Chapter 5

Box-like Object Recovery using Superquadrics

5.1 Introduction

Since the mid-eighties, when the potential of superquadrics as modeling entities for computer vision applications was pointed out for the first time [121], a variety of algorithms for image segmentation using superquadrics has been devised, and considerable progress on the robustness of the obtained results has been performed. The existing approaches attempting superquadric segmentation can be roughly divided in two categories: The approaches within which the model parameter estimation is decoupled from image segmentation, and those according to which image segmentation and model parameter estimation are interwoven.

In the context of the former category [68], [50], [108], [127], image segmentation is performed within a two-step process: Firstly, local pixel similarity criteria are used to group image points into larger features (surface patches, or boundary curves). Secondly, superquadric parameters are estimated by superquadric model matching via *alignment* to these features. These techniques belong to a broader strategy for model recovery, according to which a number of image features in the image generate a *hypothesis* about the values of the model parameters, which is verified or rejected by examination of other neighboring image features. This framework is widely known as the *hypothesis generation and verification* framework for object recovery, a more detailed description of which is presented in section 2.2.3. The disadvantage of such methods is that both their robustness and computational efficiency drop when the number of image features increases [77].

Approaches belonging to the latter category [67], [35], [41], [74], [170], [99], [77], follow the concept that image segmentation and model parameter estimation cannot be separated. These approaches do not base the parameter estimation on image features, but employ the smallest unit of information that the image contains, the pixels, for this purpose. The segmentation task is performed by *embedding* the model parameter estimation in the segmentation process. This is done by considering the segmentation problem as a mathematical optimization problem, in the context of which the parameters of all superquadrics in the image are obtained as the optimum of a global function expressing both the agreement of the models to the data on one hand, and the simplicity of the way in which the models explain the image on the other. These strategies, belong to the generic category of model based

image segmentation methods using *probabilistic* techniques.

5.1.1 Probabilistic image segmentation

The probabilistic techniques for image segmentation perform the segmentation task by employing *global* models with a small number of parameters to express the image data. In particular, they assume that the acquired image is generated by *corrupting* an *ideal* piecewise-smooth image [97]. The ideal image occurs by instantiating the parameters of a more generic structure, which is frequently referred to as *image model* in the literature. The image model parameters will be hereinafter referred to as *image* or *segmentation parameters*. The corruption is meant to express both deviations from the idealized piecewise- smooth image model, as well as flaws inherent to the data acquisition device, and is referred to as noise of the data acquisition process or simply *image noise*. Finally, the difference between the real image and the ideal image is referred to as the *residual*.

In general, the same real image can be generated by an infinite number of ideal images given a particular image model. The probabilistic techniques, determine the ideal image which actually generated the particular real image, or equivalently, estimate the image model parameters, and they achieve this task in an *optimal* way. In particular, they calculate the *posterior* probability distribution of the segmentation parameters, given the real image, and estimate the segmentation parameters by maximizing the posterior. According to [44], [156], this process is the best we can do in order to estimate the segmentation parameters, because maximization of the posterior not only minimizes the residuals, but the *average risk* of failing to select the parameters of the actual ideal image as well.

The most frequently employed image models when probabilistic approaches for image segmentation are utilized, are the *Markov Random Field* (MRF) ([61], [20], [162], [29], [106]), and the *Mixture* models ([97], [99], [12], [98], [53]). According to the former, the value of each pixel in the image is considered to be a random variable the probability of which is entirely dependent on the probabilities of the values of its neighbor pixels. Despite the elegance of modeling the image using MRFs such models are often hard to analyze, and their application tends to be computationally costly. The mixture models are much more intuitive tools for image modeling.

In the context of image modeling using mixture models, the ideal image is assumed to comprise an unknown number of objects. We now assume that these objects are three dimensional, and that each of them can be described by means of a piecewise smooth surface S, with implicit form $S(\mathbf{p}; \mathbf{X}) = 0$, which will be hereinafter referred to as *object model* or simply model. The first argument $\mathbf{p}$ of the model's implicit form, is the model parameter vector, and the second denotes a three dimensional point $\mathbf{X}(x, y, z)$ on S. Note, that in the general case, the objects in the same image can be modeled by different types of surfaces. Here we assume that all of them can be modeled by surfaces of the same type. This is why all target objects in our application can be adequately well described by superquadric models on one hand, and because this assumption makes the following analysis simpler on the other.

Given the ideal image, the real image I is obtained by means of a data acquisition mechanism,

which takes the form of additive white Gaussian noise (AWGN). If so, assuming $\mathbf{X}_i$ a three-dimensional point on the lth object S_l of the ideal image, and $\mathbf{X}$ the acquired measurement, stored at the pixel $\mathbf{x} = (x, y)$ of I, that is $I(\mathbf{x}) = \mathbf{X}$, then the conditional probability of obtaining $I(\mathbf{x})$ given that it belongs to S_l, or equivalently, the *likelihood* of $I(\mathbf{x})$ amounts to:

$$P(I(\mathbf{x})|\mathbf{p}_l) = \frac{1}{\sigma_l\sqrt{2\pi}} \exp\left(-\frac{S_l(\mathbf{p}_l; I(\mathbf{x}))^2}{2\sigma_l^2}\right), \mathbf{x} \in \mathbf{R}_l \tag{5.1}$$

The set of pixels in I, which correspond to the same object S_l, will be referred to as the *region* of the object in the image, and is denoted as $\mathbf{R}_l$ in the preceding equation. $\mathbf{p}_l$ represents the parameters of S_l, which since related to the generation of the corresponding image region, will be additionally referred to as *region parameters*. Finally, the noise variance σ_l, is considered to be an unknown constant for the region $\mathbf{R}_l$, not necessarily the same for all image regions.

We now assume that the image I comprises g regions of g different objects, and we associate each data point $I(\mathbf{x})$ in the image with a vector $\mathbf{f}$ of g elements, so that he lth element f_l of the vector is 1 if the point belongs to the lth region (where $1 \leq l \leq g$), and 0 if it does not. We further assume that I comprises N points and that $\mathbf{F}$ is an $N \times g$ array the jth row of which contains the vector $\mathbf{f}$ of the jth image point. If $\tilde{\mathbf{P}} = (\mathbf{p}_1, .., \mathbf{p}_g)$ a vector comprising the parameters of all the regions in the image, then the likelihood of the jth image point, can be written as a weighted sum or *mixture* of likelihood terms, just as in the following equation:

$$P(I(\mathbf{x}_j)|\mathbf{F}, \tilde{\mathbf{P}}) = \sum_{l=1}^{g} F_{jl} P(I(\mathbf{x}_j)|\mathbf{p}_l) \tag{5.2}$$

The form of eq. (5.2), justifies the name selected to characterize the particular image models. In this equation, $P(I(\mathbf{x})|\mathbf{p}_l)$ expresses the likelihood of $I(\mathbf{x}_j)$, given that it belongs to the l th region, (shown in eq. (5.1)), and $\mathbf{F}, \tilde{\mathbf{P}}$ the segmentation parameters.

The independence property of the gaussian data acquisition noise, allows for the straightforward calculation of the likelihood of a region l as a product of the likelihoods of the image points belonging to the region:

$$P(\mathbf{R}_l|\mathbf{F}, \tilde{\mathbf{P}}) = \prod_{j=1}^{N} F_{jl} P(I(\mathbf{x}_j)|\mathbf{p}_l), \tag{5.3}$$

and of the likelihood of the entire image, as a product of the likelihoods of all image points:

$$P(I|\mathbf{F}, \tilde{\mathbf{P}}) = \prod_{j=1}^{N} P(I(\mathbf{x}_j)|\mathbf{F}, \tilde{\mathbf{P}}) \tag{5.4}$$

Assuming now that the segmentation parameters are not fixed, but follow a *prior* probability distribution $P(\mathbf{F}, \tilde{\mathbf{P}})$, we can compute the posterior distribution of the parameters by applying the *Bayes* theorem:

$$P(\mathbf{F}, \tilde{\mathbf{P}}|I) = \frac{P(I|\mathbf{F}, \tilde{\mathbf{P}})P(\mathbf{F}, \tilde{\mathbf{P}})}{P(I)} \tag{5.5}$$

According to the probabilistic approaches for image segmentation, image segmentation is performed by determining the segmentation parameters via maximization of the posterior in the preceding equation. In particular, the logarithm of the right part of eq. (5.5) is usually maximized, so that the products of probabilities are turned into sums, which are easier to deal with. $P(I)$ is ignored, since it does not depend on the segmentation parameters. This process is known as the *Maximum A-Posteriori* (MAP) parameter estimation process, and retrieves the optimal segmentation parameters $P(\mathbf{F}^*, \tilde{\mathbf{P}}^*)$ as follows:

$$\begin{aligned} \mathbf{F}^*, \tilde{\mathbf{P}}^* &= \arg\max_{\mathbf{F}, \tilde{\mathbf{P}}} \Big(\log_2 P(\mathbf{F}, \tilde{\mathbf{P}}|I) \Big) \\ &= \arg\max_{\mathbf{F}, \tilde{\mathbf{P}}} \Big(\log_2 P(I|\mathbf{F}, \tilde{\mathbf{P}}) + \log_2 P(\mathbf{F}, \tilde{\mathbf{P}}) \Big) \end{aligned} \tag{5.6}$$

In [129], [128], as well as in [161], a more intuitive expression for the logarithm of the posterior is derived, according to which the logarithm of the prior probability of the segmentation parameters is expressed as a product of the number of elements p in $\tilde{\mathbf{P}}$, which is equivalent to the number of models in the image, times the logarithm of the number of image points N:

$$\log_2 P(\mathbf{F}, \tilde{\mathbf{P}}|I) \propto \log_2 P(I|\mathbf{F}, \tilde{\mathbf{P}}) - \frac{p}{2} \log_2 N \tag{5.7}$$

The intuitiveness of the preceding equation lies in the fact that it renders the role of the prior term in the maximization process apparent: to penalize segmentations involving a big number of object models, and thereby, to avoid image model *over-fitting* to the acquired image.

Image segmentation via direct optimization of the expression (5.7) is not a trivial task [138]. Global optimization techniques such as *simulated annealing* [61], *graduated nonconvexity* [20], *deterministic annealing* [60], *mean field annealing*, [59], and *hopfield neural network* construction [73] are perhaps the most successful for optimizing (5.7). However, their main drawback is their computational inefficiency, since they perform extensive search in a usually huge parameter space. Besides, tuning the parameters needed by most of these techniques is very often a tedious task. This is the reason why, in many cases [173], [119], [142], [99], [154], [17], [32], [3], [97] a different philosophy is adopted: They are based on the observation that local extrema of the posterior distribution of the segmentation parameters, generally correspond to sensible segmentations of the input image, and that in many cases a local optimum is often more desirable, since it corresponds to the 'closest' segmentation for a given initialization. For this reason, instead of directly optimizing (5.7), they break down the segmentation problem into two parts: a parameter initialization part, and a local optimization part. Given that the initialization of the parameters is close to the global optimum, usage of local optimization techniques is expected to deliver a robust segmentation result on one hand, within a very fast period of time on the other. The parameter initialization part is regarded as generation of an hypothesis about the values of the segmentation parameters.

The optimization part *refines* the hypothesis, by locally optimizing the posterior distribution. This is the reason why this two-stage segmentation framework will be hereinafter referred to as *hypothesis generation and refinement* strategy for image segmentation.

5.1.2 Hypothesis generation and refinement for image segmentation

The first stage of the strategy, that is the hypothesis generation part, aims at obtaining initial values of the segmentation parameters, which lie as close as possible to the actual parameter values. The degree in which this process achieves to perform its task is of highest importance for the success of the entire segmentation process. Due to the locality of the subsequent segmentation step, the initialization of segmentation parameters in a relatively long distance from the global optimum will inevitably produce an inaccurate image segmentation. Execution of this phase is assisted by information extracted from the input image by means of fast local operations such as histograming or thresholding [119], [102], discontinuity detection [110], [14], [145], [143], corner extraction [86] [31], etc. Based on this information, the matrix $\mathbf{F}$ is determined by assigning a number of image regions to objects, and the vector $\tilde{\mathbf{P}}$ by fitting the surface models to the points. Very often, the term *seeds* is used for these regions. This term is associated with the notion of smallness, and is used because the majority of the hypothesis generation techniques output regions encompassing a small number of image points. If this is not the case, the same term is as well employed, but is there used to manifest the limited expressive ability of these initially formed regions. Hence, the term *seed generation* or *seed placement* alternatively stands for the hypothesis generation process.

Given their initial values, the hypothesis refinement stage maximizes the posterior probability of the segmentation parameters via a two-step *iterative* process. According to this process, estimation of the region parameters is *decoupled* from the estimation of the number of image regions. In the first step, the number of regions in the image, and consequently the number of models, or the parameter p in eq. (5.7), is considered constant. Under this assumption the region parameter vector $\tilde{\mathbf{P}}$ is estimated on one hand, a local classification of unclassified image points is attempted, that is, $\mathbf{F}$ is updated on the other. This is realized by locally maximizing the image likelihood, that is the first term of the sum in eq. (5.7). In the second stage, the region parameter vector $\tilde{\mathbf{P}}$ is considered known. Under this assumption, eq. (5.7) is maximized by updating the number of image regions.

The most popular method for deriving segmentation parameters by image likelihood maximization, is a greedy technique widely known as *Expectation Maximization* (EM) [42], [107]. Expectation maximization is based on the observation that knowledge of the elements of $\mathbf{F}$, which are referred to as *latent variables* in this context, allows for straightforward estimation of the region parameters $\tilde{\mathbf{P}}$, and inversely, knowledge of the region parameters results in easy computation of the latent variables. Image segmentation using EM involves iterative invocations of the following two steps in succession: The *E*- step of the process performs a re-classification of image points to regions, given that the region parameters are known: The likelihood of every image point is calculated using eq. (5.1)., and it is assigned to the region to which it most likely belongs. In this way $\mathbf{F}$ is updated. Subsequently, a revised estimate

of the region parameter vector $\tilde{\mathbf{P}}$ is obtained, by maximizing the likelihood expression of eq. (5.3) for each region. This step is known as the M step of the process. The estimated parameter vector is then fed back to the E step and the whole process continues until no significant change in the segmentation parameters is observed. Note, that since this segmentation process involves point classification followed by model fitting it will be hereinafter referred to as the *classify-and-fit* process [77].

The second part of the parameter refinement stage, which will be hereinafter referred to as the *region number updating* (RNU) stage, considers the region parameters $\tilde{\mathbf{P}}$ known, and aims at finding the set of regions which comprise the best possible globally consistent image partitioning. This is performed by defining the *type* and the *sequence* of operations which when applied to the input regions updates their number so that (5.7) is maximized. Region number updating is achieved by means of four possible operations: Region *merging* [102], [173], [3], *splitting* [142], [62], [120], *rejection* [99], and *initialization* (creation) [173]. Region merging involves generation of one region out of two or more neighboring ones. Region splitting has to do with generating two or more regions out of one region. Region rejection implies discarding a region which is judged to inadequately describe the encompassed image points. Finally, region creation assigns a set of neighboring unclassified points to a newly created region. Retrieval of a globally optimal image partitioning is a problem with exponential complexity on the number of input image regions [142]. Hence, for the sake of computational complexity, usually *greedy* techniques [36], [99], [142] are employed.

The hypothesis generation and refinement framework, is a generic strategy for image segmentation, which has up to now been used for addressing a big variety of image segmentation related applications. What is of special interest to this work, is the way in which it can be applied for segmenting multiple superquadrics in range images, and the best example is the recover and select strategy for superquadric segmentation, initially described in [98], and improved in [99], [77]. The relation of this strategy to the hypothesis generation and refinement framework, a description of its operation, as well as a discussion about is advantages and problems is the subject of the section that follows.

5.1.3 Recover-and-select for superquadric segmentation

The most popular approach for segmentation of multiple superquadrics is the *recover-and-select paradigm* [98], [99], [77], which is inspired by the classical approach for range image segmentation using variable order surface fitting [17]. The recover-and-select paradigm is a robust framework for segmenting range images considered to comprise a set of objects which can be perfectly modeled via superquadric models, and adheres to the hypothesis generation and refinement framework for image segmentation.

More specifically, the hypothesis generation stage involves a grid-like placement of seeds in the image. The hypothesis-refinement stage updates the parameters of the initialized models, so that they more accurately express the corresponding data on one hand, rejects models judged to inaccurately express the data despite parameter updating on the other. The classify-and-fit part of the hypothesis refinement is realized by means of a region growing approach. According to the recover-and-select framework, the region number updating process is performed by means of *model-selection*. Model-selection allows continuation of

growth only to these models which represent the corresponding image points in an globally consistent way. Region growing succeeded by model selection, continues iteratively, until no change in the segmentation parameters occur.

In the following paragraphs we detail the description of the framework's components: Firstly, the seed placement process is described. Then the internals of the hypothesis refinement process is shown. Subsequently, we discuss how the system's components are set up together to generate a working system. Finally, a discussion about the advantages and drawbacks of the strategy concludes this section.

5.1.3.1 Hypothesis generation

The hypothesis generation or seed placement procedure within the context of the recover-and-select framework comprises two parts: Firstly, numerous superquadric models are placed in a uniform manner in the image, secondly, the models which do not accurately represent the data are rejected. The first part of the procedure is realized by generating a grid-like subdivision of the input range image in numerous uniform rectangular windows, each containing a few range points. The set of points contained in each window is assumed to be a region of points corresponding to a superquadric model. Subsequently, the parameters of the model are determined by means of maximum likelihood (eq. (5.3)), or more specifically, by minimizing the superquadric fitting cost function of eq. (3.15). Then, the *average error-of-fit* $\bar{\xi}_l$ of the model to its corresponding region is calculated as follows:

$$\bar{\xi}_l = \frac{\xi_l}{|\mathbf{R}_l|}, \tag{5.8}$$

where

$$\xi_l = \sum_{\mathbf{x} \in \mathbf{R}_l} d(I(\mathbf{x}), \mathbf{p}_l). \tag{5.9}$$

In the preceding equation, $\mathbf{R}_l$ denotes the *region* of the model l, $|\mathbf{R}_l|$ the number of pixels in the region, $\mathbf{p}_l$ is the estimated model parameter vector (or region parameter vector), and $I(\mathbf{x}) = \mathbf{X}(x, y, z)$ a range point of $\mathbf{R}_l$. $d(I(\mathbf{x}), \mathbf{p}_l)$ is the *radial euclidean distance* of the range point $\mathbf{X}$ to the superquadric model with parameter vector $\mathbf{p}_l$, given by eq. (3.17), and shown again here for easing the reader:

$$d(\mathbf{X}, \mathbf{p}_l) = |\mathbf{X}||1 - F^{-\frac{\epsilon_1}{2}}(\mathbf{p}_l; \mathbf{X})| \tag{5.10}$$

According to the second step of the seed placement process, the initialized models undergo a filtering: The models for which the average-error-of-fit is below a user defined threshold *max-average-model-error* are considered to reside in areas of the image which can not be satisfactory modeled by unique superquadric models, that is, they cross object boundaries, and they are rejected from further consideration. Model rejection at this point serves a dual purpose: Firstly, it guarantees that all remaining models satisfactory represent the data. Secondly, it reduces the number of models needed to be subsequently processed by the system, and thus reduces the computational costs.

5.1.3.2 Hypothesis refinement

The hypothesis refinement stage comprises two components: region growing and model selection. Region growing corresponds to the classify-and-fit part of the generic hypothesis refinement framework. Region growing is performed *independently* for all seeds generated by the seed placement process. Model selection corresponds to the region number updating part: Given the grown seeds, this process selects those determined to express the corresponding models in an accurate way, and discards the others. According to the hypothesis refinement stage of the recover-and-select framework, parameter refinement is performed by iteratively invoking region growing followed by model selection, until a satisfactory representation of the data via the models is achieved. In the following paragraphs, we describe the components of the hypothesis refinement process in detail.

5.1.3.2.1 Region growing: Due to model rejection by the filtering step of the seed generation procedure, image points exist which are not assigned to models. Target of the region growing procedure is the classification of these points to existing regions one hand, the refinement of the region parameters so that the newly classified points are taken into consideration on the other. This is achieved by means of an iterative process in the context of which the initialized seeds *grow* along their boundary, so that unclassified boundary points are classified as belonging to the seeds. This is a standard bottom-up procedure in computer vision ([174], [17]), for which both terms *region growing* and *model growing* are used without distinction.

Region growing is executed independently for each seed. In each iteration, the eight-connected neighbors of the boundary points of the seed are considered. Subsequently, the process examines the subset of those points whose distance to the corresponding superquadric model is lower than a user defined threshold, namely *max-point-distance*. If no such points exist, then the model is considered to be *fully-grown* and the process stops. Else, the new points are temporarily added to the region and a superquadric model is fitted to the region. If the fitting error residual is smaller than the *max-average-model-error* threshold, the new points are permanently added to the region. If the residual is large, the model is regarded as inappropriate to describe the region, and the process stops.

The major advantage of the region growing process is that it is highly intuitive. In addition, the constantly monitored analysis of data consistency to the growing model allows for rejection of outliers. However, the region growing process is a local approach for maximizing the *likelihood* of the entire image, and as such encounters difficulties in classifying points on object boundaries. Target of the model-selection process is to resolve these side-effects by imposing global consistency requirements on the segmentation process.

5.1.3.2.2 Model selection: The initialization of a big number of seeds, as well as their independent growing, leads to representation of a relatively big number of image points by more than one model. This redundancy in data representation is expressed as complete or partial overlapping of several models in the image. Model selection resolves these ambiguities in data representation. Given the parameters of all regions in the image, the process determines which models describe their corresponding regions in an globally consistent way, and *rejects* all others from further consideration. Intuitively, in the context of model selec-

tion, the models generated during the seed generation stage, and refined by means of region growing, compete with each other to be selected in the final interpretation of the image.

The model selection process penalizes redundancy in the interpretation of the image via the models, by imposing simplicity in the output of the region growing operation. The way in which simplicity can be imposed on the output of a parametric process in general, is described in detail in [129], [97], [128]. There, simplicity is defined within an information theoretic framework: The parameters of the process generating the data set, are reduced to the simplest form a variable may have: a bit. Then, a language for describing the data set in terms of bits is specified. Finally, the description length of the data set is extracted. The longer the description length, the bigger the number of the parameters of the process generating the data is, that is, the more complicated the process is, and inversely for simple processes the description length is small, which implies that the description length is a measure of the simplicity of the data generating process. Therefore, in order to impose simplicity to a process it is enough to determine its parameters so that the description length of the generated data is minimized. This operation is widely known as enforcing the *minimum description length* (MDL) criterion to the process. Note, that as pointed out in [128], [97], the estimation of a process'es parameters using MDL, is equivalent to the MAP parameter estimation, shown in (5.6). The choice between the two strategies depends on whether it is easier to specify a descriptive language, or the prior probability distributions of the parameters of the process.

Given that in our case the parametric process is the corrupted data generation from the ideal image, its parameters are the segmentation parameters delivered by the region growing operation, and the generated data is the real image, the model selection procedure defines a language for encoding the information contained in the real image, and imposes simplicity by minimizing the length of this encoding: Assuming M the number of models in the image after the end of the region growing process, the presence of models in a particular image interpretation is expressed through a boolean array $\mathbf{m^T} = [m_1, m_2, \cdots, m_M]$. A value of 1 at the position i of the array $\mathbf{m}$ means that the model m_i is included in the interpretation and a value of 0 that it is not. Initially, all elements of the array have the value 1, since M models are present after model growing.

The length of information in the image is expressed via the function $L_{image}(\mathbf{m})$, and is defined as the sum of the encoding length for points not described by models, plus the length for image models, that is:

$$L_{image}(\mathbf{m}) = L_{pointwise}(\mathbf{m}) + L_{models}(\mathbf{m}) \tag{5.11}$$

Assuming n_{all} the number of points in the range image and $n(\mathbf{m})$ the number of points described by the selected models, the length for encoding individual points is given by $L_{pointwise}(\mathbf{m}) = k_1(n_{all} - n(\mathbf{m}))$. Assuming $N(\mathbf{m})$ the number of parameters of all models in the image, and $\xi(\mathbf{m})$ the deviation between the models and the data described by models, the length of coding the image points described by models can be expressed as $L_{models}(\mathbf{m}) = k_2\xi(\mathbf{m}) + k_3N(\mathbf{m})$. k_1,k_2,k_3 user defined constants. Consequently, the length for encoding the information contained in the image is given by the eq. (5.12).

$$L_{image}(\mathbf{m}) = k_1(n_{all} - n(\mathbf{m})) + k_2\xi(\mathbf{m}) + k_3N(\mathbf{m}) \tag{5.12}$$

Application of the MDL criterion to the image is equivalent to the minimization of $L_{image}(\mathbf{m})$ with respect to $\mathbf{m}$. Since n_{all} is a constant value, minimization of eq. (5.12) is equivalent to the maximization of the function $F(\mathbf{m})$ in eq. (5.13), so that model configurations in which few number of models describe a big number of data points with low deviation are favored.

$$F(\mathbf{m}) = k_1 n(\mathbf{m}) - k_2 \xi(\mathbf{m}) - k_3 N(\mathbf{m}) \tag{5.13}$$

Hence, the image interpretation $\hat{\mathbf{m}}$ maximizing $F(\mathbf{m})$ as in eq. (5.14), is expected to describe the data set in the simplest possible way.

$$\hat{\mathbf{m}} = \arg\max_{\mathbf{m}} F(\mathbf{m}) \tag{5.14}$$

The objective function $F(\mathbf{m})$ can be expressed in terms of a matrix product:

$$F(\mathbf{m}) = \begin{bmatrix} m_1 \cdots m_i \cdots m_M \end{bmatrix} \begin{bmatrix} c_{11} & \cdots & c_{1i} & \cdots & c_{1M} \\ \vdots & & \vdots & & \vdots \\ c_{i1} & \cdots & c_{ii} & \cdots & c_{iM} \\ \vdots & & \vdots & & \vdots \\ c_{M1} & \cdots & c_{Mi} & \cdots & c_{MM} \end{bmatrix} \begin{bmatrix} m_1 \\ \vdots \\ m_i \\ \vdots \\ m_M \end{bmatrix}, \tag{5.15}$$

which can be equivalently written as:

$$F(\mathbf{m}) = \mathbf{m^T Q m} \tag{5.16}$$

The diagonal elements of the matrix $\mathbf{Q}$ define the benefit value that the inclusion of the model m_i would introduce, that is:

$$c_{ii} = k_1 |\mathbf{R}_i| - k_2 \xi_i - k_3 \dim \mathbf{p_i} \tag{5.17}$$

The off-diagonal elements express the benefit of overlap between the models m_i and m_j, as in eq. (5.18). In this equation, $|\mathbf{R}_i \cap \mathbf{R}_j|$ is the number of image points explained by both m_i and m_j.

$$c_{ij} = -\frac{k_1}{2} |\mathbf{R}_i \cap \mathbf{R}_j| + k_2 \xi_{ij} \tag{5.18}$$

ξ_{ij} is the maximum average-error-of fit originating from the explanation of the points in the intersection of the regions $\mathbf{R}_i$ and $\mathbf{R}_j$ expressed in eq. (5.19).

$$\xi_{ij} = \max \left\{ \sum_{\mathbf{X} \in \mathbf{R}_i \bigcap \mathbf{R}_j} d^2(\mathbf{X}, m_i), \sum_{\mathbf{X} \in \mathbf{R}_i \bigcap \mathbf{R}_j} d^2(\mathbf{X}, m_j) \right\} \tag{5.19}$$

The maximum of the *Quadratic Boolean Problem* formulated in eq. (5.16) is retrieved by means of a greedy approach based on the *winner-takes-all* principle, which delivers a sub-optimal solution in $O(M^2)$. Note that the matrix $\mathbf{Q}$, is symmetric, and may be sparse or banded, depending on the degree of the overlap of the models. This property of the matrix is taken into consideration during optimization, since it results in faster evaluations of $F(\mathbf{m})$. Finally, the values for the constants k_1, k_2, k_3 are chosen experimentally.

5.1.3.3 Overall approach

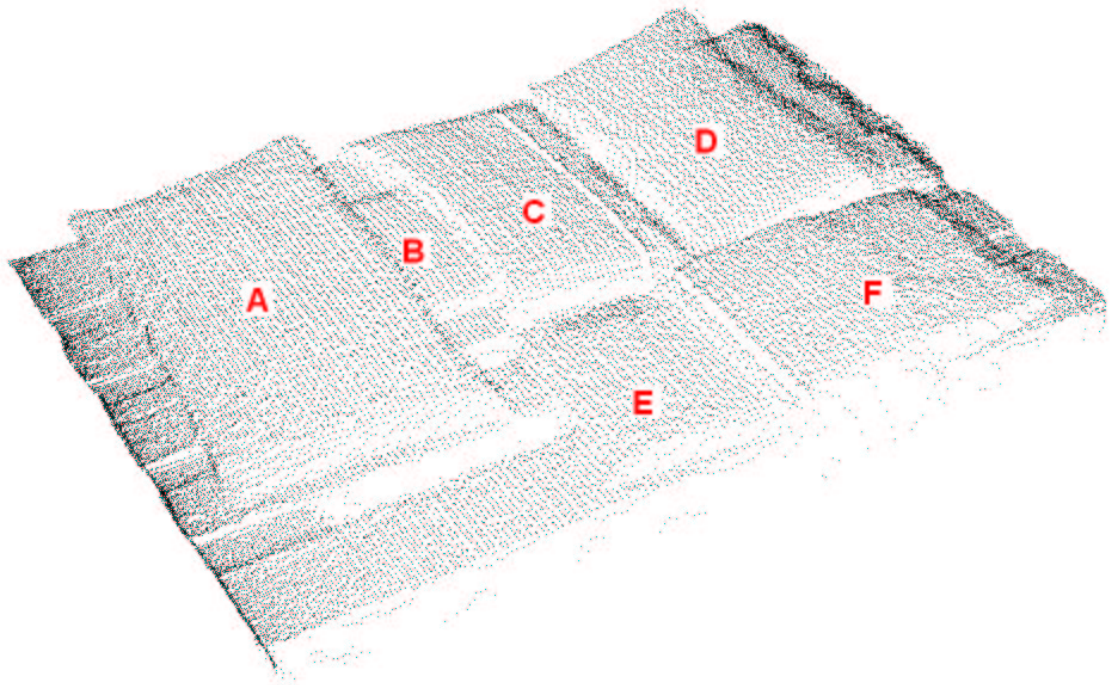
The recover-and-select paradigm achieves image segmentation in two steps: Firstly, seed generation is performed. Secondly, the actual segmentation takes place. The latter step comprises model recovery via region growing and model selection. There are mainly two ways to perform the latter step. One could wait until all seeds are fully grown, and then invoke model selection. In [77] the term *recover-then-select* is used to refer to the approach. The recover-then-select strategy has drawbacks: The computational cost of growing all models completely is prohibitive in most cases. Complete region growing will result in severe overlap of models in the image. As a consequence, the matrix $\mathbf{Q}$ of eq. (5.16) will be less sparse, and the selection process costly. The advantage associated with the process is equally important: Severe model overlap introduced by the process has as consequence the fact that only a few models will be left for further growing after the invocation of the model selection process. In addition, the more a region has grown, the better its corresponding model describes the region. Hence, the output of the model selection process will be more robust. In short, the recover-then-select approach is robust but computationally inefficient.

Computational costs can be reduced by interweaving model recovery and selection, that is by not waiting for full growth of all models for invoking model selection. This is the strategy employed in the recover-and-select framework. Note however, that very early invocation of model selection may lead to unreliable data representations. It is thus very important to determine when model-selection will be performed. In [77], a solution which balances computational efficiency and reliability is proposed: the region-growing procedure is interrupted when at least one growing model reaches twice its original size, that is four times its area, or when there are no models left to grow any further. Given the grid-like placement of seeds in the image, this criterion ensures sufficient overlap of models in one hand, as well as moderate computational costs of the growing process.

5.1.3.4 Discussion

The recover-and-select segmentation strategy exhibits advantages like robustness, accuracy and computational efficiency. Robustness and accuracy are the outcomes of combining the bottom-up iterative region growing approach for model recovery with the top-down model selection approach. The iterative region growing process is very robust against gross measurement errors. Usage of the *max-point-distance* parameter prevents such noisy points from being included into regions of growing models. However, points belonging to neighboring objects are often sufficiently close to a growing model and are erroneously included in the region of the model. To illustrate this phenomenon, we use a configuration of objects shown

(a) Intensity image



(b) Range image

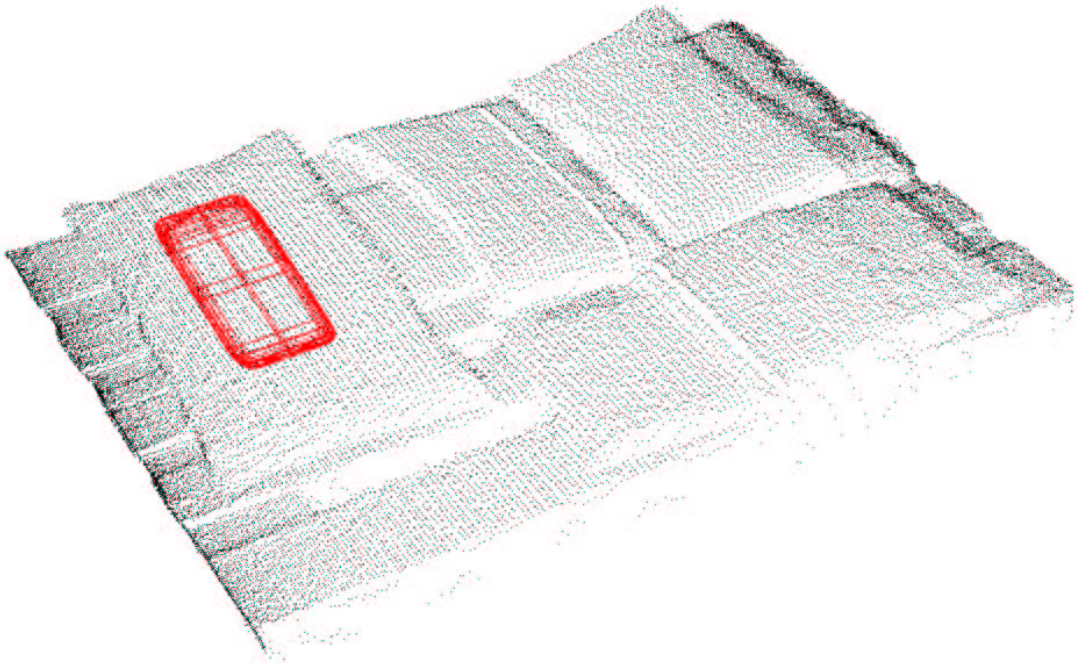
Figure 5.1: Box-like object configuration

in fig. 5.1 (a). Fig. 5.1 (b) shows the corresponding range image. The six objects in the image have been labeled with labels ranging from A to F. Fig. 5.2, demonstrates the over growing effect. In particular, fig. 5.2 (a), shows a seed initialized on the object A. Application of region growing to the seed leads to recovery of the model depicted in fig. 5.2 (b). Monotonic inclusion of points belonging to the neighboring object to the model's region leads to gradual increase of the model's fitting error residual, which causes termination of the growing process when the threshold *max-average-model-error* is exceeded. However, this happens only after the model has crossed the object boundaries, and results in inaccurate object recovery. Due to its inherent locality the region growing process has no mechanism to detect that the model has reached the boundaries of the object it represents. This phenomenon is a known weakness of region growing and is known as the model *over-growing* problem.

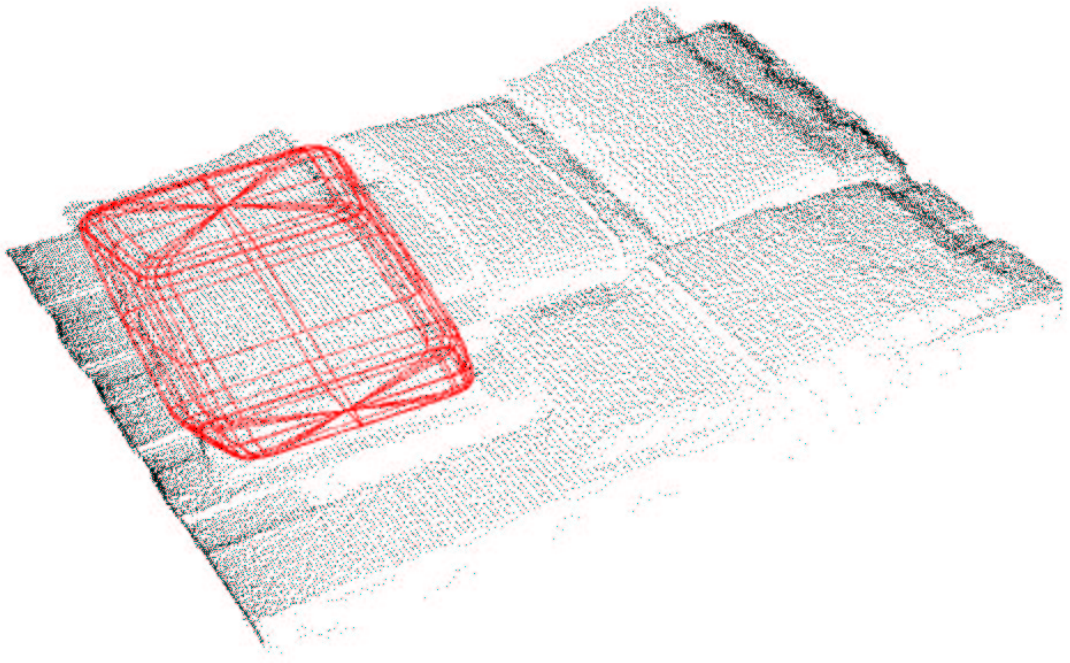
One could think, that a possible solution to the problem would be to decrease the *max-point-distance* threshold. In this case the model growing stops too early. As observed in [77] p. 156, the robustness of the region growing technique can be improved by careful seed placement. It is there experimentally observed, that when the model approaches its final and correct size, the influence of the outliers from the neighboring objects decreases. Given the fact that the model grows approximately symmetrically along the normal of the perimeter of its region, placement of the seed in the central area of the object is expected to increase the robustness of region growing. However, finding the central area of the objects already assumes some knowledge of the pose of the objects in the image which cannot be obtained before segmenting the image. For this reason, the problem of seed placement is *implicitly* solved in the recover-and-select framework, by placing numerous redundant seeds in the image and by invoking the model selection procedure after region growing, to get rid of these seeds with large fitting error, which probably cross object boundaries. The effectiveness of model selection in retaining the most reliable models has as well to do with the fact that the models grow independently. This is because probably erroneous initial model estimates do not influence the development of the recovery of other models originating from seeds better placed in the image. In addition, independent region growing offers the possibility of a parallel implementation for model recovery.

The computational efficiency of the recover-and-select framework is a consequence of the efficiency of its two constituting parts, that is region growing, and model selection. Model recovery is based in the fast superquadric fitting function eq. (3.15). Besides, the computation of the superquadric fitting function and its derivatives is independent for each range point and can be parallelized at a fine grain level in a straightforward way. In addition, model selection is realized using a greedy approach which results into a fast process even when the number of models involved is high.

We have extensively experimented with the recover-and-select framework for segmenting range images of box-like object configurations. We show here the result obtained by applying the framework to the range image of fig. 5.1 (b). More specifically, fig. 5.3 depicts the output of the seed placement process. In fig. 5.4 (a) an intermediate output of the framework is illustrated, more specifically the segmentation result after the third invocation of the model selection process. Finally, fig. 5.4 (b) shows the final result. We have experimented with a variety of parameter sets for obtaining as accurate segmentation results as possible. The parameter sets using which the best results have been achieved, and the segmentation



(a) Initialized seed



(b) Model over-growing

Figure 5.2: Model over- growing

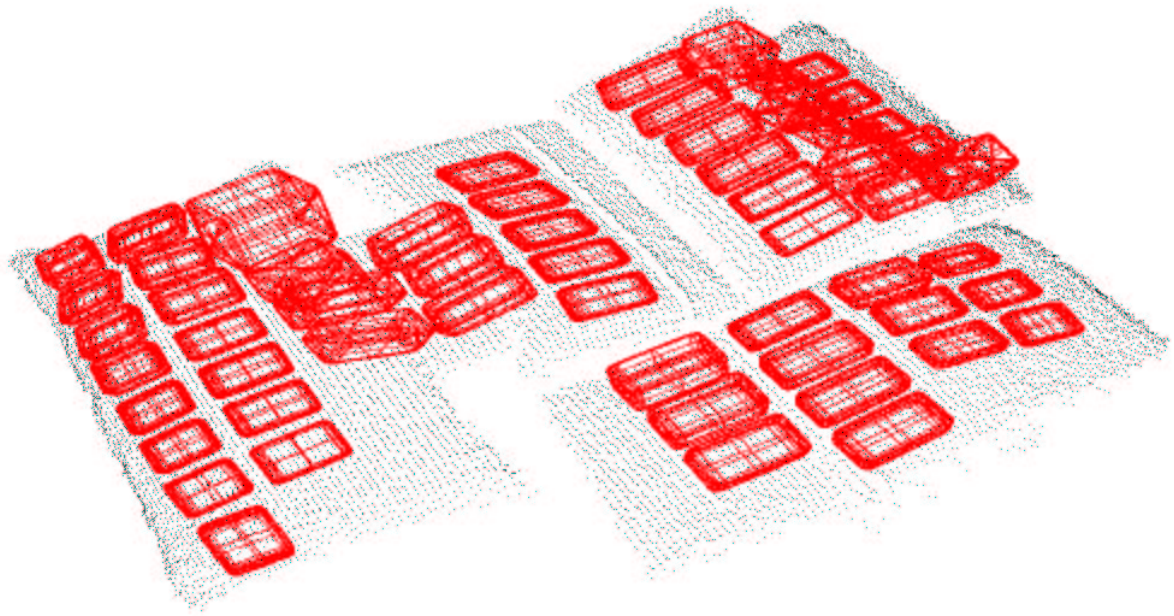


Figure 5.3: Seed placement

of fig. 5.4 (b) was obtained, are shown in table 5.1. The execution time required for running the process, was about 15 minutes in a Pentium 4 PC, with 2.8Ghz.

The preceding statement concerning the computational efficiency of the framework, as well as inspection of the obtained results, point out that there is a lot of space for improvements in terms of both its computational efficiency and robustness. Initialization of numerous seeds as a means for addressing the region over-growing problem, in combination with the frequent invocations of the non parallelizable model selection process, renders the framework inappropriate for a real time implementation. Besides, the region over-growing problem persists, despite the invocations of the model selection process: The recovered models corresponding to the objects A, E in fig. 5.4 (b) are overgrown. This has to do with the fact that the solution delivered by the model selection is sub-optimal due to its greedy implementation. The greedy model selection process does not possess the look-a-head capability to optimally arrange the models in a piecewise description. Hence, in many cases either overgrown models are favored, or an overgrown model describing a superset of a data set is selected in addition to the model optimally describing the particular data set, as is the case for the object C. The latter case is identified in [94], a work extending the recover-and-select framework, as *duplication of models on same segments*, and a solution is proposed in this work at p. 54: They suggest an adjustment of the c_{ij} term shown in eq. (5.18) so that model overlapping is severely penalized by setting:

$$c_{ij} = -\frac{k_1}{2}|\mathbf{R}_i \cap \mathbf{R}_j| \tag{5.20}$$

Although this correction improves the results of the framework, it overlooks the fact that the main reason for inaccuracy in the segmentation is not the design of the global cost function, but rather model over-growing. As experimentally observed in [77] p. 130, model selection delivers suboptimal results when models which stopped growing, because their fitting error residual has slowly accumulated to the maximum allowable value, enter the model selection procedure. Models for which gradual accumulation of fitting error is observed, are models crossing object boundaries, which verifies that model over-growing is the main cause for the system's inaccuracy.

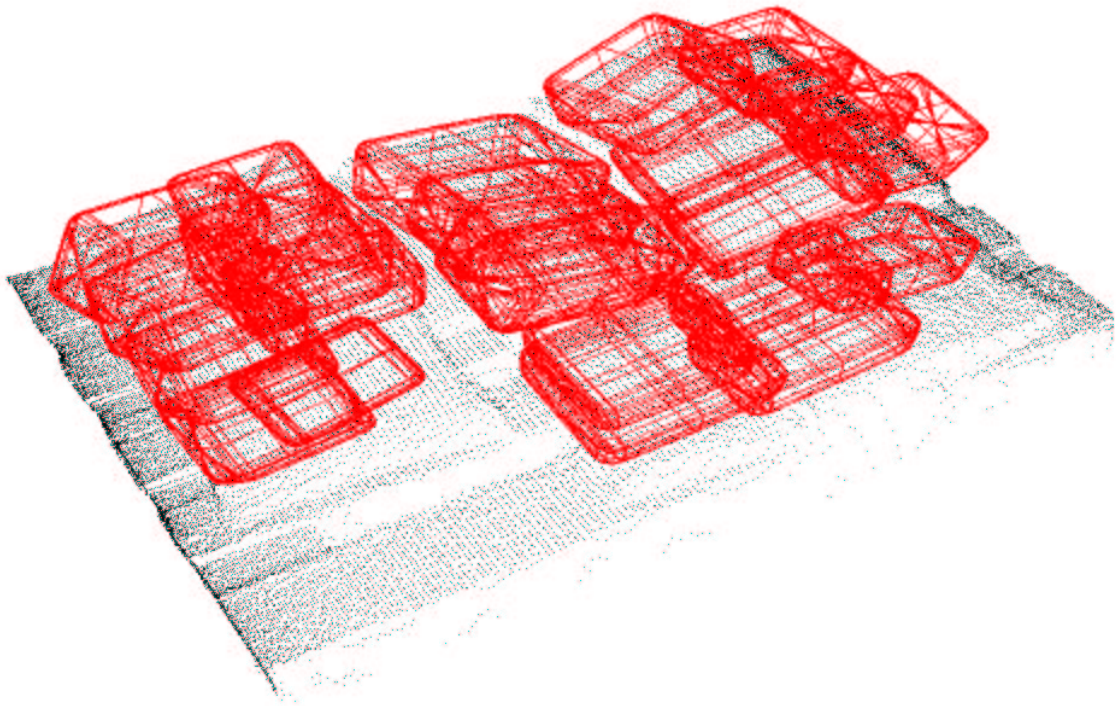
Especially in our data sets, model overgrowing appears to have a much more severe effect along the height parameter (a_3) of the models. This happens because the objects expose at most two surfaces to the laser source, hence, no measurements are delivered which could sufficiently constrain this parameter. As a consequence, the uncertainty in the estimation of the height parameter is expressed as overestimation of its value. The reason for this is that model recovery is performed using the equation (3.15), which as seen in eq. (3.16) is biased towards larger superquadrics. In addition, iterative superquadric recovery intensifies the problem due to accumulation of the error in the estimation of the height parameter in each iteration.

Parameters	*max-point-distance*	*max-average-model-error*	k_1	k_2	k_3
Values	15mm	20mm	1.0	0.05	1.0

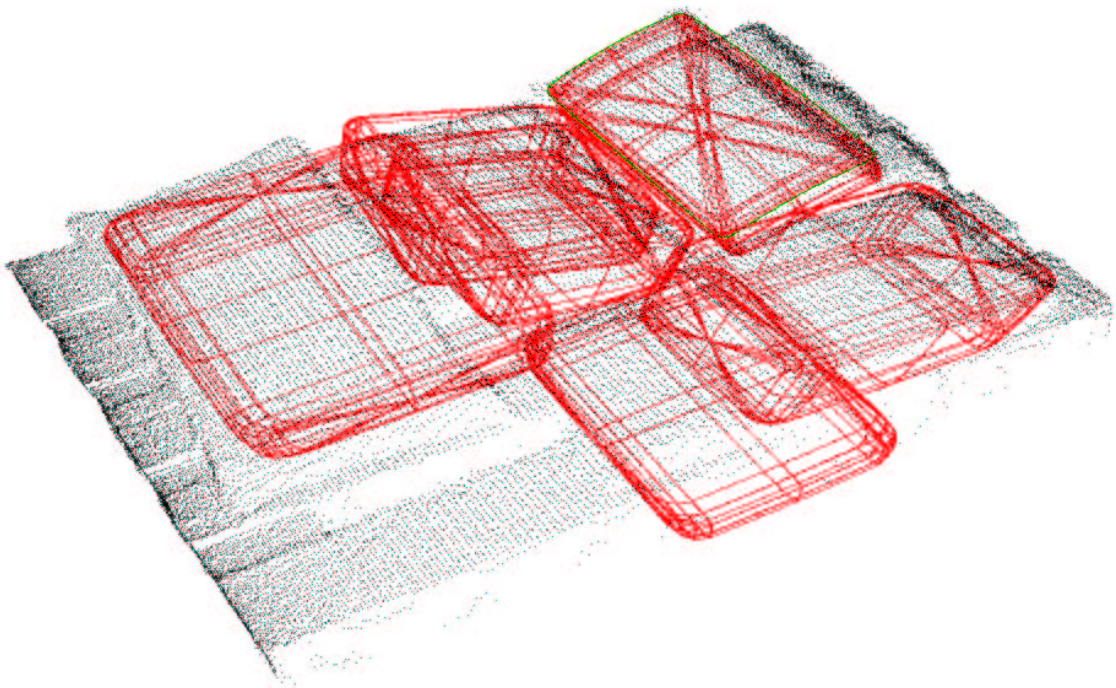
Table 5.1: Parameter values for recover-and-select

Attempts for improving the robustness of the recover-and-select paradigm by explicitly addressing the model overgrowing problem have been proposed in both [77] and [94]. In [77] p. 157, a post-processing step is introduced to improve the accuracy in boundary recovery. In [94] p. 47, it is observed that despite the post-processing introduced in [77] severe model overgrowing still occurs. In p. 48 of [94] a method for boundary localization based on a statistical test is introduced, which still does not completely resolve the problem. In these works, no solution to the problem of error in the recovery of the uncertain parameters is proposed. Resolution of uncertainty requires acquisition of additional views of the configuration [164]. Such a solution is not applicable in our case for two reasons: (**i**) the target objects are heavily occluded. It is thus impossible to acquire information about the height of the objects lying in the middle of the configuration from another viewpoint. (**ii**) Data acquisition is a slow process. Thus, the payoff of acquiring additional views does not compensate the delay that is caused for acquiring the additional data.

Since the majority of the problems associated with the recover and select paradigm stem from the region overgrowing phenomenon, its overcoming is expected to considerably improve the framework's robustness. Our approach explicitly addresses region over-growing by incorporating boundary information in the segmentation process. A short description of our framework for box-like superquadric segmentation is presented in the paragraph that follows.



(a) Intermediate result



(b) Final result

Figure 5.4: Recover-and-select recovery

5.1.4 Our approach

Our approach is inspired by the recover-and-select framework for recovering *box-like* superquadrics from range images, but it is superior to it in terms of both robustness and computational efficiency. The main reason why this happens is because *boundary information* is incorporated into the segmentation process. In the context of our approach, the same parts constituting the recover-and-select strategy are used: An hypothesis about the segmentation parameters is generated by a seed placement procedure. This hypothesis is locally refined by means of a classify-and-fit procedure applied independently on each seed, followed by a model selection which retains the models which best describe the data. However, the way in which the modules are internally implemented differs.

Our segmentation framework is employed by a system for automatic unloading of box-like objects, for which the *graspable objects* in the pile are of interest. This is the reason why, target of our segmentation approach is not to perform a *complete* image segmentation, but rather to find the graspable objects in the image. As seen in section 2.4, a box-like object is considered graspable if no object lies above it. These objects fully exposed their biggest surface to the laser source. In short, instead of attempting complete image segmentation, target of our strategy is the recovery of fully exposed surfaces of box-like superquadric objects in the input range image. Boundary information is used for addressing this problem. The edge detection technique introduced in chapter 4 outputs edge points on the boundaries of the exposed surfaces of the objects. For graspable objects, application of edge detection will output edge points uniformly distributed all around the boundary of their exposed surface. Such points can be very well modeled by the boundary of the exposed surface of a box-like superquadric model (*superquadric boundary*), defined in chapter 3, or equivalently, the fitting error residual between these points and the superquadric boundary is expected to be small. However, in order to determine if a fitted superquadric boundary actually corresponds to an exposed surface of a graspable object, we need to use range information as well: We need to know if the range points enclosed within the boundary can be as well satisfactory modeled by a box-like superquadric model, and we can determine this if we fit the model to these points. This strategy for recovering graspable objects by model fitting in both the edge image and the boundary image is the core of our approach.

In particular, given both the range and the edge image, the hypothesis generation stage of our approach forms closed contours by iteratively *dilating* the edge image. The region of points contained within each closed contour, which can be well modeled by a box-like superquadric model is considered a seed of our segmentation process. Usage of boundary information for seed placement, guarantees that seeds do not cross object boundaries on one hand, most likely generates one seed per graspable object on the other. Subsequently, the parameters of each seed are refined by a classify-and-fit process. The classify-and-fit process in our case performs local image segmentation, by iterative invocations of two modules in succession, in each of which the point classification followed by model fitting is executed in a different information domain: The region module recovers the superquadric parameters in the range image. The boundary module recovers the parameters of the superquadric boundary in the boundary image. After the completion of the dual classify-and-fit process a procedure corresponding to the model selection of the recover-and-select framework is initiated. For each recovered model, it checks the fitting error residuals in both information

domains and outputs models the fitting errors of which is low.

The robustness of our approach with regard to the recover-and-select paradigm, stems mainly from the fact that incorporation of boundary information resolves the model over-growing problem: Using boundary information for model fitting does not allow the growing models to cross object boundaries. In addition, the robustness is the outcome of the combination of two complimentary approaches for image segmentation, the top-down scan line splitting technique using which the boundary information is generated, with the bottom-up classify-and-fit segmentation approach, on which the parameter refinement framework is based. The computational efficiency of our approach stems primarily from the initialization of small number of models, which is the result of seed formation via adaptive closure of the edge map, with the minimization of the invocations of the model selection process in combination with the simplification of its operation.

Despite that we claim extending the recover-and-select framework, the fact that our algorithm is designed to recover box-like superquadric models might seem to considerably limit its application range. As we will discuss in section 5.4, our approach can be easily extended so that every kind of superquadrics can be recovered. In order to do this, two changes should be incorporated: Firstly, the edge detection process should be modified so that the output edge points are points on the RIM (see section 3.8 for the definition) of the image objects. Secondly the boundary module of the classify-and-fit process should fit the rim of the superquadric model instead of the boundary of the exposed surface to the edge image. Initial experiments in this direction have already been performed and the results are encouraging.

In the following paragraphs we detail our approach for box-like superquadric recovery: Firstly, section 5.2, addresses the problem of how boundary and region based information can be integrated within a segmentation framework, and proposes an efficient and robust way for achieving this integration based on *game theory*. In section 5.3, the way in which integration of boundary and region information is done in our framework is presented. Additionally, experiments which demonstrate the validity of our approach are presented in this section. Finally, section 5.4 discusses the advantages and drawbacks of our approach, and presents the directions of our future research.

5.2 Integration of boundary information in image segmentation

According to the recover-and-select framework, the mechanism guiding the segmentation task is the region-growing process, which is perhaps the most famous representative of the so called *region-based* approaches for image segmentation ([3], [174], [34], [56]). In the context of the region-based approaches, the decisions about the classification of image points to regions depend on *criteria*, which determine the *conformity* of the points to the regions, a measure of which is the *likelihood* of the point. As seen in section 5.1.1, when range image segmentation is concerned, the regions are assumed to be corrupted versions of piecewise smooth surfaces, the parameters of which are extracted by a statistical estimation process involving a big number of image points already known to belong to the regions. Since the classification decision is influenced by all the image points on which the region parameter

estimation is based, the region-based segmentation approaches are not susceptible to local variations of the noise in the image. The bigger the number of points taking place in the region parameter estimation, the more accurate will the estimated parameters explain the points included in the region, and the more robust the classification decision will be. However, the region-based approaches often tend to sacrifice resolution and detail in order to gain a sample large enough for the parameter estimation. Consequently, the boundaries of the regions are usually distorted, which implies that region-based segmentation is not the method of choice when the accurate reconstruction of the regions's shapes is of importance. A further problem is that region-based approaches fail to distinguish regions of small size. Finally, there is a trade-off between robustness and computational efficiency: the more points are used for parameter estimation the most costly the estimation process, and thus the overall segmentation is.

While in region-based approaches similarity between image points guide the segmentation process, the *boundary-based* methods rely on the discontinuities between image points to perform the segmentation task ([146], [130], [25], [125], [43]). These methods usually perform a local linear filtering in the image in order to detect abrupt changes in the pixel values, which are expected to correspond to points on the object boundaries. Compared to the region-based approaches, their main advantage is the computational efficiency, since the filtering is usually performed using a fast convolution of the image with a small size kernel. In addition, they locate the object boundaries much more precisely than the region based approaches. Their disadvantage however, has to do with their local nature: boundary-based approaches are very sensitive to image noise. In the case of a noisy image, boundary-based approaches may result in the detection of false alarms, that is spurious boundary points. Fortunately, as seen in chapter 4, a lot of progress has been performed in improving the robustness of the boundary-based approaches, by exploiting higher-level knowledge about the objects appearing in the image for example.

Boundary based approaches tend to produce non-connected object boundaries. This is the reason why the formation of closed contours required for segmenting the image is not feasible using boundary based methods, and a post processing stage is applied for forming such contours out of the non- connected boundary points. There are two major trends in which approaches incorporating a processing stage for connected boundary finding can be categorized: edge-following or linking, and whole boundary approaches.

Edge-linking, or *edge-following* techniques ([6], [101]) attempt creating closed contours out of the detected edge points by finding the edge pixel which more likely follows a particular edge pixel in the closed object contour. Their drawback is their relatively limited robustness, since they employ user-defined, application-dependent thresholds to guide the process. According to the *whole-boundary* methods, shape knowledge about the boundaries of the image regions is employed to achieve increased robustness of the post-processing process. In the context of these methods, the overall shape of the boundaries of the image regions is modeled by means of parametric closed curves such as energy minimizing snakes [85], shape constrained deformable models [149], [38], or level-sets of higher dimensional functions [154], [105], [118]. Curve parameter determination is achieved in two steps: Firstly the parameters of the model curves are initialized. Secondly, the parameters are *refined* by means of a non-linear iterative minimization of a cost function expressing the discrepancy between the model curves and the

boundary points of the image. The problem of whole boundary methods is that their success heavily depends on the parameter initialization step. In addition, they tend to diverge when large gaps between boundary points in the image exist. In many cases, the post processing stage for boundary finding combines the whole-boundary approaches with the edge-linking approaches as follows: An edge-linking operation is used to give a rough initialization of the region boundary, which is subsequently updated using a whole-boundary method. As we will see later, this is the way in which robust boundary finding is carried out in our system.

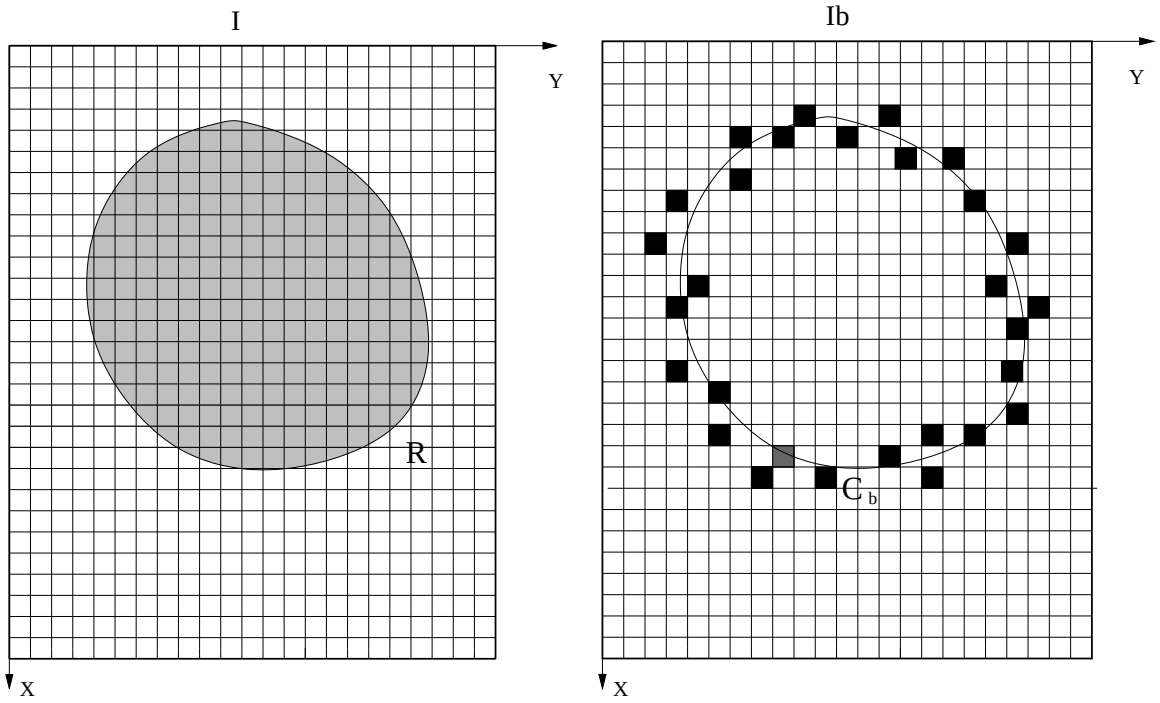
From the discussion performed in the preceding paragraphs, we can easily draw the conclusion that segmentation approaches combining the region-based and boundary-based strategies will be superior to using each in isolation, because then the limitations of the approaches are removed by the combination of their advantages. Segmentation frameworks combining the region-based and boundary-based approaches will be insensitive to image noise and will operate quite satisfactory when the image contains structures with fuzzy boundaries. At the same time, they will have increased ability for accurately localizing the regions in the image, and reconstructing their shape. In the section that follows, we will present some of the most widely known methods attempting to integrate the region based and the boundary based strategies for image segmentation and we will highlight their advantages and problems.

5.2.1 Related work

According to the majority of the approaches attempting image segmentation by incorporating both region and boundary based information, a discontinuity detector is firstly applied to the input image I, resulting in the generation of the boundary image I_b. Depending on the edge detection process applied and the needs of the segmentation process I_b may be a gradient image, a binary edge image (*edge-map*), or, as we will see later, the output of a transformation applied on these images. Subsequently, the segmentation process takes place, in the context of which information contained on both I and I_b is utilized. If so, we could roughly subdivide such systems in two categories: Those which utilize boundary information for *assisting* the region based segmentation, and those according to which boundary and region based segmentation play an *equally important* role in the segmentation process.

In the context of the former category, boundary information assists the decisions which have to be taken by the various constituting parts of a region based segmentation process. To exemplify, we consider the region based recover-and-select framework, and more specifically the *classify-and-fit* stage implemented by means of region-growing. There, decisions have to be taken regarding the inclusion of image points into a region, or for stopping the region growth. The boundary pixels in this case may give valuable hints about performing the particular operations or not, as in [48], [169], [58], [68]. Though this way of combining information may clearly improve the output of the region-based segmentation, the boundary information plays a secondary role. A much more robust segmentation result is expected to be obtained when region and boundary based information are handled in a commensurate way by the segmentation process.

Unfortunately, achieving the task is not easy: As discussed in section 5.1.1, in order to exploit region information for image segmentation, we need to know the process generating this information. Given that region information resides in the range image I, the process



(a) Region image (b) Boundary image

Figure 5.5: Sources of information for image segmentation

which generates this information is expressed by the likelihood of I. Similarly, boundary information resides in the boundary image I_b, and its exploitation within a segmentation framework requires the calculation of the likelihood of I_b. If so, if we want to handle both boundary and region information in a commensurate way for performing segmentation, we need to compute the *joint* likelihood of I and I_b, which in general is not straightforward to do.

To illustrate, we consider the figure 5.5. Fig. 5.5 (a), shows the set of pixels of the (range) image I, belonging to the unique object in the image. These pixels are shaded in the figure. Fig. 5.5 (b), shows the edge map I_b obtained via edge detection on I. A pixel $\mathbf{x}$ of the map has the value 1 only if the edge detector determined that it lies on an object's boundary. The mechanism for generating the region of fig. 5.5 (a), is given in eq. (5.3), corresponds to the region's likelihood, or equivalently, the image's likelihood, since there is only one region in I, and is rewritten here for easing the reader:

$$P(I|\mathbf{p_r}) = \prod_{\mathbf{x}:\mathbf{F}(\mathbf{x})=1} \frac{1}{\sigma_r\sqrt{2\pi}} \exp - \left(\frac{S(\mathbf{p}; I(\mathbf{x}))^2}{2\sigma_r^2} \right), \tag{5.21}$$

where $\mathbf{p_r} = (\mathbf{p}; \mathbf{F})$ the parameters of the region generating process, $\mathbf{x}(x, y)$ an image pixel, and σ_r the standard deviation of the noise process.

We can now derive a similar expression for the likelihood of the boundary map I_b. The likelihood of I_b can be computed by considering that the image is an additive white Gaussian noise corrupted version of an ideal boundary image, containing a piecewise-smooth boundary

curve, the implicit equation of which is $C_b(\mathbf{p_b}; \mathbf{x}) = 0$, where $\mathbf{p_b}$ the parameters of the curve, and $\mathbf{x}(x, y)$ an image pixel on the curve. If so, the likelihood of the boundary image will be the product of the likelihoods of the detected edge pixels, the values of which in the edge map I_b is one:

$$P(I_b|\mathbf{p_b}) = \prod_{\mathbf{x}:I_b(\mathbf{x})=1} \frac{1}{\sigma_b\sqrt{2\pi}} \exp - \left(\frac{C_b(\mathbf{p_b}; \mathbf{x})^2}{2\sigma_b^2} \right) \tag{5.22}$$

In the preceding equation we denote as σ_b the standard deviation of the boundary noise process.

Both region and boundary based processes expressed by eq. (5.21), and eq. (5.22) respectively, are similarly distributed. Besides, the parameter vectors of the processes are related since they both concern the same image region. Despite this, combination of these processes into one via computation of the joint likelihood of the two information sources is in general a daunting task [27], [28], because the processes have different views of the world. The region based process focuses on the pixel *values* $I(\mathbf{x})$ in the image, while the boundary based process on their *position* $\mathbf{x}$ on the image plane.

The majority of methods attempting to integrate region and boundary based information, [173], [119], [47] [55], [20], [37], [104], [83], usually form a giant objective function by constructing a *weighted sum* of the two likelihood functions of eq. (5.21), and (5.22). Subsequently, they simultaneously estimate both region and boundary parameters by maximizing this function. In this way, they avoid computing the joint likelihood. Such techniques will be hereinafter referred to as *single objective* techniques, and as observed in [23], while sometimes may provide constructive solutions, they often have the problem of inadequate analytic and computational tractability. In addition, the solution delivered by maximizing the benefit function of the single objective approaches may be sub-optimal [100]. Finally, as stated in [27], [28], the generated solution is very sensitive to even small variations in the assumptions on the properties of the data generating processes. Hence, the robustness of single-objective approaches is questionable. In the section that follows, we introduce an integration framework which achieves both robustness and computational efficiency when commensurately combining information sources, by neither computing the joint probability distribution, nor using the single-objective approach.

5.2.2 Game- theoretic integration

In [27], [28], the solution to the problem of integrating region and boundary based information within the image segmentation framework is inspired by *game theory*. The segmentation task is carried out by two *modules*, acting like individual *players* within a *two-person*, *non-zero sum*, *non-cooperative* game framework. The players, take decisions on two separate strategy (parameter) spaces, each related to a different information source. We assume that $\mathbf{P_r}$, and $\mathbf{P_b}$ are the strategy spaces of the first and second player respectively. Two cost functions $C_r : \mathbf{P_r} \times \mathbf{P_b} \rightarrow \Re$, and $C_b : \mathbf{P_r} \times \mathbf{P_b} \rightarrow \Re$ are associated to the two players. Segmentation is conducted by letting the players minimize their cost functions in an iterative way. More formally, considering k to be a time index, this procedure is as follows:

$$\mathbf{p_r}_{k+1} = \arg\min_{\mathbf{p_r}} C_r(\mathbf{p_r}; \mathbf{p_b}_k) \tag{5.23}$$

$$\mathbf{p_b}_{k+1} = \arg\min_{\mathbf{p_b}} C_b(\mathbf{p_r}_k; \mathbf{p_b}) \tag{5.24}$$

where $\mathbf{p_r}_i \in \mathbf{P_r}$, $\mathbf{p_b}_i \in \mathbf{P_b}$, and $(\mathbf{p_b}_1; \mathbf{p_r}_1)$ is the starting point of the minimization process. The expressions (5.23),(5.24), imply that the input of the cost function of each module is the output of the other. However, a module has no authority to change the output of the other module, but only to use the information regarding the other's output to update its own.

The iterative minimization of the cost functions in (5.23),(5.24) continues, until the decision makers cannot improve their positions, that is further minimize their cost functions. This natural stopping point is termed as the *Nash Equilibrium*. A pair of parameter sets $\mathbf{p_r^*} \in \mathbf{P_r}$, $\mathbf{p_b^*} \in \mathbf{P_b}$, constitutes a Nash equilibrium solution to the game when for every $\mathbf{p_r}$, $\mathbf{p_b}$ the following holds:

$$C_r(\mathbf{p_r^*}, \mathbf{p_b^*}) \leq C_r(\mathbf{p_r}, \mathbf{p_b^*}) \tag{5.25}$$

$$C_b(\mathbf{p_r^*}, \mathbf{p_b^*}) \leq C_b(\mathbf{p_r^*}, \mathbf{p_b}) \tag{5.26}$$

A Nash equilibrium is stable when it can be obtained as the limit of $\mathbf{p_r}_{k+1}$, $\mathbf{p_b}_{k+1}$ shown in equations (5.23) and (5.24) respectively, for any initial point. Furthermore, it is locally stable when convergence is valid for all initial conditions in some ϵ-neighborhood of the equilibrium.

In [27], [28], the first player is assumed to be a region-based module, and the second a boundary-based module. The Bayesian framework is employed in order to derive the cost function of each module. The cost function is designed to be proportional to the posterior probability of the parameters of each module, given the source of information on which each acts, that is, the region image for the region module and the boundary image for the boundary module, and the parameters of the other module. This implies that the parameters of the other module are used as *prior* terms in the cost function of the particular module. Consequently, the resulting cost functions take the form:

$$\begin{aligned} C_r(\mathbf{p_r}; \mathbf{p_b}) &= c_b(\mathbf{p_b}) + \alpha c_{r,b}(\mathbf{p_r}, \mathbf{p_b}) \\ C_b(\mathbf{p_r}; \mathbf{p_b}) &= c_r(\mathbf{p_r}) + \beta c_{b,r}(\mathbf{p_r}, \mathbf{p_b}) \end{aligned} \tag{5.27}$$

The advantages of employing the Bayesian framework for deriving the cost functions are mainly two: Firstly, it results to representing the cost functions C_r, and C_b, as a weighted sum of terms. This structure allows for the construction of an inequality condition involving the user defined parameters α and β, which when satisfied guarantees the convergence of the the iterative process of equations (5.23),(5.24) to the Nash equilibrium. Secondly, it gives the flexibility to incorporate other kinds of relevant knowledge, shape curvature information for example, if available.

Performing fusion of region and boundary based information within a game-theoretic framework, shows a variety of advantages: Firstly, as implied by the expressions (5.23),(5.24), the

boundary and region based information are equally taken into consideration by the segmentation process. Secondly, allowing the modules to act on different views of the world, we do not any more need to construct the joint likelihood in order to consider the contribution of sources in data generation. Information integration is carried out here in a straightforward way: For any one of the modules, the results of the decisions of the other module is used as priors. Furthermore, this approach of information integration is proved to be more general than the single-objective approach: As shown in [100], when the probability spaces of the modules are identical, the equilibrium solution becomes equivalent to that of the single objective approach. Besides, the benefit of adopting a decoupled scheme for integration, is computational efficiency, since each module deals with a subset of the entire parameter space in its own view of the world. Finally, robustness is achieved since each module tends to pull the other away from noise and local minima on one hand, while pushing itself towards the equilibrium on the other.

Our approach for box-like superquadric segmentation integrates boundary information in the segmentation process as proposed by the game-theoretic framework described in the preceding paragraphs. Inputs of the process are the range image, and a boundary image obtained from the range image via edge detection on the former. Image segmentation is performed by two decoupled but cooperating modules: a region module and a boundary finding module. The region module is a region growing process which recovers the superquadric parameters using information of the range image, and is influenced by the information delivered by the boundary module. The boundary module, recovers boundaries of exposed surfaces of box-like superquadric objects, using information of the boundary image, and is influenced the information delivered by the region growing module. Incorporation of boundary information into the superquadric segmentation process solves the region overgrowing problem, and contributes to the creation of a very robust segmentation framework. In addition, letting each process to deal with each source of information independently, accelerates both the recovery of superquadric parameters and the overall segmentation task.

5.3 Incorporation of boundary information for Superquadric segmentation

Our approach for box-like superquadric segmentation extends the recover-and-select framework by incorporating boundary information into the segmentation process. As the recover-and-select strategy, our approach adheres to the hypothesis generation and refinement framework: In the context of the hypothesis generation stage, boundary information is used to determine the number of objects in the image and the image pixels in the interior areas of the objects, in which superquadric models are fitted. In this way an initial approximation or *hypothesis* about the values of segmentation parameters, that is, a rough image segmentation is obtained. In the hypothesis refinement stage, this hypothesis is refined and a more accurate segmentation is acquired. Boundary information is integrated in this stage in a way inspired by the game theoretic framework, of section 5.2.2: Parameter refinement is realized by means of iterative invocations of two independent parametric modules in succession: The region module, fits superquadric models to image regions assumed they belong to unique objects. The boundary module fits the boundary of the exposed surfaces of the

superquadric models to the edge points in the boundary image. Hence, each module acts on its own information domain, or in other words, dealing with boundary related information is *decoupled* from region information and vice versa. Information integration is achieved by simply letting each module to pass the parameters of the refined models to the other, after its execution. The hypothesis refinement process ends when no significant change in the values of the model parameters can be performed.

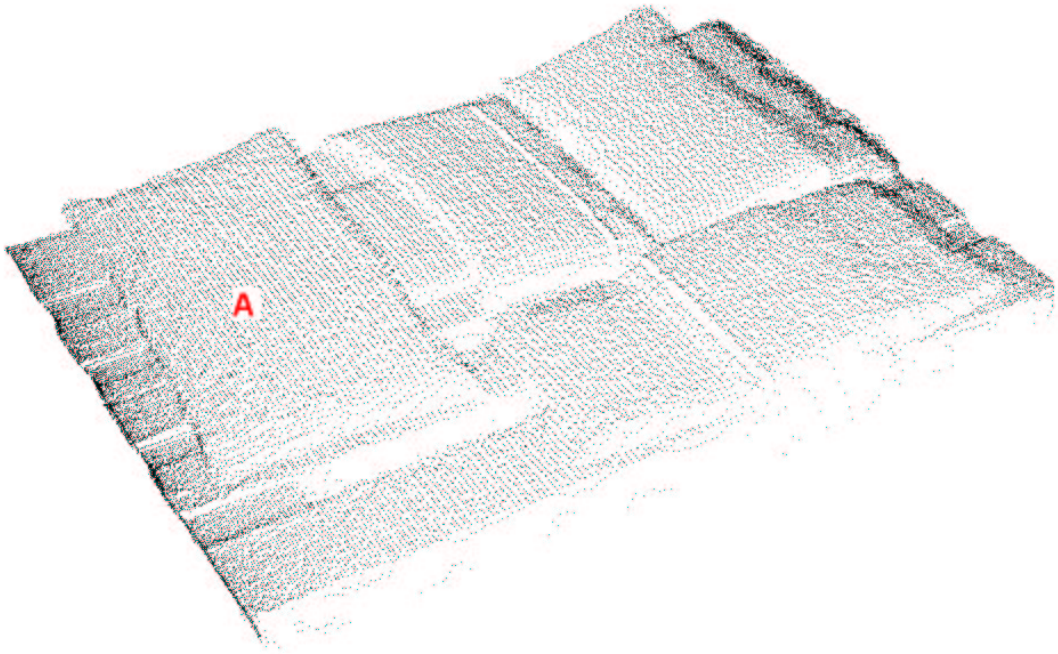
Boundary information is generated by an edge detection process, via approximation of the scan lines of the input range image I with linear segments, as discussed in chapter 4 of this work. The output of this process is an *edge image*, which contains the three dimensional points on the boundary of the exposed surfaces of the objects in I. Given the range image of fig. 5.6 (b), which corresponds to the intensity image of fig. 5.6 (a), edge detection on the former outputs the range image comprising the boldly marked points of figure 5.7 (a). Fig. 5.7 (b), shows the *edge map* corresponding to the edge image of fig. 5.7 (a). The edge map I_b is a two dimensional binary image, the value of a pixel $\mathbf{x}(x, y)$ of which is 1, that is $I_b(\mathbf{x}) = 1$, only if the corresponding range point $I(\mathbf{x})$ is an edge point. In short, the edge map contains information related to the orthogonal projection of the boundaries of the objects' exposed surfaces on the image plane. The difference in the information contained in the edge image and the edge map is that the former incorporates information related to the *depth* of the edge points. But since depth information can be derived by the range image, the depth information in the edge image is redundant.

We employ the edge map instead of the edge image for deriving boundary information. There is a number of reasons which led us to take the particular decision: Firstly, usage of the edge map rids this redundancy in information representation. Secondly it adheres to the philosophy of the game theoretic information integration, which suggests that the modules should act on information domains which do not interfere. Finally, using the edge map we end up dealing with two- instead of tree- dimensional information, which considerably increases the computational efficiency of our system.

Note, that as discussed in chapter 4, the edge map contains boundary points on the exposed surfaces of the box-like objects. Hence, using this kind of information within the segmentation process results in the recovery of objects which are exposed to the laser source, or better, of graspable objects. In short, our approach does not produce a complete image segmentation, but rather isolates the image regions corresponding to the graspable objects in the image. This should be by no means considered as drawback, since our target is usage of the approach for robotic grasping purposes.

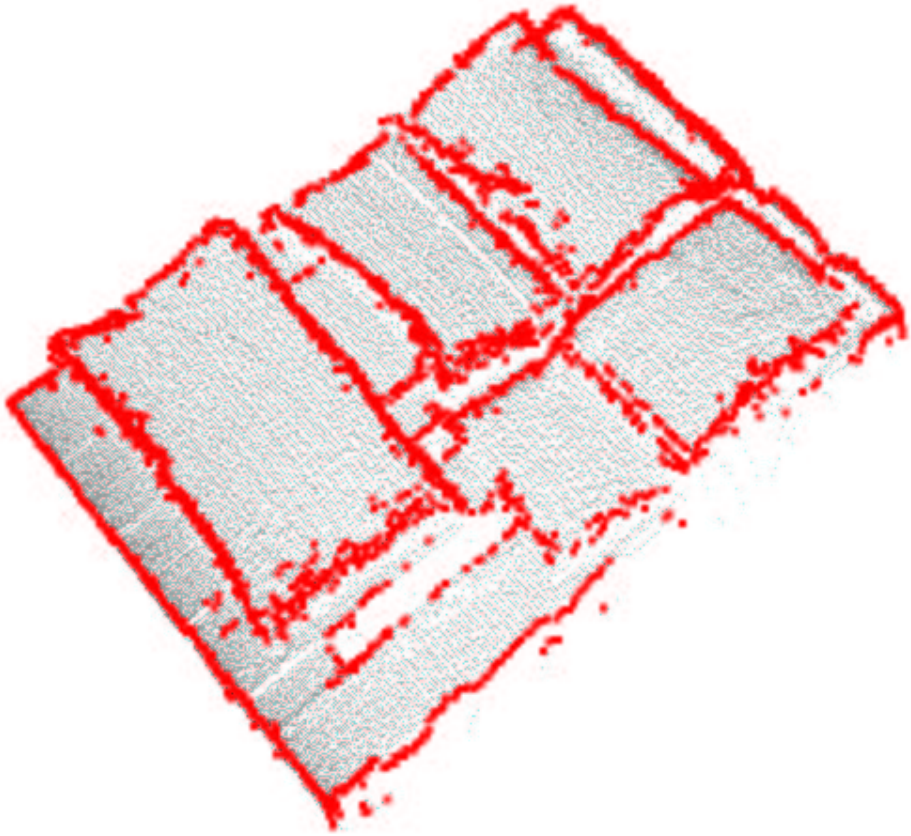
The flow diagram of our approach is illustrated in Fig. 5.8, and comprises two stages: Firstly, edge detection is performed in the input range image I, by means of the scan line approximation technique, in a way explained in chapter 4. Output of the edge detection is the edge map I_b. Secondly, the main part of the segmentation is initiated, bounded by the dashed box in fig. 5.8. Both I and I_b are employed by this stage. Output of the hypothesis generation process is a set of superquadric models in the range image. If $\mathbf{P}$ denotes a vector containing the parameters of all these models, the hypothesis refinement stage updates $\mathbf{P}$ to obtain the set of models $\mathbf{P}^*$, the output of our segmentation process.

(a) Intensity image

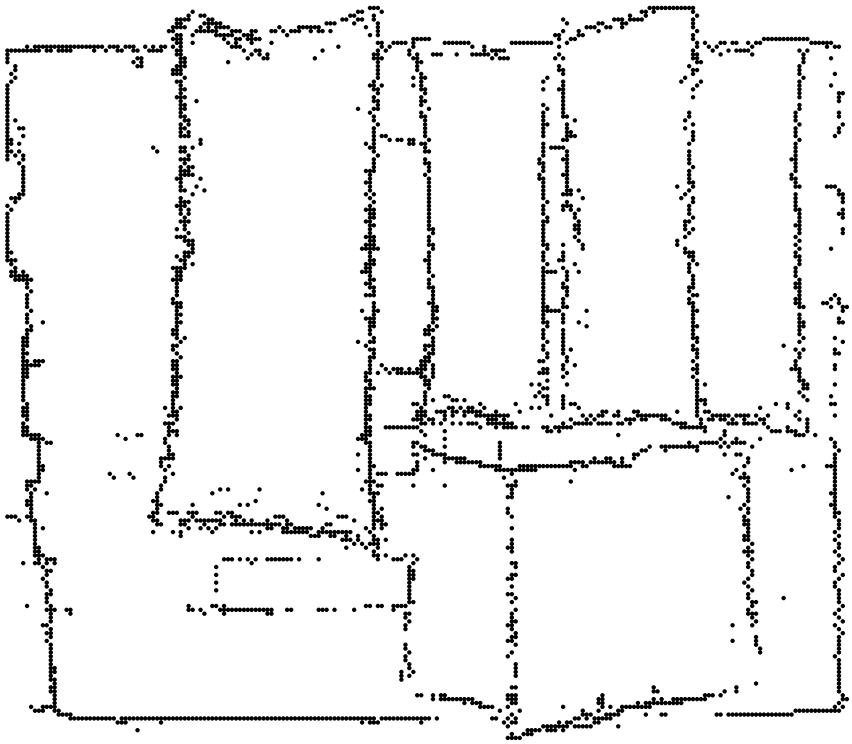


(b) Range image

Figure 5.6: Box-like object configuration



(a) Edge image



(b) Edge map

Figure 5.7: Edge detection results

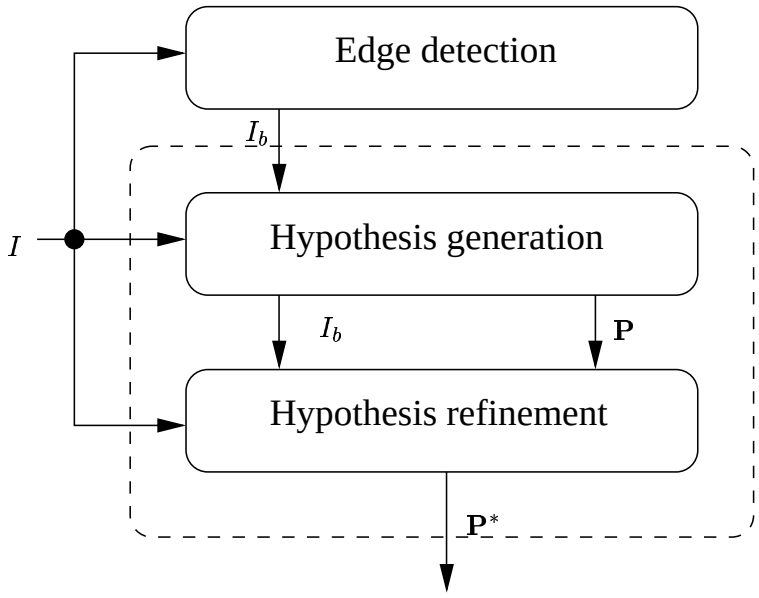


Figure 5.8: Our approach for superquadric recovery

In the paragraphs that follow we discuss in detail our approach for segmenting images of box-like superquadric objects. Firstly, we present the way in which the seed placement is performed in our input images. We then illustrate the internals of the hypothesis refinement stage of our segmentation framework. Finally, we present experiments in which the robustness, and computational efficiency of our approach is demonstrated. The experiments conducted in this paragraph are continued in chapter 6, where we give the reader the opportunity to obtain a much more clear impression on the effectiveness of our approach.

5.3.1 Hypothesis generation

Target of the hypothesis generation, or seed placement process, is an initial estimation of the segmentation parameters, that is, of the number objects in the image, the image pixels belonging to the region of each object, as well as the region parameters, that is the parameters of the models associated to the regions. The closer the segmentation parameter set delivered by the seed placement process lies to the actual segmentation parameters, the most likely it is, that the subsequent parameter refinement procedure will succeed in recovering them. Hence, an effective seed placement procedure should initialize a number of seeds (almost) equal to the objects contained in the image, its boundaries should be as similar as possible to the actual object boundaries, and the region parameters should be as close as possible to the actual region parameters.

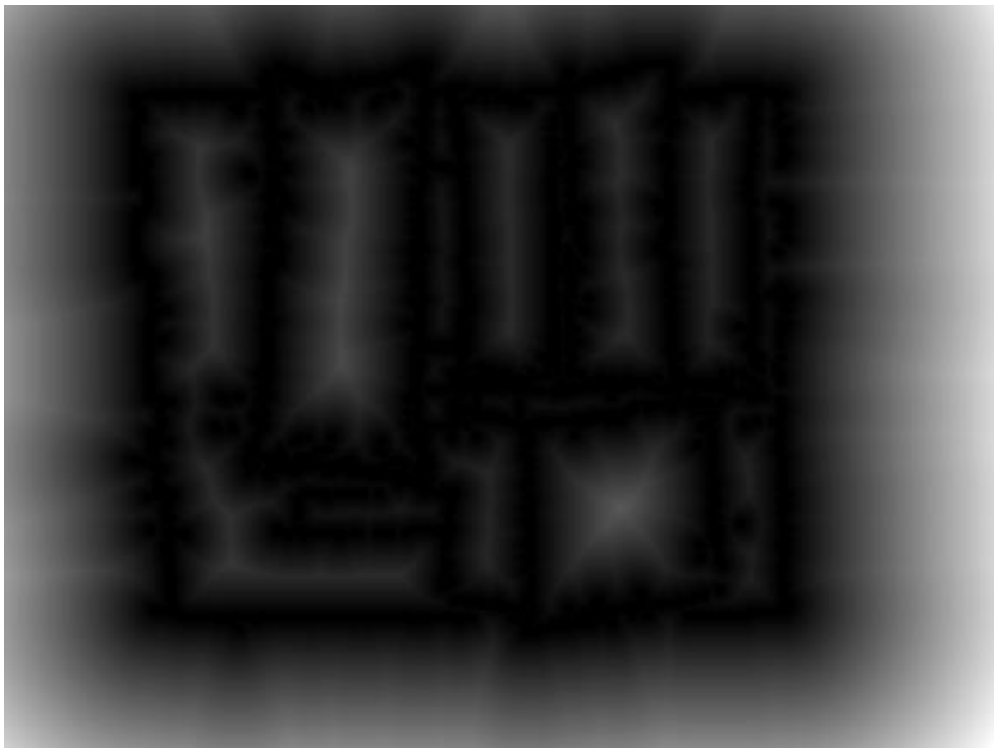
A straightforward consequence of the requirement that the seed parameters lie as close as possible to the actual region parameters, is the placement of seeds *inside* the true regions of the objects. In this case, seed parameter estimation via maximum likelihood, will be based on image points guaranteed to belong to the region of a single object, and thus will reliably describe the object. However, knowledge of the true regions of objects in the image can be obtained only after segmenting the image. An alternative way in which we may determine

these regions is to utilize information contained in the edge map. The sought regions are areas of the edge map surrounded by boundary contours. Hence, a sound strategy for seed placement is to consider the edge map, form closed contours and assign the seeds to the image regions enclosed by the contours.

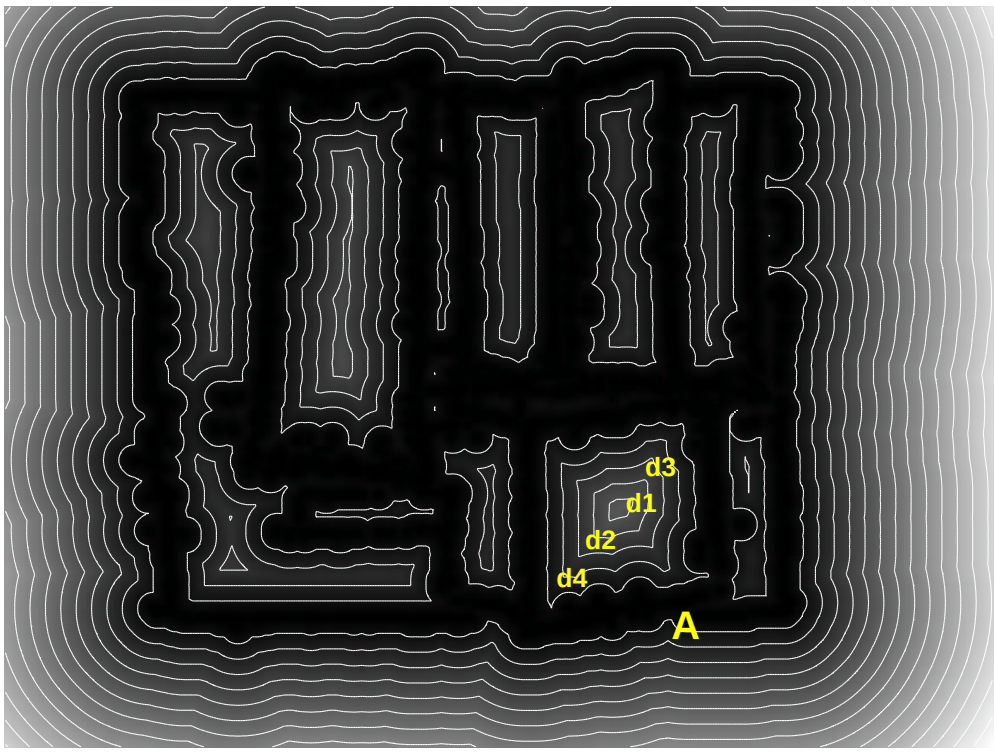
This strategy is employed by many approaches for seed generation [110], [14], [145], [143]. Such approaches, firstly apply a *distance transform* to the input edge map. Output of this transform is a two dimensional image, each pixel of which contains the distance of the particular pixel to the nearest boundary pixel. The contours in this image along which the distance has a constant value (*iso-distance contours*), enclose image areas inside boundaries of objects by definition. Hence, these areas are candidate seed regions. Given the edge map of fig. 5.7 (b), the *Euclidean distance transformed* image (EDI), is shown in fig. 5.9 (a). In this image, brighter intensities indicate pixels having higher euclidean distances from boundary pixels. Figure 5.9 (b) shows the iso-distance contours superimposed on the transformed image. In this figure, four iso-distance contours are shown, namely the d_1, d_2, d_3, d_4, the regions inside each of which is a candidate seed for the object A. The main problem that the seed placement approaches face, is how to select the most appropriate seed from this set of candidate seeds.

Usually, seed determination is performed using a user-defined threshold expressing the minimum allowable distance of the iso-distance contour to the object boundaries. Determining the value of the threshold however is not an easy task: The larger the threshold is, the bigger is the distance of the seed to the object boundaries. Using large threshold in the case ensures that no points belonging to neighboring objects are included in the seed. However, the further away from the object boundaries the seed lies, the smaller its size. The problem is that we wish to generate seeds containing as many pixels as possible, because the more the points associated with the seed the more reliable the initial estimation of region parameters will be. As far as our application is concerned, after extended experimentation we observed that it may be possible to determine the desired threshold when objects of the same size appear in the image. In the case that the size of the objects varies, seed placement by thresholding the distance transformed image of the edge map proved not to be robust. This led us into adopting an alternative approach for seed placement, which is based on mathematical morphology.

Our approach for seed placement is inspired by the technique reported in [78]. In this work, formation of closed boundary contours is achieved by *adaptive closure* of the edge map. The robustness of this strategy stems primarily from the fact that except for the information in the edge map, the range image as well as knowledge about the shape of the objects in the image are utilized. The strategy of [78], is based on the observation that any contour in the edge map can be closed by *dilating* [70], [39], [147] the edge map. Contour closure is performed in an iterative way: *Connected component labeling* [157] is applied to the input edge map, so that a set of regions is extracted. Usually, due to gaps in the contours of the edge map, one region contains pixels corresponding to more than one object, that is, the region is *under-segmented*. To distinguish between the correctly segmented and under-segmented regions, a test is performed to every region $\mathbf{R}$ generated by the component labeling step: The modeling element, a superquadric in our case, is fitted to the range points $I(\mathbf{x}) : \mathbf{x} \in \mathbf{R}$. Subsequently, the fitting error residual is computed as in eq. (5.8). If the fitting error is lower than a user defined threshold, it is assumed that the examined region corresponds to a unique object,

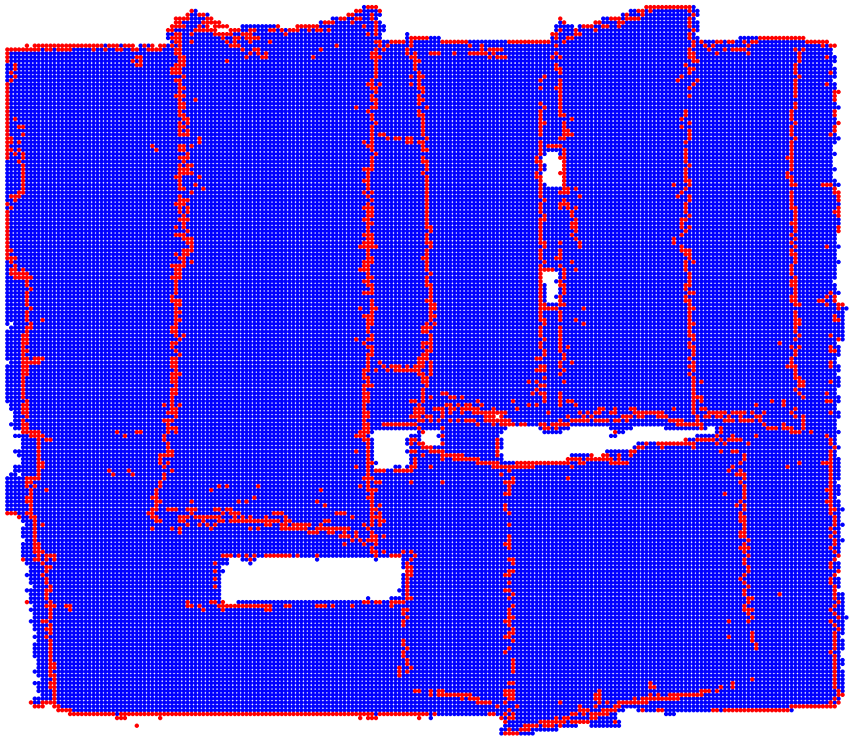


(a) Euclidean distance transformed edge map

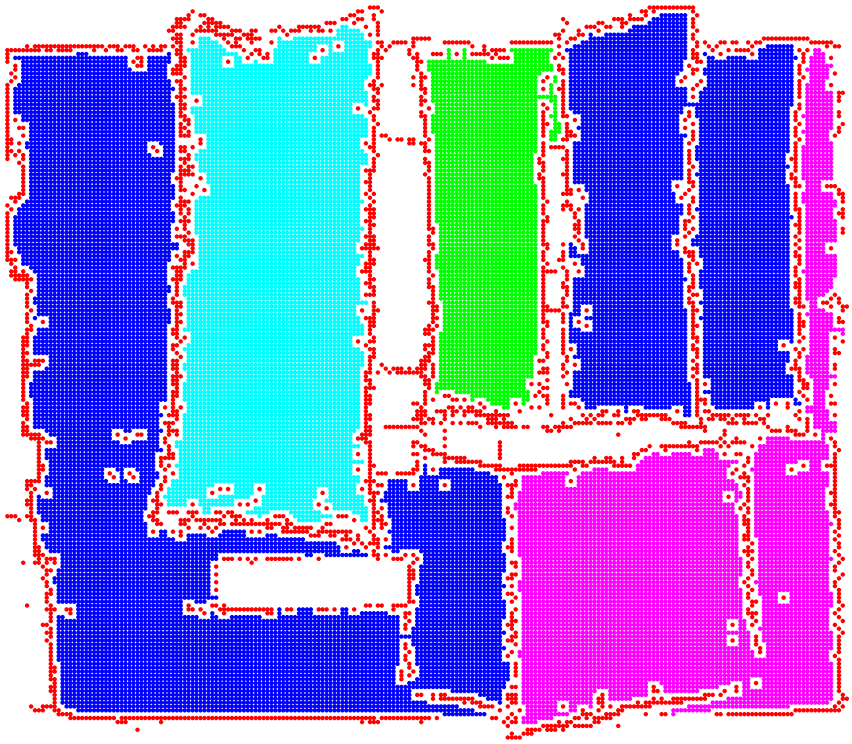


(b) Iso-distance contours on the euclidean distance transformed image

Figure 5.9: Euclidean distance transform



(a) Initial region



(b) After the first dilation

Figure 5.10: Region maps generated by the adaptive edge map closure operation

and in this case the region test is considered successful. If so, the region is placed in the list of extracted regions. This means that the superquadric parameters and the pixels of the region are saved and the region is excluded from further consideration. Unsuccessful region test implies that there still exist open contours within the examined region. In this case, one morphological *dilation* [70], [147] operation on the boundary pixels belonging to the region is performed, so that the remaining gaps in the contours are potentially closed. A component labeling step succeeds the dilation, so that new candidate regions are extracted. These are tested in the same manner as previously described. This process of component labeling followed by the region test is repeated until the generated regions have been successfully verified or they are any more regions that can be considered because their size is smaller than a used defined threshold. The output seeds of the process are the successfully verified regions.

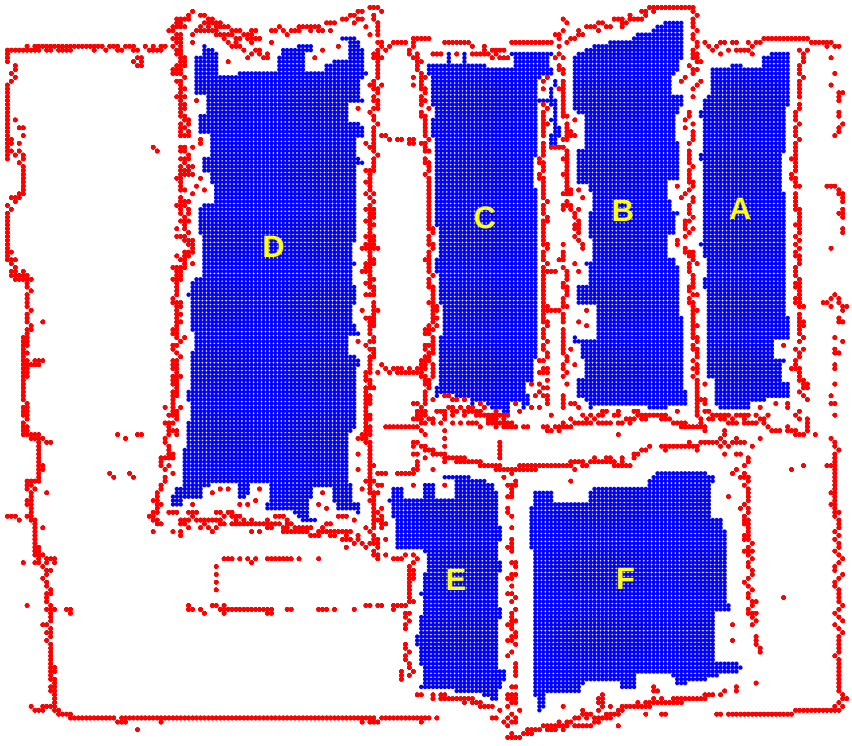
Fig. 5.10, shows two instances of application of the contour closure algorithm on the edge map of fig. 5.7 (b). Fig. 5.10 (a) shows the regions initially formed after applying connected component labeling to the edge map. The effect of under-segmentation is apparent: despite the fact that 7 objects can be viewed in the image, only 1 region is formed, due to gaps on the edge map. Fig. 5.10 (b) shows the region map after the first dilation operation. The under-segmentation is limited in this picture, since 6 regions are recovered. Application of the adaptive contour closure algorithm to each of these regions, finally results into the formation of the seed regions of fig. 5.11 (a).

The advantage of this approach for seed placement with regard to strategies performing seed placement by thresholding the distance transformed image of the edge map is robustness. In order to demonstrate this, we consider the ball-like structuring element $\mathbf{B}$ that is used by the dilation process of our adaptive contour closure strategy. Assuming d the radius of the ball, I_b the edge map, I_{bd} the euclidean distance transformed edge map, $\mathbf{C}$ the boundary pixels of the edge map, and $\mathbf{x}$ a pixel of transformed image, then according to [39], the dilation of the contour $\mathbf{C}$ using $\mathbf{B}$ as structuring element can be expressed as a threshold of the euclidean distance transformation, that is:

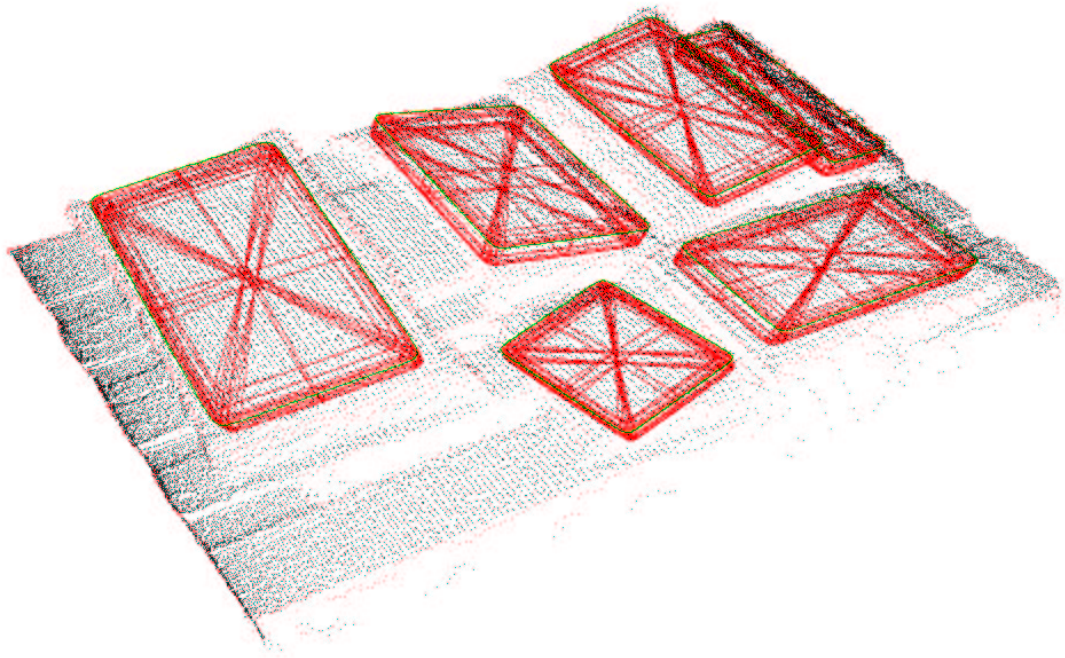
$$\mathbf{C} \oplus \mathbf{B} = \{\mathbf{x} | I_{bd}(\mathbf{x}) \leq d\} \tag{5.28}$$

In this respect, the boundary of an image region originating by a successful region test in the context of the adaptive contour closure framework *corresponds* to the iso-contour of the *minimum* distance from the object boundaries, whose interior can be satisfactory modeled by a superquadric entity. In other words, the adaptive contour closure framework offers a straightforward way for finding large seeds in the image, in which no points from neighboring objects are encountered, without the need for determining an iso-contour distance threshold, as required by the majority of methods for seed placement. The threshold needed to be determined in our case, namely the threshold deciding on the success of a region test, is much more easy to define since it mainly depends on the sensor related noise and not on the dimensions of the objects in the image.

The seeds created by the application of the adaptive contour closure approach in the edge map of fig. 5.7 are presented in fig. 5.11. More specifically, fig. 5.11 (a), shows the region map of the seeds. Fig. 5.11 (b) shows the superquadrics fitted to the range points $I(\mathbf{x})$ corresponding to the pixels $\mathbf{x}$ of each seed. A number of conclusions can be drawn by inspecting these images: Firstly, each region corresponds to a different object, and the parameters



(a) Seeds on the image plane



(b) Fitted models in $3D$

Figure 5.11: Seed placement

of the initialized models are sufficiently close to the actual object parameters. Secondly, the boundaries of the seeds are very close to the actual object contours and their shape is similar to latter. These observations imply that our seed placement procedure manages to produce an accurate initial approximation of the segmentation parameters as required. A drawback of our approach however is its relatively high computational cost, due to the iterative application of the region test. Despite this, the costs are not so high so that a real time implementation of the procedure is prohibited. This is mainly due to the fact that the edge map contains only a few gaps which means that contours can be closed within a small number of dilation operations. Seed placement took about 3 seconds on the average in a Pentium 4 PC of 2.8 GHz.

The alert reader may have already observed, that there is a region in fig. 5.11 (a) which has not been assigned to superquadric model, although it should. This is the case for the region of the object lying between and occluded by the objects C and D in the figure. The reason for discarding the particular region although it can be accurately described by a superquadric model, is that it encompasses a small number of points, and thus cannot correspond to a *graspable object*, the recovery of which is the target of our segmentation approach. Such objects fully exposed their biggest surface to the laser source, and therefore regions corresponding to such objects could not possibly comprise very few range points. Although this criterion is adequate for identifying heavily occluded non- graspable objects, it is not enough for identifying all non- graspable objects, such as the objects A, or E in the fig. 5.11 (a). In order to determine if such an object is graspable, we should additionally check if the boundary of its exposed surface is a three dimensional superellipse. Such a test is not performed by the seed generation procedure, but by the subsequent hypothesis refinement stage.

5.3.2 Hypothesis refinement

There are certain similarities between the hypothesis refinement stage of our approach and the corresponding stage employed in the recover-and-select framework for superquadric segmentation: Parameter refinement is performed *independently* for each seed. In addition, our approach consists of the same constituting elements: As in the recover-and-select framework, the classify-and-fit process performs a local refinement of the model parameters generated by the seed placement procedure. The model-selection process rejects models which describe the data inaccurately.

However, there are significant differences related to the control structure of our approach, the internal implementation of the processes, and their functionality: Boundary information is incorporated in the iterative classify-and-fit process: Each iteration of this process involves invocation of two modules in succession. The boundary module performs model parameter recovery using the edge map. The region module employs the range image for this purpose. Within each of the modules, model parameter recovery is interwoven with image pixel classification. In other words, each of the modules acts as the standard classify-and-fit process on its own information domain. In the recover-and-select framework, the model-selection is embedded in the refinement process, and is invoked quite frequently, before the end of the classify-and-fit process. Remember that the need of frequent invocations of the model selection process was the implicit solution of the region- overgrowing problem. Solution of

the region-overgrowing problem via incorporation of boundary information abolishes this need. Hence, in our case, the model selection process is invoked only once, after parameter refinement of all the models in the image has been performed, as a post-processing step. This step determines whether the model refined via the classify-and-fit process corresponds to a graspable object in the image. Note, that the classify-and-fit process does not refine all the superquadric parameters. Due to the particular object placement with regard to the sensor, not enough information able to constrain the height parameter a_3 of the box-like models can be acquired. As a consequence, this parameter is considered constant within the classify-and-fit process. An additional functionality of the post-processing step is to recover this parameter.

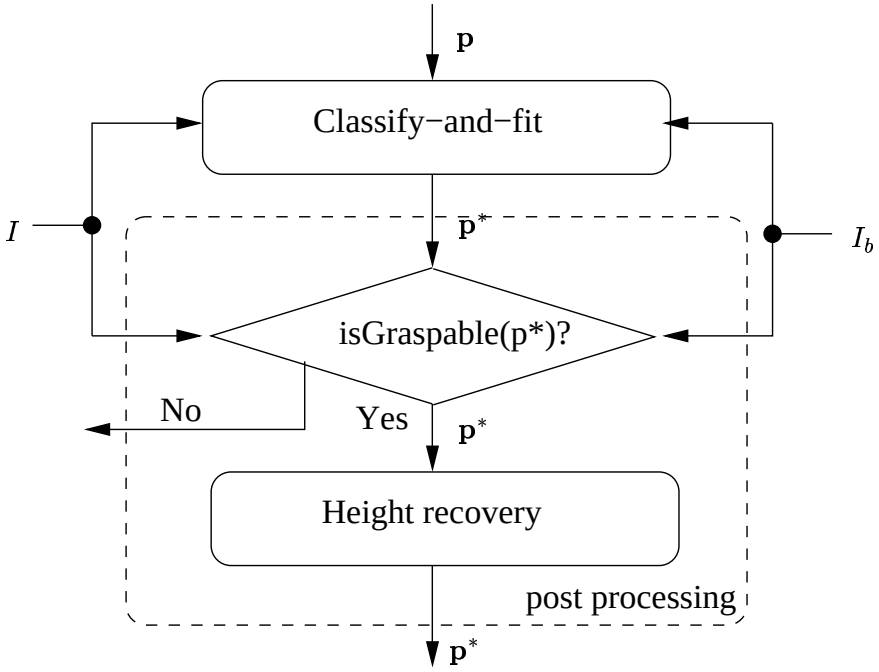


Figure 5.12: Refinement of the parameters of a single model

The flow diagram for parameter refinement of a single seed is illustrated in fig. 5.12. Given $\mathbf{p}$ the model parameter vector of the input seed, the model is firstly fitted to the input range and boundary images I and I_b respectively by means of the classify-and-fit process, to obtain $\mathbf{p}^*$. Then, the post processing step, is initiated, bounded by a dashed box in fig. 5.12. It determines whether the fitted model $\mathbf{p}^*$ corresponds to a graspable object. If yes, the height parameter of the model is determined, and the model parameter vector $\mathbf{p}^*$ is delivered as output of the refinement process. If not the model is discarded from further consideration. In the paragraphs that follow, we dive into the internals of both the classify-and-fit and the post processing stages of the parameter refinement stage.

5.3.2.1 Classify-and-fit

Input of the classify-and-fit process, is the segmentation parameters related to a particular seed created in the hypothesis generation stage. This seed corresponds to an image object,

which will be hereinafter referred to as the *object of interest* of the process. The input parameter set comprises region information, that is the image points associated to the seed, and the parameter vector $\mathbf{p}$ of the superquadric model describing these points. Target of the process is to refine these parameters, that is to finally include all the points of the object to the region and retrieve the parameter vector $\mathbf{p}^*$ of the superquadric model describing these points. In other words, target of the classify-and-fit process is to accurately segment the object of interest.

The classify-and-fit is an iterative process, the flow diagram of which is shown in fig. 5.13. The process, firstly initializes a two dimensional image, which partly incorporates region information by identifying the pixels belonging to the region of the object of interest. This image, will be hereinafter referred to as the *region image* and will be denoted as I_r. The region image is of dimensions equal to the range image. A pixel of this image has the value 1, if the corresponding pixel of the range image is determined to belong to the object of interest. Initially, the pixels of I_r set to 1, are the pixels comprising the seed.

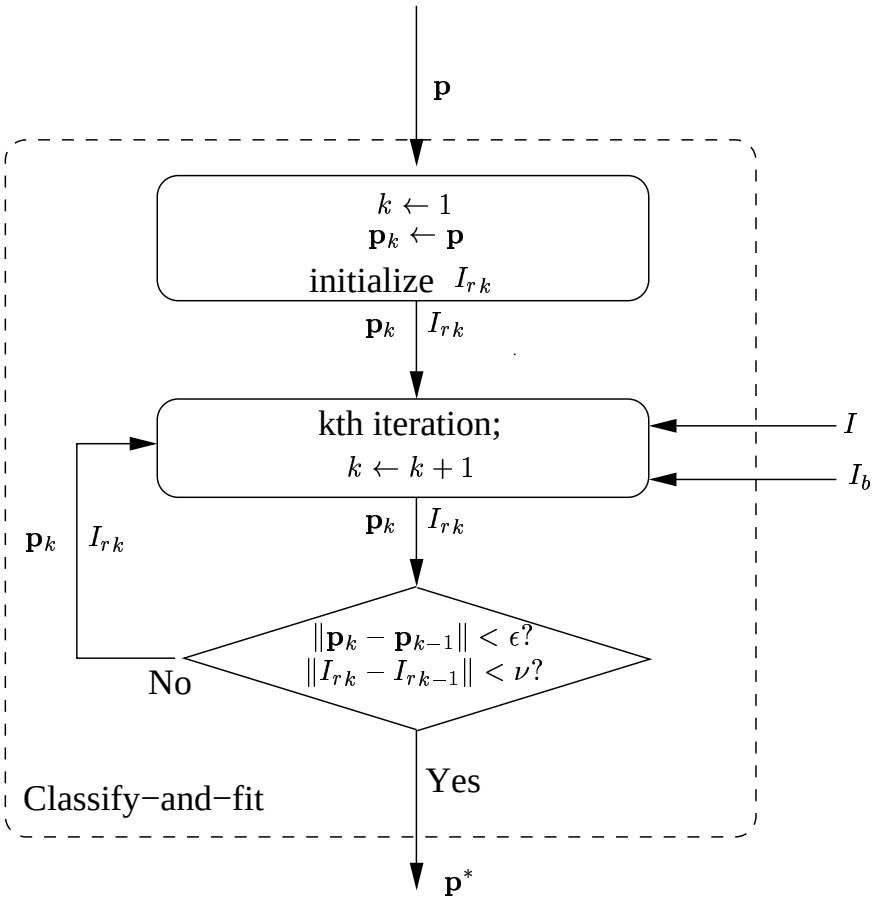


Figure 5.13: Iterative model fitting

Subsequently, the core of the classify-and-fit process starts. Each iteration, involves a sequential invocation of two modules: The boundary module updates the superquadric parameter vector by fitting the boundary of the exposed surface of the model to the edge map I_b.

The region module updates the superquadric parameters by fitting the model to the range image I. Each module recovers the parameters of the model by interweaving point classification with model parameter estimation. The process stops after its kth iteration, when two conditions hold simultaneously: The difference between the currently obtained value of the parameter vector $\mathbf{p}_k$ to the vector obtained by the previous iteration $\mathbf{p}_{k-1}$ is smaller than a user defined threshold ϵ, and the new pixels that the current iteration assigns to the object of interest, or the difference in number of pixels set to 1 between the current region image I_{rk} and the one obtained by the previous iteration I_{rk-1} is smaller than a user defined threshold ν. In this case, the current model vector is delivered as the output $\mathbf{p}^*$ of the process.

More details are exposed in the subsequent paragraphs. More specifically, the section 5.3.2.1.1 brings into attention the fact that not all the superquadric parameters need to be updated by each of the two modules of the classify-and-fit process, which results into both robustness and computational efficiency. Finally, in section 5.3.2.1.2 we describe the internals of the boundary finding module, followed by a description of the region module (section 5.3.2.1.3).

5.3.2.1.1 Decomposition of superquadric parameter recovery According to our framework, superquadric parameter recovery is performed by two modules each dealing with its own domain of information. However, not all the superquadric parameters are related to the domains on which each of the two modules acts. The reason why this happens is illustrated in figure 5.14: In this figure the box-like superquadric object A is shown. In addition, the model coordinate system $\mathbf{X_m}, \mathbf{Y_m}, \mathbf{Z_m}$ of A is shown in the image, centered in the center of gravity of A. The object is embedded in the sensor coordinate system $\mathbf{X_s}, \mathbf{Y_s}, \mathbf{Z_s}$, where $\mathbf{Y_s}$ is the depth axis of the sensor.

Assume that a region of points is determined to correspond the exposed surface of the object, the shaded patch on the exposed surface of A in the figure. Such a region is the output of our region- based module. This region constrains only a *subset* of the superquadric parameter vector: Firstly, given that the height of the object is known, the patch incorporates information related to the distance of the exposed surface of the object from the sensor, or more accurately, its distance from that the $\mathbf{X_s} - \mathbf{Z_s}$ plane. This information is expressed by the model parameter p_y. In addition, the patch constrains the rotation around the $\mathbf{X_m}$ and $\mathbf{Y_m}$ axes of the model coordinate system, which correspond to the ϕ and θ model parameters respectively. Finally, given that the patch covers a large area of the exposed surface of the object, it can be used to recover the deformations of the model, expressed by the parameters a, b, c, d. If so, region information constrains the model parameter subset $\mathbf{p_r}(p_y, \phi, \theta, \alpha, b, c, d)$ sufficiently. Hence, the region module should be allowed to recover only this particular model parameter subset.

Given the region related model parameter subset $\mathbf{p_r}$, the boundaries of the exposed surface of the object in the image plane, shown as dots in figure 5.14, can be used to accurately determine the dimensions of the exposed surface of the object expressed through the model parameters a_1, a_2, the position of its center of gravity in the image plane expressed through the parameters p_x, p_z, as well as the rotation ψ around the $\mathbf{Z}_m$ axis of the model coordinate system. Note, that in the event that region information is only available, the values of

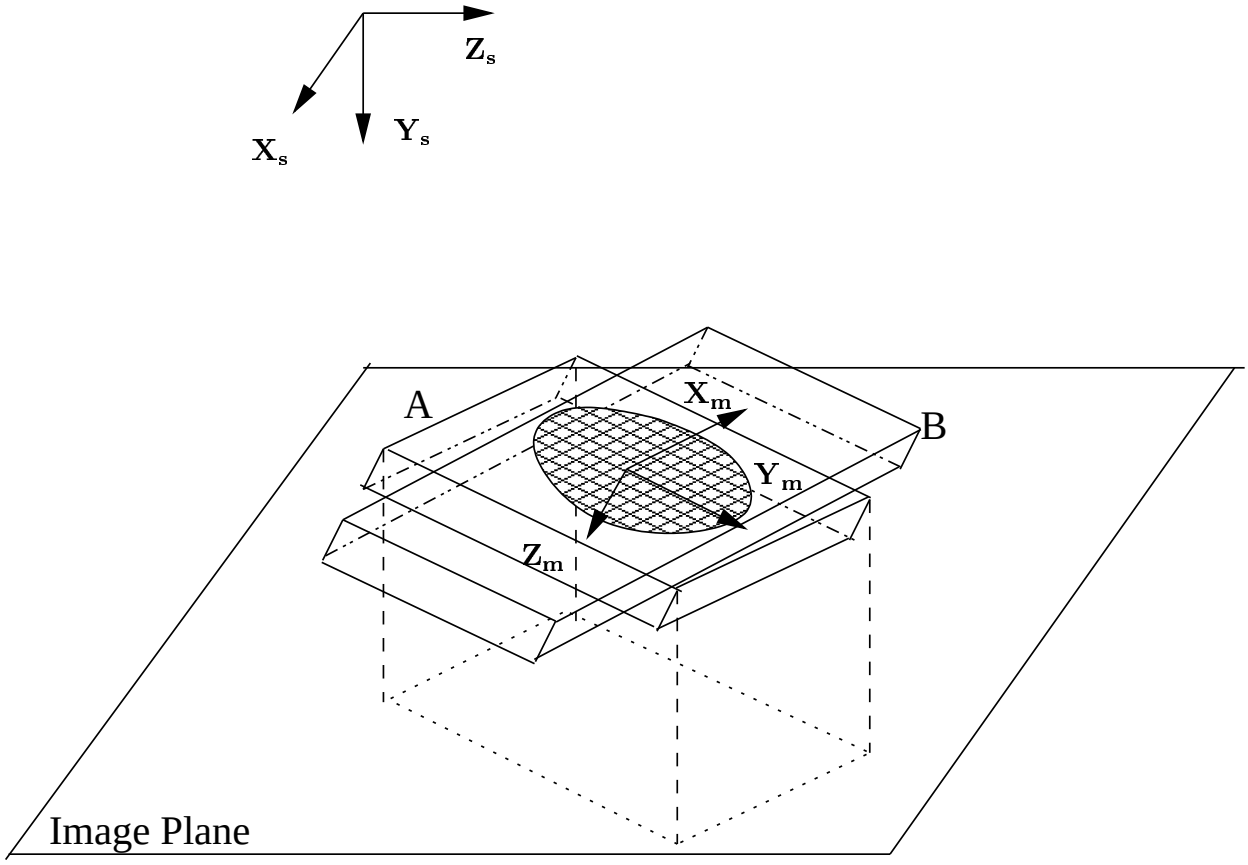


Figure 5.14: Ambiguity in model parameter determination from region information

these parameters cannot be sufficiently constrained: The object B in the figure for example, has different values for all the parameters $(a_1, a_2, p_x, p_z, \psi)$ than A, but fits the patch exactly as well as the object A. This demonstrates again the necessity of incorporating boundary information for accurate superquadric recovery. In the following, we will denote as $\mathbf{p_b}(a_1, a_2, p_x, p_z, \psi)$ the subset of the superquadric parameter vector which can be sufficiently determined by boundary information. The boundary module of our segmentation framework refines only the particular parameter subset.

As far as the remaining superquadric parameters are concerned, the shape parameters ϵ_1, ϵ_2 do not vary a lot, since all our target objects are box-like. Extensive experimentation with the target objects, showed that their value ranges between 0.1 and 0.3. For this reason, we kept the shape parameters constant to $\epsilon_1 = \epsilon_2 = 0.2$. An additional reason for not involving the shape parameters in the recovery process, is that the analytical calculation of the derivatives of the cost functions with respect to those parameters is a very tedious task, on one hand, the expressions derived contain long terms whose evaluation is computationally costly on the other. Finally, the parameter a_3, that is the size of the superquadric along the $\mathbf{Z_m}$ axis, is as well considered constant in the recovery process. Recovery of a_3 is performed separetely, within the post processing step. The model parameter vector subset remaining constant within the parameter refinement process will be hereinafter denoted as $\mathbf{p_s}(\epsilon_1, \epsilon_2, a_3)$

Decomposing the recovery of the model parameter vector results into a search for the optimal parameters in lower dimensional parameter spaces. As a consequence both the robustness and computational efficiency of the recovery process increase.

5.3.2.1.2 Boundary module: Boundary finding influenced by region information. The boundary module refines the parameter subset $\mathbf{p_b}$ of the model corresponding to the object of interest, using two dimensional boundary information. Assume that we are in the the kth iteration of the classify-and-fit process. In this case, the boundary module is about to be invoked for the kth time. Inputs of the module is the edge map I_b, and the vector $\mathbf{p_b}_k$, which is output of the $(k-1)$th invocation of the module. Besides, inputs of the module are the outputs of the $(k-1)$th invocation of the region module, that is, the model parameter subset $\mathbf{p_r}_k$, as well as the the region image I_{rk}. Output of the kth invocation of the boundary module is the updated superquadric model parameter subset $\mathbf{p_b}_{k+1}$. Figure 5.15, illustrates. The vector $\mathbf{p}_k = (\mathbf{p_s}, \mathbf{p_r}_k, \mathbf{p_b}_k)$ denotes the entire model parameter set before the kth invocation of the boundary module. For the sake of simplicity in the formulations that follow, the usage of the time indices k, $k+1$ will be deferred until the concluding paragraph of this section.

The way in which refinement of $\mathbf{p_b}$ is implemented, is inspired by [29] and [149]. In these works, the Bayesian framework is used in order to derive an intuitive way in which except for parameter refinement, information integration is as well performed. According to [29], [149] $\mathbf{p_b}$ should be estimated by maximization of the posterior probability of the entire model parameter set $\mathbf{p}$ given the input images I_b, and I_r, with respect to $\mathbf{p_b}$. This is equivalent to the maximization of the function L, which amounts to:

$$L(\mathbf{p}, I_b, I_r) = L_{prior}(\mathbf{p}) + L_b(\mathbf{p}, I_b) + L_r(\mathbf{p}, I_r) \tag{5.29}$$

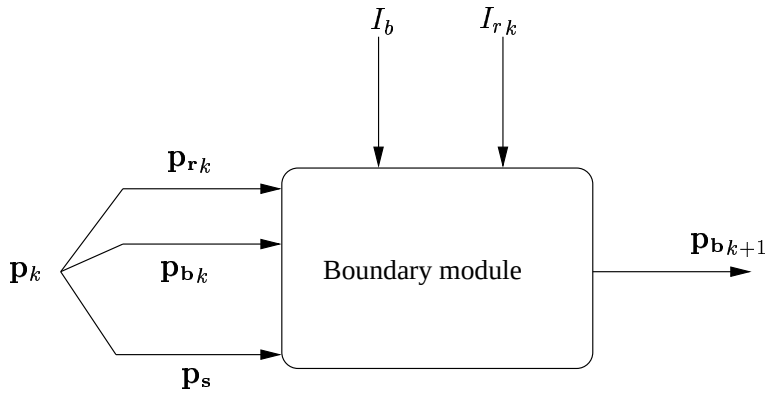


Figure 5.15: The kth invocation of the boundary module

The first term of the right part of the benefit function corresponds to the logarithm of the prior probability distribution of the model parameter vector. The second term corresponds to the log-likelihood of the edge map. Maximization of this term results into fitting the model's boundary of exposed surface (or model's *boundary*) to the edge map. Finally, the last term corresponds to the log-likelihood of the region image I_r. Maximization of this term results into fitting the model's boundary to the boundary of the region in the region image, and its inclusion in the benefit function, integrates region information within the boundary module. Adding a term accounting for the region information into the benefit function, seems like employing the single objective approach discussed in section 5.2.2 for parameter refinement. The difference between the objective function of our case and the objective functions involved in the single objective approaches is that all terms involved here deal with homogeneous information, that is two dimensional boundary information.

The likelihood of the edge map I_b, is computed under the assumption that the entire image is a noise corrupted version of the boundary of the object of interest. This does not hold in general, because the edge map contains boundary points originating from an unknown number of objects. However, we assume that this assumption holds *locally*, i.e. in the area of the boundary of the current model. If so, we can express the likelihood of the edge map using eq. (5.22). This equation assumes that C_b represents a two dimensional model contour. In our case this contour is the two dimensional contour resulting by *embedding* the three dimensional boundary of the current model into the two dimensional edge map. In order to estimate the edge map likelihood, we have to compute the distances of the pixels of the edge map in the area of C_b. Finding the pixels in the area of C_b as well as calculating their distances from the curve is performed by means of the euclidean distance transformed image of I_b, which will be hereinafter denoted as I_{bd}: Given the current parameter set of the model $\mathbf{p}$, a three dimensional point $\mathbf{X_b}$ on its boundary amounts to:

$$\mathbf{X_b} = (x_s(\mathbf{p}; \eta_b, \omega), y_s(\mathbf{p}; \eta_b, \omega), z_s(\mathbf{p}; \eta_b, \omega)), \tag{5.30}$$

where the values of x_s, y_s, z_s can be obtained using the explicit form of (3.3), with η_b shown in eq. (3.22), and ω a value in the range $[0, 2\pi)$. According to the section 2.3, the image coordinates $\mathbf{x_b} = (x, y)$ of $\mathbf{X_b}$, can be derived using the set of expressions (2.2), that is :

$$\begin{aligned} x(\mathbf{p};\eta_b,\omega) &= \left\lfloor \frac{z_s(\mathbf{p};\eta_b,\omega)}{7} \right\rfloor \\ y(\mathbf{p};\eta_b,\omega) &= \frac{C+1}{2} - \lfloor x_s(\mathbf{p};\eta_b,\omega) \rfloor , \end{aligned} \tag{5.31}$$

where C the number of columns in the range image. If so, $I_{bd}(\mathbf{x_b})$ expresses the distance of $\mathbf{x_b}$ to the closest edge pixel. If we now consider M boundary points:

$$\mathbf{X_{b}}_i = ((x_s(\mathbf{p};\eta_b,\omega_i), y_s(\mathbf{p};\eta_b,\omega_i), z_s(\mathbf{p};\eta_b,\omega_i))), \quad i = 1...M, \quad \omega_i = \frac{i\pi}{M}, \tag{5.32}$$

distributed along the model boundary, and assume $\mathbf{x_{bi}}$, their corresponding image coordinates, then the likelihood of the edge pixels in the area of the boundary, or equivalently the likelihood of I_b amounts to:

$$P(I_b|\mathbf{p}) = \prod_{i=1}^{M} \frac{1}{\sigma_b\sqrt{2\pi}} \exp - \left(\frac{I_{bd}(\mathbf{x_{bi}})^2}{2\sigma_b^2} \right), \tag{5.33}$$

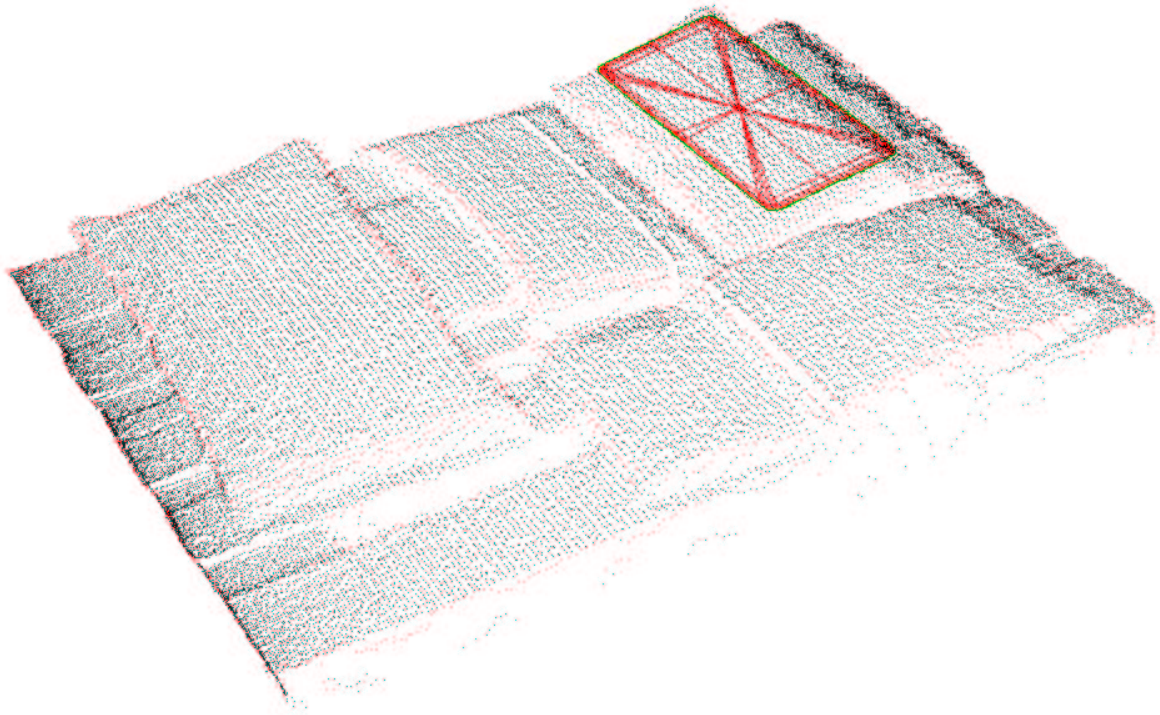
where σ_b is the standard deviation of the noise in the data acquisition process.

Our wish is to have a uniform sampling of the boundary so that the probability of generating edge pixels in some particular distance to the boundary is the same along the model boundary. However, as discussed in section 3.3, using the explicit form (3.3) we obtain points which are not uniformly distributed along the boundary. For this reason, we use the sampling suggested in [122] which outputs (almost) uniformly distributed model boundary points (the user is referred to section 3.3 for details). Besides, the bigger the number of points along the boundary we consider, the more robust will eq. (5.33) represent the information in the edge map. Unfortunately, a big number of points makes the computation of eq. (5.33) slower. In our experiments, we discovered that consideration of 200 points is a good trade-off between computational efficiency, and robustness in the computation and representation of the edge map's likelihood respectively.

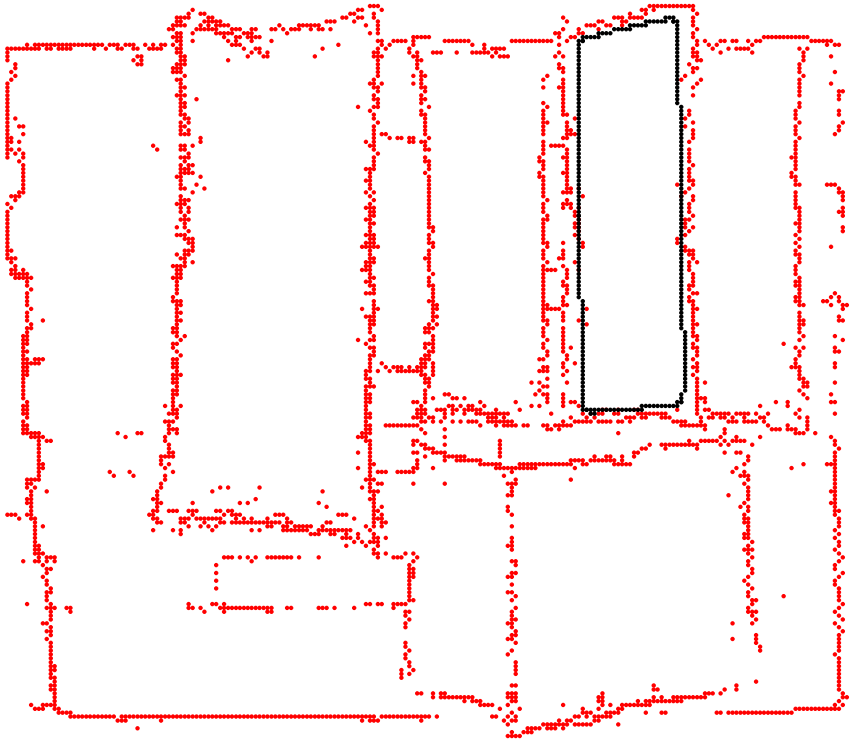
The first term L_b in the right part of eq. (5.29), corresponds to the log-likelihood of I_b, which can be directly derived by eq. (5.33), and amounts to:

$$L_b(\mathbf{p}, I_b) = \log P(I_b|\mathbf{p}) \propto -\sum_{i=1}^{M} I_{bd}(x(\mathbf{p};\eta_b,\omega_i), y(\mathbf{p};\eta_b,\omega_i))^2 \tag{5.34}$$

Maximizing eq.(5.34) with respect to $\mathbf{p_b}$ results into locally fitting the boundary of the currently examined superquadric model $\mathbf{p}$, to the edge pixels of I_b. This is illustrated in figures 5.16, 5.17: Fig. 5.16 (a), shows the input superquadric model, superimposed to the range image of fig. 5.6 (b). Fig. 5.16 (b), shows the two dimensional edge map in which the boundary of the input model is embedded. Maximization of eq. (5.34) with respect to $\mathbf{p_b}$, outputs the boundary of fig. 5.17 (b). The three-dimensional model corresponding to the updated parameter subset is shown in fig. 5.17 (a).

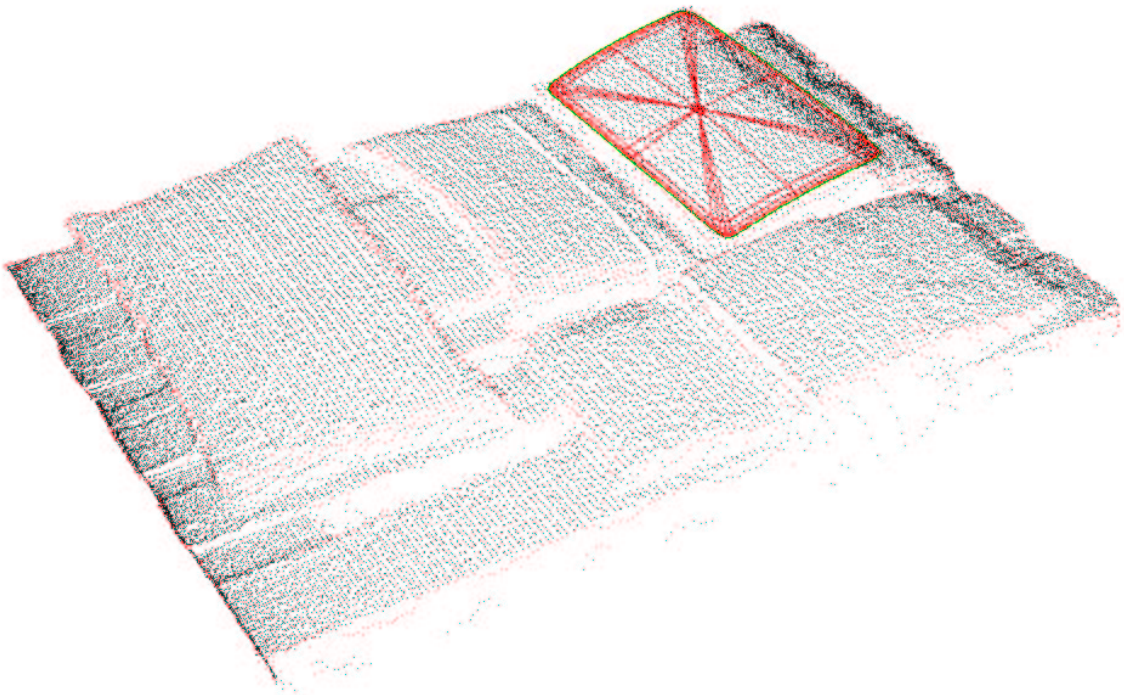


(a) Initialized model in 3D

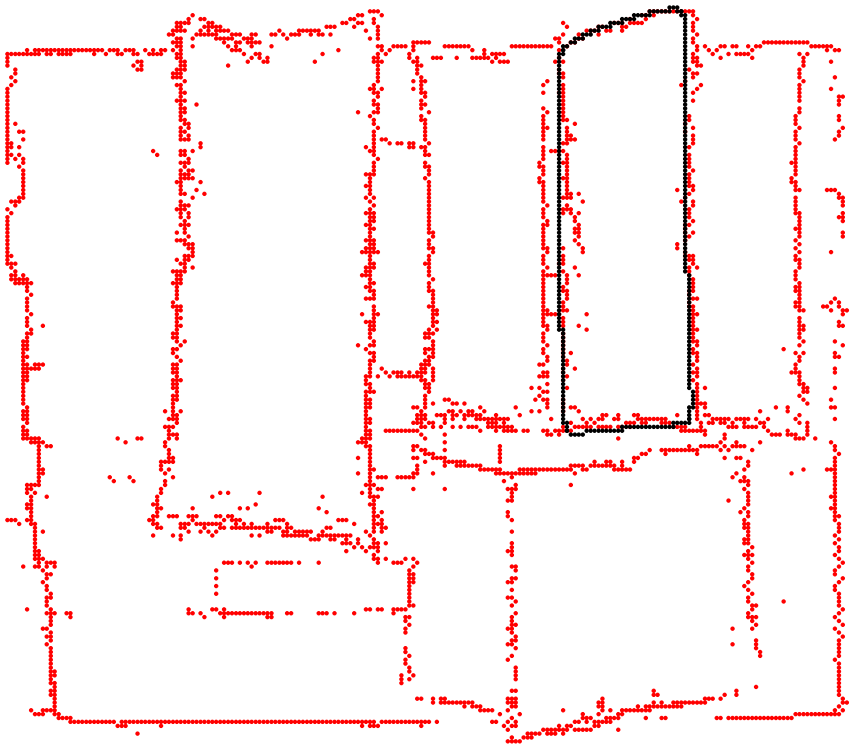


(b) Initialized boundary in 2D

Figure 5.16: Initialization for boundary fitting



(a) Recovered model in 3D



(b) Recovered boundary in 2D

Figure 5.17: Result of boundary fitting

Note, that maximization of (5.34) with respect to $\mathbf{p_b}$, requires the computation of the gradient of L_b, which amounts to:

$$\nabla L_b(\mathbf{p_b}) = -2\sum_{i=1}^{M} \frac{\partial I_{bd}(x(\mathbf{p};\eta_b,\omega_i), y(\mathbf{p};\eta_b,\omega_i))}{\partial x}\frac{\partial x}{\partial \mathbf{p_b}} + \\ + \frac{\partial I_{bd}(x(\mathbf{p};\eta_b,\omega_i), y(\mathbf{p};\eta_b,\omega_i))}{\partial y}\frac{\partial y}{\partial \mathbf{p_b}} \tag{5.35}$$

Due to the fact that the function L_b is non linear with respect to the parameters involved, iterative optimization must be carried out to update $\mathbf{p_b}$, which means that $\nabla L_b(\mathbf{p_b})$, should be computed many times. The partial derivatives $\frac{\partial I_b}{\partial x}$, $\frac{\partial I_b}{\partial y}$ are calculated numerically, and since they do not depend on $\mathbf{p_b}$, they are computed only once before starting the optimization. The derivatives $\frac{\partial x}{\partial \mathbf{p_b}}$, $\frac{\partial y}{\partial \mathbf{p_b}}$ should be computed in every iteration of the maximization process, but since their values are acquired by closed form expressions, the fitting operation is fast.

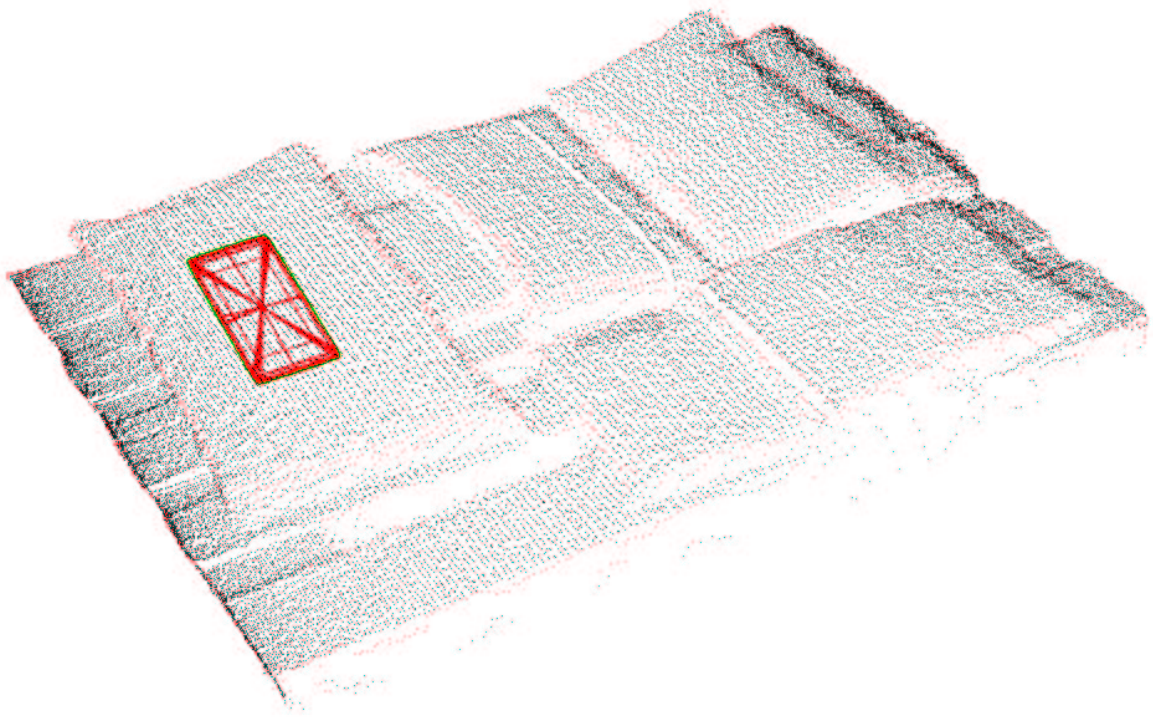
The second term in eq. (5.29), corresponds to the log-likelihood of the region image, and measures the similarity between the region enclosed by the superquadric boundary C_b to the region in I_r. Assuming $\mathbf{R_b}$ the former and $\mathbf{R}$ the latter region, according to [29], the similarity between $\mathbf{R_b}$ and $\mathbf{R}$ is proportional to the likelihood of the region image. Hence:

$$\log P(I_r|\mathbf{p}) \propto \sum_{\mathbf{x}=(x,y)\in\mathbf{R_b}} I_r(\mathbf{x}) \tag{5.36}$$

The preceding expression, sums the the pixel values of I_r in $\mathbf{R_b}$. Since only the pixels of the binary region image I_r which belong to $\mathbf{R}$ have the value 1 and all the others zero, the bigger this sum is the more similar $\mathbf{R}$ and $\mathbf{R_b}$ will be. Hence, maximization of (5.36) with regard to $\mathbf{p_b}$, forces $\mathbf{R_b}$ to become similar to $\mathbf{R}$. Figure 5.18, illustrates the situation: The shaded area of fig. 5.18 (b), the content of the region image, is the region of pixels assumed to correspond to the exposed surface of the superquadric model shown in fig. 5.18 (a), the boundary of which is embedded in figure 5.18 (b). In addition, the edge map is superimposed in the figure only for visualization purposes. Maximizing (5.36) with respect to $\mathbf{p_b}$, forces the superquadric boundary to enclose the shaded area in fig. 5.18 (b).

However, maximizing (5.36) is a computationally expensive process because it involves a sum along all the pixels of a two dimensional region. In order to reduce the computational costs, [29] employs the Green's theorem, so that the sum along the region is converted to a sum along pixels of the curve C_b:

$$\sum_{(x,y)\in\mathbf{R_b}} I_r(x,y) = \frac{1}{2}\sum_{(x,y)\in C_b}\left(N_r(x,y)\frac{\partial x}{\partial \omega} + M_r(x,y)\frac{\partial y}{\partial \omega}\right) \tag{5.37}$$



(a) Model in $3D$

(b) Region image before region fitting

Figure 5.18: Inputs of the region fitting process

where M_r and N_r amount to:

$$\begin{aligned} M_r(x,y) &= \sum_{i=0}^{x} I_r(i,y) \\ N_r(x,y) &= -\sum_{i=0}^{y} I_r(x,i) \end{aligned} \tag{5.38}$$

Despite this conversion, the computational costs involved when maximizing the right part of eq. (5.37) with respect to $\mathbf{p_b}$ are still relatively high: Prior to the beginning of the optimization process computation of the matrices M_r, and N_r is required. Given that I_r is of dimensions $N \times N$, then according to (5.38) the operation of calculating M_r, N_r is of complexity $O(N^2)$. Optimization of (5.37) is performed by means of an iterative process. In each iteration of the process, the gradient of the sum with respect to the parameter vector should be computed, which according to [28] p.182, involves the computation of the following partial derivatives: $\frac{\partial x}{\partial \omega}$, $\frac{\partial y}{\partial \omega}$, $\frac{\partial x}{\partial \mathbf{p_b}}$, $\frac{\partial y}{\partial \mathbf{p_b}}$, $\frac{\partial^2 x}{\partial \mathbf{p_b} \partial \omega}$, $\frac{\partial^2 y}{\partial \mathbf{p_b} \partial \omega}$.

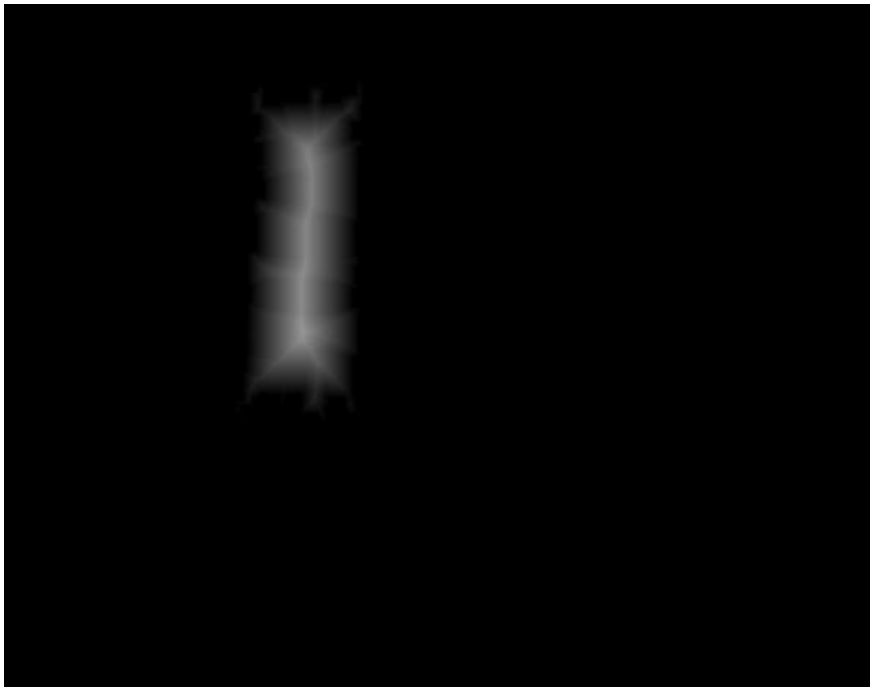
The computational costs involved in maximizing eq. (5.37) can be substantially reduced by considering that in order to match $\mathbf{R_b}$ to $\mathbf{R}$, we could instead fit C_b to the boundary of $\mathbf{R}$. In the previous paragraphs, we have seen how fitting of C_b to the binary edge map can be achieved, by means of the euclidean distance transform of the latter image. Assuming that I_{rbd} the euclidean distance transformed image of the binary image containing the boundary of $\mathbf{R}$, C_b can be fitted to the boundary of $\mathbf{R}$ by maximizing the following expression:

$$L_r(\mathbf{p}, I_r) = -\sum_{i=1}^{M} I_{rbd}(x(\mathbf{p}; \eta_b, \omega_i), y(\mathbf{p}; \eta_b, \omega_i))^2 \tag{5.39}$$

In order to obtain I_{rbd}, we adopt the following straightforward and efficient procedure, for the illustration of which the region image I_r comprising the shaded region $\mathbf{R}$ of fig. 5.18 (b) is employed. Firstly, we create the euclidean distance transformed image of I_r. This image is shown in fig. 5.19 (a). Then, we consider the complementary image of I_r and we compute its euclidean distance transformed image, illustrated in fig. 5.19 (b). Finally, we retrieve I_{rbd}, by replacing the pixels of $\mathbf{R}$ of the first image with the corresponding pixels of the other. The output image is illustrated in fig. 5.20. The iso-distance contour of value 0 is highlighted in the image, and coincides with the boundary of $\mathbf{R}$, as desired.

In analogy with the maximization of (5.34), maximizing (5.39), requires the computation of $\frac{\partial x}{\partial \mathbf{p_b}}$, $\frac{\partial y}{\partial \mathbf{p_b}}$, in each iteration of the maximization process. Computing only these partial derivatives per iteration takes much less time than computing the partial derivatives required when determining $\mathbf{p_b}$, by maximizing the right part of (5.37). Besides, these derivatives are required for maximizing the likelihood of the edge map as well, which means that their values can be determined only once and be used for both fitting the boundary to the edge map and to the region image. The complexity of the pre processing step does not increase when maximizing (5.39). As shown before, pre-computation of M_r and N_r from I_r, an operation of complexity $O(N^2)$, if the images are of dimensions $N \times N$. Extraction of I_{rbd}, involves computing the Euclidean distance transformed images of fig. 5.19, an operation of complexity $O(N^2)$ as well (see [40]). In short, maximizing (5.39) instead of the right part of (5.37)

(a) EDT region image



(b) EDT of the complementary region image

Figure 5.19: Distance transforms of the region image

for determining $\mathbf{p_b}$, increases the computational efficiency of the region fitting operation.

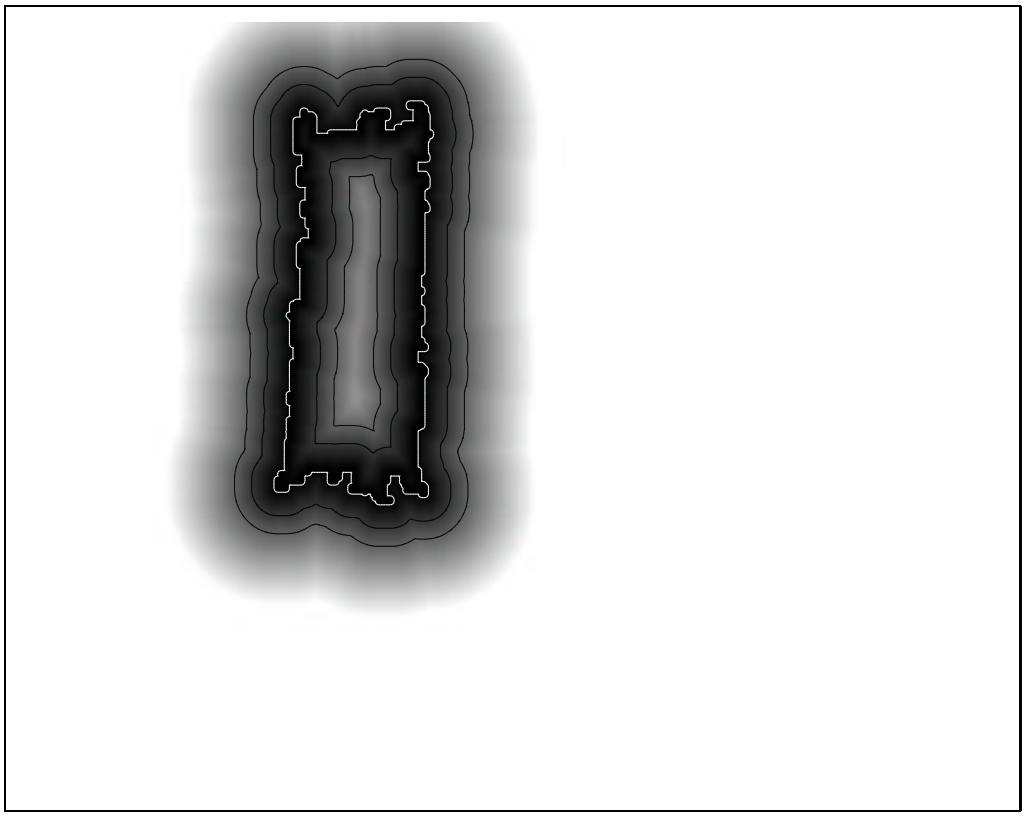


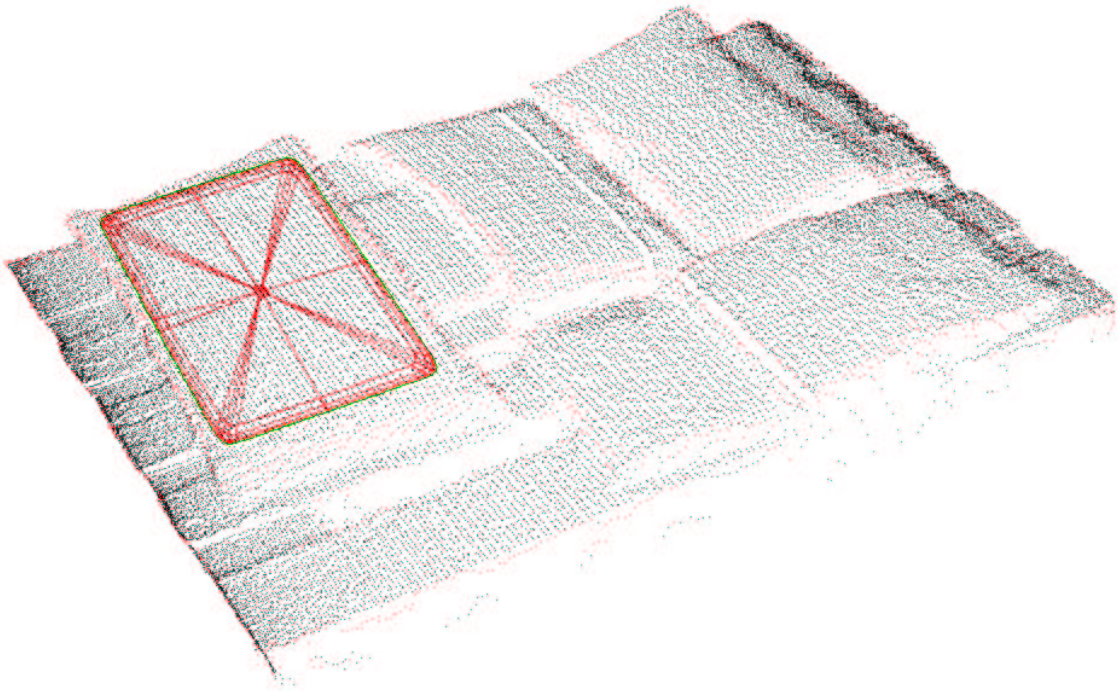
Figure 5.20: EDT image of the boundary of $\mathbf{R}$

The result of boundary estimation by maximizing (5.39), are shown in fig. 5.21. Fig. 5.21 (a), shows the region image after the end of the fitting process, on which C_b as well as the edge map are superimposed. Fig. 5.21 (b), shows the model in $3D$, which corresponds to the updated parameter subset $\mathbf{p_b}$.

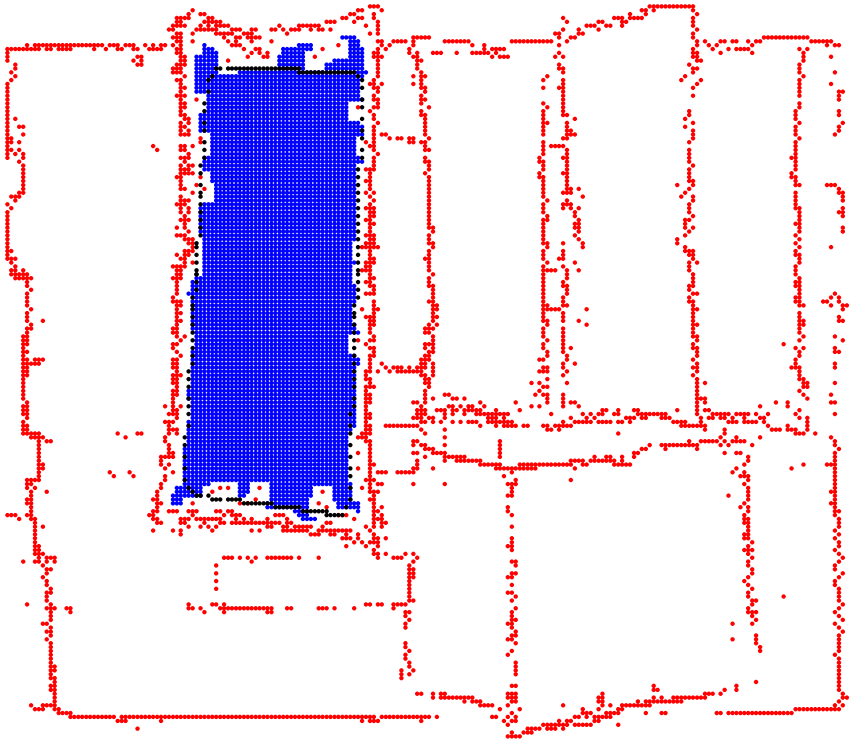
In conclusion, given $\mathbf{p_b}_k$, the input value of $\mathbf{p_b}$ before the kth invocation of the boundary module, its kth iteration involves maximization of (5.29), with regard to $\mathbf{p_b}_k$ so that $\mathbf{p_b}_{k+1}$ is acquired. We chose not to bias $\mathbf{p_b}_{k+1}$ towards a specific range of values in our application, and for this reason the prior term is not taken into consideration in the maximization process. Hence, the parameter refinement process within the context of the kth invocation of the boundary module is expressed as follows:

$$\begin{aligned} \mathbf{p_b}_{k+1} &= \arg\max_{\mathbf{p_b}_k} L(\mathbf{p_b}_k, I_b, I_r{}_k) \\ &= \arg\max_{\mathbf{p_b}_k} [L_b(\mathbf{p_b}_k, I_b) + L_r(\mathbf{p_b}_k, I_r{}_k)] \end{aligned} \tag{5.40}$$

Using eq. (5.34) and eq.(5.39) we replace L_b and L_r respectively in eq. (5.40), to obtain:



(a) Recovered model in 3D



(b) Recovered boundary in 2D

Figure 5.21: Result of region fitting

$$\mathbf{p_b}_{k+1} = \arg\min_{\mathbf{p_b}_k} \left[\mu_1 \sum_{i=1}^{M} I_{bd}(x(\mathbf{p}_k;\eta_b,\omega_i), y(\mathbf{p}_k;\eta_b,\omega_i))^2 + \right. \\ \left. + \mu_2 \sum_{i=1}^{M} I_{r_k b_d}(x(\mathbf{p}_k;\eta_b,\omega_i), y(\mathbf{p}_k;\eta_b,\omega_i))^2 \right] \quad (5.41)$$

In the preceding equation, $I_{r_k b_d}$ is the euclidean distance transformed image of the boundary of the region contained in the input region image I_{rk}. μ_1, μ_2 denote user defined weighting constants having to do with the relative significance of each sum in updating $\mathbf{p_b}$, the functionality of which will be clarified in section 5.3.3.

5.3.2.1.3 Region module: Region growing influenced by boundary information The region module refines the parameter vector subset $\mathbf{p_r}$. In addition, it updates the contents of the region image I_r. The kth invocation of the region module, succeeds the kth invocation of the boundary module. In this time point, inputs of the module is except of the range image I, the vector $\mathbf{p_r}_k$, and the region image I_{rk} obtained by the previous invocation of the module, as well the output of the kth invocation of the boundary module, the vector $\mathbf{p_b}_{k+1}$. Outputs of the module are the $\mathbf{p_r}_{k+1}$, and I_{rk+1}. Figure 5.22, illustrates: In this figure, $\mathbf{p}_{k+} = (\mathbf{p_s}, \mathbf{p_r}_k, \mathbf{p_b}_{k+1})$, denotes the entire parameter set of the superquadric model, as is before the kth invocation of the region module.

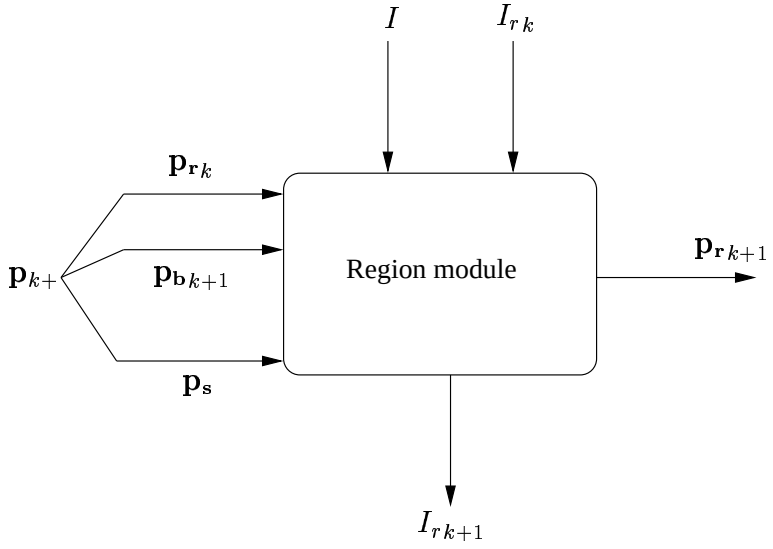


Figure 5.22: The kth invocation of the region module

The region module is realized by means of the region growing process. We decided to adopt the region growing approach for dealing with region based information, mainly because it is the standard way for recovering geometric parametric models from range images. In addition, as seen in section 5.1.3.2.1, it exhibits advantages like straightforwardness in the implementation, intuitiveness and robustness against abrupt noise.

The region module performs parameter refinement by the iterative succession of a model fitting step, followed by a pixel classification step. The model fitting step updates the parameter vector $\mathbf{p_r}$, by maximizing the likelihood of the range image with respect to $\mathbf{p_r}$. Since the region module deals with parameter updating related to a single object in the image, we can assume that the entire range image comprises only the examined object. Boundary information is taken into consideration into the region module in the computation of the image likelihood. More specifically, we assume that the image coordinates of range points corresponding to the object of interest, and thus involved in the computation of the likelihood, will be enclosed by the curve generated when embedding the boundary of the input superquadric model with parameters $\mathbf{p}_{k+}$ in the image plane. If $\mathbf{x_b}$ the set of the enclosed pixels, then using eq.(5.21), and considering that superquadrics are used as modeling elements we obtain:

$$P(I|\mathbf{p}_{k+}) = \prod_{\mathbf{x}\in\mathbf{x_b}} \frac{1}{\sigma_r\sqrt{2\pi}} \exp - \left\{ \frac{\sqrt{a_1a_2a_3}(F^{\epsilon_1}(\mathbf{p}_{k+}; I(\mathbf{x})) - 1)^2}{2\sigma_r^2} \right\}, \tag{5.42}$$

where F is the superquadric inside-outside function of eq. (3.5).

We additionally incorporate prior knowledge about the values that the parameter vector $\mathbf{p_r}$, expressed via the probability distribution $P(\mathbf{p_r})$. Knowledge incorporation is realized by letting the model fitting step maximize the logarithm of the posterior probability of the model parameters $P(\mathbf{p}_{k+}|I)$, which according to the Bayes theorem is equivalent to a sum of two terms: the prior and the likelihood term. In conclusion, the fitting step of our region module, updates $\mathbf{p_r}$, as follows:

$$\begin{aligned} \mathbf{p_r}_{k+1} &= \arg\max_{\mathbf{p_r}_k} [\log P(\mathbf{p}_{k+}|I)] \\ &= \arg\max_{\mathbf{p_r}_k} [\log P(\mathbf{p}_{k+}) + \log P(I|\mathbf{p}_{k+})] \\ &= \arg\max_{\mathbf{p_r}_k} [\log P(\mathbf{p_r}_k) + \log P(I|\mathbf{p}_{k+})] \end{aligned} \tag{5.43}$$

The prior probability of the region parameter vector is assumed to be multivariate Gaussian distribution and amounts to:

$$P(\mathbf{p_r}) = \frac{1}{(2\pi)^{\frac{l}{2}}|\Sigma_{\mathbf{p_r}}|^{\frac{1}{2}}} \exp\left\{ -\frac{1}{2}\left[(\mathbf{p_r} - \bar{\mathbf{p}}_\mathbf{r})\, \Sigma_{\mathbf{p_r}}^{-1} (\mathbf{p_r} - \bar{\mathbf{p}}_\mathbf{r})^T \right] \right\} \tag{5.44}$$

The parameters of the distribution, that is the mean vector $\bar{\mathbf{p}}_r$, and the covariance matrix $\Sigma_{\mathbf{p_r}}$ are computed within the context of a training phase, which takes place off-line prior to the beginning of the segmentation process.

By replacing $P(\mathbf{p_r}_k)$ and $P(I|\mathbf{p}_{k+})$ in eq. (5.43) with their expressions of eq.(5.44) and (5.42) respectively, and by taking the logarithms into consideration the model fitting step of the region module is as follows:

$$\mathbf{p_r}_{k+1} = \arg\min_{\mathbf{p_r}_k} \left[\lambda\, (\mathbf{p_r}_k - \bar{\mathbf{p}_r}_k)\, \Sigma_{\mathbf{p_r}_k}^{-1} (\mathbf{p_r}_k - \bar{\mathbf{p}_r}_k)^T + \sum_{\mathbf{x}\in\mathbf{x_b}} \sqrt{a_1a_2a_3}(F^{\epsilon_1}(\mathbf{p}_{k+}; I(\mathbf{x})) - 1)^2 \right] \tag{5.45}$$

In the preceding equation, λ is a factor determining the importance of the prior term in the optimization process.

The superquadric model with parameters $\mathbf{p_{k+1}} = (\mathbf{p_s}, \mathbf{p_{r_{k+1}}}, \mathbf{p_{b_{k+1}}})$ generated after the fitting step of the region module, and the region image $I_{r\,k}$ are used to generate the updated region image $I_{r\,k+1}$ within the context of the classification step of the region module. More specifically, the pixel classification step examines the set of pixels in the neighborhood of the boundary of the region included in the region image. The pixels of this set which have a small radial euclidean distance from the model $\mathbf{p_{k+1}}$, are added in the region of $I_{r\,k}$ to obtain $I_{r\,k+1}$.

Note, that according to the original region growing approach, model fitting followed by point classification should continue, until no more range points with small distance to the model exist, or until the model fitting error exceeds a user defined threshold. However, invocation of the region module in our case involves only one iteration of the fitting step followed by the classification step. The reason for this choice will become clear in section 5.3.3.

The flow diagram of the kth iteration of the classify-and-fit stage of our approach is illustrated in fig. 5.23. This figure combines the information illustrated in both figures 5.15 and 5.22. Given the model $\mathbf{p}_k$, the boundary module employs the edge map I_b and the region image of the previous iteration $I_{r\,k}$ to obtain the boundary fitted model $\mathbf{p}_{k+}$. Subsequently, given the range image I and the region image $I_{r\,k}$, the region module updates $\mathbf{p}_{k+}$ to obtain the region fitted model $\mathbf{p}_{k+1}$. The iterative updating of the model parameter vector continues until no significant changes in model parameters, and no significant difference in the contents of the region images between two successive iterations is observed.

5.3.2.2 Post processing

The post processing step succeeds the model parameter refinement stage, and serves a dual purpose: Firstly, it determines if the refined model corresponds to a graspable object. Secondly, it recovers the height parameter of the model.

A graspable box-like object is by definition an object which fully exposes its largest surface to the laser source. Full exposure of the exposed surface implies that edge points along the entire contour of the surface is captured by the boundary detection process, and are thus present in the boundary image. Therefore, for the graspable objects in the image, the fitting error residual of the boundary of the model's exposed surface is expected to be small. In addition, since such objects can be well modeled by box-like superquadrics, the fitting error residual of a superquadric model to the range points of the image belonging to the object is as well expected to be small. In short, the post processing method examines both model fitting error residuals to the range image and the boundary image, and decides that the object of interest is graspable if both are smaller than user defined thresholds *max-average-boundary-error* and *max-average-model-error* respectively. Models decided to describe non graspable objects are rejected from further consideration.

Models corresponding to graspable objects are forwarded to the second stage of the post processing step. Its target is to recover the value of the model parameter a_3, which expresses

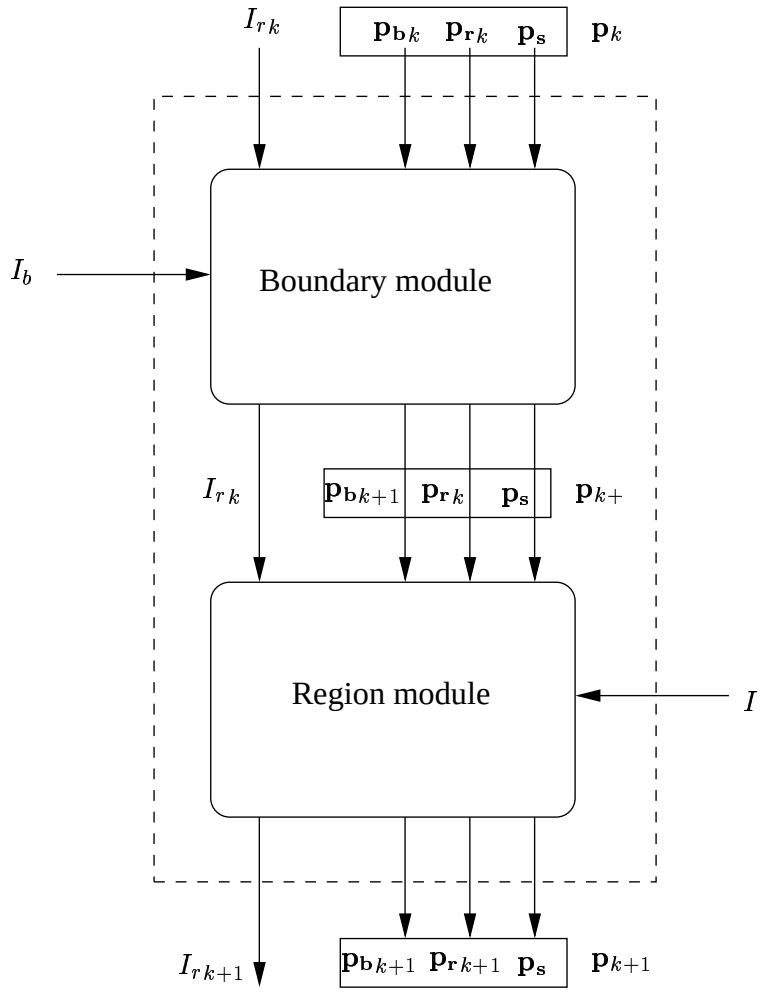


Figure 5.23: The kth iteration of the classify-and-fit process

the height of the objects in the viewing direction. Up to now, this parameter has been set up to a small value and kept constant throughout the recovery process. The reason for setting a small value for this parameter is to increase the robustness of the region module, in the sense that points belonging to objects occluded by the object of interest are not included in the model's region. In configurations of box-like objects, determination of the height parameter is not important for grasping the objects, since the grasping operation can be robustly performed when the exposed surface of the objects is localized, but for placing the objects in a target platform.

As observed in [159], estimation of this parameter from a single view of the configuration, is prone to errors even in images comprising a single object. In order to solve the problem, [68] proposes a solution based on extrapolating surfaces adjacent to the object of interest. In the event that these surfaces are *supporting* surfaces (the ground for example), artificial range points corresponding to the supporting surface are added in the range image, in order to aid the recovery of the correct height. [159] observes that these data points should not be handled as true data points, but rather as *constraining* points. A penalty function is attached to each constraining point which forces the superquadric model to shrink enough so that the constraining points lie outside the recovered model. The problem with this method is that the resulting superquadric fitting function becomes complicated, and the fitting procedure computationally inefficient. [109], addresses the problem by examining the pile as a whole, generate multiple image interpretations resulting from various values for the heights of the objects, and uses the physical law of object stability to determine the one which most likely corresponds to the image. The computational efficiency of this method however, is questionable.

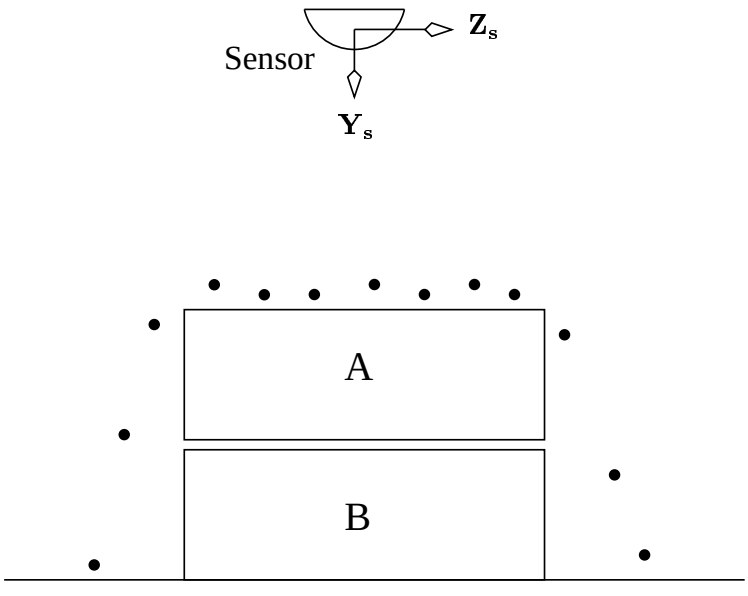


Figure 5.24: Disambiguity in determining the height of box A

A solution to the problem of determining the height of the objects in the viewing direction given a single view of the configuration, is in general not possible. The reason why this happens is illustrated in fig. 5.24. Assuming $\mathbf{Y_s}$ is the depth axis of the sensor, the dots in

the figure correspond to the range points acquired when the configuration in the figure is scanned. Given these range points it is impossible to determine if there is only one or more boxes on the platform, and therefore to find the height of the box A on the top of the pile. In order to address the problem, we employ a very simple method, which does not always guarantee the delivery of the correct solution, but gave satisfactory results in the majority of the cases, and is based in iterative region growing: We consider the pixels neighboring to the boundary of the recovered model, the corresponding range points of which have a small *orthogonal distance* [4] from the model. For each of these points we found the one with the maximum distance from the exposed surface of the object, and we set a_3 to this value. We continued this process iteratively until no more points with a small orthogonal distance to the model could be found. The value for the height of the model obtained by the last iteration is delivered as output of the process.

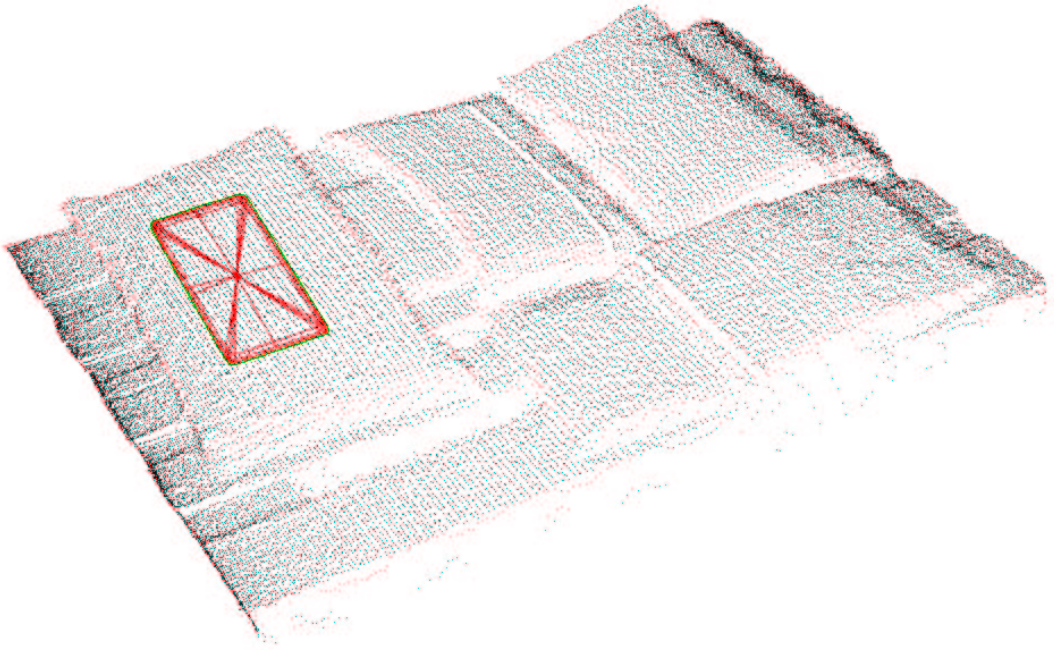
5.3.3 Experiments

In this section, we demonstrate the robustness of our recovery approach, using the configuration of sacks illustrated in figure 5.6. Note, that among all the categories of objects we have experimented with, such objects exhibit the lowest degree of rigidity. Hence, their exposed surfaces may deform in such a way so that their similarity to our deformable superquadric decreases to a considerable extend. In addition, the edge map generated from images containing such objects is usually very noisy, as can be seen in fig. 5.7. Despite this, these objects can be successfully recovered by our approach.

The experiments conducted in this section concern the recovery of one object in the pile namely of the object A of fig. 5.6. Note, that the parameters of the corresponding seed is close to the actual object parameters (see fig. 5.11). Nevertheless, in order to better manifest our system's robustness, we will use the smaller seed of fig. 5.25, to segment the object of interest. The first image of this figure, namely fig. 5.25 (a), shows the three dimensional model fitted to the image points belonging to the seed, and the second, namely fig. 5.25 (b), shows the boundary of the fitted model embedded on the edge map. Besides, the pixels belonging to the seed are shaded in the latter figure.

Our first experiment shows the inability of a region-based-only segmentation approach to segment the object of interest. Standard region growing of the seed using the the first two parameters of table 5.1, results into the model shown in fig. 5.2. This result is reproduced in fig. 5.26 (a). In addition, fig. 5.26 (b), shows the region of pixels that the region growing process determined to belong to the object of interest, superimposed to the edge map of the image. As already discussed in section 5.1.3.4, and as observed in these figures as well, region growing suffers from the over-growing problem. Note, that the seed has been placed inside the object of interest and in a relatively big distance from its boundaries. Seed placement in this position increases the robustness of the region growing approach [77] p.157. Despite this, the result of region segmentation is discouraging. In addition, segmentation using region growing is relatively time consuming. This is because the process involves multiple fitting of the model to a big number of range points. Segmentation of the particular object of interest took around 70 seconds in a Pentium 4, 2.8 GHz.

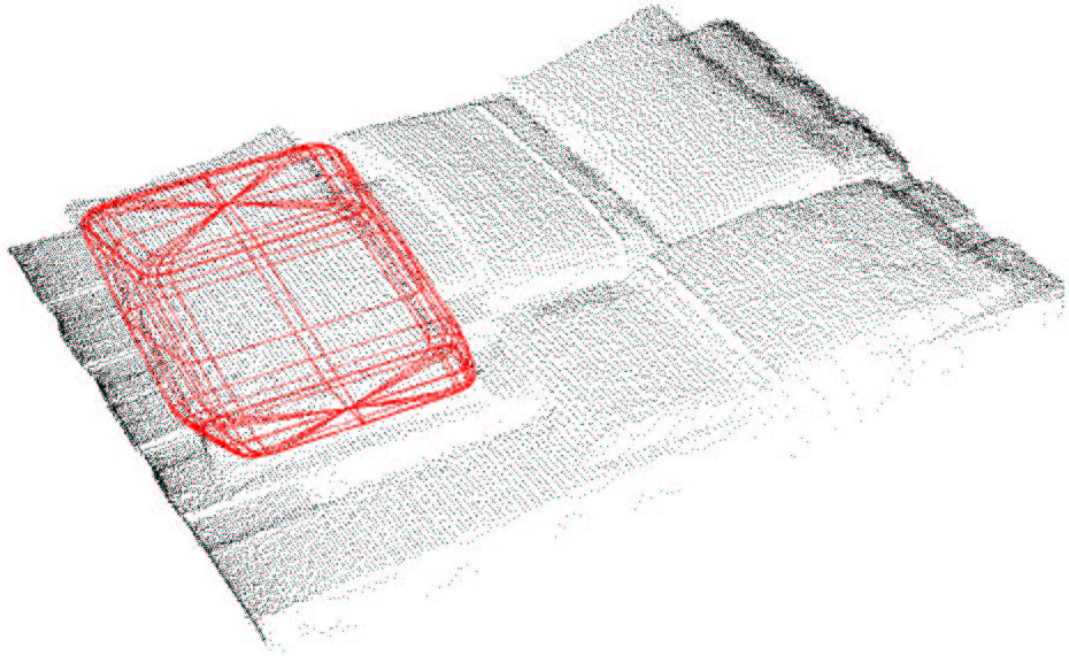
The subsequent experiments segment the object of interest using our framework, which was



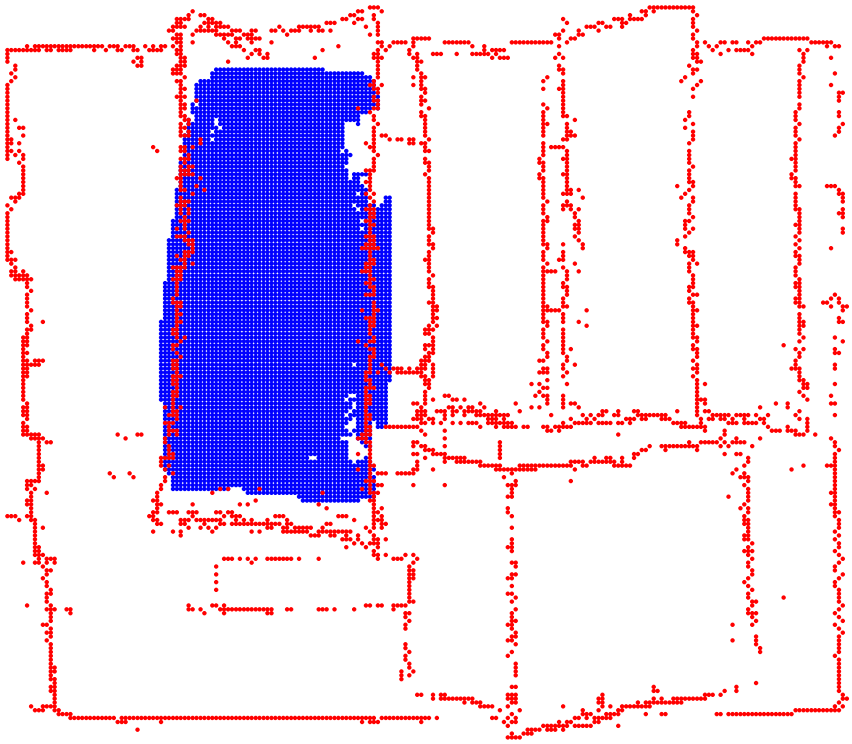
(a) Seed in $3D$

(b) Model boundary on image plane

Figure 5.25: Seed for the recovery of a bag



(a) Recovered model in $3D$



(b) Recovered region on the image plane

Figure 5.26: Output of the region (growing) based segmentation

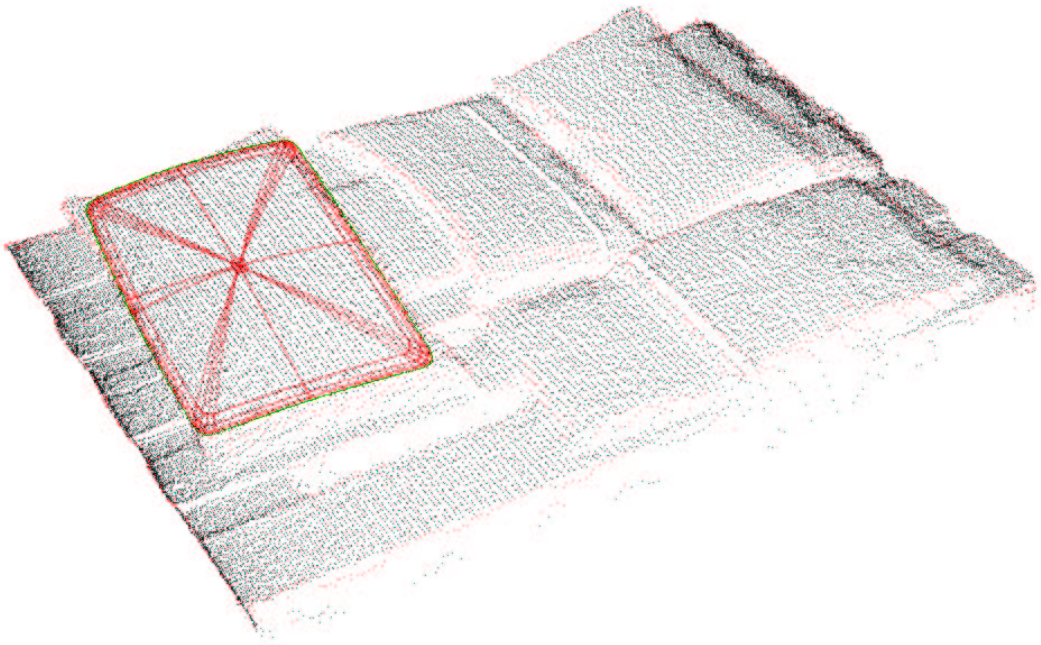
implemented in C. We used the *lsqnonlin* procedure of MATLAB, and more specifically the Levenberg-Marquard option [64] for executing all optimization processes. For performing all experiments we employed a Pentium 4, 2.8GHz PC. We used the following values for the system's parameters: The thresholds used to determine when the classify-and-fit process should stop iterating, namely the norm of the difference in the model parameter vector, and the difference in number of points in the region image between two successive iterations (see section 5.3.2.1, and fig. 5.13), were $\epsilon = 0.1$, and $\nu = 5$ points respectively. The thresholds used by the post processing step to determine when an object is graspable, that is the boundary fitting error residual and the model fitting error residual (see section 5.3.2.2), were set to *max-average-boundary-error* = 1.1 pixels and *max-average-model-error* = 15mm respectively. Concerning the region module, the maximum allowable distance of a range point to the model to be included in the model's region namely *max-point-distance* is set to 40mm. Finally, we used $M = 200$ points on the model boundary curve for fitting it to the edge map. Table 5.2 summarizes.

Parameters	**Values**
ϵ	0.1
ν	5 points
M	200 points
max-average-boundary-error	1.1 pixels
max-average-model-error	15 mm
max-point-distance	40 mm

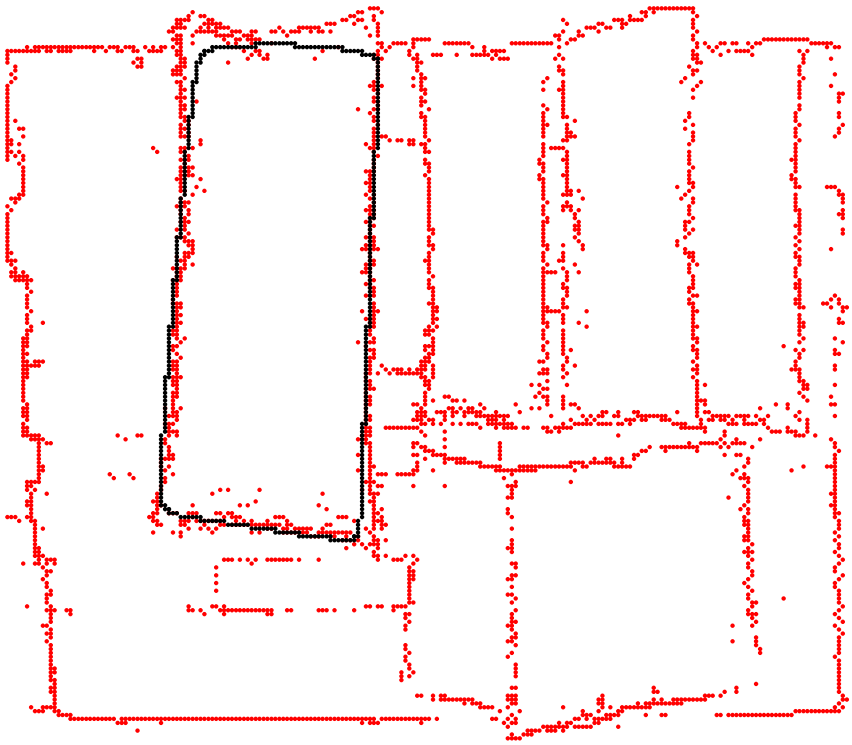
Table 5.2: Parameter values used in our framework

In particular, the experiments that follow focus on the advantages gained by integration of region with boundary based information. As discussed in section 5.3.2.1.2, and expressed in equation (5.41), the parameters μ_1, μ_2, control the degree in which integration is performed. In particular, when using $\mu_1 = 1$, and $\mu_2 = 0$, the segmentation process is *guided* by boundary information. The model obtained using these values is illustrated in fig. 5.27. More specifically, fig. 5.27 (a), shows the recovered model in three dimensions, and fig. 5.27 (b) the recovered boundary of the superquadric model embedded in the edge map. The problem of using boundary information only for object segmentation, is that noisy edge points on the exposed surface of the object of interest attract the boundary curve, so that the model does not grow as it should. Hence, in comparison to the region growing based segmentation, the boundary based segmentation suffers from the inverse problem: model under-growing. Hence, information integration is expected to mitigate both problems and lead to robust object recovery. The boundary guided method is fast and converges in about 5 seconds. Note, that the noisy points on the exposed surface of the objects occur because the edge detector fails to model the local deformations of the target objects. In the event that the target objects are rigid, (as is the case for card-board boxes for example) no noisy points appear. In this case the boundary guided segmentation approach provides a way for both robust and efficient object recovery.

Setting $\mu_1 = 0$ and $\mu_2 = 1$ corresponds to using region information to guide the boundary finding process. In this case, the boundary module fits the model's boundary only to the boundary of the objects's region. This segmentation approach, although is based on region



(a) Recovered model in $3D$

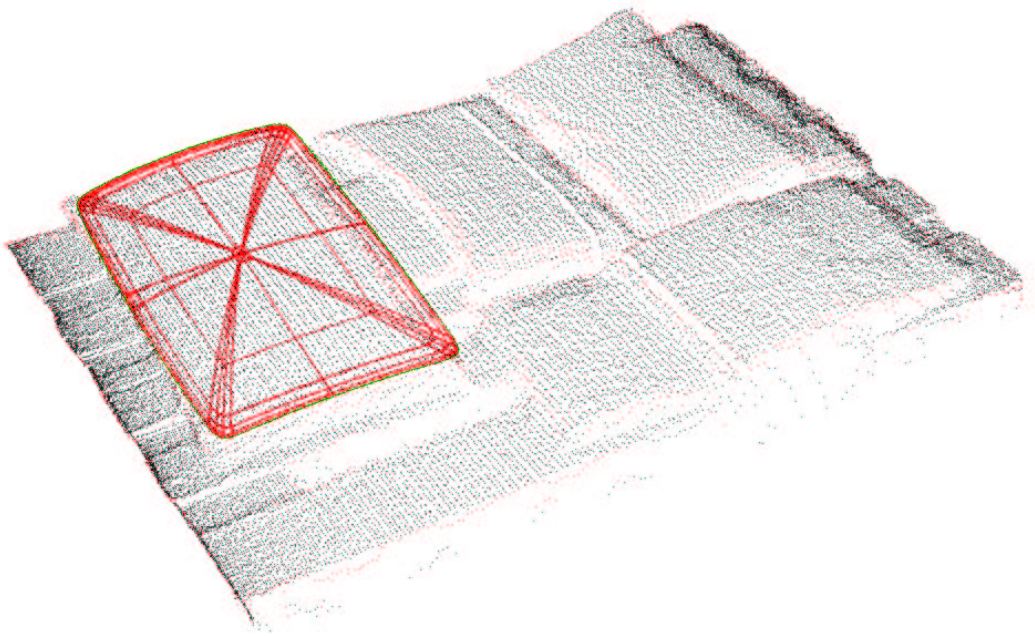


(b) Recovered boundary on image plane

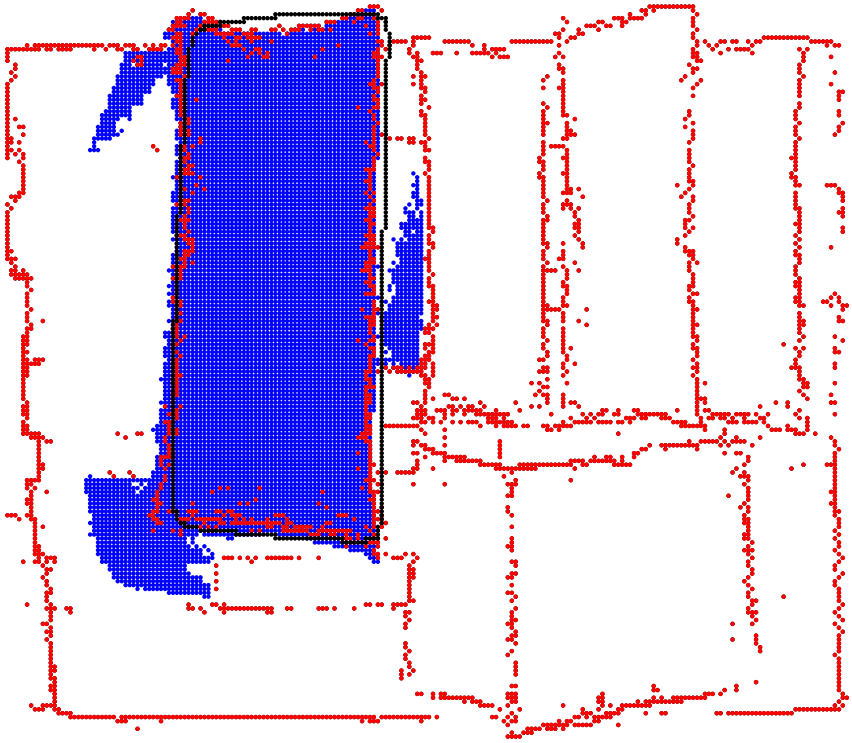
Figure 5.27: Boundary information guided segmentation ($\mu_1 = 1$, $\mu_2 = 0$)

information only, is more robust than the standard region growing operation: Here, not only the range values of the points within the region are considered for model parameter estimation, as is the case in the standard region growing operation, but the region's shape as well. The segmentation result using this strategy is illustrated in Fig. 5.28. In particular, fig. 5.28 (a), shows the recovered model in three dimensions. Fig. 5.28 (b), the boundary of the recovered model embedded on the edge map, along with the region of the object of interest. The recovered model is slightly overgrown. Model over-growing here is not as intense as what observed in fig. 5.26, and the object is much more accurately represented by the model. Note, that this happens despite the fact that the the maximum allowable distance for including a range point into the region of the object of interest is almost the double as the one used when standard region growing was applied. As a result, the number of points from neighboring objects included in the region of the object of interest is bigger now. Despite this, not all the points are used for fitting the model, but only the points contained in the sub region, whose boundary looks like a superquadric boundary. Since the robustness of the approach stems from fitting the boundary of the growing region to the model boundary, the more frequently the fitting is performed, the more robust the output is expected to be. This is the reason why in our system we let the region module execute only one iteration and then immediately invoke the boundary module. The region guided segmentation is slower than the boundary guided segmentation, since it takes about 40 seconds to converge. However, its is almost double as fast as the standard region growing operation, and the reason for this is that not all superquadric parameters are updated by the two modules, but each refines its corresponding subset of the superquadric parameter vector.

Finally, we illustrate the results obtained when $\mu_1 = \mu_2 = 0.5$. The boundary module in this case considers that boundary and region information are of equal importance for model recovery. The boundary fitting operation can be intuitively described as the simultaneous application of two forces to the boundary of the model being refined: The *boundary force*, corresponds to the first term of eq. (5.41), attracts the model's boundary towards the pixels of the edge map. When the model is small and lies far away from these points, the force has a big value. Hence, it makes the model grow fast. In this way computational efficiency is achieved. When the model has grown enough so that its boundaries lie close to the edge points, the same force *constrains* further growing of the model. In this way model overgrowing is addressed. The *region force* corresponds to the second term of eq. (5.41), and attracts the model's boundary towards the boundary of the growing region. This force is responsible for the resolution of the model under-growing problem produced by noisy edge points in the interior of the region of the object of interest. The way in which this is achieved is illustrated in fig. 5.29 (b). This figure shows an intermediate recovery stage. At this time point, noisy edge points hinder the growing of the model. However, at the same time the force exerted to the boundary by the boundary of the shaded region in the figure, is big enough and increases as the region becomes larger. After some time this force becomes bigger than the boundary force and thus the model grows. The final output of our segmentation approach is shown in fig. 5.30. Consideration of both boundary and region terms improves the result with respect to what shown in both fig. 5.27, and fig. 5.28. Actually, here neither model under-growing occurs as was the case in fig. 5.27, nor model over-growing as in fig. 5.28. The computational efficiency of this approach is lower than this of the boundary guided approach, but higher than the region guided approach. Segmentation of the object of interest took about 20 seconds.

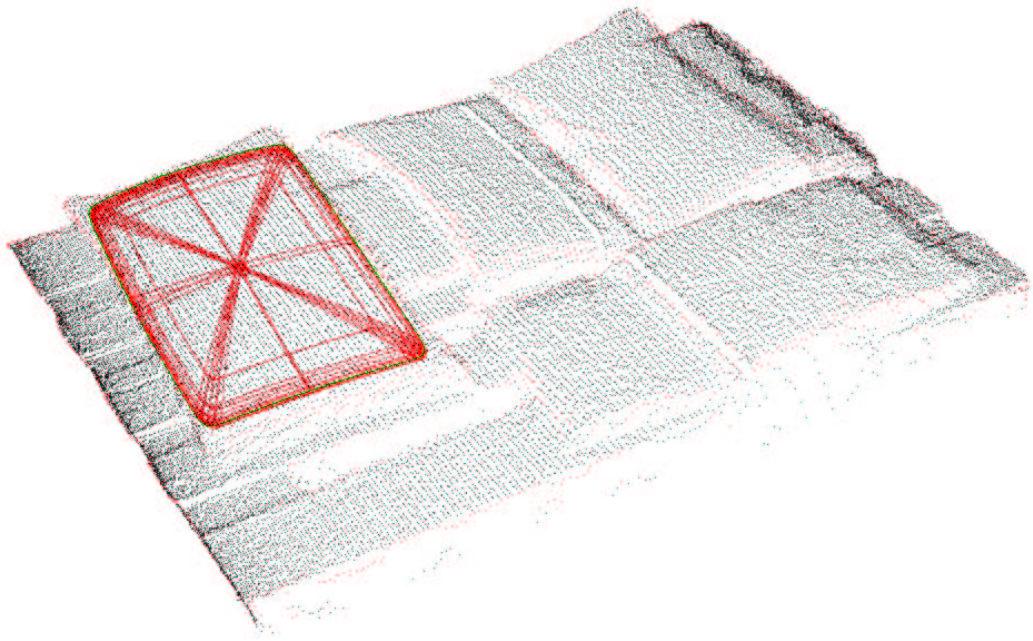


(a) Recovered model in $3D$

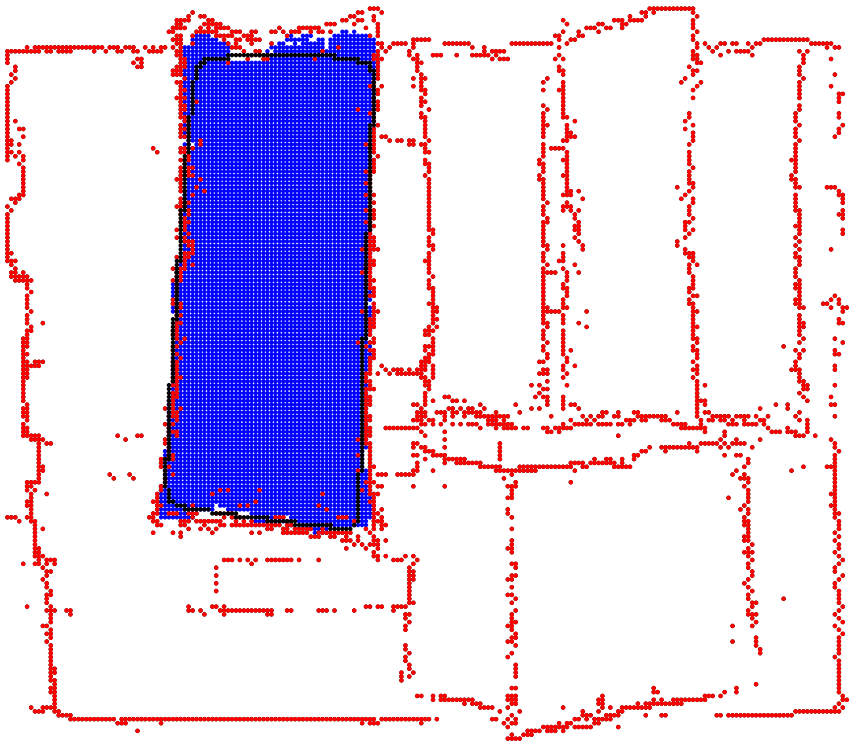


(b) Recovered boundary on the image plane

Figure 5.28: Region information guided segmentation ($\mu_1 = 0$, $\mu_2 = 1$)

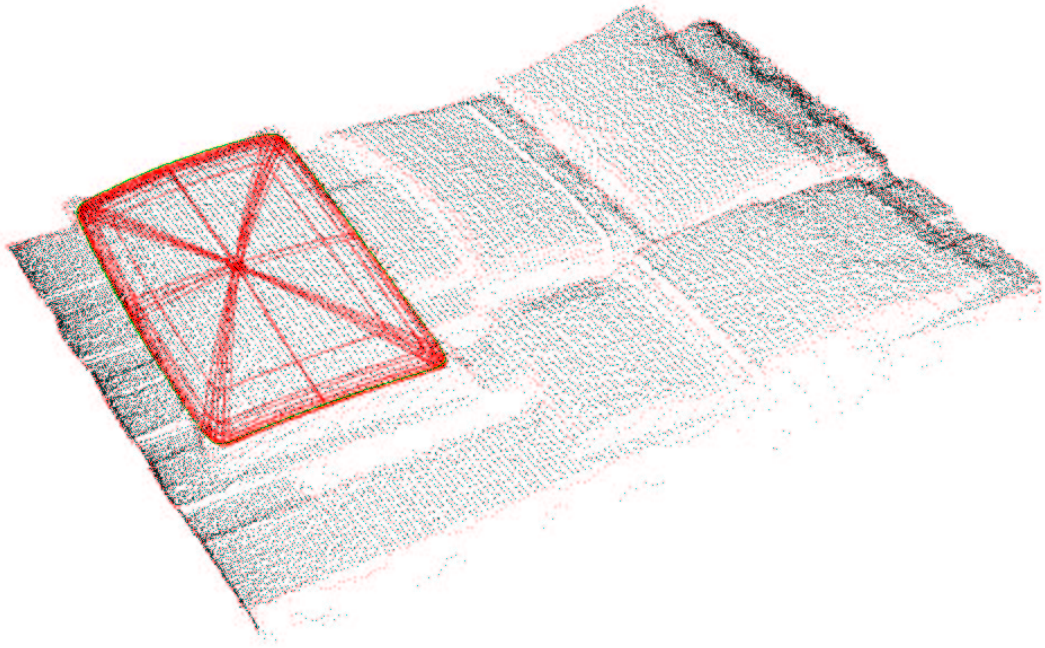


(a) Recovered boundary in $3D$

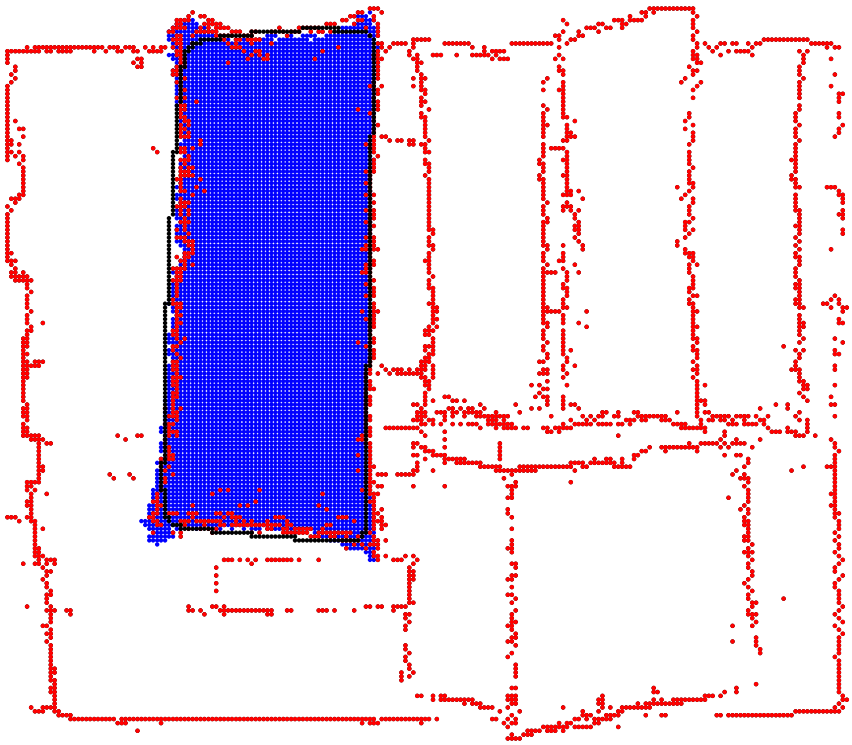


(b) Recovered model in $2D$

Figure 5.29: Intermediate output of the region-boundary based fitting ($\mu_1 = 0.5$, $\mu_2 = 0.5$)



(a) Recovered boundary in $3D$



(b) Recovered model in $2D$

Figure 5.30: Final output of the region-boundary based fitting ($\mu_1 = 0.5$, $\mu_2 = 0.5$)

Modules	Duration (sec)
Hypothesis generation	3
Hypothesis refinement	125
Hypothesis refinement per model	20

Table 5.3: Computational efficiency of our approach

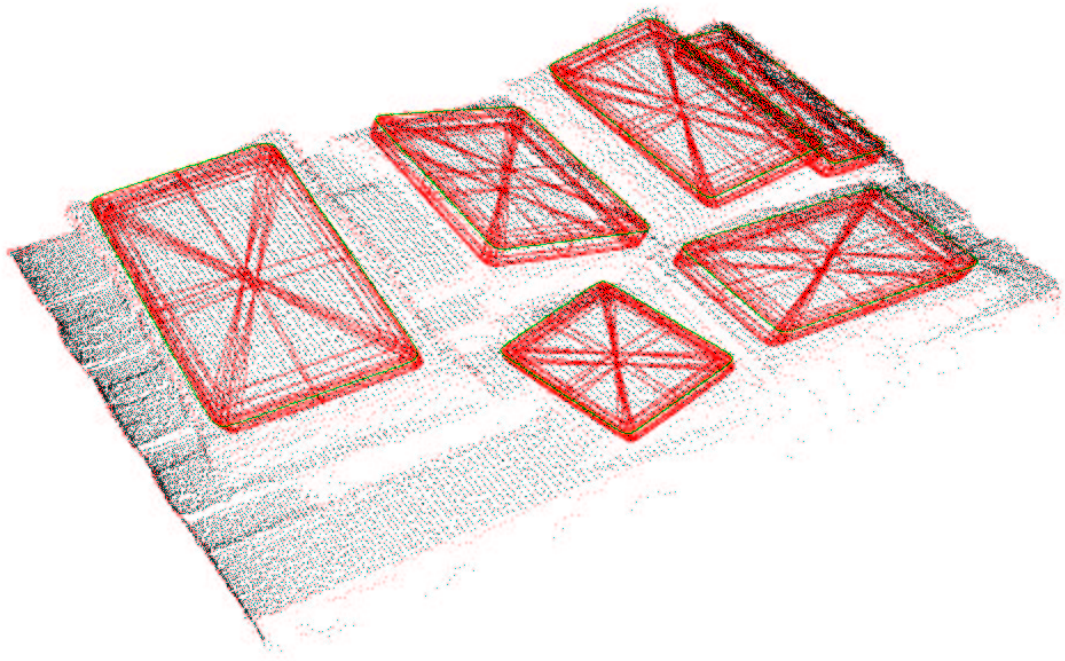
We employed our framework using the parameters of table 5.2 and $\mu_1 = \mu_2 = 0.5$ to recover all the objects of fig. 5.6. The seeds used for the recovery are shown in fig. 5.31. The segmentation result is illustrated in fig. 5.32: All the graspable objects in the configuration have been successfully recovered. It took about 2 minutes to segment the image. The average segmentation time per object was about 20 seconds. These results are summarized in table 5.3.

Note, that the results corresponding to the seeds of the non- graspable objects are not shown in this figure. The reason for this is that the fitting error residual of these models is high, so that the recovered models are discarded from the post-processing step of our segmentation approach. We demonstrate the model corresponding to a seed on a non graspable object after the end of the classify-and-fit process in fig. 5.33. In particular, fig. 5.33 (b), shows the boundary of the model embedded on the edge map: The fitting error residual of the boundary to the edge point is apparently big, which means that the model cannot correspond to a graspable object.

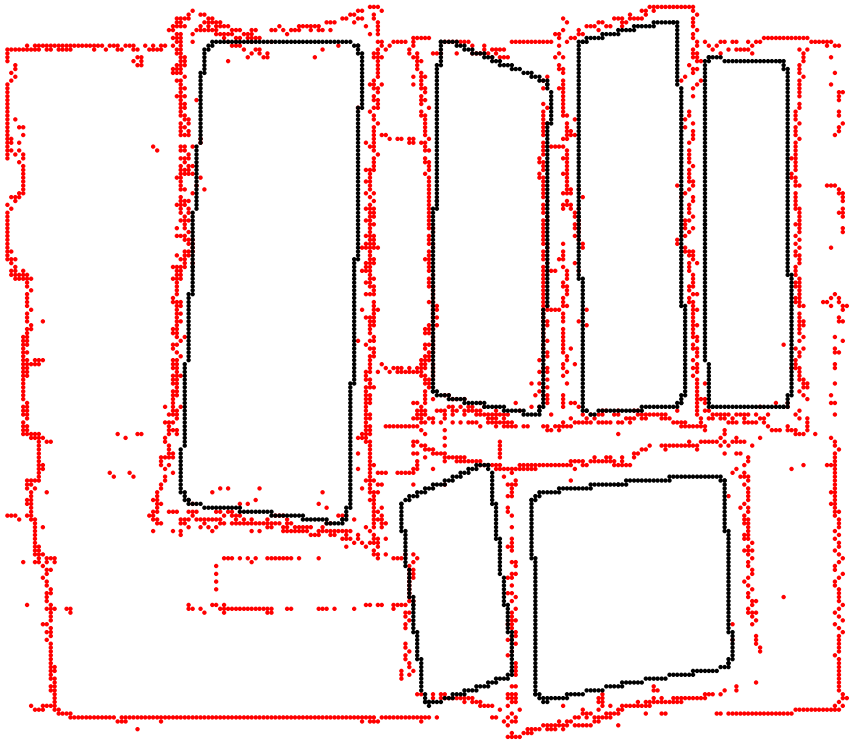
Additional experimental results on many configurations of a variety of target objects, along with quantitative results not only about the computational efficiency but for the reconstruction accuracy and the robustness of our system as well, will be presented in the chapter that follows.

5.4 Conclusions and future work

We presented a system for segmenting of box-like superquadrics in range images. Our approach extends the well- known recover-and-select framework for superquadric segmentation [98], [99], [77] by incorporating boundary information. Just as the recover-and-select framework, our approach adopts a two stage hypothesis generation and refinement strategy to perform the segmentation task. In the hypothesis generation stage, the segmentation parameters are initialized through seed placement. In the hypothesis refinement stage, the segmentation parameters are refined, through local model fitting. In both stages of the system boundary information originating from edge detection in the input range image is taken into consideration: In the hypothesis generation part, boundary information is used in order to determine the interior of the exposed surfaces of the objects on which the seeds are placed. In the hypothesis refinement stage, boundary information is used to fit the boundary of the exposed surfaces of the initialized models. More specifically, the hypothesis refinement stage is implemented so that region information and boundary information are equally taken into consideration. The way in which this is realized is inspired by a game theoretic framework

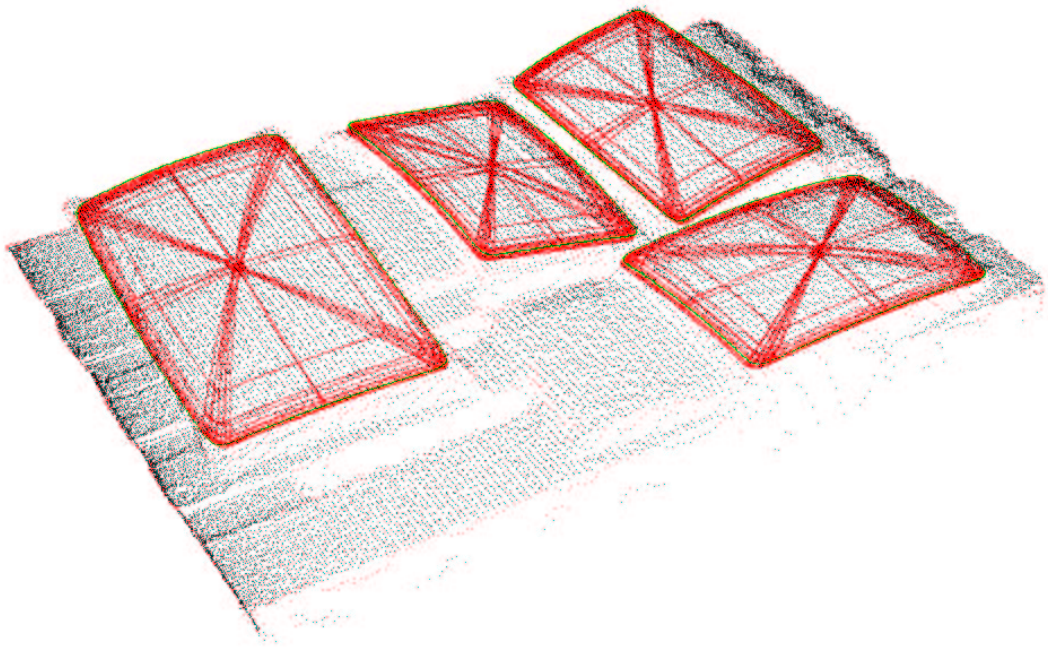


(a) Models in $3D$

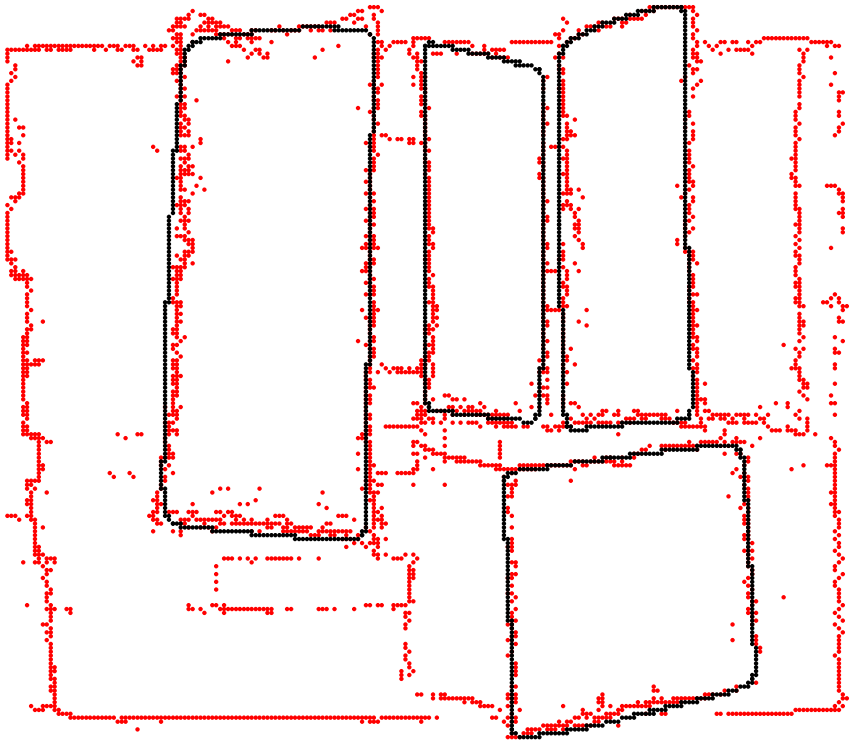


(b) Model boundaries on image plane

Figure 5.31: Seeds

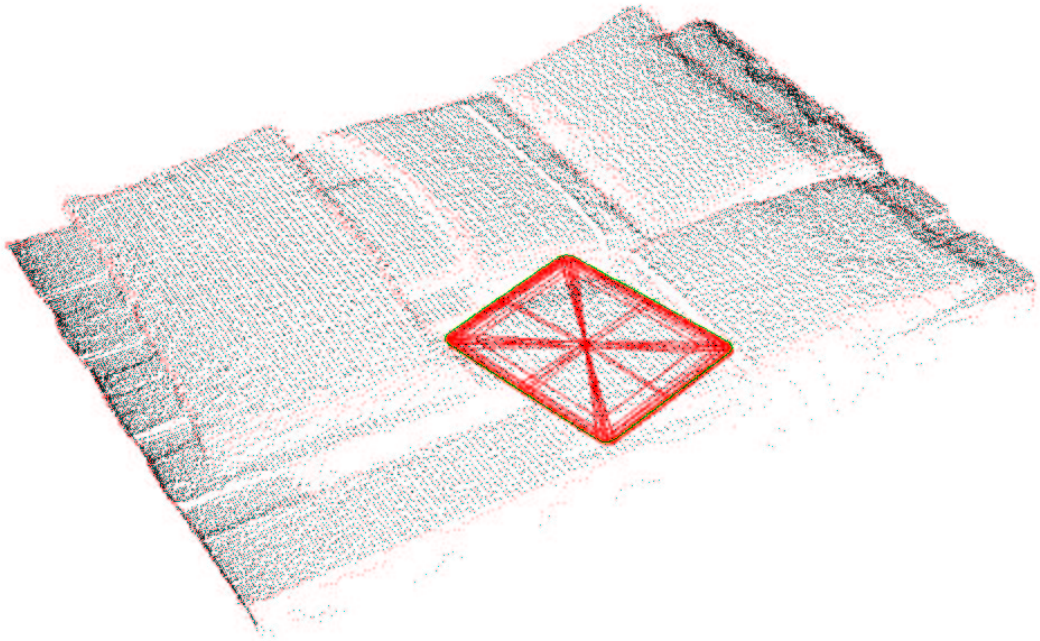


(a) Recovered models in $3D$



(b) Recovered boundaries on image plane

Figure 5.32: Recovery of piled sacks



(a) Rejected model in $3D$

(b) Rejected boundary on image plane

Figure 5.33: Model rejection

for information integration [27], [28]: Information integration is performed by iterative invocations of two modules in succession: a region module, and a boundary module. The region module, refines the model parameters by model fitting in the range image. The boundary module fits the model boundary to the edge map for this purpose. A consequence of decomposing information handling is that in our approach the superquadric parameter vector is as well decomposed: Each module does not update the the entire superquadric parameter vector, but the subset related to the information domain handled by each module.

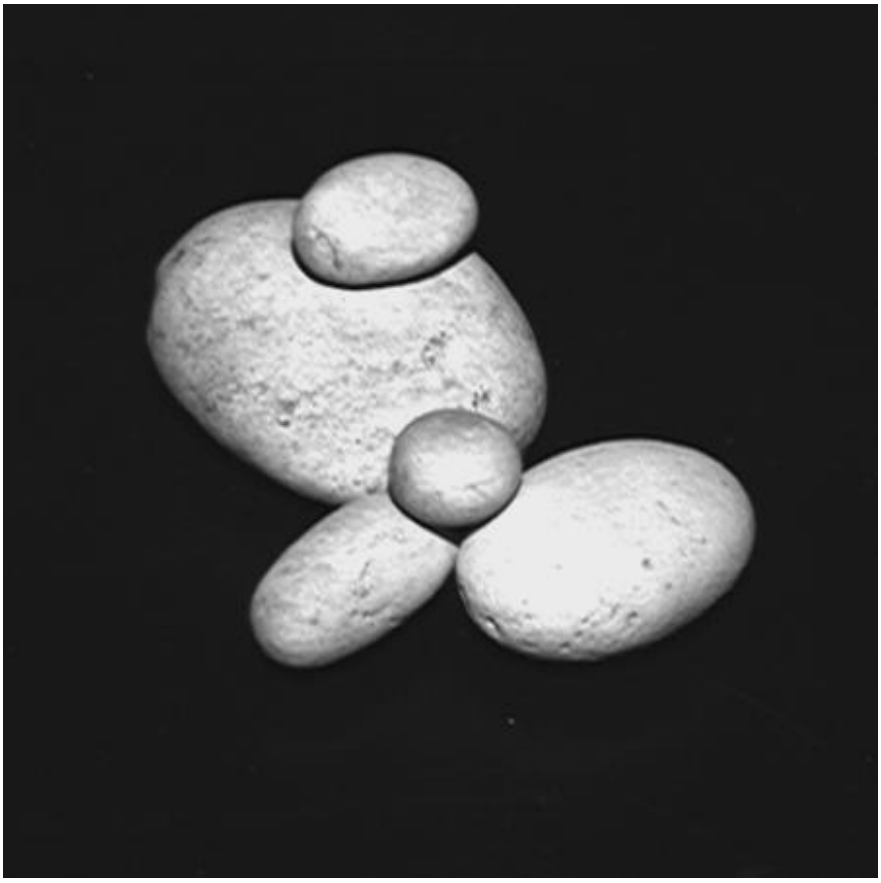
We used a representative object configuration to demonstrate that our approach outperforms the recover-and-select strategy in terms of all robustness, accuracy and computational efficiency. In particular, we showed the recover-and-select strategy is not able to accurately recover all objects in a configuration comprising sacks. On the contrary, our approach manages to satisfactory perform this task. In terms of computational efficiency, execution of our approach took about 2 minutes on a Pentium 4, 2.8GHz PC. The corresponding time the recover-and-select framework needed to perform the corresponding task was 15 minutes.

The robustness and accuracy of our system stem primarily from the fact that boundary information is incorporated in the segmentation process. Consideration of boundary information by the seed placement process results into an accurate approximation of the actual values of both the number of objects in the image as well as their parameters. Given this approximation, the subsequent parameter refinement stage is able to accurately recover the actual values, even though local operations are used. Besides, consideration of boundary information in the hypothesis refinement stage solves the region over-growing problem, which is the main cause of recovery inaccuracies of the recover-and-select framework. Finally, our system's robustness is due to the decomposition of the superquadric parameter vector recovery, since each of the two modules with which parameter recovery is achieved, search for the optimal parameter into lower dimensional parameter spaces. Quantitative results on the robustness and the accuracy of our approach will be presented in the subsequent chapter. Note, that it might be possible to theoretically prove that the stopping point of the iterative hypothesis refinement procedure is a stable *Nash* equilibrium (see section 5.2.2), that is, to prove that our system converges in the correct solution, as done in [27]. In our case, such a proof turned out to be rather tricky. Its construction is a subject of our future work.

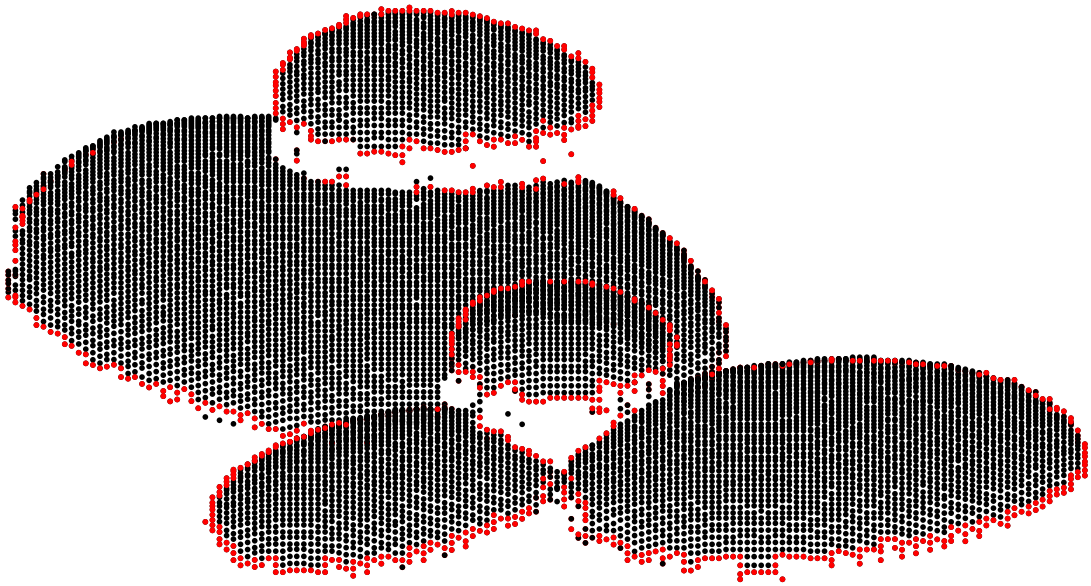
Incorporation of boundary information into the segmentation process, is as well the reason for the computational efficiency of our approach. Consideration of boundary information in the hypothesis generation process, results into a small number of initialized seeds. As a consequence, the time needed for processing these seeds is much lower than the corresponding time required by the recover-and-select framework. Besides, using boundary information for parameter recovery accelerates the growing of the models: Except for the standard region growing, models now grow as a result of the attraction forces exerted from the boundary points. In addition, computational efficiency is the outcome of decomposing the problem of superquadric recovery, since each module in the refinement stage searches for the optimal model parameter sets in lower dimensional spaces. Finally, computational efficiency is the result of using closed form parametric models like the superquadric and the boundary of the exposed surface of a superquadric for fitting. What is interesting is that similarly to the recover-and-select framework, our system offers the potential for parallel implementation. Due to the fact that parameter recovery is performed independently for each seed, a

multi-processor computer can be used, to each processor of which the refinement of a seed can be assigned. If so, our approach can be used for object recovery in real time: According to table 5.3, such a configuration should allow for image segmentation in 23 seconds.

Our system, is able to recover box-like superquadrics. What is of interest however is how our approach can be extended so that it can be employed to segment superquadrics of arbitrary shape. The first step towards this direction is to model the boundary of the objects. The parametric curve able to model the boundary is the superquadric RIM, defined in chapter 3, section 3.8. Hence, the changes that have to be incorporated into the system are mainly two: Firstly, the edge detection process should output points on the RIM of the objects. Secondly, the boundary finding module should attempt fitting the RIM of the models in the edge map. In addition, in the hypothesis refinement step, the shape parameters ϵ_1, ϵ_2 should now be updated. Note, that no changes in the control structure of our system need to be performed. Initial research toward the construction of such generic system has already been started [26]. The results are shown in fig. 5.34, fig. 5.35. More specifically, fig. 5.34 (a), shows the intensity image of a pile of stones. Fig. 5.35 (b), shows a side view of the range image. The detected edge points are boldly marked in the figure. The edge points clearly lie on the RIM of the objects. Fig. 5.35 shows the results obtained using our modified approach. Fig. 5.35 (a) illustrates the recovered models in $3D$. Fig. 5.35 (b), shows the detected boundaries on the image plane, on which the recovered RIMs are embedded. The RIMs correspond to the bold continuous ellipsoid curves in the image. The results seem very encouraging. It is true however, that the accuracy of the reconstruction is not as good as in the case of box-like superquadrics. Identifying and addressing the problems of this generic approach, is the main target of our future research.

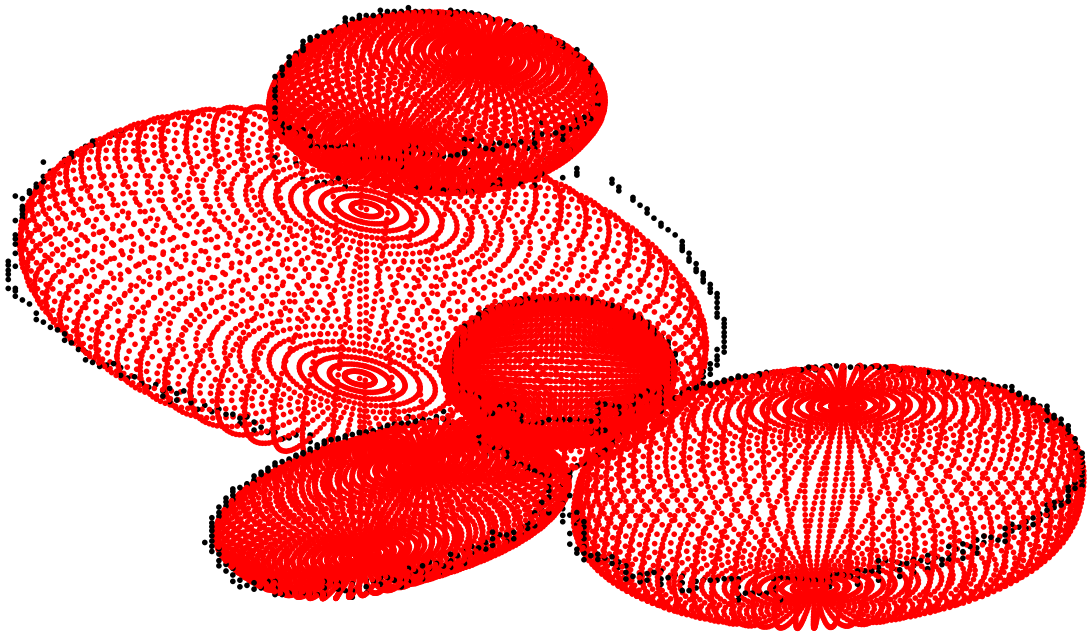


(a) Intensity image

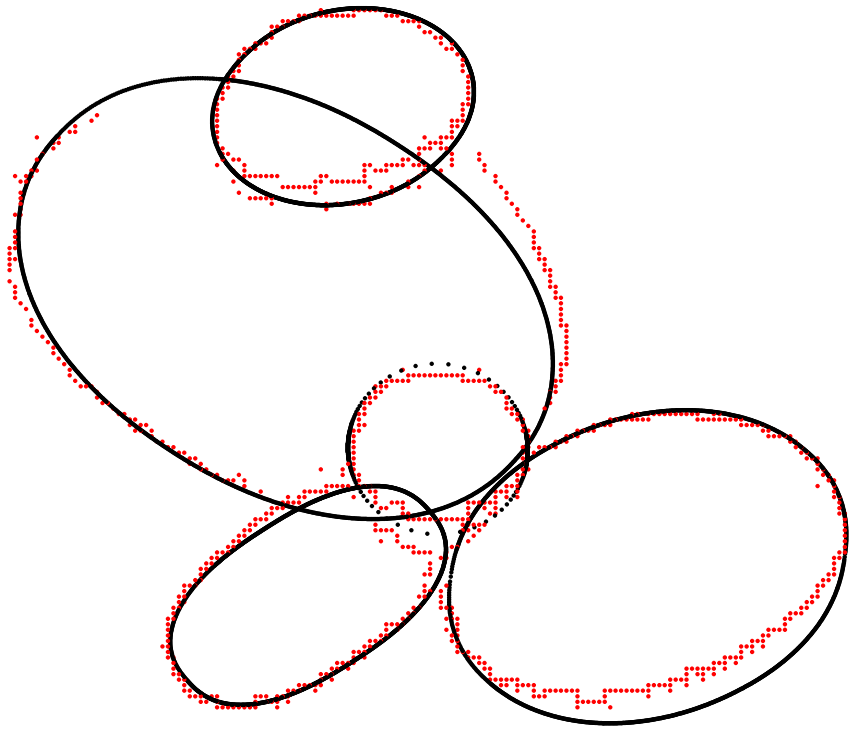


(b) Range image (side view)

Figure 5.34: Superquadric configuration



(a) Recovered models in $3D$



(b) Recovered boundaries on image plane

Figure 5.35: Superquadric recovery