Chapter 7

Recovery of Rigid Boxes using the Hough Transform

7.1 Introduction

In chapter 5 we used superquadric entities to model our target objects, and a hypothesis generation and refinement framework to perform the recovery task. In the particular chapter the object recovery problem was posed as an optimization problem, which enabled interweaving segmentation with model parameter estimation. In addition, both region and boundary based information contributed to object recovery. Here, we follow an alternative approach: We attempt object recovery using the *Hough Transform* [75], [96]. As pointed out in chapter 2 and more specifically in section 2.2.3, the Hough transform is a popular technique for recovering geometric parametric models, and its main advantage to the hypothesis generation and refinement framework is that no parameter initialization is required. In addition, it is more computationally efficient when the models comprise few parameters. However, the effectiveness of the Hough transform significantly reduces when the number of model parameters moderately increases.

Hence, usage of the Hough transform as a recovery approach requires the simplification of the entities used for modeling the target objects. In this chapter we model the *boundary of the exposed surfaces* of out target objects via *three dimensional rectangles.* Three dimensional rectangles are geometric entities that can be expressed through eight parameters. Six of them represent their pose (translation and rotation) in space, and the remaining two their dimensions (width and length). However, the disadvantage of using the particular entities for modeling the boundaries of our objects is that they cannot accurately describe the boundaries of slightly deformable objects such as bags. For this reason the approach that will be described in this chapter concerns the recovery of piled rigid boxes only. Despite this, the application range of this framework is quite vast, since rigid boxes are the objects which are most frequently encountered in industrial sites like distribution centers and post storage facilities and in the storage centers of the automobile industry.

The bounding rectangles of the exposed surfaces of our objects are recovered using the Hough transform on the *edge image* of the configuration. The edge image is acquired from a range image by means of the application of the edge detection technique described in chapter 4. Besides, we accelerate the performance of the Hough transform, by *decomposing* the recovery problem into two successive subproblems, each dealing with a subset of the boundary parameter set: The recovery of the pose parameters of the bounding rectangles followed by the recovery of their dimensions. In addition, taking into consideration the error in the localization of the edge points within the context of the transform, results into robustness and accuracy. In contrast to the recovery approach presented in 5, we use here boundary based only information. Despite the fact that only one information source is used here, the results are still accurate: Due to the rigidity of the target objects the quality of the results delivered by the edge detection process delivers results of high accuracy. Noteworthy is that region information is *implicitly* used by the edge detection process (see chapter 4), which implies that, to some extend, region information as well is taken into consideration by our approach.

In the paragraphs that follow we describe the constituting elements of our approach. More specifically, 7.2 introduces the standard hough transform, discusses its problems and proposes solutions. Section 7.3 shows the way in which the pose of the objects in space can be computed. Section 7.4, illustrates the way in which given the pose of the boundary of the exposed surface of a objects, its dimensions are computed. Section 7.5 presents some experimental results obtained by the application of our approach. Finally, section 7.6 concludes this chapter.

7.2 Parameter recovery using the Hough transform

The Standard Hough Transform (SHT) [75] is one of the most widespread methods employed for recovering multiple parametric models from images. The key idea underlying the Hough transform, is to map the difficult segmentation problem into a simple *peak detection* problem in the model parameter space. The reader is referred to [96], [53] for a detailed description of the way in which this is realized.

A problem of the SHT is its computational inefficiency when dealing with models with many degrees of freedom. Lets suppose the model sought has N parameters and each image point constraints p of them. For each image point, the SHT increments all the bins comprising a $N-p$ -dimensional manifold of an N -dimensional *accumulator*. In our case the models ($3D$ rectangles) have $N = 8$ degrees of freedom and each point constraints $p = 2$ line parameters. Even in this case, and despite the fact that our models are considerably simpler than the superquadrics used in chapter 5, applying the SHT, will be both memory consuming, since a $6D$ accumulator is needed, as well as computationally inefficient, since mapping of a single edge point requires updating a $4D$ manifold of the accumulator. Besides, a drawback of the SHT is that it does not take into consideration the error in the localization of the image points. This results in both detection of false positives and missing of objects, thus reduces the robustness of the transform.

A plethora of algorithms have been proposed to address the computational inefficiency of the SHT. Lets suppose mainly for simplicity $p = 1$, that is, we regard $2D$ images. The idea is to decrease the number of required accumulator updates per mapping by constraining the pose of the model. This is done by simultaneously mapping k $(1 < k \leq N)$ instead of one pixels to the parameter space. If so, the dimensionality of the manifold along which the accumulator must be updated drops from $N-1$ to $N-k$. In [166],[21] $k=N$ is regarded which implies $N-k=0$. In this case update of only one accumulator cell per mapping is needed. Unfortunately these approaches are not free of problems: Mapping large sets of pixels gives rise to a combinatorially explosive number of possible sets. Randomization techniques have been proposed to reduce the number of sets examined.

Other researchers propose a somewhat different solution, which is based on decomposing the Hough Transform into subproblems. Each subproblem is solved within the context of a trial: A set of points with cardinality $d(d<N)$ (distinguished set) is randomly selected. Random subsets of the remaining points with cardinality v (varying sets) are then considered, so that $d+v=N$. The union of the two sets is then mapped to the parameter space by updating one cell, since the points in the union fully constrain the pose of the model. After all the varying sets have been examined, the accumulator maxima are extracted. A trial is considered successful if those maxima satisfy user-defined criteria. The process finishes when a fixed number of trials t has been performed.

There is a variety of algorithms which may result from this framework by assigning different values to the cardinality (d) of the distinguished set, the number of varying subsets (r) examined within a trial and the number of trials (t). Leavers [95] sets $d=1$, r is automatically determined by the framework, and t is the number of connected components found in the edge map. More recently Olson with his RUDR (Recognition using Decomposition and Randomization) technique [117] considers $d=N-1$ and $r=n-1$ where n the number of edge pixels in the image. A trial is considered successful when at least m points lie on the model. Finally, if γ the probability of failure in finding a model in the image, then the number of trials is given by (7.1).

$$t=\frac{\log(\frac{1}{\gamma})}{(\frac{m}{n})^{N-1}} \tag{7.1}$$

The selection of the particular cardinality for the distinguished set results in the fact that in each trial the transform is constrained to lie on a one-dimensional manifold, that is a curve (Hough Curve), in the parameter space. Many advantages are gained from this selection: Firstly, one dimensional data structures are used for the accumulation process, so the memory requirements are reduced to $O(n)$. Secondly, the complexity is $O(tr)$ or $O(tn)$. Since all quantities in (7.1) are user-defined constants ($\frac{m}{n}$ is a constant fraction of the input data), the overall algorithm complexity turns out to be linear to the number of pixels. Thirdly, if localization errors are considered, a set of pixels maps not exactly on the Hough Curve, but to an area (error cloud) of the parameter space which lies close to the Hough Curve. The projection of the cloud to the Hough Curve will thus be a good approximation of it. This allows for simple, accurate and efficient error propagation to the parameter space. The error is expressed in a straightforward way via square boundaries in the image space whose side length (δ) is measured in pixels. The combination of these benefits are not to be found in other approaches simultaneously, up to our knowledge, and for this reason the adoption of RUDR framework seems to be the best choice for addressing our problem.

7.3 Recovery of pose

It is since years known in the computer vision community, that a visible vertex of a convex object provides the strongest constraints for accurately determining its pose [31]. Although a variety of methods for detecting corners in two dimensional edge images have been reported, this is not the case for three dimensional edge images obtained by edge detection in range images. The majority of the existing approaches (for example [31],[9] and others) use region information to extract vertices. The disadvantage is that the objects need to expose more than one surfaces to the sensor for accurate estimation of the vertex position. In [86], edge detection was performed on the input range image, object boundaries were detected using the Dynamic Generalized Hough Transform [95] and vertices were extracted by grouping orthogonal object boundaries. This allows for accurate vertex detection even if the objects expose only one surface. However the error in the localization of the edge points was not taken into consideration, which made the approach not as robust as desired.

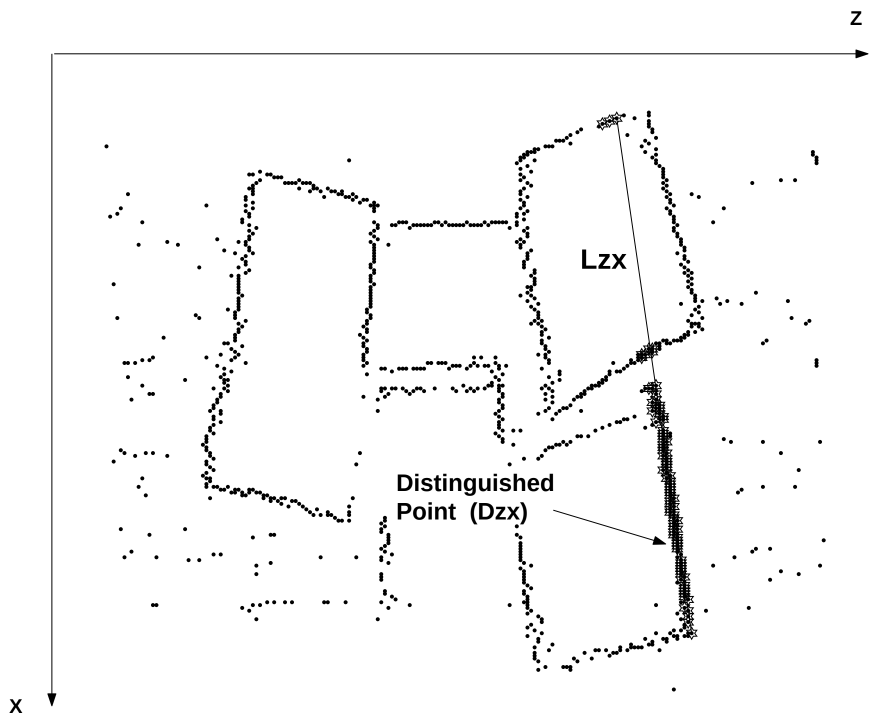
The technique discussed here comprises the same two parts, as was the case in [86]: Linear boundary detection and boundary grouping. However, there are essential differences between the two approaches: The three dimensional linear object boundaries are recovered, via application of an iterative algorithm to the edge image: In each iteration Hough Transforms are executed and a set of models are recovered, followed by a model selection process which retains the best models in terms of accuracy. The Hough Transforms are constrained as in [117], so that the edge points localization error is accurately propagated to the parameter space. Boundaries comprising a fixed fraction of the edge points are sought in each iteration. Finally, orthogonal pairs of boundaries are grouped to a vertex. The orthogonality of a pair of recovered linear boundaries is determined via a statistical test. Given the extracted three dimensional process, an alignment procedure ([31], [7]) recovers the pose of the corresponding object in space.

In the following section, we describe the way in which vertex detection is carried out in detail: More specifically, section 7.3.1 focuses on the issue of the detection of three dimensional linear segments from three dimensional boundary points using the Hough transform. Section 7.3.2 shows how, given the extracted linear segments the objects vertices are acquired. Finally, section 7.3.3, shows the output of the vertex detection operation on some object configurations, and comments on the efficiency of the approach.

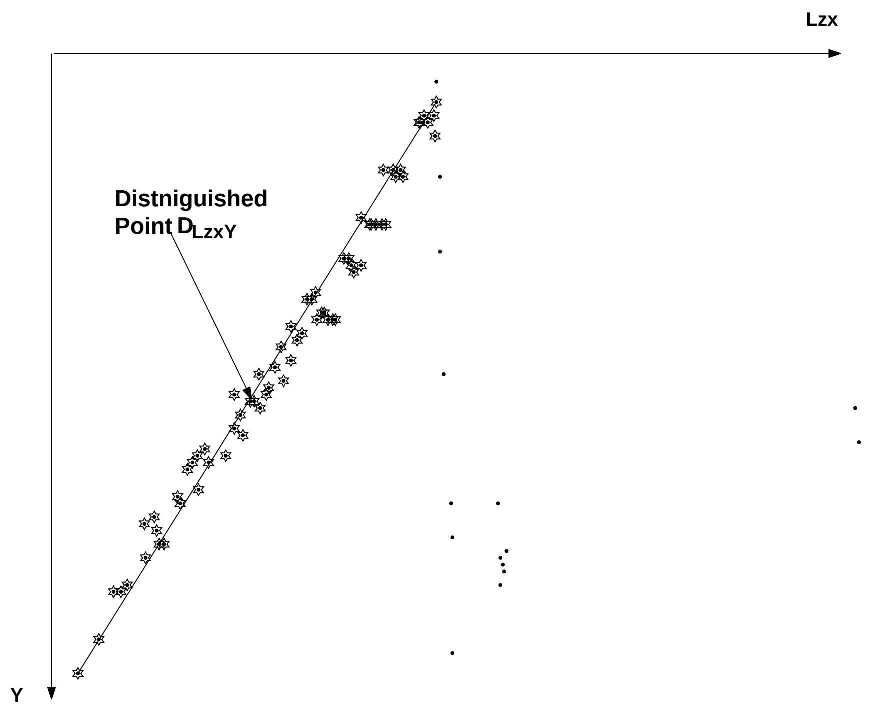
7.3.1 Line detection in $3D$

In our case, each edge point constraints two out of the four line parameters. It is thus not possible to directly apply the decomposition technique discussed in section 7.2, because we cannot select a particular cardinality of the distinguished set which will allow for constraining the transform to lie on an one-dimensional curve, as in the $2D$ case. Therefore, we came to the idea to break the problem down to $2D$ subproblems, and the natural way to do it is to examine two such subproblems, each recovering two line parameters using the RUDR technique.

In detail: A trial is initiated by randomly selecting a distinguished point $\mathbf{D}(X_d, Y_d, Z_d)$, which supposedly belongs to the linear boundary $\mathbf{L}$, shown in the edge map of Fig. 7.4 (c).



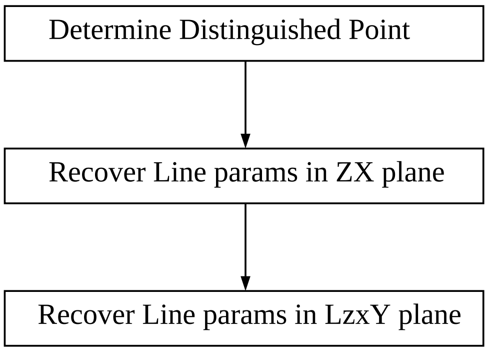
(a) Line detection in the image (**ZX**) plane

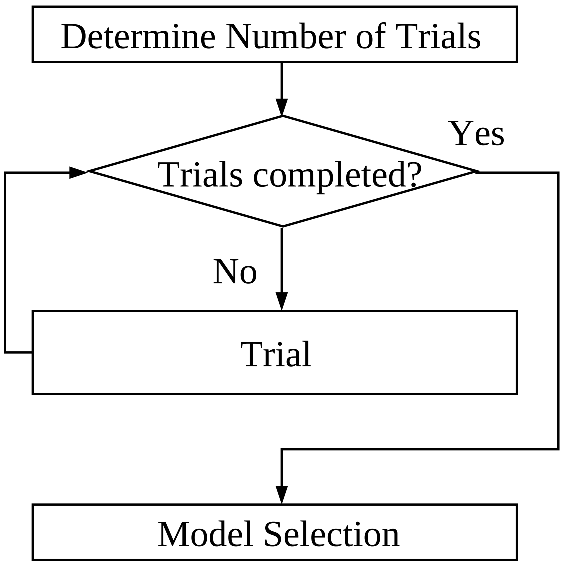


(b) Line detection in the $\mathbf{L_{ZX}Y}$ plane

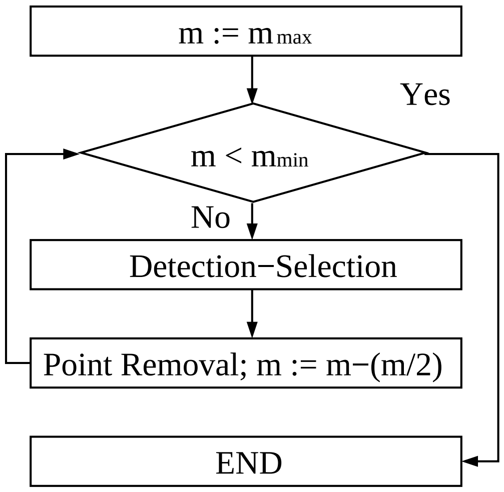
Figure 7.1: 3D Line detection in two steps

At first, the two parameters of the $2D$ orthogonal projection of $\mathbf{L}$ to the image plane ($\mathbf{ZX}$) are estimated. Fig. 7.1 (a) illustrates: The orthogonal projections (image plane coordinates) of all the edge points to the image plane are taken into account. Lets consider $\mathbf{D_{zx}}$ the projection of $\mathbf{D}$. A two-dimensional RUDR trial is performed on the image plane with $\mathbf{D_{zx}}$ as distinguished point and the parameters of $\mathbf{L_{zx}}$ are retrieved. The corresponding range points to the pixels contributed to the accumulator's maximum (drawn as "*" in Fig. 7.1 (a)) are then projected to the plane defined by $\mathbf{L_{zx}}$ and the axis $\mathbf{Y}$ of the sensor coordinate system. Lets consider now $\mathbf{D_{L_{zx}Y}}$ the projection of $\mathbf{D}$ to this plane. A second two-dimensional RUDR trial is performed on this plane with $\mathbf{D_{L_{zx}Y}}$ as distinguished point to retrieve the remaining two parameters of $\mathbf{L}$. Fig. 7.1 (b) illustrates. The range points finally determined to belong to the line $\mathbf{L}$ correspond to the $2D$ points drawn as "*" in this figure. The flow diagram of the trial for detection of a $3D$ line is depicted in Fig. 7.2 (a).


Determine Distinguished Point
Recover Line params in ZX plane
Recover Line params in LzxY plane
(a) Trial



(b) Detection-Selection



(c) Overall

Figure 7.2: Flow diagrams

7.3.1.1 Model selection

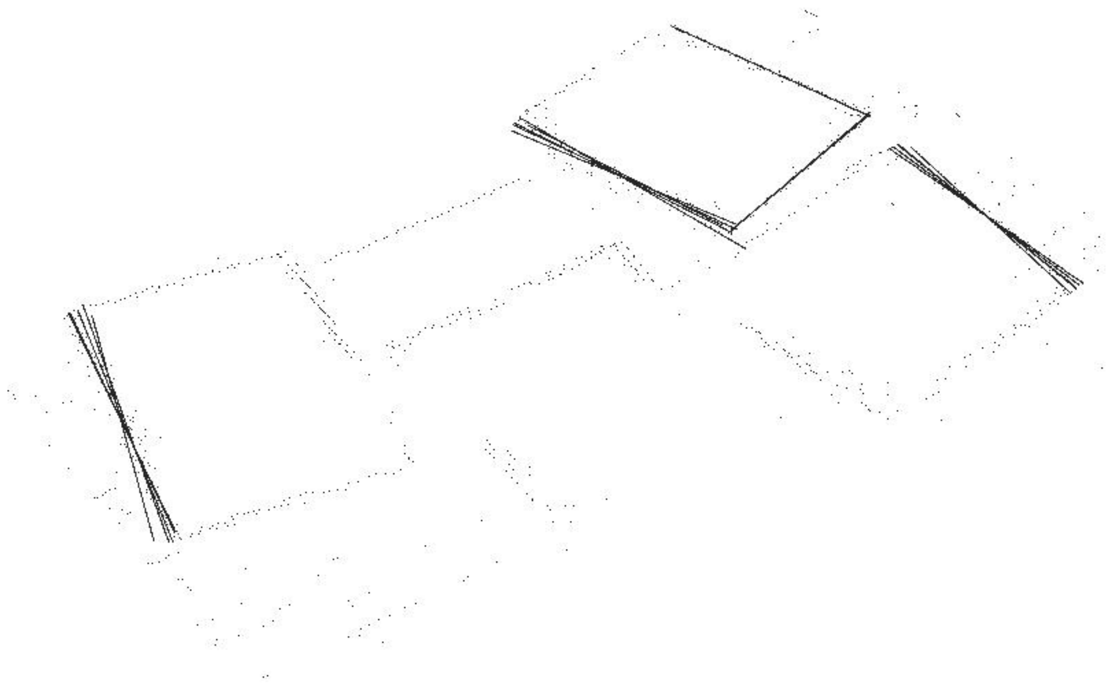
Boundary detection to the edge map of Fig. 7.4 (b) is presented in Fig. 7.3 (a), where the detected lines are superimposed to the edge map. The problem observed, is that the line detection process outputs redundant lines. This is a consequence of the randomization. Our algorithm as is, cannot guarantee that more than one points belonging to the same boundary will not be used as distinguished pixels of the recovery process. Simply removing the range points determined to belong to a boundary after a successful trial and continue the process is questionable. We cannot assure that some of these points cannot be used by a later trial to recover a model which represents the boundary better. Instead of retaining a locally sufficient model it is preferable to wait until all the trials take place and retain the models which satisfy some global optimality criteria. It is logical to assume that a recovered model should be favored over another if it describes a bigger number of image points more accurately. The latter statement is a simplified version of the Minimum Description Length (MDL) principle for model selection, which has been used quite frequently in various computer vision applications, lately in [77], [142].

We adopt the strategy of [77] p.123 for formalizing our approach, mainly due to its compactness and simplicity. A detailed description of the particular approach has already been presented in section 5.1.3.2.2, and can be applied to our case by considering that we use three dimensional lines instead of superquadrics as modeling elements. The function the maximization of which will result to the selection of an optimal set of linear models is shown in (5.13). In contrast to the approach of section 5.1.3.2.2, we *always* penalize overlapping models. Hence, the off-diagonal elements of the matrix **Q** in our case are always going to be negative and proportional to the number of points explained by both models, as expressed by (7.2).

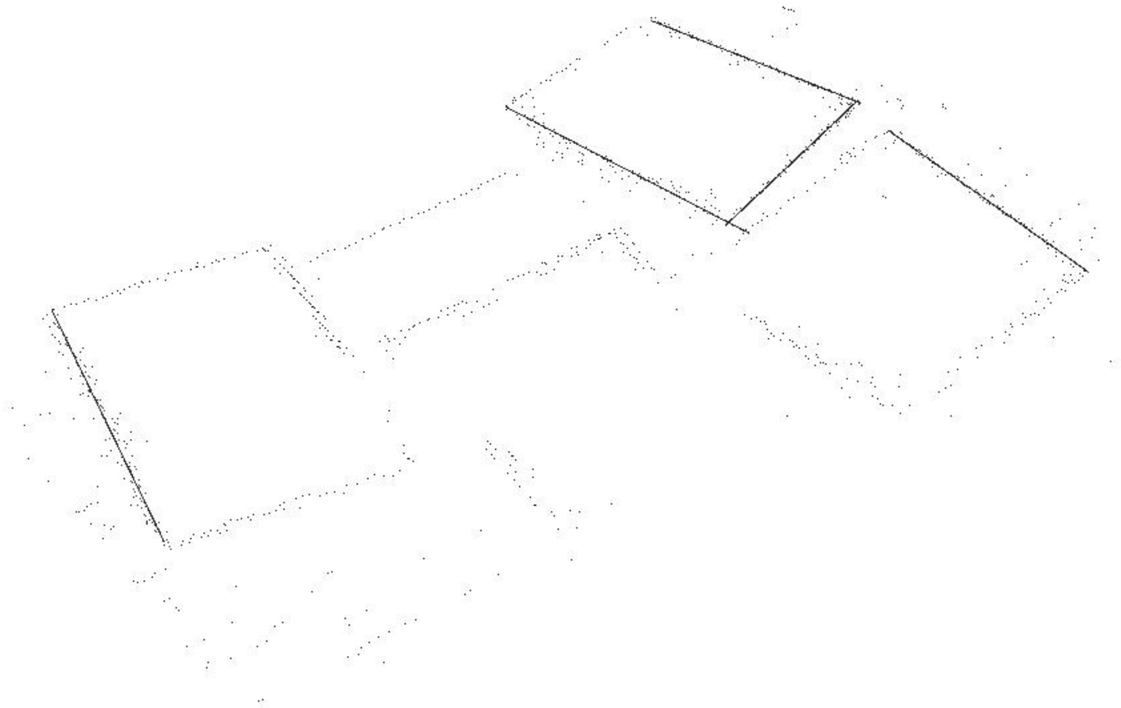
$$c_{ij} = -\frac{k_1|\mathbf{R}_i \bigcap \mathbf{R}_j|}{2} \tag{7.2}$$

Fig. 7.3 (b), illustrates the results of model selection on the lines detected in Fig. 7.3 (a).

We name the process of line Detection followed by model Selection the **D-S** process, the flow diagram of which is illustrated in Fig. 7.2 (b). The algorithm's complexity is the complexity of the line detection plus the complexity of the selection that is $O(tn)+O(M^2) \approx O(n)$, since $t = O(1)$ and $n \gg M$. In terms of memory consumption, all we need is one dimensional accumulator of size $O(n)$ plus a two dimensional matrix of size $M \times M$. The **D-S** process inherits its robustness from the RUDR approach but it is more accurate because the best segments are retained by the selection process. Note, that the idea of applying model selection after the Hough transform for improving the latter's performance is not new. In [13], the MDL criterion is used after the application of the Standard Hough Transform for detecting two dimensional lines in images. In this work however, no overlapping model penalization takes place within the MDL process. Hence, the MDL process is applied twice, which reduces the efficiency of the overall approach.



(a) Before segment selection



(b) After segment selection

Figure 7.3: Effect of segment selection

7.3.1.2 Acceleration via point removal

Our algorithm as is, performs t trials followed by model selection to extract $3D$ lines from the input edge map on which at least m points lie. The model selection, guarantees that the remaining linear segments describe the edge points to which they correspond in an optimal way. In other words, it is highly probable (this probability is given by the quantity $1 - \gamma$) that no other segments can be found comprising m or more points other than those already discovered.

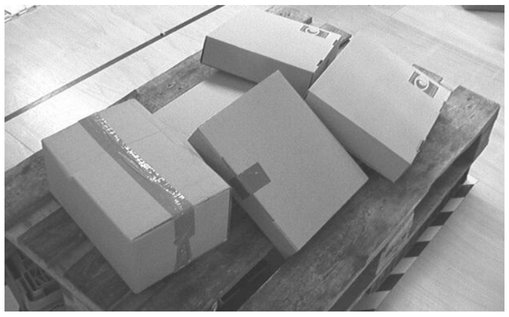
This observation results in a substantial algorithm acceleration: We adopt an iterative approach, every iteration of which comprises a **D-S** process retrieving lines with at least m points followed by a point removal, so that all points determined by the **D-S** to lie on lines are eliminated from further consideration. We start by looking for long segments, so m is assigned a big value (m_{max}) and then we gradually reduce the number of points expected to be found on a segment by $\frac{m}{2}$ until a lower threshold (m_{min}) is reached. The execution time of the **D-S** step is proportional to the number of edge points in the image times the number of trials. The latter is inversely proportional to the number of points expected to lie in the lines. By looking for lines comprising many points first, we reduce the number of trials and thus the execution time of the current **D-S** step. By point removal reduction of execution time of the subsequent **D-S** steps is guaranteed. Thus, an overall algorithm acceleration is realized. The flow diagram of the entire algorithm is depicted in Fig. 7.2 (b).

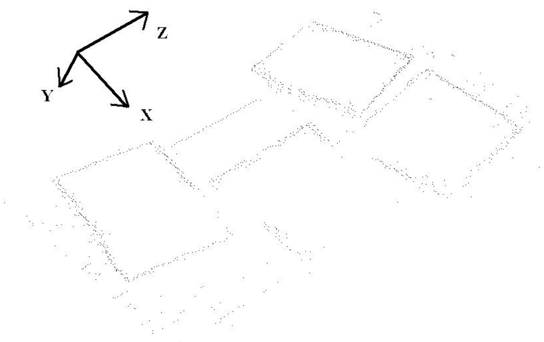
7.3.2 Boundary grouping

We define a $3D$ vertex as an aggregate consisting of two orthogonal $3D$ linear segments and a vertex point defined by their intersection. In the ideal case, two linear segments **X**,**Y** comprise a vertex if the dot product of their direction vectors **x**,**y** is zero, that is: $r = \mathbf{x}^T\mathbf{y} = 0$. However, due to uncertainty in the estimation of the segment parameters the dot product can never be exactly zero and a threshold must be introduced to determine the validity of a grouping hypothesis. The threshold depends on the uncertainty in the calculation of the line parameters and thus is difficult to define. The dot product is a bilinear function of the direction vectors. Hence, rigorous uncertainty propagation can be achieved and a statistical test can determine whether the grouping hypothesis is to be rejected or not, based on a user defined significance value. In [52] a compact framework for testing uncertain geometric relations is presented, on which our method is based. If we assume Gaussian noise and if $\mathbf{\Sigma}_{xx}$ and $\mathbf{\Sigma}_{yy}$ are the covariance matrices of the direction vectors **x** and **y** respectively, then the variance of their dot product is given by the expression:

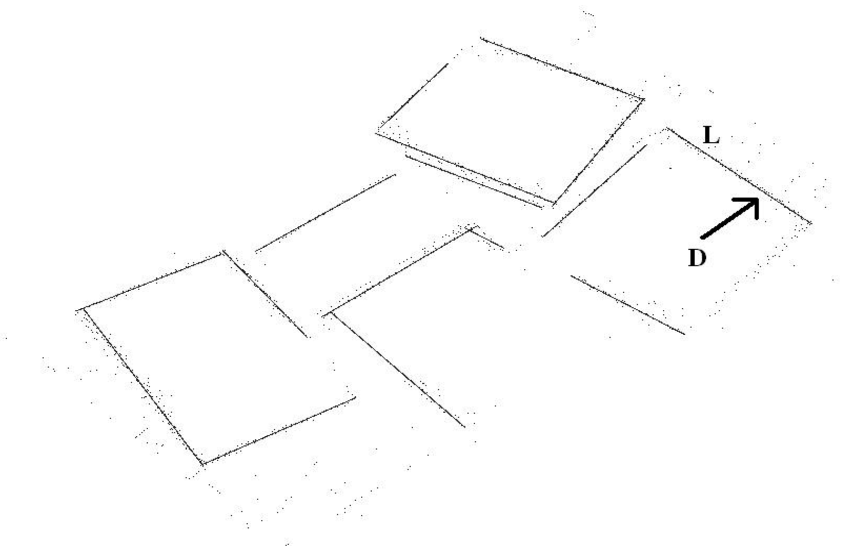
$$\sigma_r^2 = \mathbf{x^T}\mathbf{\Sigma}_{yy}\mathbf{x} + \mathbf{y^T}\mathbf{\Sigma}_{xx}\mathbf{y} \tag{7.3}$$

The optimal test statistic for the hypothesis $H_0 : r = 0$ is given by: $z = \frac{r}{\sigma_r} \sim N(0,1)$. We select a significance value α and compare the value z with the value $N_{1-\alpha}(0,1)$. If $z > N_{1-\alpha}(0,1)$, the grouping hypothesis is rejected. In all our experiments α was set to 0.05. The overall grouping algorithm has as follows: All possible pairs of detected lines are considered and those pairs passing the statistical test along with their intersection points are inserted to the set of the detected vertices.


(a) Intensity



(b) Edge Map



(c) 3D linear segments

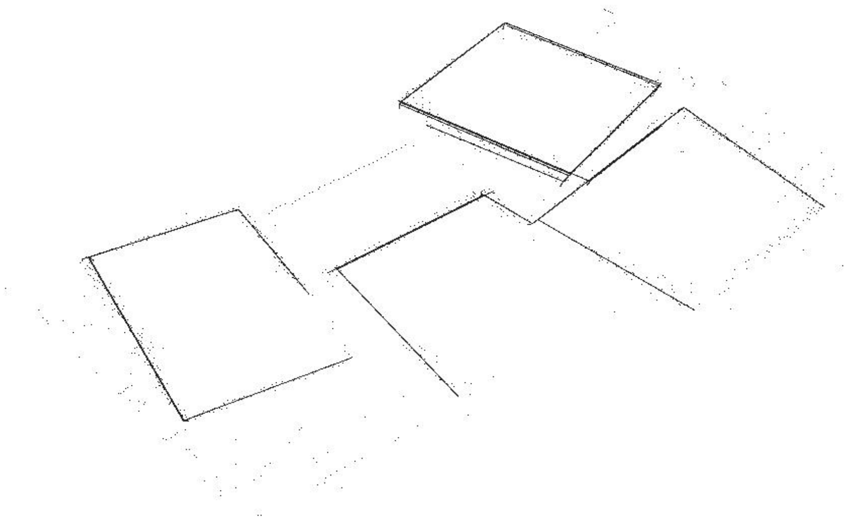

(d) Vertices

Figure 7.4: Vertex detection of rigid boxes (Configuration 1)

7.3.3 Experiments on vertex detection

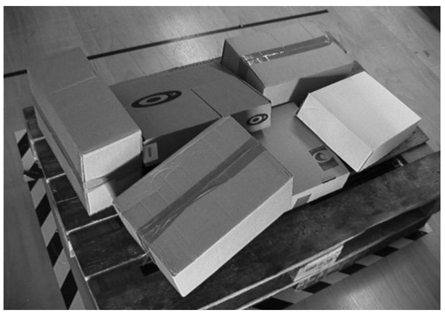
We applied our algorithm in various range images corresponding to piled boxes. Two test cases are presented here, illustrated in figures 7.4 and 7.5. The edge map, the recovered $3D$ line segments and the extracted vertices are as well depicted. In both cases all the objects linear boundaries were successfully recovered except those which were very noisy and comprised few number of points. For the boxes case we had $n = 1015$ edge points and assumed error of $\delta = 0.6$ pixels during the detection in the image plane and $\delta = 4$ pixels during line detection in the $\mathbf{L_{ZX}Y}$ plane. The corresponding values for the sacks test case were $n = 1260$, $\delta = 1$, and $\delta = 4$. In both cases, the probability of failure was $\gamma = 0.1$. This implies that we *deliberately* allow about 10 per-cent of the object linear boundaries to be missed by our algorithm for the sake of computational efficiency. The model selection parameters were set to $K_1 = 1$, $K_2 = 0.1$. Two algorithm iterations were executed: The first detected lines comprising at least 60 and the second 30 range points. The execution time for vertex detection was about 12 seconds in both cases in a Pentium $3, 600MHz$. Note that if we execute only one iteration in the context of which lines comprising 30 points are sought from the first place, the execution time rises to about 19 seconds. This verifies that the iterative algorithm and point removal actually reduce the overall execution time.

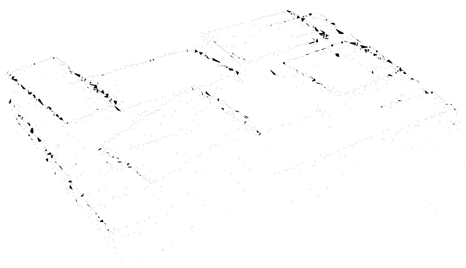
7.4 Recovery of dimensions

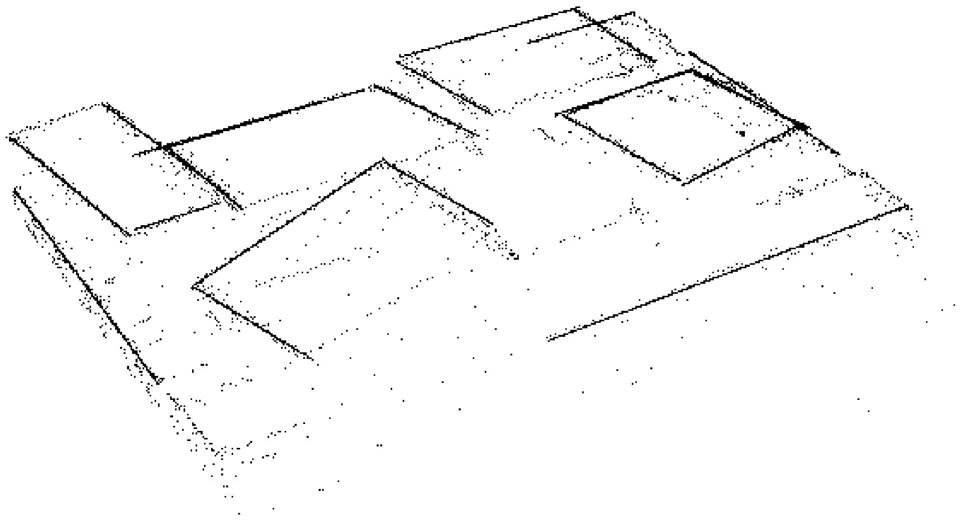
The dimensions of the boundary of an exposed (graspable) surface of a known pose, can be directly determined from two diagonal vertices of it. In our case, not all the linear boundaries and thereby not two diagonal vertices of each graspable surface can always be detected. To be able to infer the dimensions of the boundary of an exposed surface even in cases when two not diagonal or only one of its vertices is detected, we employ an approach which uses both the already extracted vertices as well as the edge points. The algorithm for finding graspable surfaces of boxes in range images is presented in Fig. 7.6.

The procedure **findGraspableSurfaces** (see Fig. 7.6, line 1) attempts to recover the graspable surfaces. Input of the procedure is the set of detected vertices $\mathbf{V}$. For every element V_i of the set, a rectangle graspable surface boundary R is initialized (line 2). The pose of R is recovered, by alignment with V_i. Thereby V_i will be hereinafter referred to as the *generating vertex* of R. Then, the algorithm finds the dimensions of R: At first it attempts to do so by finding a scene vertex which lies diagonal to V_i (line 4). If such a vertex cannot be found it attempts to recover the dimensions from edge points (line 7). If one of the two processes is successful, R is added to the list of found graspable surface boundaries $\mathbf{R}$ (line 5,8).

The procedure **dimensionsFromVertices** (line 11) aims at recovering the dimensions of the input rectangle R by finding a scene vertex which is diagonal to the rectangle's generating vertex. Such a vertex should be on the same plane to which the generating vertex belongs and its direction vectors should be parallel to the corresponding direction vectors of the generating vertex (line 13). In addition, its intersection point should reside at the first quadrant of the coordinate frame defined by the generating vertex (line 14). When a vertex satisfying the above criteria is encountered, the algorithm updates the width and length parameters of the rectangle R (line 15). There are cases however when a vertex with correct properties is found, which belongs to the boundary of an exposed surface of a different box.


(a) Intensity image


(b) Edge Map


(c) 3D linear segments

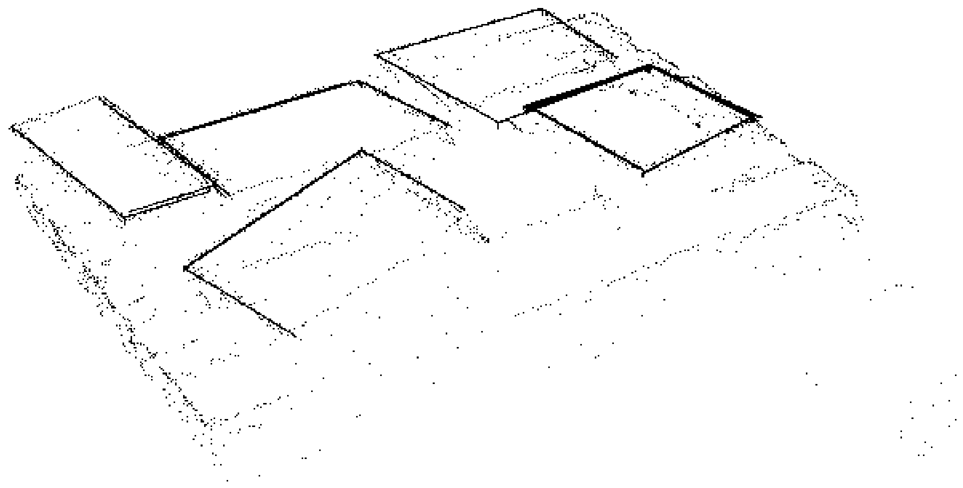

(d) Vertices

Figure 7.5: Vertex detection of rigid boxes (Configuration 2)

In order to identify such cases we regard the range points inside R. If the average distance of the points to the plane defined by R is small enough, we consider the rectangle successfully localized. This test is realized by the procedure **verify**, invoked in line 16. Points inside R are acquired via a computationally efficient region rasterization framework [153].

```
findGraspableSurfaces($\mathbf{V}$, $\alpha$, $p$):
    For every vertex $V_i \in \mathbf{V}$ /* $\mathbf{V}$ is the set of detected vertices*/
        consider Rectangle R; align $R$ with $V_i$
        If dimensionsFromVertices($R$, $\mathbf{V}$, $\alpha$) Then
            add $R$ to $\mathbf{R}$ /* $\mathbf{R}$ is the set of recovered graspable surface boundaries*/
        Else
            If dimensionsFromEdges($R$, $\alpha$, $p$) Then
                add $R$ to $\mathbf{R}$
    select($\mathbf{R}$) /* Retain the "best" boundaries*/
    Return $\mathbf{R}$

dimensionsFromVertices($R$, $\mathbf{V}$, $\alpha$):
    For every vertex $V_j \in \mathbf{V}$
        If coplanar($R$, $V_j$, $\alpha$) and parallel($R$, $V_j$, $\alpha$)
            If inFirstQuadrantOf($R$, $V_j$) Then
                update dimensions of $R$
                Return verify($R$, $\alpha$)
    Return False

dimensionsFromEdges($R$, $\alpha$, $p$):
    $\mathbf{P_c} \leftarrow$ preProcess($R$, $\alpha$) /* $\mathbf{P_c}$: the set of candidate edge points */
    $\mathbf{A_x}$, $\mathbf{A_y} \leftarrow$ accumulate($\mathbf{P_c}$)/*$\mathbf{A_x}$,$\mathbf{A_y}$: one dimensional accumulators*/
    For every peak $A_x \in \mathbf{A_x}$
        $M_x \leftarrow$ parameter value corresponding to $A_x$ (width)
        For every peak $A_y \in \mathbf{A_y}$
            $M_y \leftarrow$ parameter value corresponding to $A_y$ (length)
            $\mathbf{P_i} \leftarrow$ points which contributed to $A_x, A_y$
            $\mathbf{P_f} \leftarrow$ {points $P(x,y) \in \mathbf{P_i} : x \leq M_x \wedge y \leq M_y$}
            If $\mathbf{P_f}$.size() $> p$
                dimensions of $R \leftarrow M_x, M_y$
                Return verify($R$, $\alpha$)
    Return False
```

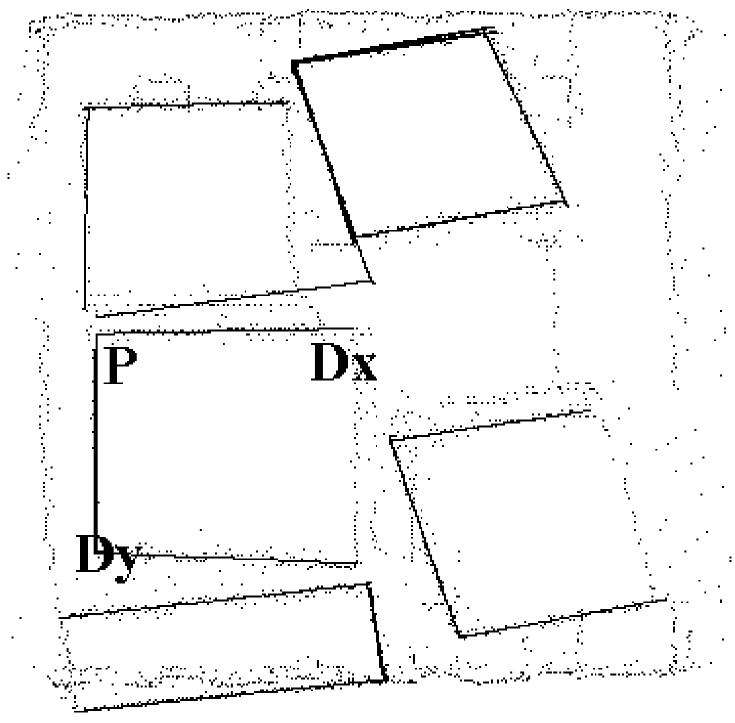
Figure 7.6: Algorithm for finding graspable surfaces of piled boxes

The procedure **dimensionsFromEdges** (line 18) recovers the dimensions of the input rectangle R, in the event of insufficient vertex information, that is when no diagonal vertex to the generating vertex of R can be found. We infer dimension information from the edge points expected to reside on R. These points should satisfy the following requirements: Firstly, they should be coplanar to the plane defined by R. Secondly, they should be in the first quadrant of the coordinate frame defined by its generating vertex. Procedure **preProcess** (line 19) realizes these actions. To illustrate, we consider the scene vertex $\mathbf{P}$ of Fig. 7.7 (a), which depicts a top down view of Fig. 7.5 (d), as the generating vertex of R. Fig. 7.7 (b), shows the coordinate frame defined by the generating vertex and the edge points found to be coplanar to the vertex. **preProcess** will output the set of edge points $\mathbf{P_c}$ on the first quadrant of the frame.

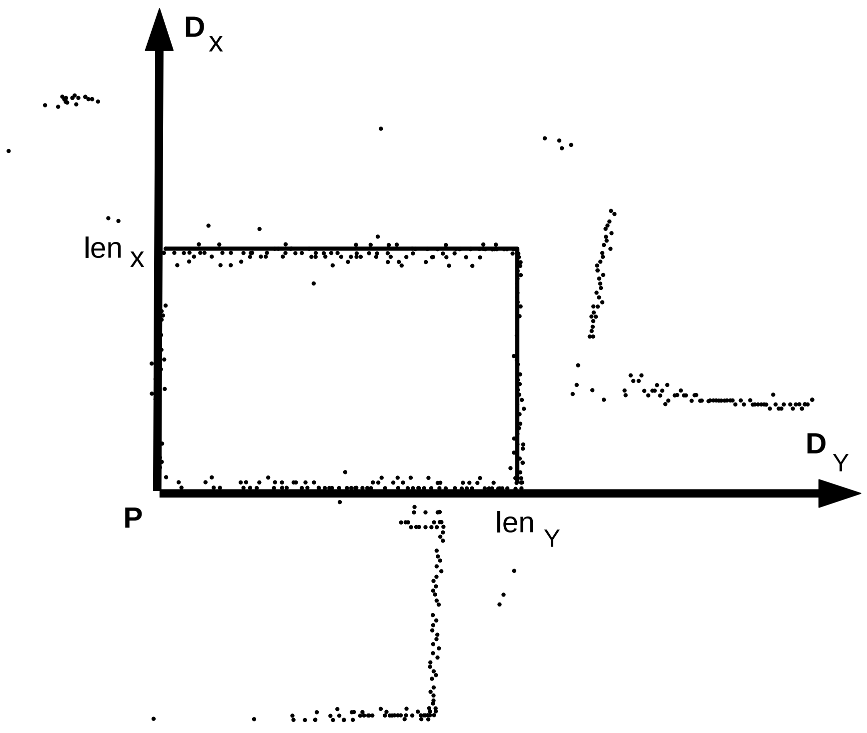
Application of a Hough transform -like technique on this set of edge points will determine the rectangle dimensions: The coordinates of the points in $\mathbf{P_c}$ along the $\mathbf{D_x}$ and $\mathbf{D_y}$ axes of the two dimensional vertex coordinate frame are accumulated in two one dimensional arrays $\mathbf{A_x}$ and $\mathbf{A_y}$ respectively (line 20 of Fig. 7.6). A search procedure for the rectangle dimensions in the accumulators follows: For each pair A_x, A_y of accumulator peaks, we examine the corresponding parameter values M_x and M_y which form an hypothesis about the width and length of the rectangle (see lines $21 - 24$). We then consider the set of edge points $\mathbf{P_i}$ which contributed to the current peaks A_x and A_y (line 25). The subset $\mathbf{P_f}$ of this set, containing points which belong to the rectangle should have coordinates lower or equal to the parameter values M_x and M_y (line 26). If the number of elements of $\mathbf{P_f}$ is bigger than a user defined threshold p, we regard the rectangle hypothesis to be successfully supported by boundary information and we update its dimension parameters (line $27 - 28$). A region based verification approach as in line 16 takes the final decision about the validity of the hypothesis (line 29). The advantage of this technique with regard to a standard implementation of the Hough transform is efficiency, since accumulation and search for peaks is performed in one dimensional structures.

Our framework attempts to recover graspable surface boundaries by examining every detected vertex (see line 2 of Fig. 7.6). This results in the localization of redundant graspable surfaces when more than one vertices per surface have been detected. The procedure invoked in line 9 selects those recovered boundaries which describe the scene in terms of global accuracy and consistency by applying a minimum description length (MDL) approach, as in sections 7.3.1, 5.1.3.2.2. In addition, independent graspable surface boundary recovery triggered by each detected vertex allows for parallel implementation of the algorithm: A separate processor can be used for dealing with each vertex of the vertex set.

Throughout our analysis we had to test relations of various geometric entities. We had to find out for example whether two detected vertices are coplanar (look at line 13 of Fig. 7.6), if the direction vectors of two vertices are parallel (line 13), if an edge point belongs to a plane defined by a detected vertex (line 19), or if the points inside a hypothesized boundary belong to the plane it defines (lines 16, 29). Introduction of empirically defined thresholds for deciding the validity of the relations leads to a non robust system. This problem can be avoided when taking into consideration the error in calculating the geometric entities and statistically testing the geometric relations. If so, all thresholds can be replaced by a unique value, the significance level. We have performed all tests statistically, using the framework



(a) Top- down view of configuration



(b) Recovered rectangle

Figure 7.7: Recovery of graspable surface dimensions from edge points

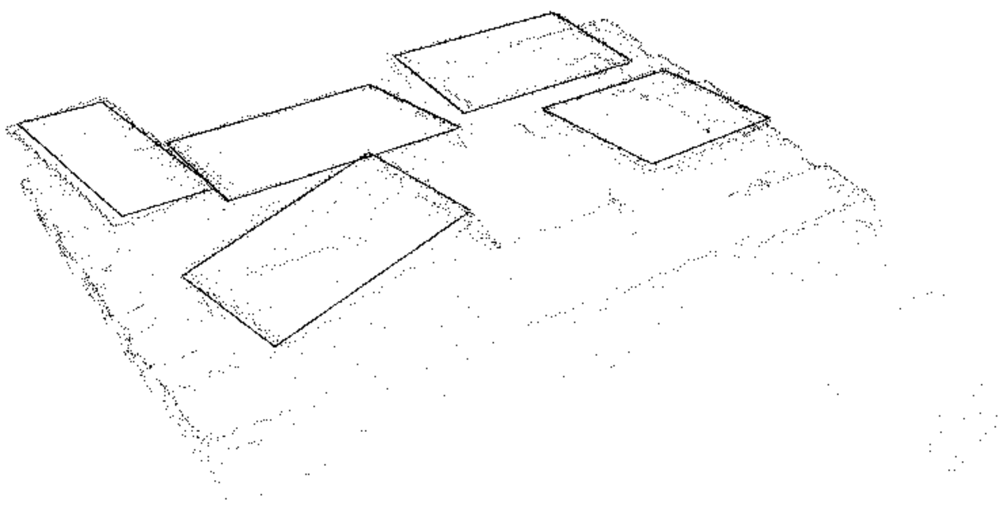
in [52], because of its simplicity and compactness. We denote the significance level by α in our pseudo code, appearing as input to every procedure where geometric relations are tested (e.g in lines 13,19,16,29).

7.5 Experiments

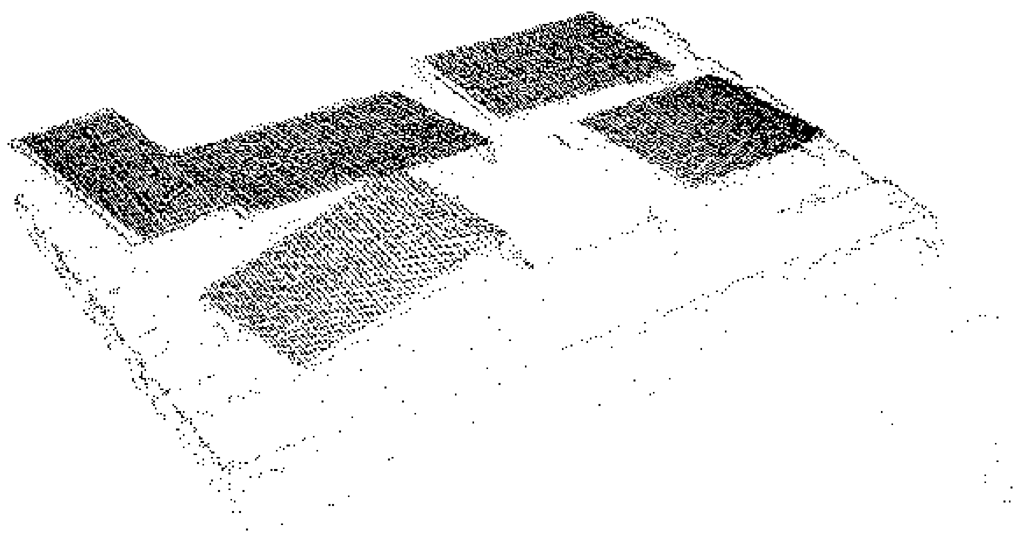
The output of our algorithm applied to the test case of Fig.7.5 (a) is shown in Fig. 7.8. Fig. 7.8 (a) depicts the detected boundaries of the graspable surfaces and Fig. 7.8 (b) shows the range points inside the detected boundaries, which led to the verification of the particular boundary hypotheses. We have performed a number of experiments of the algorithm using card board boxes. A Pentium 3, $600Mhz$ was used for our experiments. The overall average execution time of the algorithm algorithm was 55 seconds. Edge detection lasted 10 seconds, vertex detection 14 and the dimension recovery about 31 seconds. The average processing time for dimension recovery from a single vertex was 3 seconds. This means that in the event a parallel implementation for object recovery is employed the overall execution time will be less than 30 seconds on the average. In terms of robustness, our experiments demonstrated that the system only occasionally fails to recover all the graspable surfaces in the pile. According to initial accuracy measurements the translational grasping accuracy was less then 1.5 cm, almost equal to the accuracy of the sensor employed. In the future we intend to continue experiments for the system evaluation.

7.5.1 Operation example

We used the approach described in the preceding section for object recovery within the context of our robotic framework. Fig. 7.9, illustrates the system in operation. In this sequence of images the robot unloads the configuration of objects of fig. 7.9 (a). Fig. 7.9 (b)-7.9 (g) illustrate the configuration after grasping of a particular object. The last figure, demonstrate the configuration that has been created by the robot after all objects have been unloaded and placed at a user defined position. It is clear in this figure that the robot manages to grasp and hence place the objects in a quite accurate way. This demonstrates that using the Hough transform for object recovery is not only advantageous in terms of computational efficiency, but results in the creation of an accurate robotic system.



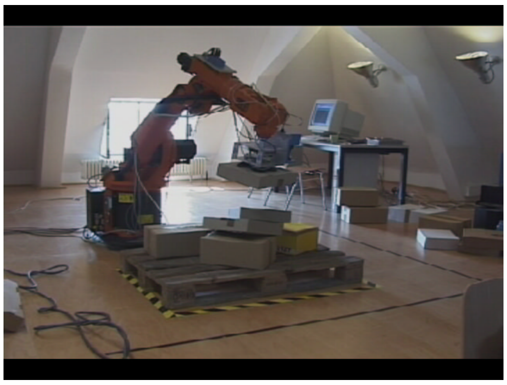
(a) Boundaries



(b) Points inside boundaries

Figure 7.8: Recovered graspable surfaces

(a) Configuration

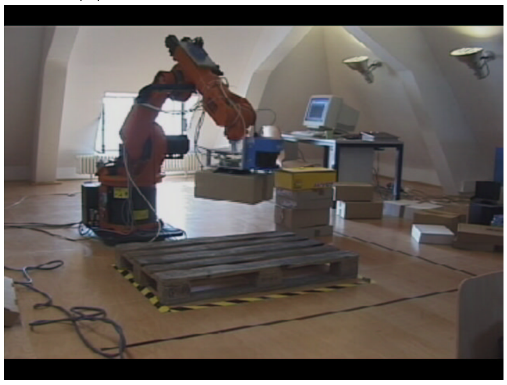

(b) Removal of first object

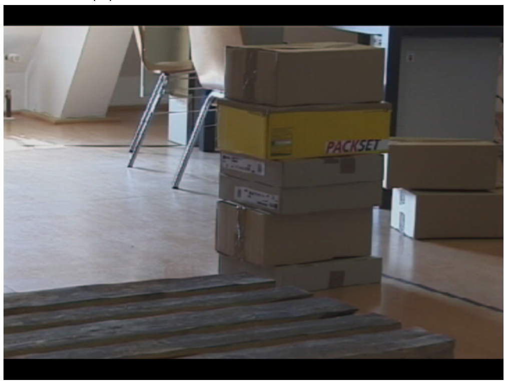
(c) Removal of second object

(d) Removal of third object

(e) Removal of fourth object

(f) Removal of fifth object


(g) Removal of sixth object



(h) Object placement

Figure 7.9: Our robotic system in operation

7.6 Conclusions and future work

In this chapter we described an approach for recovering piled *rigid boxes* from range images. This was done using the Hough transform on edge images generated by means of an edge detection process in range images acquired by a laser sensor. This approach has been employed for the creation of a real time robotic system for object unloading from target platforms. As demonstrated in the preceding section, our system exhibits computational efficiency, in combination with accuracy and robustness. Compared with the approach for object recovery using superquadrics as modeling elements presented in chapter 5, this method is considerably faster.

Our system is designed to localize the rectangle boundary of the exposed surfaces of the piled objects. The height of the objects is not recovered, and the main reason for this is that since the sensor acquires images of the top side of the configuration, not enough information regarding the height of the objects exists in the input image. As discussed in chapter 5, a solution to this problem is to use an additional sensor for object height calculation after its grasping by the robotic hand.

In the future we intend to perform a detailed *quantitative* assessment of the accuracy and robustness of this approach. Note, that since both the hypothesis generation and refinement framework and the hough transform are quite popular for recovering geometric parameter models, an in depth comparison of the approach of this chapter with the one of chapter 5 may reveal additional interesting aspects of the advantages and problems of each. This is as well target of our future work.