Visual feature learning for interactive segmentation

Bachelorarbeit
Sebastian Christoph Schöngen

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich 20 - Informatik
TU Darmstadt

Bachelorarbeit von

Sebastian Christoph Schöngen

am Fachbereich Informatik
der Technischen Universität Darmstadt.

Gutachter: Prof. Jan Peters
Betreuer: Herke van Hoof

Darmstadt 17. November 2013

Ehrenwörtliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Alle Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen sind, wurden unter Angabe der Quelle kenntlich gemacht. Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegt.

Darmstadt, den 17.11.2013

S. C. Schöngen

abstract A challenge for modern robots is to act in unknown and possibly unstructured environments. For example, rescue robots clearing cave-ins or household helpers finding the salt in a really cluttered kitchen. This means that the robot must be able to distinguish objects and recognize their shape maybe without ever having seen the objects before. Therefore many current segmentation methods cannot be applied. In this thesis we propose a new part based algorithm that is used to segment scenes using visual features in 3D space without any prior knowledge about the scene. We then use machine learning algorithms to learn the effectiveness of the different features and to create a probabilistic classifier that differentiates between good and bad segmentations. This classifier is then integrated into an interactive approach to probabilistic segmentation in order to improve its results and decrease the number of interactions needed in order to reach a satisfying segmentation.

Contents

1 Motivation

It would be great, if robots could help humans in their everyday lives. However, most robots as they exist today are working in huge factory halls and mostly perform relatively simple and repetitive tasks. There are many reasons for this, e.g. that equipping a robot with exactly the tools and knowledge for one specific task is far easier and much cheaper than creating a robot that can interact with various different environments. But the main reason for this is, that it is usually easier and more feasible to program a robot with a fixed movement trajectory than giving it the ability to act accordingly to its surroundings.

In order to tackle this problem, one prerequisite must be met: The robot needs a way to perceive and adapt to its surroundings in an autonomous way. This includes not only being able to recognize known objects, or to find the own position in a known environment. The robot also needs to be able to perceive unknown objects and generate a model of their appearance, position and rotation in space using the data it receives from its input sensors. By being able to autonomously generate a model of the environment and everything within, a robot can gain a lot of new possibilities.

Instead of following a fixed trajectory in order to interact with objects, a robot could now adapt its movements to position, rotation and shape of the object by using its sensor data. It could even perceive and model previously unseen objects or environments and adapt whatever task it currently is performing to its surroundings. For example, a possible advanced application would be to perform an exploration task of a potentially dangerous area, clearing obstacles encountered while gathering information about the terrain and its layout.

A method that can be used for learning information about the environment is called image segmentation. This method uses visual input that can be recorded online by equipping a robot with video or photo cameras. One can then apply machine learning techniques in order to enable the robot to decide independently from any exterior controller which parts of the image belong to the same object and which do not belong together. This information can then also be used to determine the shape of the seen objects and what kinds of interactions are possible. Additionally, the robot can explore its surroundings by interaction and evaluate the movements of image parts in its field of view in order to gain information about the segmentation of the scene.

In this thesis we will introduce a new method of scene segmentation that combines the ideas of various approaches and evaluate how well these perform compared to each other. It takes data about the movement of the image parts between the images taken before and after an interaction as well as information about the visual features like color, brightness and others into account which are derived from 3D image data. The focus of this thesis will be on the learning and the evaluation of the visual features, which will serve as a prior in the computation of the probabilities via movement data.

In Chapter 2 we will analyze related work in the field of segmentation and object recognition and compare them to our approach. We will also introduce the previously mentioned method of interactive image segmentation in more detail. Chapter 3 will cover how our algorithm learns to separate good and bad segmentations using visual features and how this is used to improve the performance of the interactive method. Finally in Chapter 4 we will describe the hardware used to implement our method and discuss the results it provided. We will conclude with a brief summary and possibilities for future work.

2 Related Work

In this chapter we will describe and discuss related research that has been done in the field of image segmentation and scene recognition. Afterwards we will introduce a method of movement based segmentation, that will be the starting point of this thesis.

2.1 Related work in Scene Recognition

Since robots in unknown environments are unlikely to know the objects they are going to encounter, traditional methods for object recognition [14] will not be sufficient. This is because they need object models previously created offline either by training or by crafting them by hand [11]. For example, by using CAD software or 3D scanning. The cited methods work by creating a 3D view of the scene by combining several 2D images from various directions. The created 3D scene is then compared to a database of 3D models in order to find a correspondence between a part of the perceived scene and one of the database entries. The scene can then be reconstructed accurately using the memorized models. While this can lead to a very detailed model of the scene, because it is improved by information from the database, the problem with these methods is that the database is limited and can only contain a fixed and finite set of objects to recognize. Instead, we need to be able to model new objects the first time we encounter them and recognize them at later points in time from data observed online.

Another problem is, that for many methods of autonomously modeling novel objects, the object in question needs to be encountered in isolation, preferably in front of a neutral background as in the work of e.g. Collet et al. [6]. Their algorithm can only focus on one object at a time and accurately recognize it and determine its 6 degrees of freedom position in space by comparing it to a model learned offline. However, this method is not a good approach when dealing with unknown environments. The main problem here is that at the beginning of the segmentation process there exists no knowledge about where objects are and how far they expand - or put differently: where one object ends and the next one starts. A robot might have to find an object from a cluttered table or interact with an object in a natural environment like a forest. The object we want to observe might even be partially occluded from all possible viewing directions. Therefore, we have to be able to find and segment objects in arbitrarily cluttered scenes and, if necessary, even when only small parts of an object can be seen we want to at least be able to identify it as an independent object.

Since objects may be very close to each other and have similar appearances, e.g. two similar boxes next to each other, vision alone might not suffice in many cases for finding the correct segmentation. As an alternative, an interactive approach can be used [9, 12, 16]. The robot interacts with the scene by pushing, or in an ideal case grasping, part of the scene. By comparing the scene before and after the interaction, feature tracking algorithms like SIFT (Scale Invariant Feature Transform [13]) can be used to determine which parts of the scene moved and which did not. This information is then used to find the most probable segmentation of the scene. However, interactive methods usually need more time and more input to come to a result because often several steps of interaction have to be made and before any interactions with the environment, there is no data to use at all. In order to use the explorative abilities of an interactive approach without increasing the time needed to come to a result, visual data can be combined with interactive movement tracking. This is done in the work of e.g. Katz et al. [12]. In their work they combine several hand tuned indicators about a scene's segmentation. Each indicator assigns each pair of parts a weight between 0 and 1, which are then multiplied. The result is a graph which is then split into highly connected subgraphs representing rigid bodies in the scene. One drawback of their method is that they only segment a single object composed of several rigid bodies. This means that one can only focus on one object of interest at a time and cannot segment an entire scene of multiple independent objects. Another issue is that the indicators often assign hard zeros to pairs of parts by handcrafted criteria, rendering the other indicators irrelevant. Instead we will be using a probabilistic combination of movement and visual data, both learned from machine learning methods combined into a joint probability. This visual input supplies information about the segmentation of the observed environment even before the first interaction was performed.

Most of the methods handling segmentation by visual features only work with 2D image data [4,7,8,15]. The images are then either getting segmented pixel wise or are grouped into superpixels. While the results for recognizing objects and segmenting scenes can be quite satisfying when using visual features, interactive approaches have difficulty here, because interaction with the scene is not easily done without depth perception. In order to get a better grasp of the scene's spatial depth and placement, we will be using 3D data instead as in e.g. [3, 12].

Most methods of object recognition and scene segmentation process their input and deliver one final result e.g. [12, 15] which is usually the one with the highest probability of being correct, or the best one according to some scoring system. In these methods, possible segmentations are sampled and then compared, with the best solution being returned. This process of elimination has one problem: Since the human world is full of irregularities and a robot's sensory system is always subject to noise, it will not suffice to employ a method that only delivers one best guess. Especially not when we use an explorative interactive approach, that will probably discover new information over time so that we have to revoke the most probable candidate so far. The second problem is, that when combining different segmentation methods, we have no way of comparing them, if we have no knowledge about how certain the individual methods are about their results. To address this, we will have to estimate a probability distribution over all possible segmentations. This will give us the possibility to introduce new information encountered after some more interactions, or information from another source like the visual features into our current results in a principled way. It also allows us to choose the interaction with the highest expected information gain each iteration.

One common problem - not only for segmentation, but machine learning in general - is that "the objective function being optimized is typically driven by the designer's intuition or computational convenience." [15]. This means, that often both in the calculation and the combination of features there are a lot of hyperparameters that have to be tuned by hand [12] in order to get optimal or near optimal results. By using real data from a manually segmented database of images, density estimation can give us information about how features would be distributed in good and bad segmentations. Ren et al. [15] actually measure the significance and meaning of their features from real data by calculating the likelihood of a feature being calculated from a good segmentation using these derived probability distributions. This makes the evaluation and learning of visual features less susceptible to errors that come from wrongly tuned parameters in the algorithm and increases its ability to generalize to new data. We will use this approach in the calculation of most of our features and compare the results with an implementation driven by our intuition.

The simplest approach to a segmentation would be, to decide which pixel - or in our case image point - belongs to which object [7]. Using the mean shift algorithm, the pixels are then clustered into similarly colored groups. These clusters are then combined into larger objects, creating a segmentation of the scene. This is however not easily done with 3D data, since you do not have image points filling the entire image space, but are usually limited to very sparse data at object boundaries. While there are other grouping methods that could be used even in 3D space, this can then still be interpreted as a part based approach. Another benefit of grouping image points together is that it grants access to the usage of new features, e.g., the evaluation of histograms over the individual color values or the comparison between face normals, which cannot be computed for single pixels. Therefore, we will search for easily trackable key points in the image and create parts around them so that every image point is assigned to at least one part [16].

2.2 Probabilistic Interactive Segmentation for Robots in Cluttered Environments

Under the criteria of being interactive and probabilistic, [16] developed a segmentation algorithm that is implemented using a robot and works based on the evaluation of movement data. The robot has a 3D camera and an end effector in the form of a rod attached to it. It is in a fixed position relative to a table. The robot interacts with objects on the table by pushing them with its extended reach. Then it can observe the scene by taking images from three fixed positions and combining them into point cloud data (see Fig: 2.1).

By tracking local key points the differences before and after an interaction are examined. These key points are detected and described using Scale Invariant Feature Transform (SIFT) [13]. It is determined, which image parts have moved together, independently or not at all. The system can then compute a probability for each possible segmentation that represents how probable that specific combination of parts is, given the observed movement data.

Since the number of possible segmentations grows exponentially with the number of parts, we will not be able to compute the probability for every possibility in a reasonable amount of time. Instead the expectation over the probability distribution is approximated by utilizing Gibbs Sampling [5]. As we know, not all possible segmentations are equally likely. Both very separated and (nearly) completely connected segmentations are very unlikely in real world examples. A prior can be computed for every segmentation, independently of any observed data. This prior is defined as the probability of this specific segmentation being sampled by the Chinese Restaurant Process [1]. This is included in the calculation of a segmentation's probability and can be understood as a bias towards more natural segmentations.

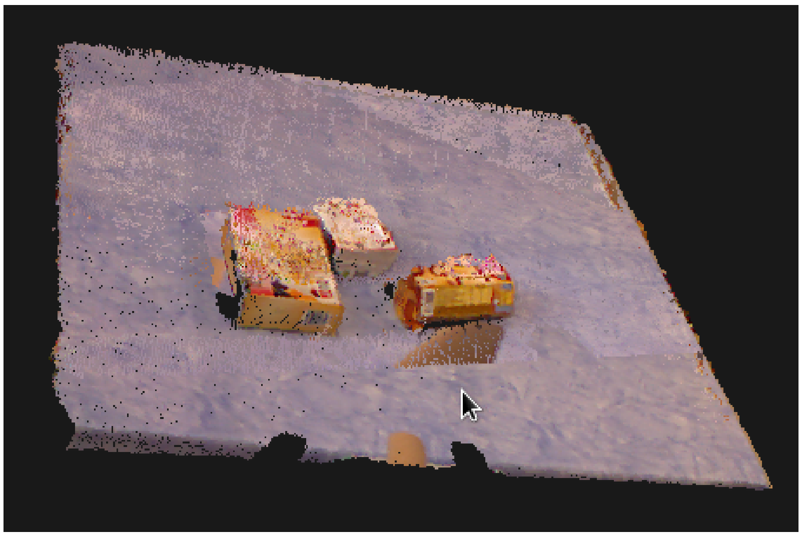


Figure 2.1: Point cloud data recorded by the robot. For every scene, three shots from different directions are taken and combined into one point cloud.

3 Interactive Probabilistic Segmentation by Learning Visual Features

In this chapter, we will introduce a method for segmenting a three dimensional image into objects based on visual features and explain how it is combined with the interactive movement based approach. We will first introduce our data model, then we will explain about the starting point of our research. We will then talk in detail about our learning algorithm, how features are extracted from image data and how we generate our training examples.

3.1 Preprocessing and Data Model

The robot's view on the scene is provided by a 3D camera. For every observation, 3D point cloud data is taken from three different angles and then combined into one single point cloud. Segmenting the scene on a point basis is impractical for two reasons. On the one hand the number of points in a scene is in the magnitude of 10^5 which would increase the computational effort to unreasonable levels and on the other hand it is not possible to compute more complex features like color histograms or face normals for single points.

Instead, we use a part based approach. These parts (also called centers subsequently) are represented as spheres around points initialized on a grid over the 3D view of the scene in a fixed radius of 6 centimeters. This means, that these regions can overlap and therefore points can temporarily belong to more than one center. We search for local key points in our view of the scene. These must fulfill two properties: The first one is, that the key points must be easy to reidentify in later frames after interaction. The other one is that they must cover all image points that may belong to an object. So if later on new objects or parts of objects are discovered, e.g. because they were previously occluded, they will be covered by creating new centers at the position of an image point that does not belong to any center yet. Key point detection and description is done using the Scale Invariant Feature Transform (SIFT) algorithm [13].

While this method introduces some inaccuracies because the boundaries of centers might intersect the boundaries of more than one object, it is still a good enough approximation for our purposes especially because we have the possibility of moving objects apart since we are using an interactive approach.

3.2 Interactive Segmentation based on Movement Tracking

In the work of van Hoof et al. [16] a probability distribution over possible segmentations is learned based on interactions and observations of movement. It is an iterative process of alternatingly estimating the probability distribution over segmentations $p(S|D_{\text{move}})$ given the movement data observed so far, and choosing the action a_{t+1} with the highest expected information gain based on the distribution incorporating uncertainty.

The intention is to calculate $p(S|D_{\text{move}})$. The vector $S = [s_1, ..., s_N]$ represents a segmentation with s_i being the assignment of the i-th part to an object. This means that the segmentation $[1, 1, 2]$ is equivalent to the segmentation $[2, 2, 1]$. The observed movement data is represented like this: $D_{\text{move}} = \{(a_t, \vec{o}_t) | t \in 0, 1, ..., T\}$. a_t is the action chosen in iteration t and $\vec{o}_t$ is a vector with $\vec{o}_t[j] = 1$ for every part j that moved and 0 for every part that did not.

Since the number of possible segmentations grows exponentially with the number of centers, calculating $p(S|D_{\text{move}})$ for every single possible segmentation S is not feasible outside of academic examples. Instead, an expectation over $p(S|D_{\text{move}})$ is approximated using Gibbs sampling [5], a Markov chain Monte Carlo method [10]. That means, that we iteratively sample the assignment of the i-th part s_i from $p(s_i|s_{\setminus i}, D_{\text{move}})$ based on the assignment of all other parts $s_{\setminus i}$ for every i. This procedure gives us independent samples from $p(S|D_{\text{move}})$ for sufficiently large numbers of iterations. For the sampling, we need $p(s_i|s_{\setminus i}, D_{\text{move}}) \propto p(D_{\text{move}}|s_i, s_{\setminus i})p(s_i, s_{\setminus i})$. The prior $p(s_i, s_{\setminus i})$ must be able to be computed without any knowledge of the scene and without knowing the number of objects. An appropriate prior distribution over assigning n parts to a previously unknown number of objects is the Chinese Restaurant Process [1]. As the focus of this thesis is on visual feature learning, we refer the reader to [16] for further details on the computation of $p(D_{\text{move}}|s_i, s_{\setminus i})$ and the choice of the next action a_{t+1}.

3.3 Visual Feature Learning

Using only movement data can suffice to reach a sufficient segmentation, but incorporating prior knowledge about the segmentation can increase the speed of the process so less actions are needed in order to reach a satisfying result. In order to do this, we want to introduce the evaluation of visual features into the computation of the prior. Previously the probability distribution was computed as

$$p(S|D_{\text{move}}) \propto p(D_{\text{move}}|S)p(S).$$

By incorporating the visual features into the calculation we now have:

$$p(S|D_{\text{move}}, D_{\text{vis}}) \propto p(D_{\text{move}}|S, D_{\text{vis}})p(S|D_{\text{vis}}).$$

Under the assumption that D_{move} and D_{vis} are conditionally independent given S we can say that $p(D_{\text{move}}|S, D_{\text{vis}}) = p(D_{\text{move}}|S)$ resulting in

$$p(S|D_{\text{move}}, D_{\text{vis}}) \propto p(D_{\text{move}}|S)p(S|D_{\text{vis}}).$$

Accordingly, we want to compute the prior $p(S|D_{\text{vis}})$ over segmentations S given the observed visual data D_{vis}. We do this by training a logistic regression classifier [2] over good and bad segmentations. Logistic regression is a two class classification method. It applies the sigmoid function, that transforms any real number into a probability between 0 and 1, on a linear function of the features. This is a similar approach to the work of Ren et al. [15]. The difference between our approaches is, that Ren et al. train a logistic regression classifier over good and bad segments s and use these to create a score for each segmentation $f(S) = \sum_i p(s_i = good)$. We on the other hand want to get a probability distribution over segmentations. This means, that we create a feature vector $\phi(S)$ for every segmentation S and then calculate

$$p(S = good|D_{\text{vis}}) = \sigma\left(\vec{w}^T \phi(S) + w_0\right)$$

where σ is the sigmoid function $\sigma(a) = \frac{1}{1-\exp(-a)}$. The sigmoid function converges to 1 for $a \to +\infty$ and to 0 for $a \to -\infty$ (see Fig: 3.1). The value of $\sigma\left(\vec{w}^T \phi(S) + w_0\right)$ represents the probability of the segmentation S being a good segmentation. In order to get a probability distribution we can normalize over all segmentations S to get a normalization factor δ. The prior is then defined as a combination of the normalized classifier and the Chinese Restaurant Process

$$p(S|D_{\text{vis}}) = \delta p(S = good|D_{\text{vis}})p_{\text{crp}}(S).$$

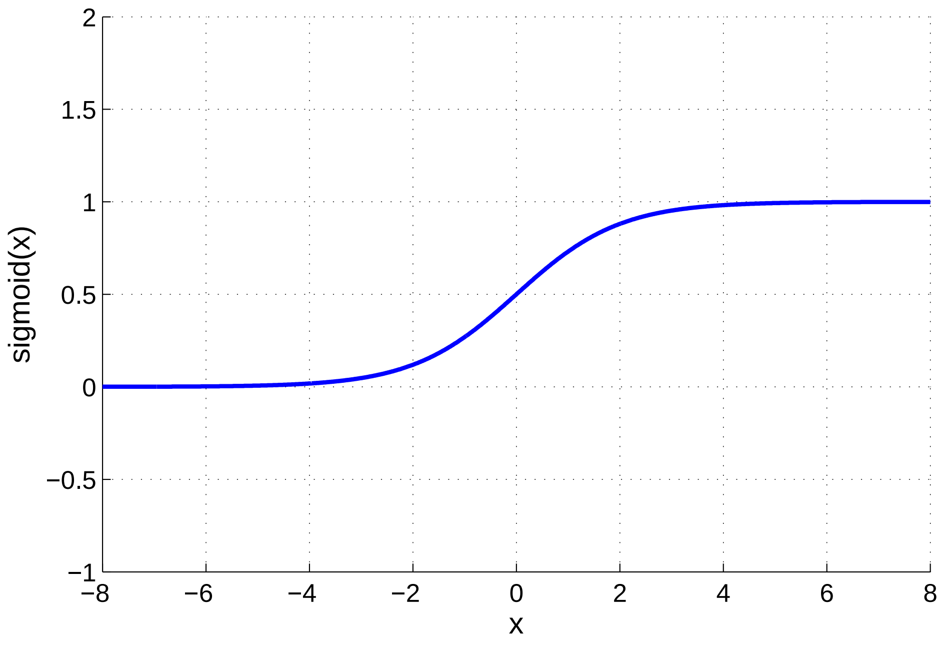


Figure 3.1: Exemplary plot of the sigmoid function. It converges towards 0 and 1 when going towards negative and positive infinity. It is used for the creation of two class classifiers using logistic regression (see [2] for more information).

3.3.1 Creating the Feature Vectors

In order to evaluate the visual data observed, we need to choose a representation of the visual features for a given segmentation S. Every segmentation S is represented as an assignment of centers c_i to objects or segments s_j. We can compute a feature vector $\phi(s_j)$ for each segment in a segmentation independently, based on the 3D image data and the segmentation's assignments of image points to the individual objects. Then we create a feature vector for each segmentation that will be composed of the sum of the features of the individual segments s_i:

$$\phi(S) = \sum_i \phi(s_i)$$

In the following we will introduce the features we used, explain their intuition and how we compute them.

Position

The position of the centers of a segment gives general information about the connectedness of the segment. The value of the position feature of a segment s is computed by calculating the mean of the distances between all centers:

$$\frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} ||c_i - c_j||_2$$

where c_i is the position vector of the i-th part and N is the number of centers in the segment.

Weighted Position

The weighted position gives additional information about where most of the points belonging to a part are. First we calculate the weighted position of each part which is the average position of a part's image points:

$$\hat{c}_i = \frac{1}{M} \sum_{j=1}^{M} p_j$$

where p_j is the position of a point j that belongs to part i. Then we can compute the mean of distances analogously to the position feature:

$$\frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} ||\hat{c}_i - \hat{c}_j||_2.$$

Color

We assume, that centers with similar colors are more likely to belong to the same object. In order to avoid hand tuning this feature, we take a statistical approach to rate color similarities as seen in the work of Ren et al. [15]. The color data for each part is stored as a three dimensional histogram with 5 bins each for the red, green and blue dimension.

Similarity between two histograms is defined as the sum of the squared error between corresponding bins:

$$\text{dist}(A, B) = \sum_{i,j,k=1}^{5} (A_{ijk} - B_{ijk})^2.$$

In order to acquire knowledge about how this distance is distributed, we created two lists of pairs of centers L_{same} and L_{diff} from correct handcrafted segmentations of real data. L_{same} contains all pairs of parts, whose elements are in the same segment and L_{diff} contains all those who belong to the same scene but to different segments. We then use density estimation to find p_{same} and p_{diff}, the distributions over the distances between these pairs. For this we assume them to be Gaussian distributed and use maximum likelihood estimators to find the parameters.

Now we can compute the color feature for a segment s using the formula

$$\text{color}(s) = \sum_{i=1}^{N} \log \frac{p_{\text{same}}(\text{dist}(c_i, s))}{p_{\text{diff}}(\text{dist}(c_i, s))},$$

where dist(c_i, s) is the distance between the color histogram of the entire segment s and its part c_i.

Brightness, Hue and Saturation

Brightness, hue and saturation are, compared to the color feature, three different views on the same information. Each of them is treated as a separate feature that is computed in an identical way to the computation of the color feature. The only difference is, that each of the features is stored in a one dimensional histograms with 10 bins.

Number of segments

Additionally, we consider the number of segments in a segmentation. In order to create a feature that is exactly the number of segments for each segmentation, the value of this feature is 1 for every segment's feature vector. This helps avoiding over- and underseparation.

3.3.2 Training the classifier

Now that we can obtain a feature vector $\phi(S)$ for every possible segmentation S, given the observed visual data D_{vis}, we can use this to train a classifier $p(S|D_{\text{vis}})$. For this we use logistic regression, which leads to the formula

$$p(S|D_{\text{vis}}) = \sigma\left(\vec{w}^T\phi(S) + w_0\right).$$

In order to acquire $\vec{w}$, we use the Iterative Reweighted Least Squares algorithm (IRLS) [2]. We now want to find the parameter vector $\vec{w}$ that optimally describes the training data. We do this by maximizing the likelihood of our parameters given the training data $p(\vec{t}|\vec{w})$. We do this by minimizing the cross-entropy error function

$$E(\vec{w}) = -\log p(\vec{t}|\vec{w}) = -\sum_{n=1}^{N}(t_n \log y_n + (1-t_n)\log(1-t_n))$$

where t_n is the target label for the training example S_n and $y_n = \sigma(\vec{w}^T\phi(S_n))$ is the probability $p(S_n = good|D_{\text{vis}})$ based on the current $\vec{w}$. The update rule for $\vec{w}$ by the *Newton-Raphson* iterative optimization [17] scheme takes the form:

$$\vec{w}_{t+1} = \vec{w}_t - \mathbf{H}^{-}1\nabla E(\vec{w}_t).$$

The derivative $\nabla E(\vec{w})$ and the Hessian $\mathbf{H} = \nabla\nabla E(\vec{w})$ of the error function are given by

$$\nabla E(\vec{w}) = \sum n = 1^N(y_n - t_n)\phi(S_n) = \Phi^T(\vec{y} - \vec{t})$$

$$\mathbf{H} = \nabla\nabla E(\vec{w}) = \sum n = 1^N y_n(1-y_n)\phi(S_n)\phi(S_n)^T = \Phi^T R\Phi$$

where Φ is the matrix of feature vectors and R is a diagonal matrix with $R_{nn} = y_n(1-y_n)$. This finally gives us the update rule

$$\vec{w}_{t+1} = \vec{w}_t - (\Phi^T R\Phi)^{-1}\Phi^T(\vec{y} - \vec{t}).$$

This algorithm poses some difficulties. On the first hand we have to be careful when initializing $\vec{w}_0$ because initializing it too high will result in a vector $\vec{y}$ with values extremely close to 1 or 0, which will make the algorithm numerically unstable. We chose to initialize our parameters as random values drawn from a normal distribution multiplied with 10^{-3}.

On the other hand, the training data might be linearly separable in feature space. This will cause the parameter vector to diverge towards $\pm\infty$, which means that we are overfitting to our data. Any new data point we encounter that lies between the currently known good and bad examples will then still be predicted to be either good or bad with near 100% certainty instead of being represented as a value close to 0.5. In other words, we cannot generalize well on new data. To counter this problem, we introduce regularization into the IRLS algorithm. The intuition behind this is to put a Gaussian prior $p(\vec{w})$ on the parameter vector. Integrating this prior into the cross-entropy error function gives us the new posterior error function:

$$E(\vec{w}) = -\log(p(\vec{t}|\vec{w})\mathcal{N}(\vec{w}|0,\alpha^{-1}I)) = -\sum_{n=1}^{N}(t_n \log y_n + (1-t_n)\log(1-t_n)) - \tfrac{\alpha}{2}\vec{w}^T\vec{w}.$$

Deriving for $\nabla E(\vec{w})$ and $\nabla\nabla E(\vec{w})$ we obtain the new update rule

$$\vec{w}_{t+1} = \vec{w}_t - (\Phi^T R \Phi + \lambda I)^{-1}(\Phi^T(\vec{y} - \vec{t}) + \lambda \vec{w}_t)$$

where λ is a constant factor based on the covariance of the Gaussian prior. The regularization parameter λ has to be optimized by experimenting. If it is set too high, the original values of the features will become irrelevant in comparison and if it is set too low, it will not have any effect at all and will not prevent overfitting (see Fig: 3.2, 3.3).

3.3.3 Generating Training Data

In order to be able to train our classifier, we need both positive and negative training examples. Meaning examples of correct segmentations which will receive a target label t of 1 and examples of incorrect segmentations labeled with a 0.

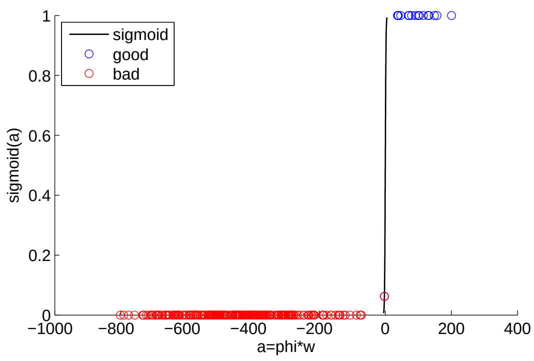
Positive examples have to be created by hand. Using the robot and the mounted 3D camera we take images of a scene to be segmented. This visual data is then divided into parts by creating local key points as described before. We then have to assign the parts to objects by hand, creating a segmentation S_n. From this segmentation and the point cloud data, we can then create one positive training example Φ_n.

Negative training examples are created in an analogous way. For every positive training example, using the same visual data, we create several negative ones simply by grouping the parts in a different way.

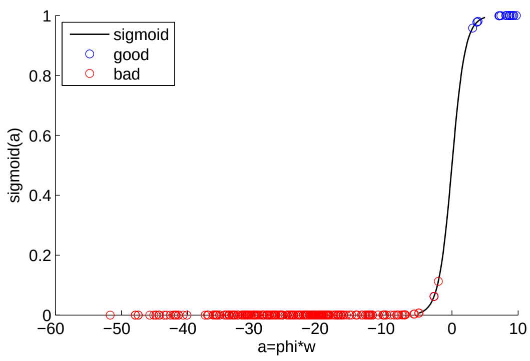
The first intuitive approach was to randomly reassign parts to objects. We do this by creating one randomly selected permutation of the correct handcrafted segmentation vector. E.g. a segmentation vector $[2,2,1,2]$ might become $[2,2,2,1]$, $[1,2,2,2]$ or $[2,1,2,2]$. For every positive example we create 5 negative examples using this method.

In addition we want to be able to compare segmentations that are more natural and also more likely to be considered during the sampling of segmentations when used together with the interactive approach. Therefore, we create additional bad examples by sampling from the Chinese Restaurant Process, that is used to define the prior on segmentations. Using the visual data and one good segmentation, we create 10 bad segmentations by replacing the assignment vector by one drawn from the CRP.

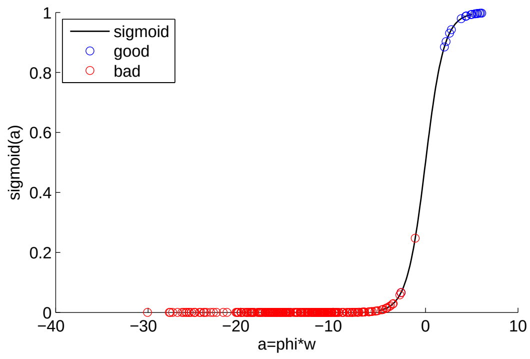
One additional segmentation to be considered is the completely segmented one. This means a segmentation where every part belongs to a separate object, e.g. $[1,2,3,4,5]$. Considering this segmentation as a bad example is of importance, because it is the one the interactive approach initially assumes before evaluating any visual or movement data and before calculating any prior.



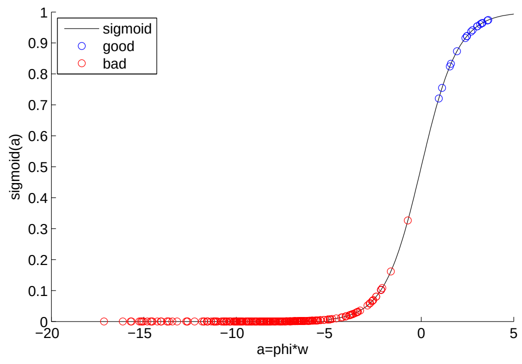
(a) without regularization



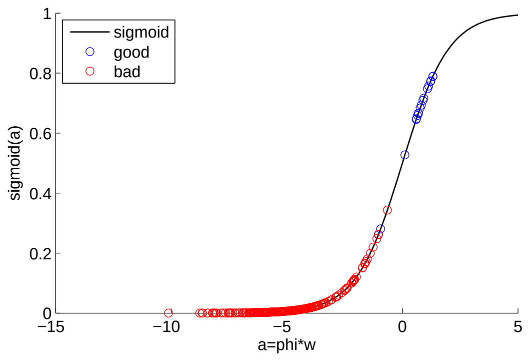
(b) w = 0.01



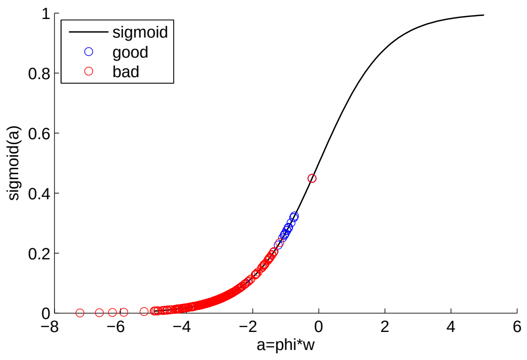
(c) w = 0.1



(d) w = 1

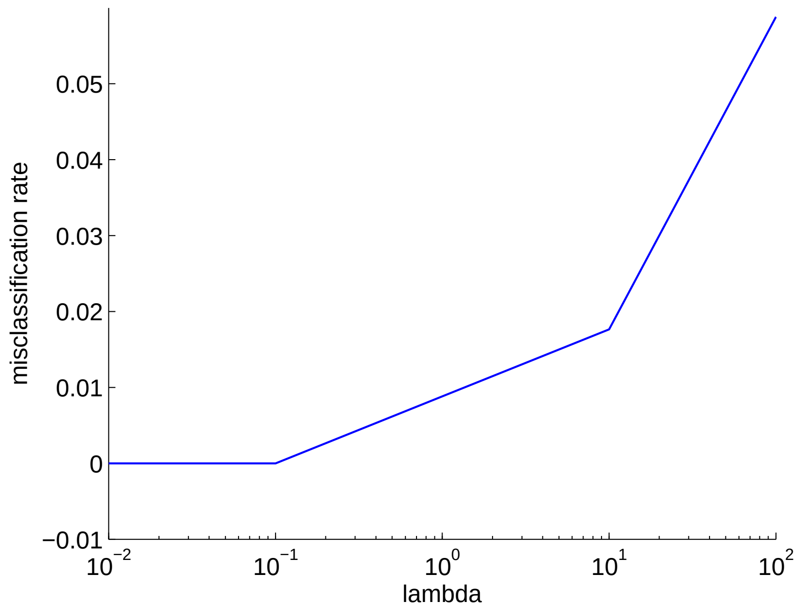


(e) w = 10



(f) w = 100

Figure 3.2: Classification results for the training examples when trained without regularization and with a regularization parameter λ of 0.01, 0.1, 1, 10 and 100 respectively. As can be seen here, λ needs to be tuned. If it is set too low, overfitting occurs, which means that we do not generalize well to new training examples. On the other side, if it is set too high it renders the original features irrelevant since the implied prior becomes too strong.



λ	misclassification rate
0	0.0088
0.01	0
0.1	0
1	0.0088
10	0.0176
100	0.0588

Figure 3.3: Average misclassification rate of the training examples for different regularization parameters. We used a dataset of 340 samples that was split into 340 segments of one element each for leave-one-out cross validation.

4 Analysis

In this chapter, we will draw the conclusion to this thesis. We will first present the results we obtained, both for independently employing the visual feature learning and for using it together with the interactive approach. Finally we will draw a conclusion about what we accomplished and talk about possible future work.

4.1 Hardware used

The robot used for the evaluation of these methods is a Mitsubishi PA-10 robot. It is described as a "portable general purpose intelligent arm" in the manual[1]. It is a robotic arm with 7 degrees of freedom, that is installed to fixed position on the floor. In our setup used for interacting with the environment and taking visual data it is additionally equipped with an end effector in form of a rod and a Microsoft Kinect™3D-camera (See Fig: 4.1).

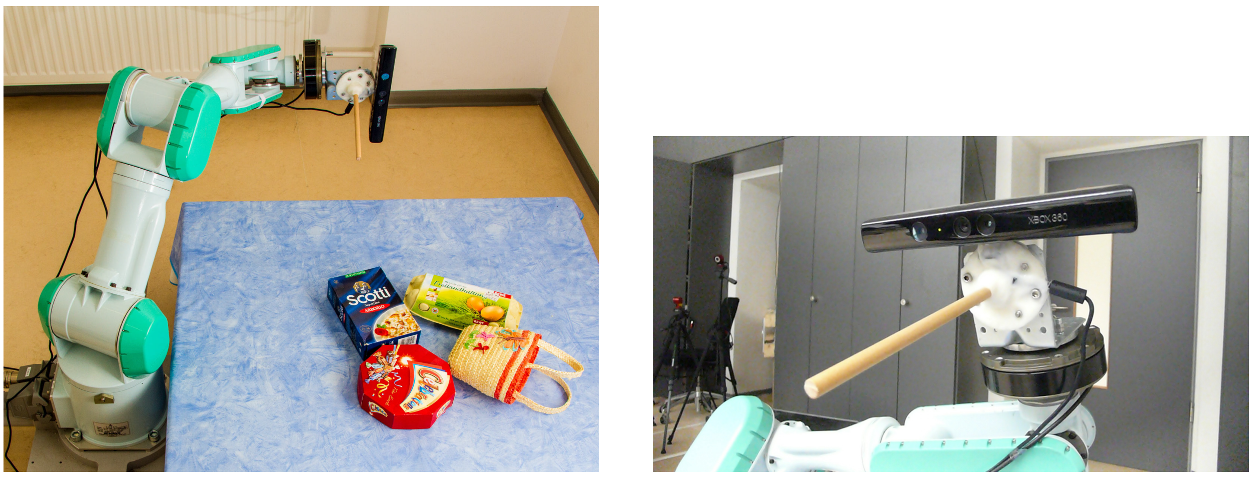
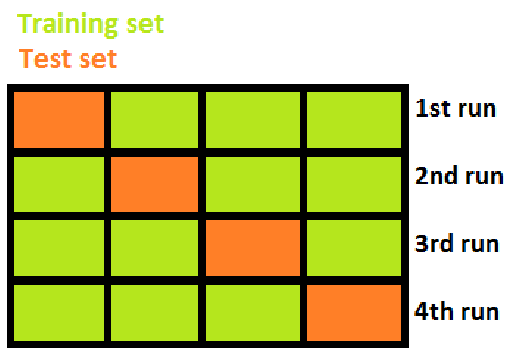


Figure 4.1: The Mitsubishi portable general purpose intelligent arm, PA-10. A 7 degrees of freedom robot arm. Shown while taking an image of a scene to be segmented. The picture on the right shows a close-up view of the robot's "hand" with the end effector and the 3D-camera used, a Microsoft Kinect™

4.2 Results of the classifier alone

In order to evaluate the functionality of a machine learning algorithm, common practice is to learn on your set of training data and to evaluate your learned parameters on a disjoint set of test data. However, real life applications often have the problem that data is not readily available and expensive and time consuming to produce. For such cases one can use cross validation [2](see also Fig: 4.2a). For evaluating how well the IRLS algorithm performed on the data we had, we employed 10-fold cross-validation. We split up the data D into 10 approximately equally large parts D_i and for every i we used all $D_{j \neq i}$ as training data and used D_i as test data for evaluation. We ended up with an average misclassification rate of 0.0107 (see Fig: 4.2b).

[1] http://code.google.com/p/drics/downloads/detail?name=General%20Operating%20Manual%20%287-axis%29.pdf



(a) When using S-fold cross-validation, the available data is split into S parts. $1/S$ of the available data is then excluded from the learning process for each run. This data is then used as test data. Depicted here is the example of a 4-fold cross-validation.

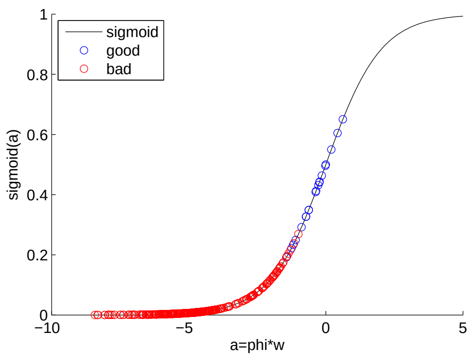
run #	misclassification rate
1	0
2	0
3	0
4	0.0714
5	0
6	0
7	0
8	0.0357
9	0
10	0

(b) Using 10-fold cross-validation, we deemed our algorithm strong enough to distinguish between good and bad segmentations based on vision alone.

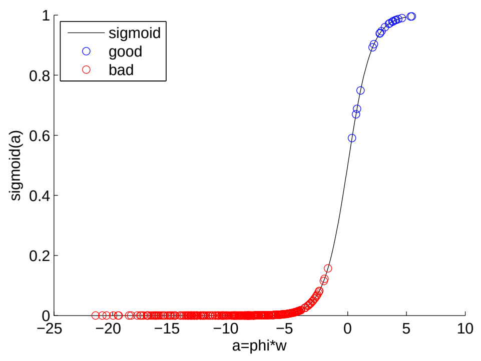
Figure 4.2: Brief exemplary overview over cross-validation in general and the results on our training set.

4.2.1 Evaluation of Feature Computation

We want to evaluate how useful our implementation of the color based features is. Therefore, we compare the results of the classifier, when learning with the visual feature vectors computed in the probabilistic way to the classifier we receive when learning on features in a more traditional way. Instead of evaluating the log-likelihood of our features for each segment, we simply take the sum over the histogram distances. Then we optimized the regularization parameter λ for the new set of features. As can be seen in Fig: 4.3, there is a visible improvement on the results when using the log-likelihood features.



(a) Results of the classifier when learning with the non-probabilistic feature computation.



(b) Results of the classifier when learning with the probabilistic feature computation.

Figure 4.3: Comparison between the two ways of computing features.

4.3 Results when used in the Interface

It is good to know that the IRLS algorithm for creating a classifier works well with the features we chose. However, our initial intention was to improve the results of the interactive approach by van Hoof et al. [16] by using visual features.

The first step was to verify that we will not get stuck in a local minimum during the evaluation of the markov chain. In order to do this, we tested the combination of the visual and the interactive approach. For several recorded examples of visual and movement data, we initialized the markov chain with the completely separated segmentation, which is the standard behavior when using the algorithm on new data. Then we used the same data but initialized the markov chain with the correct segmentation. For reducing the variance in the results that occur due to this being a stochastic process, we ran every test 10 times and averaged the results. Since comparing probability distributions over segmentations is difficult we compared the average certainty of our expected best guess, which should show a significant difference if we end up in a bad local minimum. In both cases, we received approximately the same results. Depending on the data example, especially the number of interactions taken into account, we received results between 0.4 and 0.7, but the difference for one data example was always lower than 0.001.

4.3.1 Comparison of the Interactive Approach with and without Visual Features

For the evaluation of the impact of the visual prior on the results of the interactive approach, we ran tests on previously recorded data. Due to time constraints we were not able to test our new method live on the real robot, but had to resort to evaluating the segmentation process using recorded data. We evaluated our algorithm on 20 scenes. For every scene we had the visual data, the grouping into parts around local key points and the movement data for the first few interactions. In half of the scenes, the robot had taken random actions and in the other half the robot had taken actions expected to be most informative based on the available movement data. Shown in Fig: 4.4 are the comparative results. The measure of a segmentation's quality we use is defined by a comparison to the correct segmentation [16]. With Q and P being the sets of centers that belong to the same object according to the human annotation (Q) and our prediction (P), we define the quality as

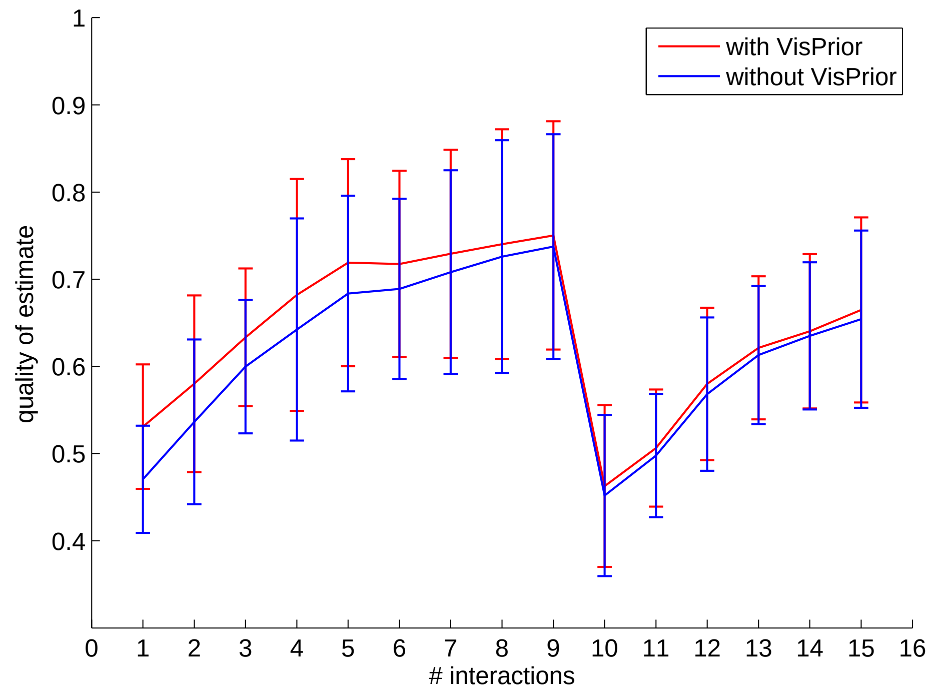
$$B = \frac{|P \cap Q|}{\sqrt{|P||Q|}}.$$

As one can see, the initial estimate after the first interaction shows a visible improvement compared to the purely interactive method. It can also be seen that the advantage gained by the visual prior diminishes slightly as the robot explores its environment via interaction. This is only natural since the robot will eventually have interacted with every part of the scene, leaving no knowledge to be gained by additional sources of information. The third conclusion we can draw from this comparison is, that informed actions generally lead to a quicker information gain (note that the graphs' y-axes are not equally scaled). This leads to the hypothesis, that we can improve the performance of the segmentation even further when taking the visual prior into account for choosing the most informative action.

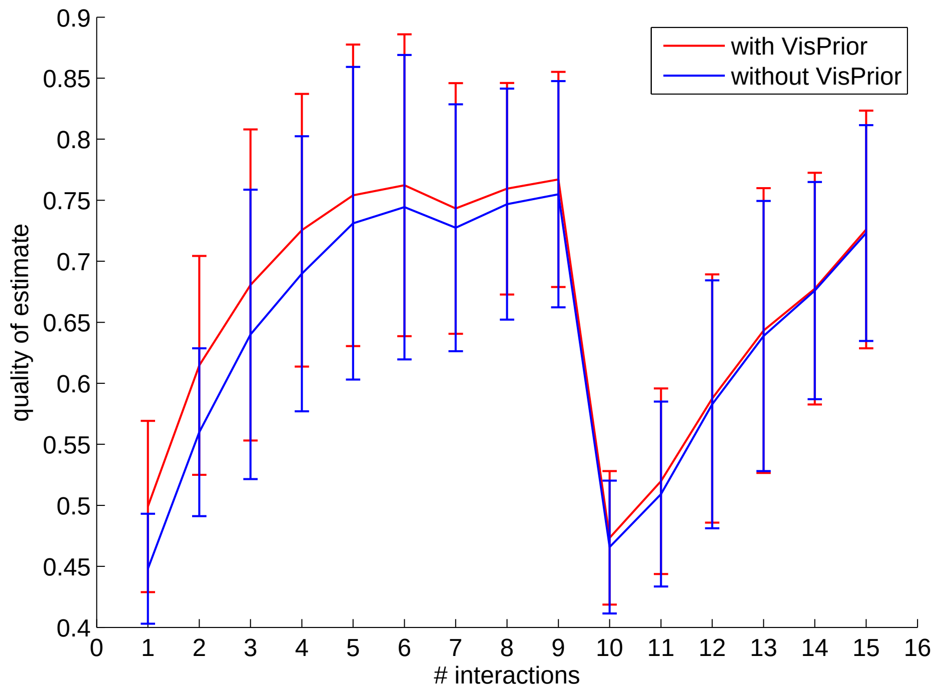
After the 9th interaction, one additional object was added to the scene. This leads to a temporary decrease in the segmentation's quality. This quality drop is especially significant, since this change in the environment is far more drastic, than the changes that naturally occur during interactive exploration. However, the algorithm is able to adapt to the new information within a few interactions. The advantage gained by evaluating the visual features after adding the object is visibly smaller than the initial one. Though this is not because the algorithm performs poorly, but because we had no visual data recorded from after the scene change that could have been evaluated, so we had to rely on the visual information without the new object.

Further evaluation on the λ parameter

Seeing the results of the cross-validation for evaluating the regularization parameter λ, the most logical choice for lambda would be either 0.01 or 0.1. These are, based on our training data, the values for which we might expect the best results for new data, since they had the lowest rate of misclassifications. However, the parameter λ is set to 1 during our research. The reason is, that with our training examples we still do not cover enough of the segmentation space. Evidence for this can be seen in Fig: 4.5 where we evaluated the final combined segmentation process using parameters learned with a regularization parameter of $\lambda = 0.01$. Apparently, we still do not generalize well on unseen data and get unpredictable results for these, ultimately preventing any improvements. For this reason, we choose the higher value of $\lambda = 1$, since being less confident might lead to a decreased information gain while overconfidence can even produce wrong information.



(a) Using random interactions



(b) Using informed interactions

Figure 4.4: Comparison of the segmentation results. The blue line represents the average results and the standard of the interactive segmentation without the movement data over the number of iterations. The red line represents the results, when the visual prior was used. We used data of 20 different setups, evaluating the movement data for up to 15 interactions. After the 9th interaction we added one object to the scene in order to evaluate the performance in changing environments.

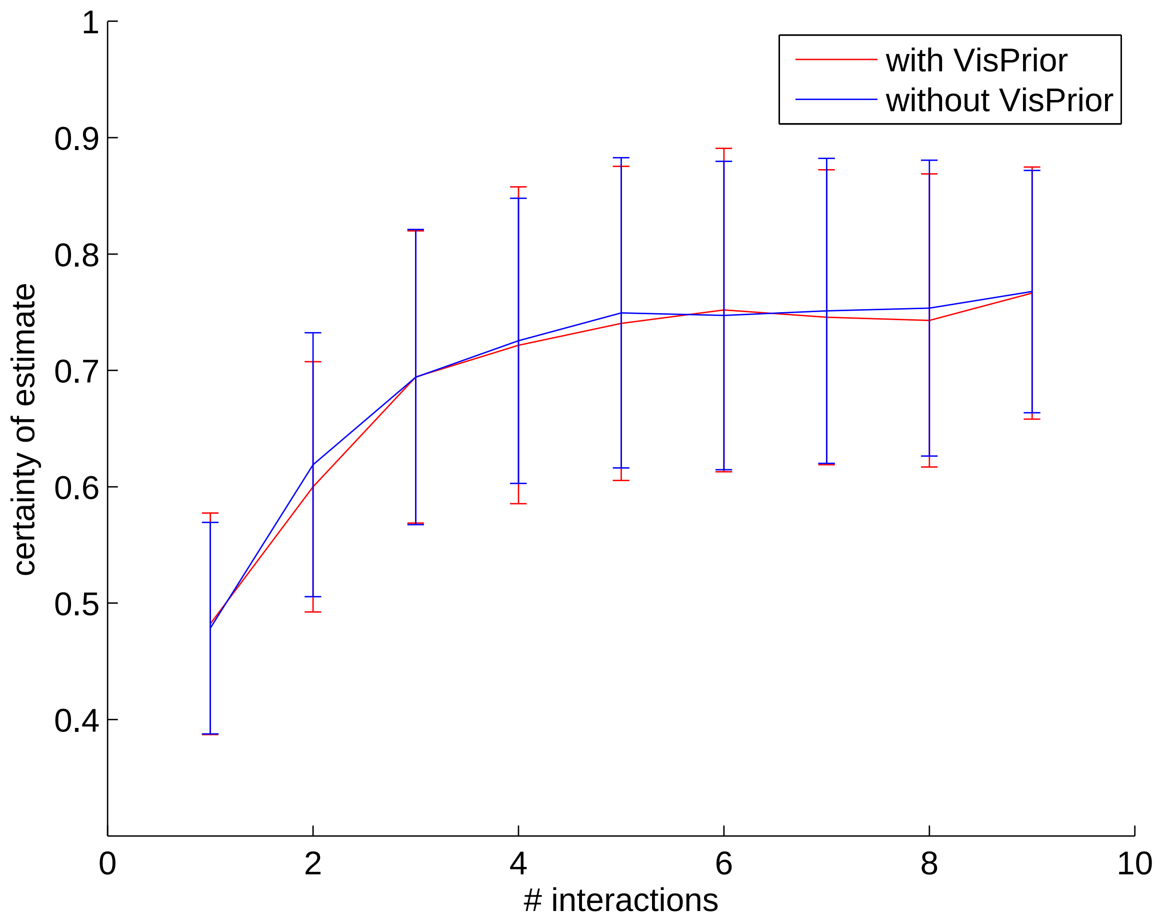


Figure 4.5: Results of the interactive segmentation comparing the purely interactive approach with the results when using a poorly trained visual prior. The problem here is that the regularization parameter $\lambda = 0.01$ was set too low and data not considered during training could not be generalized well.

5 Conclusion

In this thesis, we presented our probabilistic approach for segmentation that is based both on visual data and interaction. Our approach uses the visual data as prior knowledge in order to improve the performance of the interactive segmentation introduced by van Hoof et al. [16].

First we introduce a way of evaluating visual features using logistic regression. It shows that a reliable classifier over good and bad segmentations can be trained using only few visual features. We compare two ways of calculating these visual features and show that the probabilistic approach, which derives both meaning and significance of the features from actual data, is superior to the hand-tuned way of implementing the features.

Finally we integrate the knowledge gained from the visual data into the interactive probabilistic segmentation. We show that it reduces the number of interactions needed to reach the same amount of information about the scene. The improvement is most significant at the beginning of the segmentation process, when less movement data is available to be evaluated.

5.1 Outlook

There are still possibilities for possible future work. The most important step when continuing our research would be to test the entire combined segmentation algorithm online on a real robot. Ideally with previously unseen objects or new combinations of the objects used so far. When the results are satisfactory, one can evaluate how much of a further improvement can be gained, when incorporating the knowledge gained from the visual features into the choice of the next action.

These live tests could also include more challenging segmentation tasks. E.g., include objects that are not segmentable by vision alone, like a box sawn into pieces that are put together as one again. Or objects that move together despite being separate, like objects placed on top of each other or maybe a box in a bowl.

In addition, one can further evaluate the significance of the visual features currently used. Either by measuring the variance of the training data covered by the individual features, or comparing the results of both the classification and the segmentation when disregarding specific features. Also interesting would be the evaluation of more complex features than the ones we currently have. Especially operating in 3D space using a part based approach gives us access to many interesting features. For example, comparison of the face normals of surfaces in an object or surface continuity.

Bibliography

[1] D. Aldous. Exchangeability and related topics. *École d'Été de Probabilités de Saint-Flour XIII—1983*, pages 1–198, 1985.

[2] C. M. Bishop. *Pattern recognition and machine learning*, volume 1. Springer, 2006.

[3] M. Björkman and D. Kragic. Active 3d scene segmentation and detection of unknown objects. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3114–3120. IEEE, 2010.

[4] J. Carreira and C. Sminchisescu. CPMC: Automatic object segmentation using constrained parametric min-cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1312–1328, 2012.

[5] G. Casella and E. I. George. Explaining the Gibbs sampler. *The American Statistician*, 46(3):167–174, 1992.

[6] A. Collet, D. Berenson, S. S. Srinivasa, and D. Ferguson. Object recognition and full pose registration from a single image for robotic manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 48–55. IEEE, 2009.

[7] D. Comaniciu and P. Meer. Robust analysis of feature spaces: color image segmentation. In *Proceedings. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 750–755, 1997.

[8] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[9] P. Fitzpatrick and G. Metta. Grounding vision through experimental manipulation. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 361(1811):2165–2185, 2003.

[10] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter. *Markov chain Monte Carlo in practice*, volume 2. CRC press, 1996.

[11] A. E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–449, 1998.

[12] D. Katz and O. Brock. Interactive segmentation of articulated objects in 3d. In *Workshop on Mobile Manipulation: Integrating Perception and Manipulation, IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[13] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[14] A.S. Mian, M. Bennamoun, and R. Owens. Three-dimensional model-based object recognition and segmentation in cluttered scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1584–1601, 2006.

[15] X. Ren and J. Malik. Learning a classification model for segmentation. In *Proceedings. IEEE International Conference on Computer Vision*, pages 10–17 vol.1, 2003.

[16] H. van Hoof, O. Kroemer, and J. Peters. Probabilistic interactive segmentation for anthropomorphic robots in cluttered environments. In *Proceedings of the International Conference on Humanoid Robots (HUMANOIDS)*, 2013.

[17] T. J. Ypma. Historical development of the newton-raphson method. *SIAM review*, 37(4):531–551, 1995.