Joris Jan Bayer

Multiple Model Fitting Revisited

Master's Thesis

Graz University of Technology

Institute for Computer Graphics and Vision
Head: Univ.-Prof. Dipl-Ing. Dr.techn. Dieter Schmalstieg

Supervisor: Dipl.-Ing. Dr.techn. Michael Donoser
Evaluator: Univ.-Prof. Dipl.-Ing. Dr.techn. Horst Bischof

Graz, May 2013

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Graz, ____________________ ____________________

Date Signature

Eidesstattliche Erklärung[1]

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am ____________________ ____________________

Datum Unterschrift

[1]Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008; Genehmigung des Senates am 1.12.2008

Acknowledgements

I would like to thank Prof. Horst Bischof for granting me the opportunity to write this thesis. My highest gratitude and appreciation go to my supervisor, Dr. Michael Donoser, for his outstanding and continuous support. I would like to express special thanks to my girlfriend, who stood by me patiently throughout the creation of this work. Finally, I would like to dedicate this thesis to my parents, who granted me the possibility to follow my own path.

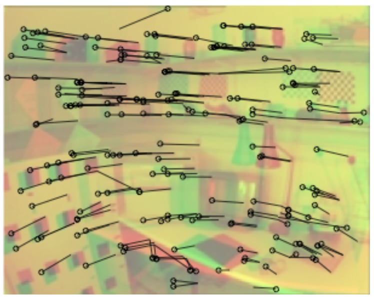
Abstract

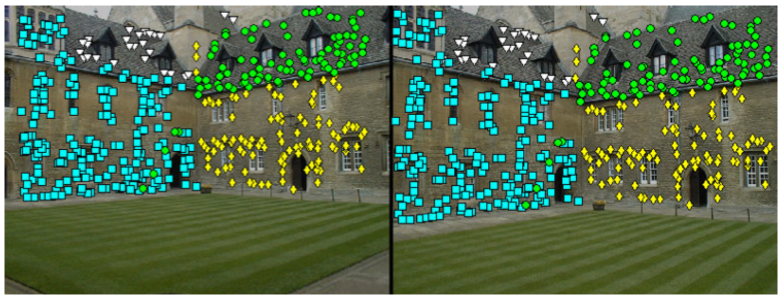
The task of robustly fitting multiple parametric models to a noisy set of observed data is a common necessity in computer vision. Applications range from geometric model fitting (e.g. plane fitting) through homography estimation to motion segmentation. This thesis explores existing model fitting approaches, almost all of which rely on the *random sampling* paradigm introduced by RANSAC in 1981. This principle solves multiple model fitting as an application-independent problem of generating candidate models from random subsets of data, from which the best models are selected based on a predefined quality measure. We mould existing approaches into a generic model fitting framework for MATLAB, built from exchangeable components for sampling strategies, outlier removal, model estimation and model selection. From the multitude of possible model fitting pipelines that emerges from this framework, we evaluate a selection of algorithms on the problem of multiple line fitting in synthetic two-dimensional data, considering both the quality of fit and the correctness of the number of models returned by the algorithms. We focus on approaches that identify clusters of observations based on their likelihood of having emerged from the same structure. For this purpose, we incorporate recent developments in graph clustering such as the Graph Shift and Authority Shift algorithms. In addition to the 2D experiments, we conduct experiments on the Freiburg and NYU datasets of depth data recorded by Microsoft's Kinect© sensor, employing the framework to the problem of plane fitting. With the second dataset, we use the included semantic labeling to extract reference planes (floors, walls, tables, etc.) as a quasi-ground truth, allowing for the evaluation of the detection precision/recall of our algorithms.

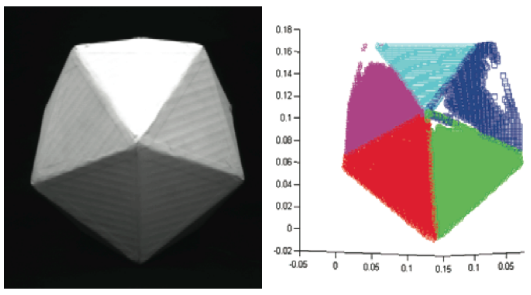
Contents

1 Introduction

A reoccurring challenge in computer vision tasks is the estimation of one or more parametric models in observed sets of noisy data. The true structures that these models represent range from simple geometric shapes (lines, circles, planes) to complex relations such as a homography between two images of the same planar scene [16, p. 123], the fundamental matrix between two cameras [16, p. 291], or motion segmentation [44]. Applications such as acoustic source localization [31] and even speech-based emotion recognition [11] show that model fitting is not a problem limited to the field of computer vision. Typically, observations that actually belong to a structure of interest are outnumbered by random, unrelated observations, which calls for robust methods in order to solve the fitting problem.


(a)


(b)


(c)

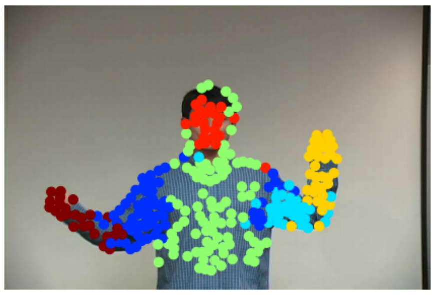

(d)

Figure 1.1: Examples of single and multiple model fitting applications. From left to right: estimation of the fundamental matrix, multiple homography estimation, range segmentation, motion segmentation. Sources: [13], [46], [41], [44].

1.1 Single model fitting

Let us first consider the scenario in which we expect only a single structure in the observed data. In this case, the observed data points can simply be divided into two categories: *inliers*, belonging to the structure of interest, and *outliers*, random observations outside the structure. If the number of outliers is zero, the single model fitting problem comes down to regression:

With a *residual function* $r(\theta, \mathbf{x})$ that measures the error of fitting data point $\mathbf{x}$ with model θ, the well-known *least squares-estimator* chooses a model θ that minimizes the sum squared error

$$SSE = \sum_{j=1}^{n} \left(r(\theta, \mathbf{x}^j) \right)^2 , \tag{1.1}$$

for data points $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^n$. This estimator is optimal for Gaussian error distributions [30, p. 4]. However, one can easily show that even a single outlier can heavily corrupt the outcome of such an estimator, as one may see from Figure 1.2.

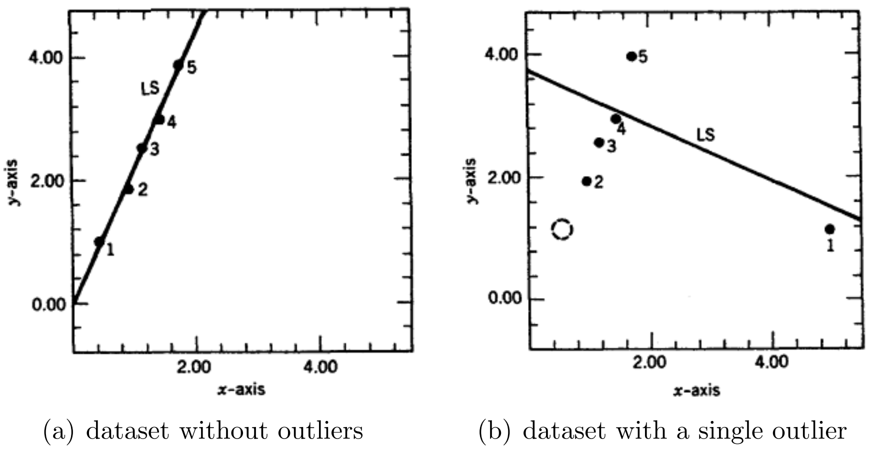


Figure 1.2: Fitting a line to a dataset of five points. The least squares estimator fits models optimally for Gaussian error distributions, but does not handle outliers well. Source: [30, p. 5].

A multitude of approaches that are robust towards outliers has been proposed over the years, such as LMedS, which replaces the sum in Equation 1.1 with the median, or the class of M-estimators, which replaces the quadratic term in Equation 1.1 with a function of r such that the influence of outliers is diminished. These methods tend to break down, however, if the number of outliers is greater than 50%, as they were designed under the assumption that the majority of data support the structure [41].

For outlier rates above 50%, the "hypothesize-and-test" paradigm has been widely embraced, which picks a definitive model from a large number of candidate models, called *hypotheses*, generated from minimal subsets of data points. RANSAC [12] is the most prominent representative of this technique, which will be discussed in detail in Chapter 2.

1.2 Multiple model fitting

Apart from the many applications of single model fitting, applications in which multiple structures or relations are to be extracted from observations are equally common. As single model fitting is the problem of estimating a model θ for given data $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^n$, the goal of multiple model fitting is the identification of *multiple structures*, i.e., finding a set of models $\theta_1, \theta_2, \ldots, \theta_m$ that fits the s structures present in the data. Note that the task not only includes the parametrization of the models — in most real world problems, the correct estimation of the number of models m is equally relevant.

The presence of multiple structures inherently leads to a more complex categorization of data points: From the viewpoint of a single structure, *inliers* are the points that belong to the structure in question. *Pseudo-outliers* are observations that are associated with a structure, bot not with the current structure. Finally, data points that aren't affiliated with any structure are denominated as *gross outliers*. Figure 1.3 illustrates these relationships. There is also a distinction among inliers themselves: while most inliers might be part of a single structure, some of them might be a member of multiple structures. For geometric structures such as lines, for example, these shared inliers occur at the intersections between the structures.

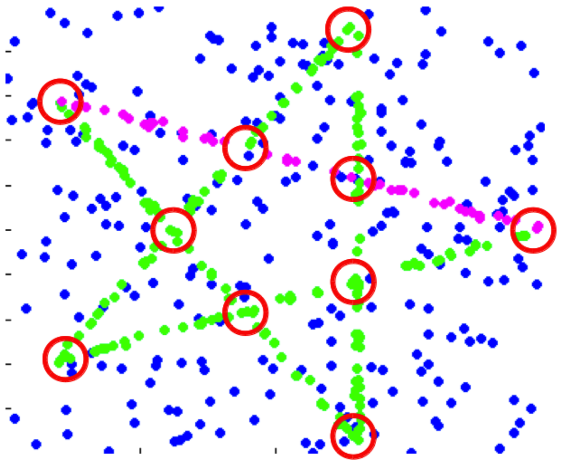


Figure 1.3: Types of points in a multiple model fitting problem. From the point of view of one structure, there are inliers (pink), pseudo-outliers (green), and gross outliers (blue). Multiple structure membership is possible where two structures intersect.

Confronted with the task of fitting multiple models, one might easily come up with the idea of applying a single model fitting procedure such as RANSAC sequentially, removing inliers of the previously fitted model before each iteration. This has been shown to be effective in many cases, but there are a few drawbacks: First of all, a stopping criterion has to be devised, be it a fixed number of iterations or a threshold for the quality of the fit, such as the number of acquired inliers for the current structure. The establishment of an adequate stopping criterion is all but trivial [4]. Secondly, by removing inliers after each run, these algorithms make the assumption that every point is associated with one structure at most. This, however, makes different models "compete" for shared inliers, which might reduce the fitting accuracy.

This has motivated the emergence of several approaches that allow multiple structure membership. Methods such as Mean Shift [9] and Randomized Hough Transform [43] work on the parameter space, while Kernel Fitting [4] clusters data points in an application-independent space based on their affinities with randomly generated hypotheses. Other approaches do not allow multiple model membership, but still yield good results, such as J-linkage [36], a hierarchical clustering, and Label Costs [10], which solves the multiple model fitting problem as a labeling problem. Examples for different multiple model fitting applications can be seen in Figures 1.1(b), 1.1(c) and 1.1(d).

1.3 Outline

We will survey existing solutions for single and multiple model fitting in Chapter 2. Apart from the fact that most of these approaches rely on random sampling, many have in common a certain abstraction that moves the fitting task away from the application domain: structures are represented by parametric models, which can be generated from a small subset of observations, the so called p-subset. A measure is required to determine the *residual* between a model and an observation, such that inliers can be dichotomized from outliers. Hence, whatever the domain of the application and the class of models may be, the algorithms will work as long as a model can be defined by p observations, and a function to measure residuals is provided.

For evaluation purposes, a third mapping is necessary: in order to determine the fitting accuracy, we need a distance function that measures the error between an estimated model and the corresponding structure defined by the ground truth.

The first contribution of this thesis will be a generalization of existing approaches into a handful of general *strategies*. One example for a strategy is Sequential RANSAC. Other approaches such as J-linkage [36], Local Graph Clustering [24] and Kernel Fitting [4] share a common strategy, which can be formulated as a generic pipeline consisting of *random hypothesis generation*, optional *gross outlier removal*, the computation of *affinities* between observations based on similar residuals with the hypotheses, a *clustering* of observations in the space defined by said affinities, a function to construct models from clusters, and finally, an optional model merging procedure, which may include information-theoretic criteria to select the best set of models for the presented problem. The actual algorithms used for each of these components can be chosen at will.

The above generalization of techniques materializes in our *model fitting framework* for MATLAB, which will be discussed in Chapter 3. It incorporates existing source code from approaches such as J-Linkage, Kernel Fitting and Label Costs [10]. This framework allows new combinations of elements used by existing model fitting pipelines, and incorporate recent developments in graph clustering algorithms. From the highly modular design of the framework, described in Section 3.1, a multitude of possible model fitting setups emerges, a subset of which is selected for evaluation and therefore described in Section 3.2.

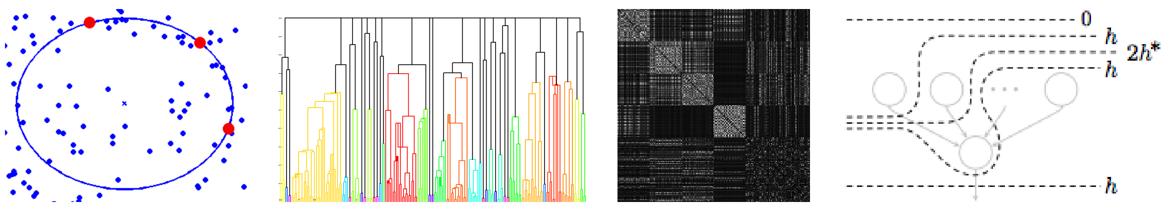
In Chapter 4, we evaluate our setups, both existing and new, on synthetic two-dimensional data, applying the algorithms to the problem of line fitting. We focus on the employment of existing model fitting pipelines in combination with new graph clustering algorithms such as Graph Shift, with some promising results. Section 4.2 centers on the *accuracy* of the algorithms when the number of ground truth structures is known. Subsequently, Section 4.3 assesses the *cardinality error*, i.e., how well the algorithms estimate the number of structures in the ground truth.

A few of the approaches evaluated in Chapter 4 are put to the test in Chapter 5, where they are applied to the real world problem of plane fitting. The data for these experiments come from two publicly available datasets of RGB-D data, i.e. color images with depth information, recorded by the Microsoft Kinect ©sensor. We visualize the results of our model fitting algorithms on the "Freiburg" dataset [35] in Section 5.1. In Section 5.2, we make use of the semantic labels in the NYU depth dataset [28] to extract reference planes, which we then try to detect with our algorithms, resulting in a comparison based on precision and recall.

From the results acquired in Chapters 4 and 5, we draw our final conclusion in Chapter 6.

2 Related Work

A multitude of approaches has been proposed for the general task of fitting multiple models to a set of observations. We will begin this chapter with a discussion of the RANSAC family of robust single model fitting algorithms, which lay the groundwork for all approaches that employ the "hypothesize-and-test" paradigm of choosing a winning model hypothesis from a large set of randomly generated candidates. We will also look briefly into Hough-based methods, which scan the entire *parameter space* for the most plausible model. Then, we will discuss several state-of-the-art multiple model fitting pipelines, each of which claims advantages over their predecessors in terms of parametrization complexity, fitting accuracy or speed. Figure 2.1 illustrates some concepts of multiple model fitting, each of which will be explained in detail later on.



(a) random sampling (b) hierarchical clustering (c) dense sub-graphs (d) graph cuts

Figure 2.1: Concepts used in (multiple) model fitting. Source for (c): [10].

2.1 Hypothesis generation

Almost all algorithms explored in this chapter are based on the assumption that a candidate model, or hypothesis, can be constructed from a minimal subset of p observations. That is, *at least* p observations are required to uniquely determine one model. In $\mathbb{R}^2$, any two distinct points $\mathbf{x} = [x_1, x_2]^T, \mathbf{y} = [y_1, y_2]^T$ define a line $[a, b, c]^T$. This follows from the pair of linear equations

$$\begin{aligned} ax_1 + bx_2 &= -c \\ ay_1 + by_2 &= -c. \end{aligned} \tag{2.1}$$

Similarly, three non-collinear points uniquely define a circle, which is illustrated in Figure 2.1(a). A fundamental matrix can be uniquely determined by seven point correspondences [16, p. 290], etc. The number of required points differs from application to application, just like the function that creates a model from these points, but as long as they are defined, the property that a parametric model can be determined by a p-subset paves the way for the paradigm of *random sampling*, i.e. the generation of random hypotheses by randomly chosen observations from a dataset.

2.2 The RANSAC family

RANSAC, short for "Random Sample Consensus", was first proposed by [12] in 1981 and constitutes a reference algorithm for fitting a single model to noisy, outlier-ridden data. It relies on the principle that a parametric model for the structure in question can be uniquely determined by a fixed number of observations, the so called p-subset. The algorithm creates a large set of hypotheses $\theta_1, \theta_2, \dots, \theta_H$ different p-subsets randomly picked from the n observations. The *consensus set* of such an hypothesis is defined as the set of observations $\mathbf{x}^j$ for which the residual remains below a threshold τ,

$$C_i = \left\{ \mathbf{x}^j \; : \; |r(\theta_{\mathbf{i}}, \mathbf{x}^j)| \leq \tau \right\}. \tag{2.2}$$

Members of C_i are also called the *inliers* of θ_i. The hypothesis θ_{i^*} with the most inliers is chosen as the winning hypothesis, and the final model is constructed by least squares estimation using the entire consensus set C_{i^*}.

Besides the inlier threshold τ, the principal parameter that has to be established is the number of hypotheses H necessary to have a high certainty of success. When the number of outliers is known, or can be estimated, H can be determined probabilistically: The algorithm is successful if at least one of the H picked p-subsets contains only inliers of the true structure (a so called *inlier hypothesis*). Hence, if the probability of picking an inlier is q, then the probability of success is

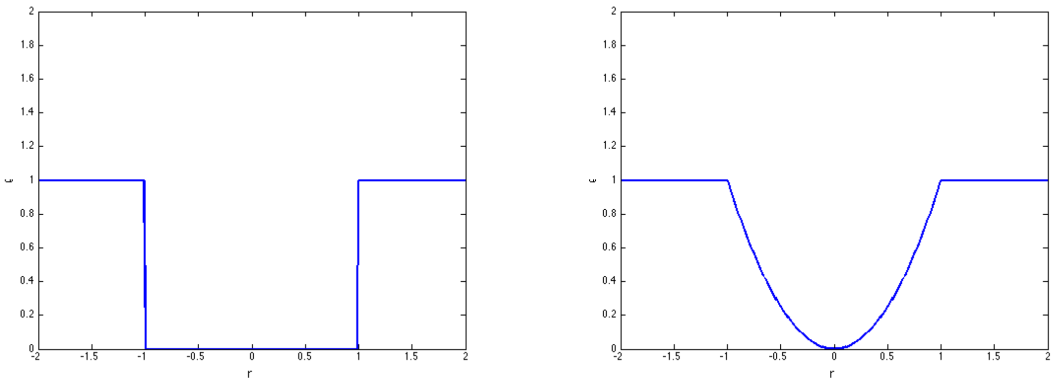
$$Pr\left\{\text{At least one inlier hypothesis}\right\} = 1 - (1 - q^p)^H. \tag{2.3}$$

In other words, if one desires a probability of success of, for example, 95% for a line fitting application with 50% outliers, then the number of randomly sampled hypotheses would have to be at least

$$H = \frac{\log(1 - 0.95)}{\log(1 - q^p)} \approx \frac{-1.3}{\log(1 - \frac{1}{2}^2)} \approx \frac{-1.3}{-0.125} \approx 11. \tag{2.4}$$

2.2.1 RANSAC variants

One way to improve RANSAC is to take into account inlier residuals, instead of simply ranking hypotheses by the size of their consensus sets. Effectively, regular RANSAC picks the hypothesis which minimizes a sum of the cost function displayed in Figure 2.2(a), while for the same computational price, it could also minimize the cost function in Figure 2.2(b), which yields equal or better results, especially when the inlier threshold τ is set too high [38]. This alteration was dubbed MSAC.



(a) $\rho_1(r)$ (b) $\rho_2(r)$

Figure 2.2: The difference between RANSAC and MSAC lies in the cost function that is minimized. RANSAC effectively minimizes $\sum_{j=1}^{n} \rho_1(r_j)$, while MSAC takes inlier residuals into account by minimizing $\sum_{j=1}^{n} \rho_2(r_j)$. These plots assume $\tau = 1$.

Torr and Zisserman point out that there is no reason to prefer RANSAC over MSAC, as MSAC brings a benefit without adding computational cost. Their MLESAC is a variant that uses maximum likelihood estimation to further improve fitting results [38]. PROSAC [8] generates hypotheses from growing sets of preferred observations. This requires a partial order of observations, such as the similarity between two descriptors in point matching problems. This increases the probability of an early inlier hypothesis being drawn, and is thus a form of *guided sampling*. Another example of guided sampling is NAPSAC [27], which uses a guided sampling technique that selects only one point completely at random. The rest of the points required for a minimal subset are randomly picked from a surrounding hypersphere. The method requires a metric defined on the space of observations, plus a radius for the hypersphere. As points close to an inlier are more likely to be inliers as well, this accelerates the sampling process. A similar technique that also prefers points that lie close to each other is presented in Section 2.5.

2.2.2 RANSAC with multiple structures

If one wishes to find more than one structure, the simplest approach is to run RANSAC sequentially, "peeling" off inliers of the current winning model after each iteration, and re-applying RANSAC to its outliers. This strategy has been used successfully by Torr [37] and Vincent and Laganiere [40], among others. Another RANSAC-based approach designed for multiple model fitting is multiRANSAC , which fits a fixed number of models simultaneously, while dynamically altering the number of iterations by a data-driven bound. The author's experiments show that this simultaneous approach delivers more stable estimations than its sequential counterpart [46].

The disadvantage of these RANSAC-based approaches is that if a point is an inlier to multiple structures, it will be removed at some point, possibly deteriorating the fitting accuracy of a following structure, which in the worst case might not even be identified, as Figure 2.3 illustrates.

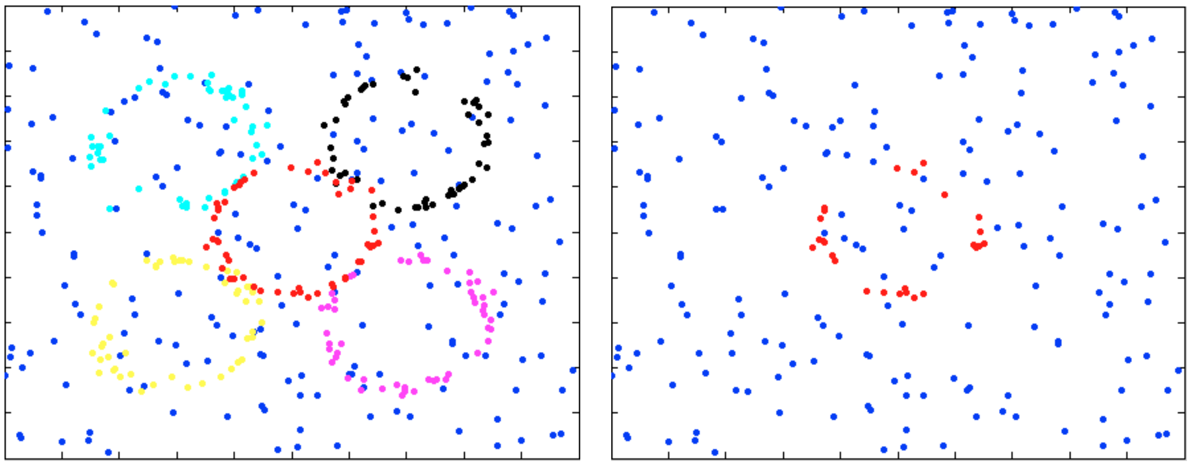
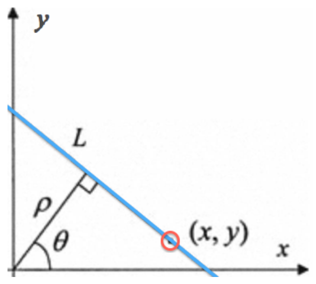

(a) Initial set of points and structures. (b) Points left after 4 iterations.

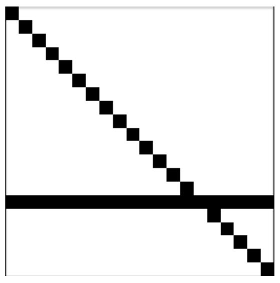
Figure 2.3: Example scenario in which Sequential RANSAC might be unable to identify the innermost circle, because its inliers were removed by previous iterations. Colors symbolize true structure membership.

2.3 The Hough Transform

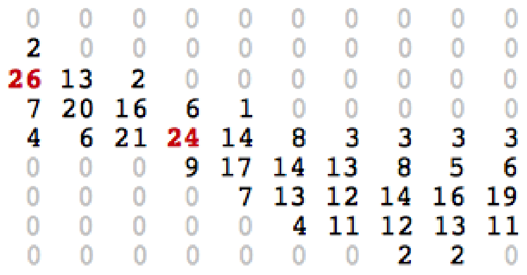
The Hough Transform [17] was originally designed to find lines in a binary image $\mathbf{I} \in \{0,1\}^{n\times n}$. The parameter space of possible lines (ρ, θ), defined by the relationship $\rho = x\cos\theta + y\sin\theta$, is quantisized into a finite number of bins. Each non-white pixel of $\mathbf{I}$ then casts a vote for every line (ρ, θ) that passes through the pixel, creating local minima for strong lines, as depicted in Figure 2.4.



(a) Relationship between line (ρ, θ) and points (x, y).



(b) An example image.



0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
26	13	2	0	0	0	0	0	0	0
7	20	16	6	1	0	0	0	0	0
4	6	21	24	14	8	3	3	3	3
0	0	0	9	17	14	13	8	5	6
0	0	0	0	7	13	12	14	16	19
0	0	0	0	0	4	11	12	13	11
0	0	0	0	0	0	0	2	2	0

(c) Discrete parameter space with accumulated votes for the image in (b). Rows correspond to different values for ρ, columns to values of θ.

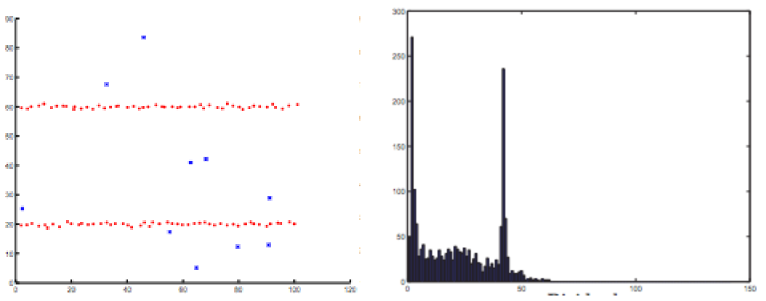
Figure 2.4: The Hough Transform transforms a binary image into an array of votes for a finite set of lines. Each line causes a local maximum in parameter space.

One can easily imagine the Hough Transform as a strategy for multiple line fitting, by picking a fixed number of the strongest local maxima, or by selecting all lines which reach a minimum number of votes. The Generalized Hough Transform [2] adapts the Hough principle to find arbitrary, even non-parametric shapes in images by casting votes for the linear transform of a *template* cast into the image, rather than voting for the shape itself.

The problem of the Hough Transform is that it searches parameter space exhaustively, which for high dimensional models can lead to long computation times and memory overloads. Also, the resolution of the parameter space has to be cautiously defined. The *Randomized Hough Transform* solves these problems by only regarding models generated by minimal subsets, much like RANSAC does, and casting votes into a continuous parameter space, where models with similar parameter vectors are considered equal [43]. Instead of using this quantization, which requires a threshold to determine which models ought to be considered equal, Mean Shift can be used to find the centers of dense clusters in parameter space [9].

2.4 Residual Histogram Analysis

While Sequential RANSAC requires an inlier threshold and a stopping criterion, Hough-based methods break down for very high amounts of noise and outliers. These shortcomings inspired Zhang and Kosecka [45] to design a robust, parameterless algorithm that estimated both the number of structures and their models' parameters. After generating a set of random hypotheses, a histogram of residuals is generated for each point (Figure 2.5). The authors observe that hypotheses generated by points from one and the same structure generate a peak in the histogram of most points. By picking the *median* of the number of histogram peaks over all points, the number of true structures can be estimated.



(a) Dataset (b) Residual histogram for one point

Figure 2.5: Models generated from inliers generate peaks in the residual histograms of individual points [45].

Points with histograms containing a different number of peaks are discarded. One point, $\mathbf{x}^*$, being the point with the strongest peaks, is selected in order to find a representative model for each peak. Since each peak can be composed of hypotheses around the true structure as well as hypotheses that have the same residual magnitude "by accident", the corresponding peak in the histogram of a randomly chosen point $\mathbf{x^j} \neq \mathbf{x}^*$ is identified, and only hypotheses that are present in both peaks are used for the final estimation.

Although RHA does not require a number of models as input, it does depend on the setting of a residual bin size for the histogram, as well as a threshold to distinguish peaks [41].

2.5 Guided sampling

In Section 2.2.1, we mentioned how PROSAC accelerated the process of random sampling by following a partial order defined on the set of observations. Let us now consider two guided sampling methods which prefer points which have a higher probability of being inliers of the same structure, thereby accelerating the process of finding good hypotheses.

2.5.1 Sampling neighboring points

A simple approach of guided selection is used by J-Linkage and was also used by multi-RANSAC [46]. After one point $\mathbf{x}$ has been drawn, the selection prefers neighboring points over more distant points by setting the probability that point $\mathbf{y}$ is drawn after $\mathbf{x}$ to

$$P(\mathbf{y}|\mathbf{x}) \propto \exp\left(-\frac{1}{\sigma^2} d(\mathbf{x},\mathbf{y})^2\right) \quad \text{for } \mathbf{x} \neq \mathbf{y}. \tag{2.5}$$

Because neighboring points are more likely to belong to the same structure as more distant points, this increases the likelihood of obtaining a model with sufficient inliers. This strategy comes with a certain loss of generality: it requires the definition of a distance function between observations.

2.5.2 Accelerated Hypothesis Generation

Another approach to find better hypotheses in less time relies on hypothesis rating after a small amount of m hypotheses have already been generated. For each data point $\mathbf{x}^j$, the absolute residuals $|r_1^j|, |r_2^j|, \ldots, |r_m^j|$ are computed, where $r_i^j = r(\theta_i, \mathbf{x}^j)$. By sorting this vector, and extracting the first h entries, we obtain the top-h hypotheses for each point. With this information, a *pairwise* affinity measure of data points can be established by

$$w(\mathbf{x}, \mathbf{y}) = \frac{1}{h} \left| top_h(\mathbf{x}) \cap top_h(\mathbf{y}) \right| \quad \text{for } \mathbf{x} \neq \mathbf{y}, \tag{2.6}$$

which counts how many top-h hypotheses the points have in common. The probability that a point $\mathbf{y}$ is drawn after $\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^k$ have already been selected is then chosen to be proportional to

$$w_k(\mathbf{y}) = \prod_{j=1}^{k} w(\mathbf{y}, \mathbf{y}^j). \tag{2.7}$$

In summary, data points with similar top-h sets are more likely to belong to the same structure, so when we give them a higher probability of being drawn, we increase the chance of creating a good hypothesis. The top-h lists are updated after each batch of generated hypotheses in order to include all hypotheses so far [5].

2.6 Model Selection

An important aspect of multiple model fitting is the estimation of the number of true structures in the data. For this purpose, criteria from information theory are often considered. Informally speaking, these criteria rely on the principle that if two sets of models explain a dataset equally well, one should always prefer the simpler one.

A formalization of this idea is the Minimum Description Length principle (MDL), where the target is to minimize a cost function

$$C(X, \Theta) = C_{model}(\Theta) + C_{error}(X, \Theta) \tag{2.8}$$

for a dataset X and a model Θ. $C_{model}(\Theta)$ determines the cost of using a model with Θ's complexity, while $C_{error}(X, \Theta)$ encodes the error made by using said model [22].

Model Selection is often used to choose the *type* of model best suited for a single model fitting task. For example, if a set of point correspondences between two images are given, the fundamental matrix cannot be recovered uniquely if the data are degenerate, and the relation between correspondences has to be described by a homography [39]. In the scenario of multiple model fitting with a known model type, Model Selection can be used to select the right *set* of models. For this, simply consider the set of candidate models returned by a given algorithm as a single combined model, which can be simplified by removing or fusing candidates.

Popular MDL-like criteria include Akaike's criterion (AIC, [1]) and the Geometric Robust Information Criterion (GRIC, [39]). The latter cost function is used in the model merging scheme of Kernel Fitting, an approach that will be discussed in Section 2.8. The Label Costs approach, discussed in Section 2.9.3, also employs the principle of punishing model complexity, minimizing an objective function that punishes both bad fits and the introduction of new labels (i.e. models) for the data points.

2.7 Clustering algorithms

The paradigm of random hypothesis generation goes hand in hand with the occurrence of *clusters* in one space or another: if a sufficient amount of hypotheses is generated from the inliers of one and the same structure, they tend to form a cluster in model parameter space [41]. In Section 2.4, we saw that structures also reflect as peaks in residual histograms, i.e. clusters in residual space. Finally, it has been observed that inliers of the same structure themselves form a cluster in certain representational spaces ([36], [4]). This means that the members of the same inlier cluster can be used to reconstruct a model after their cluster has been identified.

Clustering is an extensively studied topic in the field of unsupervised classification. The definition of what a cluster is differs slightly from method to method, but in principle it denotes a subset of the data with a high density, that is high affinities or short distances between the subset's members. The K-Means algorithm [26] is a well known clustering algorithm, but it has some severe limitations: First of all, it requires the number of clusters K as input parameter. Secondly, it assumes hyperspherical or -ellipsoidal shapes for the clusters [20, p. 29]. Thirdly, it assigns *all* data points to a cluster, which makes it unsuitable for robust multiple model fitting as long as no explicit outlier removal is performed. Last but not least, K-Means operates on a continuous, metric *feature space*, which might not be available.

The Mean Shift algorithm [9] improves on three of the above items: It does not require the number of clusters beforehand, and it does not require the clusters to be elliptical, as it "shifts" each observation towards a stationary point of an underlying density function. By identifying these stationary points instead of partitioning the data, Mean Shift can be made robust to outliers [9]. However, it is limited to a continuous feature space just like K-Means.

2.7.1 Hierarchical clustering

For non-continuous feature vectors, hierarchical or agglomerative clustering is a popular clustering method. Instead of working on a continuous feature space, hierarchical clusterers rather work on a symmetric *distance matrix* $\mathbf{D}$, which captures the proximity of each pair of points, such that

$$\mathbf{D}_{ij} = d(\mathbf{x}^i, \mathbf{x}^j), \tag{2.9}$$

where the definition of the distance function $d(\mathbf{x}, \mathbf{y})$ is of no concern to the clustering algorithm. The output of the clustering algorithm is a *hierarchy* of clusters, which at the bottom consists of singleton sets for all observations, and at the top consists of a single set containing all observations. A typical cluster tree can be seen in Figure 2.6. The algorithm proceeds as follows:

1. Place every observation in its own cluster.
2. Merge the two clusters with the lowest cluster distance.
3. Repeat step 2 until the number of clusters is one.

The definition of the cluster distance in step two is relevant to the outcome of the algorithm: *Single linkage* defines it as the shortest distance between two elements of distinct clusters, while *complete linkage* takes the longest distance between elements [36]. Toldo and Fusiello define a linkage technique specialized to the purpose of multiple model fitting.

2.7.2 J-linkage

At the heart of Toldo and Fusiello's approach lies the representation of observations in *conceptual space*: After the generation of randomly sampled hypotheses $\theta_1, \theta_2, \ldots, \theta_H$, residuals between hypotheses and observations are computed and thresholded using a predefined inlier threshold τ, in the same way consensus sets are constructed by RANSAC. This results in a H-by-n binary matrix

$$S_{ij} = \begin{cases} 1 & \text{if } r(\theta_i, \mathbf{x}^j) < \tau \\ 0 & \text{otherwise.} \end{cases} \tag{2.10}$$

The rows of S are the indicator vectors of the hypotheses' consensus sets. Instead of assuming the point of view of the hypotheses, however, the creators of J-linkage look at

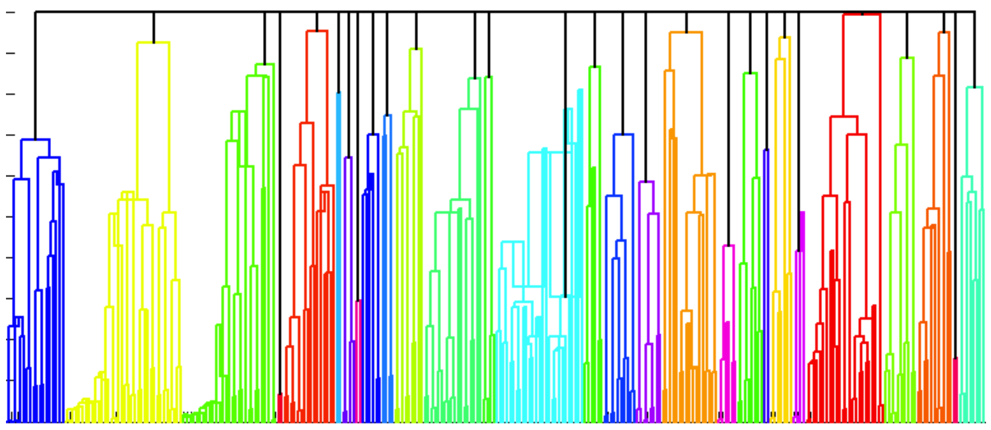


Figure 2.6: Typical output of the J-linkage algorithm. The colors correspond to clusters which do not share common hypotheses. Small clusters are discarded by the algorithm.

the *columns* of S. These indicate the *preference set*, PS_j, for each point $\mathbf{x}^j$, which contains all hypotheses for which $\mathbf{x}^j$ is an inlier.

Toldo and Fusiello now define the distance between two observations as the *Jaccard*-distance of their preference sets:

$$d(\mathbf{x}^i, \mathbf{x}^j) = 1 - \frac{|PS_i \cap PS_j|}{|PS_i \cup PS_j|}. \tag{2.11}$$

The preference set of a *cluster* is defined as the intersection of all contained preference sets. With these two definitions, a cluster hierarchy is constructed as described earlier. The algorithm partitions the cluster tree by seperating clusters which have a distance of (almost) 1, which means that their combined preference sets have no common hypotheses. This means that in the top layer, the points in a cluster have at least one hypothesis in common. After rejecting small clusters, the final model for each top-level cluster is obtained by estimating the model parameters in the least square-manner considering all the points in the cluster.

Toldo and Fusiello showed the effectiveness of J-linkage in experiments with line, circle and plane fitting. In addition, Fouhey et al. successfully employed J-linkage to estimate multiple homographies between image pairs in their 2010 paper on multiple plane matching [15].

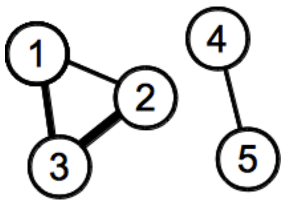
Even though J-linkage does not require an explicit number of structures as an input, the number of gross outliers has to be known in order to reject the correct amount of clusters [4]. A further disadvantage is that each point can only be assigned to a single model, which gives rise to the same problems as the peeling technique of Sequential RANSAC.

2.7.3 Graph clustering

We have seen that hierarchical clustering offers us the possibility to cluster objects without knowledge of the objects' representation, but rather based on given distances between the objects. One can easily see that these distances define a weighted, undirected graph: For any symmetric matrix $\mathbf{M} \in \mathbb{R}^{n \times n}$ there is a graph $G = (V, E, w)$ with vertices $V = \{1, \dots, n\}$, edges $E = \{(i, j) : M_{ij} > 0\}$, and edge weights $w(i, j) = \mathbf{M}_{ij}$. The equivalence of matrix and graph is illustrated in Figure 2.7.

$$\begin{matrix} 0 & 1 & 2 & 0 & 0 \\ 1 & 0 & 3 & 0 & 0 \\ 2 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{matrix}$$

(a) A symmetric matrix.



(b) The corresponding graph.

Figure 2.7: A symmetric matrix defines an undirected graph with edge weights.

This leads to the concept of *graph clustering*, the goal of which is the identification of a dense subgraph of strongly connected vertices. There exist multiple algorithms to identify these subgraphs, also called *clusters*, *dominant sets* or *modes* of a graph, such as Spectral Clustering [23] and Replicator Dynamics [29].

The affinity matrix $\mathbf{M}$

Most graph clustering literature speaks of *affinities* between observations, rather than *distances*, as in Section 2.7.1. This corresponds to the intuition that proximity between nodes should be expressed by higher edge weights in a graph. The two concepts are of course equivalent: The affinity between two observations is simply the inverse of their distance. For example, the distance matrix used by J-linkage can be equivalently expressed by the affinity matrix

$$\mathbf{M}_{ij} = \frac{|PS_i \cap PS_j|}{|PS_i \cup PS_j|}, \tag{2.12}$$

where $|PS_i \cap PS_j| / |PS_i \cup PS_j|$ is the inverse J-distance (see Section 2.7.2). From Figure 2.8, one can see that the affinities between inliers of the same structure are usually higher than their affinities to other data points.

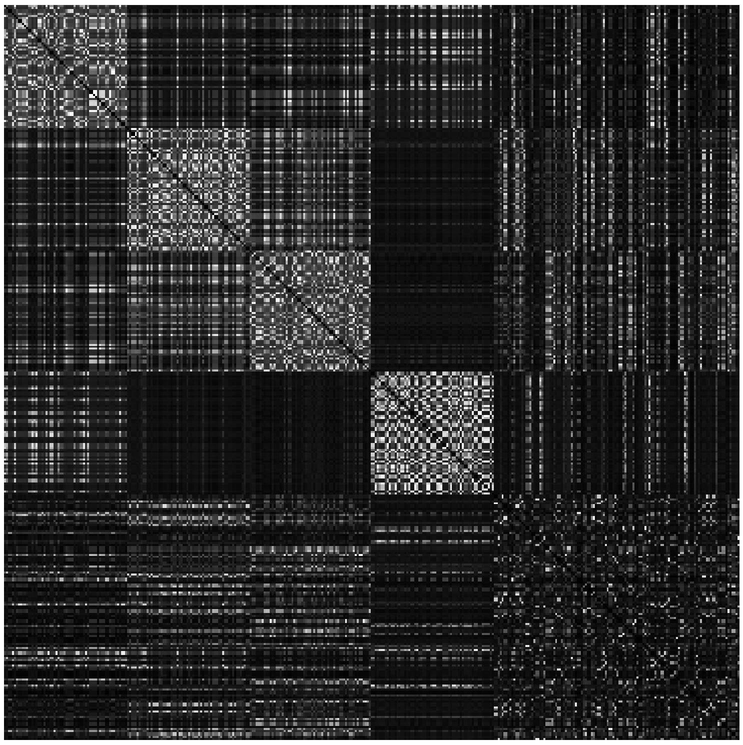


Figure 2.8: An affinity matrix for 300 points (four structures with 50 inliers each, plus 100 gross outliers). High intensities correspond to high affinities. For the viewer's convenience, the rows and columns are ordered by true structure membership. One can see clearly that inliers of the same structure associate strongly, while gross outliers do not have such strong associations.

Defining cluster membership

Before clusters can be identified, we need a method to *define* what a cluster is. Naturally, one could define a cluster as a subgraph, represented by a binary indicator vector $\mathbf{c} \in \{0,1\}^n$, with

$$c_j = \begin{cases} 1 & \text{if vertex } j \text{ is part of the cluster,} \\ 0 & \text{otherwise.} \end{cases} \tag{2.13}$$

However, graph clustering techniques such as Spectral Clustering and Replicator Dynamics allow the entries of $\mathbf{c}$ to take values from the entire interval $[0,1]$, while requiring a total sum of $||\mathbf{c}||_1 = 1$ (i.e., a *stochastic* vector). This, in turn, allows the problem of *finding* a dense subgraph to be formulated as an optimization problem

$$\mathbf{c}^* = \arg\max_{\mathbf{c}} \mathbf{c}^T\mathbf{M}\mathbf{c} \qquad \text{s.t.} \qquad \parallel \mathbf{c} \parallel_1 = 1 \text{ and } c_j \geq 0. \tag{2.14}$$

$\mathbf{c}^T\mathbf{M}\mathbf{c}$ is the sum of all intra-cluster affinities, and is therefore also called the intra-cluster score or *cluster coherency* of $\mathbf{c}$. Each entry c_j of $\mathbf{c}$ can be thought of as the *probability* of vertex j being part of cluster $\mathbf{c}$.

2.7.4 Mode finding with Replicator Dynamics

Replicator Dynamics is a game-theoretic technique which finds a **local** maximum of 2.14 by iterating the *replicator equation*

$$c_j^{(t+1)} \leftarrow c_j^{(t)} \frac{(\mathbf{M}\mathbf{c}^{(t)})_j}{\mathbf{c}^{(t)T}\mathbf{M}\mathbf{c}^{(t)}}, \tag{2.15}$$

over discrete timesteps t, which converges towards a stationary point $\mathbf{c}^*$, which constitutes a valid probablisitc cluster as long as the algorithm is initialized with a stochastic vector. The equation was originally used to model the changes in relative frequency of self-replicating entities [29]. The definitive set of cluster members can be recovered from $\mathbf{c}^*$ simply by selecting indices of $\mathbf{c}^*$ with non-zero entries.

Finding multiple modes

Replicator dynamics has been used by Hairong Liu and Shuicheng Yan to discover common visual patterns of two images, a specific application of multiple model fitting [24]. In order to find multiple modes, this approach runs the replicator dynamics procedure for every node j by deliberately biasing the initialization vector $\mathbf{c}^(0)$ of the equation to the neighborhood $\mathcal{N}(j)$ of j, setting

$$c_i = \begin{cases} \frac{1}{|N(j)|+1} & \text{if } i = j \text{ or } i \in \mathcal{N}(j) \\ 0 & \text{otherwise.} \end{cases} \tag{2.16}$$

Only modes with a sufficient coherency are kept. Replicator dynamics can *drop* vertices from a cluster, but never *accept* new vertices to said cluster, because

$$c_j^{(t)} = 0 \quad \text{implies} \quad c_j^{(t+1)} = 0, \tag{2.17}$$

i.e. when the probabilistic membership of a vertex to a cluster is zero in time step t, it remains zero in all following time steps [25]. Therefore, the success of the algorithm depends on wether the local maximizer is in fact part of the neighborhood of the current node i, which the authors call the *local property*. Note that the affinity matrix in the paper is defined by the differences in distances of point correspondences, not by the more general distance between models and observations.

Graph modes with similar members are merged by the algorithm, which presumes the *non-intersection property*, defined by the authors as the property that structures do not overlap. In scenarios in which these two properties are not fulfilled, the success of the algorithm is doubtful.

The proposed algorithm calls the Replicator Dynamics algorithm for n different initialization vectors. This exhaustive exploration can be justified by the fact that each run only inspects the neighborhood of the current node, which means that most entries of $\mathbf{c}$ are zero from the beginning, which drastically decreases the runtime of Replicator Dynamics because only non-zero entries of $\mathbf{c}$ have to be updated. Thus, the runtime of local graph clustering depends on the sparsity of the graph in question: the smaller the neighborhoods, the faster the clusterer.

2.7.5 Graph Shift

In order to overcome the problem of Replicator Dynamics being "trapped" in local neighborhoods, Liu & Yan devised an algorithm that enables the *expansion* of a graph mode from a subgraph G_S to the entire graph G. Defining

$$a(\mathbf{c}, \mathbf{d}) = \mathbf{c}^T \mathbf{M} \mathbf{d} \quad \text{and} \quad g(\mathbf{c}) = a(\mathbf{c}, \mathbf{c}), \tag{2.18}$$

the authors prove that a mode $\mathbf{c}$ of G_S is also a mode of G if and only if every cluster-vertex affinity is less or equal to the cluster coherency, i.e. $a(\mathbf{c}, \mathbf{j}) \leq g(\mathbf{c})$ for all vertices j, where $\mathbf{j}$ is the indicator vector with all zero entries accept at j. This is illustrated in Figure 2.9.

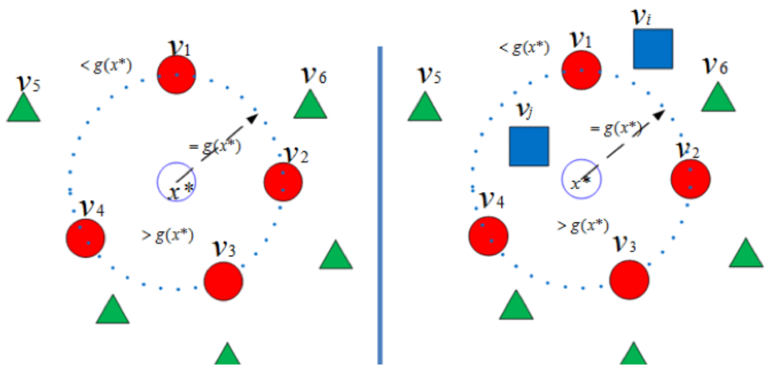


Figure 2.9: Relation between the mode of a subgraph and the mode of a graph. Cluster members (red dots) all have the same affinity with the cluster $\mathbf{c}$, and thus lie on the circle $g(\mathbf{c})$. Other vertices have lower affinities with the cluster and lie "outside" the circle (green triangles). Once the subgraph is expanded to the entire graph, $\mathbf{c}$ remains a mode only if there are no new vertices j with a cluster affinity higher than $g(\mathbf{c})$ (blue dots). Source: [25].

With this knowledge, the authors propose an algorithm consisting of two main steps, Replicator Dynamics and neighborhood expansion. Let $G_{\mathbf{c}}$ be the subgraph of G consisting only of vertices i for which $c_i \neq 0$. Then the algorithm proceeds as follows:

1. Starting from an initialization vector $\mathbf{c}^{(0)}$, it identifies a cluster $\mathbf{c}^{(1)}$ of $G_{\mathbf{c}^{(0)}}$ using Replicator Dynamics as before.

2. If $\mathbf{c}^{(1)}$ is also a mode of G, the algorithm is done (see above).

3. If not, then the found probabilistic cluster $\mathbf{c}^{(1)}$ is updated to $\mathbf{c}^{(1)'} = \mathbf{c}^{(1)} + \mathbf{\Delta c}$, which decreases the entries of nodes already in the cluster, and rewards each neighbor j of $\mathbf{c}^{(1)}$ which has a high affinity $a(\mathbf{c}, \mathbf{j})$.

4. The updated cluster $\mathbf{c}^{(1)'}$ is now used as the initialization vector for Replicator Dynamics, which identifies mode $\mathbf{c}^{(2)}$ of the graph $G_{\mathbf{c}^{(1)'}}$. This process is repeated until $\mathbf{c}^{(t)}$ is a mode of G.

One can see that by continuously expanding the subgraph, the Replicator Dynamics subroutine will identify a mode of G at some point. The update vector $\mathbf{\Delta c}$ is tailored such that vertices outside of $\mathbf{c}$ are rewarded in a way proportional the gain in coherency they would bring to the cluster, and vertices inside $\mathbf{c}$ are punished proportionally to their cluster membership:

$$\Delta c_j \quad \propto \quad \begin{cases} \max\{a(\mathbf{c}, I_j) - g(\mathbf{c}), 0\} & \text{if} \quad c_j = 0, \\ -c_j & \text{else.} \end{cases} \tag{2.19}$$

It seems plausible that this algorithm could perform better than the one from Section 2.7.4 if initialized equally, given the ability to expand modes. The authors compare Graph Shift with the Spectral Clustering technique, claiming higher clustering precision and much shorter runtimes.

2.7.6 Iterative Graph Clustering

In Section 2.7.4, we saw how Replicator Dynamics can be used to find multiple modes by biasing its initialization vector towards a single vertex in each run. Instead of changing replicator dynamics' initialization vector for every vertex in G, which is only feasible when G is sparse, we could however also change the affinity matrix itself after each run: The first cluster $\mathbf{c}^{(1)}$ is identified by applying replicator dynamics to the original affinity matrix $\mathbf{M}$. To find a second cluster, the intra-cluster affinities of the first cluster are diminished by a factor λ before the algorithm is run again, leading to an updated version of $\mathbf{M}$:

$$\tilde{\mathbf{M}} = \mathbf{M} - \lambda(\mathbf{c}^{(1)}\mathbf{c}^{(1)T}) \tag{2.20}$$

As this can lead to negative entries, which are undesirable, we set all negative entries of $\tilde{\mathbf{M}}$ to zero. Finally, we enhance the non-negative updated version $\hat{\mathbf{M}}$ by a scalar multiplication, in order to keep the total energy of the matrix constant:

$$\mathbf{M}^{(2)} = \frac{\sum_{i,j=1}^{n} \mathbf{M}_{ij}}{\sum_{i,j=1}^{n} \hat{\mathbf{M}}_{ij}} \cdot \hat{\mathbf{M}}. \tag{2.21}$$

Then, Replicator Dynamics is run on $\mathbf{M}^{(2)}$, and the new solution $\mathbf{c}^{(2)}$ is injected into the original problem to find the second local maximum. This procedure is repeated until a sufficient amount of modes has been found, or the affinity of the found clusters slips under a fixed threshold. This technique, which we dub Iterative Graph Clustering, thus requires a stopping criterion just like Sequential RANSAC, but it has the advantage of being faster than the local graph clustering approach from Section 2.7.4 if the affinity matrix is not sparse. Iterative Graph Clustering could also be carried out with other mode finders such as Graph Shift or Authority Shift, the clusterer presented below.

2.7.7 Authority Shift

Introduced by Cho & Lee in 2010, this relatively new approach is a graph clusterer based on Google's *PageRank* algorithm [7]. This algorithm follows a random walker on a large graph, going from vertex i to vertex j with a certain probability p_{ij}, and sometimes jumping to a random vertex. It thus requires a *transition matrix* $\mathbf{P}$, containing the probabilities of transitions p_{ij}, and a parameter α regulating the number of random jumps. Consistent with the theory of Markov chaings, this random walker has a *steady state* vector $\mathbf{r}$, satisfying

$$\mathbf{r}^T = \alpha \mathbf{r}^T \mathbf{P} + (1-\alpha)\mathbf{v}^T, \tag{2.22}$$

where the entries of $\mathbf{v}$ creates a bias of the steady state towards certain nodes, which is why $\mathbf{v}$ is called the *personalization vector*. The personalized PageRank PPR is defined as

$$PPR(\mathbf{v}) = \mathbf{r}^T \mathbf{P} \quad \text{such that} \quad \mathbf{r}^T = \alpha \mathbf{r}^T \mathbf{P} + (1-\alpha)\mathbf{v}^T, \tag{2.23}$$

If $\mathbf{v}$ only has one non-zero entry $v_i = 1$, then we write $PPR(i)$ for the PageRank vector personalized to a single node, and $PPR(i,j)$ for its j-th entry, which represents the importance of node j for node i. Cho & Lee define higher order PageRanks for single nodes by

$$PPR_n(i) = PPR(PPR_{n-1}(i)), \text{ with } PPR_1(i) = PPR(i). \tag{2.24}$$

The relationship between this random walker modeled on the World Wide Web and the problem of graph clustering can be explained as follows: First of all, the affinity matrix $\mathbf{M}$ defining a graph can be transformed into a transition matrix $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_n]^T$ simply by making its rows $\mathbf{m}_1^T, \mathbf{m}_2^T, \ldots, \mathbf{m}_n^T$ stochastic, i.e.

$$\mathbf{p}_i^T = \frac{\mathbf{m}_i^T}{\| \mathbf{m}_i \|_1}. \tag{2.25}$$

Second of all, we can shift every vertex towards the most "important" node in its (higher order) personalization vector, its so called authority, which we will denote by

$$Auth_n(i) = \arg\max_j PPR_n(i,j). \tag{2.26}$$

For each node i, we look for $a = Auth_n(i)$, then for $Auth_n(a)$, etc., until we pass a node a second time, which means that we have reached a circle, called the *authority sink*. An authority sink, together with all the nodes for which this node traversal ends in said sink, represents a cluster. We can repeat this clustering for increasing order of n, until all vertices end up in the same cluster. Thereby, we have constructed a hierarchy of clusters, as shown in Figure 2.10.

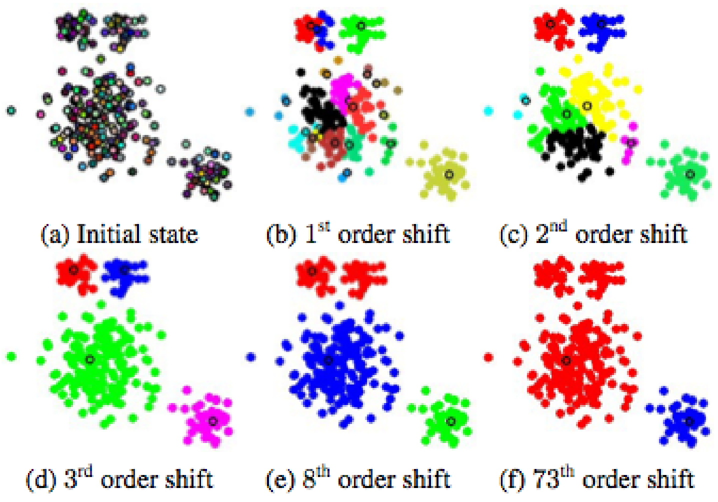


Figure 2.10: An example of the hierarchical output of Authority Shift. Higher order PageRanks lead to coarser clusters. Source: [7].

To make the algorithm more efficient and stable, the authors introduce supernodes in the hierarchical construction. While clustering the data for increasing order k, clusters are handled as supernodes as soon as the increase of k changes the clustering results.

Properties of Authority Shift

Since the PageRank function was developed for the World Wide Web, which is a very large graph, the values for PPR_n can be computed with high efficiency. The disadvantage of Authority Shift is that it assigns *all* data points to a cluster, thereby not inherently removing outliers. This problem was tackled by the authors in their 2012 paper, “Mode-seeking on graphs via random walks” [6], where they propose an adaption of Authority Shift which produces dense subgraphs instead of clusters, thereby making the algorithm applicable to outlier-ridden data. Unfortunately, no source code of this approach was available at the time this thesis was written, which is why we tested and evaluated the regular variant of Authority Shift with an explicit outlier removal step, as it will be explained in Chapters 3 and 4.

2.8 Kernel Fitting

Let us now consider an eleborate multi-model fitting pipeline by Chin et al. [4]. Similarly to previously discussed approaches, it handles multiple model fitting as a problem of *clustering data points*. The main idea of their approach is the employment of approved statistical learning methods such as Principal Component Analysis to facilitate the clustering problem. It also employs explicit outlier removal and information-theoretic model selection.

The Ordered Residual Kernel

As with previous approaches, the basis of Kernel Fitting is a set of hypotheses $\theta_1, \theta_2, \ldots, \theta_H$, generated by randomly sampled data points (see Section 2.2). For each data point $\mathbf{x}^j$, this results in residuals

$$r_i^j = r(\theta_i, \mathbf{x}^j) \quad \text{for} \quad i = 1, \ldots, H. \tag{2.27}$$

Based on these residuals, the authors design an affinity measure (or *kernel*) between data points that (a) does not require the definition of an inlier threshold and (b) allows the application of statistical learning methods. For point (a), they rely on residual ordering. Recall the affinity measure based on top-h lists of best fitting hypotheses from Section 2.5.2, where data points are compared by the number of hypotheses shared in their top-h lists. In order to capture not just the similarity of two points' top-h lists, but also the fraction of hypotheses that two points have in common in other parts of the ordered hypothesis list, we can define the *difference of intersections* kernel (DOIK):

$$k^t(\mathbf{x}, \mathbf{y}) = \frac{1}{h}\left(|top_{th}(\mathbf{x}) \cap top_{th}(\mathbf{y})| - |top_{(t-1)h-}(\mathbf{x}) \cap top_{(t-1)h}(\mathbf{y})|\right), \tag{2.28}$$

which is a measure of common hypotheses in the t-th part of the ordered hypothesis lists. The combination of all DOIKs $k^1(\cdot,\cdot), \ldots, k^{H/h}(\cdot,\cdot)$ allows us to compare observations not just by their top-h lists, but by less important windows of their ordered hypotheses lists as well. Naturally, windows at the front of the list should weigh highest in the comparison, while the weights of the DOIKs should decrease towards the end of the list. In other words, the weight of a window should be inversely proportional to the position of the window. This is accomplished by the Ordered Residual Kernel

$$k(\mathbf{x}, \mathbf{y}) = \frac{1}{Z} \sum_{t=1}^{H/h} \frac{1}{t} k^t(\mathbf{x}, \mathbf{y}), \tag{2.29}$$

where $Z = \sum_{t=1}^{H/h} \frac{1}{t}$. Chin et al. prove that $k(\cdot,\cdot)$ is a valid *Mercer* kernel, i.e. a function that allows the application of the *kernel trick*: the kernel implicitly defines a transformation ϕ by

$$\langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle = k(\mathbf{x}, \mathbf{y}), \tag{2.30}$$

to a *feature space* where dimensionality reduction, outlier removal and clustering can be performed by powerful and mathematically sound machine learning techniques. The advantage of this mapping is illustrated in Figure 2.11.

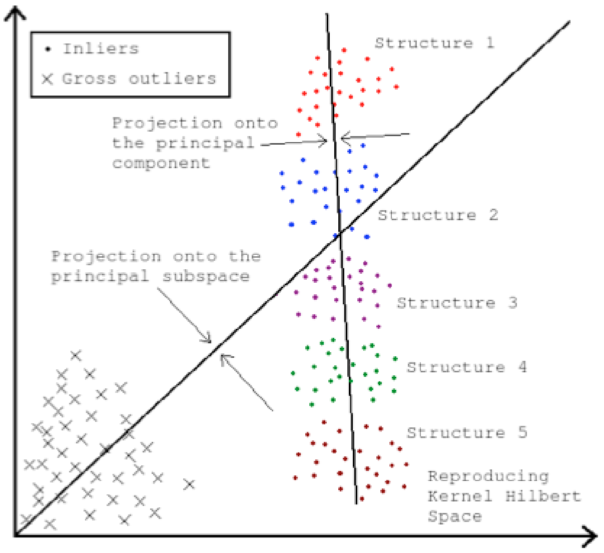

• Inliers
× Gross outliers
Projection onto the principal component
Structure 1
Structure 2
Structure 3
Structure 4
Structure 5
Projection onto the principal subspace
Reproducing Kernel Hilbert Space

Figure 2.11: The mapping of data points into feature space facilitates the removal of outliers (black crosses close to origin) and the clustering of inliers (colored dots). Source: [4].

Outlier removal and structure discovery in feature space

Gross Outlier Removal

The transformed data $\mathbf{A} = [\phi(\mathbf{x}^1) \dots \phi(\mathbf{x}^n)]$ are projected to their principal subspace by computing the eigenvalue decomposition of $\mathbf{A}^T\mathbf{A}$, yielding projections $[\mathbf{b}^1 \dots \mathbf{b}^n]$. Note that $\mathbf{B}$ can be computed even though $\mathbf{A}$ is never explicitly given. The authors point out that observations $\mathbf{x}^j$ with high $\| \mathbf{b}^j \|_2$ are more probable to be outliers. This means we can separate them from projections with low norms (corresponding to inliers) using a Gaussian mixture model.

Structure discovery

In preparation of structure discovery, the remaining data after outlier removal are projected to their principal components using *Kernel PCA* [32, p. 151] in order to reduce dimension and facilitate the clustering problem. As in many other approaches discussed so far, the data are assumed to cluster around the underlying structures due to the nature of $k_{\tilde{r}}(\cdot, \cdot)$, which gives higher values to point pairs that emerge from the same structure. The data are clustered using *spectral clustering* [33] on an affinity measure defined by of the projected data.

Model merging scheme

Initially, the k clusters resulting from structure recovery are transformed into models by LMedS [30] estimation, and together they form the set of models $\mathcal{M}_k$. In order to reduce the number of models, a pair of model is merged if the resulting model can still "explain" the data. For each possible pair of models (θ_1, θ_2) in $\mathcal{M}_k$, a new set of models is created by combining the inliers of θ_1 and θ_2 and reestimating a new model θ from their inliers. This setp results in $\binom{k}{2}$ candidate sets of models. The candidate set with the lowest cost $\phi(\mathcal{M})$ becomes the next layer in the hierarchy, $\mathcal{M}_{k-1}$. This process is repeated until the final layer $\mathcal{M}_1$, has been constructed, containing a single model. Finally, the layer l^* which best explains the data, i.e. $l^* = \arg\max_l \phi\left(\mathcal{M}_l\right)$, is chosen as the final set of models.

The cost function was taken from [39], and is defined as

$$\varphi(\mathcal{M}) = -2L + \beta\big(n \log(d) + p|\mathcal{M}| \log(dn)\big) \tag{2.31}$$

where n is the total number of inliers, p and d are model and data dimensions, and

$$L = \sum_{i=1}^{|\mathcal{M}|} \left[n_i \log\left(\frac{1}{\sqrt{2\pi}\sigma_i}\right) - \sum_{j=1}^{n_i} \frac{r_{ij}^2}{2\sigma_i^2} \right] \tag{2.32}$$

with n_i inlier residuals $r_{i1}, r_{i2}, \ldots, r_{in_i}$ for each model, and their standard deviation σ_i.

The first term in Equation 2.31 expresses the cost of making fitting errors, while the second term punishes the total model complexity.

Results

Chin et al. successfully apply their model fitting pipeline to line fitting, homography estimation and motion segmentation. For the first application, their approach beats Randomized Hough Transform, Residual Histogram Analysis and J-linkage by far in terms of the MSFE, an error measure which will be defined in Chapter 4.

2.9 Other Approaches

In addition to the strategies in Section 2.7, which all cluster data points in one representation or another, we will discuss three other strategies, each of which can be considered state-of-the-art.

2.9.1 Dynamic and Hierarchical Multi-Structure Fitting

The authors of Accelerated Hypothesis Generation further develop the idea of top-h lists into a full framework for multiple model fitting [42]. The workflow of the approach is shown in Figure 2.12 and will be explained in detail further below.

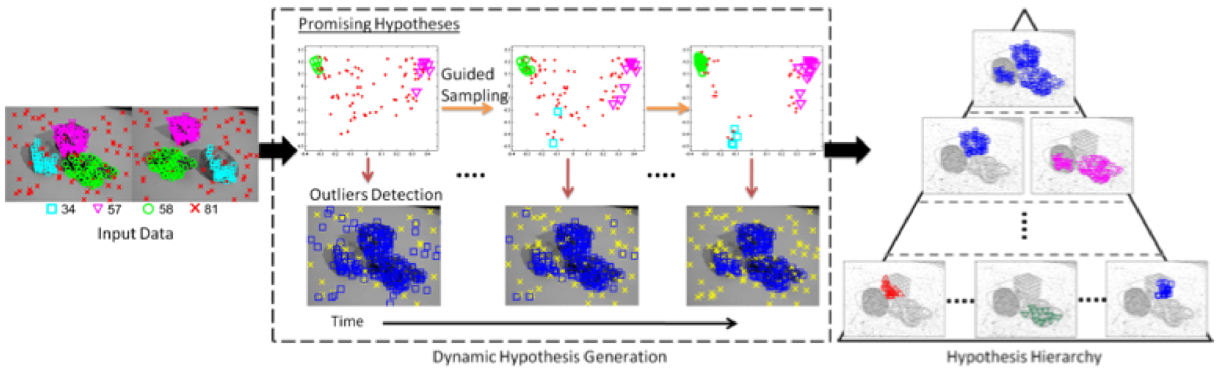


Figure 2.12: Dynamic and Hierarchical Multi-Structure Fitting consists of two steps: guided hypothesis generation and the construction of a hypothesis hierarchy. Outliers are removed along the way. Image source: [42].

First of all, top-k lists are introduced for each generated hypothesis, which contain the k best fitting data points of the model. A distance measure for hypotheses θ_i, $\theta_{i'}$ is defined based on the *order* of data points in their top-k lists. For example, if a data point $\mathbf{x}^j$ is the point best fitted by θ_i, and the fourth-best for $\theta_{i'}$, then $\mathbf{x}^j$ adds a value proportional to $4 - 1 = 3$ to the distance $d(\theta_i, \theta_{i'})$.

Based on this distance measure, a quality measure for hypotheses *relative* to other hypotheses in a set Θ is given by

$$w_i^{\Theta} = \sum_{\theta \in \Theta} \exp\left(-\frac{1}{2\delta^2} d(\theta_i, \theta)^2\right). \tag{2.33}$$

Hypothesis Filtering & Guided Sampling

After a batch of b hypotheses has been sampled randomly, the best hypothesis for each data point $\mathbf{x^j}$ is determined by $e_j = \arg\max w_i^{top_h(\mathbf{x_j})}$. Only these *exemplars* are kept — all other sampled models are discarded. Now, the p-subset necessary for the creation of the next candidate model is sampled from the top-k list of the *least* promising exemplar e_j. This choice has two effects:

- If $\mathbf{x}^j$ is an outlier, the top-k list of its exemplar is probably fairly random, which leads to the *exploration* of new structures (early stage).
- If $\mathbf{x}^j$ is an inlier to a structure, the top-k list probably contains other points from the same structure, leading to *exploitation* of said structure (later stage).

Gross Outlier Removal

In order to filter out gross outliers, at each time step t, i.e. after the generation of a batch, each data point $\mathbf{x^j}$ is assigned a value s_j that averages the residuals $r(\mathbf{x^j}, e_j^k)$ over time steps $k = 1, \ldots, t$. Points with large s_j (outliers) are separated from points with small s_j (inliers) by a Gaussian mixture model.

Hypothesis Hierarchy

After a set of good hypotheses has been sampled iteratively as described above, a hierarchy is created. Neighboring models are merged if they share more than 50% of their top-k data, resulting in hypothesis clusters. The hypothesis with the smallest sum of squared residuals is chosen as the representative of the cluster in the next hierarchy level. This procedure results in a multi-scale collection of models, with a more detailed description of the data at the lower level, and coarser models at the top.

2.9.2 AKSWH

In their paper [41] from 2011, Wang et al. (also the authors of Kernel Fitting) propose a full multiple model fitting framework which does not require any manual configuration of parameters. The inlier/outlier threshold necessary for RANSAC is replaced by scale estimation, other parameters are estimated using information-theoretic approaches.

The algorithm starts with a set of generated hypotheses, which can either have been sampled at random or in a guided manner. For each hypothesis θ_i, an ordered list of absolute residuals

$$|\tilde{r}_i^1|, |\tilde{r}_i^2|, \dots, |\tilde{r}_i^n| \quad \text{with} \quad |\tilde{r}_i^1| \leq |\tilde{r}_i^2| \leq \cdots \leq |\tilde{r}_i^n| \tag{2.34}$$

is computed. Note that this list orders the residuals of *points* per *hypothesis*, contrary to Section 2.8, where residuals were sorted per point. An iterative scale estimator proposed by the authors, depending on the K-th ordered residual[1] $\tilde{r}_K^j$, gives each hypothesis an inlier scale. The *weight* of each hypothesis θ_i is determined by

$$w_i = \frac{1}{n} \sum_{j=1}^{n} \frac{KN\left(r(\theta_\mathbf{i}, \mathbf{x^j})/h_i\right)}{s_i h_i}, \tag{2.35}$$

where n is the number of data points, s_i is the previously determined scale for the hypothesis, $KN(\cdot)$ is the Epanechnikov kernel

$$KN(r) = \begin{cases} \frac{3}{4}(1 - r^2) & \text{if } |r| \leq 1 \\ 0 & \text{otherwise,} \end{cases} \tag{2.36}$$

and h_i is an estimated bandwidth depending on both residuals and scale. Note that this weight is not much more than a count of inliers within a data-driven bandwidth around a hypothesis, much like the inlier count used by RANSAC.

Hypothesis Filtering

The authors try to reduce the number of relevant hypotheses by rejecting hypotheses which contain little information. They define a probability measure

$$p_i \propto w_{\max} - w_i^2, \tag{2.37}$$

from which the total entropy over all models is computed. Hypotheses that contribute little to this entropy are rejected.

[1] K is set to 10% of data, $\lfloor \frac{n}{10} \rfloor$, in the paper.

Clustering Hypotheses

Just like data points cluster around structure models in conceptual space, hypotheses cluster around structures in parameter space. To identify these clusters, the J-linkage algorithm from Section 2.7.2 is used. Thresholding is again done using the data-driven entropy approach described above. A major difference (and advantage) to classic J-linkage is that hypotheses, not data points, are clustered. The number of filtered hypotheses is much smaller than the total number of data points, which means this form of J-linkage is much more efficient than the original.

Each cluster is represented by its most significant hypothesis. A problem that occurs is that multiple clusters might develop on one and the same structure (e.g. clusters around two similar lines when the inlier noise is high). Therefore, the authors propose a fusing step. The fusing of clusters (i.e. representative hypotheses) is also based on information theory: the pairwise *mutual information* is computed for each hypothesis pair. Pairs with mutual information > 0 are fused.

Summary

Hypotheses are sampled, then weighted using a data-driven scale estimator, filtered based on their information content, clustered by J-linkage and then fused. The remaining hypotheses are the output of the algorithm. The only parameter set by hand is the value K for the scale estimator.

2.9.3 Label Costs

Approximate Energy Minimization with Label Costs [10] forumlates multiple model fitting as a labeling problem with data costs, a pairwise cost function, and label costs. The idea is to minimize an energy

$$E = \sum_{i=1}^{n} D(f(\mathbf{x}^i)) + \sum_{i,j=1}^{n} V(\mathbf{x}^i, \mathbf{x}^j) + \sum_{l \in L} c_l \cdot \delta_l(f), \tag{2.38}$$

where f denotes a labeling, $D(f(\mathbf{x}))$ denotes the cost of labeling data point $\mathbf{x}$ with label $f(\mathbf{x})$, L is the set of all possible labels, c_l is the cost of introducing label l, and $\delta_l(f)$ is zero or one depending on wether an $\mathbf{x}$ exists in the data such that $f(\mathbf{x}) = l$. $V(\mathbf{x}^i, \mathbf{x}^j)$ is a *pairwise* cost function, which might incorporate a priori knowledge about the relationship between observations. In many applications, there is a spatial relationship between observations, and points that lie closer together are more likely to have the same label. Since this function rewards regions of coherent observations, $V(\mathbf{x}^i, \mathbf{x}^j)$ is also referred to as the *smoothness* term.

If $V(\mathbf{x}^i, \mathbf{x}^j) \equiv 0$, this problem is known as the *uncapacitated facility problem*, which is NP-hard. A greedy heuristic solution for this special case was proposed by Kuehn & Hamburger in 1963 [21].

If $c_l \equiv 0$, the α-expansion algorithm [3] offers a fast heuristic. Delong et al. generalized this heuristic to handle label costs besides unary and pairwise costs [10], and incorporated it into the PEARL algorithm:

An affinity matrix $\mathbf{M}$ is created from observations and initial hypotheses as in previous sections. Then, the energy in Equation 2.38 is optimized using α-expansion, yielding a labeling $\mathcal{L}_1$. After removing the outlier cluster, a new set of hypotheses is estimated from the labeled inlier sets. This procedure is repeated until the labeling is stable, i.e. no observations change their labeling.

A typical choice for $V(\mathbf{x}^i, \mathbf{x}^j)$ is the Potts model [10],

$$V(\mathbf{x}^i, \mathbf{x}^j) = \begin{cases} 0 & \text{if } i = j \\ w_{ij} & \text{otherwise.} \end{cases} \tag{2.39}$$

Besides PEARL, the C++ library made public by the authors also incorporates Kuehn & Hamburger's solution for $V(\mathbf{x}^i, \mathbf{x}^j) \equiv 0$, which yields results up to 30 times faster [10].

2.10 Summary

In this chapter, we have discussed a series of algorithms designed for or related to multiple model fitting. Classic Hough-based methods exhaustively scan the parameter space from which models are chosen. Other algorithms rely on randomly sampling minimal subsets of data points, from which candidate models (so called hypotheses) can be constructed. Many approaches require a manually set inlier threshold, which determines the granularity of the fitting problem. Some also require other parameters such as the number of true structures, a threshold for a stopping criterion, or algorithm-specific regulating terms such as the rate of random jumps in Authority Shift. There are algorithms that claim to be parameterless, such as Residual Histogram Analysis, which scans the residual histogram of each point for peaks. In practise, even the most high end model fitting pipelines require some form of configuration: Kernel Fitting has a window size h, and AKSWH requires a residual index K in order to perform scale selection. Generally speaking, we can conclude that many approaches can be broken down into components that are shared among other approaches, a property we will make use of in the next chapter to construct a multiple model fitting framework from exchangeable building blocks.

3 Model fitting framework

One of the goals of this thesis was the comparison of the approaches described in Chapter 2. For this purpose, we constructed a general framework incorporating the algorithms that we discussed previously. Some components were implemented by hand, others were available as parts of existing model fitting pipelines for which source code was available online. These were wrapped in order to fit the framework's demands. The overall requirements to our framework were the following:

- Provide a common interface to different sources of data, either from available datasets or from generators of synthetic data.
- Provide model specific functions for model construction, residual functions, etc.
- Provide core functionality independent from the model class, such as inlier/outlier dichotomy, fitting strategies and graph clusterers.
- Provide general strategies that mold the above components into powerful model fitting pipelines.

All these requirements demand a highly modular design of exchangeable building blocks, as we will see in Section 3.1.

3.1 Design of the framework

We intended to create a highly generic and flexible model fitting framework for Matlab, which can be internally configured to run any existing model fitting algorithm (or a recombination of existing approaches), while operating as a "black box" to the outside, taking observations as input and delivering discovered models as its output. Its flexibility applies to the different algorithms, but also to the class of input data, i.e. its dimension (2D, 3D) and type (lines, circles, planes, homographies, ...). The design of the framework is shown in the chart of Figure 3.1. The most high level function that can be defined in the configuration is the *strategy*: This function acts as a controller that coordinates the flow of the used components. It defines the order in which subroutines are called, without explicitly naming these subroutines. For example, the J-linkage approach described in Section 2.7.2 and the Iterative Graph Clustering technique from Section 2.7.6 are both variants of a strategy we named *Data Affinity Clustering*: their difference lies in the affinity kernel (J-distance vs. Ordered Residual Kernel), and in the clusterer (J-linkage vs. graph clustering) applied to these affinities.

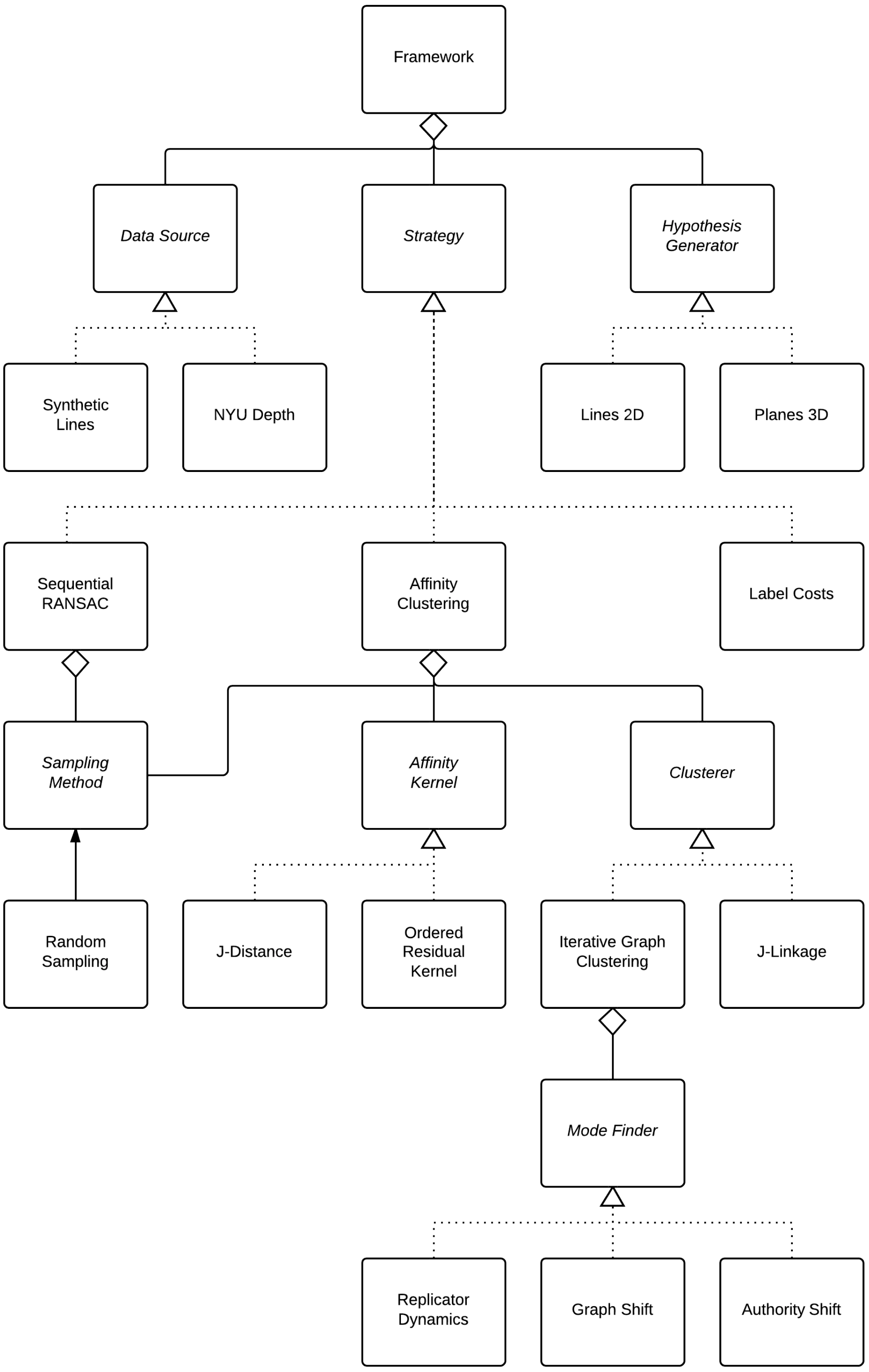


Figure 3.1: Partial diagram of the framework design. For clarity, not all implemented building blocks are shown. Arrows with triangular heads represent implementations (e.g. *Sequential RANSAC* **is a** *Strategy*), while diamond-shaped arrow heads signify a composition (e.g. *Sequential RANSAC* **has a** *Sampling Method*).

3.2 Methods

After having categorized a broad spectrum of multiple model fitting methods in Chapter 2, we will now discuss the different combinations of components that form the setups evaluated in Chapter 4. We will also provide a nomenclature for the setups. We divide the used methods into strategies, the top level building blocks of our framework. Our main strategy, Data Affinity Clustering, is further divided by the methods used to identify clusters.

3.2.1 Sequential RANSAC

The simplest of our strategies, Sequential RANSAC consists of two steps: calling RANSAC to identify the most prominent structure, and removing its inliers from the set of observations. These steps are repeated until a certain stopping criterion is met, which can be:

- Stop after a predefined number of iterations (denominated as **SRa**).
- Stop if the number of inliers is lower than a certain threshold (setup **SRb**).
- Stop if the information-theoretic cost of the previous set of models is lower than the cost of the current set (setup **SRc**).

Both the main loop of the strategy and the RANSAC algorithm itself were implemented by the present author. The different setups of Sequential RANSAC are visualized in Figure 3.2. All setups use the least squares method for the final model estimation.

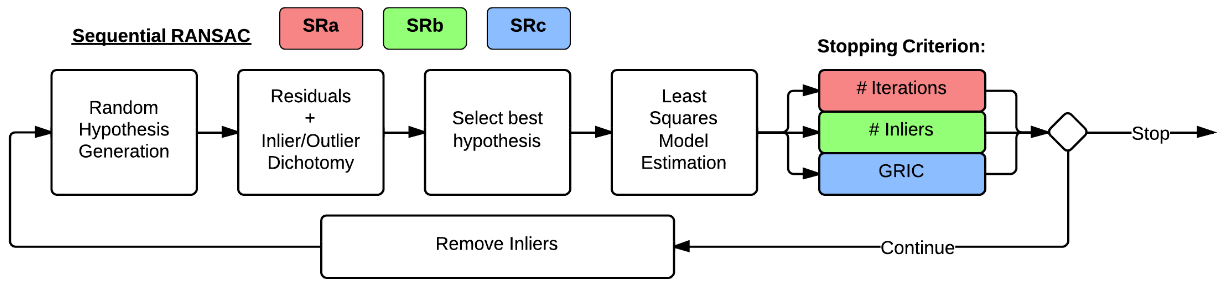


Figure 3.2: The three variants of Sequential RANSAC for our evaluation.

For **SRc**, we used the same cost function as in Section 2.8.

3.2.2 Data Affinity Clustering

This strategy encompasses a large category of algorithms, each of which clusters data points, or rather their similarities, in terms of matching hypotheses encoded in an affinity matrix. Its flow consists of the following steps:

1. Generate hypotheses from randomly sampled subsets.
2. Perform inlier/outlier dichotomy.
3. Compute affinities between data points.
4. optional remove gross outliers.
5. Identify multiple clusters using the resulting affinity matrix.
6. Estimate models from cluster members.
7. optional fuse similar models.

The following subsections will handle the different setups under this strategy, organized by the type of clustering that are employed.

3.2.3 J-linkage

Relying on the J-distance kernel, this clusterer is the one described in Section 2.7.2. For the computation of the J-distance kernel and the J-linkage algorithm itself we used the original code published by the authors online[1]. The J-linkage pipeline is shown in Figure 3.3. The algorithm depends on an inlier threshold for the inlier/outlier dichotomy, and, in order to choose the right amount of clusters from its cluster hierarchy, a minimal cluster size (**JLb**) or the number of total models (**JLa**). The final set of models may be merged by Kernel Fitting's cost-based model selection step (IT model selection in the figure, **JLc**). A major difference between our usage of J-linkage and the one proposed by the authors is the sampling process, which is guided in the original paper, but for which we used random sampling as it is the standard hypothesis generator in our model fitting framework.

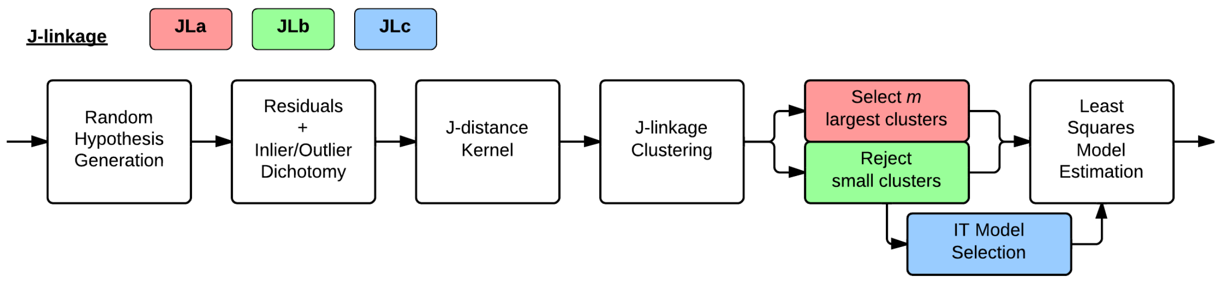

J-linkage
JLa
JLb
JLc
Random Hypothesis Generation
Residuals + Inlier/Outlier Dichotomy
J-distance Kernel
J-linkage Clustering
Select *m* largest clusters
Reject small clusters
Least Squares Model Estimation
IT Model Selection

Figure 3.3: Three variants of the J-linkage pipeline.

3.2.4 Stand-alone Graph Shift

The Graph Shift algorithm was designed as an enhancement to the local graph clustering approach discussed in Section 2.7.4. Therefore, its source code[2] can be configured to be

[1] www.toldo.info/roberto/files/JLinkage.rar
[2] https://sites.google.com/site/lhrbss/

either a stand-alone clusterer, seeking modes in local neighborhoods, or a single-run mode finder, which will be discussed in Section 3.2.6. For the stand-alone variant, the code allows the caller to freely define from which nodes the Graph Shift algorithm should evolve: In the simplest case, we evolve from each single node and initialize it with a membership value of 1 (see Section 2.7.5). It is also possible to define larger subgraphs with non-uniform cluster memberships. However, we discarded this option in an early stage because it did not improve results.

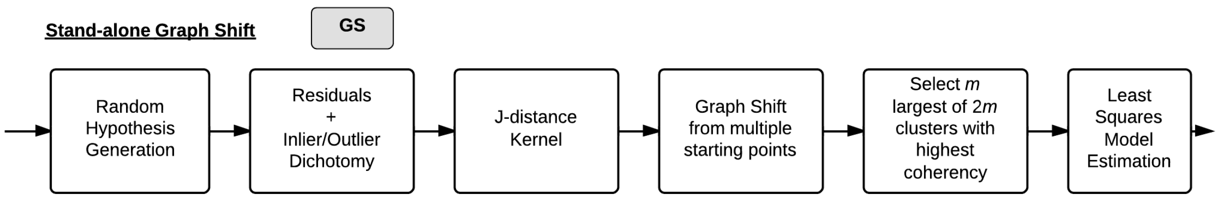

Stand-alone Graph Shift
GS
Random Hypothesis Generation
Residuals + Inlier/Outlier Dichotomy
J-distance Kernel
Graph Shift from multiple starting points
Select *m* largest of 2*m* clusters with highest coherency
Least Squares Model Estimation

Figure 3.4: Graph Shift as a stand-alone model fitting setup.

As we did not consider stand-alone Graph Shift for the evaluation of cardinality estimation, Figure 3.4 includes only one setup, **GS**, which creates a fixed number of models from the largest clusters returned by Graph Shift (see Section 4.3).

3.2.5 Stand-alone Authority Shift with Outlier Removal

Just like Graph Shift, The Authority Shift algorithm described in Section 2.7.7 is in itself a clusterer which provides a label for each observation. Hence, it seems sensible to apply it directly, without iteratively looking for a single mode as we do in Section 3.2.6. As discussed earlier, the variant of Authority Shift which seeks dense subgraphs, and thereby implicitly filters out gross outliers, had no source code available at the time of this writing. Nevertheless, we were interested in Authority Shift's performance, so we used the source code[3] for the Authority Shift partitioning clusterer, and combined it with the explicit gross outlier removal module from Section 2.8.

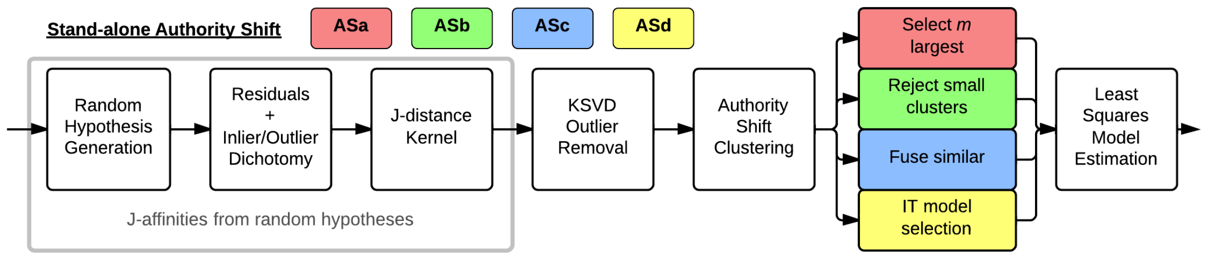

Stand-alone Authority Shift
ASa
ASb
ASc
ASd
Random Hypothesis Generation
Residuals + Inlier/Outlier Dichotomy
J-distance Kernel
J-affinities from random hypotheses
KSVD Outlier Removal
Authority Shift Clustering
Select *m* largest
Reject small clusters
Fuse similar
IT model selection
Least Squares Model Estimation

Figure 3.5: Four variants of stand-alone Authority Shift, focussing on the number of clusters, i.e. models, that are returned. The first three steps will be shown as a single component in further figures.

[3] http://cv.snu.ac.kr/research/~authorityshift/

The number of models produced by the Authority Shift pipeline was controlled in four ways, leading to four different setups: **ASa** simply selects a fixed number of largest clusters, **ASb** rejects small clusters, **ASc** fuses similar clusters based on their parametric representation, and **ASd** uses cost-based model selection. Authority Shift was also employed with the Ordered Residual Kernel, resulting in the setup **ASO** shown in Figure 3.6.

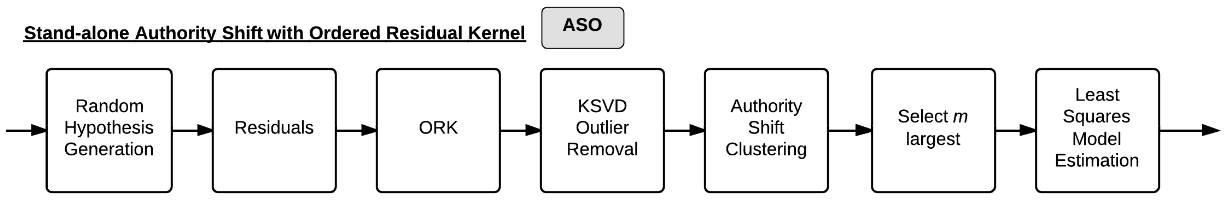

Stand-alone Authority Shift with Ordered Residual Kernel
ASO
Random Hypothesis Generation
Residuals
ORK
KSVD Outlier Removal
Authority Shift Clustering
Select *m* largest
Least Squares Model Estimation

Figure 3.6: Stand-alone Authority Shift with ORK affinities.

3.2.6 Iterative Graph Clustering

Here, the iterative clustering approach described in Section 3.2.6 was embedded in the general model fitting strategy of random sampling, affinity computation, clustering and model estimation. After finding an initial mode (i.e. cluster), the affinity matrix $\mathbf{M}$ is updated as proposed.

Labeling Although strictly speaking, the labeling of data points is not part of the model fitting problem (models are reconstructed from the members of a cluster, regardless of multiple cluster membership), the assignment of a data point to a single model is very helpful for visualization purposes. In order to convert the set of probabilistic cluster memberships $\mathbf{c}^1, \dots, \mathbf{c}^m$ returned by Iterative Graph Clustering to a labeling where each observation gets exactly one label, we select the maximal membership for each observation:

$$L(\mathbf{x}^j) = \arg\max_i c^i_j. \tag{3.1}$$

For the actual identification of modes in the affinity graph, different algorithms were used:

Iterative Replicator Dynamics

The code for the Replicator Dynamics algorithm originates from the ICG repository and is compiled to a 64 bit .mex file. We used a regularization term λ in order to counter-weigh self-affinities of observations, changing the input matrix $\mathbf{M}$ to

$$\mathbf{M}_\lambda = \mathbf{M} - \lambda\mathbf{I}. \tag{3.2}$$

We ran the the iterative procedure for a fixed number of iterations, from which either a fixed number of clusters is selected based on their size (**iRDa**), *or* only clusters with sufficient size and cluster coherency are selected, after which a fusing step fuses similar models, by simple comparison (**iRDb**) or cost-based model selection (**iRDc**). These last two setups were also repeated with an explicit outlier removal step, creating setups **iRDd** and **iRDe**, as shown in Figure 3.7.

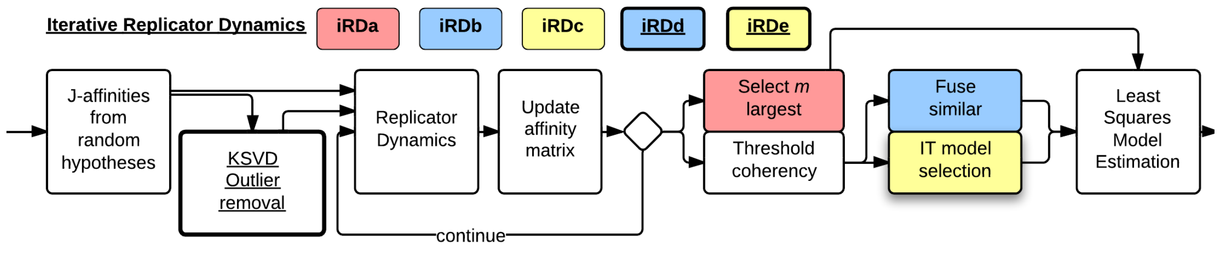


Figure 3.7: Five setups of Replicator Dynamics within the Iterative Graph Clustering pipeline.

Iterative Graph Shift

Instead of evolving Graph Shift from every single node, we now consider Graph Shift as a single mode finder initialized by a random subgraph. We added the option of evolving the algorithm only from a fraction f of vertices, using either $\lfloor \frac{n}{f} \rfloor$ randomly chosen vertices or the $\lfloor \frac{n}{f} \rfloor$ vertices with the highest sum of affinities to other vertices. For an evaluation of how f influences the quality of the algorithm's output, see Section 4.2.4.

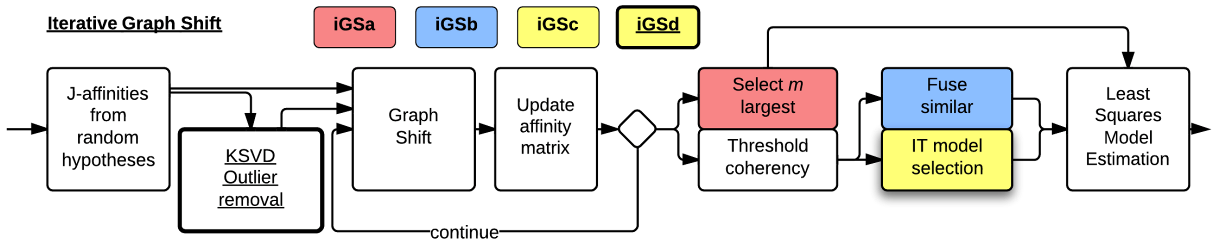


Figure 3.8: Four setups of Graph Shift embedded in the Iterative Graph Clustering approach.

As with other approaches, there are three variants (**iGSa**, **iGSb**, and **iGSc**) of Iterative Graph Shift, distinguished by the way the final set of models is selected. As one may see in Figure 3.8, we also propose a fourth variant, **iGSd**, that uses explicit outlier removal, mainly to allow a fair comparison between Graph Shift and Authority Shift, which requires outlier removal in all variants.

Iterative Authority Shift

As with Graph Shift, the Authority Shift source code was used in two ways. First, as a stand-alone clusterer, which returns the clusters in the last layer of the Authority Shift hierarchy (see Section 2.7.7). Secondly, as a mode finder for our Iterative Graph Clustering approach, yielding only the largest cluster of the algorithm's output. As this variant yielded disappointing results in our first row of experiments of Section 4.2, only one setup **iAS** was considered, shown in Figure 3.9.

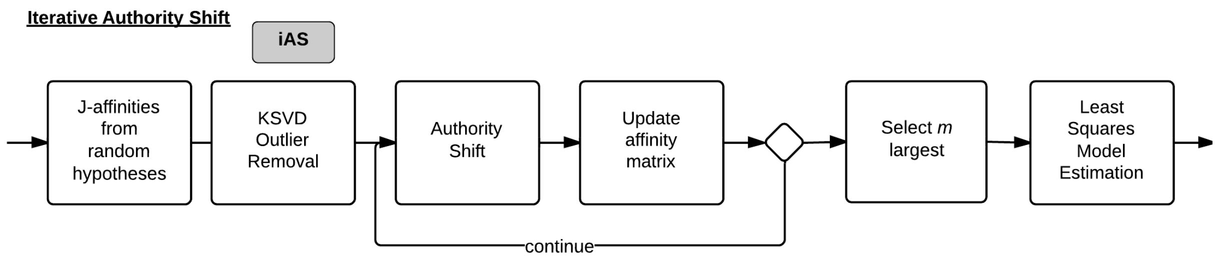

Iterative Authority Shift
iAS
J-affinities from random hypotheses
KSVD Outlier Removal
Authority Shift
Update affinity matrix
Select *m* largest
Least Squares Model Estimation
continue

Figure 3.9: Authority Shift as a single-cluster algorithm embedded in the Iterative Graph Clustering approach.

3.2.7 Kernel Fitting

Besides J-linkage, Kernel Fitting (Section 2.8) is the second reference implementation that can be categorized under the Data Affinity Clustering strategy. We applied the algorithm **KF** unaltered, using the publicly available MATLAB code[4]. Its workflow is shown in Figure 3.10.

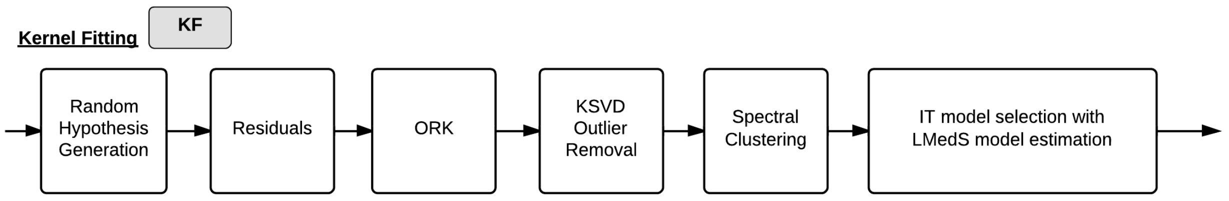

Kernel Fitting
KF
Random Hypothesis Generation
Residuals
ORK
KSVD Outlier Removal
Spectral Clustering
IT model selection with LMedS model estimation

Figure 3.10: The Kernel Fitting pipeline.

3.2.8 Label Costs

The Label Costs approach by Delong at al. is a strategy on its own within our framework, as it is neither RANSAC-like nor based on Data Affinity Clustering. The function that evaluates the cost for a current set of models was taken from the library published by the

[4]http://cs.adelaide.edu.au/~tjchin/doku.php?id=code

authors of [10] themselves[5]. The encompassing PEARL algorithm, on the other hand, was implemented by hand. Since the underlying graph cut algorithm only accepts integer values for label and data costs, we multiplied and all residuals by $1000 \cdot \frac{1}{\sqrt{2}}$ before rounding them. The model fitting pipeline **LC** constituted by this approach is visualized in Figure 3.11.

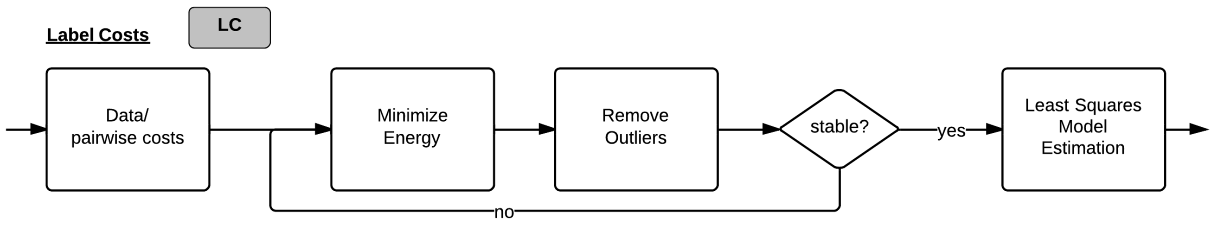


Figure 3.11: Workflow of the Label Costs approach.

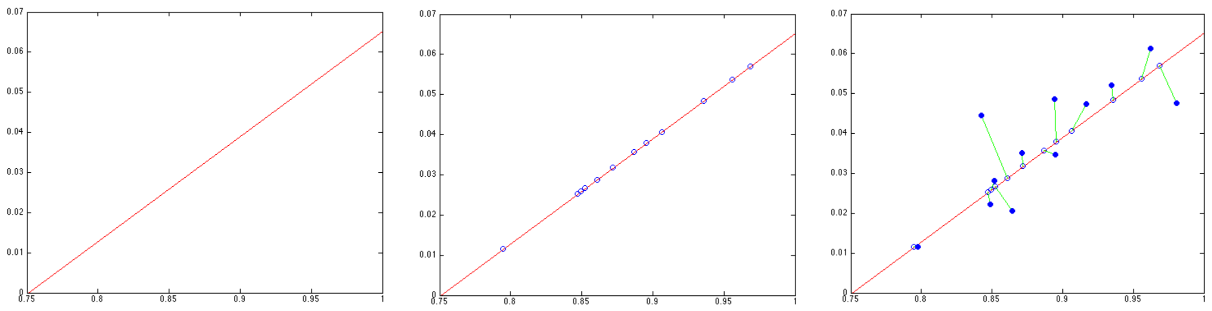
[5]http://vision.csd.uwo.ca/code/

4 Experiments on synthetic 2D data

In this chapter, we will evaluate the different setups of the model fitting framework discussed in Chapter 3 on synthetically generated two-dimensional data. This type of data provides a basis for relatively fast initial testing and evaluation, and similar data were used by [4], [5], [10], [19], [36] and [41]. Although our implementation includes functions for the generation of circle fitting problems as well, we will only consider the application of *line fitting*. This chapter is divided into three sections: First, we will define the artificial data used for the performance of the experiments in Section 4.1. Thereafter, Section 4.2 analyses the algorithms' fitting *accuracy* when the number of structures is known. Finally, Section 4.3 evaluates the estimation of the number of structures, as well as the fitting accuracy, when the number of structures is unknown.

4.1 Generation of Synthetic 2D Data

We implemented a generator of simple, two-dimensional geometric data, which constitutes the input of the framework. n points are sampled from the domain $[0, 1] \times [0, 1] \subseteq \mathbb{R}^2$. A certain number n_o of these points are marked as gross outliers and sampled randomly from a uniform distribution. The remaining $n - n_o$ points are assigned as inliers to a predefined number of four lines, the true structures. These structures are constructed by randomly picking two points from the domain, and are hidden from the model fitting algorithms and only appear as the shape of the Gaussian distributed inliers. Once a structure is defined, an inlier is created by picking a point on the structure, and moving it by a small random vector chosen from a Gaussian distribution with standard deviation σ (see Figure 4.1).



(a) A line is defined, (b) points are chosen on the line, (c) Gaussian noise is added to the points.

Figure 4.1: Generation of synthetic two-dimensional input data.

In our experiments, the number of true inliers per line was set to 50, which amounts to 200 true inliers per dataset. The number of gross outliers was varied from 0 to 700 in steps of 100, which translates to rounded gross outlier percentages of 0%, 33%, 50%, 60%, 67%, 71%, 75% and 78%.

4.2 Evaluation with a predefined number of structures

As mentioned in the introduction, there are two main criteria to be rated in multiple model fitting. This section measures the algorithms' accuracy, i.e. how well the identified models fit the ground truth structures. In the case of two-dimensional line fitting, this means measuring the distance

$$d(\hat{\theta}, \theta) = \min \left\{ ||\hat{\theta} - \theta||, ||\hat{\theta} + \theta|| \right\}, \tag{4.1}$$

where θ and $\hat{\theta}$ are normalized homogeneous representations (normal vector plus distance to origin) of the lines involved. In order to evaluate this criterion in the multiple model fitting scenario, a matching is required between ground truth structures and estimated models, which is achieved by pairing structures and models that have minimal distance:

$$\bar{\varepsilon} = \frac{1}{\min(s, m)} \sum_{k=1}^{\min(s,m)} \min D_k \tag{4.2}$$

where s is the number of structures in the ground truth and m is the number of models yielded by the algorithm for the same image. D_1^s is the set of distances $d(\hat{\theta}, \theta)$ for each ground truth structure $\hat{\theta}$ and estimated model θ. D_2^s is the same set, but without the previous minimum ($\min D_1^s$), etc. To ensure that no setup has an unfair advantage (setups that return many models have a higher chance of yielding high accuracy), the number of models in the algorithms' output is fixed to the number of structures in the ground truth, i.e., $m = s$. Two examples of average fitting errors are given in Figure 4.2.

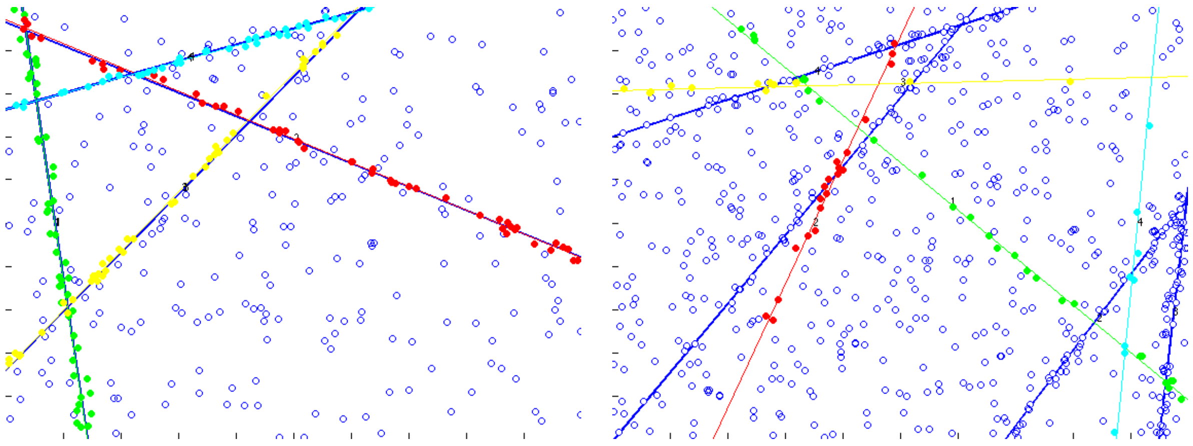

(a) $\bar{\varepsilon} = 0.0025$ (b) $\bar{\varepsilon} = 0.2843$

Figure 4.2: Examples of the average fitting error when the number of structures is given.

Every setup in this section uses random hypothesis generation and a least squares estimator to create hypotheses as well as the final models, except Kernel Fitting, which uses an LMedS estimator [4].

4.2.1 Evaluation of Sequential RANSAC

The first algorithm we evaluated was Sequential RANSAC (**SRa**), which seems a natural starting point considering the low amount of free parameters: the number of iterations of the algorithm can be fixed at four, since that is the number of structures in our ground truth. The second parameter is the number H of random hypotheses that are sampled in each iteration. We decided on $H = 1500$, instead of $H = 5000$ used in [4]. With this amount of hypotheses, the probability q of finding a hypothesis constructed only from inliers of the same structure is greater than 99%, even for the highest amount of outliers:

$$q = 1 - (1 - (\frac{n_s}{n})^2)^H = 1 - (1 - \frac{50}{900}^2)^{1500} \approx 0.99, \tag{4.3}$$

where n_s is number of observations per structure and n is the total number of observations [12].

The third and last parameter to be tuned is the inlier threshold τ, which decides which observations are part of a model's consensus set. This value is set to $\tau = 2\sigma$ in [4]. We decided to compare some multiples of σ, the results of which can be seen in Figure 4.3.

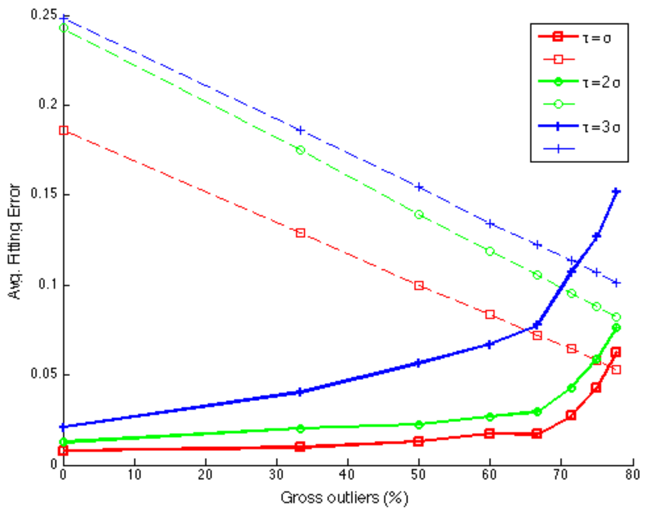


Figure 4.3: Average line fitting error over 100 runs of Sequential RANSAC (**SRa**) for various outlier percentages and inlier thresholds. The dashed lines indicate the average amount of observations assigned to any model for the corresponding run. A lower threshold induces higher accuracy, but smaller consensus sets.

We can see that the most precise fitting occurs with a sharp threshold of $\sigma = \tau$. Beside the fitting error, we also analyzed the average consensus set size (as a fraction of the total number of points). This size naturally decreases as the number of gross outliers grows. It seems interesting that with a threshold of $\tau = 2\sigma$, the algorithm is almost as precise as with a sharp threshold, while returning larger consensus sets.

4.2.2 Evaluation of J-linkage

J-linkage was the first of the broad class of affinity clustering algorithms that we tested. Just like Sequential RANSAC, all affinity-based clusters rely on the construction of random hypotheses and their consensus sets. The difference is that these hypotheses are constructed only once, not separately for each structure. In order to achieve a high probability of drawing inlier hypotheses for all structures, we increase the number of hypotheses to $H = 3500$. For the highest amount of outliers, the probability amounts to

$$Pr\{\text{inlier hypothesis for each structure}\} = 1 - \sum_{k=0}^{m-1} \binom{H}{k} p^k (1-p)^{H-k} \approx 0.99. \qquad (4.4)$$

This equation was also used by [36]. J-linkage was the only algorithm in this section that performed better with $\tau = 2\sigma$ than with $\tau = \sigma$. Instead of discarding clusters based on their absolute size, which is the usual approach of selecting models in J-Linkage, we selected the four largest clusters in order to make a fair comparison to Sequential RANSAC. This corresponds to the setup named **JLa** in Section 3.2. Results are summarized in Section 4.2.6.

4.2.3 Evaluation of stand-alone Authority Shift

As mentioned earlier, the version of Authority Shift employed in our framework does not implicitly separate inliers from gross outliers. This is illustrated with an example in Figure 4.4. For this reason, we always ran the algorithm in combination with the outlier removal step from Section 2.8. This form of outlier removal turned out to work excellent on our synthetic data — as shown in Figure 4.6, it essentially removes all gross outliers successfully. This is remarkable because the outlier removal was used for a different data kernel, i.e. the J-distance kernel instead of the Ordered Residual Kernel.

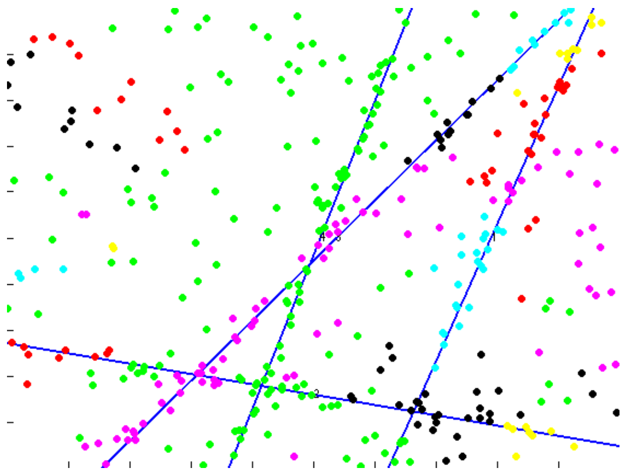


Figure 4.4: If no explicit outlier removal is performed, the Authority Shift algorithm fails to extract clusters consisting of inliers only, making exact model fitting impossible (colors correspond to cluster memberships as estimated by the algorithm).

The characteristic parameter of this clustering algorithm is α, the ratio between walk and jump of the PageRank algorithm expressed in Equation 2.23. Roughly speaking, lowering α increases the percentage of random jumps in the random walk that defines the clustering algorithm [7]. The authors set $\alpha = 0.95$ in their point-clustering experiment. However, we received better clustering results with lower values. To gain certainty, we varied α between 0.05 and 0.95. Curiously, the accuracy of the algorithm doesn't change significantly for any value of α below 0.8, but the error is larger for $\alpha = 0.95$, as one can see from Figure 4.5.

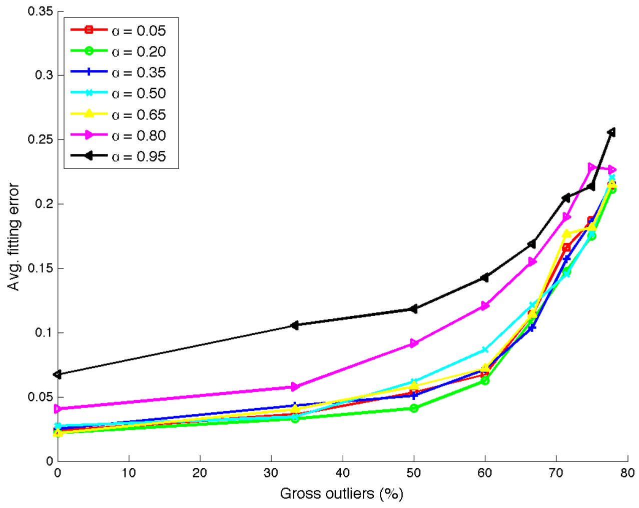


Figure 4.5: Averaged results of Stand-alone Authority Shift (**ASa**) with different values for α over 100 runs and varying outlier percentages. Small α correspond to a high amount of random jumps in the clustering algorithm. The setting used for point clustering by the algorithm's authors, $\alpha = 0.95$, performs worst.

In order to fix the number of returned clusters, we pick the layer of Authority Shift's cluster hierarchy (see Section 2.7.7) which holds at least s clusters. From those, the s largest are selected. This is the setup we called **ASa** in Section 3.2.

The good performance of very low α might an artifact the outlier removal step. As we see in Figure 4.6, the KSVD outlier removal we use is very good at filtering out outliers as long as their total amount is not too large.

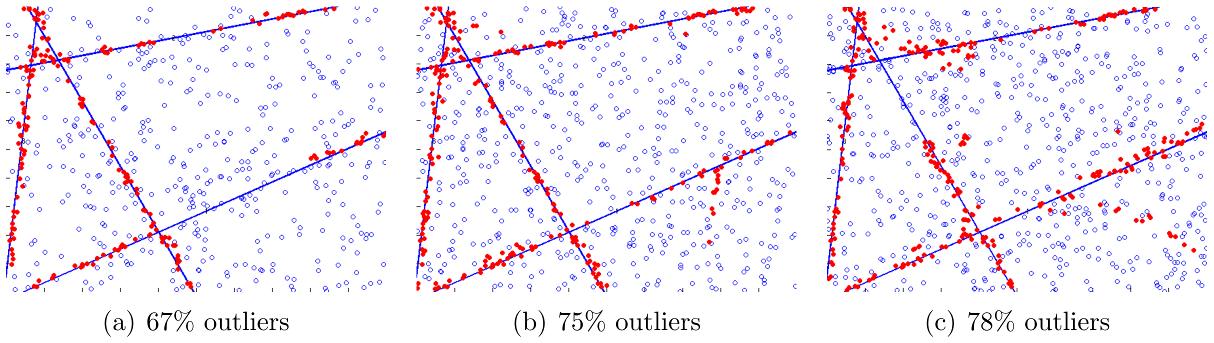
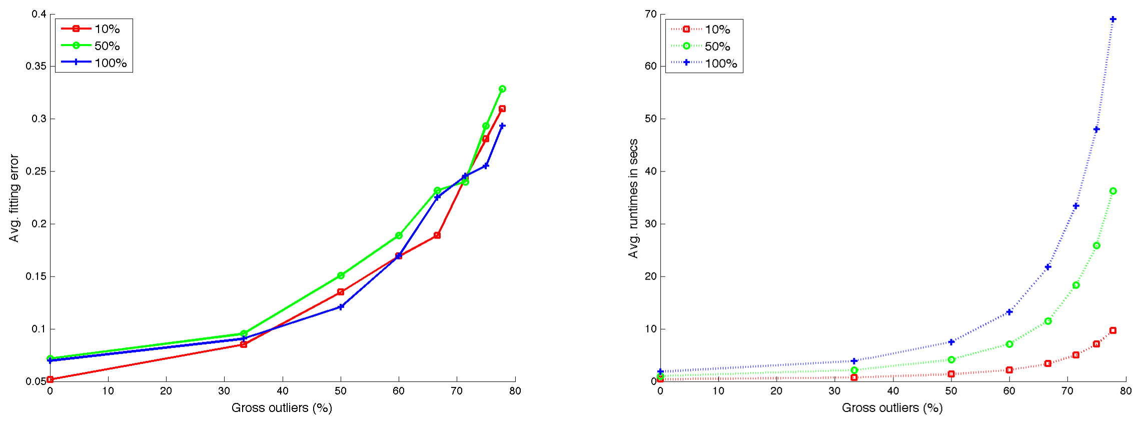

(a) 67% outliers (b) 75% outliers (c) 78% outliers

Figure 4.6: Authority Shift does not remove outliers from its clusters, so we combined it with an extra outlier removal step. The KSVD outlier removal from [4] seems to be very good at dichotomizing inliers with gaussian noise ($\sigma = 0.01$) from randomly distributed outlier noise. Red points are identified as inliers by the removal step. Since it practically removes all outliers, the fitting task seems to become very easy for any algorithm. Only when the outlier percentage rises above 75%, the outlier removal classifies some outliers as inliers.

4.2.4 Evaluation of stand-alone Graph Shift

As mentioned in Section 3.2.6, the Graph Shift implementation from [25] is a combination of the local graph clustering approach from Section 2.7.4 and the Graph Shift algorithm. It requires the definition of subgraphs for the initialization and the number of nodes it can expand in each expansion phase. Furthermore, it lets the user fix the number of inliers per structure, k, returned for every mode, which it acquires by computing the k nearest neighbors of a small set of mode members. Additional parameters, mostly concerning precision, were set as recommended by the implementation's documentation and the included demo code.

We set k to two thirds of the number of points per structure to get a good final fit. We took the $2s$ clusters with the highest coherency, and from those selected the s largest to make sure the number of structures is fixed (**GS**). We set the number of nodes in the expansion phase to the number of points per ground truth structure. The initial subgraphs were set to individual nodes, just like it was done in the demo. The number of starting points was varied from one tenth of the nodes to the full number of nodes. Using a smaller set of starting points speeds up the algorithm significantly, and even increases accuracy for most outlier counts, as visualized in Figure 4.7.



(a) Fitting Error (b) Runtimes

Figure 4.7: Results for stand-alone Graph Shift (**GS**), initialized with different amounts of starting points. All results were averaged over 100 runs on randomly constructed data sets. Choosing fewer starting points speeds up the algorithm considerably, and even seems to improve the accuracy for outlier rates below 70%.

4.2.5 Evaluation of Iterative Graph Clustering

We continued with the evaluation of the Iterative Graph Clustering method from Section 2.7.6, which updates the affinity matrix after each iteration such that the algorithm "steers away" from the previous cluster (Section 3.2.6). We used both Replicator Dynamics and Authority Shift, as well as Graph Shift, as mode finders.

Iterative Replicator Dynamics

Starting with Replicator Dynamics as a mode finding method, again we retrieved better results with inlier threshold $\tau = \sigma$ than with $\tau = 2\sigma$. We set the number of iterations to twice the number of ground truth structures s, and from those selected the largest s (**iRDa**). With fewer iterations, the algorithm often returns small clusters centered around line intersections. By increasing the number of iterations and selecting the largest clusters, this problem is circumvented.

The regularization term λ, which creates negative self-affinities by subtracting a constant from the diagonal of the affinity matrix, greatly influences the performance of replicator dynamics. We tried out several values, as can be seen in Figure 4.8, before settling with $\lambda = 3$.

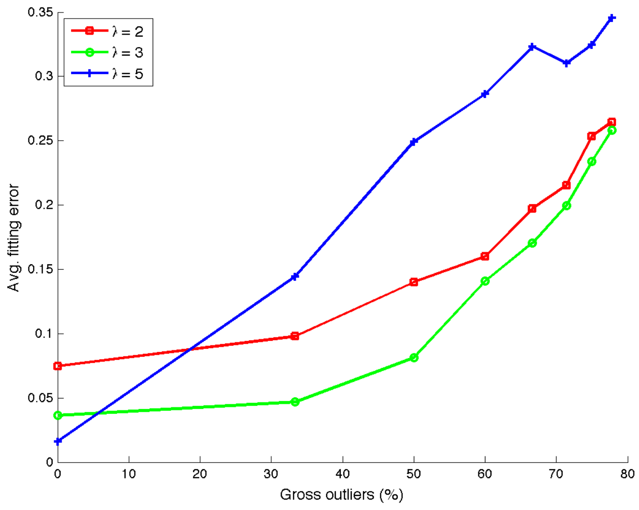


Figure 4.8: Results of Iterative Replicator Dynamics (**iRDa**) for regularizations of $\lambda = 2$ (red), $\lambda = 3$ (green), $\lambda = 5$ (blue). Affinity matrix $M \leftarrow M - \lambda I$. Results were averaged over 100 runs. Inlier threshold $\tau = \sigma = 0.01$.

Iterative Authority Shift

Authority Shift was the second mode finder we analyzed in the iterative clustering setup. Since the Authority Shift implementation does not return probabilistic clusters, we simply set the those entries of the cluster vector $\mathbf{c}$ that are member of the identified cluster C to $1/|C|$ after each iteration, giving each point in the cluster equal probabilistic membership.

For the Authority Shift algorithm **iAS** itself, we used the same settings as in Section 4.2.3, using $\alpha = 0.2$ as walk/jump parameter. The results of this setup will be discussed in this section's summary.

Iterative Graph Shift

Instead of configuring Graph Shift with a multitude of starting points, we initialized it with a single subgraph within our iterative graph clustering loop (**IGSa**). Just like in the previous paragraphs, the subgraph membership vector $\mathbf{c}$ was chosen at a random point on the simplex $||\mathbf{c}||_1 = 1, c_i \geq 0$. An interesting parameter to vary here is the size of the subgraph, i.e., the number of non-zero entries of $\mathbf{c}$. We tried relative sizes of 1%, 10%, 50% and 100% of all data points. Interestingly, the results were comparable for sizes between 1% and 50%, but turned out much worse when the entire graph was used. Using the entire graph obviously reduces the advantage of searching for a mode locally. The runtimes for these variants were nearly identical, the different fitting errors are plotted in Figure 4.9.

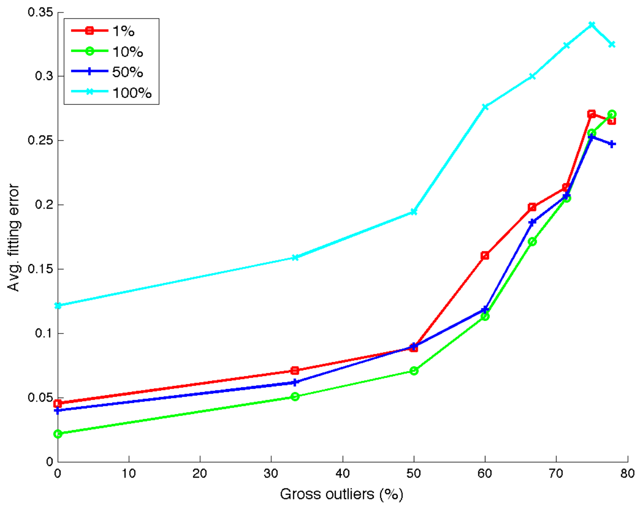


Figure 4.9: Avg. fitting error of Iterative Graph Shift (**iGSa**) for different subgraph sizes. The percentage indicates how many points were used in the initial subgraph of each iteration. The results were averaged over 100 runs.

4.2.6 Comparison

To compare the different algorithms, we took the best performing setup of each approach and plotted them in a common graph (Figure 4.10). The high overall fitting error of J-linkage is remarkable. Authority Shift (**ASa**) does well as a stand-alone approach, while yielding high errors when we incorporate it in our iterative clusterer (**iAS**). For Graph

Shift, the opposite is the case: In our iterative framework, Graph Shift (**iGSa**) has a similar accuracy as Replicator Dynamics, while producing a significantly higher error on its own (**GS**).

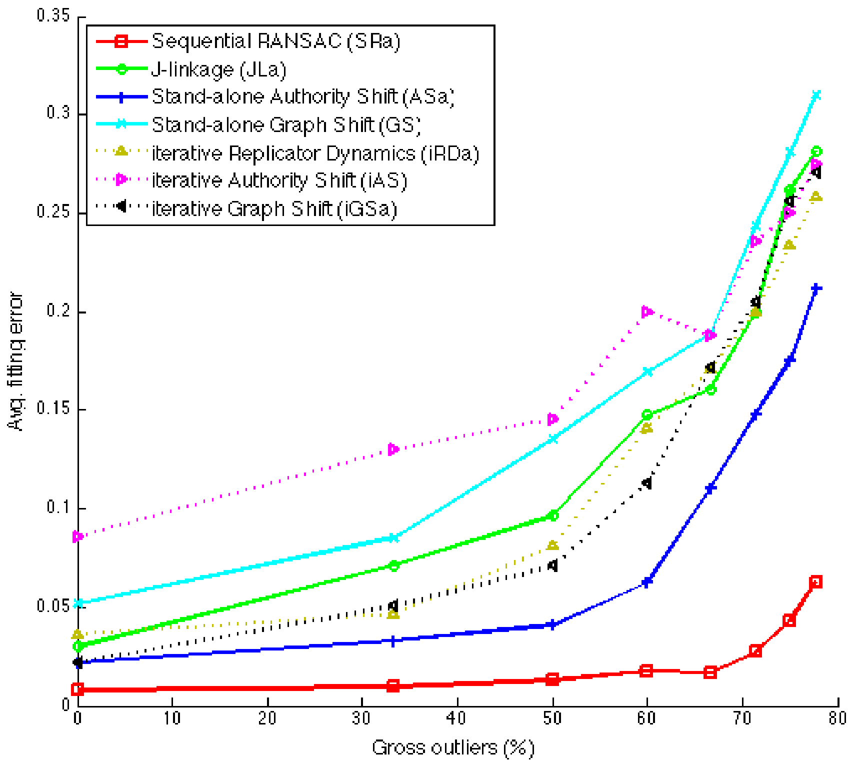


Figure 4.10: Comparison of fitting errors with a fixed number of returned models. All algorithms in this plot (except Sequential RANSAC) operated on an affinity matrix based on the J-distance measure.

4.2.7 Experiments with the Ordered Residual Kernel

Our framework allows us to replace the J-distance kernel used until now with a different affinity measure, for example the Ordered Residual Kernel (ORK) from Section 2.8. The computation of the ORK is slower than that of the J-distance kernel, but given the excellent results Chin et al. presented, it seems worth it to experiment with the combination of ORK and our best affinity clusterer.

Kernel Fitting

Being a reference implementation, we evaluated the Kernel Fitting approach presented in Section 2.8 at first. Again, the number of selected hypotheses was set to 3500. The step size of the ORK algorithm was set to $\frac{H}{50} = 70$. In order to fix the number of models returned by the algorithm, we cut the model merging scheme of Section 2.8 short by breaking from the merging loop as soon as the number of models equals our desired number.

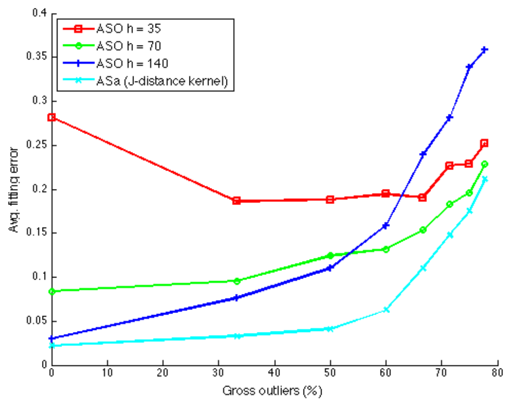
This approach proved slower than most other approaches we evaluated so far. The computation of the kernel matrix and the model selection phase were the most costly (each took up to 5 seconds on our reference machine). However, its performance was much better than that of our affinity clusterers (Figure 4.12), while still not beating simple Sequential RANSAC .

Authority Shift with ORK affinities

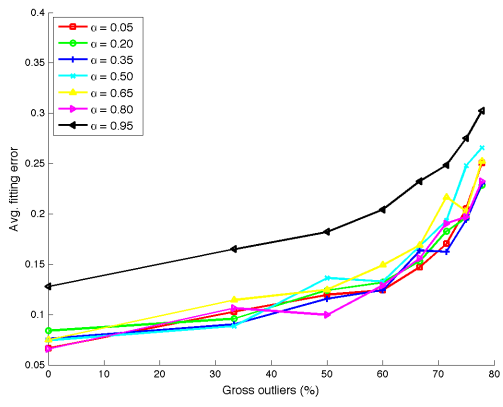
Since we already used the outlier removal stage of the Kernel Fitting approach with Authority Shift in previous experiments, an obvious choice for an additional setup is the combination of Authority Shift with the ORK affinity matrix (**ASO**). Note that the difference between Authority Shift with ORK and the original Kernel Fitting approach consists of the clustering step as well as the final model selection, which is a computationally heavy step in Kernel Fitting.

Evaluation over step size h We initially assumed a fixed value of $\alpha = 0.2$ (see Section 4.2.3) for the Authority Shift algorithm. The main parameter for the Ordered Residual Kernel is its step size h. This integer determines how many "Difference of Intersection Kernels" (DOIKs) are summed to form the Ordered Residual Kernel itself, as we explained in Section 2.8. For example, as we generate 3500 random hypotheses, a step size of $h = 70$ would produce $3500/70 = 50$ DOIKs. The authors used step sizes of $h = 20$ for 1000 hypotheses in their demo code, and $h = 100$ for 5000 hypotheses in the experiments presented in their paper [4]. Therefore it seems safe to assume a step size that amounts to one fiftieth of the number of generated hypotheses. We verified this by experimenting with values of $h = 35, 70$ and 140. Looking at Figure 4.11(a), the value $h = 70$ indeed seems a good compromise, producing relatively small fitting errors for any amount of gross outliers.

Evaluation over walk/jump parameter α Since the previous experiment with a fixed value of $\alpha = 0.2$ did not achieve the same high accuracy as the same algorithm on the J-linkage kernel, we tried to improve it by choosing a different value for α. As in Section 4.2.3, we varied α between 0.05 and 0.95 in steps of 0.15. Although the fitting error could be reduced slightly when we used $\alpha = 0.35$ (Figure 4.11(b)), this approach did not beat our previous Authority Shift approach in terms of accuracy.



(a)



(b)

Figure 4.11: (a) Fitting error of Authority Shift applied to the Ordered Residual Kernel (**ASO**) for different step sizes h. Using walk/jump parameter $\alpha = 0.2$, the ORK version of Authority Shift doesn't reach the low error produced by our original Authority Shift (**ASa**), which works with the J-distance kernel. **(b)** Varying the α-parameter did not significantly improve the accuracy of **ASO**. Results were averaged over 100 runs.

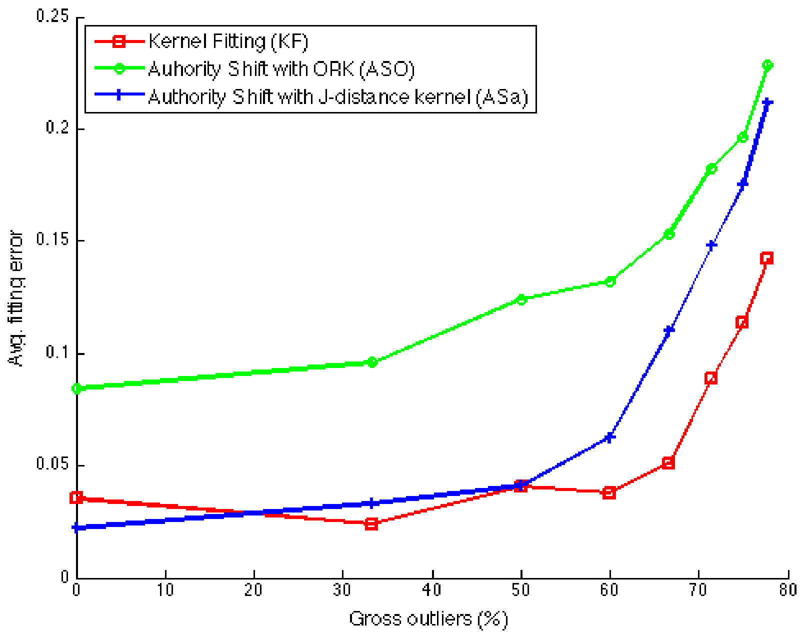


Figure 4.12: This plot compares the accuracy of Kernel Fitting (**KF**), Authority Shift on the ORK kernel (**ASO**), and regular Authority Shift (**ASa**). Kernel Fitting outperforms both Authority Shift-based approaches, while replacing the J-distance kernel with the Ordered Residual Kernel does not bring any improvement to Authority Shift in terms of accuracy. Results were averaged over 100 runs.

4.3 Evaluation without cardinality restrictions

In the previous section, we discussed the performance of different algorithms when the number of structures they had to fit was known a priori. This enabled us to focus on the fitting *accuracy*. Now, we want to compare the same approaches in terms of cardinality estimation, i.e., how well do they estimate the number of structures in the data?

For this purpose, we introduce two measures:

- The *cardinality error* simply measures the difference $|s-m|$ between the true number of structures s and the number of models m returned by the algorithm.
- The *multi-structure fitting error* (MSFE), introduced by [4], incorporates both the cardinality error and the fitting accuracy:

$$\varepsilon \quad = \quad |s-m| \quad + \sum_{k=1}^{\min(s,m)} \min D_k. \tag{4.5}$$

The second term is the sum of fitting errors, comparable to the error in Equation 4.2. The first term is always a non-negative integer, while the second term usually remains below one. This means that from an MSFE of for example $\varepsilon = 3.17$, we can deduce that the algorithm returned three models too many (or too little), and that its fitting error was 0.17. However, since we average the MSFE over 100 runs in our evaluation, we plot both the MSFE and the cardinality error in all the figures in this section.

Obviously, comparing the MSFEs of different algorithm setups is only useful as long as the number of structures in the ground truth remains constant. Fortunately, this is the case in our experiments. The MSFE has one counter-intuitive property: When an algorithm finds fewer models than there are structures in the ground truth, it is punished to the left of the decimal point, but not in the decimal places. For example, if an algorithm makes a fitting error of 0.4 per structure, and detects only three out of four structures, the MSFE amounts to $1 + 3 \cdot 0.04 = 1.12$. Therefore, the difference between cardinality error and MSFE is 0.12. However, if the number of structures had been known to the algorithm, the total fitting error would have been $4 \cdot 0.04 = 0.16$, which on the first glance may appear as a higher error than the 0.12 from the MSFE. Also, since we only show averaged errors, an algorithm with an MSFE of 0.5 may be one that estimates the cardinality correctly every time, but fits very poorly, or one that produces a cardinality error of 1 in 50% of the runs, but fits the models very exactly. Examples of multiple model fitting results with different MSFEs are given in Figure 4.13.

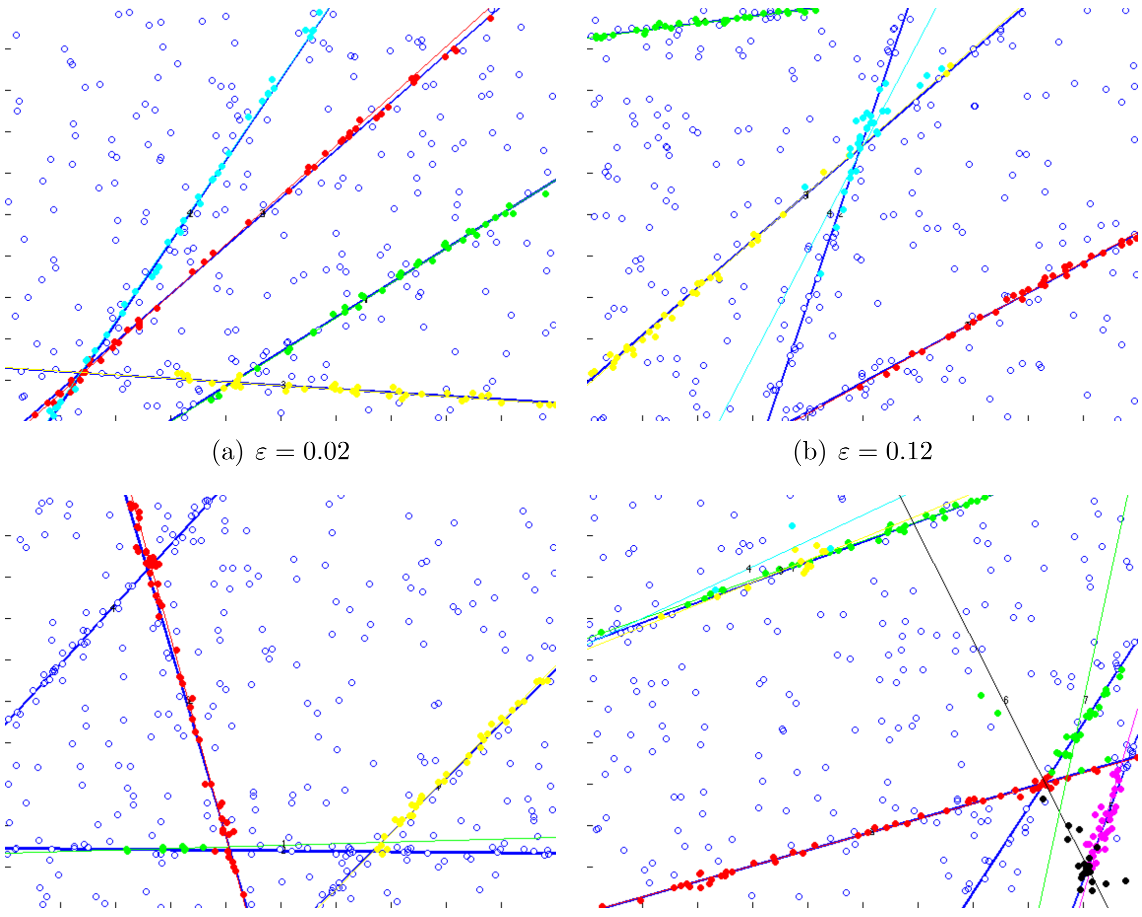

(a) $\varepsilon = 0.02$ (b) $\varepsilon = 0.12$ (c) $\varepsilon = 1.05$ (d) $\varepsilon = 3.24$

Figure 4.13: Fitting results with various MSFEs (multi-structure fitting errors).

4.3.1 Kernel Fitting and model selection

We decided to start our evaluation with the Kernel Fitting algorithm, since its source code features a powerful approach to model selection, which we described in Section 2.8. The parameter β of Equation 2.31 determines the relative importance between obtaining a good fit for the data and the complexity of the total model (i.e. the number of models), and has to be tuned. The authors of Kernel Fitting [4] fixed it at $\beta = 20$. We tried $\beta = 5$ and $\beta = 10$ as well, with $\beta = 10$ being the best choice, at least for datasets with more than 50% outliers, as we can see in Figure 4.14.

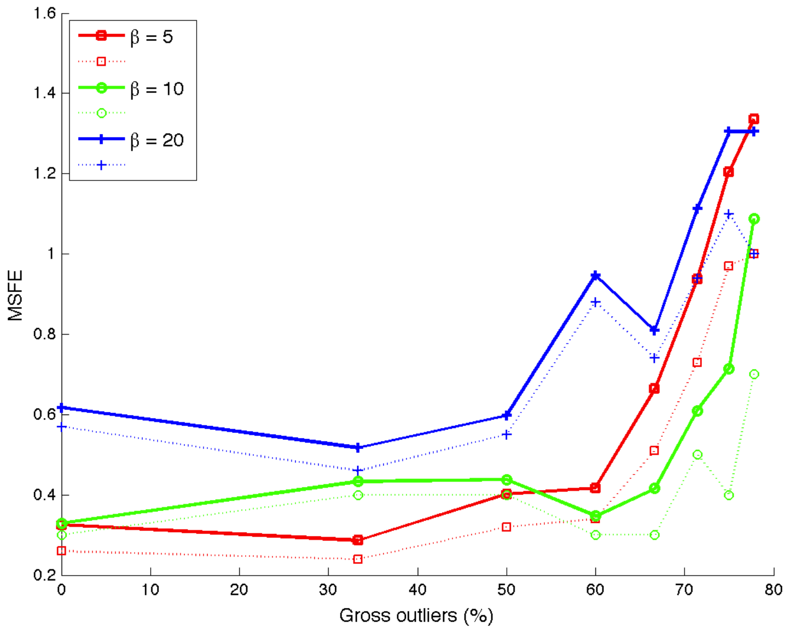


Figure 4.14: Kernel Fitting (**KF**) results for various values of β, which weights the cost of fitting quality and model complexity. Thick lines represent the MSFE, thin lines represent the cardinality error. Results were averaged over 100 runs.

Note on computational efficiency We were able to speed up model selection by a factor of four, simply by caching merged structures while constructing the hierarchy.

4.3.2 Evaluation of Sequential RANSAC

We resume evaluation with the simplest approach of sequentially applying RANSAC to the data. Since we do not want to give the algorithm the number of structures beforehand, we have to determine a different stopping criterion. The simplest would be a minimal number of inliers per model: as soon as the number of inliers falls below a threshold in iteration k, the iteration stops and returns only the first $k-1$ structures. This is setup **SRb**. The second criterion involves evaluating the cost of explaining the data, for which we use the same cost function as in Section 4.3.1 (setup **SRc**).

For the first stopping criterion, we defined a percentage of data rather than a fixed number for the minimal consensus set size. This partially resolves the problem of increasing outlier counts. We could, of course, define the minimal size as a function of the number of inliers and gross outliers in the ground truth, but this would give the algorithm an unfair advantage over other approaches.

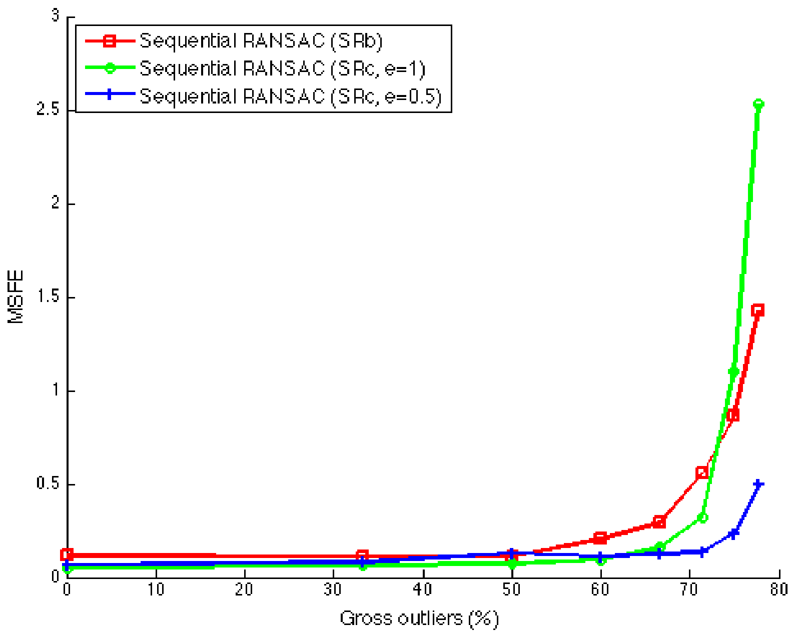


Figure 4.15: Comparison of different stopping criteria for Sequential RANSAC. One can see that an information criterion (blue,green) can achieve better results than a simple size-based criterion (red). For the information criterion, the artificial outlier residual r_o plays a major role.

For the second criterion, we compare the cost of the set of models in the current iteration with the cost of the set of models in the previous iteration. As soon as the current cost $\varphi(\mathcal{M}_k)$ exceeds the previous one, the iteration is stopped and the previous set of models is returned. Using the results from the previous subsection, we set $\beta = 10$. Since the cost function in Equation 2.31 only considers inlier residuals, we handled outliers by introducing an artificial "outlier model", for which every non-inlier has a residual of r_o. We initially set $r_o = 1$, which constitutes the highest distance any point can have to a line within the domain $[0, 1] \times [0, 1]$, but achieved better results with $r_o = \frac{1}{2}$. Making outliers too expensive obviously encourages the algorithm to introduce new models, leading to a higher MSFE, as can be seen in Figure 4.15.

In contrast to the evaluation with a known number of structures, Sequential RANSAC performed better with $\tau = 2\sigma$ than with $\tau = \sigma$ as far as the MSFE is concerned. This may have something to do with the fact that the algorithm removes all inlier observations after each iteration. With $\tau = 2\sigma$, the cut is more coarse, removing observations that otherwise could lead to the construction of a second, similar model.

4.3.3 Evaluation of J-linkage

The parameter which implicitly regulates the number of models returned by J-linkage is the minimal cluster size n_{min}. Although the authors of J-linkage [36] set this value to $n_s/2$, i.e. half the number of inliers per ground truth structure, we found that the algorithm performs best with values around ten, corresponding to $n_s/5$. We conducted runs with $n_{min} = 9, 10, 11$ to show the delicacy of this choice (Figure 4.16). Just like in the accuracy evaluation, an inlier threshold of $\tau = 2\sigma$ yielded the best results.

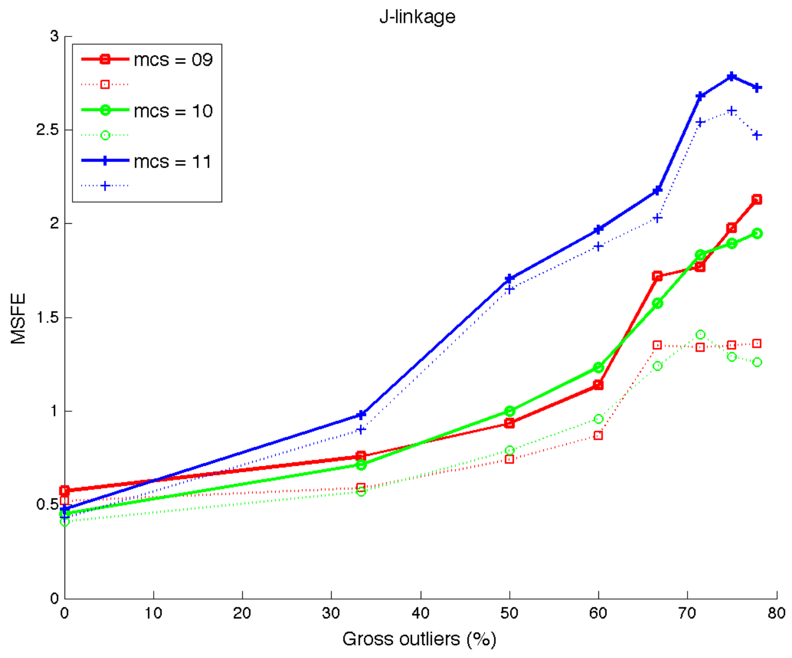


Figure 4.16: MSFE and cardinality errors for J-linkage (**JLb**) with various minimal cluster sizes. Inlier threshold $\tau = 2\sigma$. Results were averaged over 100 runs.

After the success of using a cost criterion for model selection with Kernel Fitting in Section 4.3.1, we tried the same with J-linkage, the results of which are presented in Figure 4.17.

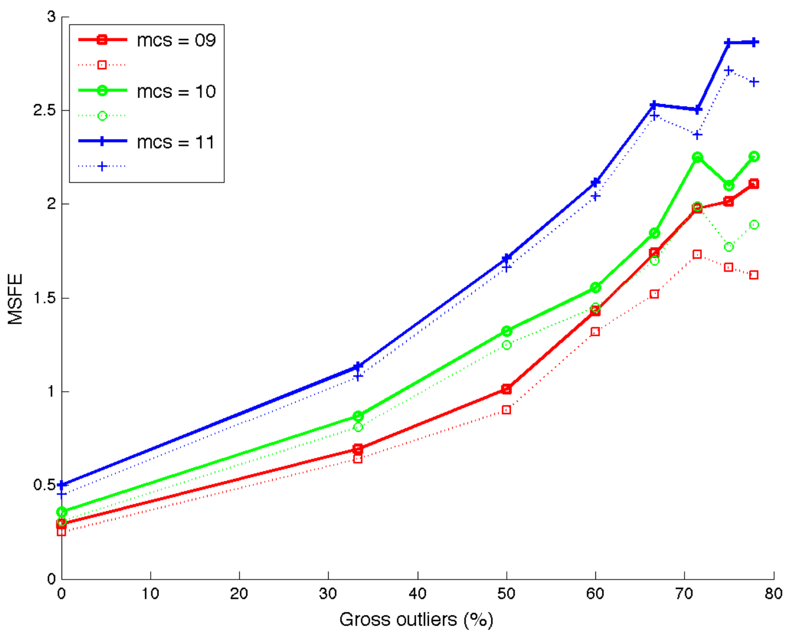


Figure 4.17: MSFE and cardinality errors for J-linkage with Model Selection (**JLc**). Inlier threshold $\tau = 2\sigma$. Results were averaged over 100 runs.

4.3.4 Evaluation of stand-alone Authority Shift

For Authority Shift, we kept the value of $\alpha = 0.2$ from accuracy evaluation, but tried different minimal cluster sizes and model fusers. All variants performed best with $\tau = \sigma$. First, we fixed the minimal cluster size to $0.3n_s$ (about one third of the number of inliers per ground truth structure), and ran the clusterer with three different fusing setups: the first leaves the clusters returned by Authority Shift unaltered (**ASb**). The second fuses lines with similar line representations (**ASc**). The third setup performs information-based model selection as in Section 4.3.1 (**ASd**). For the last setup, we also tried reducing the minimal cluster size, shifting the focus of the model fitting task towards the model selection procedure. This reduced the MSFE and fitting error even further. The entire experiment is visualized in Figure 4.18.

The most evident and exciting result in Figure 4.18 is the good performance of the variant with cost-based model selection. Without a cost-based model selection step, the algorithm is best configured with a minimal cluster size of $15 = 0.3n_s$. A lower threshold for the cluster size obviously returns more promising clusters, while false positives are filtered out by the powerful model selection procedure. Apparently, the combination of outlier removal, a correctly configured clustering step and an information-based selection phase are necessary building blocks for accurate multiple model fitting with graph clusterers.

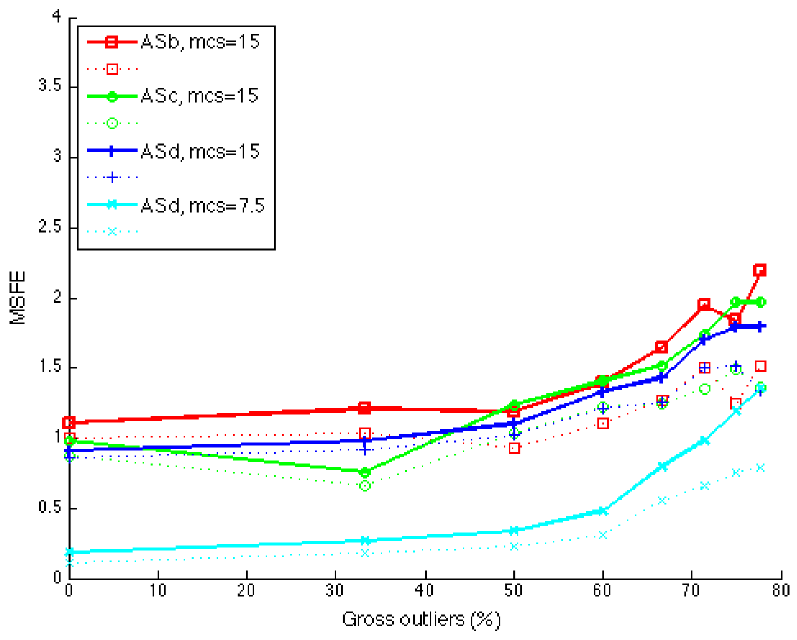


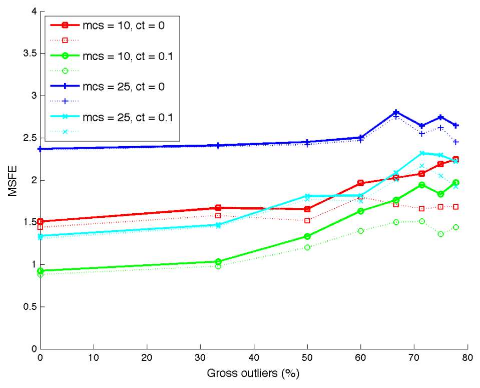
Figure 4.18: MSFE and cardinality errors for different setups of Authority Shift. Results were averaged over 100 runs.

4.3.5 Iterative Graph Clustering

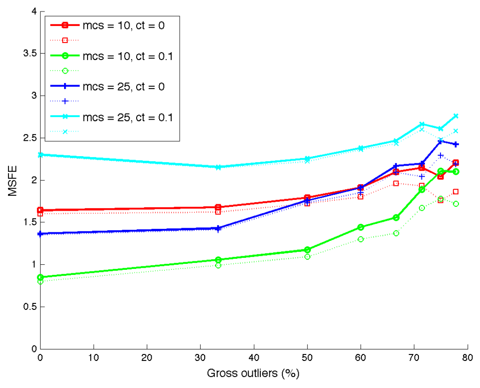
Of course, the previous insight is not new: For the largest part, the credit for the good performance of the algorithm shown in Figure 4.18 goes to the authors of Kernel Fitting[4]. In fact, we used their components and merely exchanged the clustering step. However, we did achieve an even lower MSFE than the original Kernel Fitting approach, at least on our own data. Moreover, computation of the J-distance kernel is faster than the computation of the Ordered Residual Kernel. To see if we can reach similar performance with different clusterers, we evaluated iterative graph clustering with Replicator Dynamics and Graph Shift. We discarded the iterative variant of Authority Shift, **iAS**, as it did not perform well in our accuracy evaluation.

Replicator Dynamics

For Replicator Dynamics, we set the maximal number of iterations to 20, but also installed a cluster coherency threshold and a minimal cluster size, which we varied as shown in Figure 4.19. One can see that using the expensive cost-based model fusing step does not improve the fitting quality significantly, if at all.



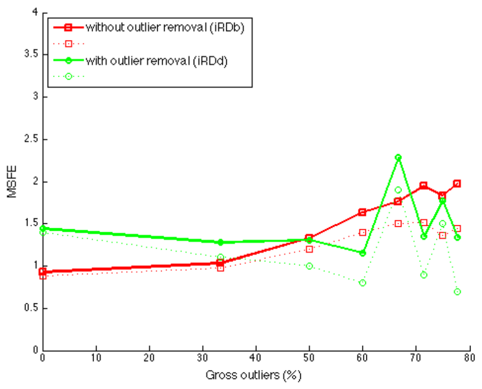
(a) **iRDb**



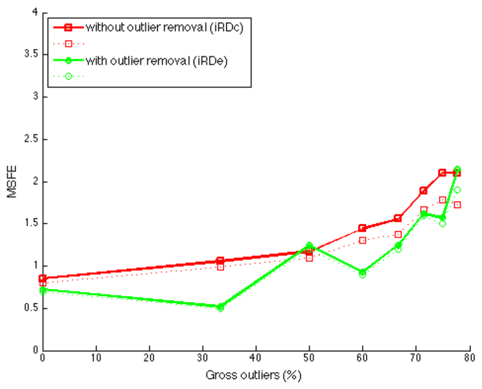
(b) **iRDc**

Figure 4.19: Multiple structure fitting errors and cardinality errors for Iterative Replicator Dynamics (a) with and (b) without cost-based model selection. "mcs" stands for minimal cluster size, "ct" for coherency threshold. Results were averaged over 100 runs.

So far, the quality of the setups are comparable to the ones in Section 4.3.4. In order to find out if outlier removal brings the same boost to Replicator Dynamics as it did to Authority Shift, we evaluated the Iterative Replicator Dynamics setup $n_{min} = 10, \tau_c = 0.1$ **with** KSVD outlier removal conducted prior to the clustering. This led to the results of Figure 4.20. The outlier removal does improve the results for most outlier counts, but only when cost-based model fusing is conducted afterwards, and not as much as one might expect.



(a) **iRDb** / **iRDd**



(b) **iRDc** / **iRDe**

Figure 4.20: Multiple structure fitting errors and cardinality errors for Iterative Replicator Dynamics with (a) and without (b) cost-based model selection, both with and without prior outlier removal. Results were averaged over 100 runs.

Iterative Graph Shift

We deliberately discarded the stand-alone variant of Graph Shift (**GS**) because it did not perform well in our accuracy evaluation. For Iterative Graph Shift, we chose the basic settings for the algorithm as in Section 4.2.5. The minimal cluster size does not have to be set since the Graph Shift source code always returns fixed-sized clusters. Finally, the coherency threshold was set to $\tau_c = 0.1$ as with Iterative Replicator Dynamics. With cost-based model selection (**iGSc**), we achieved a good fitting quality for outlier percentages up to 50%. Even without cost-based model selection (**iGSb**), results were not worse than for most of our other setups. However, the results improved remarkably when we added outlier removal to the pipeline (**iGSd**), with which the MSFE remained well below one, and below 0.5 for outlier percentages up to 60%. (Figure 4.21). Interestingly, the variant without outlier removal scored slightly better for outlier ratios under 40%. However, we do not claim that this difference (0.0433 on average) is statistically significant. Overall, (**iGSd**) performed even better than the Authority Shift setup **ASd** from Section 4.3.4, with an error difference of 0.1247 on average.

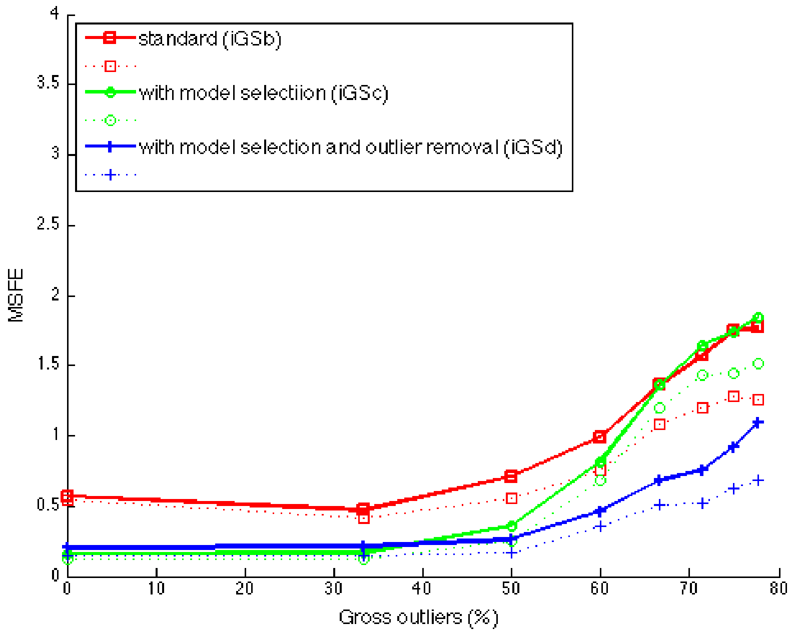


Figure 4.21: MSFE and cardinality errors for different setups of Iterative Graph Shift. Results were averaged over 100 runs.

4.3.6 Note on cost-based model fusing

Note that the smaller the minimal cluster size, the more workload is shifted from the clustering algorithm to the model selection algorithm. In theory, we could pass all 3500 random hypotheses directly to model selection, which would probably lead to good results, but would require $\binom{3500}{2} + \binom{3499}{2} + \cdots + \binom{3}{2} \approx 7 \cdot 10^9$ evaluations of the cost function. Hence, the decision which hypotheses are passed to model selection is a trade-off between fitting quality and runtime.

4.3.7 Label Costs

Besides Kernel Fitting, Label Costs (**LC**) constitutes another state-of-the-art approach to multiple model fitting. As we know from Chapter 2, we have to define data and label costs for the optimization procedure, the definition of which has to be carefully tuned.

Parameter tuning

The cost of labeling a point $\mathbf{x}^j$ with a label i was defined as proposed for line fitting by the authors in [10, equation 13]:

$$D_j(i) = -\ln\left(\frac{1}{\sqrt{2\pi}\sigma} \exp - \frac{r_{ij}^2}{2\sigma^2}\right), \tag{4.6}$$

where $r_i j$ is the residual between hypothesis i and point $\mathbf{x}^j$, and σ^2 is the variation of the residuals. We introduced an extra outlier label 0. Since there are no residuals for this label, we manually tuned the cost $D(f(\mathbf{x} = 0))$ for assigning this label to an observation. The relation between this outlier cost and the cost of introducing a new label, $c_l = c$, is a delicate one: If the cost $D(f(\mathbf{x} = 0))$ of assigning the outlier label to an observation is too low, all observations are labeled as outliers. If it is too high, all observations are assigned to a structure and the fitting becomes imprecise. When the cost of introducing a label is low, it becomes cheaper to introduce an extra (unwanted) model than to assign outliers. This is especially true when the percentage of gross outliers in the data is high. When the cost of introducing a label is too high, the algorithm obviously fails to identify all four structures (Figure 4.22). It would seem best to configure the relation between these two costs as a function of the true outlier percentage, but as we said, we deliberately do not pass the outlier count to the algorithms.

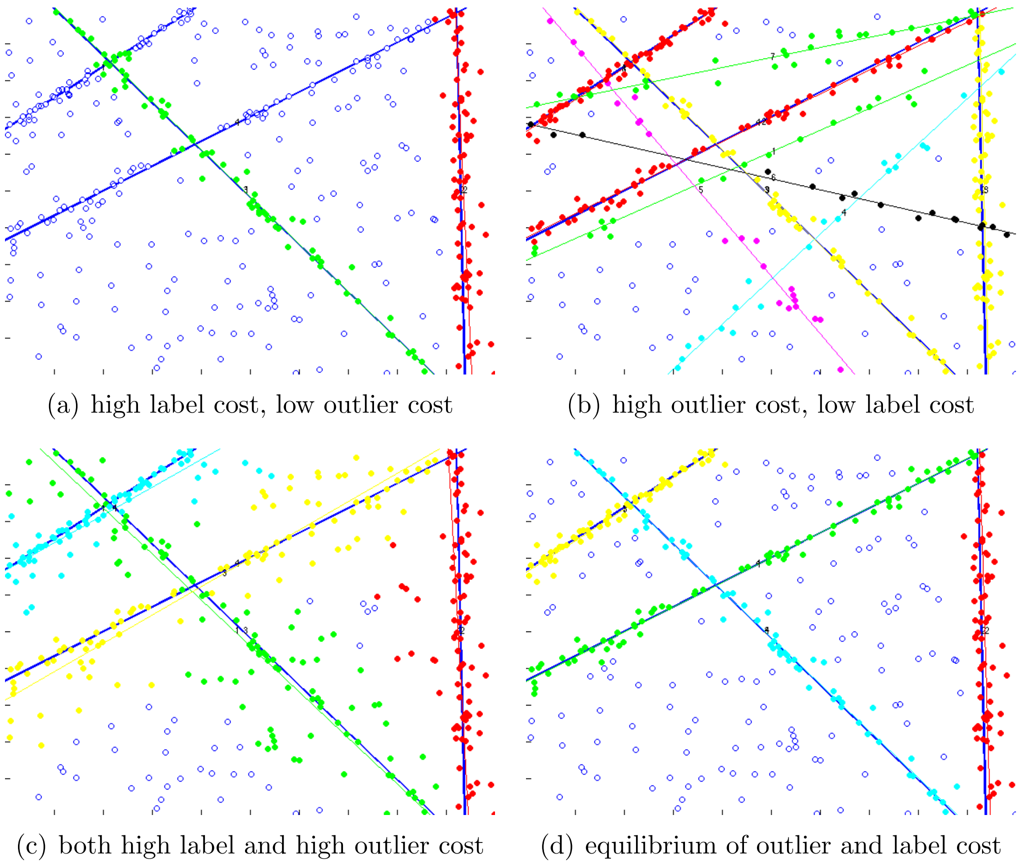

(a) high label cost, low outlier cost
(b) high outlier cost, low label cost
(c) both high label and high outlier cost
(d) equilibrium of outlier and label cost

Figure 4.22: Example of how the definition of cost functions influences the output of the **LC** model fitting algorithm. MSFEs from (a) to (d): $2.07, 5.02, 0.12, 0.01$.

As in [10], the smoothness term $V(\mathbf{x}, \mathbf{y})$ was set to zero because the data are independent and identically distributed.

4.3.8 Comparison of setups

We summarized the performance of the best setups from this section in Figures 4.23,4.24 and 4.25. In order to make a fair comparison, we decided to group the evaluated setups into three classes:

- Setups without gross outlier removal and without cost-based model selection (Figure 4.23),
- Setups without gross outlier removal, but with cost-based model selection (Figure 4.24),
- Setups with both gross outlier removal and cost-based model selection (Figure 4.25).

Finally, we compare the “top five” of these algorithms in Figure 4.26.

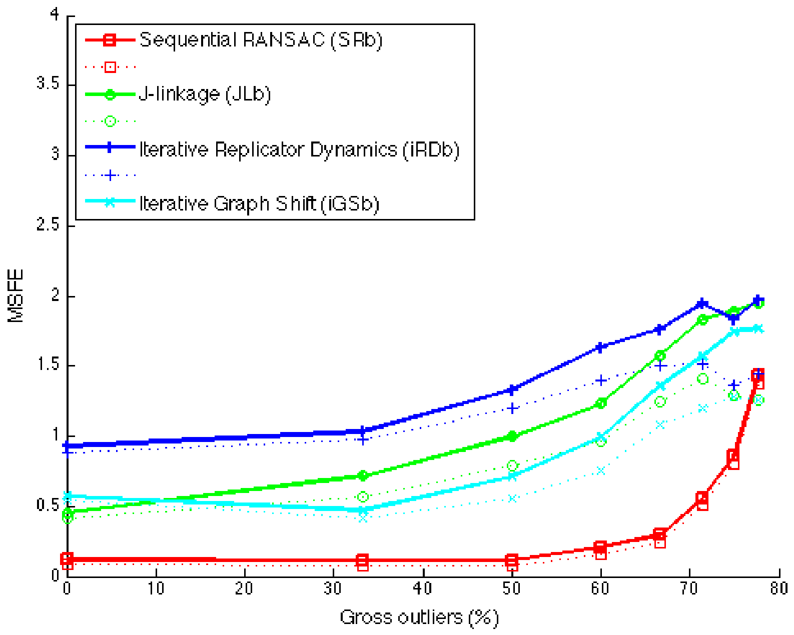


Figure 4.23: Comparison of algorithms without gross outlier removal or cost-based model fusing. Remarkably, the cardinality errors for all approaches meet at the highest outlier rate, while the MSFEs lie farther apart. It seems likely that Sequential RANSAC has the same difficulties estimating the number of structures as the other algorithms, but fits them more accurately. Results were averaged over 100 runs.

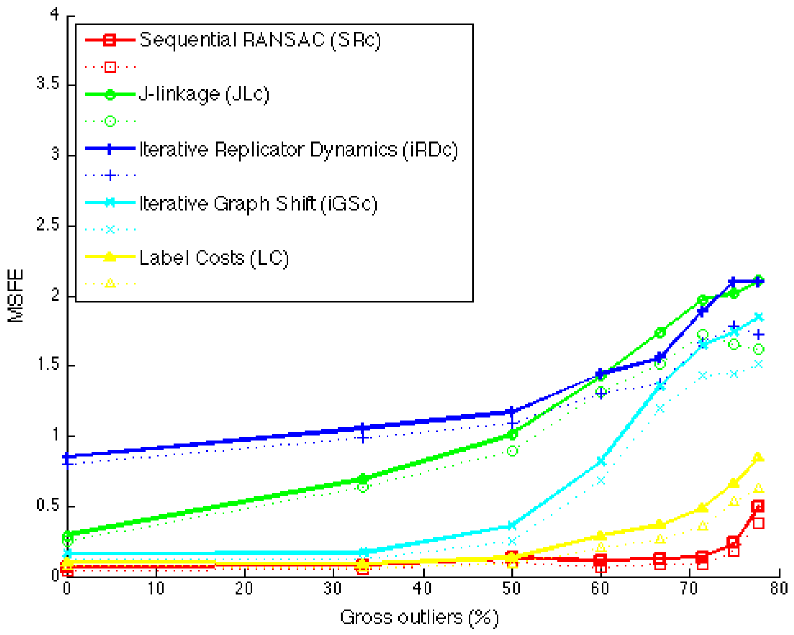


Figure 4.24: Comparison of algorithms without gross outlier removal, but with cost-based model fusing. We decided to add Label Costs to this plot, since it uses a cost-based criterion similar to ours. One can see that the introduction of cost-based model selection does not greatly improve cluster-based algorithms, but the cost-based stopping criterion of Sequential RANSAC boosts its performance considerably. Results were averaged over 100 runs.

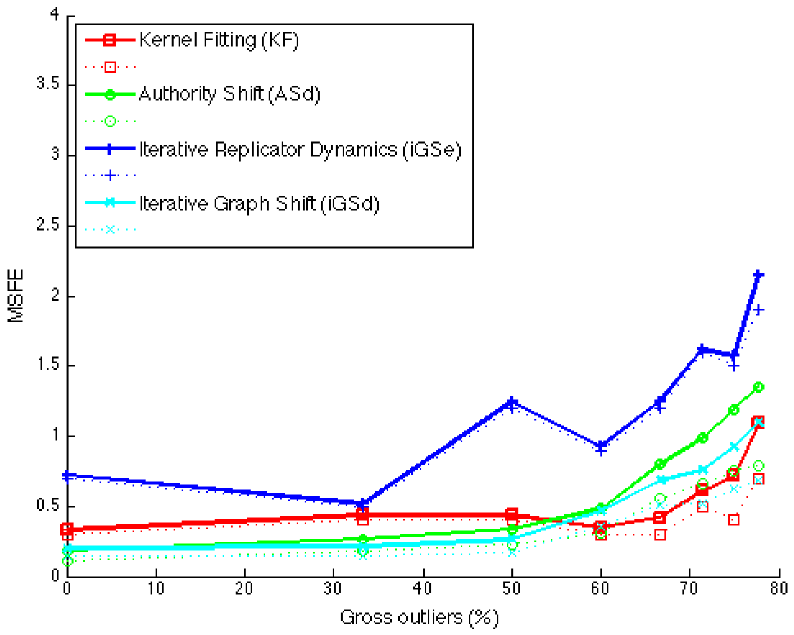


Figure 4.25: Comparison of algorithms with both gross outlier removal and cost-based model fusing.The gap between Iterative Graph Shift and Iterative Replicator Dynamics remains large. This corresponds to the results of [25], which claims Graph Shift to be more successful at identifying dense subgraphs than simple Replicator Dynamics. The second notable fact is that for outlier rates up to 50%, our Graph Shift and Authority Shift setups attained a lower MSFE than their Kernel Fitting counterpart. Results were averaged over 100 runs.

The Authority Shift and Graph Shift setups **ASd** and **iGSd** displayed in Figure 4.25 are essentially variants of Kernel Fitting, simply using a different kernel and a different clusterer. They both use the same functions for outlier removal and model selection as the reference implementation by Chin et al. Their paper [4] claims a much lower MSFE for line fitting, but as mentioned earlier, they use the same set of lines in each run of the experiment, which makes it easier to tune parameters accordingly. Also, our versions are faster, not just because we cache structures in the model selection phase, but also because the computation of the J-distance kernel is more efficient than that of the Open Residual Kernel (about 3 seconds versus up to 9 seconds on our reference machine). For details on runtimes, see Section 4.4.

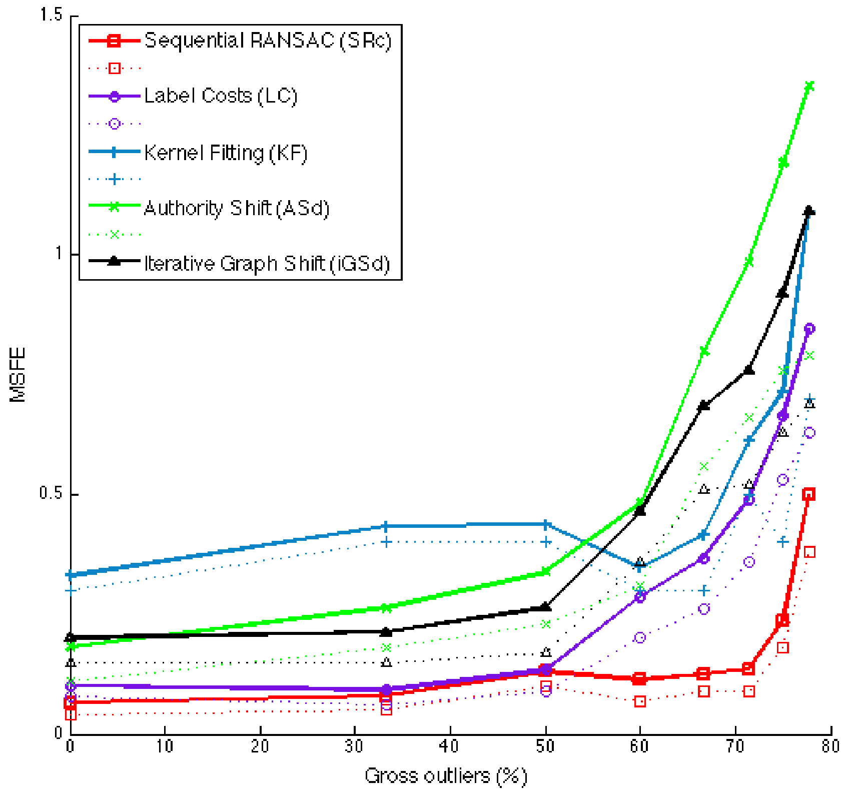


Figure 4.26: Top five of all algorithms compared in this section. Sequential RANSAC and Label Costs remain unbeaten, but our approaches based on Authority Shift and Graph Shift achieve results comparable to Kernel Fitting.

Summary of manual parameter inputs

The different approaches that were evaluated in this section were not treated completely equally. Due to the nature of each algorithm, most of them require minimal knowledge about the application and/or the ground truth. Which algorithms take which parameters is summarized in Table 4.1.

Besides these explicit parameters, some algorithms require delicate manual tuning of parameters for each domain. It is safe to say that this form of parameter tuning passes *implicit* knowledge about the application and the ground truth to the algorithms.

Algorithm	Inlier noise scale	Inlier count
Sequential RANSAC with stopping criterion	●	
Kernel Fitting		
J-linkage	●	●
Authority Shift	●	●
Iterative Replicator Dynamics	●	●
Iterative Graph Shift	●	●
Label Costs		

Table 4.1: Ground truth-related parameters that were passed to the algorithms of this section. The inlier count refers to the number of observations per ground truth structure. Algorithms do not rely on the inlier count explicitly, but we use it to define a minimal cluster size.

4.4 Runtime comparison

In this section, we compare the runtimes of different setups from Section 4.3. It is difficult to conduct a completely objective runtime comparison between the setups: Some of the functions incorporated in our framework, such as the construction of affinity matrices, were implemented in C/C++, while others were implemented in MATLAB. Due to the nature of MATLAB programming, some parts of each pipeline may have been implemented very efficiently, while others rely on nested `for`-loops and are therefore less efficient. Therefore, we conduct a qualitative runtime analysis, based on the average amount of time each setup required (measured by MATLAB's `tic` and `toc` functions) on our reference machine, a 2.3 GHz dual core with 4GB RAM.

In Figure 4.27, we summarized the runtimes we recorded for each algorithm setup. Some variants of evaluated approaches had very similar runtimes (e.g. those with vs. without cost-based model selection). To maintain clarity, we only plotted one curve for these variants.

Iterative Replicator Dynamics was the fastest of our clusterers, closely followed by those variants of Authority Shift and Iterative Graph Shift that include an explicit gross outlier removal step. It is obvious that such a step reduces the complexity for a clusterer. This is certainly the case for Graph Shift, which on our reference machine proved to take up to 14 seconds if applied without outlier removal. All the more astonishingly, the usage of an outlier removal step did not significantly influence the runtime of Iterative Replicator Dynamics, which is why it has only one curve in Figure 4.27.

The slowest algorithm in our evaluation was Kernel Fitting. We managed to speed up the algorithm considerably by caching structures in the model selection phase, but even so, it barely surpassed the slowest variant of Iterative Graph Shift. Note that said variant operates on the full affinity matrix, while Kernel Fitting uses the smaller affinity matrix that results from explicit gross outlier removal.

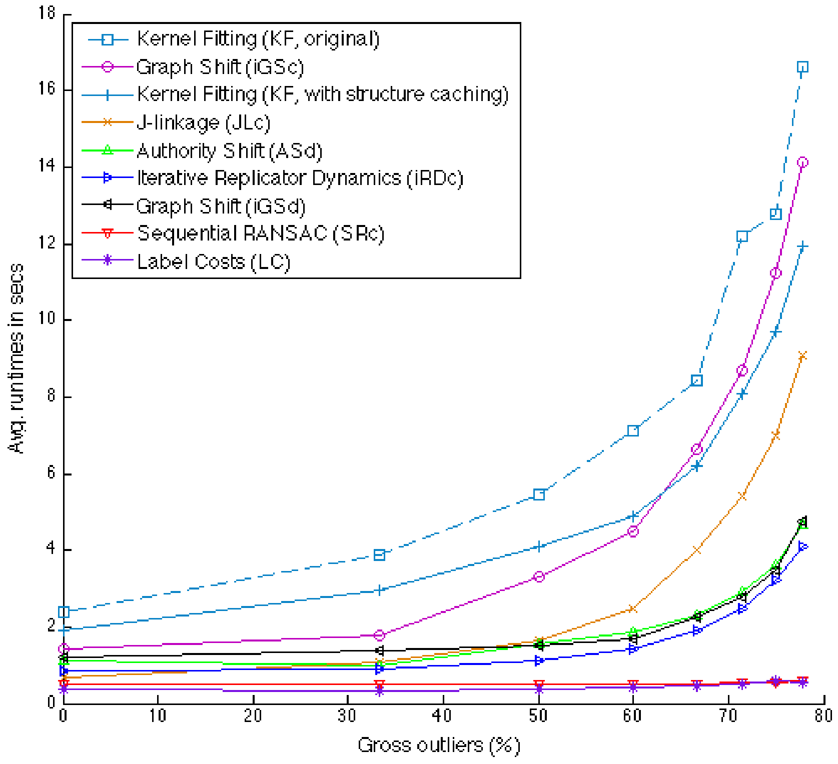


Figure 4.27: Average time in seconds required by the algorithms from Section 4.3.

In general, the runtimes of all algorithms based on clustering seem to grow quadratically with the number of gross outliers, i.e., the number of observations n. This complies with theoretical considerations: Building a J-linkage cluster tree takes $\mathcal{O}(n^2)$ [34], Replicator Dynamics has a complexity of $\mathcal{O}(t|E|)$, where t is the number of iterations of the algorithm and $|E|$ is the number of edges in the affinity graph, which in the worst case is $\Theta(n^2)$ [25]. On the other hand, approaches like Sequential RANSAC and Label Costs exhibit runtimes that are low and remain more or less constant. This is even more remarkable when one takes into account that these are the strategies that score highest in terms of fitting accuracy and cardinality estimation (see Sections 4.2 and 4.3).

5 Experiments on real world 3D data

After our line fitting experiments, we applied our model fitting framework to the problem of plane fitting in three dimensions. Instead of generating synthetic data, we opted for real world data from two publicly available RGB-D datasets. In both cases, the data consists of depth images recorded by a Microsoft Kinect© depth sensor.

In order to convert these depth data to three-dimensional data in euclidean space, the calibration of the color camera has to be known. For each pixel location (y, x) of the depth image D, the conversion to $P \in \mathbb{R}^3$ occurs as follows:

$$D(y,x) = z \quad \Longleftrightarrow \quad P = \frac{1}{f}\left[\frac{(x-c_x)z}{f_x}\ ,\ \frac{(x-c_y)z}{f_y}\ ,\ z\right]^T , \tag{5.1}$$

where (c_x, c_y) is the optical center and f_x and f_y are the focal lengths of the camera. f is a scale factor which enables the depth values to be stored as integers, for example in .png-files [18].

5.1 Experiments on the Freiburg depth dataset

The University of Freiburg provides a large dataset of image sequences with depth data, alongside camera trajectories which constitute a ground truth for SLAM (simultaneous localization and mapping). We use the depth data to generate point clouds as described above, on which we perform plane fitting. The dataset was presented in [35] and is available at https://vision.in.tum.de/data/datasets/rgbd-dataset.

5.1.1 Visualization

Drawing planes as an overlay of an RGB image is tricky. It is possible to draw transparent polygons representing said planes, but this requires the definition of four corner points. Therefore, we decided to visualize planes only by means of their inliers. As with our two-dimensional fitting results, we plot the inliers (or rather, the backprojection of these points onto the image plane) with different color labels, each color representing a different model.

Each Freiburg dataset consists of many consecutive image frames, but our fitting algorithms handle each frame as a separate set of observations. If we chose the label colors arbitrarily, label colors would change per frame, causing flickering effects. One solution for this problem would be to compute a mapping of labels from one frame to the next, but this approach could be prone to errors, and it does not disclose any information about the location and orientation of the plane, which the user would have to guess from the context of the background image. So instead, we translate the homogeneous representation of the plane to a color vector, thereby implicitly assigning different colors to different models, while simultaneously color-coding the orientation and distance of each plane.

From plane parameters n_1, n_2, n_3, d we construct a three dimensional HSV color $(h, s, v) \in [0, 1]^3$, where the v component constitutes the *brightness*, and h and s stand for *hue* and *saturation* [14]. We define the brightness as a function of the plane distance d, such that a plane close to the camera appears brighter than a parallel plane that is farther away. Since d can theoretically take any value in $[0, \infty)$, we use a function that is defined for all $d \geq 0$. Function $v(d)$ in Equation 5.2 is designed to be at its maximum at zero distance to the camera, and to slowly approximate zero for greater distances. Furthermore, it reaches brightness 1/2 at $d = 5$. Its course is plotted in Figure 5.1(a).

$$v(d) = \frac{1}{2^{\frac{1}{5}d}}. \tag{5.2}$$

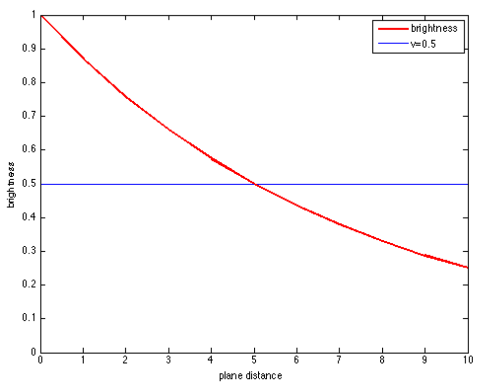
The h and s components are defined by the spherical coordinates of the vector $\mathbf{n} = (n_1, n_2, n_3)^T$. We assume that $||\mathbf{n}|| = 1$ and that $n_3 >= 0$. Then the spherical coordinates are

$$\varphi = atan2(n_2, n_1), \quad \theta = arccos(n_3), \quad r = 1. \tag{5.3}$$

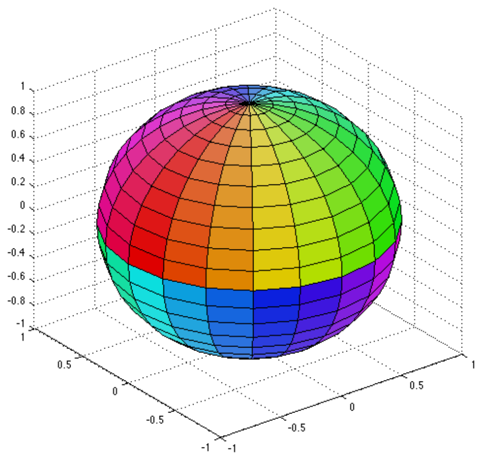
From this we go to color components by

$$h = \frac{\varphi}{2\pi}, \quad s = \frac{1}{2} + \frac{\theta}{\pi}, \tag{5.4}$$

which results in the color space seen in Figure 5.1(b). Note that s only assumes values between 1/2 and 1, such that all labels have nicely saturated colors.



(a) Brightness is defined as a function of the plane distance.



(b) Hue and saturation are defined by the spherical coordinates of the plane normal. Opposing normal vectors have the same color.

Figure 5.1: Color and brightness functions for the color coding of plane representations.

5.1.2 Determining the inlier scale

Before we can compare different algorithm setups in the three-dimensional application, we have to determine an inlier threshold τ. In a perfect world, we could simply use a very small threshold. One would expect, for example, that points on the same desk are all close to coplanar. However, errors in measurement and camera distortion can result in significant variation, which is why we determined the inlier threshold empirically. For the dataset freiburg1_360 we tried values between $\tau = 0.001$ (one millimeter) and $\tau = 0.05$ (five centimeters), for each of which we ran Sequential RANSAC with 500 hypotheses per iteration on the data. The results thereof can be seen in Figure 5.2. We decided that a threshold of $\tau = 0.01$ (one centimeter) is the best compromise between precision, i.e. the fraction of inliers that were correctly labeled, and recall, i.e. the fraction of points on a planar object that were recognized as inliers.

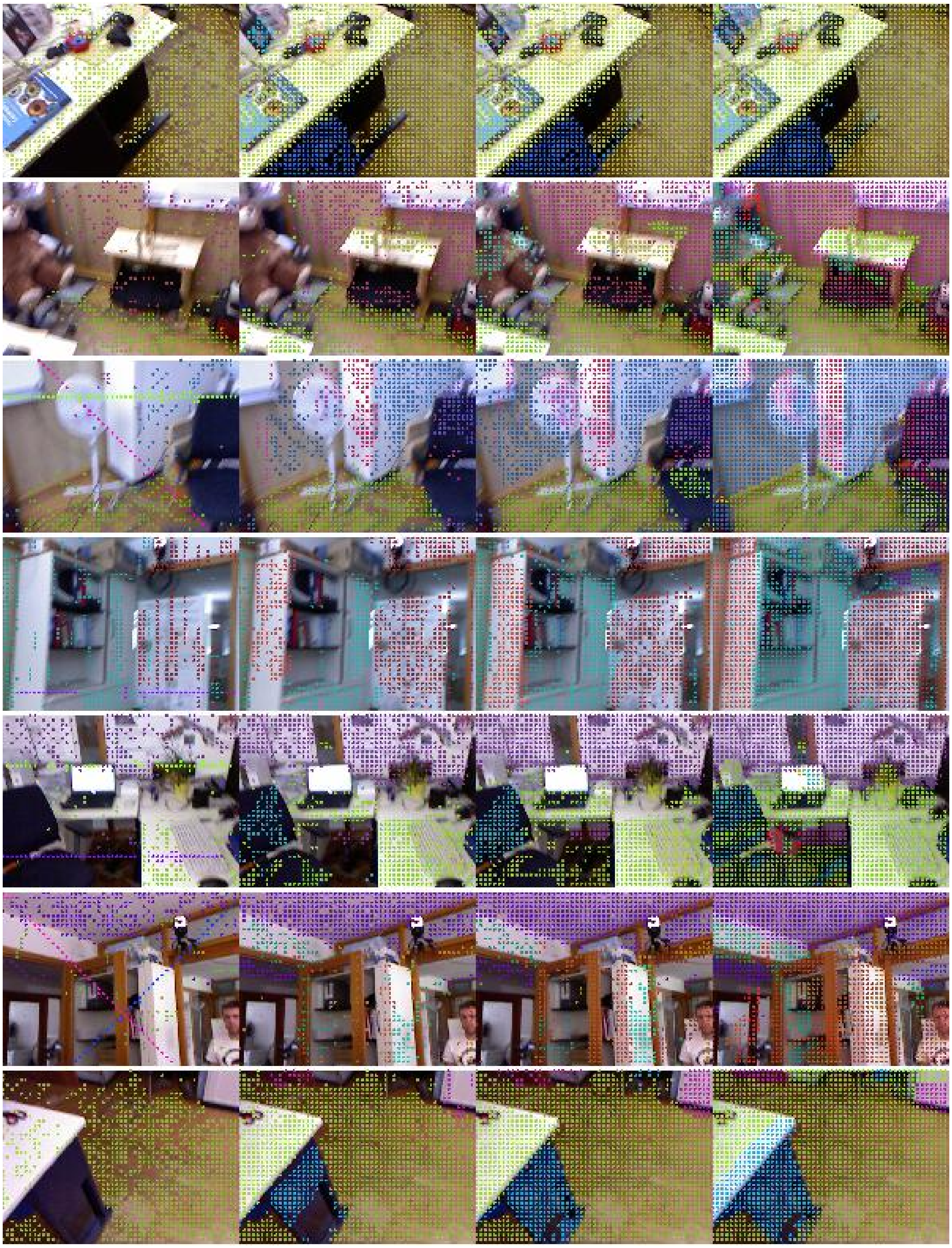


Figure 5.2: Testing different inlier thresholds with Sequential RANSAC (4 iterations). From left to right: $\tau = 0.001, 0.005, 0.01, 0.05$. The threshold in the leftmost column is too sharp to detect all relevant planes, and it introduces artifacts: when a hypothesis is constructed from a subset of which the *image* coordinates are collinear, then it forms a plane perpendicular to the camera. With a very low inlier threshold, these planes score higher than others, because a number of points is guaranteed to lie *on* the plane, and they are thus selected by the RANSAC algorithm. $\tau = 0.05$ is too imprecise, as we can see in the bottom right image, where a part of the desk surface is labeled incorrectly. $\tau = 0.005$ and $\tau = 0.01$ (center columns) both seem to be legitimate choices.

5.1.3 Plane fitting with Iterative Graph Shift

As mentioned earlier, the Graph Shift main function returns very small clusters, a problem which is solved by adding a fixed number of k nearest neighbors, in terms of affinities, to each cluster. In the case of plane fitting, this causes the following problem: If k is too large, small plane segments might be fitted inaccurately because too many neighbors are added. If k is too small, the iterative clustering procedure will visit the same plane many times, since it is only discouraged to move away from a small portion of the plane (see Section 3.2.6).

Therefore, we introduced a slight alteration of the graph shift code: Instead of returning a cluster with a fixed amount of non-zero entries, we estimate the actual plane model corresponding to the cluster, and declare all inliers of the plane as cluster members (in case there is an explicit outlier removal step, this only concerns the observations left thereafter). This might be mathematically inelegant, but it solves the problem of the clustering algorithm being "trapped" around one large structure.

We set the number of iterations of the meta-clusterer to 10 (resulting in ten models at most), and did not install a coherency threshold or minimal cluster size. The number of nodes for the graph shift expansion phase was fixed at $k = 100$. We doubled the number of hypotheses to 1000. The experiment was repeated with gross outlier removal prior to the clustering, as seen in Figure 5.4. Here, the outlier removal step seems less effective than with synthetic two-dimensional data, usually discarding no more than one third of the data. This might be explained by the fact that the outlier removal procedure models the histogram of norms used for the dichotomy by a Gaussian mixture model [4], a method that probably performs best on data with artificial Gaussian noise such as our 2D data. However, one could also argue that in the indoor scenes on which we apply plane fitting, most of the depth data come from objects such as floors, ceilings, doors and desks, and thus contain a naturally high percentage of inliers.

5.1.4 Label smoothing

So far, the decision which label is given to an observation was made by the model fitting strategy. Under Sequential RANSAC and J-linkage, each observation is assigned to one model at most. With the Iterative Graph Clustering method, on the other hand, observations can be assigned to multiple clusters, and we select the final label as the model for which the observation has the highest probabilistic cluster membership. These assignments, though not incorrect in terms of the fitting problem, can seem wrong to the human eye, as one may see in Figure 5.3(a). Therefore, we run a simple relabeling function before visualizing the results: If a point $\mathbf{x}^j$ with label i_1 is also an inlier of model i_2, and the neighbors of the point in the image plane are predominantly labeled i_2, then the label of $\mathbf{x}^j$ is changed to i_2. This generates a more consistent, smoother labeling, as seen in Figure 5.3(b).

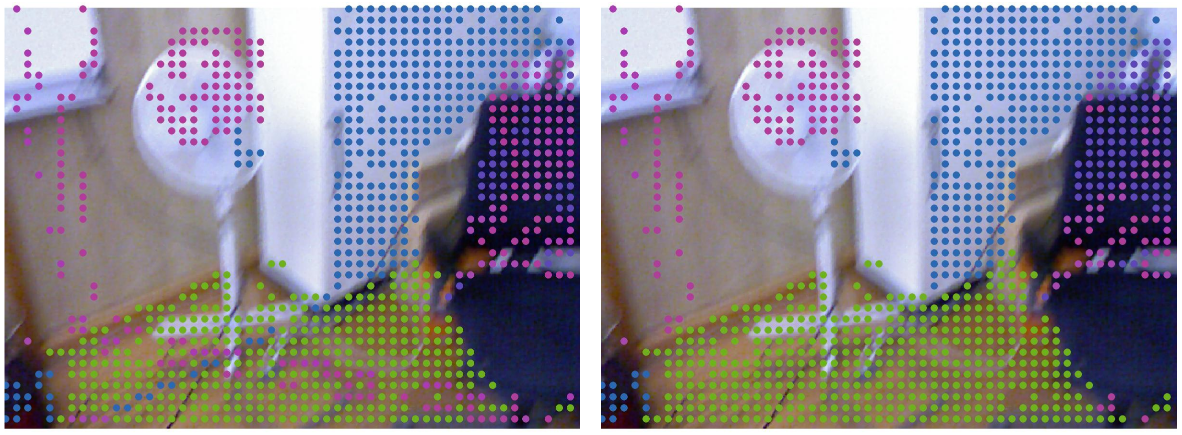

(a) Plane fitting results without label smoothing. (b) Plane fitting results with label smoothing.

Figure 5.3: Smoothing the labels of the fitting results brings about a more consistent result.

5.1.5 Summary

Besides the two Graph Shift variants, we ran the stand-alone version of Authority Shift on the Freiburg dataset. The walk/jump parameter was set to $\alpha = 0.2$ as in earlier experiments. Judging from the visual output of Figure 5.4, the performance of the algorithms is comparable, with the exception of Sequential RANSAC, which seems to identify more significant planes than the graph clusterers. Be that as it may, it is hard to come to an objective conclusion about the quality of the algorithms based solely on visual output, which is why we will attempt to carry out an objective evaluation on depth data in Section 5.2.

5.2 Experiments on RGB-D data from the NYU depth dataset

While the Freiburg dataset allows for nice visualizations of the discussed algorithm's power, it does not provide a ground truth for plane fitting. Therefore, we also ran experiments on the NYU depth dataset[1], which consists of 1449 non-sequential RGB-D images from indoor surroundings and a full ground truth for semantic segmentation [28]. We use these annotated data to construct a ground truth by identifying planar regions such as floors walls, desks, doors, etc. and extracting planes from them.

[1]The dataset can be downloaded at http://cs.nyu.edu/~silberman/datasets/nyu_depth_v2.html.

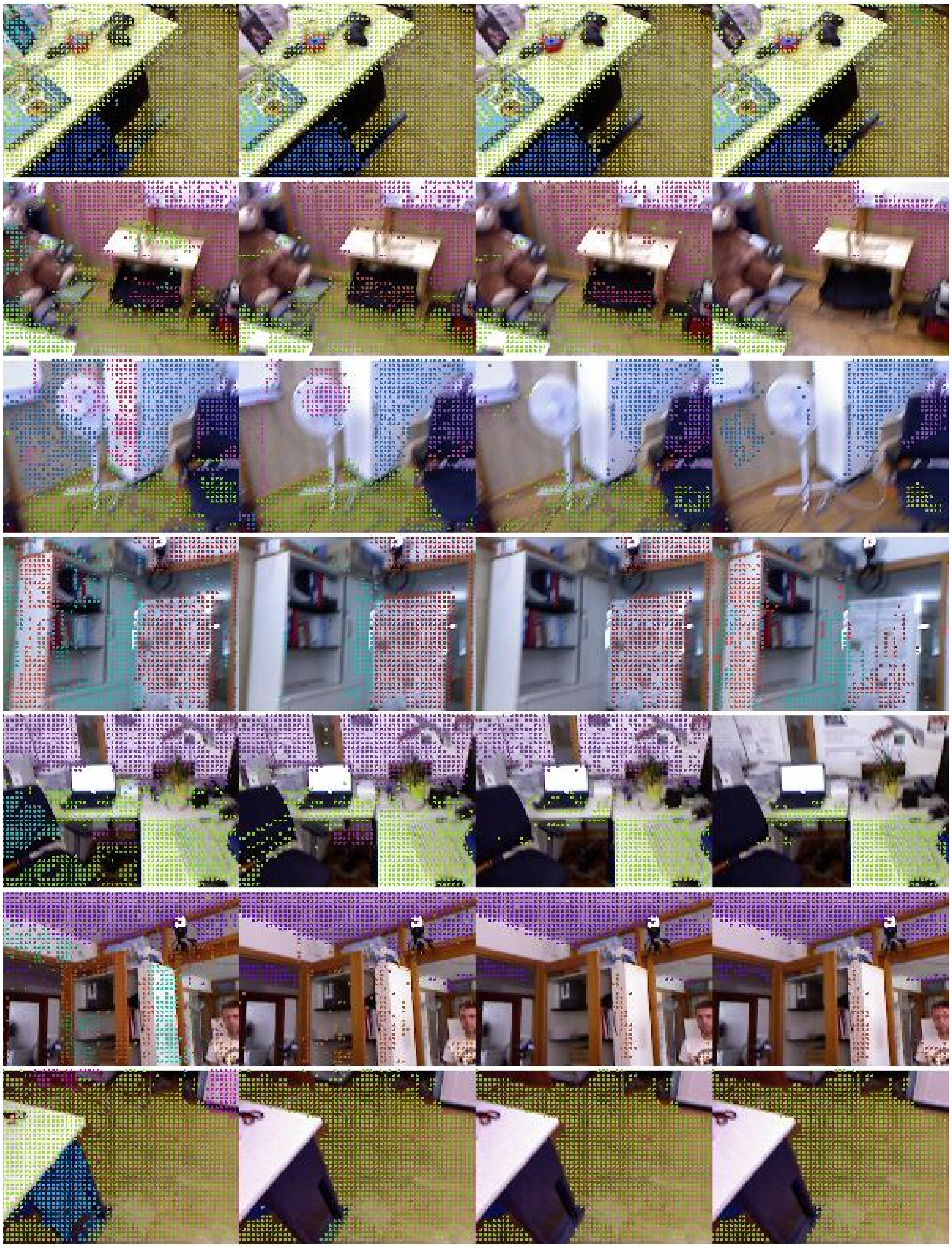


Figure 5.4: Comparison of four algorithms on the Freiburg depth dataset. From left to right: Sequential RANSAC, Iterative Graph Shift, Iterative Graph Shift with Outlier Removal, and stand-alone Authority Shift.

5.2.1 Construction of the ground truth

We construct a ground truth for plane fitting by fitting a single plane to each object or region in the dataset. Every pixel in an image has two associated labels: A *class* label identifying the object category, ranging from “air conditioner” to “yoga mat”, and an *instance* label, which distinguishes separate objects of the same class. Together, these two labels identify the pixels belonging to a single object or region. From these pixels, we construct a point cloud as described in Equation 5.1, and apply RANSAC to fit a plane.

Note that we do not use every pixel, but rather sample the images at a grid size of 10, effectively reducing the amount of data by a factor of 100, as in Section 5.1. We discard objects which consist of less than 100 points automatically. Having applied RANSAC to a point cloud, we count the number of inliers, defined by a fixed inlier threshold of 0.05. If a plane yields more than 300 inliers, it is accepted to our ground truth. Thereby, we construct a set of planar objects, which we can see in Figure 5.5. Finally, planes of the same image that share more than 50% of their inliers are fused. The relatively high inlier threshold was chosen in order to retrieve high true inlier counts: Lower thresholds lead to low true inlier counts, for which the algorithms require more random hypotheses in order to find them (see Equation 5.6).

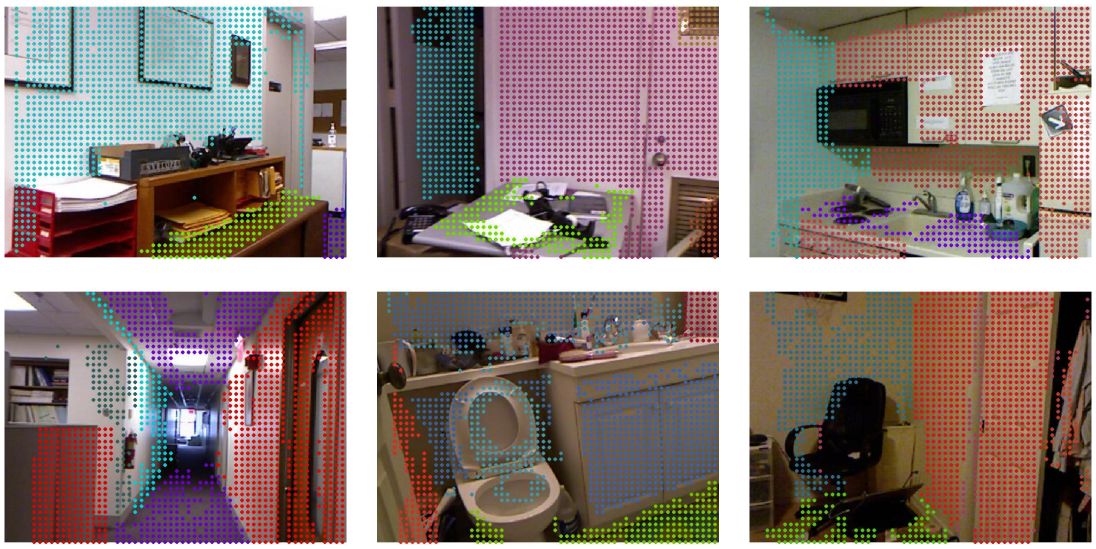


Figure 5.5: Examples of RGB images of the NYU depth dataset with planes constructed from planar regions (walls, ceilings, floors) in the depth data.

If our method does not return any planes for an image, it is discarded. In total, 1394 images were selected, with the number of structures per image varying between one and ten. Details are shown in Figure 5.6.

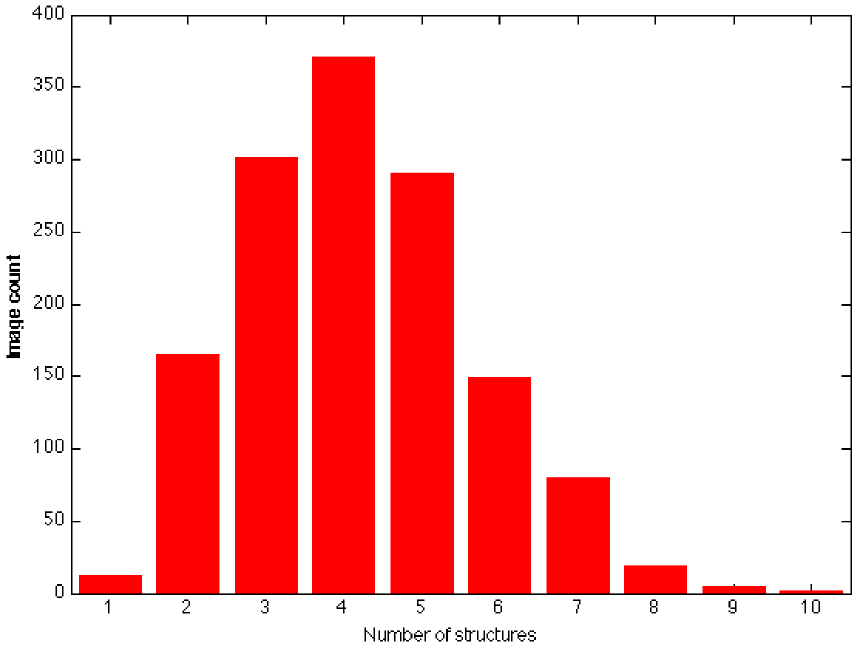
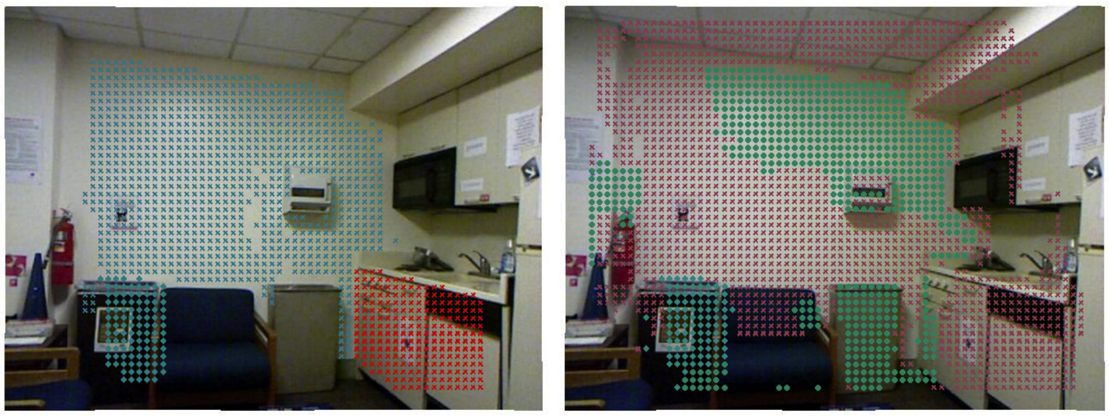


Figure 5.6: Numbers of planes found by our plane extraction procedure per image. 12 images contain only one significant plane, 165 images contain two, etc.

The resulting reference set is not a perfect ground truth: If a scene contains multiple objects that are aligned in a way such that many of their points lie on the same plane, but none of the objects alone produce a plane, then an algorithm that correctly identifies the plane is punished for fitting it. However, this kind of coincidence is also possible in synthetic 2D data: If randomly inserted gross outliers align to form a line by accident, the same problem occurs.

5.2.2 Evaluation

For the evaluation of fitting algorithms on our reference set of planes, we could again use the MSFE from Chapter 4. However, in practice the MSFE seems to be too biased towards the estimation of the *number* of structures, regardless of how well the estimated models fit. For synthetic line fitting problems, the MSFE was adequate because the only structures the algorithm *could* fit were those annotated in the ground truth. With the set of NYU reference planes, however, it is much more common that planes are fitted to gross outliers, while ground truth planes are missed. This leads to counter-intuitive values for the MSFE, as illustrated in Figure 5.7.



(a) ground truth (b) fitting result

Figure 5.7: Authority Shift achieves a MSFE of $\varepsilon = 0.16$ on this image, even though only one of four ground truth planes is correctly fitted. The low MSFE derives from the fact that the number of planes is estimated correctly.

Therefore, we employ precision and recall measures to evaluate our algorithm setups: A *true positive* is found if a reference plane and fitted model share more than 50% outliers, i.e.

$$TP(\hat{\theta}, \theta) = \begin{cases} 1 & \frac{|\hat{C} \cap C|}{|\hat{C} \cup C|} > 0.5 \\ 0 & \text{otherwise} \end{cases} \tag{5.5}$$

for structure $\hat{\theta}$ with consensus set $\hat{C}$ and model θ with consensus set C. We can make use of the J-affinity function from Section 2.7.4 to compute this value. We define *recall* in the usual way, as the number of true positives divided by the number of structures, and *precision* as the number of true positives divided by the number of models. When a reference plane is detected more than once, it only counts as one. Thereby we punish algorithms that produce a lot of double detections.

Data

Instead of running every setup on all 1394 sets in our ground truth, which would take very long as the cost of computing affinities grows quadratically, we decided to use only every tenth image, resulting in a sequence of 140 images total.

Setup of Sequential RANSAC

We started evaluation by running our simplest algorithm, Sequential RANSAC. Naturally, the inlier threshold for the algorithm was the same as in the ground truth. The algorithm ran for ten iterations at most, and stopped whenever fewer than 100 inliers were found, because we know from our ground truth that no plane with fewer inliers is featured. Planes that share more than 50% inliers were fused. The number of hypotheses per iteration was set to 500.

Setup of Authority Shift

For Authority Shift, we chose a walk/jump parameter of $\alpha = 0.10$ after a few short trials with different values. The number of random hypotheses was increased to $H = 2500$, such that even those planes with only 300 inliers (smallest in the reference set) have an acceptable probability of being found (see also Equation 2.3):

$$Pr\{\text{At least one inlier hypothesis for the smallest plane}\} = 1 - (1 - \frac{300}{3072})^{2500} \approx 90\%. \tag{5.6}$$

As with Sequential RANSAC, planes were fused based on their shared consensus. Interestingly, performing cost-based model selection did *not* improve results. This might be explained by the fact that planes in our extraction process are also fused by consensus as a post-processing step before ending up in the ground truth: Similar methods might produce similar results.

Setup of Iterative Graph Shift

For Iterative Graph Shift, we kept the parameters at the same values as on the Freiburg dataset, with an additional simple plane fuser as before. We also activated explicit outlier removal before clustering, mainly to reduce the complexity of the affinity graph and thereby speeding up the clustering process.

Summary

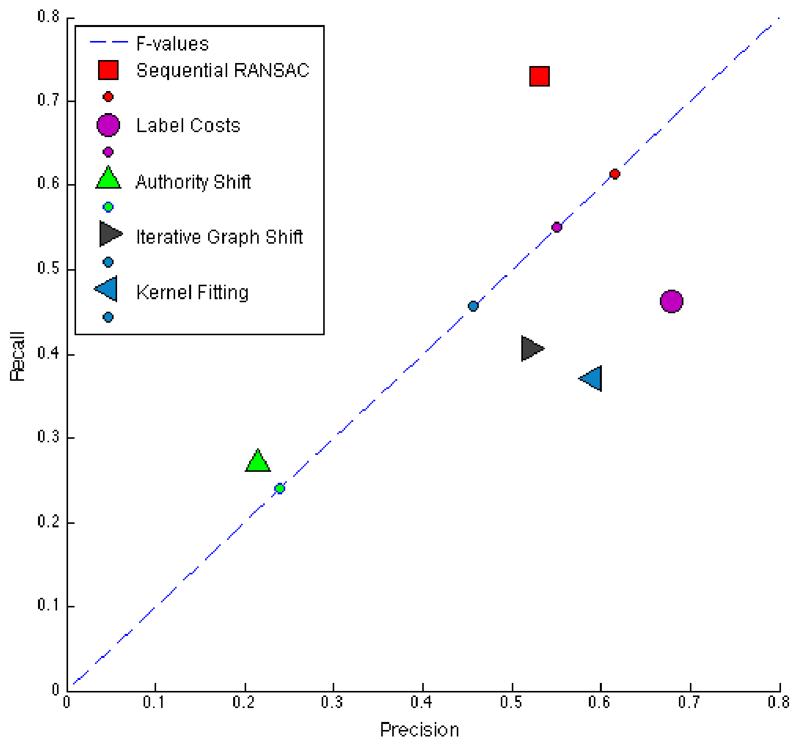


Figure 5.8: Average precision/recall pairs for different setups applied to our NYU reference planes. The corresponding F-Values are plotted on the diagonal.

Besides the three algorithms described in the previous paragraphs, we ran Kernel Fitting and Label Costs with their default setups. We visualize the fitting output on seven exemplary images in Figure 5.9. The results of the precision/recall evaluation are shown in Figure 5.8. The large markers indicate the average precision and recall values, P and R, for each algorithm, with the respective F-Values, $F = 2 \cdot P \cdot R/(P+R)$, are projected onto the diagonal. As one can see, Authority Shift performed worst by far, with a precision of 0.22 and a recall of 0.27. It is likely that the gross outlier removal, that worked so well on synthetic data, fails to remove non-uniformly distributed outliers. Since the version of Authority Shift that we use clusters *all* data rather than finding modes, it is unsuited for precise model fitting.

On the other hand, we see that our version of Graph Shift competes with state-of-the-art approaches, and even has a higher recall than the default setup of Kernel Fitting (0.41 vs. 0.37). As Kernel Fitting has a higher precision on its turn (0.52 vs. 0.59), both algorithms yield the same F-Value at 0.46.

Although precision was highest for Label Costs ($P = 0.67$), recall was highest for Sequential RANSAC ($R = 0.72$). In terms of the F-Value, Sequential RANSAC is the clear winner, with $F = 0.61$ vs. $F = 0.55$ for Label Costs. One could argue that the former has an unfair advantage because the ground truth planes were established by RANSAC as well, but it is still astonishing that such a simple and efficient algorithm yields such superior results.

A word about runtimes

Even though we sampled data points from the depth image with a grid size of ten, reducing the size of the point cloud from about 300,000 to 3072, the algorithms that rely on the affinity matrix had a hard time coping with its complexity. On our reference machine, the Authority Shift setup recorded an average of 70 seconds for an image, Graph Shift required about 80 seconds. Computing the 3072-by-3072 affinity matrix was the component that took longest with about 35 seconds, but also the outlier removal step took 22 seconds, a substantial amount of time. Kernel Fitting was again slowest, with an astonishing average runtime of 180 seconds, i.e. three minutes. By comparison, Label Costs only ran for about five seconds, and Sequential RANSAC was fastest with about half a second per image.

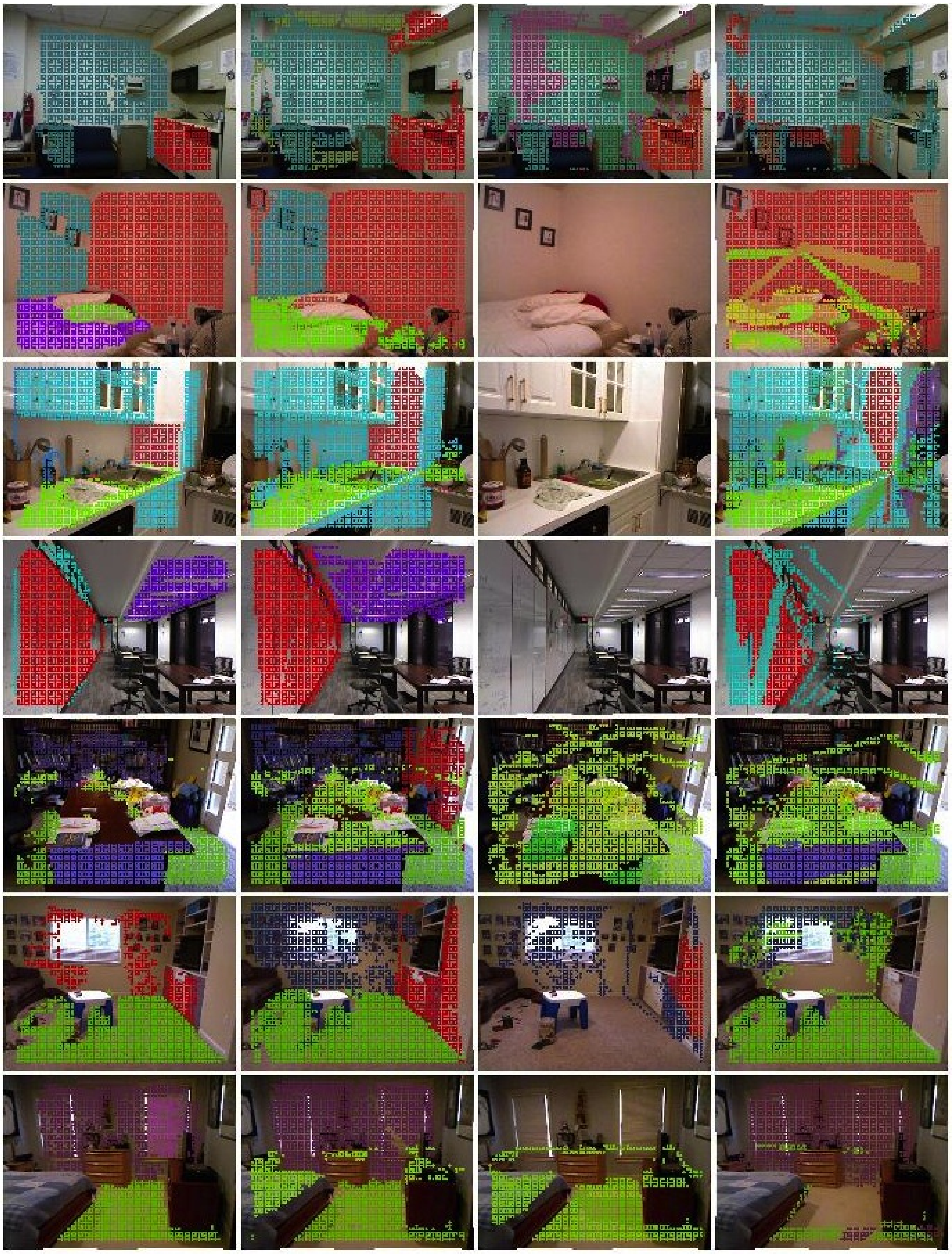


Figure 5.9: Examples of planes fitted by different algorithms. From left to right: ground truth, Label Costs, Authority Shift, and Iterative Graph Shift. One can see that Authority Shift fails to establish *any* models in some images. On the other hand, Label Costs produces very plausible sets of models.

6 Conclusion

The goals of this thesis were threefold: the first goal was a survey of existing methods for the problem of robust multiple model fitting, and their generalization into abstract model fitting categories. The most prominent of these categories is the group of data affinity clusterers, which produce models from clusters which are found in a space defined by the affinity kernel. Examples of this category are J-linkage, Kernel Fitting and Iterative Graph Clustering. The second goal of this thesis was to translate this abstraction to a modular model fitting framework, implemented in MATLAB. The third and final goal was the exploration of new combinations of model fitting setups within the implemented framework, as well as their evaluation and comparison to state-of-the-art approaches.

For the first goal, we started out with the discussion of RANSAC, the mother of robust model fitting techniques. Even though it is only suited for the identification of a single model, its principal of selecting a winning model from randomly sampled hypotheses forms a quintessential basis for many existing approaches in the field of robust model fitting. From RANSAC, we moved on to Sequential RANSAC, the most obvious extension of the algorithm that handles multiple structures. In its simplest form, it requires the definition of the number of iterations, which determines the *cardinality* of the resulting set of models. However, we showed that this stopping criterion can be replaced with criteria that measure the number of inliers or the total model cost.

In the clustering category, the first approach we investigated was J-linkage, presented by Toldo & Fusiello in 2010. Instead of iteratively creating new sets of hypotheses, it generates them in one batch, after which it clusters data points represented by their preference set, i.e. the set of hypotheses for which a data point is an inlier. Each cluster is turned into a model by solving the overdetermined set of equations defined by the cluster's members. The J-linkage technique inherently returns a variable number of models. However, it still requires the definition of a minimal cluster size, which implicitly regulates the number of returned models.

After J-linkage, we explored the subcategory of graph clusterers, methods that identify dense subsets of the affinity graph. By applying these clusterers iteratively, multiple clusters and thus models can be identified, either by randomly changing the starting nodes of the clusterer, or by altering the affinity matrix after each iteration. As a stopping criterion, one can threshold the cluster coherency, i.e. the sum over all intra-cluster affinities.

The most elaborate approach based on data affinity clustering that we surveyed was Kernel Fitting, proposed by Chin et al. in 2009, which incorporates a large spectrum of components including scale estimation, explicit gross outlier removal, and cost-based model selection. For the computation of affinities, the Ordered Residual Kernel is used, for which no inlier threshold has to be defined. The only parameter that has to be tuned is the step size of the kernel, which regulates how many of an observation's best fitting hypotheses are deemed important.

Other approaches such as Dynamic Structure Fitting, which alternates random sampling and hypothesis ranking, and AKSWH, which determines all its internal parameters by techniques from information theory, were also examined. In general, the amount of parameters required by each algorithm varies greatly. However, we can conclude that it is impossible to define a parameterless algorithm that fits multiple models exactly to our requirements: After all, the choice of granularity and thereby the inlier noise scale is always subjective (if my desk is a plane, then what about my keyboard? does it belong to the same plane?), and hence cannot be determined perfectly by a data-driven method.

With the knowledge we acquired during the realization of the first goal, we implemented a model fitting framework, which due to its modular design made it very easy to exchange, discard and enhance single components of different model fitting approaches. This framework constituted the second goal of this thesis. For one part, it was built with publicly available libraries, such as that of J-linkage, Graph Shift and Authority Shift, and especially Kernel Fitting, which contributed several functions related to outlier removal and cost-based model selection. For another part, it consists of source code written by the author of this thesis. These "home-made" elements include wrappers for library functions and the control flow of generic model fitting strategies, and also functions for data generation and algorithms, for example RANSAC and LMedS, and different functions for the purpose of evaluation. The framework also provides a basis for countless thinkable extensions, e.g. new applications such as multiple homography estimation, but also different methods for hypothesis generation, scale selection, or model estimation.

With the help of the model fitting framework described above, we were able to try out and evaluate many different variants of model fitting setups, both old and new. We started with the simple problem of line fitting on synthetic data. By varying the number of gross outliers in each generated dataset, we were able to test the algorithms' robustness towards these outliers. First, only the accuracy of the fits themselves were analyzed, by passing information about the number of structures to the algorithms. In a second run, we also considered the performance of different setups in terms of cardinality estimation, i.e. how well the number of structures in the ground truth was matched. An early conclusion was that the simplest approach, namely Sequential RANSAC, still performed best on these data. Moreover, we created a faster variant of Kernel Fitting by replacing the Ordered Residual Kernel with the J-distance kernel and running Graph Shift instead of a spectral method, which retained comparable results to original Kernel Fitting. On the other hand, the opposite tactic—iteratively applying graph clustering methods to the Ordered Residual Kernel—did not yield interesting results.

Finally, we applied a selection of algorithms to the problem of plane fitting in three dimensions, using data from the Freiburg dataset of RGB-D information recorded by Microsoft Kinect© sensors. We visualized the resulting models by labeling inliers and color coding the planes' normal vectors. We also proposed a method for establishing a plane fitting ground truth from semantically labeled depth data originating from the NYU dataset, by applying RANSAC with a fixed threshold to labeled object instances and estimating planes from those objects that had enough inlier support. Making use of this ground truth, we compared the detection powers of a handful of algorithms.

The remarkable conclusion that we can draw from our plane fitting experiments is that the fastest algorithms are also the best in terms of the error functions that we evaluated. The main argument in favor of graph clustering techniques was the property that observations can be assigned to multiple models, which is not possible in methods such as Sequential RANSAC or J-linkage. However, we have seen that this flaw doesn't seem to undermine the overall quality of the fitting results, at least not for geometric structures such as lines and planes. It also seemed that no model fitting algorithm that relies on the computation of affinities between all observations can be deemed fit for very large sets of observations, due to the sheer amount of necessary computations. Even if we introduced an affinity measure that leads to very sparse affinity graphs, which would speed up graph clustering algorithms such as Replicator Dynamics, the construction of the affinity matrix alone would still require $n \cdot H + \binom{n}{2}$ computations for n points and H hypotheses. If we created 1000 hypotheses and used the J-distance kernel for 10000 points (not an unrealistic number for three-dimensional point clouds), the computation of all affinities would involve the comparison of $\binom{10000}{2} \cdot 1000 \approx 50$ billion bits. On the other hand, one should not forget that there exist applications where the number of observations is much smaller than that, such as point correspondences established from interest point descriptor matches for the purpose of homography estimation. this type of applications, both Kernel Fitting and J-linkage have demonstrated excellent results ([4], [15]).

All in all, we have seen many interesting developments in the field of clustering in general, and in the field of graph clustering in particular. The foundation we laid with our model fitting framework might enable future research in both these fields. With the combination of faster clusterers, smarter affinity kernels, and the ever growing computational power of today's machines, we expect countless promising developments in the domain of multiple model fitting to occur in the following years.

Bibliography

[1] Hirotugu Akaike. "A new look at the statistical model identification." In: *Automatic Control, IEEE Transactions on* 19.6 (1974), pp. 716–723 (cit. on p. 13).

[2] Dana H Ballard. "Generalizing the Hough transform to detect arbitrary shapes." In: *Pattern recognition* 13.2 (1981), pp. 111–122 (cit. on p. 10).

[3] Y. Boykov, O. Veksler, and R. Zabih. "Fast approximate energy minimization via graph cuts." In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 23.11 (2001), pp. 1222–1239 (cit. on p. 31).

[4] Tat-Jun Chin, Hanzi Wang, and D. Suter. "Robust fitting of multiple structures: The statistical learning approach." In: *Computer Vision, 2009 IEEE 12th International Conference on*. 2009, pp. 413–420 (cit. on pp. 4, 13, 15, 24, 25, 41, 43, 44, 46, 51, 53, 55, 59, 66, 74, 86).

[5] Tat-Jun Chin, Jin Yu, and David Suter. "Accelerated Hypothesis Generation for Multi-structure Robust Fitting." In: *Computer Vision – ECCV 2010*. Vol. 6315. 2010, pp. 533–546 (cit. on pp. 12, 41).

[6] Minsu Cho and Kyoung-Mu Lee. "Mode-seeking on graphs via random walks." In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. 2012, pp. 606–613 (cit. on p. 23).

[7] Minsu Cho and Kyoung MuLee. "Authority-shift clustering: Hierarchical clustering by authority seeking on graphs." In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. June 2010, pp. 3193 –3200 (cit. on pp. 22, 23, 45).

[8] Ondrej Chum and Jiri Matas. "Matching with PROSAC-progressive sample consensus." In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 1. IEEE. 2005, pp. 220–226 (cit. on p. 8).

[9] Dorin Comaniciu and Peter Meer. "Mean shift: A robust approach toward feature space analysis." In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 24.5 (2002), pp. 603–619 (cit. on pp. 4, 10, 13, 14).

[10] A. Delong et al. "Fast approximate energy minimization with label costs." In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. June 2010, pp. 2173–2180 (cit. on pp. 4–6, 30, 31, 40, 41, 62, 63).

[11] Cigdem Eroglu Erdem et al. “RANSAC-based training data selection for emotion recognition from spontaneous speech.” In: *Proceedings of the 3rd international workshop on Affective interaction in natural environments*. ACM. 2010, pp. 9–14 (cit. on p. 1).

[12] Martin A. Fischler and Robert C. Bolles. “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography.” In: *Commun. ACM* 24.6 (June 1981), pp. 381–395. ISSN: 0001-0782 (cit. on pp. 3, 7, 43).

[13] Andrew W Fitzgibbon. “Simultaneous linear estimation of multiple view geometry and lens distortion.” In: *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. Vol. 1. IEEE. 2001, pp. I–125 (cit. on p. 1).

[14] J.D. Foley. *Computer Graphics: Principles and Practice, Second Edition in C*. Addison-Wesley systems programming series. Addison Wesley Professional, 1996, p. 590 (cit. on p. 71).

[15] D.F. Fouhey, D. Scharstein, and A.J. Briggs. “Multiple Plane Detection in Image Pairs Using J-Linkage.” In: *Pattern Recognition (ICPR), 2010 20th International Conference on*. 2010, pp. 336–339 (cit. on pp. 15, 86).

[16] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Vol. 2. 2000 (cit. on pp. 1, 7).

[17] John Illingworth and Josef Kittler. “A survey of the Hough transform.” In: *Computer vision, graphics, and image processing* 44.1 (1988), pp. 87–116 (cit. on p. 9).

[18] *Intrinsic Camera Calibration of the Kinect*. Apr. 2013. URL: http://vision.in.tum.de/data/datasets/rgbd-dataset/file_formats#intrinsic_camera_calibration_of_the_kinect (cit. on p. 70).

[19] Hossam Isack and Yuri Boykov. “Energy-Based Geometric Multi-model Fitting.” In: *Int. J. Comput. Vision* 97.2 (Apr. 2012), pp. 123–147 (cit. on p. 41).

[20] A.K. Jain, R. P W Duin, and Jianchang Mao. “Statistical pattern recognition: a review.” In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22.1 (2000), pp. 4–37 (cit. on p. 13).

[21] Alfred A. Kuehn and Michael J. Hamburger. “A Heuristic Program for Locating Warehouses.” In: *Management Science* 9.4 (1963), pp. 643–666 (cit. on p. 31).

[22] Bastian Leibe, Ales Leonardis, and Bernt Schiele. “Combined object categorization and segmentation with an implicit shape model.” In: *Workshop on Statistical Learning in Computer Vision, ECCV*. 2004, pp. 17–32 (cit. on p. 12).

[23] Marius Leordeanu and Martial Hebert. “A Spectral Technique for Correspondence Problems Using Pairwise Constraints.” In: *Proceedings of the Tenth IEEE International Conference on Computer Vision - Volume 2*. ICCV ’05. 2005, pp. 1482–1489 (cit. on p. 16).

[24] Hairong Liu and Shuicheng Yan. “Common visual pattern discovery via spatially coherent correspondences.” In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. 2010, pp. 1609–1616 (cit. on pp. 4, 19).

[25] Hairong Liu and Shuicheng Yan. “Robust Graph Mode Seeking by Graph Shift.” In: *Machine Learning, 27th International Conference on*. 2010 (cit. on pp. 19, 20, 47, 66, 69).

[26] James MacQueen et al. “Some methods for classification and analysis of multivariate observations.” In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 281-297. California, USA. 1967, p. 14 (cit. on p. 13).

[27] DRMPHSTS Nasuto and JM Bishop R Craddock. “Napsac: High noise, high dimensional robust estimation-it’s in the bag.” In: (2002) (cit. on p. 8).

[28] Pushmeet Kohli Nathan Silberman Derek Hoiem and Rob Fergus. “Indoor Segmentation and Support Inference from RGBD Images.” In: *ECCV*. 2012 (cit. on pp. 5, 75).

[29] M. Pavan and M. Pelillo. “A new graph-theoretic approach to clustering and segmentation.” In: *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*. Vol. 1. 2003, (cit. on pp. 16, 18).

[30] P.J. Rousseeuw and A.M. LeRoy. *Robust Regression and Outlier Detection*. Wiley-Interscience, 2003, p. 14 (cit. on pp. 2, 26).

[31] Yoko Sasaki, Satoshi Kagami, and Hiroshi Mizoguchi. “Multiple sound source mapping for a mobile robot by self-motion triangulation.” In: *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE. 2006, pp. 380–385 (cit. on p. 1).

[32] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004 (cit. on p. 26).

[33] Jianbo Shi and J. Malik. “Normalized cuts and image segmentation.” In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22.8 (Aug. 2000), pp. 888 –905 (cit. on p. 26).

[34] Robin Sibson. “SLINK: an optimally efficient algorithm for the single-link cluster method.” In: *The Computer Journal* 16.1 (1973), pp. 30–34 (cit. on p. 69).

[35] J. Sturm et al. “A Benchmark for the Evaluation of RGB-D SLAM Systems.” In: *Proc. of the International Conference on Intelligent Robot Systems (IROS)*. Oct. 2012 (cit. on pp. 5, 70).

[36] Roberto Toldo and Andrea Fusiello. “Robust Multiple Structures Estimation with J-Linkage.” In: *Computer Vision – ECCV 2008*. Vol. 5302. 2008, pp. 537–547 (cit. on pp. 4, 13, 14, 41, 45, 57).

[37] Philip HS Torr. “Geometric motion segmentation and model selection.” In: *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 356.1740 (1998), pp. 1321–1340 (cit. on p. 9).

[38] Philip HS Torr and Andrew Zisserman. "MLESAC: A new robust estimator with application to estimating image geometry." In: *Computer Vision and Image Understanding* 78.1 (2000), pp. 138–156 (cit. on p. 8).

[39] P. H. S. Torr. "Bayesian Model Estimation and Selection for Epipolar Geometry and Generic Manifold Fitting." In: *Int. J. Comput. Vision* 50.1 (Oct. 2002), pp. 35–61 (cit. on pp. 13, 26).

[40] T. Vincent and R. Laganiere. "Detecting planar homographies in an image pair." In: *Image and Signal Processing and Analysis, 2001. ISPA 2001. Proceedings of the 2nd International Symposium on*. 2001, pp. 182–187 (cit. on p. 9).

[41] Hanzi Wang, Tat-Jun Chin, and D. Suter. "Simultaneously Fitting and Segmenting Multiple-Structure Data with Outliers." In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 34.6 (June 2012), pp. 1177 –1192 (cit. on pp. 1, 2, 11, 13, 29, 41).

[42] Hoi Sim Wong et al. "Dynamic and hierarchical multi-structure geometric model fitting." In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. Nov. 2011, pp. 1044 –1051 (cit. on p. 27).

[43] Lei Xu, Erkki Oja, and Pekka Kultanen. "A new curve detection method: randomized Hough transform (RHT)." In: *Pattern recognition letters* 11.5 (1990), pp. 331–338 (cit. on pp. 4, 10).

[44] Jingyu Yan and Marc Pollefeys. "A General Framework for Motion Segmentation: Independent, Articulated, Rigid, Non-rigid, Degenerate and Non-degenerate." In: *Computer Vision – ECCV 2006*. Vol. 3954. Lecture Notes in Computer Science. 2006, pp. 94–106 (cit. on p. 1).

[45] Wei Zhang and Jana Kosecká. "Nonparametric estimation of multiple structures with outliers." In: *Dynamical Vision*. Springer, 2007, pp. 60–74 (cit. on pp. 10, 11).

[46] M. Zuliani, C.S. Kenney, and B.S. Manjunath. "The multiRANSAC algorithm and its application to detect planar homographies." In: *Image Processing, 2005. ICIP 2005. IEEE International Conference on*. Vol. 3. 2005, (cit. on pp. 1, 9, 11).