Institute of Parallel and Distributed Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Studienarbeit Nr. 2443

Instance-Based Learning of Affordances

Viktor Zielke

Course of Study:	Informatik
Examiner:	Prof. Dr. rer. nat. Marc Toussaint
Supervisor:	M. Sc. Stefan Otte
Commenced:	November 6, 2013
Completed:	May 8, 2014
CR-Classification:	I.2.10, I.4.8, I.5.2

Abstract

The discovery of possible interactions with objects is a vital part of an exploration task for robots. An important subset of these possible interactions are affordances. Affordances describe what a specific object can afford to specific agent, based on the capabilities of the agent and the properties of the object in relation to the agent. For example, a chair affords a human to be sat-upon, if the sitting area of the chair is approximately knee-high. In this work, an instance-based learning approach is made to discover these affordances solely through different visual representations of point cloud data of an object. The point clouds are acquired with a Microsoft Kinect sensor. Different representations are tested and evaluated against a set of point cloud data of various objects found in a living room environment.

Contents

List of Figures

List of Tables

List of Listings

List of Algorithms

1 Introduction

Imagine you are taking a walk in the woods and you get a pebble in your shoe. At first you ignore it hoping it won't trouble you much, but obviously it gets stuck right under your big toe, where you press down on it whenever you take a step. To remove the pebble from your shoe you are looking for some surface to sit down. Usually, you won't find any chairs in the woods and the last bench you saw was a couple of kilometers in the opposite walking direction. On the side of the trail you can see some sawed off tree trunks stacked, which seem to be stable enough and afford to be sat upon even though they are neither a bench or a chair. Your trousers might get dirty sitting on the trunks but at least you can get rid of that pesky pebble!

As humans we adapt to our environment and adjust our actions and view on the world according to our needs or the goals we want to achieve. Part of the view on the world is the identification of what the objects in the world can afford us. Chairs afford to be sat-upon, doors afford to be opened and closed, tables afford a surface to place other object upon, a TV affords entertainment and information, etc. If we go to a furniture store to buy a new office chair, we can identify different models of office chairs as chairs which afford to be sat upon even though they might differ in their color, shape or size. There seems to be some visual invariant in the recognition of a chair, which is not only the big sign in the furniture store which tells us that we are in the "Office chairs" section of the store.

For personal robots, this identification poses a big challenge. Not only do chairs have to be recognized as chairs but any subcategory of a chair has to be identified as a chair. This is not only true for chairs but for any object in a household environment. The learning of affordances of objects opposed to object recognition in a dynamic environment like a common household is more robust against changes in the environment. If new objects enter the environment it is more important for action planning of a robot to discover what this novel object affords the robot, instead of identifying the object as some specific object. Both, the affordance of an object and object recognition, are closely related. If a robot recognizes a novel object as some previously learned object with known affordances, the learning of the affordances of the newly recognized object is simply a mapping from the known affordances of the already learned objects to the affordances of the newly recognized object. Thus, a different approach can be made, which tries to learn an affordance not based on the recognized object, but based on visual features of the object.

In this work an approach is made to learn affordances of common household furniture based on visual features with different representations of the data and evaluating these different data

representations in experiments. Visual data is acquired with a Kinect camera. As a learning strategy, instance-based learning was chosen which to the knowledge of the author of this work was not found in other work used for learning affordances.

In the following, related work is presented in Section 2 and the background for this work is given in Section 3. Section 4 describes the proposed approaches and data representation. Evaluation of the proposed approaches and data representation is given in Section 5. Section 6 concludes this work with a discussion and possible extensions of this work.

2 Related Work

In this section we will take a look at some related work for this paper.

In the work of L. Stark and K. Bowyer "Achieving Generalized Object Recognition through Reasoning about Association of Function to Structure"[SB91], attempts were made to classify a 3D CAD model into the category chair and its sub-categories of a chair. Classification is done based on the boundary descriptions of a 3D model. The model is segmented into *functional elements* which can be single surfaces, groups of surfaces acting together to fulfill a certain function or three-dimensional portions of the model. *Functions* are here defined as e.g. *provide arm support*, *provides stable support*, *provides knee support*, etc. These functions are defined through *Procedural Knowledge Primitives* (PKP). These PKP include *relative orientation, dimensions, stability, proximity and clearance*. The sub-categories of a chair are represented as an hierarchically graph with sub-category nodes which are represented at their lowest level with PKPs. For a novel input, the shape of the input is calculated based on the known PKPs in the hierarchically graph, starting at the root node, which represents the most generic category of a chair. For the novel input, the best sub-category is chosen based on simple tests of the input shapes in comparison with the elements in the hierarchically graph.
Even though this work does not explicitly mention affordances, their definition of functions are closely related to affordances. The description of chairs based on features, in this work called PKP, can be adapted to different furniture to include a bigger classification class than chairs and the hierarchically graph can be extended as well. A drawback of this work, is the predefined structure of the hierarchically graph which does not learn new structures of chairs but has to be manually extended to allow for novel sub-categories of chairs. There are several more works concerned with the concept of object recognition from function and shape [PSR08, GESB95, FRSS07].

In "Behavior-Grounded Representation of Tool Affordances"[Sto] by A. Stoytchev the focus on learning affordances was based on an exploratory approach. A robotic arm had to use different kinds of sticks (e.g. straight stick, L-shaped stick, T-shaped stick, etc.) to move a hockey puck to a certain goal position. Affordances in this work are defined as possible position changes of the hockey puck under a certain action with a specific stick. For example, the action "Extend Arm" for the T-shaped stick would move the puck in the direction of the extending arm. Furthermore all possible actions are parametrized to allow for more refined actions. The set of possible actions (also called behaviors in this work) consists of *Extend Arm*, *Contract Arm*, *Slide arm left*, *Slide arm right* and *Position wrist*. During training, these parametrized actions were used on the puck and the result on the puck was observed and saved in a table

with parameters chosen for certain actions, which stick was used and what were the effects on the puck. During the first experiment the robotic arm had to move the puck to four goal areas with one specific tool and had to choose appropriate parametrized actions to move the puck into these goal areas. The most common cause for failure during these experiments was caused by unreachable positions of puck after it was pushed outside the extended work space of the robotic arm with the specific tool. An exception to this was the tool *straight Stick*, as the available actions for the robot would not allow him to use the stick to pull the puck back to a goal area after it was pushed beyond the area. The author suggested to extend the action set with a new action *turn-the-wrist-at-an-angle-and-then-pull* to remedy this drawback. Solving similar future problems, the author suggested to let the robot learn new exploratory actions to explore all possible actions. In a second experiment the robotic arms adaptability to deformed tools was tested. As tools are only recognized by color, the red shaped T-stick was replaced with a L-shaped stick which had the same color as the the T-stick. The robot still acted as if he had a T-stick, as the recognition is based on the color of the stick, but as certain actions were no longer possible their success rate to move the puck were lowered and they were no longer considered as possible actions to move the puck. The robot successfully adapted in all the trials of the second experiment.
This work showed an interesting exploratory approach to discover the capabilities of tools based on the available actions of the robot. The description of affordances differs to the description used in this work as it's not in labels as "push-able", "grab-able" etc. but affordances are described as a pair of parametrized action and tool together with the resulting position change of the puck. This can be quite easily remedied, as a tool that can be grabbed has the affordance to be grab-able and a puck that can be pushed has to affordance to be push-able etc. As can be seen in other work as well, the active interaction with objects to discover affordances or verify predicted affordances is a promising approach for autonomous discovery of affordances of objects. A shortcoming, in my opinion, is the lack of exploring new possible actions with the tools provided or recognizing a subset of possible successful actions depending on the common visual properties between novel tools and already trained tools.

Similar to the previous related work, L. Paletta *et al.* verify predicted affordance through interaction with objects in their work "Learning to Perceive Affordances in a Framework of Developmental Embodied Cognition". The focus of this work is the learning of the affordance *lift-able* based on visual features. A conceptional framework ABACUS (Affordance Based Adaptive Control Using Self-Experience) was developed, which uses affordances extracted from the environment for the control of its actions. During the first step of this framework, SIFT features, geometric information, color and other features were extracted from images with lift-able and non-lift-able objects. These features were then used to predict the affordance of the depicted object. Two categories of objects were used for training and the experiments: Both objects had a cylindric base with either a flat surface or a cone shaped surface on top of the base. A magnetizing end-effector was used to lift the objects, where objects with a flat top surface could be successfully lifted and objects with a cone shaped top surface could not be successfully lifted. To learn the features to recognize objects which were lift-able, reinforcement

learning was used by the authors. Through interaction (trying to lift the object), the visually predication of lift-ability was verified. The reinforcement learner successfully learned the features *rectangle* and *top-region* as two features which defined the lift-ability of an object. Although only two object categories were used in the experiments and only one affordance was used, this strategy shows the successful approach of selecting appropriate features from a set of features to classify the affordance of an object. How this approach handles bigger classes of objects and affordances still needs to be evaluated.

To explore strategies for learning affordances, one is not bound to the real world. In the work "What can i do with this? Finding possible interactions between characters and objects"" by P. Sequiera a virtual environment was used to learn affordances provided by objects to an agent. The perspective of where affordances lie are different here to the perspective of the other related work. In this work, affordances are encoded as properties of objects which can be discovered on certain actions of an agent. So the affordances are a part of the environment and are not part of the capabilities of the agent as in the previous work. To answer their question of "How can an agent identify the possibilities of interaction with an object and the consequences of that interaction, based on previous past experiences with other objects?", they first of all describe four possible perspectives on the interaction of agents with objects. The first perspective is through *Objects and World Modeling*. One widely-used representation here is the concept of *Smart Objects*, which encapsulated all the relevant information of objects in their description. The second perspective is the perspective of *Planning*, which gives the agent all the possible correct interactions and the suitable sequence of interactions with a specific object. In *Object categorization*, the third perspective, objects are classified into categories and objects offer certain action for an agent based on their category. The last perspective is the perspective of *Affordances* in the context of object-agent interaction. The authors closely follow the definition of affordance as it was defined by J.J. Gibson: "Offerings or action possibilities in the environment in relation to the action capabilities of an actor. Affordances are independent of the actor's experience, knowledge, culture, or ability to perceive. Their Existence is binary - an affordance exists or it does not exist." The authors oriented their work based on the perspective of affordances, which makes it interesting as a related work. They introduce a framework for managing object-agent interactions called SOTAI (Smart ObjecT-Agent Interaction). It is grounded on the basic concept, that objects and agents are in an environment where agents can interact with objects. These objects are encoded as actions which offer informations on specific actions. Agents receive this information through sensory channels (called streams). Tokens are elements of the stream, e.g. "color orange". A combination of a stream and a token is called a sensation, e.g. the pair of visual stream and the token "color orange". Certain interactions of an agent with an object trigger sensations. In the training phase of the agent, he interacts with an object through several actions and the triggered sensations of the agent are stored during the start of the action and at the end of an action, e.g. during several interactions with the fruit object "orange" the sensations { smell, fragrant } and { shape, spherical } were triggered in combination, which suggests a correlation of these two sensations. If two sensations are frequently triggered at the same time,

they are combined into a base fluent. A base fluent indicates that a pair of sensations occurs frequently in combination. Base fluents can be chained in a cause-effect way if for two base fluents $b1$ and $b2$, $b2$ occurs after $b1$ finished frequently enough. Such a relationship between base fluents is called a *context*. A context tells us that for a base fluent $b1$ may be the cause for another base fluent $b2$. If $b2$ is the causing base fluent in a context for another base fluent $b3$, a chain can be created $b1 \rightarrow b2 \rightarrow b3$. If two base fluents are both cause for another base fluent, these two base fluents can be combined to a fluent. This modeling of cause-effect of interactions on objects by an agent was used in tests to learn different fluents and base fluents, which described under which interactions which affordances were offered by an object. A fluent learned in an experiment included: { Taste: sweet, Health: healthy, Taste: pleasant, Pain: pleasant, Power: energizing }, which can be interpreted as a general description for food. This and other fluents were learned during a simulation of an agent in a world with eight objects, where the agent chose random actions to interact with objects and then combined the sensations to fluents.
The work showed a successful approach for combining simple descriptions of affordances into complex structures of affordances of one object through a combination of different features. Different affordances could be combined on basis of their observed frequency. The sequence in which they occur is an important aspect to consider, which showed that chains can be created to change the provided affordances of an object through interactions. These generated chains could be used in action planning to achieve a change of offered affordances of an object by appropriately interacting with these objects.

In “Learning Objects and Grasp Affordances through Autonomous Exploration” [KDPB09] by D.Kraft *et al.*, the affordance grasp was learned through a stereo vision system and exploring different grasping actions. Their proposed approach first generates a visual representation of the object by extracting the contour of the object. From the extracted contours, possible grasping locations are identified which allow a robotic arm to grasp the object. These possible grasping locations are tested with an heuristic approach which creates generic grasps. If the object could be grasped successfully with the heuristic approach, the object is placed in front of the stereo vision system. When the object is placed in front of the stereo vision system, further possible grasping locations are extracted and accumulated to define the grasping model of the object. From this model, several grasping locations are tested to build grasping hypothesis of the different location which indicates how successful a grasp at a specified location can be. With this approach the grasping performance of an object could be increased during the test from $41\ \%$ to $81\ \%$.
This work is interesting as it not only classifies an object as graspable or not-graspable, but segments the object into locations which afford successful grasping. If a personal robot can identify a door as graspable, the location (door handle) of where to grasp the door to access the grasp-affordance is not clear. This approach can be adapted to other affordances like push-able, pull-able, etc. to allow for a wider class of affordances where the successful access to an affordance of an object is dependent on the location of the affordance of the object.

In a closely related work by A. Aldoma *et al.* "Supervised Learning of Hidden and Non-Hidden 0-order Affordances and Detection in Real Scenes"[ATV12], affordances of objects are detected by recognizing an object in a 3D scene. The authors divided affordances into three orders, where 0-order affordances represent affordances which solely depend on the object of interest and their current configuration in the world, e.g. sit-able, liquid-containment or roll-able. 1^{st}-order affordances are related to the object-robot embodiment relationship e.g., for a very small robot a sofa does not represent a sit-able affordance but for robots which are big enough, a sofa can afford to be sat-upon. 2^{nd}-order affordances are defined as what an object can afford a robot based on the robots capabilities, e.g. a sofa can afford to be sat-upon for a robot which is big enough according to the 0-order affordance notation, but the sofa does not afford sit-able to a robot which is big enough but is not capable of sitting. These different orders of affordance can also be seen as different perspectives of where affordances reside. These different perspectives are further described in Section 3.2. Furthermore, 0-order affordances are divided into hidden and non-hidden affordances. Hidden affordances are defined as affordances which are part of the object but not access-able in the current configuration of the object, which implies that through interaction with the object it can be placed in a configuration in which all it's 0-order affordances are access-able, e.g. a glass has the affordance of a liquid-containment but if the glass is not in a stable configuration, e.g. rolling on it's side over a surface, it's affordance liquid-containment is considered to be hidden. The discovery of hidden affordances of an object is an important aspect for a robot which can manipulate the object as it can help the robot infer the further possible interactions with an object, if the object is manipulated in such a way that this hidden affordance becomes a non-hidden affordance. So an important task of this work is the calculation of a stable pose for objects. A stable pose is defined as a pose which doesn't change as long as there is no external agent disturbing it. So the goal of this work is to identify hidden and non-hidden 0-order affordances by estimating poses of objects and recognizing objects. For the recognizing different descriptors of object geometry were tested on different generalized machine-learning methods. In a first step a training set of 3D CAD models was generated by manually labeling this models with affordances and whether these affordances were hidden in the current pose or not. These affordances are: *rollable, containment, liquid-containment, unstable, stackable-onto* and *sit-able*. Five different 3D descriptors were used for the description of objects. These include: Spherical Extent Descriptor (SEE)[SV01], Normal Distributions Sliced (NDS), SHOT [TSDS10], Spin Images [JH99] and Point Feature Histogram [RBTH10]. For a description of these features, please refer to the provided references. For the classification, three popular classification methods were tested, namely: Support Vector Machines (SVM), Boosting and Random Forests. The different descriptors and classification methods were evaluated on a set of 45 CAD models provided by the Princeton Shape Benchmark dataset. In the set were models included, which have multiple affordances, among them chairs, benches, mugs, bottles etc. The evaluation showed that some affordances, e.g. stackable-onto, had a lower correct classification rate than other affordances, the descriptors all performed similar well and Random Forests performed less well than SVM or Boosting. A suggestion by the authors is to select the best descriptor and classification method based on the affordance which has to be classified. Furthermore the

whole approach was tested on real world 3D data acquired with a Microsoft Kinect sensor. For this evaluation the object recognition was replaced with another object recognition pipeline as presented in "CAD-Model Recognition and 6DOF Pose Estimation Using 3D Cues"[AVB+11]. To take into account the noisy data acquired by the Microsoft Kinect a different object descriptor was chosen, specifically the Clustered Viewpoint Feature Histogram (CVFH), which is capable of providing a good description even in the case of noisy and partially occluded data. Alongside CVFH, the Viewpoint Feature Histogram (VFH) and Shape Distributions on Voxel Surfaces (SDVS) were used as descriptors as comparison basis for CVFH. Evaluation demonstrated that CVFH outperformed VFH and SDVS by up to $10\ \%$ more accurate affordance detection with an overall accuracy of $70\ \%$ in affordance detection for learned affordances and $84\ \%$ accuracy for manually labeled affordances.
The basic concept of this approach is the discovery of affordances through object recognition. Affordances are considered to be part of a specific object, so the discovery of affordances is reduced to the recognition of objects. The knowledge of which objects have which affordances is already stored in the robot or is encoded in the object description. The authors showed that this knowledge could be trained by using manually labeled 3D CAD models but it lacks the possibility of allowing a robot to discover new object classes which can not be recognized by the object recognition algorithm but exhibit the affordances already learned. For example, the object recognition pipeline would be able to recognize a novel scene which depicts a door, if a door was already learned but a novel scene which depicts a window would be unable to be recognized if there was not already a window learned. Even though a window and a door have a big subset of equal 0-order affordances, the affordances of the window would not be learned. This drawback could be remedied by an appropriate generalized representation of the data which depicts a door and a window. Despite these drawbacks, the presented approach of the authors showed the best results and used very similar definitions of affordances as are used in this work.

The presented related work shows different definitions and views on affordances as well as common exploration techniques. In most of the reviewed work, affordances are verified through active interaction with objects while focusing on the learning of one specific affordance. The work above by A. Aldoma *et al.* is one of the few exceptions, to the authors knowledge, which tries to learn a set of different affordances for a set of objects solely based on visual data. A similar approach will be presented in this work, which will be first preceded by a background section which will introduce the various topics involved for this work.

3 Background

3.1 Instance-Based Learning

Instance-Based learning is in a sense opposite the model-based learning approach. In model-based learning strategies are developed on how a specific problem instance can be abstracted into a model and how one can learn with such a model, how one can learn from the modeling process or what can be learned from a such a model. This is contrary to the instance-based learning approach, which stores specific problem instances and employs strategies to use these stored instances for learning. A big advantage of instance-based learning is its capability of adapting to new problem instances, as they can be simply stored in memory in the simplest variant of IBL. A model-based learning approach has to adopt these new problem instance into it's existing model which can be computationally expensive. IBL is a lazy-learning method as it postpones the processing of the stored instances until a request to handle a novel instance arrives. The reply is constructed by processing the stored training instances and after finishing the request, any intermediate result calculated during the construction of the reply is disregarded and not stored. Formally, IBL can be described as follows:
Consider X to be the instance space with $x \in X$ an instance. Let D_X be a distance metric for X and $L = \{\lambda_1, ..., \lambda_m\}$ a finite set of labels. $\langle x, \lambda_x \rangle \in X \times L$ denotes a labeled instance, where $X \times L$ are the possible observations. Furthermore, let $B \subseteq X \times L$ be the stored labeled instances, also called knowledge base. The problem of classifying a new unlabeled instance x_0 is then the problem of estimating a label λ_{x_0} for x_0.

3.1.1 k-Nearest-Neighbors

A simple IBL algorithm is the k-Nearest-Neighbors (k-NN) algorithm. In k-NN, a new instance x_0 is classified by searching the k, a user defined value, nearest labeled instances $x \in D$ and taking a majority vote[1] among the k nearest labeled instances for the label λ_{x_0} of x_0. One can see here the advantages and disadvantages of IBL. Expressing "near" with a similarity function can pose quite a challenge depending on the instance space. Suppose our instance space X represents all the known animals on earth and L consists of the ability to jump ($L = \{jump, can't jump\}$), expressing nearness as a distance of two animals in the phylogenetics

[1] A majority vote is taken for discrete valued λ and the mean for continuous λ.

tree might give better results than using a metric on the amount of limbs an animal has. For example, let x_0 be an African bush elephant which has to be classified and let B, the knowledge base of labeled instances, contain several animals. In a phylogenetics-tree based metric, the distance between an African bush elephant and other elephants, like the African forest elephant (which we assume to be an element of B with the label $\lambda_{Africanforestelephant} = can'tjump$), is small and the distance to a German Shepard Dog in comparison, is big. In the amount-limbs metric both the African forest elephant and the German Shepard Dog have the same distance from the unclassified African bush elephant and both of them can have an equal impact on the voting of the label $\lambda_{Africanbushelephant}$.

When choosing the amount of nearest neighbors $k > 1$, weighting the neighbors by their distance is a good technique to reduce the influence of neighbors for a sparse knowledge base B. Finding a suitable k highly depends on data at hand and no general value can be given, but using cross-validation and various values for k, a suitable value can be found. Searching the k-nearest-neighbors for a novel instance can be quite computational expensive as the distance for all elements in B have to be computed for the novel instance. To reduce the computation time, A.Moore introduced KD-trees [Moo91] which are an effective way to search for nearest neighbors.

3.1.2 IB1, IB2 and IB3 Algorithm

An extension of the k-NN algorithm was proposed by D.W. Aha *et al.* in their paper "Instance-Based Learning Algorithms" [AKA91]. One of the problems of the k-NN algorithm was, that during training all labeled instances were simply stored in the knowledge-base B, which leads to potentially huge memory requirements, saving noisy data and including labeled instances which are not relevant for the decision boundary of the classification. The goal of their algorithm is to learn a *concept*, which is a function that maps instances to categories. This can be formally written as:
For each label $\lambda \in L$, let $C_\lambda \subseteq X$ denote the set of instances $x \in X$, such that $\{x, \lambda\}$ can be observed. Furthermore, let $C_U = \bigcup_{\lambda \in L} C_\lambda$ be the *concept description*. The concept $\mathcal{C}$ is then defined as $\mathcal{C} : X \to C_U$.

Their proposed algorithms are build upon three components. A *Similarity Function* which computes a similarity for two given instances. A *Classification Function* which receives the result of the *Similarity Function* and assigns a category to an instance. The third component, is the *Concept Description Updater*, which decides whether to include the given newly labeled instance to the concept description (which represents the stored instances, i.e. the "knowledge base" of the robot) or not.
The IB1 algorithm (see Algorithm 3.1) is the simplest one and the basis for the next two algorithms. It functions very similar to a 1-NN algorithm.

Algorithm 3.1 IB1 with CD = Concept Description. Taken from [AKA91]

```
CD ← ∅
for all x ∈ TrainingSet do
    for all y ∈ CD do
        Sim[y] ← SIMILARITY(x, y)
    end for
    y_max ←some y ∈ CD with maximal Sim[y]
    if CLASS(x) = CLASS(y_max) then
        classification ← correct
    else
        classification ← incorrect
    end if
    CD ← CD ∪ {x}
end for
```

First a similarity for an instance x of the training set is calculated for all labeled instances in the concept description, which basically is a distance metric. Then the closest neighbor, y_{max} is calculated, classification is done and the labeled instance x is added to the concept description. Eventually the concept description will consist of all labeled instances of the training set. As the size of the concept description has a direct influence on the computation time, a second algorithm was proposed, the IB2. Tests with IB1 showed that most of the misclassified instances were near the classification boundary, the idea for IB2 (see Algorithm 3.2) was to only store the misclassified instances in the concept description. This reduced the memory requirement of the IB1 algorithm.

Algorithm 3.2 IB2. Taken from [AKA91]

```
CD ← ∅
for all x ∈ TrainingSet do
    for all y ∈ CD do
        Sim[y] ← SIMILARITY(x, y)
    end for
    y_max ←some y ∈ CD with maximal Sim[y]
    if CLASS(x) = CLASS(y_max) then
        classification ← correct
    else
        classification ← incorrect
        CD ← CD ∪ {x}
    end if
end for
```

As IB2 only stored the misclassified instances, noisy instances became a problem because they were stored in the concept description and therefor have a higher impact on the training process. This was no problem in IB1 as all training data was saved. With this background, IB3 (see Algorithm 3.3) was developed which combines IB2 with an evaluation method for the stored instances. The evaluation method maintains a classification counter for each instance in the concept description. Instances in the concept description which have a high amount of misclassifications, have a low classification counter and are therefor removed from the concept description. Tests done by D.W. Aha *et al.* showed that IB3 outperforms IB1 and IB2 in terms of accuracy and memory requirement.

Algorithm 3.3 IB3. Taken from [AKA91]

$CD \leftarrow \emptyset$
for all $x \in TrainingSet$ **do**
  for all $y \in CD$ **do**
    $Sim[y] \leftarrow$ SIMILARITY(x, y)
  end for
  if $\exists y \in CD|$ACCEPTABLE(y) **then**
    $y_{max} \leftarrow$some $y \in CD$ with maximal $Sim[y]$
  else
    $i \leftarrow$a randomly-selected value in $[1, |CD|]$
    $y_{max} \leftarrow$some $y \in CD$ that is the i-th most similar instance to x
  end if
  if CLASS$(x) \neq$ CLASS(y_{max}) **then**
    $classification \leftarrow$ **correct**
  else
    $classification \leftarrow$ **incorrect**
    $CD \leftarrow CD \cup \{x\}$
  end if
  for all $y \in CD$ **do**
    if $Sim[y] \geq Sim[y_{max}]$ **then**
      Update y's classification record
      if y's record is significantly poor **then**
        $CD \leftarrow CD - \{y\}$
      end if
    end if
  end for
end for

In this work, the IB2 algorithm was used as it is simple enough to understand its properties while still providing low memory requirements.

3.2 Affordance

> "The verb *to afford* is found in the dictionary, the noun *affordance* is not. I have made it up." (J.J. Gibson, 1977)

The term affordance was first introduced by J.J.Gibson in "Theory of Affordances" [Gib77]. By this term, he describes a sort of relationship between an object which offers an affordance to an agent who interacts with the object. J.J. Gibson best explained it through various examples. A surface has the affordance stand-upon-able for an agent if it is nearly horizontal, nearly flat, sufficiently extended relative to the agent and its substance is rigid, e.g. the water strider insect can walk upon water, humans can not. A surface which is knee-high affords to be sat upon. "Knee-high" is different for a child compared to an adult. If a chair "looks" sit-on-able, its affordance is perceived visually. There can be hidden and false affordances. Looking at a cigarette lighter, it's not obvious that it can be used to open a beer bottle, this is called a hidden affordance. An affordance is considered to be false, if the perceived affordance does not reflect the actual affordance, e.g. a placebo button. In a scenario, where a robot perceives affordances and learns them, there needs to be a mechanism to identify false affordances to correctly classify and learn the perceived affordance. One possible mechanism would be to test or act upon the perceived affordance of e.g. push-ability of a ball, by pushing the ball and observing the result of this action and verify it against an expected result. Affordances are not limited to objects, according to J.J. Gibson. Mediums, substances, surfaces, objects, places and other agents can have affordance for another agent.

Affordance remains a vague term and there are different formalizations in different disciplines. E. Şahin *et al.* try to formalize affordance in context of robot control in their paper "To Afford or Not to Afford: A New Formalization of Affordances Toward Affordance-Based Robot Control" [SCD+07]. In their work, E. Şahin *et al.* propose definitions of affordance in context of different perspectives. They define three perspectives, the perspective of the environment, the perspective of the agent, who interacts with the environment, and the perspective of an observer, who observes the interaction between agent and environment without being part of the environment-agent system (see Fig. 3.1 for an illustration).

In the perspective of the agent, the affordance relationship between agent and environment is assumed to reside in the agent, i.e. the agent assumes to have an affordance in relation to the environment (e.g. the robot dog has push-ability affordance in relation to the red ball).

In the perspective of the environment, affordances are extended properties of the environment which can be perceived by different agents. Here, the environment offers a multitude of affordances to specific agents, which can differ due to the capabilities of the agents. As an example, a red ball (environment) can offer push-ability to a small robot dog (agent), grab-ability and throw-ability for a human (agent), etc.

In the third perspective, the perspective of the observer, affordances are described by observing an environment-agent system, and the interactions between the agent with the environment.

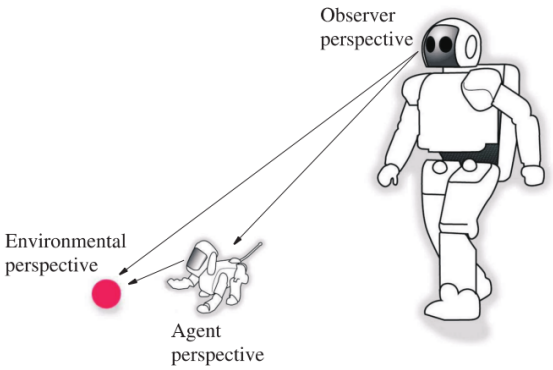

Figure 3.1: Three perspectives for affordance description, where the observer (humanoid robot) is invisible to the agent (robot dog) who interacts with the environment (red ball). Taken from [SCD+07]

This perspective is interesting for learning and discovering affordances, in which the agent is a robot or human with identical or similar capabilities as an observing and learning robot.

My work will focus on the perspective of the agent and learning affordances the agent has with a set of objects (environment).

3.3 Kinect

For this work a Kinect 360 was used with the background that this work will be tested on a PR2 robot of Willow Garage[2]. As there are no official resources available to the public concerning the specification of the Kinect 360, some of the specification were reverse-engineered. The specifications found vary slightly in their values and therefor we will consider the following specifications, which represents the values most common among the different specifications found, as true for this work.

The Kinect consists of an IR emitter, IR camera, RGB camera, audio peripherals and a motorized base which can tilt the Kinect head. Audio peripherals and the motorized base are of no interest for this work and will not be mentioned further. The Kinect is capable of capturing depth and color simultaneously at up to 30 FPS. The IR sensor[3] is capable of a resolution of 1280x1024, but due to bandwidth limitation of the USB connection, the resolution is reduced to 640x480. The angular field of view is estimated at 57 degrees horizontally and 43 degrees vertically. Taken from the datasheet of the IR sensor, a theoretical operating range of 0.5 meters to 5

[2]For more information on the PR2 please refer to: http://www.willowgarage.com/pages/pr2/overview
[3]MT9M001: http://www.aptina.com/products/image_sensors/mt9m001c12stm/

meters is possible, but tests showed that the range between 0.8m to 3.5m delivers good values with acceptable errors [KE12]. Error sources can be divided into three groups:

Sensor: For a properly functioning device, errors can be seen due to inadequate calibration of the camera and/or inaccurate measurement of disparities. K. Khoshelham *et al.* further showed in their work [KE12], that the random error of depth measurements increase quadratically with increasing distance and at maximum range of 5 meters is approximately 4 cm. The depth resolution decreases with increasing distance, which results in a point spacing in depth direction of 7 cm at 5 meters. The point spacing can be observed as a sort of layering of a flat surface (see Figure 3.2 for an example).

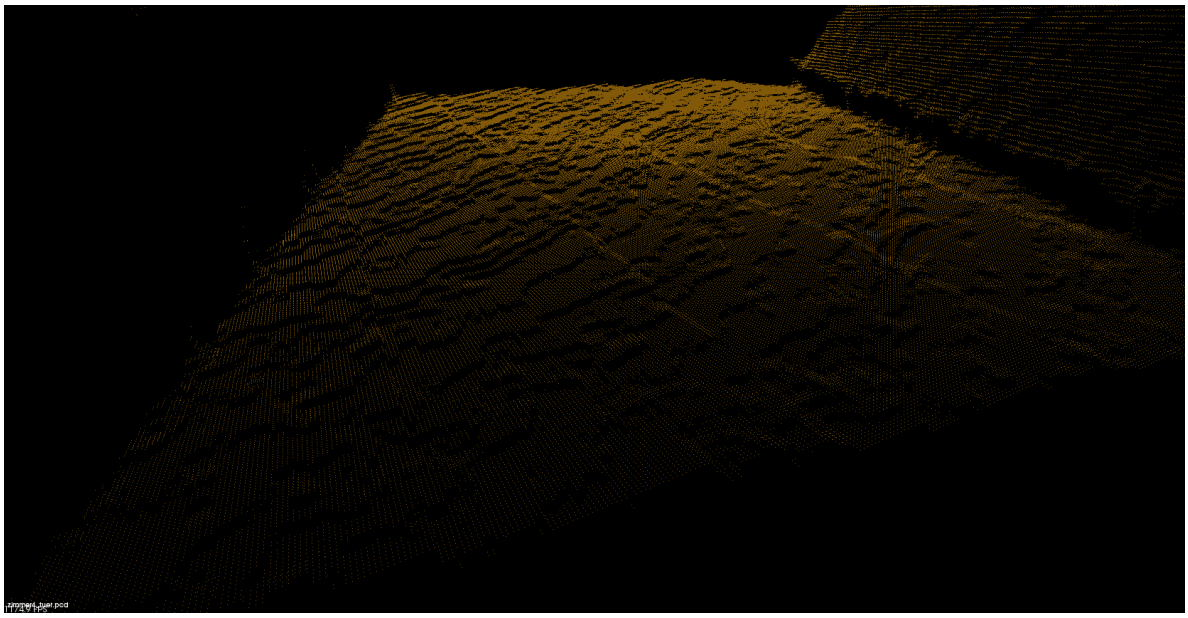

Figure 3.2: Point Cloud of a door showing the layering effect caused by point spacing.

Measurement setup: Strong light can lead to gaps and outliers in the depth measurement, so lighting conditions have to be considered. The practical operating range of 0.8-3 meters has to be taken into account as well.

Properties of the object surface: As the Kinect utilizes an IR emitter, smooth and shiny surfaces pose a challenge to receiving satisfying depth measurements.

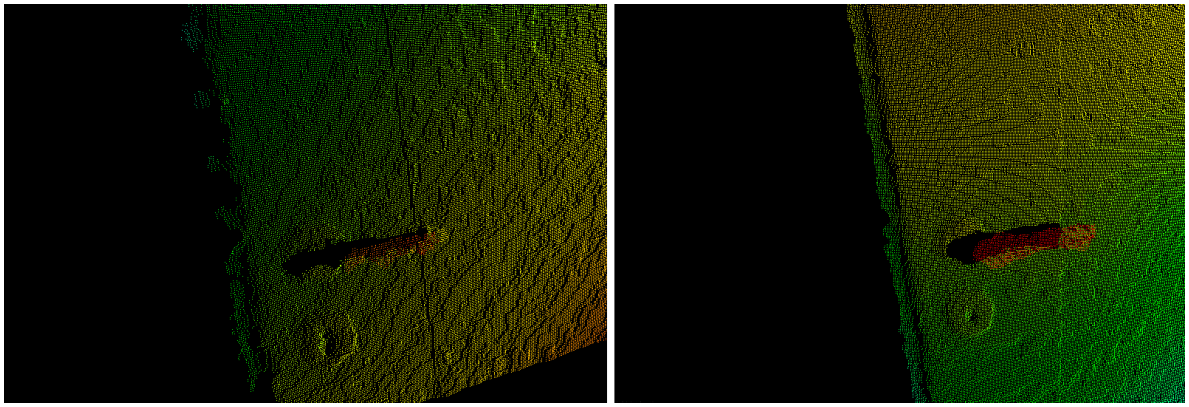

Figure 3.3: Point Cloud of a metallic door handle (red) showing the difference of object surface properties. (Left) Original scene. (Right) Door handle wrapped with paper to cover metallic material. One can see the difference in size of the points representing the door handle once with metallic material and once with paper covering the metallic material.

3.4 Point Cloud Library

The Point Cloud Library (PCL) is a standalone open-project for 2D/3D image and point cloud processing. The foundation of algorithms was first developed in the PhD thesis of Rusu B. Radu [Rus10] as a *framework for Semantic 3D Object Model acquisition from Point Cloud Data*. The work was then continued at *Willow Garage*[4] where the framework was used in 3D data processing for Willow Garage's PR2 in conjunction with the *Robot Operating System* (ROS) [5] and eventually it was extended to a standalone library. In March 2011, PointClouds.org was the new home of PCL and it's first version 1.0[6] [RC11].

The current version of PCL is 1.7.1 and the project has around 140 contributers at GitHub [7]. PCL contains several state of the art algorithms for filtering, surface reconstruction, segmentation, features, sample consensus, I/O, kdtree, etc., grouped in modular libraries.

In this work, PCL was extensively used for data acquisition and manipulation.

[4] http://www.willowgarage.com/
[5] http://www.ros.org/
[6] http://www.willowgarage.com/blog/2011/03/27/point-cloud-library-pcl-moved-pointcloudsorg
[7] https://github.com/PointCloudLibrary/pcl

3.4.1 Data Acquisition

In it's first version, PCL implements a framework for grabbing and accessing data for OpenNI-based cameras. Open Natural Interaction (OpenNI) is an industry-led, non-profit organization focused on natural interaction devices, applications that use those devices and middleware that facilitates access and use of such devices . One of the main members of OpenNI is the company *PrimeSense*, which is responsible for the technology of the *Kinect by Microsoft*, *Xtion Pro* camera by *Asus* and others. The framework not only allows the access and saving the data through a few lines of code but can be easily visualized through a visualization library based on *VTK*[8].

As of version 1.7.1, not only OpenNI-based cameras are supported but also PXC devices, the Velodyne High-Definition-Laser, RobotEye by Ocular Robotics and more.

PCL has it's own file format: The Point Cloud Data (PCD) file format.

Although numerous different file formats for 3D point cloud data already exist (PLY, STL, OBI, etc.) they all have various disadvantages for the efficient use in PCL as they were developed for their respective use and purpose.

PCD consists of two parts: the file header and the actual data for each point. The header contains properties of the point cloud data stored in the file. Among the more interesting properties are *FIELDS*, which defines the name and dimension that a point can have. Dimensions can be XYZ, RGB (color), Normals or even moment invariants. These dimensions can be combined in one PCD file to display and save an array of information for one single point. Another important property are *WIDTH* and *HEIGHT*, which can have two meanings depending on their values. For unorganized point clouds, WIDTH contains the total number of points with HEIGHT having the value 1. For organized point clouds, WIDTH represents the total number of rows of the point cloud and HEIGHT the total number of columns, e.g. WIDTH 640 and WIDTH 480, which results in 307200 points total in the file. Organized point clouds resemble a matrix-like structure, where the point cloud data is split into rows and columns. This format is used in stereo and Time-of-flight cameras, like the Kinect. The advantage of this format lies in knowing the relationship between points and using this knowledge to speed up the computation time and lowering the costs of search algorithms like nearest neighbor search and other algorithms in PCL.

Furthermore the property *DATA* specifies the format in which the data of the points is saved. It's value can be either *ASCII* or *binary*. This makes it possible for users to easily edit the data with other tools in the ASCII format or to have a very fast access to the point cloud data in the binary format. PCL also provides tools to convert PCD files saved in binary to ASCII and vice versa. Listing 3.1 shows exemplary content of a PCD file with a the complete header properties.

[8] http://www.vtk.org/

Listing 3.1 Exemplary content of the PCD file format.

```
# .PCD v0.7 - Point Cloud Data file format
VERSION 0.7
FIELDS x y z
SIZE 4 4 4
TYPE F F F
COUNT 1 1 1
WIDTH 235671
HEIGHT 1
VIEWPOINT 0 0 0 1 0 0 0
POINTS 235671
DATA ascii
0.26563522 0.10937872 1.8352032
0.26914468 0.10939294 1.8351189
0.2743496 0.1097804 1.8450058
0.27787808 0.10979469 1.8449211
0.28140658 0.10980898 1.8448364
0.28493506 0.10982329 1.8447517
0.28846353 0.10983757 1.844667
0.29199201 0.10985186 1.8445822
...
```

3.4.2 Data Manipulation

Data Acquisition represents only one of a range of modular libraries in PCL. Another major part of those modules consists of visualization, representation and manipulation of the point cloud data. The following paragraphs give a short overview of the significant modules used in this work.

- Module Filters: The module Filters contains algorithms and mechanisms to remove outliers, reduce noise and apply filtering methods like a PassThrough in point cloud data. The methods in Filters are very useful for preprocessing the point cloud data for further manipulation and representations.
- Module Segmentation: In real applications, dividing the point cloud data in relevant and irrelevant data for that specific application is essential for a proper execution. Additionally, clustering relevant data into distinct groups might be necessary. These tasks can be handled with the module Segmentation.
- Module Registration: The goal in registration is to combine two or more datasets into a global consistent dataset. The general approach is to find corresponding points in both datasets and then to transform both datasets in such way to minimize the distance between those corresponding points. The registration process is considered to be complete, if the distance between corresponding points is below a threshold or the maximum amount of iterations is reached.

- Module Sample Consensus: The module Sample Consensus contains algorithms like RANSAC and models which describe geometric figures like planes, cones, sphere, etc. This combination allows to fit geometric models to point cloud data.

4 Instance-Based Learning of Affordances

In this section we will take a look at the presented problem and proposed solutions.

4.1 Problem Description

Object recognition is one of the biggest challenges for a service robot in a home environment. Recognizing objects is important for a robot to navigate through a room or executing actions like delivering objects to a human. If the task of a robot is to explore a new area or adapt to dynamic environments, like households, object recognition serves to identifying objects with which a robot can interact or which of these recognized objects are obstacles and have to be avoided during navigation. In the definition of J.J.Gibson, affordances describe possible interactions of an agent with an object. This knowledge is very useful for any task-driven robot as tasks can be described in a more general manner, e.g. instead of the task "Bring Cup A, if you can't find Cup A, bring Cup B." a task can be described as "Bring me a object which affords liquid-containment", which is a general description of a cup. Though, the creation of a general description of affordances of objects in a computer vision sense remains a great challenge. In the related work researched by the author, affordances were either stored in the object description or stored in the knowledge base of the robot, so the goal was an object recognition, following an assignment of the affordance knowledge for a specific object to the recognized object, e.g. the pair (cabinet drawer, pull-able) was trained into the knowledge base of a robot, then the affordance discovery of a new cabinet drawer was based on the object recognition of the new cabinet drawer as a cabinet drawer. Another common approach is the focus on one particular affordance, e.g. grasping, and a focused active interaction of the robot to verify this affordance, e.g. for the affordance grasping, only grasping motions were executed for the verification. Learning by imitation was used in other work as well, which would place the robot into the perspective of an observer, which is not the perspective used in this work (for more on perspective on affordances, see 3.2).

4.2 Proposed Approaches

The proposed approaches are based on the capabilities and algorithms present in the Point Cloud Library and try to learn and discover affordances solely based on different visual representations of an object.

4.2.1 Iterative Closest Point

The first proposed approach follows the concept as in other work of the object recognition. The IB2 algorithm, as outlined in Section 3.1.2, is employed with a similarity score acquired from the Iterative Closest Point (ICP) algorithm. The ICP algorithm is given a source and a target point cloud. In each iteration, the closest point in the target cloud is estimated for each point in the source cloud. A transformation is estimated for the source cloud, which tries to minimize the mean squared distance between these reference points. Then the transformation is applied to the source cloud. The algorithm terminates either after a user-defined amount of iterations, the mean squared distance is below a user-defined threshold or the difference between the previous transformation or the currently estimated transformation is below a user imposed threshold. After the ICP algorithm terminates the mean squared distance is used as a similarity value for the IB2 algorithm (see Algorithm 4.1) .

Algorithm 4.1 Similarity with ICP algorithm

```
procedure SIMILARITY(PointCloud a,PointCloud b)
    source cloud = a
    target cloud = b
    align source cloud to target cloud.
    if ICP has terminated then return error squared distance
    end if
end procedure
```

Ground-truth affordances are encoded in the saved point clouds, which allows for an effective classification. A novel point cloud is classified with the same affordance as the most similar point cloud chosen through the IB2 algorithm. If the affordance of the novel point cloud is the same as the affordance of the most similar point cloud, the classification was correct. ICP was chosen to calculate the similarity score, as point clouds of similar objects would have a higher similarity score compared to point clouds of very different objects. Major drawbacks of ICP include:

- The estimation of the closest point for each point in the source cloud is prone to estimate local minima of the closest point, which leads to optimization of transformations which do not tend towards the global minima of the closest points.

- The runtime of ICP is noted as $\mathcal{O}(n^2)$, which lead in preliminary tests to runtimes of up to 4 hours for two raw point clouds on a Intel i7-3770 CPU with 3.40GHz.
- As ICP is an iterative algorithm, threading could not be implemented to reduce the runtime.

4.2.2 PrimGeom

The second approach is an exploratory approach which was developed with the idea in mind that an object must have certain shape properties which provide an agent with affordances. An agent which is pull-able needs to have a shape which allows an agent to have a stable grasp and this shape must be connected to the object to allow for a change of position upon pulling the object. For example, the shape of a door handle allows to pull a door, so a door has the affordance pull-able because it has a shape which allows for a grasping and pulling motion. This is not only considered true for a door but for any object consisting of a planar surface, a graspable shape and hinges, which allow movement. According to this line of thought, objects are represented as primitive geometric shapes to estimate their affordance based on the primitive geometric shapes which constitute the object. This representation will be called PrimGeom in the rest of this work. Figure 4.1 gives an overview of the acquisition of the PrimGeom.

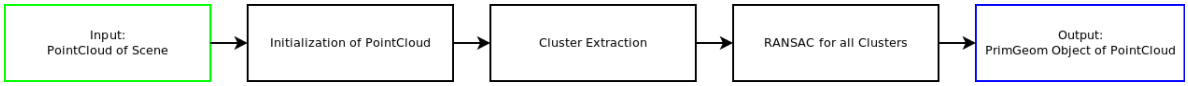

Figure 4.1: Overview of second approach of representing a point cloud as a PrimGeom object.

To acquire the PrimGeom representation of objects, point clusters are extracted from the point cloud depicting an object. The goal of the cluster extraction is to estimate a primitive geometric shape for points in the point cloud which are close in a euclidean sense. The *Euclidean Cluster Extraction*[Rus10] algorithm was employed to extract these clusters (see Figure 4.3 for an example). Due to noise, very small clusters, which are not considered significant for the identification of affordances, a minimum cluster size of 300 points per cluster is imposed on the extraction algorithm and points which are further than 2 centimeters away from each other are considered to be in different clusters.

After segmenting the cloud into clusters, different primitive geometric shapes are used to estimate the clusters. The primitive geometric shapes are chosen based on the present models in PCL and are: plane, sphere, cone and cylinder. For the primitive geometric fitting step, a RANSAC algorithm is utilized. Among all the geometric shapes which are fitted to the data, the shape which will represent the cluster is chosen based on the amount of point inlier's the model has. Preliminary tests showed that a sanity check on the selected geometric shape was necessary, as for certain point clusters the model coefficients of the geometric shapes did not

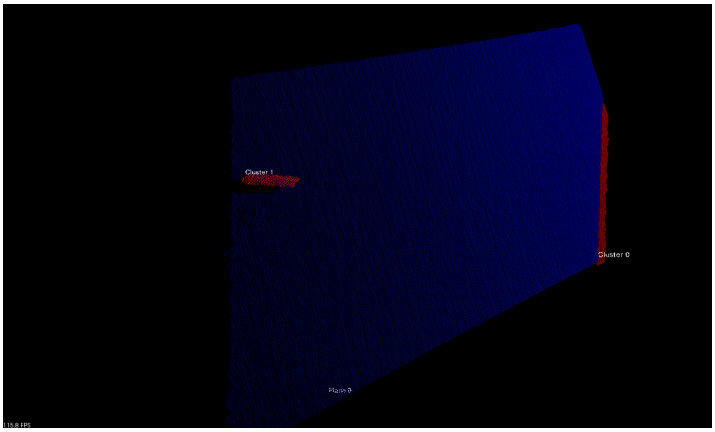

Figure 4.2: Point cloud of a part of a door with three clusters acquired from the *Euclidean Cluster Extraction* algorithm.

reflect any realistic values (e.g. a plane-like cluster was approximated with a sphere with a radius of 50 meters, which is an unrealistic size of an object which can be measured with a Microsoft Kinect). This posed a problem as the model coefficients were used as a similarity score for the clusters in one of our similarity scores. If such unrealistic model coefficients were found during the primitive geometric shape fitting step, the next best model was chosen which had realistic values. Realistic values are considered to be a maximum area of $5.5meters^2$ for planes, $4meters$ as a maximum radius for spheres and cylinders and cones with a maximum volume of $19meters^3$. These values were determined due to the physical and practicable measurable dimensions of objects of a Microsoft Kinect.

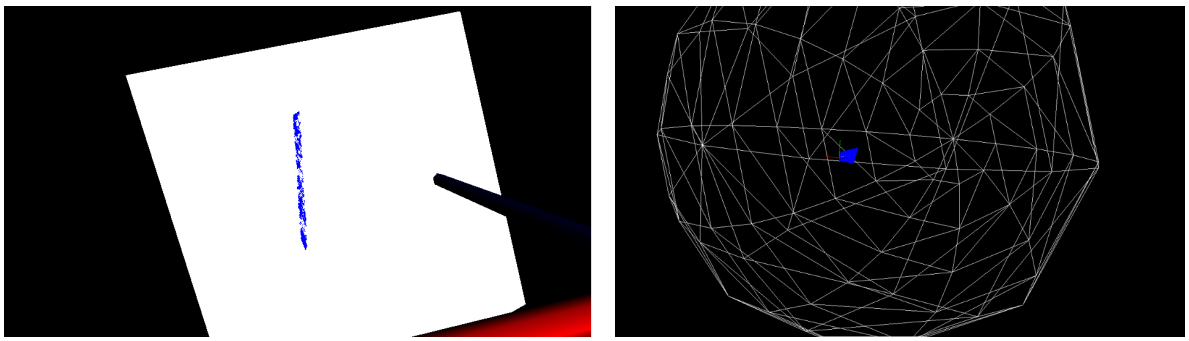

Figure 4.3: Point clouds with fitted primitive geometric shapes. (Left) A point cluster (blue) with a fitted plane model (white). (Right) A plane-like shaped point cluster (blue) which was fitted with a sphere model with unrealistic model coefficients.

Based on the acquired clusters and their primitive geometric shape representation (called cluster type from here on) different similarity scores were developed. The first method would

look at the difference of the amount of clusters for each object. Objects with a similar amount of clusters would be considered similar and would have the same affordance. In the second method the amount of equal cluster types of two objects would be used as score. For the third method a heuristic similarity score was developed, which tried to assign a similarity score based on the cluster type and the model coefficients of the primitive geometric shapes which represented the cluster. The model coefficients for the primitive geometric shapes are:

- Plane:
  - 3D coordinates of the plane's normal.
  - Fourth Hessian component of the plane's equation.
- Sphere:
  - 3D coordinates of the sphere's center.
  - Radius of the sphere in meters.
- Cylinder:
  - 3D coordinates of a point located on the cylinder's axis.
  - 3D coordinates of the direction of the cylinder's axis.
  - Radius of the cylinder in meters.
- Cone:
  - 3D coordinates of the cone's apex.
  - 3D coordinates of the direction of the cone's axis.
  - The opening angle of the cone.

For the similarity of spheres and cylinders the radius coefficient was used. Due to the implementation of the model coefficients of the plane and cone model in PCL, different scores had to be developed for their similarity scores. As the plane model coefficients are not invariant to transformations of the viewpoint, the area of the plane is estimated and used as a similarity score between planes. The coefficients of the cone model have the same invariant problems as the plane, so the volume is estimated and used as a similarity score for clusters which have cone as a cluster type. The estimation of area and volume is done through *Qhull*[1] library which PCL uses for this purpose. To be able to compare two clusters with different cluster types, the centroid of both clusters are aligned and a transformation matrix is acquired with the Iterative Closest Point algorithm to transform both clusters as closely as possible. After this transformation, the spatial difference of both point clusters is obtained through a segmentation algorithm. The size of point cloud acquired with the spatial difference algorithm is then used

[1]See: http://www.qhull.org/

as a score for clusters with different cluster types. To reiterate the different scores with another formulation:

- For two point clusters a and b, both with cluster type “plane”, the score is defined as:

$$s_{pl}(a,b) = \frac{a_{\text{area}}}{b_{\text{area}}}, \quad w.l.o.g.\ a_{\text{area}} > b_{\text{area}},\ b_{\text{area}} \neq 0 \tag{4.1}$$

- For two point clusters a and b, both with cluster type “sphere”, the score is defined as:

$$s_{sp}(a,b) = \frac{a_{\text{radius}}}{b_{\text{radius}}}, \quad w.l.o.g.\ a_{\text{radius}} > b_{\text{radius}},\ b_{\text{radius}} \neq 0 \tag{4.2}$$

- For two point clusters a and b, both with cluster type “cylinder”, the score is defined as:

$$s_{cy}(a,b) = \frac{a_{\text{radius}}}{b_{\text{radius}}}, \quad w.l.o.g.\ a_{\text{radius}} > b_{\text{radius}},\ b_{\text{radius}} \neq 0 \tag{4.3}$$

- For two point clusters a and b, both with cluster type “cone”, the score is defined as:

$$score_{co} = \frac{a_{\text{volume}}}{b_{\text{volume}}}, \quad w.l.o.g.\ a_{\text{volume}} > b_{\text{volume}},\ b_{\text{volume}} \neq 0 \tag{4.4}$$

- For two point clusters a and b with different cluster types the score is defined as: $SP_{diff} = size(\text{spatial difference of both point clusters})$

$$score_{diff} = 1 - \frac{SP_{\text{diff}}}{MAX_{\text{DIFF}}}, \quad with\ \ MAX_{\text{DIFF}} = 640 * 480 = 307200 \tag{4.5}$$

Additionally, for the comparing of clusters with different cluster types a similarity factor was used to penalize different primitive geometric shapes. The factor was chosen heuristically, based on how the RANSAC algorithm fitted the point clusters to the different primitive geometric shapes:

	Plane	Sphere	Cylinder	Cone
Plane	$1 \cdot score_{pl}$	$0.5 \cdot score_{diff}$	$0.25 \cdot score_{diff}$	$0.25 \cdot score_{diff}$
Sphere	$0.5 \cdot score_{diff}$	$1 \cdot score_{sp}$	$0.25 \cdot score_{diff}$	$0.25 \cdot score_{diff}$
Cylinder	$0.25 \cdot score_{diff}$	$0.25 \cdot score_{diff}$	$1 \cdot score_{cy}$	$0.5 \cdot score_{diff}$
Cone	$0.25 \cdot score_{diff}$	$0.25 \cdot score_{diff}$	$0.5 \cdot score_{diff}$	$1 \cdot score_{co}$

Based on the above described scoring system, a similarity score between two objects is calculated by searching for the maximum average sum of similarity scores between all cluster pairs of the two objects. Formally:
For two objects a and b, consisting of $\mid n \mid$ and respectively $\mid m \mid$ point clusters, let $score_{i,j}$ denote the similarity score of the $i-th$ cluster of object a and the $j-th$ cluster of object b. Computing all scores of all clusters of object a and object b according to their type, we will get a set $\mathsf{S} = \{score_{1,1}, \cdots\cdots, score_{i,j}, \ldots, score_{n,m}\}$. For example, if cluster i of object a is a plane and cluster j of object b is a plane as well, $score_{i,j} = score_{pl}(i,j)$. Analog for the other types and if the types of cluster i and cluster j are different, we will use $score_{\text{diff}}$ as the $score$ for cluster i and cluster j. The goal is now to select elements from S such that the sum of the selected elements is maximal. As object a and object b can have a different amount of clusters, we will only consider as many scores as there are clusters in the object with the least amount of clusters. Another constraint is that if we select a $score_{i,j}$ all other $scores_{i,\{1,\ldots,m\}}$ and $scores_{\{1,\ldots,n\},j}$ are discarded from S as we have then assigned a score to the clusters i and j. The sum of the selected scores is then averaged to get uniform scores. The score for two objects a and b is then formally:

$$score_{a,b} = \arg\max_{i,j} \left(\frac{1}{n} \sum^{n} score_{i,j} \right), \quad \text{w.l.o.g. } n > m \tag{4.6}$$

To reduce the computation time of $score_{a,b}$, the result is greedily approximated by always selecting the maximum $score_{i,j} \in \mathsf{S}$, $\forall i \in n, \forall j \in m$ with the same disregarding step.

$$score_{approx\ a,b} = \frac{1}{n} \sum^{n} \max_{i,j} \left(score_{i,j} \right), \quad \text{w.l.o.g } n > m \tag{4.7}$$

In the next section we will take a look at the experiments done to evaluate the presented approaches with different representations of the point cloud data of the objects.

5 Experiments

5.1 Data Set for Experiments

The data used in the experiments were furniture taken from an artificial living room setting (see Fig. 5.1) with a *Microsoft Kinect 360*. The furniture includes:

- An armchair.
- A door.
- A lamp.
- A table-like box.
- Two cabinets.
- A window.

In the following, the two cabinets will be distinguished as "black cabinet" and "wood cabinet".

Figure 5.1: The artifical living room setting with some furniture.

For each furniture more than 50 point clouds were saved from different viewpoints and in different configurations (e.g. door open, door closed, etc.). From these point clouds, three datasets were generated with 20 point clouds for each furniture, with 10 randomly selected

Furniture	Affordance
armchair	rigid
box	rigid
black cabinet	grab_pull
wood cabinet	grab_pull
door	grab_push
lamp	rigid
window	grab_push

Table 5.1: Affordance table – Assignments of affordance labels to furnitures.

point clouds for each furniture in the training set and 10 randomly selected point clouds in the test set. These three datasets include:

1. Raw dataset: No modification on the data taken with the Kinect sensor.
2. Extracted dataset: Each point cloud was manually segmented to acquire a point cloud only depicting the object of interest.
3. Extracted + Aligned dataset: Each point cloud in the extracted dataset is aligned with a point cloud of the same scene from a slightly different viewpoint. The goal of the aligning is to acquire denser point clouds and reduce the influence of sensor based errors (see Section 3.3).

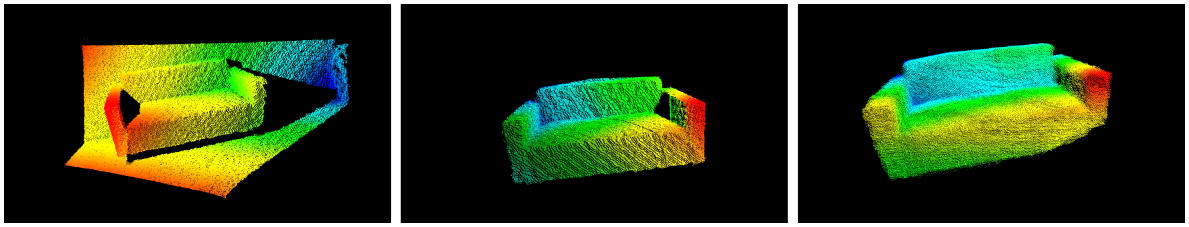

Figure 5.2: Exemplary data from the three dataset each of the armchair furniture. (Left) Raw dataset. (Middle) Extracted dataset. (Right) Extracted + Aligned dataset.

For the classification of affordances, each furniture was assigned a simple affordance label. See table 5.1 for the assignments. The affordance assignments were chosen with consideration of the capabilities of the *PR2* robot of Willow Garage[1]. The affordance *rigid*, represents the affordance of an unmoving object which affords to be an obstacle to the robot. *Grab_pull* represents the affordance of objects which afford to be grab- and pull-able, analog for *grab_push*.

[1]http://www.willowgarage.com/pages/pr2/overview

In an initialization step, each point cloud can be downsampled with a VoxelGrid filter. In the downsampling process a voxel grid is laid over the point cloud data with a user-defined voxel size. All points inside a voxel are then approximated as the centroid of all points contained in that voxel. For this work a voxel size of 1 centimeter was chosen to preserve enough detail of the clouds while reducing the influence of noise.

For the experiments, all datasets in combination with the initialization steps were used, but, to the authors regret, due to unexpected long runtimes (116 hours and still running) and time restrictions, only a part of the experiments could be finished. These experiments are in particular experiments use the Raw dataset or any method which uses ICP.

The Figures 5.3, 5.4 and 5.5 show the average amount of clusters extracted by the cluster extraction algorithm, once without downsampling and once with downsampling, for each dataset.

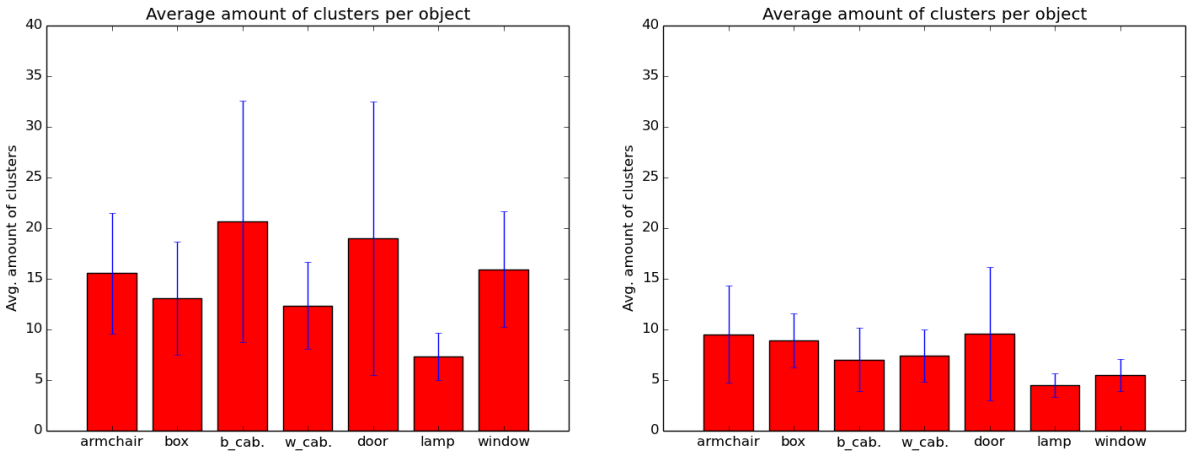

Figure 5.3: Average amount of clusters extracted per furniture and its standard deviation for the Raw dataset. (Left) Without downsampling. (Right) With downsampling.

Overall, one can see that the average amount of clusters extracted is reduced after downsampling the clouds in the dataset. This can be seen quite clearly for the Raw dataset and the Extracted+Aligned dataset. The high amount of average clusters in the Extracted+Aligned dataset without downsampling can be explained through the aligning process, which can produce quite noisy data if the two point clouds used for the aligning process have significantly different viewpoints. Through downsampling this noise is handled quite well, as the amount of the average clusters extracted is significantly reduced, as well as the variance in the amount of clusters extracted per object. The Raw dataset benefits through the downsampling in the same way as the Extracted+Aligned dataset. In the Extracted dataset, only small changes can be observed, which is expected as the data was manually segmented and noisy data was removed.

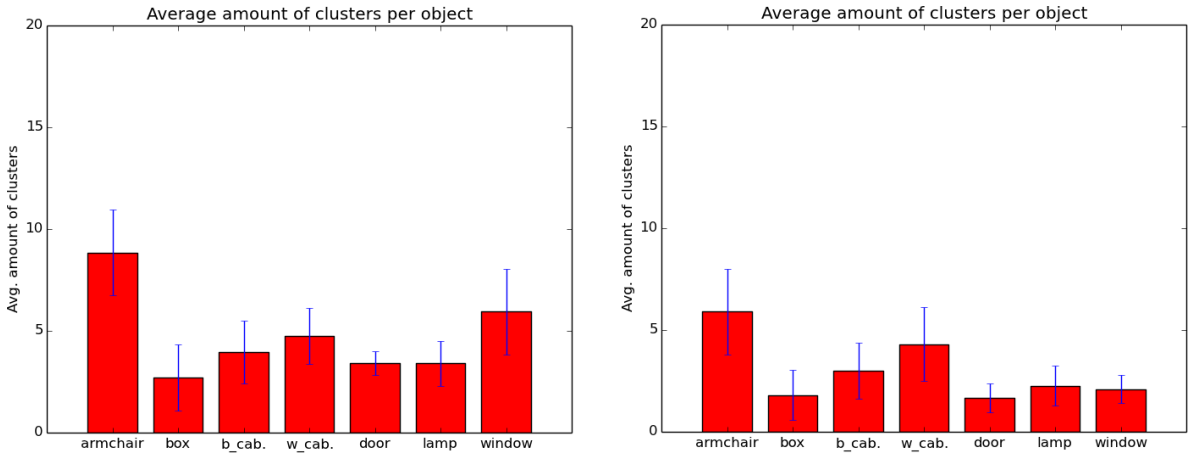

Figure 5.4: Average amount of clusters extracted per furniture and its standard deviation for the Extracted dataset. (Left) Without downsampling. (Right) With downsampling.

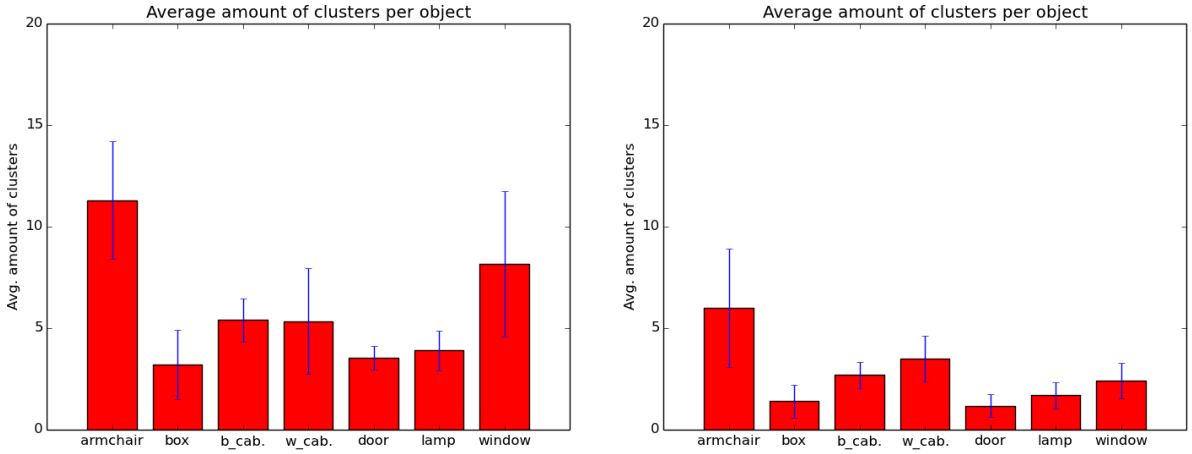

Figure 5.5: Average amount of clusters extracted per furniture and its standard deviation for the Extracted + Aligned dataset. (Left) Without downsampling. (Right) With downsampling.

Figure 5.6 shows the accomplished accuracy of the different approaches (ICP + DS = First approach + downsampling, # = Amount of clusters in the PrimGeom approach, # + DS = as before together with downsampling, Eq. = Amount of equal cluster types in the PrimGeom approach, Eq. + DS = with downsampling, Heu. + DS = The heuristic approach and downsampling) on the datasets (RAW = Raw dataset, EX = Extracted dataset, EX+A = Extracted + Aligned dataset).

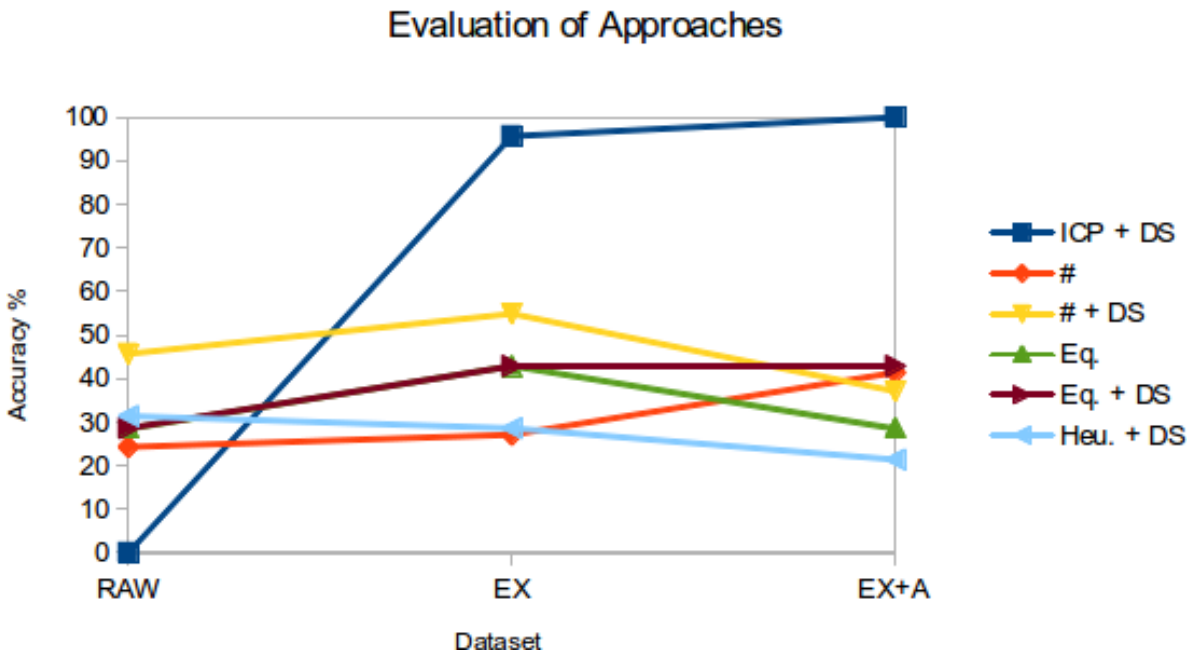

Figure 5.6: Accomplished classification rate with the different approaches on the different datasets. (RAW with ICP+DS is a dummy value, the experiment could not be finished in due time)

One can observe that the second proposed approach performs bad overall and the first approach shows the best results. This can be due to several factors. The methods used to calculate a similarity score for the IB2 algorithm do not seem to be distinctive enough to robustly classify the different affordances. Some of these methods are as bad as picking an affordance randomly for a furniture. Furthermore, it can be seen that the methods of the second approach perform equally well on all the datasets. One of the biggest differences can be seen in the computation time of these methods. Through downsampling of the point clouds, the computation time for the same dataset and method could be reduced by a factor of up to 170, e.g. reducing the test time for the Extracted+Aligned dataset for the first method of the PrimGeom approach from approximated $1.18 \cdot 10^4$ seconds to 69 seconds for a decrease in accuracy of 4.3%. For the first approach with ICP, one can see a slight increase of accuracy with the Extracted+Aligned dataset which might suggest a representation of the data with denser clouds is more favorable for the ICP algorithm. This can be further verified through a bigger dataset for training and testing.

Suggestions on how to improve the performance of the PrimGeom approach, are discussed in the following section along with possible extensions for the overall work and a conclusion.

6 Discussion and Conclusion

The work presented the discovery of affordances with Instance-Based Learning on different data representations in form of different datasets and the possibility of downsampling. Additionally, a shaped-based data representation was explored as a basis for affordance learning. Although the experiments were unsatisfactory, it is the belief of the author that further improvements can be made for this approach to improve its performance. The cluster extraction for the point clouds is highly depended on the data present and representative clusters could be extracted from point clouds by manually adjusting parameters for the cluster extraction algorithm. A robust cluster extraction of clusters which are significant for the discovery of the object affordance is therefor one of the main possible extensions. Furthermore, were the model coefficients provided by the primitive geometric shapes not suitable for a unified calculation of similarity scores. This significance of this drawback can be reduced by extending the shape description with spatial information of these shapes. Objects like doors, windows and cabinet drawers usually have a planar surface with an additional shape located in suitable grasping height and they are located either near the edges (e.g. doors) of the planar surface or located near the center (e.g. cabinet drawer). Additionally the pose of these shapes in the world can be used to calculated the relative position of the shapes. For example, chair-like objects which afford to be sat upon have two nearly orthogonal planar surfaces which are near each other. Another possible adjustment would be to replace the models used in this work by superquadrics which were currently explored as possible model descriptions for PCL[1]. Superquadrics allow for a unified description of a variety of geometrical shapes, which would make the various methods used for calculating a similarity score in the heuristic method.

Even though ICP gave the best results in terms of accuracy, its computation time is unacceptable for any real-time tasks. The overall training and testing phase can be sped up by utilizing threads for pairs of different point clouds the iteration phase can't be sped up in its current implementation. Another drawback for ICP is its optimization of the transformation to a local minima. This problem was recently addressed by J. Yang *et al.* in their work "Go-ICP: Solving 3D Registration Efficiently and Globally Optimally"[YLJ13], which extended the ICP algorithm to robustly optimize towards the global minima with a reduced computation time. The results presented by J. Yang *et al.* seem promising and can possibly replace the ICP algorithm used in this work.

[1]A report can be found at: https://github.com/aichim/superquadrics

The overall instance-based learning framework employed in this work, namely IB2 by D.W. Aha *et al.*, can be extended to the IB3 algorithm, which has the best performance regarding computation time and memory requirement according to D.W. Aha *et al.* This algorithm can be further extended by considering not only the most similar instance for the classification process, but in similar sense as in k-NN, consider the k most similar instances for the classification of affordances of an object. Along this line of thought, the classification can be extended by not only classifying an object with a single affordance but using a multi-labeling strategy to learn multiple affordance labels for an object.

Another challenge in this work was the handling of raw data extracted with the Kinect sensor. The raw point clouds exhibit a large portion of noisy data which is not significant for the classification of the affordance of an object. The environmental context of an object can be used as additional information for affordance classification but for current classifications algorithms they pose a hindrance as they require disproportional more memory and computation time compared to the information they provide. A useful extension for raw point cloud data would be the detecting and filtering of interesting objects in raw point cloud data. Such a detecting and filtering algorithm is presented by O. Erler in [Erl11] and can be a useful step towards autonomous learning of affordances by a personal service robot.

Bibliography

[AKA91] D. W. Aha, D. Kibler, M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, 1991. doi:10.1007/BF00153759. URL http://link.springer.com/10.1007/BF00153759. (Cited on pages 7, 18, 19 and 20)

[ATV12] A. Aldoma, F. Tombari, M. Vincze. Supervised Learning of Hidden and Non-Hidden 0 -order Affordances and Detection in Real Scenes. pp. 1732–1739, 2012. (Cited on page 15)

[AVB+11] A. Aldoma, M. Vincze, N. Blodow, D. Gossow, S. Gedikli, R. Rusu, G. Bradski. CAD-model recognition and 6DOF pose estimation using 3D cues. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pp. 585–592. 2011. doi:10.1109/ICCVW.2011.6130296. (Cited on page 16)

[Erl11] O. Erler. Aufmerksamkeitsgetriebene Objektexploration für autonom lernende Roboter. (4341785), 2011. (Cited on page 44)

[FRSS07] G. Froimovich, E. Rivlin, I. Shimshoni, O. Soldea. Efficient search and verification for function based classification from real range images, 2007. (Cited on page 11)

[GESB95] K. Green, D. Eggert, L. Stark, K. Bowyer. Generic Recognition of Articulated Objects Through Reasoning About Potential Function. *Comput. Vis. Image Underst.*, 62(2):177–193, 1995. doi:10.1006/cviu.1995.1049. URL http://dx.doi.org/10.1006/cviu.1995.1049. (Cited on page 11)

[Gib77] J. J. Gibson. *The Theory of Affordances*. Lawrence Erlbaum, 1977. (Cited on page 21)

[JH99] A. E. Johnson, M. Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(5):433–449, 1999. (Cited on page 15)

[KDPB09] D. Kraft, R. Detry, N. Pugeault, E. Başeski. Learning objects and grasp affordances through autonomous exploration. *Computer Vision ...*, pp. 1–10, 2009. URL http://link.springer.com/chapter/10.1007/978-3-642-04667-4_24. (Cited on page 14)

[KE12] K. Khoshelham, S. O. Elberink. Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications. *Sensors*, 12(2):1437–1454, 2012. doi: 10.3390/s120201437. URL http://www.mdpi.com/1424-8220/12/2/1437. (Cited on page 23)

[Moo91] A. Moore. A tutorial on kd-trees. Extract from PhD Thesis, 1991. Available from http://www.cs.cmu.edu/simawm/papers.html. (Cited on page 18)

[PSR08] M. Pechuk, O. Soldea, E. Rivlin. Learning function-based object classification from 3D imagery. *Computer Vision and Image Understanding*, 110(2):173–191, 2008. doi:10.1016/j.cviu.2007.06.002. URL http://linkinghub.elsevier.com/retrieve/pii/S1077314207000884. (Cited on page 11)

[RBTH10] R. B. Rusu, G. Bradski, R. Thibaux, J. Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 2155–2162. IEEE, 2010. (Cited on page 15)

[RC11] R. B. Rusu, S. Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China, 2011. (Cited on page 24)

[Rus10] R. B. Rusu. Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments. *KI - Künstliche Intelligenz*, 24(4):345–348, 2010. doi:10.1007/s13218-010-0059-6. URL http://link.springer.com/10.1007/s13218-010-0059-6. (Cited on pages 24 and 31)

[SB91] L. Stark, K. Bowyer. Achieving generalized object recognition through reasoning about association of function to structure. *Pattern Analysis and Machine Intelligence, ...*, 1991. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=99242. (Cited on page 11)

[SCD+07] E. Sahin, M. Cakmak, M. R. Dogar, E. Ugur, G. Ucoluk. To Afford or Not to Afford: A New Formalization of Affordances Toward Affordance-Based Robot Control. *Adaptive Behavior*, 15(4):447–472, 2007. doi:10.1177/1059712307084689. URL http://adb.sagepub.com/cgi/doi/10.1177/1059712307084689. (Cited on pages 6, 21 and 22)

[Sto] a. Stoytchev. Behavior-Grounded Representation of Tool Affordances. *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 3060–3065. doi:10.1109/ROBOT.2005.1570580. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1570580. (Cited on page 11)

[SV01] D. Saupe, D. V. Vranić. *3D model retrieval with spherical harmonics and moments*. Springer, 2001. (Cited on page 15)

[TSDS10] F. Tombari, S. Salti, L. Di Stefano. Unique signatures of histograms for local surface description. In *Computer Vision–ECCV 2010*, pp. 356–369. Springer, 2010. (Cited on page 15)

[YLJ13] J. Yang, H. Li, Y. Jia. Go-ICP: Solving 3D Registration Efficiently and Globally Optimally. *2013 IEEE International Conference on Computer Vision*, pp. 1457–1464, 2013. doi:10.1109/ICCV.2013.184. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6751291. (Cited on page 43)

All links were last followed on May 7, 2014.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature