. 2d euclidean distance transform algorithms: A comparative survey. *ACM Comput. Surv.*, 40(1):2:1–2:44, February 2008. ISSN 0360-0300.

[15] S. Gul and M. F. Khan. Automatic Extraction of Contour Lines from Topographic Maps. In *Digital Image Computing: Techniques and Applications (DICTA), 2010 International Conference on*, pages 593–598, 2010.

[16] C. H. Hamilton and A. Rau-Chaplin. Compact Hilbert indices: Space-filling curves for domains with unequal side lengths. *Information Processing Letters*, 105(5):155–163, 2007.

[17] D. Hilbert. Über die stetige Abbildung einer Linie auf ein Flächenstück. *Mathematische Annalen*, 38:459–460, 1891.

[18] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[19] C.-H. Lamarque and F. Robert. Image analysis using space-filling curves and 1d wavelet bases. *Pattern Recognition*, 29(8):1309 – 1322, 1996.

[20] D. G. Lowe. Object Recognition from Local Scale-Invariant Features. In *International Converence on Computer Vision (ICCV)*, pages 1150–1157, 1999.

[21] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *In VISAPP International Conference on Computer Vision Theory and Applications*, pages 331–340, 2009.

[22] T. Ouni, A. Lassoued, and M. Abid. Gradient-based Space Filling Curves: Application to lossless image compression. In *IEEE International Conference on Computer Applications and Industrial Electronics (ICCAIE)*, pages 437–442, 2011.

[23] L. Paulevé, H. Jégou, and L. Amsaleg. Locality sensitive hashing: A comparison of hash function types and querying mechanisms. *Pattern Recogn. Lett.*, 31(11):1348–1358, 2010. ISSN 0167-8655.

[24] G. Peano. Sur une courbe, qui remplit toute une aire plane. *Mathematische Annalen*, 36(1):157–160, 1890.

[25] R. Pradhan, S. Kumar, R. Agarwal, M. P. Pradhan, and M. K. Ghose. Contour Line Tracing Algorithm for Digital Topographic Maps. *International Journal of Image Processing (IJIP)*, 4(2):156–163, 2010.

[26] T. Riemersma. A balanced dithering technique. *C/C++ Users J.*, 16(12):51–58, December 1998. ISSN 1075-2838.

[27] A. Rosenfeld and J. L. Pfalz. Distance functions on digital pictures. *Pattern Recognition*, 1:33–61, 1968.

[28] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-Based Image Retrieval at the End of the Early Years. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(12):1349–1380, 2000.

[29] M. Stommel and K.-D. Kuhnert. Part Aggregation in a Compositional Model Based on the Evaluation of Feature Cooccurrence Statistics. In *International Conference on Image and Vision Computing New Zealand (IVCNZ)*. IEEE, 2008.

[30] M. Stommel and K.-D. Kuhnert. Visual Alphabets on Different Levels of Abstraction for the Recognition of Deformable Objects. In *Joint IAPR International Workshop on Structural, Syntactic and Statistical Pattern Recognition (S+SSPR)*, LNCS 6218, pages 213–222. Springer, 2010.

[31] C. Strecha, A. M. Bronstein, M. M. Bronstein, and P. Fua. LDAHash: Improved Matching with Smaller Descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(1):66–78, 2012.

[32] H. Xu. An Approximate Nearest Neighbor Query Algorithm Based on Hilbert Curve. In *Internet Computing Information Services (ICICIS), 2011 International Conference on*, pages 514–517, 2011.

# References

[1] M. Beetz, N. Blodow, F. Balint-Benczedi, Z.-C. Marton, D. Nyga, F. Seidel and C. Kerl, “RoboSherlock: Unstructured Information Processing for Robot Perception,” International Journal of Robotics Research, submitted for review.

[2] M. Stommel, S. Edelkamp, T. Wiedemeyer and M. Beetz, “Fractal Approximate Nearest Neighbour Search in Log-Log Time,” British Machine Vision Conference (BMVC), Bristol, 2013.

[3] M. Stommel, M. Beetz and W. L. Xu, “Inpainting of Missing Values in the Kinect Sensor’s Depth Maps Based on Background Estimates,” Sensors, submitted for review.

[4] M. Stommel, M. Beetz and W. L. Xu, “Model-Free Detection, Encoding, Retrieval and Visualisation of Human Poses from Kinect Data,” IEEE/ASMA Transactions on Mechatronics, submitted for review.

[5] K. Buys, C. Cagniart, A. Baksheev, T. D. Laet, J. D. Schutter and C. Pantofaru, “An Adaptable System for RGB-D Based Humand Body Detection and Pose Estimation,” Journal of Visual Communication and Image Representation, 2013.