MULTI-MODEL ESTIMATION IN THE PRESENCE OF OUTLIERS

David F. Fouhey

Adviser: Daniel Scharstein

A Thesis

Presented to the Faculty of the Computer Science Department

of Middlebury College

in Partial Fulfillment of the Requirements for the Degree of

Bachelor of Arts

May 2011

ABSTRACT

The estimation of models or structures from outlier-contaminated data containing multiple models has a large number of applications in computer vision, the study of the automated understanding of visual data: for instance, geometric figures may be detected from 2D points, and planar surfaces in a scene may be found in pairs of images of the scene using feature matches. This thesis describes a number of contemporary algorithms for multi-model estimation and some of their historical antecedents, as well as an evaluation methodology for the multi-model estimation problem.

ACKNOWLEDGEMENTS

This thesis would not exist without the patience and assistance of many other people over the past few years.

I would especially like to individually thank Daniel Scharstein for his many years and many hours of hard work, great advice, and encouragement and Amy Briggs for her support and guidance over the past four years. As a pair, I would also like to thank them for their supervision over two summers that directly led me to my pursuit of research and discovery of computer vision. I would also like to thank Matthew Dickerson and Timothy Huang for their seemingly boundless enthusiasm for teaching, as well as their patience and many helpful pieces of advice. Finally, I would like to thank Bob Grossblatt for getting me fascinated with computer programming in the first place.

I would like to thank Justine for her support, encouragement and patience, as well as my friends for their understanding. Finally, I cannot thank my parents enough for many years of love and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

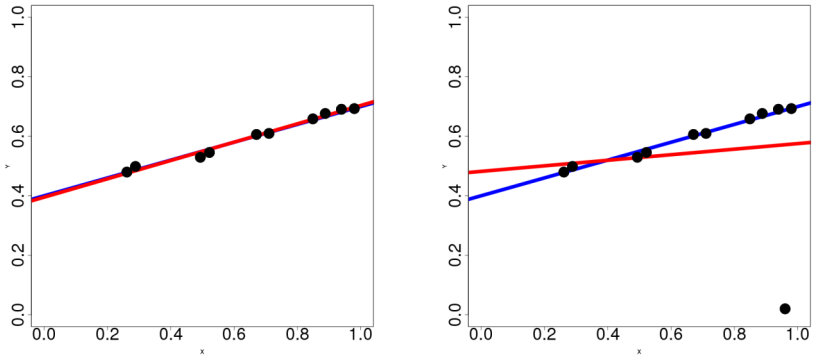
LIST OF FIGURES

CHAPTER 1

INTRODUCTION: LEAST SQUARES AND ITS DISCONTENTS

Given a class of model (e.g., a line or circle) and a set of data points contaminated by noise, one can often find a "best-fit" model. In many situations in the physical and social sciences, we wish to find a function $f(x)$ that best predicts the value of $y = f(x)$ given a value x. Commonly, given some data points $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, we try to minimize the sum of squared *residuals*, or deviations from the prediction of the model. In many cases, we may naturally define the residual as the quantity $|f(x_i) - y_i|$, as $f(x_i)$ is the prediction of the model and y_i is the actual value. The model-estimation technique that minimizes the sum of squared residuals is commonly known as ordinary least-squares as it produces the model that has the least sum of squares of the residuals out of all the possible models. Using such techniques, we can see through the noise in the data to get a good sense of the underlying generative process. However, although techniques such as least-squares are successful at mitigating the impact of noise on model fitting, they are likely to produce inaccurate model estimates in the presence of outliers, points that do not fit a model, or when there are multiple instances of models.

As an example, we present the task of fitting lines to four sets of data points in the Cartesian plane. In this case, ordinary least-squares, with which the reader is most likely to be familiar, minimizes the distance between the points and the line in only the y dimension, and is thus unsuitable. Instead, we use a technique that minimizes the sum of squared distance between the points and the fitted line. In general, we could solve such a problem using a technique called Total Least Squares [30]; however, in the specific case of two-dimensional points and lines, we use a technique called Principal Components Analysis (PCA), which will be more



(a) One model, no outliers

(b) One model, one outlier

(c) Two models, no outliers

(d) Two models one outlier

Figure 1.1: The impact of outliers and multiple models on PCA. In each diagram, the ground-truth models are plotted in blue and the estimated model is plotted in red.

thoroughly explained in Section 2.2. Under either name, the technique is a least-squares approach, as its criterion for model selection is that the model minimizes the sum of squared residuals. In Fig. 1.1, we use PCA on four data sets containing one or more lines.

Each data set in Fig. 1.1 contains either one model or two, and either no outliers or one. We first created the *ground-truth* models, or the underlying models that represent the "actual" parameters of the lines, and then placed points on each line; the points falling on these lines, and thus matching the models, are called *inliers*. We then perturbed the points using zero-mean Gaussian noise with

moderate magnitude ($\sigma = 1\%$), and in two cases, added an *outlier*. We show the ground-truth models in blue, and the model that PCA finds in red. Evaluating PCA's performance here is simple: if the red model matches a blue model, then it has succeeded; if not, it has failed.

PCA succeeds at detecting a model when there is only one model and no outlier, as in Fig. 1.1(a), and fails in every other case. The underlying cause is that PCA, like every other least-squares technique, finds a model that minimizes the squared residuals for *every* data point. When the data points arise from only one process or underlying model, this is a satisfactory goal and PCA succeeds; however, when points not arising from the same process are introduced, in either the form of another model or outliers, PCA may fail: in Fig. 1.1(b), 1.1(c), and 1.1(d), the line has to satisfy the mutually conflicting goals of minimizing the inliers' residuals and minimizing the residuals of an outlier, the residuals of the inliers of a second model, or both. The natural question, which is the central question of this thesis, is then the following: *can an algorithm be formulated for detecting multiple models in noisy and outlier contaminated data?*

Two observations about the data we have just presented drive our presentation of the answer to this question.

The first is that the similarity in failure of both the multiple model and outlier cases suggests a natural intermediate step to the end goal: inliers to one model are (unless the models intersect) outliers to another model. Stewart, recognizing this in [28], referred to such points as *pseudo-outliers*, points that do not match a model because they belong to another model. Traditional outliers, or points that do not match any model, are referred to as *gross outliers*. An algorithm that is capable of correctly detecting models in the presence of outliers of either kind would have a better chance of leading to a solution to our overarching goal:

Table 1.1: The data points of Fig. 1.1(d) presented in tabular form.

X, Y	X, Y	X, Y
(0.17, 0.29)	(0.14, 0.31)	(0.49, 0.53)
(0.10, 0.82)	(0.17, 0.08)	(0.94, 0.69)
(0.85, 0.66)	(0.52, 0.55)	(0.14, 0.65)
(0.15, 0.45)	(0.96, 0.02)	(0.29, 0.50)
(0.26, 0.48)	(0.71, 0.61)	(0.10, 0.96)
(0.17, 0.29)	(0.89, 0.68)	(0.98, 0.69)
(0.11, 0.83)	(0.09, 0.91)	(0.67, 0.61)

finding a ground-truth model correctly is much better than finding a model that fits neither ground-truth model. Thus, an outlier-robust single-model estimation technique is a crucial first step.

The second observation becomes apparent when one compares Table 1.1 with Fig. 1.1(d). The models are immediately apparent when looking at the plot; however, when presented with pairs of numbers, it is exceedingly difficult to recognize anything. The ease with which humans can see the models in the plots suggests a connection to computer vision, the study of making automated inferences about visual data. The structure in the data is immediately obvious without the mathematical notion of a line or the precise measurements of the locations of the data points; the detection is handled immediately and effortlessly by the human visual system. How the visual system accomplishes this is not the subject of this thesis. Instead, that it is accomplished not only gives us hope that algorithms exist but also a cue that they have applications for understanding visual data.

In this thesis, we describe the development of contemporary solutions to the task of estimating multiple models from noisy and outlier-contaminated data and their role in computer vision. To limit our scope, we focus our discussion on general algorithms, which may be readily applied to multiple domains with minimal adjustment, rather than model-specific estimation techniques. Accordingly, tech-

niques that are not generally applicable are not included. For instance, although there is much work on plane detection as a particular instance of multi-model estimation, plane-specific algorithms are not discussed. Similarly, since algorithms based on the Hough Transform (e.g., [42, 14]) are intractable for models with more than a few parameters, they are not included.

Our presentation begins with a discussion of techniques for estimating single models in outlier-free data in Chapter 2. Although they are not robust to outliers, such techniques serve as the building blocks for outlier-robust techniques and provide a way to introduce the model classes that we will discuss later in this thesis. In Chapter 3, we discuss a classic outlier-robust single-model estimation algorithm, Random Sample Consensus (RANSAC) (1981) [15]. RANSAC is used in or inspires a number of early multi-model estimation techniques, including Sequential RANSAC and MultiRANSAC (2005) [44]; other techniques, such as Mean Shift (2002) [11], Residual Histogram Analysis (RHA) (2006) [43], J-Linkage (2008) [32], Kernel Fitting (2009) [9], and PEARL (2010) [13] formulate their approach to multi-model estimation in alternative ways. We discuss both the RANSAC-oriented and some of the non-RANSAC-oriented techniques in Chapter 4.

Throughout the process, we will discuss a number of contexts in which the task of estimating multiple models from outlier-contaminated data appears. In particular, we will focus on the detection of geometric figures (e.g., lines and circles) from data sets of 2D points and the estimation of planar surfaces from correspondences, or pairs of 2D points from pairs of images of a scene seen at two different views that represent the same physical 3D location. A great number of other applications exist. For instance, in motion segmentation, motions (e.g., of a car and a person) must be estimated from the movement of individual points; since the data contains noise and may contain outliers, a multi-model estimation algorithm is a

sensible approach [29, 8]. Similarly, just as one can estimate lines from 2D point sets, planar surfaces may be estimated from 3D point clouds [32, 34] as well as the ones implicitly present in stereo depth maps [17].

The myriad of approaches and applications naturally leads to the question of how we can compare the performance of multi-model estimation techniques, perhaps to select the best one for a particular application, and what conclusions we can draw about some of the approaches described? We introduce an approach for multi-model estimation performance evaluation, and describe some results in Chapter 5. Finally, in Chapter 6, we draw conclusions regarding the present state and future directions of multi-model estimation algorithms.

CHAPTER 2

OUTLIER-FREE SINGLE MODEL ESTIMATION

Although we are interested in techniques capable of estimating models in the presence of outliers, these outlier-robust techniques make use of past techniques for estimation that are not robust. In general, these earlier techniques fall into two categories: exact solutions, which fit the data points with no error, and estimates that minimize an error metric. Included in the latter category is what is commonly referred to as a least-squares estimate. As both are used in outlier-robust estimation, we will naturally present both kinds of estimators for a number of classes of models, although we will focus primarily on the estimates as they are more difficult to derive.

We begin with the linear regression model commonly used to introduce and motivate the least-squares approach in linear algebra classes. We then move on to models more common in vision-related tasks, such as lines and circles, which we fit with least squares solutions as well. We finish with a discussion of the least-squares fitting of homographies, linear transformations that describe the change in appearance of a planar surface under a viewpoint shift.

Throughout this chapter, we will make heavy use of basic linear algebra. Readers having familiarity with matrices as systems of linear equations and basic operations on them, along with vectors, including their addition and lengths, should be comfortable. As is usually the case, mentally performing the operations in the equations presented is enormously helpful in understanding them. We will denote matrices in upper case bold (e.g., $\mathbf{X}$), and vectors in lower case bold (e.g., $\mathbf{u}$), and the ith entry of a vector $\mathbf{u}$ as $\mathbf{u}_i$. We primarily use column-vectors; in the text, where the typesetting of a column vector would be awkward, we denote the 3×1 vector with entries x, y and z as $[\ x \quad y \quad z\ ]^T$, or the row-vector transposed.

2.1 Linear Regression

Suppose we have data points, as in the introduction, $X = \{(x_1, y_1), \ldots, (x_n, y_n)\} \subset \mathbb{R}^2$, and we wish to fit a linear function $y = ax + b$ to them. For instance, adapting an example from [1], we might wish to find a model for how rapidly crickets chirp as a linear function of temperature. Our data might then consist of temperatures as x values and chirps per second as y values. If we only have two data points, we can find an exact model using only middle-school mathematics: the slope, a, of the line is given by

$$a = \frac{y_2 - y_1}{x_2 - x_1} \tag{2.1}$$

and its y-intercept, b, is given by

$$b = y_1 - ax_1. \tag{2.2}$$

If we have more than two data points, although it is possible that a solution will fit every data point exactly, it is overwhelmingly likely that it will be impossible to find such an exact solution. For instance, consider the data in Fig. 2.1: no line goes through every data point, although a linear trend is apparent in the data. Instead of finding a perfect fit (which is impossible), we must then seek the next best option, a line that is the best, or least-"bad" fit according to some metric.

We must define what "bad" is. In our case, as we want a model to predict the number of cricket chirps per second given the temperature, we want something that quantifies how far off the prediction is. The natural approach is the y-residual: given a data point (x_i, y_i) and a linear model $y(x) = ax + b$, the y-residual is

$$|y_i - y(x_i)| = |y_i - ax_i + b|. \tag{2.3}$$

For mathematical convenience that will become apparent later on, we will take the square of this, or $(y_i - ax_i + b)^2$ [22]. To extend this residual over all of our data

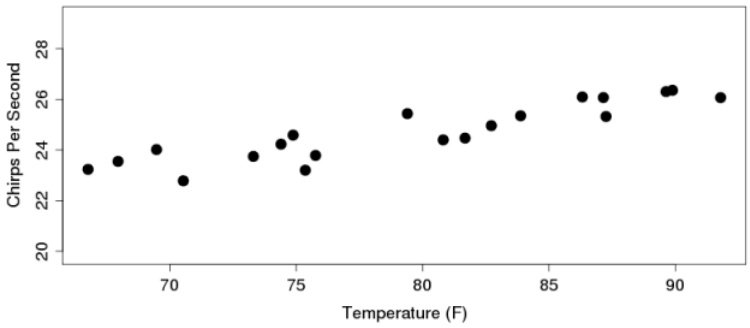


Figure 2.1: Cricket chirps vs. temperature (F) data

points, we simply take the sum, yielding the sum of squared residuals

$$SSR = \sum_{i=1}^{n} (y_i - ax_i + b)^2. \tag{2.4}$$

A best-fitting model, if we use the y-residual, is then given by the model parameters a and b such that Equation 2.4 is minimized.

It turns out that we can find such a model using linear algebra, but that we must first express our problem in a satisfactory way; in particular, we must pose our data fitting problem as finding a solution to a system of linear equations. If we construct a $n \times 2$ matrix $\mathbf{X}$ and n-dimensional vector $\mathbf{y}$,

$$\mathbf{X} = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{2.5}$$

we can then express a model $y = ax + b$ as a parameter vector $\mathbf{p} = [\ a \quad b\ ]^T$. The predictions of the model $\mathbf{p}$ on the data points $\mathbf{X}$ are then given by the product of

$\mathbf{X}$ and $\mathbf{p}$

$$\mathbf{Xp} = \begin{bmatrix} x_1a + 1b \\ x_2a + 1b \\ \vdots \\ x_na + 1b \end{bmatrix}. \tag{2.6}$$

The ith row in $\mathbf{Xp}$ is the value $y(x_i)$; each row, or linear equation (when considered with $\mathbf{y}$), corresponds to one data point. Under ideal conditions (if the data fit the model perfectly), we would have no y-residuals and $y_i = \mathbf{y}_i$ would equal $y(x_i) = (\mathbf{Xp})_i$. Each row would be an exact equality, and we could find an exact solution. However, we do not have such luck, but must instead use both $\mathbf{X}$ and $\mathbf{y}$ to find a best $\mathbf{p}$. An initial guess, which is correct, is to take $\mathbf{y}$ and $\mathbf{Xp}$ as vectors in $\mathbb{R}^n$ and examine the distance between them, or the length of $||\mathbf{Xp} - \mathbf{y}||$. The squared Euclidean length of $||\mathbf{Xp} - \mathbf{y}||$ is given by:

$$\left(\sqrt{\sum_{i=1}^{n}(\mathbf{y}_i - (\mathbf{Xp})_i)^2}\right)^2 = \left(\sqrt{\sum_{i=1}^{n}(y_i - ax_i + b)^2}\right)^2 = \sum_{i=1}^{n}(y_i - ax_i + b)^2. \tag{2.7}$$

The square of the distance between the vectors (Equation 2.7) is the sum of squared residuals (Equation 2.4). Therefore, if we can then find the $\mathbf{p}$ that minimizes the squared distance between $\mathbf{Xp}$ and $\mathbf{y}$, then we have the $\mathbf{p}$ that minimizes the sum of squared residuals. Thus, our original problem is then reduced to finding this $\mathbf{p}$.

Having posed our problem in terms of a system of linear equations, we can then move onto the task of solving it using a result from linear algebra: given a system of linear equations $\mathbf{Xp} = \mathbf{y}$ for which there is no exact solution, one can find a solution $\hat{\mathbf{p}}$ such that $||\mathbf{y} - \mathbf{X}\hat{\mathbf{p}}||$ is minimized.

As vector lengths are non-negative, and $f(x) = x^2$ is always increasing over all non-negative real numbers, any $\hat{\mathbf{p}}$ that minimizes $||\mathbf{y} - \mathbf{X}\hat{\mathbf{p}}||$ must also minimize $||\mathbf{y} - \mathbf{X}\hat{\mathbf{p}}||^2$. Accordingly, the minimizing $\hat{\mathbf{p}}$ is the $\mathbf{p}$ that minimizes the sum of

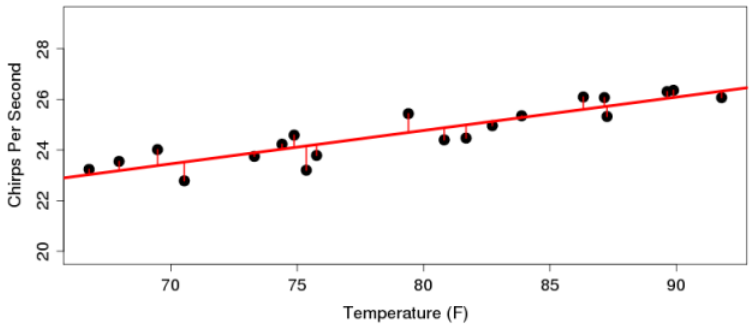


Figure 2.2: Cricket chirp data with fitted line and residuals

squared residuals. This $\hat{\mathbf{p}}$ is given concisely by

$$\hat{\mathbf{p}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}. \tag{2.8}$$

For practical reasons, this is often solved in an alternative way [1, 22], but for our purposes, it will suffice. Plugging in our cricket data into the definitions of $\mathbf{X}$ and $\mathbf{y}$ and then using Equation 2.8, we find that $\hat{\mathbf{p}} = [0.1318 \quad 14.2311]^T$, or that the best fitting linear function is $y = 0.1318x + 14.2311$. This line, and the y-residuals are plotted in Fig. 2.2.

Our derivation suggests the future utility of this approach. As Equation 2.8 nowhere makes use of our starting linear function, it seems reasonable that we should be able to pose different problems in the form $\mathbf{Xp} = \mathbf{y}$ and use least-squares to find a squared residual-minimizing solution. By filling $\mathbf{X}$ and $\mathbf{y}$ with values for which $||\mathbf{y} - \mathbf{Xp}||$ represents an error metric that we want to minimize, we can find similar best fits [1, 22]. Immediately, this means that we can extend our approach, for instance, to a degree 2 polynomial $y = ax^2 + bx + c$ or sine function

$y = a\sin(x) + b$, which can be produced by keeping $\mathbf{y}$ as before and setting

$$\mathbf{X} = \begin{bmatrix} x_1^2 & x_1 & 1 \\ \vdots & \vdots & \vdots \\ x_n^2 & x_n & 1 \end{bmatrix} \qquad \text{or} \qquad \mathbf{X} = \begin{bmatrix} \sin(x_1) & 1 \\ \vdots & \vdots \\ \sin(x_n) & 1 \end{bmatrix} \tag{2.9}$$

respectively. Later in this chapter, we will examine more complex formulations.

Despite this apparent power, there is a potential drawback: acute readers may have noticed that both Equations 2.1 and 2.8 are not defined for all possible values. In particular, Equation 2.1 is undefined if $x_1 = x_2$, and in Equation 2.8, $\mathbf{X}^T\mathbf{X}$ is not invertible if the columns of $\mathbf{X}$ are not linearly independent [22]. One pair of points causing both conditions to fail is is $(0, 0)$, $(0, 1)$, a vertical line[1]. Generally, any exactly vertical line cannot be fitted with least-squares linear regression. Since our model is a linear function which cannot have an infinite slope. Nonetheless, while $\{(1, -1), (1, 0), (1, 1), (1, 2)\}$, for instance, is unsuitable for producing a *linear function*, it still defines a *line*. Further, consideration of even near-vertical lines casts doubt on the wisdom of using the y-residual: we should really be seeking to minimize the line-point, or orthogonal, distance. If we are to consider lines as geometric objects in a principled way, we must then use a different approach that works for all lines and which minimizes orthogonal distances.

2.2 PCA

To find an approach that works for all lines, we must turn to Principal Components Analysis (*PCA*). For purposes of space and clarity, we will present it with only an explanation of its computation and one property that is relevant to our application. We will not present a justification of its correctness or an explicit demonstration of

[1] Any line passing through $(0, 0.5)$ is in fact a linear function that minimizes the sum of squared residuals.

its application to the construction of a lower-dimensional representation of data, but both may be found in [1, 3].

Given a set of data points $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^d$, PCA is computed as follows.

1. Center X: compute the mean

$$\boldsymbol{\mu} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i \tag{2.10}$$

and then subtract $\boldsymbol{\mu}$ from each vector $\mathbf{x}_i$, yielding $X' = \{\mathbf{x}_1 - \boldsymbol{\mu}, \ldots, \mathbf{x}_n - \boldsymbol{\mu}\}$ $= \{\mathbf{x}'_1, \ldots, \mathbf{x}'_n\}$. This is illustrated by the translation from Fig. 2.3(a) to Fig. 2.3(b).

2. Compute the sample covariance: since X' has zero mean, this is simplified to:

$$S = \frac{1}{n-1} \sum_{i=1}^{n} \mathbf{x}'_i \mathbf{x}'^T_i, \tag{2.11}$$

or a sum of column vectors multiplied by row vectors. For zero-mean two-dimensional data $X'_2 = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, the sample covariance matrix is

$$S = \frac{1}{n-1} \begin{bmatrix} \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} x_i y_i & \sum_{i=1}^{n} y_i^2 \end{bmatrix}. \tag{2.12}$$

3. Extract eigenvalues and eigenvectors: Compute the eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_n$ of S and their corresponding eigenvalues $\lambda_1, \ldots, \lambda_n$, sorted in descending order of eigenvalues. For a given $m < n$, the m principal components of X are given by $\{\mathbf{u}_1, \ldots, \mathbf{u}_m\}$. The span of the first principal component, $\mathbf{u}_1$ is drawn in Fig. 2.3(c).

It turns out that $H = \text{Span}\{\mathbf{u}_1, \ldots, \mathbf{u}_m\}$ is the m-dimensional hyperplane which the members of X' best fit with respect to orthogonal distance: if $\text{proj}_A(\mathbf{x}'_i)$ is the orthogonal projection of $\mathbf{x}'_i$ onto A, then H is the hyperplane that minimizes the

sum of squared orthogonal distances between the points in X' and their orthogonal projections onto H, or

$$\sum_{i=1}^{n} ||\text{proj}_H(\mathbf{x}'_i) - \mathbf{x}'_i||^2. \tag{2.13}$$

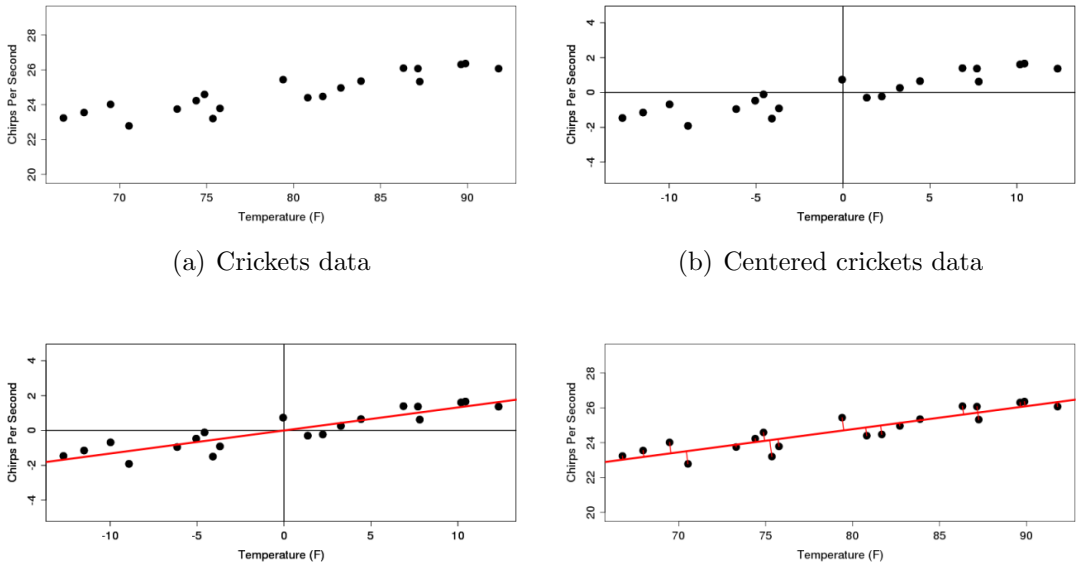
Since data points can be translated without affecting distance, we can translate H and X' by $\boldsymbol{\mu}$ to get the hyperplane that minimizes the sum of squared orthogonal distances between the hyperplane and X, our original collection of points.

As a one-dimensional hyperplane is a line, this is of great interest to us for line fitting: $\mathbf{u}_1$ translated by $\boldsymbol{\mu}$ (the line between $\boldsymbol{\mu}$ and $\boldsymbol{\mu} + \mathbf{u}_1$) is the line that minimizes the sum of squared orthogonal distances (or residuals for generalized line fitting); it is plotted in Fig. 2.3(d) along with the orthogonal residuals. As a sanity check, we can examine its parameters if we take it as a linear function: $y = 0.1323x + 14.1920$. The values are close to those for the least-squares linear regression estimate, which is expected since the line is nearly horizontal (so the distances of the orthogonal residuals are mainly in the y dimension).

We now show for the sake of completeness that PCA works for the vertical line case for which Least-Squares linear regression fails. Let us take the points $X = \{(1, -1), (1, 0), (1, 1), (1, 2)\}$. Their mean is $\boldsymbol{\mu} = [1 \;\; 0.5]^T$, which we subtract from each of the points, yielding $X' = \{(0, -1.5), (0, -0.5), (0, 0.5), (0, 1.5)\}$. We then compute the sample covariance matrix

$$S = \begin{bmatrix} 0 & 0 \\ 0 & 5/3 \end{bmatrix}. \tag{2.14}$$

S has only one eigenvector, $\mathbf{u}_1 = [\; 0 \;\; 1 \;]^T$, and its eigenvalue is 1. We finally translate $\mathbf{u}_1$ with $\boldsymbol{\mu}$ to get the final line, which is between $\boldsymbol{\mu} + \mathbf{u}_1 = [\; 1 \;\; 1.5 \;]^T$ and $\boldsymbol{\mu} = [\; 1 \;\; 0.5 \;]^T$. This is a vertical line, which is the desired result.



(a) Crickets data

(b) Centered crickets data

(c) Centered crickets data with span of principal component

(d) Crickets data, principal component translated, and orthogonal residuals plotted

Figure 2.3: Illustration of PCA for cricket chirp line fitting

2.3 Circle Fitting

We now present approaches for fitting a circle to data. An exact fit for the three points that define a circle may be found by constructing the circumcircle of the triangle formed by the three points. However, it is not immediately clear how to construct a sensible error-minimizing solution for more than three points.

Again, as was the case in previous fitting tasks, we are given a set of data points $X = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, and we wish to find a solution that minimizes an error metric. Conventionally, we parametrize a circle as its center and radius (c_x, c_y, r); the points on it are defined as all $(x, y) \in \mathbb{R}^2$ satisfying

$$(x - c_x)^2 + (y - c_y)^2 = r^2. \tag{2.15}$$

The residual for a data point (x_i, y_i) and circle (c_x, c_y, r) is then naturally defined

as the geometric distance between the point and the circle

$$\sqrt{(x_i - c_x)^2 + (y_i - c_y)^2} - r. \tag{2.16}$$

The correctness of Equation 2.16 becomes apparent when considered along with a series of transformations which do not affect distance: first, center the coordinate system at (c_x, c_y) and then rotate the coordinate system so that (x_i, y_i) lies on an axis. Neither of these change distances; however, the distance between a point on an axis and a circle with radius r at the origin is the distance to the center minus the radius.

It turns out that there is no closed-form solution of the kind we found for linear functions in Section 2.1 that minimizes the sum of squared residuals defined by Equation 2.16; instead, the solution must be found with a closed-form solution for an alternative error metric or using a numerical iterative approach [5]. Given the theme of the chapter, we opt to present an alternative error metric that performs well in practice even if it is not the ideal solution. Other approaches and their relative advantages may be found in the work of Chernov, in particular [5].

The approach that we present is conventionally ascribed to Kasa [20], although Chernov notes that the approach has been rediscovered by a large number of subsequent authors. It makes use of the seemingly more complex parametrization of a circle (c_x, c_y, r) as (a, b, c), where

$$\begin{aligned} a &= 2c_x \\ b &= 2c_y \\ c &= r^2 - (c_x^2 + c_y^2). \end{aligned} \tag{2.17}$$

As before in linear regression, we place the x values in a matrix $\mathbf{X}$; however, we also place the y values in $\mathbf{X}$. Further, the vector we denoted $\mathbf{y}$ last time is filled

with more complicated data, the squared distances of the points from the origin:

$$\mathbf{X} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} x_1^2 + y_1^2 \\ x_2^2 + y_2^2 \\ \vdots \\ x_n^2 + y_n^2 \end{bmatrix}. \tag{2.18}$$

If we then introduce a vector containing our parameters $\mathbf{p} = [\ a \quad b \quad c\ ]^T$, the advantage of our alternative parameters becomes clear. Recall that given a system of linear equations $\mathbf{Xp} = \mathbf{y}$, we can find a $\hat{\mathbf{p}}$ that minimizes $||\mathbf{Xp} - \mathbf{y}||$, which is simply

$$\sum_{i=1}^{n} (\mathbf{y}_i - (\mathbf{Xp})_i)^2. \tag{2.19}$$

We now compute the value of $(\mathbf{y}_i - (\mathbf{Xp})_i)$ to see the form of each term t_i of the minimized sum. Fairly obviously, we start with

$$\begin{aligned} t_i &= x_i^2 + y_i^2 - (ax_i + by_i + c) \\ &= x_i^2 + y_i^2 - ax_i - by_i - c. \end{aligned} \tag{2.20}$$

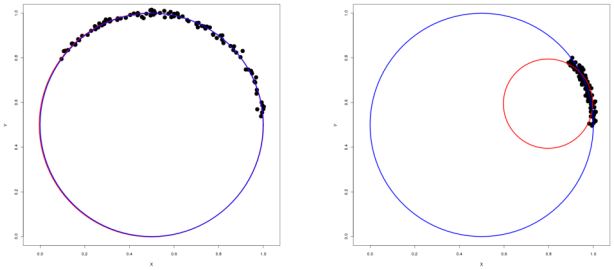
Substituting in the more conventional parameters using the identities introduced by Equation 2.17, we get

$$t_i = x_i^2 + y_i^2 - 2c_x x_i - 2c_y y_i - (r^2 - c_x^2 - c_y^2). \tag{2.21}$$

Readers may notice the two quadratic equations present in the resulting symbol soup, tipped off by $-2c_x x_i$ and $-2c_y y_i$; we may more sensibly arrange the t_i as

$$t_i = \left[(x_i - c_x)^2 + (y_i - c_y)^2\right] - r^2. \tag{2.22}$$

The portion in brackets is the squared distance to the center of the circle, or the squared radius according to the data point, and the r^2 is simply the squared radius of the circle according to the model. Thus, if r_i' is the radius according to point (x_i, y_i), the model $\mathbf{p}$ found by the least-squares fit is the model that minimizes the



(a) 40% Arc Coverage (b) 10% Arc Coverage

Figure 2.4: The circle giving the generative model is shown in blue; the estimated circle is drawn in red.

sum of differences $r_i'^2 - r^2$. This is similar to, but not equal to the squared error measurement of Equation 2.16, which was $(r_i' - r)^2$.

Although the error metric that we minimize is not the one we originally wanted to minimize, it turns out that the results are sufficient for our purposes. The primary shortcoming noted by Chernov in [5] is that the Kasa fit tends to underestimate the radius of the circle if an insufficient portion of the circle is covered by the data. Two fits to incomplete arcs (40% and 10%) are plotted in Fig. 2.4. In each, the points have been subjected to zero-mean Gaussian noise with $\sigma = 1\%$. Despite having samples of only 40% of the arc, in Fig. 2.4(a), the Kasa fit achieves an estimate scarcely indistinguishable from ground-truth. The observation about underestimating the radius is still true, as can be seen in Fig. 2.4(b), where only 10% of the arc is sampled and the estimate is poor. However, it is not clear to what extent the challenge of estimating such a circle is relevant to practical computer vision tasks. Further, if such situations do arise, a combined Levenberg-Marquardt iterative minimization with an algebraic initialization is capable of successfully fitting the points with overwhelming probability, even at arc lengths as small as 1.5% [5].

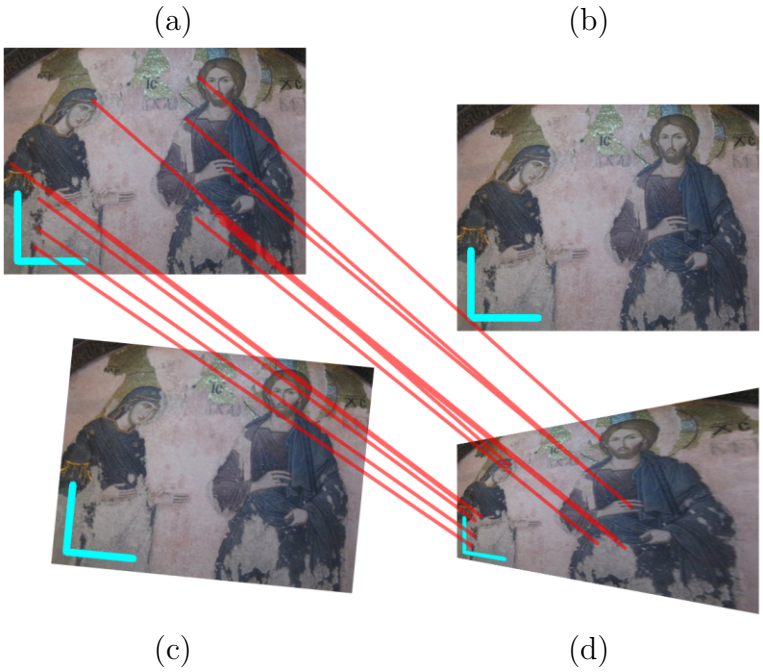


Figure 2.5: A depiction of image transformations: (a) original image; (b) translation; (c) perspective; (d) Euclidean.

2.4 Homography Fitting

We examine one final model-fitting task before we introduce the problem of outliers. Unlike the previous three models (linear functions, lines, and circles), our last model does not correspond with any high-school level mathematics, but instead gives us our first glimpse at a model with immediately apparent applications to vision.

Consider any pair of the images depicted in Fig. 2.5. We can describe our view of each of these images by, intuitively speaking, changing our viewing location with respect to the original image: imagine looking at the mosaic in a museum. Equivalently, we can describe how the original image itself is "transformed" into the other image by describing the destination in the second image of each point in the first image. Effectively, if $\mathbf{x} = [\ x \quad y\ ]^T$ and $\mathbf{x}' = [\ x' \quad y'\ ]^T$ are points in

the first image and second image corresponding to one another, for instance any two endpoints of the red lines in Fig. 2.5, then the transformation between the two images is described by some function $T : \mathbb{R}^2 \to \mathbb{R}^2$ such that $T([\ x\ \ b\ ]^T) = [\ x'\ \ y'\ ]^T$.

Intuitively, that the image in Fig. 2.5 remains a quadrilateral under the transformation suggests an estimate for how many parameters this function should have: all we have to do to describe the change in x and y of each corner of the quadrilateral, and so the transformation should have 8 parameters. This assessment turns out to be correct: we can describe each of the four transformations in Fig. 2.5 with the equation $\mathbf{Hx} = \mathbf{x}'$, where $\mathbf{H}$ is a 3×3 matrix, called a *homography* or *planar homography*, containing the parameters of the transformation[2]. Such an equation is presented below; note the 1 in the lower-right entry, giving us our 8 parameters.

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x'w \\ y'w \\ w \end{bmatrix} \tag{2.23}$$

Throughout our discussion of homographies, we will use the term homography to refer to the matrix $\mathbf{H}$, the associated function $T([\ x\ \ y\ ]^T) = \mathbf{H}[\ x\ \ y\ \ 1]^T$, and the notion of the transformation in the abstract sense.

Unfortunately, to use a homography, we have to work with an alternate representation of our data points. Specifically, $[\ x\ \ y\ ]^T$ becomes $[\ x\ \ y\ \ 1\ ]^T$, and the result that we get, $[\ x'w\ \ y'w\ \ w\ ]^T$, must be divided by the parameter w to yield the point that we actually want $[\ x'\ \ y'\ ]^T$. These vectors are referred to as *homogeneous coordinates*, where the two vectors are equal if and only if they differ only by scale [30]. One converts regular coordinates to homogeneous ones by adding a 1 to the end and one converts homogeneous coordinates to conventional

[2]Mathematically, not every homography may be represented in this form with 8 parameters. There are failure cases when the bottom-right entry in the matrix is 0. However, for the types of transformations encountered in practice, setting the bottom-right element to 1 suffices.

ones by dividing $x'w$ and $y'w$ by the final element of the homogeneous coordinate, w. Homogeneous coordinates may have non-zero values in the first two entries and a zero-valued third entry; these have no representation in conventional (x, y) form [30]. In addition to being useful for computing things like the intersection of lines (see Szeliski [30]), homogeneous coordinates permit us to use Equation 2.23: by including the 1 in the vector, we can translate every point by a constant amount in both the x and y dimension.

We now present one of the homographies used to create Fig. 2.5 to illustrate how the parameters work[3]. Consider the transformation Fig. 2.5b, referred to as translation, which may be written as $(x', y') = (x + t_x, y + t_y)$. Using this, we may derive two equations which help us derive a homography:

$$\begin{aligned} x' &= x + t_x &&= 1 \cdot x + 0 \cdot y + t_x \cdot 1 \\ y' &= y + t_y &&= 0 \cdot x + 1 \cdot y + t_x \cdot 1. \end{aligned} \tag{2.24}$$

We still have the w to manage in the resulting homogeneous coordinate, but we can simply fix it to 1 with:

$$1 = 0 \cdot x + 0 \cdot y + 1 \cdot 1. \tag{2.25}$$

Plugging in, we get that the following satisfies our above constraints on:

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x'w \\ y'w \\ w \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}. \tag{2.26}$$

Examining how w is calculated suggests that any transformation with a bottom row of $[\ 0\ \ 0\ \ 1\ ]$ is more "well-behaved" with regards to w than ones with non-zero entries. Such transformations are referred to as affine transformations and have

[3]Note that in both Szeliski [30] and Hartley and Zisserman [18], the homographies are parametrized differently from this way. We are following the notation from Wren's document [41] to accompany [12].

special properties: for instance, one may represent the affine transformation with a 2×3 matrix. These transformations are discussed in Szeliski [30] and in depth by Hartley and Zisserman [18].

In general, however, we cannot simply write a clear explicit formula for (x', y') in terms of (x, y) if we have something as complicated as the perspective transformation in Fig. 2.5. However, our approach for guessing the parameter count of the transformation suggests an alternate approach. If we have corresponding points in each image, then we should be able to find the underlying homography. We further hope that our approach should be able to find the best fit if there is localization noise in the data.

Again, we start out with a collection of data points $X = \{(x_1, y_1, x'_1, y'_1), \ldots, (x_n, y_n, x'_n, y'_n)\}$; in this case, (x_1, y_1) in image 1 corresponds with (x'_1, y'_1) in image 2. We want to find a homography that best captures this transformation; how we might obtain these correspondences will be discussed in a later chapter. If $\mathbf{H}$ is the homography implicitly under consideration, then for each correspondence i, we write the predicted locations of the points under the homography using the following equations

$$\mathbf{H} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} \hat{x}_i w_i \\ \hat{y}_i w_i \\ w_i \end{bmatrix}. \tag{2.27}$$

If the points can be exactly modeled with a perspective transformation, then $\hat{x}_i = x'_i$ and $\hat{y}_i = y'_i$. Note for later that there is a w_i for each point.

Before we write out a solution for finding an estimate for $\mathbf{H}$, we summarize our notation: (x_i, y_i) and (x'_i, y'_i) are the corresponding points in image 1 and 2 for the ith correspondence. With respect to an implicit $\mathbf{H}$, $(\hat{x}_i, \hat{y}_i)$ is the mapped location of (x_i, y_i) under the transformation $\mathbf{H}$ and w_i is the associated scale parameter that we remove to get $(\hat{x}_i, \hat{y}_i)$.

In the previous minimization setups, we have aimed at minimizing a single value for each data point: in linear regression, we minimized the y-residual, and in the Kasa circle fit, we minimized the difference between the squared radius according to the point and the squared radius of the model. Here, we seek something more subtle. Ideally, we would like to minimize the squared residuals:

$$\sum_{i=1}^{n} \left(\sqrt{(\hat{x}_i - x_i')^2 + (\hat{y}_i - y_i')^2} \right)^2 = \sum_{i=1}^{n} (\hat{x}_i - x_i')^2 + (\hat{y}_i - y_i')^2. \tag{2.28}$$

In all of the setups that we have presented so far, each term of our target function has generally corresponded with one row of the matrix $\mathbf{X}$ and vector $\mathbf{y}$; however, we can easily index the sum differently and there is no particular reason why the x and y terms need to be produced in the same row. If, in the end, Equation 2.28 can be written as $||\mathbf{Xp} - \mathbf{y}||$, that is all that matters. We will use a setup due to Wren [41], which comes from a document to accompany the work of Criminisi et al. [12]. We reshape our homography into an 8×1 vector, and, for each data point, introduce a row for x and y values, resulting in a $2N \times 8$ matrix $\mathbf{X}$ and $2N \times 1$ vector $\mathbf{y}$:

$$\mathbf{Xp} = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1'x_1 & -y_1'y_1 \\ & & & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_n'x_1 & -x_n'y_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y_n'x_n & -y_n'y_n \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{bmatrix} = \begin{bmatrix} x_1' \\ y_1' \\ \vdots \\ x_n' \\ y_n' \end{bmatrix} = \mathbf{y}. \tag{2.29}$$

We can see what value is minimized by a least-squares solution to this equation by dissecting the $2i$th row of $||\mathbf{Xp} - \mathbf{y}||$. Note that $\mathbf{y}_{2i} = x_i'$. Expanding out, we

get

$$\begin{aligned}(\mathbf{Xp})_{2i} - \mathbf{y}_{2i} &= (ax_i + by_i + c + 0d + 0e + 0f - gx'_i x_i - hx'_i y_i) - x'_i \\ &= ax_i + by_i + c - gx'_i x_i - hx'_i y_i - x'_i \\ &= ax_i + by_i + c - x'_i(gx_i + hy_i + 1).\end{aligned} \tag{2.30}$$

Note that we have two of the entries from $[\ x'_i w_i \ \ y'_i w_i \ \ w_i]^T$ in the above equation: $w_i = gx_i + hy_i + 1$ and $\hat{x}'_i w_i = ax_i + by_i + c$. Therefore Equation 2.30 is equal to:

$$\begin{aligned}(\mathbf{Xp})_{2i} - \mathbf{y}_{2i} &= ax_i + by_i + c + x'_i(gx_i + hy_i + 1) \\ &= \hat{x}_i w_i - x'_i w_i \\ &= w_i(\hat{x}_i - x'_i).\end{aligned} \tag{2.31}$$

This is the x-residual multiplied by w_i. Similarly, we find that

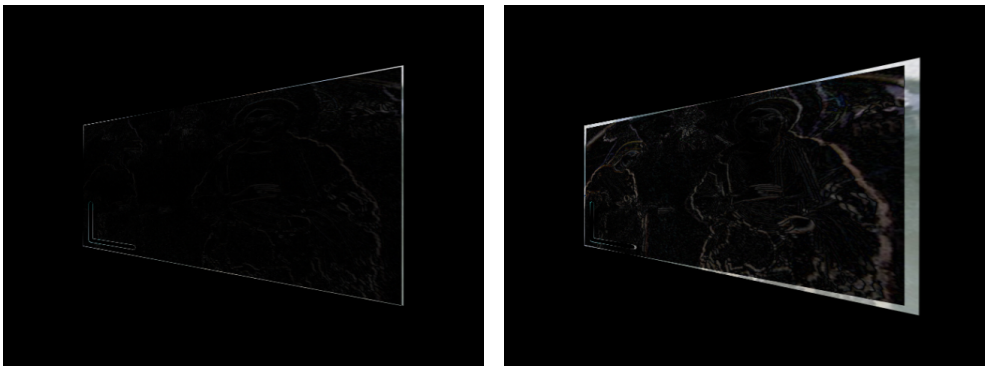
$$\begin{aligned}(\mathbf{Xp})_{2i+1} - \mathbf{y}_{2i+1} &= \hat{y}_i w_i - y'_i w_i \\ &= w_i(\hat{y}_i - y'_i).\end{aligned} \tag{2.32}$$

Accordingly, the minimizing $\mathbf{p}$ minimizes

$$\begin{aligned}\textstyle\sum_{j=1}^{2n}(\mathbf{Xp})_j - \mathbf{y}_j)^2 &= \textstyle\sum_{i=1}^{n} w_i^2(\hat{x}_i - x'_i)^2 + w_i^2(\hat{y}_i - y'_i)^2 \\ &= \textstyle\sum_{i=1}^{n} w_i^2((\hat{x}_i - x'_i)^2 + (\hat{y}_i - y'_i)^2),\end{aligned} \tag{2.33}$$

or the squared distance between the predicted points and actual points *weighted by the square of the scaling parameter* w_i. This is not what we want since our residuals are weighted by parameters that we do not have control over. Nonetheless, as we may find an exact solution to the system of equations with 4 correspondences (i.e., where Equation 2.33 is zero), this setup provides our exact solution: the weightings come into play only when the x and y residuals are not zero, and if the x and y residuals are not zero, then we do not have an exact solution.

It turns out that there is no closed form solution to minimize the residuals. Szeliski [30] articulates this and advocates an iterative method. One of the benefits of an iterative approach is that with an initial estimate of the parameters $\mathbf{p}$, one



(a) No outliers (b) One Outlier

Figure 2.6: Difference images for a least-squares fit: Fig. 2.6(a) with no outliers; Fit. 2.6(b) with one outlier. Black regions indicate perfect alignment.

can attempt to compensate for the resulting w_i by using the current estimate of w_i and dividing through. The most principled approach, according to Szeliski, takes this slightly further, and formulates the minimization so that it can be solved using the iterative Gauss-Newton algorithm. On the other hand, Hartley and Zisserman [18] do not advocate for iterative approaches, but instead present a more complex, closed form equation. Nonetheless, although not theoretically perfect, the system in Equation 2.29 performs well enough in practice. An example transformation is given in Fig. 2.6(a): we compute the per-channel absolute difference between the image transformed by the ground-truth homography used to generate Fig. 2.5 with the one transformed by the homography fitted to the red line correspondences. The darker the image, the better the fit; in this case, apart from a very slight edge from the white background where the image edges do not line up pixel-perfectly, the resulting homography is well fitted.

2.5 Conclusions

We have outlined the process of estimation for four classes of models: linear functions, lines as geometric objects, circles, and homographies. In the first two cases, we were able to obtain a closed-form solution to minimize our desired residual function. In the other two cases, no such closed-form solutions exist and we used alternative error metrics that perform sufficiently well in practice; however, in each of these cases, we could also find an approximate minimum with an iterative approach.

However, in each of these estimations, we implicitly assumed that there were no outliers. We have already demonstrated in the introduction that PCA fails in their presence. We might similarly insert an outlier data point into our linear function fitting data set, for instance of a cricket chirping at 500 chirps per second at temperature 80° F, and produce an incorrect fit. Further, the outlier need not be so dramatic or apparent: to generate Fig. 2.6(b), we introduced an outlier produced by taking a single point in image 1, transforming it using the correct homography, and simply translating it by $t_x = -10$, $t_y = 10$ (less than 3% of the width and height). Although there is only a single outlier, which is wrong by only about 15 pixels, as can be seen, the fitted homography is significantly incorrect.

To handle such an outlier, we will have to turn to an entirely different paradigm; this is the topic of the next chapter.

CHAPTER 3

OUTLIER-ROBUST SINGLE MODEL ESTIMATION

The spectacular failure of least-squares techniques in the presence of outliers suggests that we will have to take a different approach when estimating models in outlier-contaminated data. In particular, we must abandon our sole criterion of minimizing the residuals of every data point with respect to the model: as we articulated earlier, this leads to issues where we must satisfy two mutually conflicting goals (e.g., two models). In its place, we must adopt a different criterion for measuring the quality of a model and a method for achieving this criterion.

To elicit some requirements for our new criterion and to preemptively demonstrate that more standard robust estimators will not suffice in the context of vision, we present an analysis of Least Median of Squares (LMedS) fitting due to Rousseeuw [26].

Rousseeuw argued that the median rather than sum or mean of the squared residuals should be minimized. Intuitively, this makes sense: if the data has 25% outliers, a model accurately describing the inliers will have low residuals for 75% of the data. In fact, LMedS can be proved to correctly estimate models in data containing up to 50% outliers so long as certain assumptions about data composition hold.

It turns out that the practical calculation of LMedS renders such guarantees invalid and that even if they held that they would be insufficient for the tasks in computer vision. It is not immediately apparent how the minimizing model can be achieved in closed form; implementations instead generally find an approximation by generating a random sampling of models and then choosing the sampled model that minimizes the median of squared residuals [28, 40]. Wang and Suter point out in [40] that this is problematic: although we can argue that the approximations

should perform sufficiently well, using an approximation eliminates any guarantee of correctness. Further, Wang and Suter argue that even if we can implement the exact solution to minimize the median of squares, LMedS is still inappropriate in vision: the correctness proofs of LMedS assume a uniform distribution of outliers, which fails to hold when there are pseudo-outliers belonging to another structure. Finally, in computer vision, outlier compositions greater than 50% are common, especially in the presence of multiple structures [40]. As an example, if there are three structures in data with 50 data points each, each model sees $100/150 = 66\%$ outliers, even without any gross outliers.

Our purpose for introducing LMedS is not to denigrate it, but instead to elicit some requirements for a satisfactory model estimation algorithm. One primary concern is that the algorithm should be able to estimate models in the presence of both gross outliers and pseudo-outliers: we are not presently concerned with the estimation of multiple models, but if we are to analyze or even tolerate multiple models, then our single-model techniques should provide one of the models rather than an incoherent estimate. Further, our approach should be able to function with a very high outlier composition, even if we cannot prove it: the assumptions necessary for proofs and proof-invalidating approximations in practical calculations for LMedS suggest that proofs of correctness have limited utility for this problem.

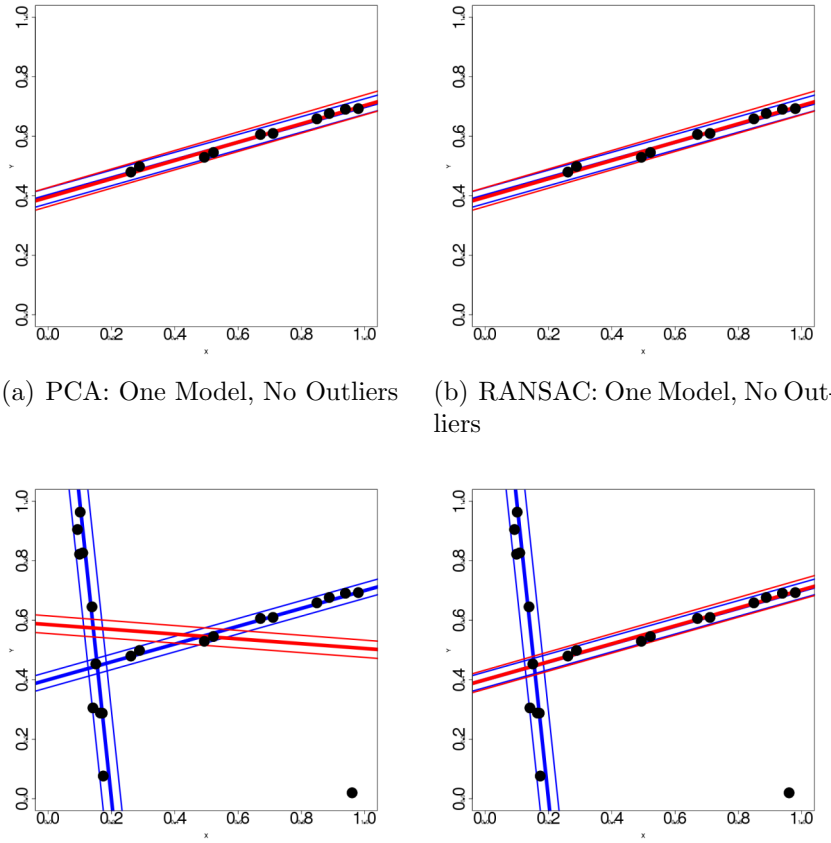
In this chapter, we introduce Fischler and Bolle's celebrated Random Sample Consensus (RANSAC) algorithm [15] for outlier-robust single model estimation, which satisfies the above requirements. In the process, we also provide a domain in computer vision in which robust model estimation is useful, the detection of planar surfaces.

3.1 RANSAC

We motivate RANSAC by reusing the data points from the introduction with a slight modification to how we plot the data. Again, data points were generated on lines, and then perturbed with zero-mean isotropic (independent and with the same variance in each dimension) Gaussian noise with magnitude σ. As before, red lines represent the model found by the technique and blue lines represent the ground-truth data. However, we have added additional, thinner, lines representing the region of points 3σ away from each model. This distance is chosen for its customary use with the 1 dimensional Gaussian distribution, in which 99.7% of the distribution falls within 3σ. In the two-dimensional case, a similar portion of the distribution (about 98.9%) is within 3σ from the mean. Assuming that we know the magnitude, or scale, of the noise of the data, 3σ functions as a reasonable cutoff for determining whether a point belongs to a model: we will, in the long run, reject only about 1.1% of points actually generated by the model while maintaining a fairly tight bound on what we accept. These points and lines are plotted in Fig. 3.1.

We can use these boundary lines to motivate a dramatically different goal than either Least Squares or Least Median of Squares. Examining Fig. 3.1 suggests that we can answer the question "how well does a model fit the data" by looking at how many points fall near enough to the model. RANSAC uses this intuition that strong models should fit a large number of data points to formulate a new criterion for model selection: we want to find the model that fits the largest number of data points.

Just as we did for least squares, we must similarly define a notion of how badly points fit. However, as we do not want to re-derive RANSAC for each model class, we do this in an abstract way. If we have a model μ and a collection of data points



(a) PCA: One Model, No Outliers

(b) RANSAC: One Model, No Outliers

(c) PCA: Two Models, One Outlier

(d) RANSAC: Two Models, One Outlier

Figure 3.1: Comparing RANSAC with PCA

$D = \{p_1, \ldots, p_n\}$, we first define an error function $R(\mu, p)$ that maps a model and point to a non-negative real number. In the case of lines, R is the line-point distance, for circles, it is the circle-point distance, etc.

The points that fit to within a threshold are known as the model's *consensus set*; this threshold is commonly known as an *inlier threshold*, as it describes how closely points must fit a model have to be to be considered its inliers. Formally, we might define the consensus set of a model μ with respect to a data set D and inlier threshold ϵ as

$$\mathrm{CS}(\mu, D, \epsilon) = \{p \in D : R(\mu, p) < \epsilon\}. \tag{3.1}$$

Our goal can now be articulated formally as follows: given the data set D and inlier threshold ϵ, find the μ that will maximize the size of the consensus set; formally, find the $\hat{\mu}$ such that the cardinality $\mathrm{CS}(\hat{\mu}, D, \epsilon)$ is maximized.

It is crucial to note that this goal has no formal mathematical basis or justification, but is instead a heuristic. In contrast, one may derive that minimizing the sum-of-squared errors, as in least-squares linear regression, is equivalent to finding the maximum-likelihood solution under an assumption of Gaussian noise [3]. Due to the Central Limit Theorem, which asserts that the mean of a collection of samples from most probability distributions converges to a Gaussian as the sample size goes to infinity [21], this assumption might make sense in a large number of contexts; thus, the goal of least-squares linear regression might be, in some cases, the only sensible goal. We do not have such connections for the consensus-set size heuristic, and in Chapter 5 we will see how this function may contradict our notions of correctness.

The idea of implementing this goal should be met with skepticism: we are proposing the search of an infinite multi-dimensional search space (the parameter space) to maximize a complicated function. Recall that we were only able to solve least-squares problems by posing them as solutions to systems of linear equations and, in many cases, accepting alternative error metrics; in comparison to our goal of maximizing the cardinality of the consensus set, the least-square problems look trivial.

If we have only a few parameters, we could discretize our search space into a finite number of bins and search those. This is more or less what the Hough Transform does: roughly speaking, in the Hough Transform, each data point "votes" once for the models that it matches, and models in the discrete search space which have the most "votes" are the likely hypotheses (for more see Szeliski [30], Xu et al. [42],

and Duda and Hart [14]). However, a naïve discretization approach of partitioning each parameter's space into a finite number of bins becomes intractable for models with many degrees of freedom: for instance, discretizing the space for homographies into 64 bins in each dimension would require managing $64^8 = 2.81 \times 10^{14}$ bins; if we only use 4 bytes per bin, this is an astounding 2048 terabytes. Even if we turn to sparse methods for accumulating the votes, this seems insurmountable, even to just find the bin with the maximum votes.

It turns out that there is a way to search such a large space effectively if we are willing to accept the possibility of failing to detect the model. The critical observation is that we do not start out with complete ignorance of what models are likely to have strong consensus sets. In fact, the cues for which models are likely to be strong are latent in the data we are provided. Consider Fig. 3.1: we could take all the pairs of points in the data and compute exact models for them. This would inevitably generate a very large number of models similar to our ground-truth models (and thus with large consensus sets). Generalizing to arbitrary models, we can take all of the subsets of the data with the minimum number of points required to estimate a model, termed *minimum sample sets*, and search through these. We also refer to a model generated by a minimum sample set as a *minimum sample model*. We are searching a finite discretized subset of the model space just like in the binning approach; however, we compute a data-dependent, and, we hope, more informative discretization. If we denote the number of data points as N and the minimum number of points required to estimate a model as MSS, then there are only $\binom{N}{MSS}$ models. For instance, if we have 1000 correspondences, we then only wish to search through $\binom{1000}{4} = 9.94 \times 10^{11}$ models.

Although this is 0.3% the size of our binned search space, it is still far too large. To overcome this, we reiterate the observation that a very large number of

the minimum sample models correspond to ground-truth models. If this is true, then we can presumably look at $M \ll \binom{N}{MSS}$ minimum sample sets: although it is not guaranteed to succeed, we can be confident that if we use a sufficiently large M, then we will probably find at least one corresponding to a ground truth model. We will be able to distinguish such a model by its large consensus set. We can now tersely describe RANSAC's approach: approximately maximize the consensus set size by searching through M models determined by randomly sampling minimum sample sets from the provided data set.

Having now introduced its primary components (consensus set size as a model-validity indicator and random sampling of the model space with minimum sample sets), we present the RANSAC algorithm. Let $\mathcal{D}$ be the space of data points (e.g., $\mathbb{R}^2$ for points) and $\mathcal{M}$ be the space of models (e.g., $\mathbb{R}^3$ for lines and $\mathbb{R}^8$ for homographies); suppose we have the following functions and values:

1. the minimum number of points required to estimate a model, MSS;
2. an inlier threshold ϵ;
3. an error function $R : \mathcal{M} \times \mathcal{D} \to \mathbb{R}^+ \cup \{0\}$;
4. a model estimation function $E : \mathcal{D}^{MSS} \to \mathcal{M}$;
5. a sampling procedure $S : P(\mathcal{D}) \to \mathcal{D}^{MSS}$, where $P(\cdot)$ is the power set function.

These define any particular domain (e.g., line fitting) sufficiently well that we can describe RANSAC for any problem instance, which we present in Fig. 3.2.

As a note, we might apply an additional check on the size of the consensus set of the resulting model. This procedure will always return *some model*; if we are not are not sure whether any model is present in the data, it might behoove us to ensure that we are not accepting a poorly-supported model.

RANSAC($DataPoints \subset \mathcal{D}$,$M \in \mathbb{N}$, $\epsilon \in \mathbb{R}^+$)

```
1  // Keep track of the best model ranked by consensus set size
2  BestModel, MaxCSSize = None, −1
3  for i = 1 to M
4       // Sample a model and compute its consensus set
5       MSSPoints = S(DataPoints)
6       MSSModel = E(MSSPoints)
7       MSSModelCS = {d ∈ DataPoints : R(MSSModel, d) < ε}
8       if |MSSModelCS| > MaxCSSize
9            BestModel, MaxCSSize = MSSModel, |MSSModelCS|
10 return BestModel
```

Figure 3.2: The RANSAC algorithm

3.1.1 How many samples?

Note that RANSAC not only takes a data set, but also a number of samples to draw. Intuitively, it seems that the number of samples would impact the probability of succeeding at our task. How the two are related is, in fact, crucial to RANSAC's justification. If finding the model with overwhelming probability takes 10^{11} iterations, then we have not improved at all, but in fact regressed.

We can evaluate the probability of RANSAC succeeding as a function of the outlier composition and number of minimum sample sets. Let us assume that we select the data points for the minimum sample set by simply sampling *MSS* data points at random. If the probability of drawing an inlier to a particular model is p, then the probability of picking a minimum sample set corresponding to this model is p^{MSS} since we have to get exclusively inliers to succeed[1]. The probability of us drawing a minimum sample set that contains a non-inlier to the model in question (gross outlier or pseudo-outlier) is then $1 - p^{MSS}$. Since an outlier will make the

[1]Technically, one should write this as sampling data points without replacement rather than with replacement. However, for large enough N, whether or not we replace has a negligible impact on the probability of drawing inliers and introduces notation that is more difficult to manage. If one wants to be pedantic, we can set p to the smallest probability of picking an inlier, and thus provide a lower bound on the probability.

model inaccurate, this is the probability of failing to draw a coherent model in one iteration of RANSAC. Now, suppose we draw M such minimum sample models: the probability of getting one or more inlier-only minimum sample sets is simply one minus the probability of getting no inlier-only minimum sample sets, or:

$$1 - (1 - p^{MSS})^M. \tag{3.2}$$

as is described in [30]: each drawing is independent and has $(1 - P^{MSS})$ probability of failure. Szeliski gives a convenient identity to provide a number of "trials" or minimum samplings to perform given the probability of drawing an inlier p, minimum sample size MSS, and desired probability of success P:

$$M = \left\lceil \frac{\log(1 - P)}{\log(1 - p^{MSS})} \right\rceil \tag{3.3}$$

where we use $\lceil \cdot \rceil$ to ensure that we have a whole number of samples.

Suppose we want to find a correct homography from 1000 correspondence data points with 99% probability. If we have one model with 250 points, and 75% outliers, then we set $p = 250/100 = 0.25$, $MSS = 4$, and $P = 0.99$. M is then $\log(1 - 0.99)/\log(1 - (0.25)^4) = 1176$. This is significantly better than any of our previous search spaces (9.94×10^{11} and 2.81×10^{14}). Nonetheless, given the subject of this thesis and our previous observation that pseudo-outliers dramatically boost the effective outlier composition of a data set, it seems inappropriate to assume a single model. Suppose instead that we have 75% gross outliers and 5 models with 50 data points each. The number of samples required is given by the application of Equation 3.3 with an updated value of p, $50/1000 = 0.05$. Plugging in, we get $736,825$; although this is much better than our previous values, it is way too large for RANSAC to be tractable.

One fault with the previous analysis is that this is not the value that we actually want to compute for the case of detecting a single model out of many: we

only want to find a coherent model, not one particular model. We provide an alternative version of Equation 3.3 for the detection of any of the present models by simply revising the probability of failure at any trial; the remaining mathematical machinery remains correct.

Suppose we have W models with m points each and a total of n points (and thus $n - mW$ outliers). We will again view the main loop of RANSAC as a series of trials. We succeed in any sampling round if we sample a coherent model, or points entirely from any one model. We can therefore partition the event space of each round into the following disjoint events: we sample entirely from model i, which we denote as M_i, for each value of i; and we produce an incoherent sample and thus fail the trial, which we denote as F. Since we have partitioned the event space, $1 = P(F) + \sum_{i=1}^{W} P(M_i)$. For every i we define $P(M_i)$ as uniformly selecting points:

$$P(M_i) = \left(\frac{m_i}{N}\right)^{MSS}. \tag{3.4}$$

The probability of producing an incoherent model $P(F)$, and thus failing, is just one minus the other parts of the partition of the event space:

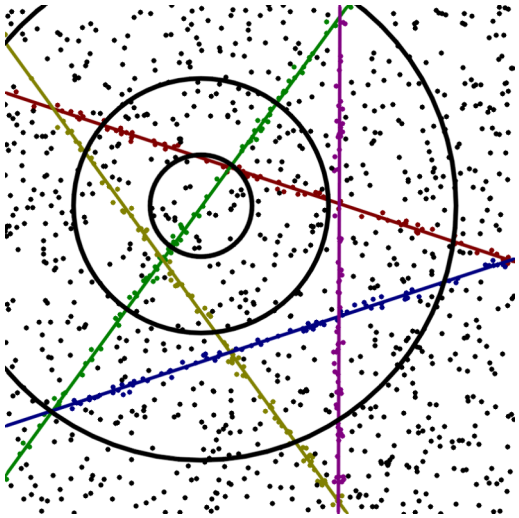
$$P(F) = 1 - \sum_{i=1}^{W} P(M_i) = 1 - \sum_{i=1}^{W} \left(\frac{m}{N}\right)^{MSS} = 1 - W\left(\frac{m}{N}\right)^{MSS}. \tag{3.5}$$

The rest of Szeliski's setup remains as before, and we find that

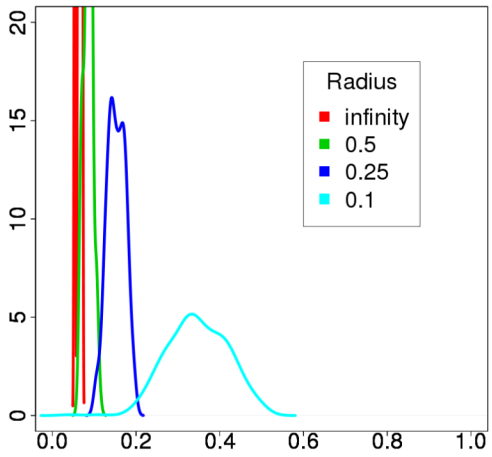
$$M = \log(1-P)/\log\left(1 - W\left(\frac{m}{N}\right)^{MSS}\right). \tag{3.6}$$

We can see how many samples we must draw for the example that motivated our original analysis of the multiple model equation: 5 models of 50 points each ($m = 50$, $W = 5$), 75% gross outliers ($N = 1000$), 4 data points to a minimum sample set (MSS), and a desired success probability of 0.99. Plugging in, we get a disappointing $147,364$ models to sample[2].

[2]Note that the seeming closeness to exact division by $s = 5$ to get this from the previous value is a misleading; adjusting the values of MSS, W, and m results in ratios further from exactly 5.



(a) A data set containing 5 models with 50 inliers each and 80% gross outliers



(b) PDFs of $P(C \mid r, D)$ for each r

Figure 3.3: Geometric distance as a cue: Fig. 3.3(b) depicts the PDFs, marginalizing over all data points, of $P(C)$ for a variety of radii

Although this new estimate produces satisfactory values for classes requiring fewer points to estimate a model (for instance, if in the previous example we require only two points, enabling 0.99 probability requires only 574 samples), this approach is will not work for homographies: despite the seemingly surmountable $p = m/N = 50/1000$, it is easy to forget that the total probability of M_i is further weighted by p^{MSS-1}, which is enormously small. Although RANSAC will work for some models, if we want to succeed at high-parameter-count model estimation in data with large numbers of outliers with high probability, we will have to improve $P(M_i)$.

3.1.2 Non-uniform sampling

The insight into improving our probability of sampling a coherent model is that selecting each point independently and uniformly is not the best sampling strategy. Once we have selected one point, to minimize the number of iterations, we want a

sampling strategy for the remaining points that maximizes $P(M_i)$. Let us denote as C (for coherence) the event that after picking our first point, we pick a point that permits the formation of a coherent model (i.e., a point sharing a model with the first point). Under uniform sampling, $P(C)$ is merely the proportion of points in the data that match the given model, and so $P(C) = p$. $P(M_i)$ can then be written as follows:

$$P(M_i) = pP(C)^{MSS-1}$$

The question then becomes: how can we get a better $P(C)$ than just p?

Consider Fig. 3.3(a), which depicts a data set bounded in the unit square with multiple models (5 with 50 inliers each with different colors) and 80% gross outliers. Suppose we picked the point at the center of the concentric rings as our first point; if we were to limit our uniform selection of the subsequent points to the innermost ring, our probability of picking a coherent model would be dramatically improved: compare the distribution of models within this ring with that of the entire image.

We can verify our intuition by computing the empirical distributions of the proportion of points that belong to the same model in the circle of radius r centered at an inlier point. Note that if we sample uniformly from within this region, the proportion of points that belong to the same model gives the probability that the next sampled point will permit the formation of a coherent model. Thus, the proportion of points inside the ring centered at each p with models matching p's model gives $P(C)$. We take a simple frequentist approach and, for a given radius r using the data D from Fig. 3.3(a) we compute for every data point $p \in D$

$$P(C \mid p, D, r) = \frac{|\{q \in D : \text{Dist}(p, q) < r, \text{Model}(p) = \text{Model}(q)\}|}{|\{q \in D : \text{Dist}(p, q) < r\}|}, \tag{3.7}$$

where $\text{Dist}(\cdot, \cdot)$ is the standard Euclidean distance and $\text{Model}(\cdot)$ maps data points to their ground-truth models. We compute a density function estimate (more or less giving a sense of the underlying distribution using a histogram of points) using

a Gaussian kernel (with bandwidth auto-selected by R) to produce an estimate of the PDF of the value of $P(C \mid r, D)$. As can be seen, if we include every point ($r = \infty$), then we get a sharp peak close to 0 (corresponding to $P(c) = p$): if we sample uniformly, it is likely that $P(C) = p$. On the other hand, if we set $r = 0.1$ (the size of the smallest ring in the image), $P(C)$ is much more favorably distributed; the median and mean are at about 0.34 (which is about 6 times more likely than p): if we sample within a 0.1 radius, we are likely to find $P(C) = 0.34$. Running this procedure on correspondence data (considering the geometric locations in only one image) produces similar results.

How much better is $P(C) \approx 0.34$ for estimating homographies with many outliers? We can plug $P(C)$ into our updated version of $P(M_i)$ and then plug that into Equation 3.6:

$$M = \log(1 - P) / \log\left(1 - W\left(p \cdot 0.34^{MSS-1}\right)\right). \tag{3.8}$$

Setting $P = 0.99$ as before, $W = 5$, $MSS = 4$, and $p = m/N = 50/1000$, we get an astonishing $M = 467$ samples. Let us now compare this with the previous estimates: our uniform discretization of homographies had 2.81×10^{14} bins; our initial discretization via minimum sample sets was 9.94×10^{11} models; and our subset using repeated-uniform sampling required $147,364$ models to have a success probability of 0.99. Needing only 467 samples for a 0.99 success probability is a terrific improvement. We present the number of iterations necessary to achieve a 99% success probability as a function of $P(C)$ in Table 3.1.

Although it provides insight into our problem, simply uniformly sampling within a radius is problematic. If there is noise and if we exclusively sample points that are close to each other, it will be difficult to accurately assess the true parameters of the model; further, the approach will not work if an insufficient number of points are within the radius. Nonetheless, the insight that closer points

Table 3.1: Number of iterations required to have 0.99 probability of success as a function of $P(C)$

$P(C)$	M	$P(C)$	M
0.05	147364	0.25	1177
0.15	5456	0.3	680
0.2	2301	0.35	428

are likely to share the same class proves useful in practical problems.

Kanazawa et al. first introduced this concept in [19] for the purpose of plane detection with a RANSAC-style procedure. Given a set of correspondences $D = \{(x_1, y_1, x'_1, y'_1), \ldots, (x_n, y_n, x'_n, y'_n)\}$, an initial correspondence (x_i, y_i, x'_i, y'_i) is selected uniformly. Then, the remainder of the minimum sample set is drawn so that a point (x_j, y_j, x'_j, y'_j) is chosen with probability determined by a distribution similar to a Gaussian over the squared distance between (x_j, y_j) and (x_i, y_i) (i.e., using the locations in only the first image). Following the parametrization of [32, 44], the probability of picking point $\mathbf{p}_j = (x_j, y_j)$ having picked point $\mathbf{p}_i = (x_i, j_i)$ is given by:

$$P(\mathbf{p}_j \mid \mathbf{p}_i) = \frac{1}{Z} \exp\left(-\frac{(x_j - x_i)^2 + (y_j - y_i)^2}{\sigma^2}\right) \tag{3.9}$$

where the value of Z, the normalization factor, is set to $\sum_{j=1}^{n} P(\mathbf{p}_j \mid \mathbf{p}_i)$ to make $P(\mathbf{p}_j \mid \mathbf{p}_i)$ a valid probability mass function.

This is short-hand for a 2-dimensional isotropic Gaussian. Suppose we have a joint density function P' for (x_j, y_j) with mean $\boldsymbol{\mu} = [\ \ x_i\ y_i\ \ ]^T$ and covariance $\boldsymbol{\Sigma} = \sigma^2 I_{2x2}$. As the x and y dimensions are independent, we can rewrite $P'(\mathbf{p}_j \mid \mathbf{p}_i)$ as the product of two 1-dimensional Gaussians:

$$P'(\mathbf{p}_j \mid \mathbf{p}_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_j - x_i)^2}{2\sigma^2}\right) \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_j - y_i)^2}{2\sigma^2}\right) \tag{3.10}$$

which can naturally be combined to:

$$\begin{aligned} P'(\mathbf{p}_j \mid \mathbf{p}_i) &= \frac{1}{2\pi\sigma^2} \exp\left(\frac{-(x_j - x_i)^2 - (y_j - y_i)^2}{2\sigma^2}\right) \\ &= \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(x_j - x_i)^2 + (y_j - y_i)^2}{2\sigma^2}\right). \end{aligned} \tag{3.11}$$

This is equation 3.9 up to the normalization factor and with a slightly adjusted σ. However, when sampling, we can multiply each of the selection probabilities with a constant since each probability retains the same proportion of the total probability.

The only parameter to be chosen is σ, which controls the scale of the sampling: if it is set too high, then the sampling strategy becomes effectively uniform; if it is set too low, then only the very nearest neighbors of the first point sampled are likely to be included. Various schemes have been proposed, ranging from heuristic settings [32] to satisfying an identity relating the average distance with the probabilities of selection [19, 44].

Kanazawa sampling can be applied in non-homography model estimation domains as well. Toldo and Fusiello recognized its utility for the detection of geometric objects from 2D points [32], and in general, if model membership and proximity are correlated, then Kanazawa sampling may be beneficial. In tasks where there is salient information about model membership in the image, it may be possible to further refine sampling or even supplant the standard distance function entirely: in [24], a segmentation of medical image data provides an alternative distance function for sampling that encodes the underlying dense data.

The more complicated distribution used in Kanazawa sampling necessitates a different estimate for the number of models to sample. An estimate may be found in [32], using the average inlier-inlier distance and inlier-outlier distance. In the end, however, only the approximate value of the estimates is important in any application: one cannot provide the actual inlier-inlier distance in a practical

problem, but rather an approximation, as doing so would necessitate solving the problem itself.

3.1.3 Conclusions about RANSAC

Before moving on to an application of robust model estimation, we pause to summarize our presentation of RANSAC.

Given a set of data points, we have exchanged the goal of minimizing squared residuals for the goal of maximizing the size of the consensus set: the former works well when there are no outliers but fails in their presence; the latter works well in both cases. In exchange for a tractable approach, which requires on the order of a few hundred to a few thousand iterations, we have to provide some additional information: an inlier threshold ϵ, which defines the consensus set, and in some cases, a method of sampling points that performs better than uniform selection. The method is not guaranteed to work and is based on a heuristic, but as was argued in the analysis of Least-Median-of-Squares regression at the beginning, guarantees of performance are often of limited use if they come with inapplicable assumptions about the composition of the data.

RANSAC is not, however, the only robust single model estimation algorithm. A number of variants, mainly aimed at increasing efficiency are discussed in Szeliski [30]. Other authors have focused on improving performance, including approaches such as the Maximum Density Power Estimator [40], Adaptive Scale Kernel Consensus [39], and the pbM-Estimator [4]. Surveys and limited evaluations of the single-model case may be found in [40, 39], and [10] provides a full evaluation of a number of the single-model approaches.

(a) Image 0 (b) Image 1

Figure 3.4: Two views of a scene

3.2 RANSAC-based Plane Detection

We now provide an application of RANSAC to the detection of planar surfaces: given a set of noisy and outlier-contaminated correspondences between two images, we can detect planar surfaces by detecting a homography with RANSAC. In this Section, we briefly discuss the detection of both the correspondences that form the data points for the model-estimation task and the planar surfaces that form the output.

Consider the two images in Fig. 3.4. A human is able to very rapidly recognize that both images depict the same scene and determine which parts of one image correspond with which parts of the other. Accomplishing this in the way a human does is exceedingly difficult for a computer: to a computer, an image is nothing more than a matrix of light intensities and thus a computer cannot make use of the intuitive understanding of a human. Codifying this intuition would be so difficult a task that it would make the subject of this thesis appear as easy as arithmetic. We instead choose a more mechanical approach that can be easily performed by computers.

We use the following approach. Suppose we could find a class of features re-

peatedly, even under a variety of transformations of the image (e.g., a viewpoint shift). Matching similar parts of two images might then be as simple as finding the overlap in features in both images: if the same features can be found in images regardless of transformations, then if two images share common data, we will find some common features. As an unrealistic analogy, if one could extract enough information about stars from images of the night sky to reliably distinguish individual stars (e.g., Betelgeuse and Eta Carinae), then one could, subject to some conditions, use the overlap in stars between two images to compute information about the relationship between the two views.

We summarize a few qualities that we want in our features. We should be able to detect features if they are present in the image: if we cannot reliably detect the same features, then we cannot reliably find overlap. Further the features should be easily localized: if we do not have accurate data about the location of a feature, then it will be difficult to extract any meaningful information from the overlapping features. Finally, it should be easy to figure out when features extracted from two different images are the same, or probably the same.

As should be expected, we cannot satisfy these constraints perfectly but must instead settle for doing a satisfactory job, which can be accomplished with a class of features known as keypoints [30]. Szeliski and other authors naturally divide the feature extraction task into two parts: detection, when feature locations are detected, and description, when the image at these locations is encoded. Our qualities then naturally map to these two steps: our detector should detect the same features at precise locations and our descriptor should encode them in a way where equality can be rapidly and effectively determined. A variety of techniques exist for both steps; one early successful approach is Lowe's Scale-Invariant Transform (SIFT) [23]; a higher-level survey is contained in [30]. There are two commonly

used end-to-end techniques which package a feature detector and descriptor together, namely the aforementioned SIFT due to Lowe [23] and Speeded Up Robust Features (SURF) due to Bay, et al. [2]. Although the diagrams throughout this section are generated using SIFT, there is enough similarity between SIFT and SURF at the higher conceptual level that most of the discussion applies to SURF as well.

A comprehensive description of how SIFT detects and encodes features is beyond the scope of this thesis. We instead seek to provide an intuitive high-level overview of each step. We highly recommend [23] to the interested reader, as it provides a well-written and remarkably accessible and enjoyable description. We begin our presentation with the notion of image scale, which is integral to both SIFT's detection of keypoints and its encoding of their descriptors in a way that permits the matching of keypoints, even if the scene's overall scale changes dramatically.

Suppose one took a picture of a car at a distance of 1 meter, 20 meters, and 100 meters from the same angle (i.e., the photographer walked directly backwards). If we look at the 25×25 pixel patch of each image depicting the same portion of the car, there will be dramatically different data in the box. At 1 meter, we may have a portion of a letter; at 20 meters, we may have the license plate box and a quarter of the back of the vehicle; and at 100 meters, we may have the vehicle itself. Different scales of an object will then pose significant problems to detecting the same features.

To permit the matching of images at different scales, both SIFT and SURF examine the image at different scales. These intuitively correspond with the apparent scale of objects in the real world. Given an image, we cannot fill in details to imitate a closer view (i.e., a lower scale). We can, however, imitate the effect of

viewing the scene from further away (i.e., a higher scale) by repeatedly smoothing and downsampling: a series of images, termed an octave, is computed by repeatedly smoothing an image but not downsampling; after the requisite number of increasingly smoothed images are computed, the image is downsampled, and a new octave is started. Smoothing is achieved by convolution with a Gaussian with a given scale (for more on convolutions and image filters, see Szeliski [30]).

A number of images from different octaves appear in Fig. 3.5. For visualization purposes, the images are equally sized; however, to communicate the relative size in terms of pixels, a square (with 4 bins inside) with constant width of 20 pixels with respect to pixels has been placed in the image. If we can come up with a feature detector and descriptor that will function effectively at different scales, all we have to do is search across a variety of scales to readily handle the aforementioned problem of different image data.

Given a collection of scale-space representations of an image, SIFT computes the difference between the images in each octave with adjacent smoothing parameters. Thus, if one visualizes an octave as a stack of o images with increasing smoothness, SIFT computes the difference between images in the stack. This produces $o - 1$ difference images. The keypoints detected by SIFT are extrema in these difference images: if we index the difference images of an octave as D_i, then x, y is a keypoint in image D_i if it is the minimum or maximum of the pixels in the $3x3$ square centered at x, y in D_i, D_{i-1}, and D_{i+1}. After a number of filtering steps, for instance, to ensure that the keypoint is sufficiently larger or smaller than its neighbors, these scale-space extrema are the keypoints provided by the detector portion of SIFT.

The locations of SIFT features in our example image pair is presented in Fig. 3.6. Note the tendency of keypoints to appear where there is lots of tex-

(a) Original (b) Scale 1

(c) Scale 2 (d) Scale 3

Figure 3.5: A visualization of the scale-space: for visualization purposes, the images have been made the same size and to communicate the relative sizes in pixels, a square with fixed width in pixels has been placed in the center of each image.

ture, and for keypoints to be virtually missing where there is little to no texture (e.g., the sky or roof).

Given a feature location and scale, SIFT encodes the surrounding region of the image by constructing a histogram of the gradient orientations at that particular scale in a region whose size is fixed across scales; this binning system is roughly represented by the squares in Fig. 3.5. As we argued earlier, the use of scale-space representations surmounts the problem of the different scales, thus enabling SIFT to be scale-invariant (i.e., its features can be detected in two images regardless of scale). Rotational invariance is accomplished by detecting an orientation of the patch at the given feature location; the square in Fig. 3.5 is then rotated in a way such that the rotation relative to the underlying image data is identical if the image is rotated.

The histogram computed by SIFT is turned into a vector in a high-dimensional space, ($\mathbb{R}^{128}$) by concatenating the bins of the histogram. Intuitively, keypoints

(a) Image 0 (b) Image 1

Figure 3.6: Two views of a scene: SIFT keypoint locations are drawn with crosses having size proportional to keypoint scale

representing the same location in the real world will have very similar histograms, and thus descriptor vectors. However, we cannot guarantee that a strict equality will hold, and if we demand strict equality, we will erroneously misclassify many good matches as not being equal. Consider then the Euclidean distance metric between two vectors which, given $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^n$, is defined as

$$ED(\mathbf{x}, \mathbf{x}') = \sqrt{\sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{x}'_i)^2}\,. \tag{3.12}$$

It seems sensible to compute the Euclidean distance between two vectors to get a dissimilarity metric: large dissimilarity in histogram bins is penalized more heavily on account of the squares, which roughly corresponds with what might be desirable. Thus, although we cannot simply check equality in each entry, we have a way of representing how similar two features are.

We now turn to the problem of matching the two images presented in Fig. 3.6 using the displayed keypoints.

Having run SIFT (or SURF) on two images, we have a collection of keypoints. It is helpful to think of each of these as a 2-tuple in $\mathcal{L} \times \mathcal{D}$, where $\mathcal{L}$ is the location space ($\mathbb{R}^2$) and $\mathcal{D}$ is the descriptor space ($\mathbb{R}^{128}$). Let us denote $I_0 \subset \mathcal{L} \times \mathcal{D}$ as the set

of keypoints in image 0, and similarly I_1 as the set of keypoints in image 1. We now wish to, as we stated in the beginning, find their overlap. Although the location component of the keypoints will be critical for making sense of the images, the locations of the keypoints do not help us match keypoints without prior knowledge of the scene. Instead, we must restrict our search to the descriptor space. The problem is that we do not have an equality function, but instead a dissimilarity measurement. To find the matching keypoints, we must transform this dissimilarity into a notion more like equality.

It is tempting to either geometrically partition the descriptor space into equivalence classes, or to define a strict cutoff on distance for equality. The first is problematic simply because it requires drawing boundaries: if we partition the descriptor space into non-singleton equivalence classes, then there will be keypoints on either side of borders that are as close or closer to each other that other points within their respective equivalence classes. The second will not work for more subtle reasons; Lowe [23] says that this is empirically ineffective, since "some descriptors are much more discriminative than others": in effect, while descriptor distance corresponds roughly to feature similarity, the same distance corresponds to different degrees of dissimilarity for different pairs of keypoints.

The solution presented in [23] relies on the idea of distance, but uses a data-dependent threshold. In short, for every keypoint in one image, we match it with the keypoint in the other image that is closest with respect to the descriptor distance, and accept the match if the next nearest keypoint in the second image is far enough away (described as a ratio d_r). The algorithm is presented in Fig. 3.7.

Here, $\mathrm{DD}(\cdot, \cdot)$ is the descriptor distance between two keypoints. Intuitively, if a keypoint in the other image is a really the match, then it will be the best match by a large margin since it will be very similar; if a keypoints is not a particularly

SIFTMatch(I_0, I_1, d_r)

// Return the matches in I_1 of the keypoints in I_0
// Go through I_0 and find the matching elements of I_1
$Matches = \emptyset$
for $k \in I_0$
    // Search for the nearest and second-nearest neighbors
    $NN = \arg\min_{k' \in I_1} \mathrm{DD}(k, k')$
    $NN2 = \arg\min_{k' \in I_1 - \{NN\}} D(k, k')$
    if $\mathrm{DD}(k, NN) / \mathrm{DD}(k, NN2) < d_r$
        // If the best-match is distinctive enough, accept the match
        $Matches = Matches \cup \{(k, NN)\}$
return *Matches*

Figure 3.7: The SIFT keypoint-matching procedure

good match (and probably incorrect), then other features will be similarly close. The value of d_r can be tuned to a value between 0 and 1. Lowe suggests a value of 0.8 in [23] as effective, although depending on how many outliers can be accepted and how many true correspondences must be present, another value may be better. A lower setting will reject more incorrect matches, but while also rejecting more correct matches; a higher setting will reject fewer correct matches while rejecting fewer incorrect matches. As d_r approaches 1, every potential match is accepted.

This algorithm produces a collection of matches, or corresponding parts between the images. These matches are visualized in Fig. 3.8. As can be seen in the structure of the lines, the SIFT matches include many actual matches that reflect the underlying structure of the scene; however, it is also immediately obvious that some false matches are included as well: look for lines that disagree with the general trend.

As we articulated in Chapter 2, we can fit a homography to describe the transformation of a planar surface. We asserted that RANSAC was capable of detecting a model, even in the presence of outliers and pseudo-outliers, and thus Fig. 3.8

Figure 3.8: SIFT matches between images

seems to be a good test data set for RANSAC: it has outliers and multiple planar surfaces.

As a review, recall that we repeatedly sample minimum sample sets to which we exactly fit models; we keep track of the model that has the largest consensus set (points that match the model within a given bound) seen so far. When we have performed this a given number of times, we return the largest consensus set we have seen and its model. Since we are applying RANSAC to a particular problem domain, we also need to fill in the details of the requisite functions. Our sampling function, S, is Kanazawa sampling in one image; our estimation function, E is given by the exact solution to the system of equations in Equation 2.29; the error function, R is given by the Euclidean distance between the projection of the location in image 1 under the homography and the location in image 2. We set the number of iterations to 1000 and the inlier threshold, ϵ, to 2 pixels.

As can be seen in Fig. 3.9, the results are quite good: the correspondences selected by RANSAC overwhelmingly belong to one of the dominant planar surfaces. A few correspondences selected do not fit the intuitive interpretation of the plane. However, note that the homography does not incorporate any knowledge about the underlying dense scene information. Thus, although from the point of view of homographies, the points represent a coherent model, in our particular ap-

Figure 3.9: RANSAC results

Figure 3.10: SIFT matches, less the RANSAC results

plication of model estimation, homographic coherency is strongly correlated, but not equivalent to planar coherency.

As an aside, in the particular case of planar surface detection from correspondences, outliers with respect to location may be eliminated. In [16], (also used in [25]) a triangulation of the points was used to eliminate such spurious outliers by location: given a group of points that fit a homography, the Delaunay triangulation of the correspondences' locations in one image is computed; edges with length more than one standard deviation above the mean are removed, and the one or more resulting disconnected subgraphs are treated as separate planar surfaces (with insufficiently large clusters being discarded).

We conclude this chapter with the presentation of our original figure depicting SIFT matches with one adjustment: we subtract the correspondences detected

by RANSAC. This is shown in Fig. 3.10. The removal of the initial matches has significantly thinned the density of correspondences on one of the surfaces, but (as the result of RANSAC matches a single plane), the remaining planes' correspondences are retained.

The presentation of the remaining correspondences may motivate an observation: if we were able to detect a structure from data with W structures, should we then not be able to detect a structure from data with $W-1$ structures? This observation motivates the first algorithm that we present in the next chapter, Sequential RANSAC.

CHAPTER 4

OUTLIER-ROBUST MULTIPLE MODEL ESTIMATION

As was suggested at the end of the last chapter, we should be able to repeatedly apply RANSAC to extend RANSAC to the multiple model case. This is conventionally referred to as Sequential RANSAC [44, 43, 32, 9], and is such a simple concept that no particular author is given credit for its invention.

Starting in 2005, considerable interest arose in whether this approach could be improved upon[1]. Zuliani et al. noticed a shortcoming of Sequential RANSAC and introduced a data set that illustrated it and an algorithm, MultiRANSAC [44], that they claimed corrected it. Although it makes a number of important changes, MultiRANSAC remains fundamentally tied to RANSAC's view of the model-estimation problem as it uses criteria defined by consensus sets to find its models.

More recent authors have moved away from searching using consensus sets, in favor of other criteria. In this chapter, as we introduce later algorithms, we discuss shortcomings of formulating model estimation with consensus sets. The approach of sampling models with minimum sample sets is retained in these algorithms; however, later developments use the models as a method of making approximate inferences about the model space (akin to Monte Carlo methods) rather than as candidate models; Kanazawa sampling then becomes akin to a data-driven prior on models.

Zhang and Kosecká proposed Residual Histogram Analysis [43] in 2006, which searches for models using histograms of residuals. Although their method has been

[1]Mean Shift [11], and a number of robust single model techniques, such as MDPE [40] or MLESAC [35] which may be applied in sequence, were proposed before 2005. Nonetheless, Mean Shift was not originally applied to model estimation, and MDPE and MLESAC are single-model estimation algorithms. Thus the focus on general multi-model estimation algorithms effectively began with Multi-RANSAC.

superceded by other algorithms with respect to performance, as we will see in Chapter 5, the concepts behind it are crucial in the development of later methods, and the attempt to minimize reliance on a-priori knowledge about that data set has been duplicated in later methods. Toldo and Fusiello introduced J-linkage [32] in 2008; the conventional approach of RANSAC is inverted, and models are found by clustering points bottom-up according to which points they match. In recent years, multi-model estimation techniques have become even more removed from the original approach of RANSAC: Chin et al.'s Kernel Fitting [9] (2009) performs model estimation with a variety of linear algebra techniques applied to the output of a kernel function. In this chapter, we discuss each of these algorithms in turn.

Although they are not covered in this thesis for space considerations, there are a number of other methods that deserve mention. Comaniciu and Meer's mean shift technique [11], which has applied to tasks such as filtering and segmentation, can also be used for multi-model estimation. Delong et al. recently posed multi-model estimation as an energy fitting problem in [13]. Finally, any other outlier-robust model estimation technique mentioned in Chapter 3 may be used in sequence for multi-model estimation.

Two strong themes in multi-model estimation will be ubiquitous in our presentation of these techniques. The first, as was hinted at before, is that there is a focus on finding a space to search that is both effective and convenient. Recall that although we were able to provide a closed-form solution for outlier-free problems, we had to resort to a technique that amounted to educated guessing when we introduced outliers and thus no longer had a mathematically convenient function to optimize. Although consensus sets and model-parameter spaces are easy to comprehend, we will see that many newer approaches seek solutions in alternative ways and in fundamentally different spaces.

The second is that there will be a focus not only on how well approaches perform, but also on how many parameters need to be supplied. The motivation for this is mundane: if a technique requires a parameter which cannot be provided in a particular application, then the technique cannot be used. These techniques often need to operate in an unsupervised fashion, in which poor performance due to incorrect assumptions or guesses about the data cannot be corrected by a human. For instance, in [16], the authors decided to use J-linkage instead of MultiRANSAC since in their particular problem of finding planar surfaces, the number of planes could not be specified a priori as required in MultiRANSAC.

Although it seems tempting to simply count the number of parameters required by an algorithm to assess how much prior information or tuning it will need, such logic is fundamentally flawed. If the algorithm does not require a data set statistic, for instance the inlier threshold ϵ, but instead has some internal parameter that controls the same mechanisms, using the algorithm becomes more difficult: rather than requiring a straight-forward data set that can often by provided a-priori by manually looking at the range of input data, one has a complicated parameter whose impact on the output is difficult to assess. In short: although requiring less prior knowledge is useful, it is not helpful if it makes it more difficult to use an algorithm. We thus not only report the input parameters (such as the inlier threshold and number of iterations), but also any internal parameters that need to be set and which might affect performance.

Table 4.1 gives the notation of the various parameters that are used by the algorithms. To whatever degree possible, the notation is consistent with existing literature; this is inevitably not always possible, and there may be conflicts. An internally consistent notation has then been chosen to facilitate comparison rather than mimic the notation of each.

Table 4.1: Notation for algorithm parameters

Symbol	Meaning
ϵ	The inlier threshold; determines if a point is in a consensus set
W	The number of models in a data set
T	A lower bound on the size of a consensus set
N	The number of data points
M	The number of randomly sampled minimum sample sets

SequentialRANSAC($DataPoints \subset \mathcal{D}$,$M \in \mathbb{N}$, $\epsilon \in \mathbb{R}^+$, $W \in \mathbb{N}$, $T \in \mathbb{N}$)

```
1  // Keep track of the models we have found
2  Models = {}
3  while (!StoppingCondition)
4        // Get a good model with RANSAC
5        BestCS = RANSAC(Data, M, ε)
6        if SatisfactoryModel(BestCS)
7              Data = Data − BestCS
8              Models = Models ∪ {BestCS }
9  return Models
```

Figure 4.1: The SequentialRANSAC algorithm

We begin the material by continuing where we left at the end of the last chapter, the sequential application of RANSAC.

4.1 Sequential RANSAC

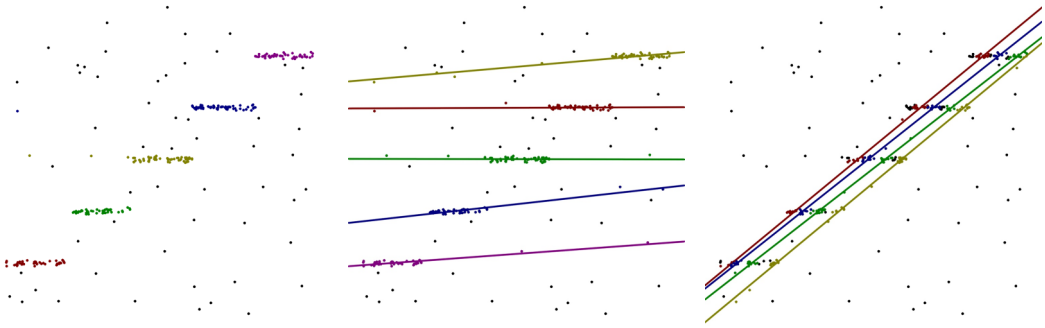
As was discussed earlier, we can apply RANSAC sequentially to yield a collection of models. Since there is no one clear originator of Sequential RANSAC, there are naturally a number of different approaches. We will attempt to capture some of the varieties in our presentation while maintaining the domain-independence of the approach, and suggest reasons why each might be superior in any given situation.

We begin with the algorithm itself, which we present in Fig. 4.1. In addition to the usual bold-face for keywords (e.g., **if**, **while**), we have indicated portions of the code where authors differ with boldface. We present a sample of various

implementations, and how they correspond to the above algorithm. In [19], although the RANSAC step is slightly modified from the usual form of RANSAC, the model-acceptance test is a simple consensus set threshold, and the algorithm terminates when a model has been rejected; this consensus-set-size threshold is implicitly adopted in the implementation used in the evaluation section of [32]. In [38], there is no acceptance test, and the algorithm's termination is not explicitly given. In [44, 9], no acceptance test is similarly given, but from other parts of the paper, it may be deduced that the algorithm terminates when W models have been found.

Thus, one can formulate the stopping and acceptance conditions as a function of either W or T. Depending on the proposed usage and available parameters, either parametrization may make more sense. We note briefly that if T is the minimum ground-truth consensus set size, then the threshold should be set to αT for some α with $0 < \alpha < 1$: RANSAC may not find precisely the same data points as the ground-truth, and requiring it to achieve this will result in the improper rejection of nearly perfect models.

Zuliani, Kenney and Manjunath recognized a shortcoming of this sequential approach: poor model estimates in one round may degrade estimates in a later round [44]. To demonstrate this, they produced a data set, known as the stairs, which is potentially ambiguous in interpretation. An instance of this data set with 5 stairs (Zuliani et al. originally used 4), and two possible interpretations are presented in Fig. 4.2. In Fig. 4.2(a), the underlying data is presented; each point's ground-truth label is represented with a color. The interpretation that is obvious to a human is presented in Fig. 4.2(b) in which the estimated models correspond with only one ground-truth model. The "incorrect" interpretation is presented in Fig. 4.2(c): each of the estimated models lies across each ground-truth model.



(a) Data (b) Correct Interpretation (c) Incorrect Interpretation

Figure 4.2: The stairs data set and two interpretations

Sequential RANSAC, Zuliani et al. observed, tended to produce results falling into the "incorrect" category.

One of the primary causes of this behavior is RANSAC's sole criterion of maximizing the consensus set[2]. Although we suggested earlier that consensus set size was a good measure of model validity, using this as the only criterion seems shortsighted. Consider Fig. 4.3, in which we present 50 models generated via Kanazawa sampling with $\sigma = 0.05$. If geometric distance is correlated with model membership, then it perhaps might make sense to consider not only the size of a model's consensus set, but also, in some sense, how likely the points in the consensus set are to share a model. The points in the consensus sets of the correct models in the stairs data set, informally speaking, share a larger number of models in common than the points in the consensus set of the incorrect models. We will later formally capture some of this intuition in our discussion of J-linkage in Section 4.4, which moves away from consensus-set-oriented approaches.

This problematic behavior is further exacerbated by the sequential approach, as observed by Zuliani et al. in [44]. The initial selection of an incorrect model, such as an incorrect stairs model, contributes heavily to later failure in model

[2][40] provides an interesting discussion of the two approaches of either maximizing the consensus set (i.e., RANSAC) or the goodness of fit (i.e., Least Squares) and its proposed algorithm purports to do both.

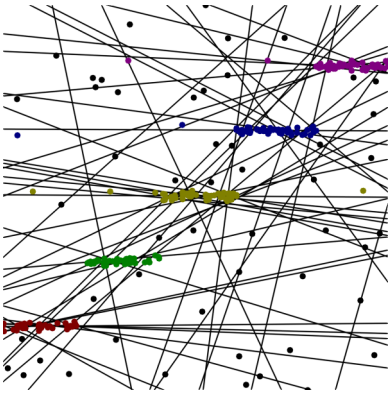


Figure 4.3: 50 models sampled with Kanazawa sampling ($\sigma = 0.05$)

estimation. As an illustration, suppose we have a stairs data set with 5 stairs with 50 points per model. Then, suppose that our initial model estimate from RANSAC spans the stairs, including 15 points from each stair step. Then, when we update our data set by removing the consensus set of the model, we have five models with 35 points each. Now, suppose we are considering either a "correct" or "incorrect" interpretation. The correct interpretation of each model will have only 35 points; thus, if one can find a stair-spanning model that has, on average, more than 7 points per stair, then the "incorrect" interpretation will be selected for a larger consensus set again. This further reinforces the incorrect interpretation.

4.2 MultiRANSAC

Zuliani, Kenney, and Manjunath argued that Sequential RANSAC's failure was rooted in its sequential nature and that a parallel approach, which they termed MultiRANSAC [44], was superior. Intuitively, MultiRANSAC searches for the best collection of W models by iteratively updating a collection of W models with W new minimum sample models using a fusion procedure that merges collections of consensus sets. Crucially, MultiRANSAC enforces that the W models have disjoint

consensus sets as a way of ensuring that W distinct models are captured. To see how this might be necessary, consider the stairs data set, but where one model has twice the number of points. One might be able to find two models that capture 60% of the large ground-truth model with two different models which capture slightly different subsets. Then, if there are 5 models to be estimated, one might get larger consensus sets by selecting two of these models, followed by three models from the remaining 4 ground-truth lines. If we require disjoint consensus sets, then this can be avoided entirely: having selected the larger of the two models fitting the largest ground-truth model, we are unable to select another model that corresponds with the same ground-truth model as it would share points with our initially selected model.

We now fill in the lower-level details of MultiRANSAC. We first describe the procedure by which two collections of W consensus sets may be combined. The merged set of consensus sets is constructed by taking the current $2W$ consensus sets in descending order of cardinality, subject to the constraint that the final set must be disjoint. This algorithm is depicted in Fig. 4.4.

There are two points of divergence between the above and the description found in [44]; we address them here to assuage the reader. Line five of UpdateCS, as described in [44] should read *every* consensus set, rather than *any*, as described by the preceding paragraph. Further, the description of UpdateCS does not address what should be done if no disjoint fusion can be achieved; here, we have elected to simply return the consensus set that we are updating. In practice, while the fusion failure case is by no means common, it is common enough that it needs to be acknowledged.

The main body of MultiRANSAC is similarly straightforward, and detailed in Fig. 4.5. W minimum sample sets are repeatedly drawn disjointly, and UpdateCS

```
UpdateCS(CurrentCSS ⊂ 𝒟, ProposedCSS ⊂ 𝒟, W ∈ ℕ)
 1  // Update the set of W current consensus sets, CurrentCSS, using
 2  // a new set of W consensus sets, ProposedCSS
 3  NextCSS = ∅
 4  for CS ∈ (CurrentCSS ∪ ProposedCSS) in decreasing cardinality
 5        // Stop when we have W models
 6        if |NextCSS| == W
 7              break
 8        // If we can add CS while maintaining pairwise disjointness, add it
 9        if CS ∩ CS' == ∅  ∀ CS' ∈ NextCSS
10              NextCSS = NextCSS ∪ {CS}
11  // If we fail, return the best-so-far set
12  if |NextCSS| < W
13        return CurrentCSS
14  return NextCSS
```

Figure 4.4: The UpdateCS subroutine

is used to fuse a "running-best" consensus set collection with the newly drawn set. A bound on the number of iterations is computed using the collection of consensus sets.

MultiRANSAC requires prior knowledge of the number of models, W, in order to function; however, in exchange, it is also able to automatically estimate the number of iterations required for a given probability of success. To do this, it uses the size of the current consensus sets $N_{I,1}, N_{I,2}, \ldots, N_{I,W}$ to estimate the size of the ground truth consensus sets. The probability, according to [44], is then a function of the current consensus sets, number of models W, and number of data points N,

$$q = \frac{\binom{N_{I,1}}{MSS}\binom{N_{I,2}}{MSS}\cdots\binom{N_{I,W}}{MSS}}{\binom{N}{MSS}\binom{N-N_{I,1}}{MSS}\cdots\binom{N-\sum_{w=1}^{W-1}N_{I,W}}{MSS}}. \tag{4.1}$$

The key to understanding Equation 4.1 is recognizing that $N_{I,j}$ here gives an estimate of the ground-truth model j. The numerator is then the number of ways to pick W minimum sample sets from the ground-truth models. The denominator is approximately the number of ways to pick a collection of W minimum sample

MultiRANSAC($DataPoints \subset \mathcal{D}$, $W \in \mathbb{N}$, $\epsilon \in \mathbb{R}^+$)

```
1  h = 0; Iters = ∞
2  CurrentCSS = ∅
3  while h ≤ Iters
4       h = h + 1
5       // Build a consensus set collection in ProposedCSS, a set of W sets
6       ValidDataPoints = DataPoints; ProposedCSS = ∅
7       for w ∈ 1, ..., W
8            MSSModel = E(S(DataPoints))
9            MSSCS = {d ∈ ValidDataPoints : R(MSSModel, d) < ε}
10           // This ensures disjointness
11           ValidDataPoints = ValidDataPoints − MSSCS
12           ProposedCSS = ProposedCSS ∪ {MSSCS}
13      if CurrentCSS ≠ ∅
14           CurrentCSS = UpdateCS(CurrentCSS, ProposedCSS, W)
15      else
16           CurrentCSS = ProposedCSS
17      Iters = UpdateIter(CurrentCSS)
18 return CurrentCSS
```

Figure 4.5: The MultiRANSAC algorithm

sets in sequence, removing their consensus sets, as is done in the main loop. This is then the probability of drawing a correct collection of W ground-truth models in sequence. One then takes q, and a desired bound on the probability of failure, ϵ_{Fail}, and computes

$$\text{UpdateIter}(CSS, W) = \left\lceil \frac{\log \epsilon_{\text{Fail}}}{\log(1-q)} \right\rceil . \tag{4.2}$$

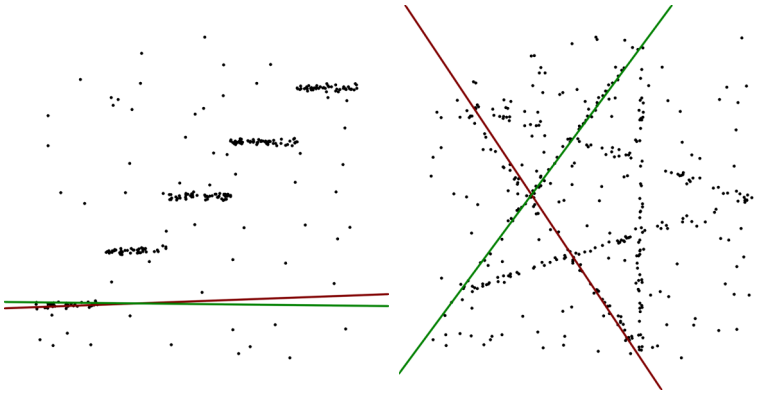
N may be lowered if Kanazawa Sampling is used: the effective number of points one is selecting from at any given time is not N, but some $\hat{N} < N$ since certain points are so improbable for selection that they will not be chosen for practical purposes. However, we found that the method for computing $\hat{N}$ given in [44] sometimes produced impossibly low values. Since the use of $\hat{N}$ can only increase the estimated probability of success on any given round, it can only decrease the number of iterations expected before a successful drawing. If time is not of a

concern, using N should suffice.

The themes discussed earlier – the difficulty of choosing an appropriate space and the problem of necessary a-priori knowledge about the data – reappear in the analysis of MultiRANSAC.

MultiRANSAC's use of consensus sets reveals a fundamental shortcoming of consensus-set-oriented approaches, the difficulty of disambiguation of models. Because multiple minimum sample models are drawn in both Sequential and MultiRANSAC, and each will inevitably have slightly different parameters, we might draw a model redundantly (e.g., Fig. 4.6(a)). Both Sequential RANSAC and multiRANSAC aim to prevent this by enforcing a disjointness constraint. Sequential RANSAC does this by removing inliers after each model has been discovered; MultiRANSAC does this in UpdateCS. Unfortunately, as can be seen in Fig. 4.6(b), ground-truth models may not be disjoint, and a large number of the points may belong to two ground-truth models. One might wish to propose some sort of disjointness-measurement that distinguishes different models intersecting (e.g., see Fig. 4.6(b)) from two instances of the same model with different parameters (e.g., Fig. 4.6(a)). If we are to only use the consensus sets, however, this seems unlikely to succeed: the genuinely different models of Fig. 4.6(b) intersect at more points than the redundant models of Fig. 4.6(a) (6 vs. 3). We will show in a the final section how this disjointness criterion ends up causing MultiRANSAC to fail on the stairs example, illustrating the difficulties of model estimation with consensus sets. If we cannot distinguish two models in the point space, then it might makes sense to work elsewhere; later authors have noted these sorts of issues, and have overwhelmingly moved away from consensus sets.

More mundanely, MultiRANSAC requires a-priori knowledge of the number of models in the data. As was mentioned earlier, in many unsupervised situations,



(a) Two Stairs Models (b) Two Star Models

Figure 4.6: Two manually selected non-disjoint models. Left: intersection of 3 points; right: intersection of 6 points.

this is inappropriate. Nonetheless, in other situations, especially in constrained environments or tasks, this may not be an issue.

4.3 Residual Histogram Analysis

Residual Histogram Analysis (RHA) [43] reformulates the model-estimation task in an alternative space, avoiding both the problematic consensus-set-oriented approach and the requirement of knowledge of data set statistics. Instead of searching for strong models as models with large numbers of points in their consensus sets, RHA finds strong models using peaks in histograms of the residuals of points with respect to a collection of models. The intuition is that the modes in histograms are generated by models corresponding to ground-truth structures.

M minimum sample sets are drawn, and the residuals between each pair of model and points are computed. Histograms are computed for each point's residuals with respect to all the models and smoothed, and modes are sought in each histogram. The number of models is determined by the median number of modes found over all data points. Finally, the actual models are selected from the mode

RHA($DataPoints \subset \mathcal{D}$, $M \in \mathbb{N}$, $T_\tau \in \mathbb{R}^+$)

1 Generate M minimum sample models $Models$
2 $RS[i] = \{R(DataPoints[i], m) : m \in Models\}$
3 Set $Histograms[i]$ as the smoothed histogram of $RS[i]$
4 $Modes[i] = FindModes(Histograms[i], T_\tau)$
5 $ModelCount$ = median mode count in $Modes[1..N]$.
6 // Get the data points that correctly determined the model count
7 $Valid = \{i : |Modes[i]| = ModelCount\}$
8 // Get a data point whose histogram bins determines the models
9 $Strongest = \arg\max_{s \in Valid} \text{PeakStrength}(s)$
10 // and another data point from S
11 $Arbitrary = RandomlySelect(Valid)$
12 **return** ModelDetect($Strongest, Arbitrary$)

Figure 4.7: The main RHA Algorithm

bins of two selected data points. The approach is detailed in Fig. 4.7. This process requires no knowledge of ϵ, W, or T; in exchange, however, it requires seeking modes in a noisy histogram, which is accomplished using a mode-distinctiveness measure.

We now present the details of the RHA algorithm. Generating the smoothed histogram is relatively straight-forward and we instead present an overview of the two complicated steps, the RHA mode-detection algorithm and the model estimation step. We present the mode detection scheme at a fairly high level, since the low-level details render presentation cumbersome and are not conducive to understanding.

We begin with the mode finding algorithm, whose code appears in Fig. 4.8. The 1D-histogram mode finding approach begins with the actual peaks and valleys of the smoothed histogram (i.e., any bin whose neighbors are both smaller or larger), and examines the modes (i.e., peaks) in increasing distinctiveness. The distinctiveness of a mode is defined as the ratio between it, and the shallower (and thus higher) adjacent valley. So, if there is a mode in a histogram $H[1..B]$ (with B

```
FindModes(H[1..B], T_τ)
 1  AcceptedModes = ∅; Extrema = [ ]
 2  for i = 1, ..., B
 3       if H[i] is an extrema (mode or valley)
 4            Extrema = Extrema + [i]
 5  Update Extrema by removing repeated modes and valleys
 6  while there is at least one mode in Extrema
 7       l = the least distinctive mode
 8       // Check if the mode is distinctive enough
 9       if τ(l) > T_τ
10            AcceptedModes = AcceptedModes ∪ {l}
11       delete l from Extrema
12       Update Extrema by removing repeated modes and valleys
13  return AcceptedModes
```

Figure 4.8: RHA's mode finding algorithm

for bins) at j, and the adjacent minima (which are probably not in adjacent bins) are in bins i and k, then the distinctiveness measurement $\tau(j)$ is:

$$\tau(j) = H[j] / \max\{H[i], H[k]\}. \tag{4.3}$$

If a mode is distinctive enough, it is added to an accepted mode list; regardless of its distinctiveness, it is deleted after consideration. Following deletion, the collection of extrema is updated so that modes and valleys alternate: the current extrema are iterated in order and when repeated valleys or modes are encountered in a row, only the strongest (i.e., lower or higher respectively) is retained.

Note that there is a parameter T_τ which controls how many modes are accepted as valid (and thus how many models we estimate). This, in effect, plays the role of T.

Having computed histograms, RHA selects one point from the set of points that correctly estimated the number of models by choosing the one that maximizes a peak distinctiveness measure. Specifically it chooses the point that maximizes the product of the histogram mode bins (thus ensuring a point with many points in

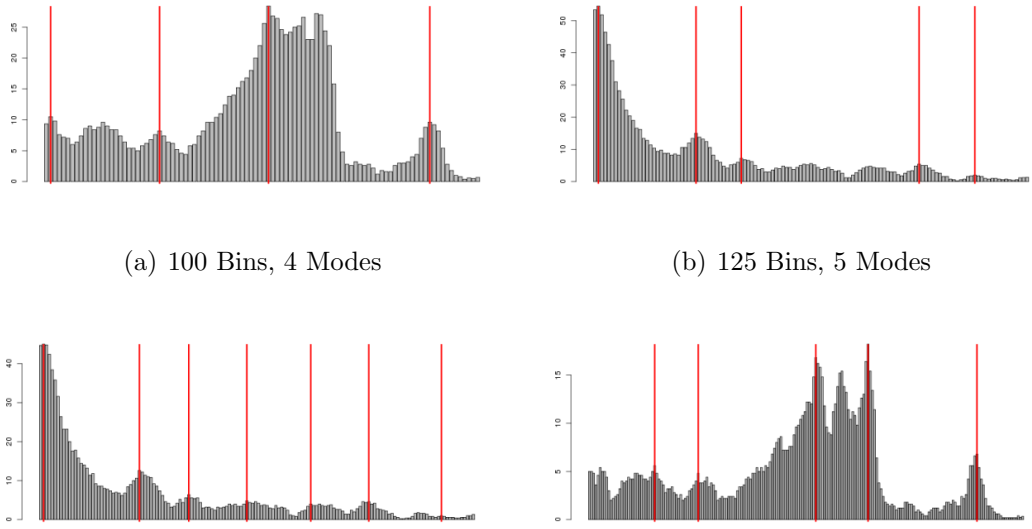
```
ModelDetect(distinctive,Arbitrary)
1  Models = ∅
2  for i = 1,...,Mode Count
3       CandidateModels = Models in to mode i in Histograms[Distinctive]
4       Sort CandidateModels by residual with respect to Arbitrary.
5       Models = Models ∪ {Middle(CandidateModels)}
6  return Models
```

Figure 4.9: RHA's model detection algorithm

each bin). This point and another arbitrarily selected one are used to select the estimated models.

We now present how to detect models from two of these histograms. The mode bins in the histogram of the "strongest" data point provide candidates for the models: recall that the mode histogram bins correspond to ground-truth models and bins contain estimated models. An arbitrary data point that also correctly estimated the number of models is used to select which candidate is used from each of the distinctive point's mode bins: the mode with the median residual among the candidates with respect to the arbitrary data points is selected. This is formally presented in code in Fig. 4.9. The intuition is that a model that the other point fits moderately well with respect to all the other points in the bin is likely to be similar to most of the models in the bin (i.e., the ground-truth model sought).

Residual Histogram Analysis is a fundamental turning-point in the development of multi-model estimation algorithms. In addition to formulating model estimation in a way that avoids explicit consideration of consensus sets, RHA avoids the use of a-priori knowledge about the data set. Nonetheless, the T_τ parameter arguably needs tuning, and different parameters for the construction of the histograms produce different estimates of the number of models. For instance, consider Fig. 4.10, which shows the smoothed histogram bins of the strongest histogram for a variety of histogram bin counts, and the detected modes with red lines. The number of



(a) 100 Bins, 4 Modes

(b) 125 Bins, 5 Modes

(c) 150 Bins, 7 Modes

(d) 200 Bins, 5 Modes

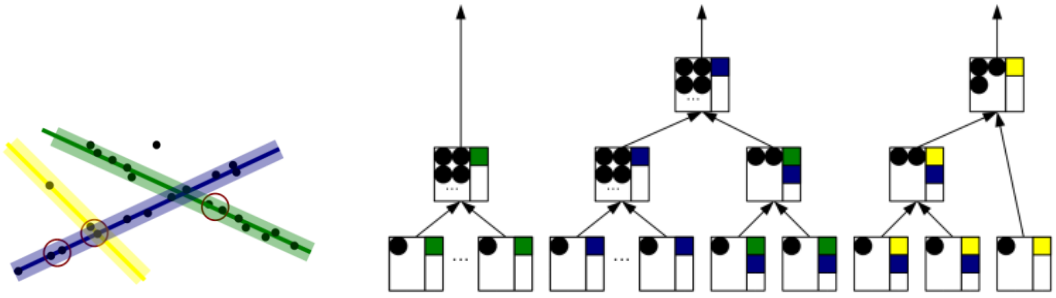
Figure 4.10: Residual histogram with strongest peaks for different histogram bin counts

modes, and thus estimated model count, varies from 4 to 7. As is suggested by the figure and corroborated by Toldo and Fusiello in [32], the mode-finding step of RHA is its weakness. In practice, this means that RHA is not competitive with other methods, which is demonstrated in [43] itself. Nonetheless, it achieves impressive results given that it requires no knowledge of the data.

4.4 J-Linkage

J-linkage [32] takes a cue from RHA, and avoids a consensus-set oriented approach in favor of working in the preference space. Recall that the preference set is like the consensus set, but with the roles of data points and models reversed. Specifically,

$$\mathrm{PS}(d, \mathit{Models}, \epsilon) = \{m \in \mathit{Models} : R(m, d) < \epsilon\}. \tag{4.4}$$



(a) Data and Minimum Sample Models

(b) The J-linkage Clustering Scheme: supporting points are on the left, and the preference set is represented in the box on the right. Clustering proceeds from bottom up.

Figure 4.11: An illustration of the J-linkage clustering scheme

Toldo and Fusiello further define the preference set of a set of data points as the intersection of their individual preference sets, and thus the models that all of them fit.

$$\begin{aligned} \mathrm{PS}(D, \mathit{Models}, \epsilon) &= \bigcap_{d \in D} PS(d, \mathit{Models}, \epsilon) \\ &= \{m \in \mathit{Models} : R(m, d) < \epsilon \ \forall \ d \in D\}\,. \end{aligned} \tag{4.5}$$

Rather than take models and see what points match them, J-linkage uses what models each point matches to determine which points probably belong together. If we have M minimum sample sets, each point d may then be represented by a M-dimensional vector v in the high-dimensional binary space $\{0,1\}^M$, which Toldo and Fusiello term the conceptual space, such that v_i is 1 if and only if model i is in d's preference set. While this way of looking at preference sets is not useful in the practical computation of J-linkage, the understanding of the preference set of a point as an alternative representation is enormously helpful for understanding the model-estimation problem. Further, the understanding of the notion of representing a point in an alternative form will be helpful in understanding Kernel Fitting.

Consider the contrived line fitting example in Fig. 4.11(a), in which three min-

J-linkage($DataPoints \subset \mathcal{D}, M \in \mathbb{N}, \epsilon \in \mathbb{R}^+, T \in \mathbb{N}$)

```
1   Clusters = {{d} : d ∈ DataPoints}
2   Compute M minimum sample models Models
3   while True
4       // Select the pair of clusters with minimum distance
5       A, B = argmin_{A,B∈Pairs(Clusters)} D_J(PS(A, Models, ε), PS(B, Models, ε))
6       minDist = D_J(PS(A, Models, ε), PS(B, Models, ε))
7       // If minDist is 1, then all clusters have disjoint preference sets
8       if minDist == 1
9           break
10      Clusters = (Clusters − {A, B}) ∪ {A ∪ B}
11  return {c ∈ Clusters : |c| ≥ T}
```

Figure 4.12: The J-linkage algorithm

imum sample models have been drawn. Points matching the same collection of models are likely to belong to the same ground truth model, as we observed in our discussion of the shortcomings of the consensus set approach with regards to sequential RANSAC. J-linkage uses this intuition to develop a clustering approach: beginning with singleton clusters (i.e., each point in its own cluster), clusters are merged together in decreasing order of similarity of model preference. This is depicted graphically in Fig. 4.11(b), and in code in Fig. 4.12.

Although we have a large number of points with identical preference sets in our contrived example, if we were to draw 1000 models, we would find very few points sharing identical preference sets, and therefore we need a way of measuring how similar two sets are. Toldo and Fusiello propose the Jaccard distance

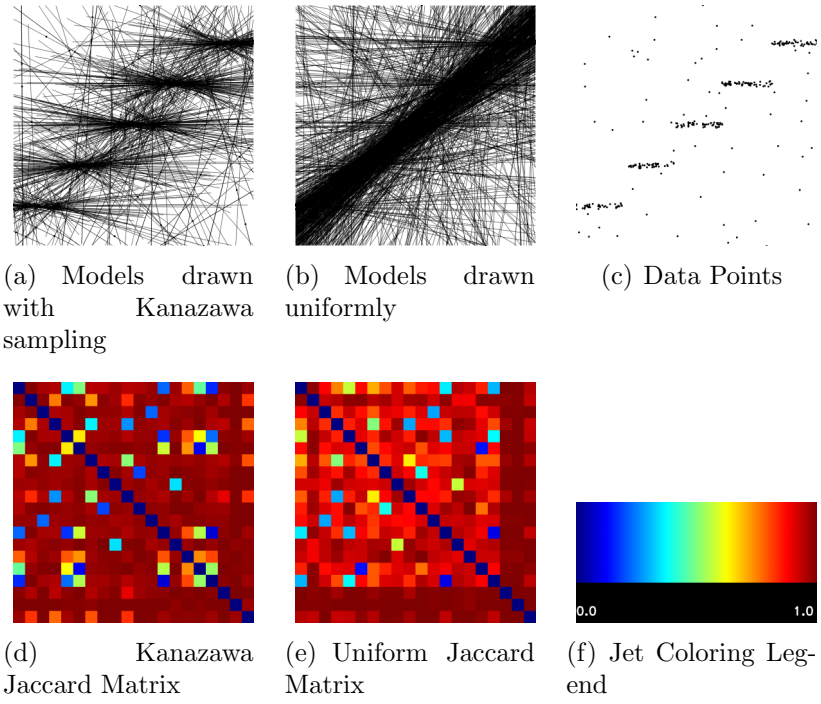
$$D_J(A, B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|} \tag{4.6}$$

which gives the algorithm its name (**J**accard **Linkage**). The Jaccard distance evaluates to 0 for identical sets (as $A \cup B = A \cap B = A = B$) and to 1 for disjoint sets (as $|A \cap B| = 0$); in general, the Jaccard distance is between 0 and 1, inclusive.

When the clustering stops, each cluster has at least one model that fits all of

the points. Further, points corresponding to ground truth models are in larger clusters and points that are outliers are in smaller clusters (as no models fit large numbers of outliers). These clusters are similar to consensus sets; however, consensus sets are formed by taking an actual model and finding points that match it, whereas the clusters are points that fit a group of models (or, in the abstract, of a particular class of models). The smaller outlier clusters may be removed either with a predefined threshold [9] or by removing smaller clusters until the number of removed points equals the expected number of outliers [32]; our presentation takes the former approach. We provide a graphical illustration of the clustering in Fig. 4.11(b): note that points with identical preference sets are always merged first.

Since the preference space is defined by the sampled models, special care must be taken when using J-linkage. In particular, the models that are sampled have an impact on which points are likely to be clustered by changing their preference sets and thus their representation in the conceptual space. In Fig. 4.13, we use different sampling strategies on a data set, and compute the Jaccard distance between each the preference sets of pairs of data points. These are represented as images, where the pixel at row i and column j is colored according to the Jaccard distance between the preference sets of points i and j, using the coloring scheme depicted in Fig. 4.13(f). Note that for both model selection strategies, the distance between a point and itself (the pixels on the main diagonal) is 0; note further that the perceived artifact, the visible rectangle is merely the result of listing inliers before outliers in the data input: inliers are likely to have similar preference sets to each other. For the uniform strategy, the Jaccard distance between points tends to evaluate to a higher value, but leads to few pairs of disjoint preference sets (as seen by the absence of the dark red pixels). In the Kanazawa strategy



(a) Models drawn with Kanazawa sampling

(b) Models drawn uniformly

(c) Data Points

(d) Kanazawa Jaccard Matrix

(e) Uniform Jaccard Matrix

(f) Jet Coloring Legend

Figure 4.13: Different sampling strategies lead to different distances between preference sets: (a,b) Models sampled with Kanazawa and Uniform strategies; (c) The underlying data points; (d,e) A visualization of the Jaccard distance between preference sets of a selection of data points according to the sampled models. (f) the legend for (d,e).

visualization, there are more sets with significant overlap, as well as more that are nearly disjoint. Essentially, Kanazawa sampling sharpens the contrast between points that belong together and background noise. In practice, we observed that J-linkage performs worse in geometric figure fitting if uniform sampling is used. This seems unsurprising, since it was assumed that Kanazawa sampling was to be used in [32].

Despite these issues, J-linkage remains an overall formidable model estimation technique. By avoiding the use of the consensus set representation, it avoids the drawbacks of RANSAC-based techniques. This, however, comes at the cost of a clustering approach, which requires a consensus-set size threshold to distinguish valid models from outlier clusters: if this is set too low, spurious models are re-

ported; if this is set too high, actual models are rejected. Further, it demonstrates $O(N^2M)$ time and space complexity, where N is the number of data points and M is the number of sampled models: each time we merge clusters, we only need to update N inter-cluster distances.

4.5 Merging J-Linkage

J-linkage's requirement that the data points of a cluster have at least one preferred model in common may lead to the fragmentation of models: if no one minimum sample model fits a ground-truth model correctly, then the ground-truth model may be found as two or more separate clusters. For instance, if Kanazawa sampling is used in plane detection, it may be unlikely to find a minimum sample model that both contains only inliers from one model and covers a sufficiently large portion of the support region of the plane to give an accurate estimation of aspects of the perspective transformation[3]. This tendency was noted in [31] where J-linkage was used for the detection of vanishing points from detected edges, and [16] presented a solution to the problem of fragmentation in the context of the detection of planar surfaces from image pairs. In the former, a model-specific merging scheme is formulated with Expectation-Maximization, and in the latter, a model-independent scheme is proposed. We discuss the latter, as it is defined in general.

After J-linkage stops clustering the data points, clustering is restarted with a new distance between clusters: given two clusters A and B of points, a best-fit solution $\hat{m}$ to $A \cup B$ is computed and the average μ is taken of the error of the points, $\{R(d, \hat{m}) : d \in A \cup B\}$. The pair of clusters with the least error is

[3]The information necessary to estimate foreshortening in closely grouped points may be masked by noise; however, on a local scale, incorrect foreshortening estimates will have limited impact on the accuracy of the models.

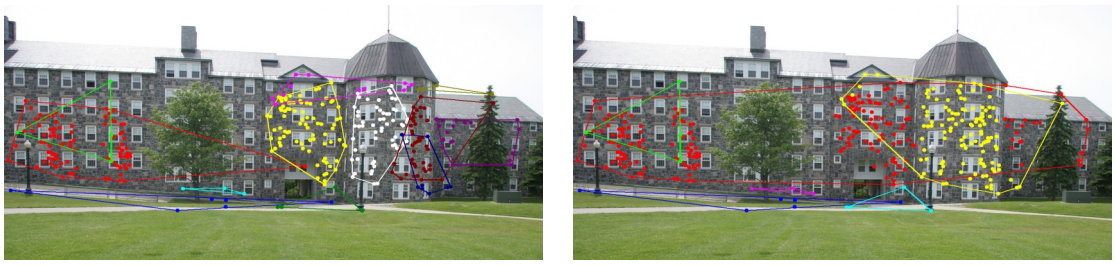
merged until the minimum average merging error exceeds ϵ. This permits clusters of points that have no common model amongst the ones sampled, but for which there is a common model to be merged. Note that there is either no filtering done by consensus set size, or one with a very low threshold; in [16], a consensus set size of 6, barely above the minimum sample size of 4, was required. If a high threshold is used, then one risks the removal of parts of the valid clusters.

We present results of this merging scheme in Fig. 4.14. J-linkage's initial results are displayed in Fig. 4.14(a); note that the red and yellow clusters belong to the same ground-truth plane, but J-linkage cannot find a model that accurately fits all the points in both. The results of the merging scheme are presented in Fig. 4.14(b): points belonging to the same planar surface have been merged together. Note that the yellow points in the merged image function as a planar surface, even though they are multiple physical planes, due to limitations on the viewing angle between the image and the other image in the pair.

We refer to this approach as Merging J-linkage in Chapter 5, when we discuss the evaluation of multi-model estimation algorithms. Since it is defined with the same functions that are used in all the other approaches and requires no more information about the data set than J-linkage, it can be applied to all the tasks that J-linkage can be used on, at least in theory. Nonetheless, in Chapter 5, we find that in practice, its applications outside of plane detection are limited.

4.6 Kernel Fitting

We conclude with Chin, Wang, and Suter's Kernel Fitting [9], which works well, like J-linkage, while requiring no knowledge of the data set parameters, like RHA. This comes at a cost: unlike all the algorithms up to this point, Kernel Fitting relies on sophisticated mathematical techniques, and its implementation without the aid



(a) Initial J-linkage detection

(b) Clustering after J-linkage

Figure 4.14: (a) Plane detection results with J-linkage (b) after the merging scheme proposed in [16]. In each, the points belonging to each cluster are rendered in different colors and their convex hulls are depicted as well.

of a numerical linear algebra library would be ill-advised. This mathematical sophistication seems inevitable: Kernel Fitting must estimate the models without knowledge of the noise scale, outlier count, or model sizes.

Central to Kernel Fitting is Chin et al.'s Ordered Residual Kernel (ORK), a function satisfying criteria which enable the use of a variety of mathematical techniques. The full details of kernel methods are unimportant to an understanding of how Kernel Fitting is performed and we will introduce all concepts necessary; interested readers may wish to consult Chapter 6 of [3][4]. A kernel (in [9], a Mercer Kernel) is a symmetric function $k(\mathbf{x}, \mathbf{y}) = k(\mathbf{y}, \mathbf{x})$, which is paired with another function $\phi(\mathbf{x})$, which maps $\mathbf{x}$ into a feature space F, in which $\mathbf{x}$ is represented in a more salient fashion. If Mercer's Condition holds, the two functions are related by the following identity:

$$k(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^T \phi(\mathbf{y}). \tag{4.7}$$

Thus, if Mercer's condition holds, the actual nature of ϕ and F can be ignored, and we can work entirely using k. In particular, if we let A be the matrix of the

[4]As a whole, the chapter primarily discusses kernels for use with Radial Basis Function Networks and Gaussian Processes; sections 1 and 2 might prove helpful for following [9].

data points $d_1, \ldots, d_N$ mapped under ϕ into F, or

$$A = [\phi(d_1)\ \phi(d_2)\ \cdots\ \phi(d_N)] \tag{4.8}$$

then we may compute $A^T A$ as

$$A^T A = \begin{bmatrix} \phi(d_1)^T\phi(d_1) & \phi(d_1)^T\phi(d_2) & \cdots & \phi(d_1)^T\phi(d_n) \\ \phi(d_2)^T\phi(d_1) & \phi(d_2)^T\phi(d_2) & \cdots & \phi(d_2)^T\phi(d_n) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(d_n)^T\phi(d_1) & \phi(d_n)^T\phi(d_2) & \cdots & \phi(d_n)^T\phi(d_n) \end{bmatrix}. \tag{4.9}$$

Using Equation 4.7, this is the kernel matrix K, such that

$$K = A^T A = \begin{bmatrix} k(d_1, d_1) & k(d_1, d_2) & \cdots & k(d_1, d_n) \\ k(d_2, d_1) & k(d_2, d_2) & \cdots & k(d_2, d_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(d_n, d_1) & k(d_n, d_2) & \cdots & k(d_n, d_n) \end{bmatrix}. \tag{4.10}$$

Therefore, we can work with the feature space representation of the data, A, indirectly via the kernel matrix, K, which we can compute with the kernel function $k(\cdot, \cdot)$. We now discuss the particular kernel of Kernel Fitting, and how it is used to perform multi-model estimation.

Before we begin, we give a roadmap of Kernel Fitting, and thus our presentation. Beginning with N points and M minimum sample sets, a kernel matrix K is computed. A technique is applied to transform the data points into a space which permits the detection of outliers; these outliers are removed, yielding a reduced kernel matrix K'. K' is then used to oversegment the remaining inliers, and a merging scheme is used to reassemble these models into the final model estimates.

We note further that, unlike previous sections, we will not provide much detail. The full treatment of Kernel Fitting requires a lot of notation and we instead provide a detailed overview that captures the overall effect of the approach. Interested readers may wish to consult not only [9], but also [6].

4.6.1 The ordered residual kernel and its computation

Like the other methods, Kernel Fitting samples M minimum sample models, and similarly to J-linkage and RHA, it uses the approach of analyzing data points via their relationship to these models; further, like J-linkage, it uses these models to represent the data points in a more salient way. However, its formulation of the representation fundamentally departs from previous approaches. In J-linkage, an inlier threshold was used to create a binary representation (the preference set, or the vector in the conceptual space); in RHA, the actual residuals were retained, but only in the aggregate in the form of a histogram. In Kernel Fitting, the order in which models are preferred, (i.e., the order of their residuals) is used: points belonging to the same ground-truth models should have similar orders of preferred models. The ORK quantifies similarity in model preference, but does so in a way that satisfies Equation 4.7 and does not, unlike J-linkage, require ϵ: whether a model matches a point according to a particular cutoff is not important; instead, one is concerned with its preference relative to the other models.

Before defining the ORK, we define the Difference of Intersection Kernel (DOIK), from which the ORK is formed. Suppose we have lists defining the order in which two data points d_i and d_j prefer models, $\mathbf{L}_i$ and $\mathbf{L}_j$; this is the representation of a data point in Kernel Fitting. For clarity note that $L_i[k]$ is a natural number giving the index into the sampled models of the model that point i fits kth best. We can then consider a series of adjacent and disjoint windows of size h over the ordered collection of residuals, beginning with the most strongly preferred models. At window number t, we introduce the models with residuals $(t-1)h+1$ to th in order of preference. Given step size h, which Chin et al. assume without loss of

generality[5] divides M, and window number t, the DOIK $k_\cap^t$ is defined as:

$$k_\cap^t(\mathbf{L}_i, \mathbf{L}_j) = \frac{1}{h}\left(\left|\mathbf{L}_i^{th} \cap \mathbf{L}_j^{th}\right| - \left|\mathbf{L}_i^{(t-1)h} \cap \mathbf{L}_j^{(t-1)h}\right|\right) \tag{4.11}$$

where $\mathbf{L}_i^k$ denotes the elements of $\mathbf{L}_i$ up to the kth. Intuitively speaking, $k_\cap^t$ is the number of models preferred by both points i and j in the first th models for each, less those preferred in common in the first $(t-1)h$ models.

The ORK is defined as a weighted sum of DOIKs over all values of t. Specifically, the ORK places greater weight on the most preferred models: if points belong to the same ground-truth model, then the points they best fit should be similar, but we cannot say the same about the least-preferred models. The ORK is defined as

$$k_r(i,j) = \frac{1}{Z}\sum_{t=1}^{M/h} z_t \cdot k_\cap^t(\mathbf{L}_i, \mathbf{L}_j) \tag{4.12}$$

where $Z = \sum_{t=1}^{M/h} z_t$ and z_t provide the weightings to each DOIK evaluation. In [9], $z_t = 1/t$, but in the accompanying code, a more complicated weighting approach is included as well.

Direct computation of the ORK from Equation 4.12 is computationally intensive for even moderately sized data sets; however, a clever approach enables its computation in linear time with respect to the number of models (and thus residuals) and provides the additional benefit of offering insight into the workings of the ORK. Specifically, we can view the ORK as evaluating the DOIK in rounds, where each round increases the number of models a point "accepts". The DOIK $k_\cap^t$ examines a window of models ($(t-1)h+1$ through th) to see how many new models are accepted by both data points in comparison to the past rounds when we add the window to "accepted" models. In other words, $k_\cap^t$ is the number of models that are newly accepted in round t. Note crucially that this is not a fixed number, as can be seen in the following contrived example: suppose $\mathbf{L}_1 = [1, 2, 3, 4]$

[5]Note that M and h are user-controlled and so ensuring this condition is trivial

```
ORK(L_i, L_j, W, M, h)
 1  Allocate tables T_i[1..M], T_j[1..M] and set all entries in both to −1
 2  Allocate accumulator A[1..(M/h)] and set all entries to 0
 3  for k = 1, ..., M
 4       t = k/h
 5       // Ignore low-weight DOIKs;
 6       // models k accepted after this will have T_i[k] = −1
 7       if W[t] < 0.01 break
 8       T_i[L_i[k]] = T_j[L_j[k]] = t
 9  for k = 1, ..., M
10       if T_i[k] == −1 or T_j[k] == −1 continue
11       // t is the round in which model k is accepted
12       t = max(T_i[k], T_j[k])
13       A[t] = A[t] + 1
14  // Multiply by z_t = 1/t and 1/h
15  for t = 0, ..., M/h − 1
16       A[t] = A[t]/((t + 1)h)
17  return Σ_{t=1}^{M/h} W[t] · A[t]
```

Figure 4.15: The code to calculate the Ordered Residual Kernel

and $\mathbf{L}_2 = [3, 4, 1, 2]$ with step size $h = 2$. Then $k^1_\cap = 0$ since $\{1, 2\} \cap \{3, 4\} = 0$; however, $k^2_\cap = \frac{1}{2}(4 - 0) = 2$.

We can then examine each model, and determine in which window it is accepted to compute the DOIK. Specifically, we create tables $\mathbf{T}_1[1..M]$ and $\mathbf{T}_2[1..M]$; $\mathbf{T}_1[k]$ holds the round t in which model k is accepted for $\mathbf{L}_1$. We then compute the ORK using the procedure depicted in Fig. 4.15, where $\mathbf{W}$ contains additional normalized weights for the DOIK.

Thus, intuitively, the ORK examines how many models the points can "agree upon", giving more weight to the ones that the points prefer the most. If two points agree upon a model they both prefer a lot (i.e., have low residual with respect to), then Kernel Fitting puts a lot of weight on the intersection, which contributes highly to the sum.

Note that although this might seem simpler to compute than the sum of ker-

nels definition, the sum of kernels definition permits rapid verification of Mercer's condition. Specifically, it can be shown that if k_1, k_2 are kernels satisfying Mercer's condition and c is a positive constant, both ck_1 and $k_1 + k_2$ are Mercer kernels. The DOIK may be written as a positive fraction of another kernel, the intersection kernel, making it a valid kernel; the ORK may also be written as the sum of DOIKs weighted by positive constants (as can the approximation presented), thus also making it satisfy Mercer's condition.

We then compute K for a set of data points as follows. First, we sample M minimum sample models, and compute the order, $\mathbf{L}_i$ in which each point d_i prefers the models. Then we set $K_{i,j} = k_r(\mathbf{L}_i, \mathbf{L}_j)$ for all i and j between 1 and N.

In practice, the ORK is also combined with a Gaussian Kernel to incorporate the observation that nearby points frequently share model assignments. Thus, rather than use compute the ORK by itself, we compute

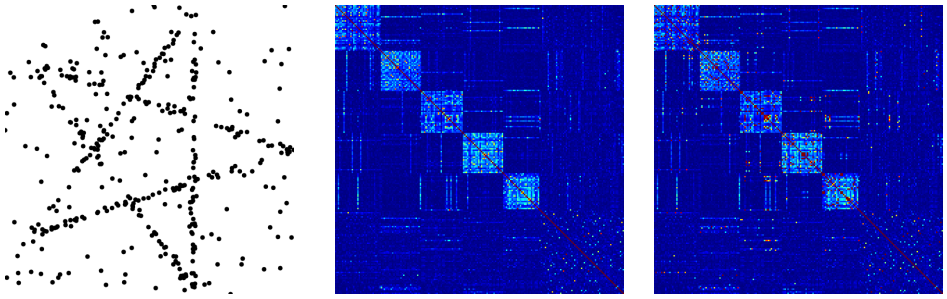
$$k'_r(\mathbf{x}_i, \mathbf{x}_j) = k_r(\mathbf{L}_i, \mathbf{L}_j) + k_G(\mathbf{x}_i, \mathbf{x}_j) \tag{4.13}$$

where k_G is the Gaussian kernel, which resembles the probability distribution with the same name, apart from some constants:

$$k_G(\mathbf{x}_i, \mathbf{x}_j) = \exp(-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma^2). \tag{4.14}$$

Note that $k_G(\cdot, \cdot)$ evaluates to a higher value for closer points and that the rate at which increasing distance causes $k_G(\cdot, \cdots)$'s value to decrease is a function of the scale parameter σ. Chin et al. set the scale of the kernel to the average Euclidean distance between a vector and its nearest-neighbor.

As a way of illustrating the output of the Ordered Residual Kernel with and without the addition of the Gaussian kernel, both have been computed and thenn visualized in Fig. 4.16 as was done for the Jaccard distance matrices. We first ordered the data points so that each data point matching a ground-truth model



(a) The data (b) The ORK visualized (c) The ORK+Gaussian Kernel visualized

Figure 4.16: A visualization of the kernel matrix used in Kernel Fitting

appears consecutively, with the outliers appearing last. The color runs from 0 (darkest blue) to 1/3 of the maximum entry in either matrix (darkest red), where higher values saturate (i.e., are bounded to fit within the range) and lower (i.e., negative) values are impossible. The pixel at row i and column j represents the kernel evaluated for points i and j. It should be readily observable that k_r evaluates to high values for points belonging to the same ground-truth model, given the squares on the main diagonal. Further, the symmetry of the image along the diagonal confirms the symmetric property $k_r(\mathbf{L}_i, \mathbf{L}_j) = k_r(\mathbf{L}_j, \mathbf{L}_i)$ mentioned earlier. As can be seen in the images, adding the Gaussian kernel produces a similar matrix to the one produced using just the ORK. In practice we observed that including the Gaussian kernel produces stronger results.

4.6.2 Kernel-based outlier removal

The Kernel Matrix can then be used to work with the data points in the feature space F indirectly through the identity $K = A^T A$. Since the feature space for a Gaussian Kernel has an infinite number of dimensions [3], this is an impressive approach: we can work with N data points in a space with infinite dimensions with a $N \times N$ matrix. Specifically, similar to PCA from Section 2.2, the first n

eigenvectors of K sorted by their eigenvalues provide a basis for the n-dimensional principal subspace of A. Further, Chin et al. note that since k_r evaluates to high values for points belonging to the same ground-truth model, the dominant directions of the principal subspaces of A are determined by the inliers; accordingly, when the data in the feature space is projected onto the principal subspace, inliers have high norms and outliers have low norms. This may be observed in Fig. 4.17(a), in which the norms of the points have been plotted in order: there is a significant drop off when the data reaches the outliers. Thus, by thresholding the data points with respect to the norms of their projections, the outliers may be removed from the data set.

Chin et al. propose two methods for determining a threshold for discriminating outliers from inliers. The first is to use the empirically justified threshold of 3/10 of the square of the maximum norm. The second, and somewhat more principled and interesting approach is to fit a Gaussian Mixture Model (GMM) to the norm data. A discussion of how to do this with Expectation-Maximization (EM) may be found in [3][6]. In general, mixture models may be tricky: various information criteria (e.g., Akaike or Bayesian) using the log-likelihood of the data and number of parameters must be used to determine how many mixtures are present, and initializing the parameters is tricky. However, we expect only two mixtures in our data, and Chin et al.'s code provides a clever way to initialize parameters: one mixture's mean starts at 0 and the other's at the maximum norm; both are given reasonable variance and responsibilities (or priors). As the EM algorithm proceeds, these converge towards sensible means, variances, and responsibilities. Once the

[6]See Section 9.2, in particular pages 438-439, for a recipe for the general case. Do not panic at the apparent complexity; the 1-dimensional case may be derived from the general case and is simple enough to be easily implemented without a linear algebra package. Further note that fitting a GMM with EM is frequently and incorrectly conflated with EM itself; [3], Sections 9.3 - 9.4 contain a more general explanation, as does [1], Section 11.2 (although the latter is a much more challenging read).

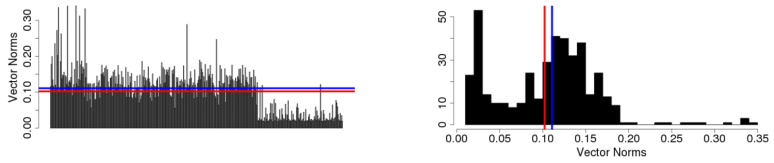
GMM has been fitted, [9] suggests a number of ways to find the threshold. For instance, so long as there are two valid clusters (if there is one, then this indicates that there might not be many outliers), the average of the two means might be taken. We draw the result of both thresholdings in Fig. 4.17(b), in which we have imitated the style of a figure from [9]: as can be seen, both produce similar results.

Once a threshold has been determined, one may remove many outliers by removing all data points with norms below the threshold. The kernel matrix K is then reduced to an inlier-only matrix K' by removing the rows and columns corresponding to outlier data points, and the outliers are not used later in the algorithm. Examining Fig. 4.17(d), which depicts the output of max-norm thresholding, we find that the threshold has removed all of the outliers, while retaining almost of all of the inliers.

4.6.3 Model detection and merging

Once the outliers have been removed, Kernel Fitting works with the inlier-only kernel matrix K' to detect the structures in the inliers. Rather than attempting to estimate the models in one step, it takes the approach of deliberately oversegmenting the data, and then merging it.

To oversegment the data, Kernel Fitting first constructs an effective representation of the data, and then clusters it with k-means. The representation it uses comes from a variant on PCA, Kernel PCA, which is discussed in [3]. Specifically, K' is transformed into its centered equivalent: if we denote A with the outlier points removed as C, we wish to consider C centered in the feature space F, which we denote as $\tilde{C}$. Just as $K = A^T A$ and $K' = C^T C$, Kernel Fitting constructs $\tilde{K} = \tilde{C}^T \tilde{C}$, which enables the computation of the principal components of the inlier data in the feature space, as it centers the feature space representation. The



(a) Norms for data points in order

(b) Histogram of norms

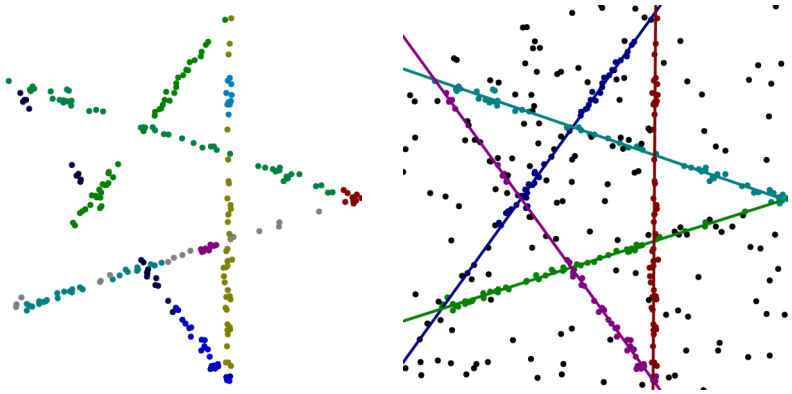
(c) The data points before thresholding

(d) The data points following thresholding

Figure 4.17: A visualization of the data points' norms when projected onto a principal subspace; note the tendency of outliers to have low norms. Lines indicating the threshold, as determined by maximum norm and GMM, are drawn in red and blue respectively.

projection of the inliers onto these principal components produces a space that is salient for the clustering of inliers into models with k-means. Two approaches may be used to produce this space: [9] advocates the use of Normalized Cuts on the data projected onto the principal components to determine the number of clusters and produce an effective space to perform the clustering; the accompanying code [6], however, uses Normalized Cuts to reveal the number of clusters, but performs k-means on the data projected onto the principal components unmodified. We present the result of oversegmentation on the unmodified projections in Fig. 4.18(a); different clusters have different colors.

Finally, the oversegmented inliers are clustered back into the final models.



(a) Oversegmentation of the data less points labeled as outliers. Model assignments are shown in color.

(b) Final model assignments and lines fitted to the consensus sets

Figure 4.18: The result of model selection

Again, a number of approaches are possible due to discrepancies between the code and publication. In [9] a scheme is proposed in which clusters are merged so long as their merger permits the satisfactory explanation of the data by the resulting structures. A suitable inlier threshold is automatically estimated from the residuals of the inliers and of the outliers, and models are fitted onto models with LMedS [26]; since the data points are virtually outlier free, and since the clusters contain only one model, LMedS is appropriate here. The accompanying code uses an information-criterion-based model selection scheme, along with LMedS. In the code, the Geometrically Robust Information Criterion [36] is used to select the number of models that minimize the criterion. Finally, a later work [7] by Chin et al. develops an alternative approach to model selection that uses support vector machines. In practice, we found that the information-criterion approach systematically failed on some data sets, merging all points together, but performed better than the iterative approach on other data sets.

We now summarize and present some observations about Kernel Fitting. To reiterate, Kernel Fitting uses minimum sample models to represent each data point

by the order in which it prefers models. By using kernel methods, it constructs a more salient representation, A, of the data points in a feature space F, and manipulates these more salient representations with the kernel matrix $K = A^T A$. First, outliers are removed, then the points are oversegmented into clusters, and finally, the clusters are put together using a model selection scheme.

All of this is done without knowledge of any of the usual parameters, although at the cost of few internal parameters that require minimal to no tuning. Specifically, Kernel Fitting requires a number of models to generate, a step size h, and the inlier/outlier thresholding scheme. However, M is an inherently necessary parameter, and h depends entirely on M according to [9]. Further, the empirically justified 3/10 max-norm inlier/outlier thresholding scheme appears to be surprisingly robust to different data set composition, and thus needs no adjustment for different noise scales or outlier compositions. As we will see in the final section, Kernel Fitting does have some failure modes; often in association with the model selection scheme.

4.7 Conclusions

We now summarize the algorithms and their properties. Following [9], we summarize the parameters used by each algorithm in Table 4.2. The algorithms fall into three classes: ones that automatically estimate all parameters; ones that use ϵ, the inlier threshold, and W, the number of ground-truth models; and ones that use ϵ and T, a minimum acceptable consensus size. As was argued earlier, the W and T parameters offer alternate ways of achieving the same task (distinguishing sensible from non-sensible models), and so there are effectively two classes: automatic and manual methods. The degree to which the automatic methods is debatable, as they require the setting of internal parameters. Nonetheless, one can often find an

Table 4.2: Parameters used by each algorithm: ϵ is an inlier threshold; T is a minimum acceptable consensus set size; W is the number of ground-truth models

	Seq. (1)	Seq. (2)	Multi	J-L	Merg. J-L	RHA	Kernel Fitting
ϵ	✓	✓	✓	✓	✓		
T	✓			✓	✓		
W		✓	✓				

internal parameter that works on a very wide range of data sets (outliers, noise, and model classes).

We may similarly separate the algorithms into classes: algorithms that use consensus sets (Sequential RANSAC, MultiRANSAC); algorithms that form alternative representations of data points using minimum sample models (J-linkage variants, and Kernel Fitting); and RHA, which finds models using histograms of residuals. Each approach has its disadvantages: the consensus set is a poor choice for disambiguating multiple models; a data-point-centric approach often induces a N^2 term in the runtime (since every point may have to be compared with every other point); and RHA's formulation in terms of histograms introduces the problematic task of finding modes in noisy histograms.

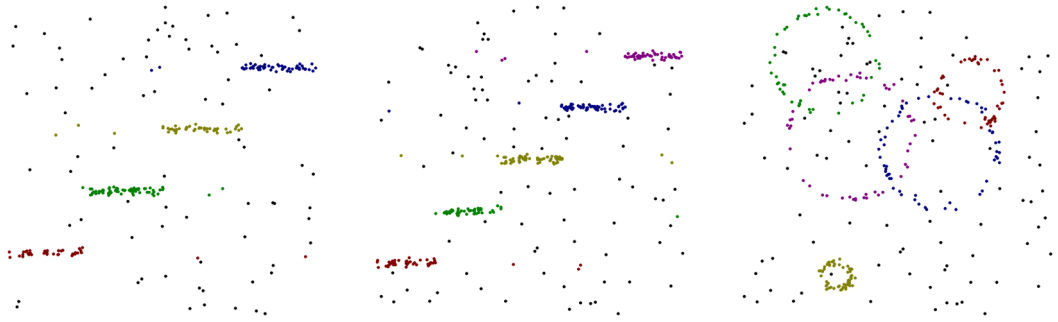
Given the diversity of approaches, it is difficult to pick an approach. One can narrow down the list of algorithms by removing algorithms that require more information than can be provided, but for any combination of parameters that may be provided, multiple algorithms may be used. The only criteria left are then performance (i.e., how effective is the approach?) and efficiency (i.e., how resource-intensive is the approach?). Efficiency may be evaluated by either examining the asymptotic efficiency of the algorithms or by running them for a series of data sets of increasing size. Performance, on the other hand is a more challenging problem. In the next chapter, we will discuss the challenges of evaluating multi-model estimation algorithms and how they may be surmounted.

CHAPTER 5

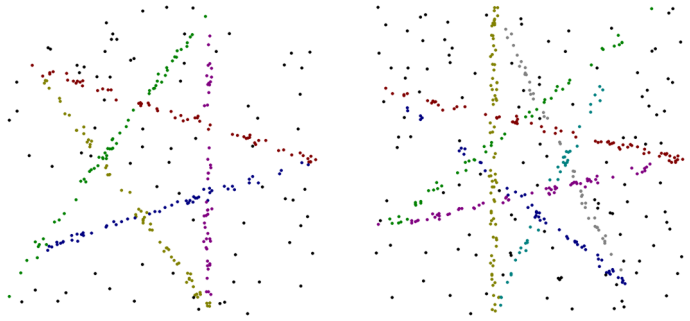
OUTLIER-ROBUST MULTI-MODEL EVALUATION

The final part of this thesis discusses the evaluation of the performance of multi-model estimation algorithms. Despite an increase of interest in recent years, there has been no comprehensive study of how to evaluate the performance of these algorithms. There have been efforts in papers proposing new approaches; however, these are limited in scope, and have been primarily focused on demonstrating that the proposed algorithm performs competitively in comparison to the state-of-the-art. There have been efforts for related problems: Choi et al. developed an approach for single-model robust estimation in [10], and Tron and Vidal presented a survey and evaluation of motion segmentation algorithms in [37]. Despite the apparent similarity, we will argue in this chapter that evaluating the multiple-model problem is a fundamentally different task than the single-model problem, and thus that not only is the content of [10] inapplicable to the multi-model problem, but also that the methodology itself is similarly inapplicable. To distinguish the task from the one solved in [37], we note that motion segmentation is only one possible task for the application of multi-model estimation algorithms, and that neither outliers nor noise were varied in [37] to assess performance.

We present a novel quantitative evaluation of multi-model estimation algorithms, and its application to the six algorithms presented in Chapter 4. We first introduce the data sets used, including both synthetic ones for 2D figure-fitting and real-world ones for plane detection that use a new technique to quantify the localization noise induced by subsampling. We then discuss the challenges of evaluating multi-model algorithms, as well as our methodology and scoring techniques for overcoming these challenges. Finally, we discuss our results, including many insights into the effectiveness of contemporary multi-model estimation algorithms.



(a) Stairs4 (b) Stairs5 (c) Circles5



(d) Star5 (e) Star7

Figure 5.1: Synthetic data sets: 30% outliers, 0.65% noise

5.1 Data Sets

We evaluated algorithms on the detection of three classes of models: lines, circles, and planar homographies. The data sets used for each task are entirely synthetic in the case of lines and circles (*Stairs4*, *Stairs5*, *Star5*, *Star7*, *Circles5*), and real-world in origin in the case of plane-fitting (*Planes2*, *Planes3*).

We have presented examples of a number of the data sets earlier. However, for the sake of completeness, we depict examples of each synthetic data set in Fig. 5.1. We also present, for each plane-fitting data set, the source images and a depiction of the planar surfaces present in the data in Fig. 5.2.

We will discuss each model-estimation task in turn. To fully describe a task, we will need to describe the data sets used and three functions or procedures for the manipulation of data points (from $\mathcal{D}$) and models (from $\mathcal{M}$). As was the case

in our formulation of RANSAC, these three functions are: $E : P(\mathcal{D}) \rightarrow \mathcal{M}$, which maps a set of points to a model; $R : \mathcal{M} \times \mathcal{D} \rightarrow \mathbb{R}^+ \cup \{0\}$, which determines the error of a point with respect to a model; and $S : P(\mathcal{D}) \rightarrow \mathcal{D}^{MSS}$, which selects a random sample of data points with sufficient cardinality to estimate a data set. We note that E's role is slightly expanded from our earlier presentation in RANSAC: previously, we limited it to an exact estimate (requiring MSS data points); for our presentation below, we also require it to compute a best-fit estimate of the model since some methods (e.g., Merging J-linkage) require it. In general, this is trivial as the best-fit solution for the minimum sample set coincides with the exact-fit solution. In the case of synthetic data, we will briefly discuss the generation of data sets, and in the case of plane-detection data, we will introduce a novel method for the control of data set parameters in real-world data. We will further discuss the definition of ground-truth data. As we will discuss in Section 5.2, we use a classification-based approach, and so our ground-truth consists of labels for data points indicating which ground-truth models each point fits.

5.1.1 Geometric figure fitting

We use two classes of data sets for line-fitting. The first is stairs, in which W horizontal line segments of width $1/W$ are placed across the unit square like steps. The second is stars, in which a W point star is inscribed in the unit square. We use one data set for circle-fitting, in which circles with different radii are placed in the unit square; to challenge the algorithms, some of the circles intersect. For the estimation and error functions for geometric figure fitting, we use the functions defined in Chapter 2, specifically PCA and the Kasa least-squares circle fitting procedure, and the line-point and circle-point distance respectively. For sampling, we use Kanazawa sampling, as specified in Chapter 3.

(a) Planes2, Image 0 (b) Planes2, Image 1 (c) Planes2 Labeled Correspondences

(d) Planes3, Image 0 (e) Planes3, Image 1 (f) Planes3 Labeled Correspondences

Figure 5.2: Plane fitting data sets. Left: image 0; center: image 1; right: depiction of a processed data set with 20% outliers.

To generate synthetic data sets, we produce a probabilistic generative model with the given ideal models. Given W ideal models, we produce a mixture model where each ideal model has a $1/W$ chance of being selected for sampling. Once a model has been selected, points are sampled on the model uniformly. For line segments, one can do this by parametrizing the line segment $l = \{(x_0, y_0), (x_1, y_1)\}$ as

$$\begin{pmatrix} (x_1 - x_0)t + x_0, & (y_1 - y_0)t + y_0 \end{pmatrix}$$

where $0 \leq t \leq 1$. We can then sample t uniformly to sample on l. For circles, we sample over the unit circle, expand the unit circle's radius and then translate the circle by the center. Specifically, for a circle (c_x, c_y, r), we select t uniformly with $0 \leq t < 2\pi$. The point is then:

$$\begin{pmatrix} c_x + r\cos(t), & c_y + r\cos(t) \end{pmatrix}$$

Thus, we have a hierarchical model in which each model defines a probability

distribution, where we select each model with uniform probability.

We can then create a test set, or particular instance of the data set with a given noise level and outlier composition. We first sample I data points using our inlier distribution, where we set I to $50W$; the expected number of points sampled from each model is then 50. As a practical matter, we may sensibly sample the same I inliers at each invocation without undesirable side-effects by saving the random number generator state upon entry to the inlier sampling procedure, replacing it with a fixed state, and then restoring it upon exit. We add noise sampled from the appropriate isotropic Gaussian distribution to each inlier. Finally, we add outliers uniformly sampled from the appropriate region; in all synthetic cases, this is the unit square.

The range of noise scales and outlier fractions match or exceed other published evaluations (apart from two extreme values) [9, 32, 43, 44]. Past noise levels (i.e., standard deviations) have included 0.375% and 0.75% [32, 44], 0.55% to 2.5% [9], and 0.1% to 5% [43]. Past gross outlier fractions have ranged from 0% to 50% [32], 60% [44], $0\% - 77\%$ [9], and $20\% - 50\%$ [43]. Note the distinction between gross outlier fractions (i.e., the number of points generated from the outlier process) and the total outlier percentage (i.e., both gross and pseudo-outliers). Our evaluation uses seven noise standard levels ranging from 0.45% to 1.05% in steps of 0.1%, and nine outlier fractions ranging from 0% to 80% in steps of 10%. We thus have $5 \times 7 \times 9 = 315$ synthetic test sets.

We conclude by discussing the setting of ground-truth labels for synthetic data. We define the models that a point fits as its preference set with respect to the generative models' parameters with ϵ set to 3σ; equivalently, we may describe the consensus set of each inlier model as the consensus set of the model. As discussed before, this threshold corresponds to approximately 98.9% of the 2D isotropic

Gaussian distribution. This approach more accurately captures the data that was generated than simply tagging points as they are produced by the inlier models: outliers that fit a model by chance are correctly labeled, and points fitting multiple models (e.g., in the star or circle data sets) are similarly accurately labeled. Thus, when evaluating an algorithm's performance, the ground-truth data accurately captures the nature of the data, rather than the process by which it was created.

5.1.2 The generation of plane fitting data sets

The plane-fitting task is similar to the geometric figure-fitting task, except with respect to the data set's origin. Algorithms use the least-squares procedure given in Chapter 2 to compute homographies from correspondences, and use the magnitude of the residual, $||\mathbf{H}(\mathbf{p}) - \mathbf{p}'||$, to compute the error of a correspondence $(\mathbf{p}, \mathbf{p}')$ with respect to a homography $\mathbf{H}$. Finally, as in geometric figure-fitting, Kanazawa sampling is used.

Past evaluations have generally included the task of detecting planar homographies from collections of correspondences. Generally a feature detector is used on a pair of real-world images, and the features are matched to produce correspondences, as was done in Section 3.2. Following the detection of correspondences, evaluations have typically either synthetically created noise and outliers, or not attempted to quantify the noise and outlier processes. For instance, isotropic Gaussian noise models with $\sigma = 0.5$ pixel [43] or 1 pixel [44] have been used, or the original localization noise has been retained [9, 13]. To generate outliers, generally a uniform outlier process [9, 43] has been used.

Although the use of real-world data seems to be an improvement in terms of realism in contrast to the synthetic tasks, previous approaches to noise and outliers undermine this realism. By using synthetic noise and outliers, one gains the ability

to control the noise, but the resulting data is arguably no more reflective of real-world conditions than the synthetic data sets; if this is true, then one is perhaps better off using completely synthetic data sets, as was done in [19], enabling total control over all the data parameters. On the other hand, by retaining the original noise or outlier processes, one has no control over the data set parameters, making it impossible to examine the impact of noise or outliers on the performance of algorithms. Control over data parameters is unnecessary to demonstrate that an algorithm performs satisfactorily; however, for an evaluation to analyze the performance of an algorithm over a variety of conditions, such control is crucial. Ideally, one would like to be able to control the noise and outlier processes (to enable a range of test sets) but retain the real-world nature of the data.

We introduce an approach to accomplish these seemingly mutually exclusive goals. For noise, we fall into neither extreme of synthesizing noise or ignoring it. Instead, we use a common pre-processing step for image data, resizing or subsampling, to induce localization noise, and then a novel technique to quantify the noise produced. Thus, while we cannot generate noise matching a particular scale, we can induce a range of noise that we can analyze and describe. For outliers, we use manual labellings to detect feature mismatches.

We summarize our method before explaining each step. Beginning with a source image pair, we resize the images to a series of decreasing dimensions or scales. In our tests, we start with images at 3500×2625; our resized scales start at 1750×1312, and the width is decreased by a factor of 0.95 until a dimension is below 400. We resize the source image pairs to this series of scales, and run SIFT [23] on each scale, including the source scale. We compute feature matches for the resize scales, and then run an interscale resolution procedure between the SIFT keypoints of the resized images and the keypoints from the source images.

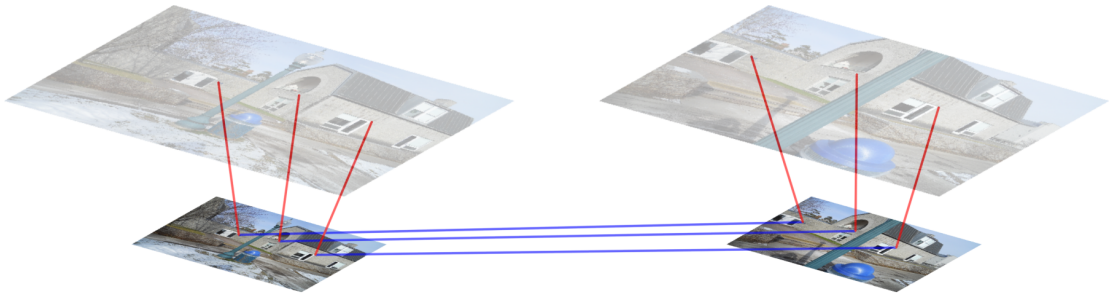


Figure 5.3: A graphical depiction of the interscale resolution procedure. Blue lines depict matches between resized images. Red lines depict matches between resize-scale images and source-scale images.

The source image's keypoints approximate the actual location of the keypoints: the keypoints in the source images are the keypoints from the resized images, but detected at a different scale. This provides a way to analyze the localization error for a particular resize scale. We then use manually determined outlines of the planes and RANSAC to find inliers and outliers. Each resize scale provides a particular noise scale; we then sample outliers from all noise scales and inliers from a particular noise scale to synthesize specific noise and outlier combinations.

We now detail the interscale resolution procedure by which we find the source-scale instances of a resized image's keypoints. Here, unlike in the standard SIFT matching procedure, we may use the keypoints' locations to aid our search for a match. Specifically, since the two images are the same image at different resolutions, we may limit our search to the nearest keypoints. For practical purposes, we convert the keypoint locations to a standard representation or coordinate system in which the image is represented by a rectangle with corners at $(0, 0)$ and $(1, \min\{h, w\}/\max\{h, w\})$ (i.e., we inscribe the image in the unit square).

We first define some notation. Let R and R' be the keypoints in resized images 0 and 1 and let S and S' be the keypoints in source images 0 and 1. Further, recall that $\mathrm{DD}(\cdot, \cdot)$ is the distance between two keypoints in the descriptor space and

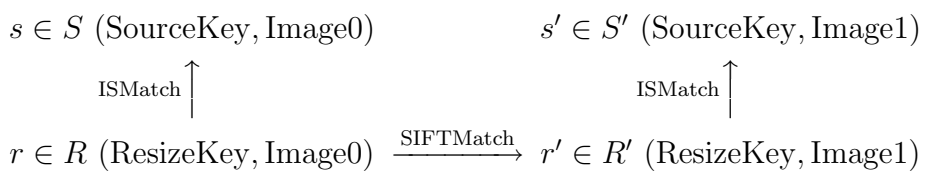


Figure 5.4: Keypoint diagram indicating the relationship between the source and resize keypoints.

similarly, let $\mathrm{ED}(\cdot, \cdot)$ be the distance between two keypoints in the image space. A depiction of our goal may be found in Fig. 5.3, and a summary of our notation may be found in Fig. 5.4.

We start with a set of correspondences $C \in R \times R'$ between the resized image 0 and 1. For each correspondence (r, r'), we find an initial inter-scale correspondence for r and r'. We set the search window of r to the 50 nearest elements of S to r. We accept an initial match between r and the element of its search window that minimizes the descriptor distance, so long as it is smaller than the average descriptor distance between the correspondences between the resized images (i.e., $\{\mathrm{DD}(r, r') : (r, r') \in C\}$). We then have an interscale mapping between R and S that is not necessarily one-to-one. We then enforce a one-to-one mapping by examining each s in S, and then each $r_1, \ldots, r_n$ that maps to s, and only finalizing the match between r_i and s that minimizes the descriptor distance. As an algorithm, this is depicted in Fig. 5.5.

We repeat the single-image resolution procedure for each image independently, and only retain correspondences where both keypoints are resolved at the source scale, producing a collection of resolved correspondences $C' \in R \times R' \times S \times S'$, where each correspondence is a 4-tuple of keypoints, a keypoint r in resized image 0, its match r' in image 1, and their source-scale matches s and s' respectively. We then verify the source-matches in the descriptor space by rejecting the resolved correspondence if s' is not the closest keypoint to s in the search window of r' or if s

```
ResolveSingleImage(R,S,DDBound)
 1  SToRMatch = New Dictionary()
 2  for r ∈ R
 3        SWindow = nClosest(50, S, r)
 4        PotentialMatch = arg min_{s∈SWindow} DD(r, s)
 5        if DD(r, PotentialMatch) > DDBound continue
 6        if PotentialMatch ∈ SToRMatch
 7              SToRMatch[PotentialMatch] = SToRMatch[PotentialMatch] ∪ {r}
 8        else
 9              SToRMatch[PotentialMatch] = {r}
10  // Filter out the matches
11  for s ∈ SToRMatch
12        Candidates = SToRMatch[s]
13        FinalMatch = arg min_{r∈Candidates} DD(r, s)
14        Link(FinalMatch, s)
15  return SToRMatch
```

Figure 5.5: The single image interscale resolution procedure

is not the closest keypoint to s' in the search window of r. Finally, we eliminate any remaining incorrect matches by removing the correspondences containing points in the upper 1% of points with regards to distance between r and s and r' and s'. This tends to remove a handful of wildly incorrect matches. We present pseudo-code for this in Fig. 5.6.

This produces, for each resized image, a collection of correspondences with accurate matches between the resized images and the source images. The localization error may then be estimated by the difference between s and r. We present a plot of the localization residuals (i.e., $r - s$ and $r' - s'$) in Fig. 5.7. The distribution is fairly compact, and off-center. The implicit zero-mean assumption in past noise models is then incorrect. This is, however, irrelevant: any translation in the noise distribution will merely be incorporated into the homography. Thus, we are principally interested in higher moments of the distribution. It turns out if we examine each dimension independently (or more formally fit a Gaussian with a diagonal

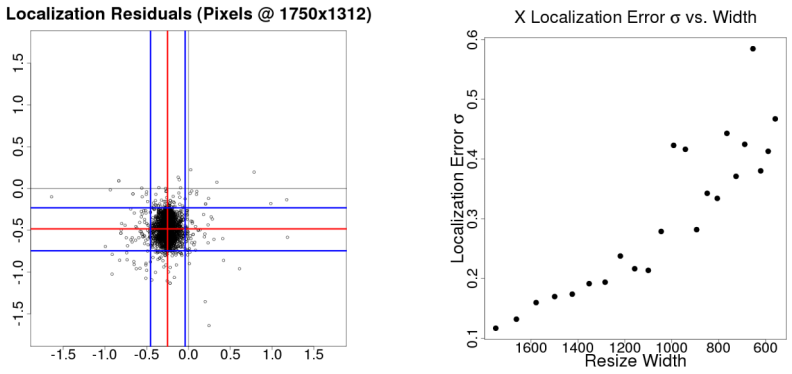
ISMatch($C \in R \times R'$,S,S')

// Compute descriptor-distance bound
$DDBound = \text{Mean}(\{\text{DD}(r, r') : (r, r') \in C\})$
ResolveSingleImage($R, S, DDBound$); ResolveSingleImage($R', S', DDBound$)
$L = \emptyset$
for $(r, r') \in C$
    if NotLinked(r) **or** NotLinked(r')
        continue
    $SWindowR = nClosest(50, S, r)$; $SWindowR' = nClosest(50, S', r')$
    if $s \neq \underset{s \in SWindowR'}{\arg\min}\ \text{DD}(s, s')$ **or** $s' \neq \underset{s' \in SWindowR}{\arg\min}\ \text{DD}(s, s')$
        continue
    $L = L \cup \{(r, r', s, s')\}$
// Get location errors for all keypoints under consideration
$LocError = \{\text{ED}(r, s) : (r, r', s, s') \in C'\} \cup \{\text{ED}(r', s') : (r, r', s, s') \in C'\}$
$EDBound = \text{Percentile}(99, LocError)$
return $\{(r, r', s, s') \in L : \max(\text{ED}(r, s), \text{ED}(r', s')) < EDBound\}$

Figure 5.6: The full interscale matching procedure

covariance matrix), then only 2% of the data in each dimension falls beyond 2σ, which is half of what one would expect with a normal distribution. To confirm this analysis, the residuals were tested with the Shapiro-Wilk multivariate normality test [27], and failed with overwhelming probability. Although this indicates that the empirical distribution is not normal, the analysis of the tails of the distribution suggests that the Gaussian has heavier tails than the empirical distribution, and is thus more challenging if it is at all different.

We conclude our description of plane-fitting data set generation with a discussion of how we produce data sets using the interscale matching procedure, beginning with our detection of the homographies for each ground-truth model. We run SIFT on source images 0 and 1, producing keypoint collections S and S', and then on a series of rescalings of the source images as described earlier, producing keypoint collections $R_1, \ldots, R_n$ and $R'_1, \ldots, R'_n$. At all of the resize scales, we generate a collection L_i of localized SIFT correspondences with



(a) Localization Residuals

(b) Localization σ against resize width

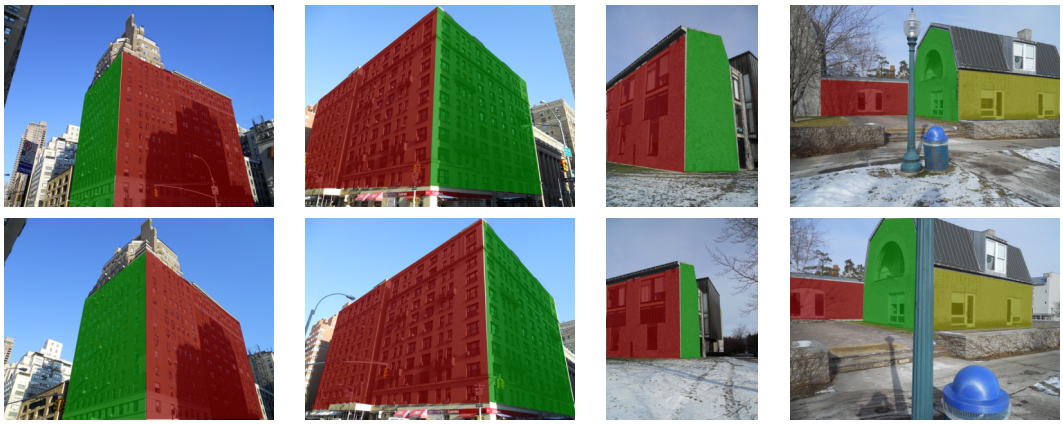
Figure 5.7: An analysis of localization noise: (a) the resulting distribution of localization errors for one image rescale size; the mean is depicted in red, and 95% confidence intervals in each dimension are depicted in blue; (b) the standard deviation of the errors for a range of image sizes

ISMatch(SIFTMatch(R_i, R'_i), S, S'). We then determine the model of the dominant surfaces of the scene, using manually determined polygons that outline the planes in both images. We make use of all the information gathered at every resize scale, and form the union L of every L_i. For each pair of polygons defining a planar surface, we select the subset of L whose correspondences are consistent with the polygons in each respective image. We detect the consensus set of the dominant planar surface within the spatially-conforming points with RANSAC, and then refit the homography with all of the source-scale locations of the consensus set. These fitted homographies act as the equivalent of the ground-truth parameters for the generative models in the synthetic data set.

We finish our discussion by explaining how these homographies and our collection of localized homographies L are used to generate data sets for a range of noise levels and outlier compositions. We first restrict our attention by retaining only the resize scales whose noise levels we wish to use. Then, at each resize scale i, using the homographies $H_1, \ldots, H_W$ representing the ground-truth planar surfaces, and the collection L_i of localized SIFT correspondences, we find the consensus sets

$P_{i,j}$ of each H_j in each L_i, and place the correspondences matching no model into a global pool of outliers O; we then have a global outlier pool with mismatches from every scale and, for each scale, a collection of inliers $P_{i,1}, \ldots, P_{i,W}$ for each planar surface. We fix a single set of consensus set sizes from one of the scales to minimize the number of independent variables, and then for each noise level i and outlier level, sample the same number of inliers from the appropriate inlier collection $P_{i,j}$ for each model j and the appropriate number of outliers from the outlier pool. Since our homographies are our inlier-models, the resulting collections of outliers and inliers accurately reflect the data as will be perceived by a model; therefore, the resulting collections of labeled inliers and outliers constitute our ground-truth labellings. We need the global outlier pool since we want to be able to unnaturally boost the outlier composition; sampling from multiple scales is not problematic as we are unconcerned with the noise of the outliers.

Many image pairs were processed with this approach, and two were used to produce *Planes2* and *Planes3*. Four of them, including the two used in the evaluation, are presented in Fig. 5.8. Two were taken in Manhattan, and the other two on Middlebury College's campus. We used Pair 9 and 16 to produce Planes2 and Planes3 respectively since they had the strongest inlier sets and had enough viewpoint disparity to reliably distinguish the planes. Planes2 contains 229 and 100 data points from each of the two dominant planar surfaces of the scene, and has a natural outlier composition of $40 - 65\%$ (depending on the size of the image). Planes3 contains 68, 50, and 142 data points from each dominant planar surface, and has a natural outlier composition of $30 - 65\%$. However, in addition to the building facades, it contains a number of smaller planar surfaces. We do not want to penalize algorithms for detecting models that are present in the data, and so we label these smaller surfaces, ensuring their inliers are withheld from the



Pair 7 (Not Used) | Pair 9 (*Planes2*) | Pair 13 (Not Used) | Pair 16 (*Planes3*)

Figure 5.8: Four image pairs, including the two data sets used in the evaluation

outlier pool, but do not emit them, giving a data set in which there are only three surfaces. We chose 4 noise settings for each image pair, ranging from 0.22 pixels to 0.42 pixels in one pair and 0.69 pixels to 0.80 pixels in another, and we varied gross outlier fractions from 0% to 80% in steps of 10% as before. This produces $2 \times 4 \times 9 = 72$ test sets for plane fitting.

5.2 Testing Methodology

Accurately quantifying the performance is a difficult task. Each algorithm in question is randomized, and so its output on a particular data set is a random variable. Accordingly, the performance of an algorithm is best characterized in the aggregate rather than in a qualitative fashion as in [32, 43]. This is, however, a difficult task: a performance metric must establish correspondences between estimated and ground-truth models, even when there are discrepancies between their count, and accurately assess not only roughly correct but also wildly degenerate configurations.

Past quantitative approaches may be broken into ones using a similarity metric

between estimated model parameters and ground-truth ones [9, 31] and those which treat the task as a classification problem [19, 34, 37, 44], in which the percentage of points "correctly" labeled is considered. Similarity metrics do not translate between model fitting tasks, do not have intuitive interpretations, and are quite tricky to accurately define. For instance, one might use the Euclidean distance to define a similarity metric between homographies (by concatenating all of the free parameters into a vector); however, a difference of 0.5 in an entry corresponding to the change in scale is likely to be more of a concern than the same difference in an entry corresponding to translation if the correspondence locations are in pixels and range from 0 to 1000. Accordingly, we use a classification-based approach. Although past work has included classification approaches, our metrics are fundamentally novel. In [34], only a binary inlier-outlier classification rate was used, ignoring the distinctions between multiple models; in [37, 19], the error metric is only described as the number of misclassified points; and in [44], the evaluated algorithms were given the number of models, and so the evaluation approach did not have to deal with mismatches in the number of models. In contrast to past work, we rigorously define our metrics and not only use them to analyze performance, but also analyze them themselves.

5.2.1 Scoring metrics

We describe our approach by giving a taxonomy of classification metrics. In any classification scenario, an algorithm's score is determined by

$$\frac{\#Points\ Correctly\ Detected}{\#Total\ Points} \tag{5.1}$$

and thus scores range from 0.0 (completely incorrect) to 1.0 (completely correct). To automatically score an algorithm's output, we must consider when a point is

considered correct (*a correctness criterion*) and which points are considered in the classification score (*the classification set*). For our particular case, multi-model estimation algorithms, we must also define how to determine a mapping, denoted ϕ, between estimated and ground-truth models (*a model mapping*). Before we begin, we introduce some notation. Let the ground-truth and estimated consensus sets be denoted $\mu_1, \mu_2, \ldots, \mu_W$ and $\mu'_1, \mu'_2, \ldots, \mu'_E$ respectively. Each μ_i or μ'_i is a subset of the data points, and $\mu_1 \cup \cdots \cup \mu_W$ and $DataPoints - (\mu_1 \cup \cdots \cup \mu_W)$ are the ground-truth inliers and outliers respectively.

Correctness Criteria: There are a number of approaches for determining whether a point is correct or not. The most simple approach is to simply consider whether a point was correctly determined as an inlier or outlier, as in [34]. This is perhaps limited: the point of a multi-model estimation algorithm is its ability to discern multiple models in data. Therefore, if one has a mapping ϕ from estimated to ground-truth models, then one can use a model-aware criterion. A point is counted as correct if and only if one of its estimated models maps to one of the ground-truth models or if it is correctly identified as a ground-truth outlier. One might take this a step further and use a more complex or restrictive notion of correctness, but for now this definition suffices for considering multiple points.

Classification Sets: There are two probable options for the classification set. The most immediately obvious one is to consider all data points. However, including all data points leads to decreasing emphasis on accurate inlier assessment as the percentage of outliers rises. For instance, with 80% outliers, an algorithm can get a guaranteed score of 0.8 by not estimating any models. In practice, this leads to an artifact in the resulting graphs: when an algorithm does poorly by not estimating many models, its performance appears to improve as a function of outliers. One potential solution is to only consider the inliers for classification. When

this is done, the opposite problem may occur and algorithms are not penalized for including outliers.

Model Mappings: We conclude our taxonomy with methods for determining a mapping from estimated to ground-truth models, beginning with Maximum-Intersection. Specifically, we map the estimated model μ'_j to the ground-truth model μ_i that it shares the most points with, or

$$\phi(\mu'_j) = \arg\max_{i \in 1,\ldots,W} |\mu'_j \cap \mu_i|. \tag{5.2}$$

While Maximum-Intersection does well when $W = E$, it can act as an optimal model-fusion procedure and grossly distort performance when $W > E$. For instance, if an algorithm can distinguish inliers from outliers but not individual models, it could give each inlier data point its own model, and Maximum-Intersection would assign each estimated model (and thus inlier data point) to the correct ground-truth model. If coupled with any correctness criterion and classification set, the algorithm would achieve a perfect score, although its performance was mainly artificially achieved during the scoring procedure (and thus not replicable in a real-world scenario). In practice, this means that a scoring metric based upon standard Maximum-Intersection model mapping will fail to penalize redundant and extraneous models.

To correct this, we introduce another mapping procedure, Strongest-Intersection, in which only the $\min(W, E)$ strongest inliers are considered. Weaker models do not count as outliers, but may not contribute towards inlier detection. Although it bears resemblance to the approaches used in [32, 9], it is distinct. In [32], W was effectively provided to each algorithm during its execution in an attempt to ensure that algorithms were evaluated on equal footing; this fundamentally alters the algorithms. In [9], the W models were selected not by size, but instead in such a fashion that the models minimized the error metric. Thus, the models best

Table 5.1: The composition of classification metrics

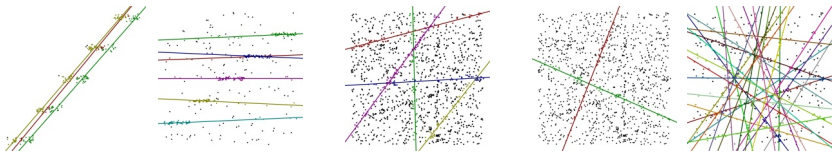
	Correctness		**Class. Set**		**Model Map**	
	Inl.-Outl.	Model-Aware	All	Inliers	Max-Isect.	Str. Isect.
Inlier/Outlier	✓		✓			
Many-to-1		✓	✓		✓	
N-str-to-1		✓	✓			✓
N-str-to-1		✓		✓		✓

corresponding to the data points are selected; this naturally requires a correction factor to penalize algorithms for extraneous models. However, the correction factor results in a metric that is not readily comprehensible.

Only a few of the possible metrics are useful. If we use a binary inlier-outlier classification (and thus do not need to resolve models) over all points, we get the *Inlier/Outlier* metric (which was used in [34]). The model-aware correctness criterion and a Maximum-Intersection mapping over all points produces the *Many-to-1* scoring metric. Using the model-aware correctness criterion, the Strongest-Intersection mapping, and all points and inliers respectively produces *N-strongest-to-1* and *N-strongest-to-1-inliers*. Finally, we include a non-classification score, Model Count, which assess how well the algorithm detected the number of models present in the data. In keeping with the bounds of the classification scores, it is defined as $\min(W, E)/\max(W, E)$ and is thus bounded to within 0.0 and 1.0.

We present some failure modes of the defined scoring approaches in Fig. 5.9. In general: Inlier-Outlier and Model Count are ineffective; Many-to-1 fails to penalize redundant detections; N-strongest-to-1, and all other all-data-point metrics have issues balancing the importance of outliers; and N-strongest-to-1-inliers has severe issues when large numbers of models are produced in addition to the correct ones. We advocate the N-strongest-to-1 metric: its failure mode (that models are not estimated) is easy to detect, and it produces significantly less inaccurate scores in comparison to the only other credible alternative, N-strongest-to-1-inliers. We

Sample results by different algorithms



Issues:	wrong model assignment	redundant models	wrong models but correct count	missing models	extra models
Inlier-outlier:	**0.98**[a]	**0.93**[a]	0.65	**0.60**[d]	0.65
Model count:	0.60	0.83	**1.00**[b]	0.40	0.17
Many-to-1:	0.27	**0.93**[c]	0.61	**0.57**[d]	0.63
N-strongest-to-1:	0.27	0.82	0.61	**0.52**[d]	0.61
N-str-to-1-inl.:	0.27	0.82	0.30	0.09	**0.95**[e]

Figure 5.9: Common failure modes of scoring functions on actual output: (a) Inlier-outlier does not evaluate actual model assignments; (b) accurately estimating the number of models does not imply accurately estimating them; (c) Many-to-1 does not penalize redundant or extraneous detections; (d) an outlier-inclusive classification set decreases the importance of accurately estimating inliers; (e) extra models that include only inliers are not penalized under N-strongest-to-1-inliers.

could perhaps merge multiple scores to eliminate this artifact; however, we believe that having a comprehensible score that occasionally needs explanation is better than having a black-box score that is impossible to explain, but which seems to produce the graphs we want.

5.2.2 Testing procedure

We conclude our description of the testing methodology by describing the testing procedure. Even with a proper scoring metric and data sets, properly running the tests is challenging. In particular, the needs for a-priori knowledge between the algorithms poses issues: not only does one need to consider how to handle MultiRANSAC's need for the number of ground-truth models, but one also needs to know how to set parameters such as ϵ and T, which do not have clear definitions.

We implemented each algorithm, except for Kernel Fitting, for which we used the provided code [6]. Our evaluation framework (approx. 5000 lines of code) was written in Python, although a few frequently-called mathematical primitives

Table 5.2: Fraction of the volume of the isotropic 2D Gaussian PDF that falls within given radii

Radius	σ	1.5σ	2σ	2.5σ	3σ	3.5σ	4σ
Fraction	39.3%	67.5%	86.5%	95.6%	98.9%	99.8%	99.97%

(e.g., Gaussian PDF evaluations) were written in C and linked in with SWIG. We additionally implemented Kernel Fitting ourselves, with the paper's model-selection scheme, using Numpy for the requisite linear algebra machinery. When we found that the code's model-selection scheme did not work, we tried the paper's: we discuss the performance of each scheme in the discussion of each data set.

We provide each algorithm all of the parameters it needs; although it may seem to provide algorithms requiring more information an advantage in the evaluation, the other alternatives are less appealing. In [32], the algorithms were all given W to ensure an equal comparison with MultiRANSAC. This crucially occurs inside the algorithms, rather than during an evaluation; thus, the scoring methodology (in this case, the visual system), has no chance to integrate the remaining models or their absence into the score. Further, this assumes that W is a different type of a-priori knowledge than ϵ or T. Taking the approach of modifying algorithms to use all of the parameters to the logical conclusion requires modifying all of the algorithms to use all of the potential data parameters; this would, of course, defeat the purpose of the study as the algorithms themselves would not be evaluated. In the other extreme, one could classify the algorithms into groups that require the same parameters; this is unsatisfactory as well: just because Kernel Fitting can automatically estimate ϵ does not mean that one of its main "competitors" is J-linkage and that it cannot be used in a scenario in which a suitable estimate of the noise scale could not be obtained. Thus, although it seems unfair, we argue that providing each algorithm all of the parameters it needs produces the most informative and accurate evaluation.

Although W is an easy parameter to determine, both ϵ and T do not have immediately apparent "ground-truth" values. As was the case in the generation of ground-truth labels for points, we set ϵ to 3σ for synthetic data. The volume of the Gaussian PDF falling within a series of radii is presented in Table 5.2. For plane-fitting data, we set ϵ to 2 pixels at a resolution of 1750×1312. One could use [33] to automatically determine the optimal ϵ value; however, when the noise is synthetic, it makes more sense to use that knowledge in a straight-forward fashion rather than introduce another algorithm, and in the case of plane-detection, we found 2 pixels to be empirically effective. Finally, we set T to 50, the expected size of the consensus sets of each model for synthetic data, and to the minimum consensus set size for plane-fitting. Algorithms such as J-linkage and Sequential RANSAC use a fixed fraction of this to discriminate between legitimate and spurious models.

Each algorithm was tuned, and then optimized for the evaluation task. We manually tuned parameter settings for each algorithm to a fixed number that produced the best results. We then added a final post-processing step to every algorithm, in which we refitted the algorithm's consensus sets: we took the outputted consensus sets, found their best-fit models, and reestimated the consensus sets using those models. In all cases, this improved the performance of the algorithms. For instance, Kernel Fitting was designed to recover the underlying models rather than their consensus sets, and so would be unfairly penalized without this step by a classification-oriented evaluation scheme.

We ran each of the 6 algorithms on each of the $315 + 72$ data sets 15 times for a total of $34,830$ test runs. The tests were spread over a cluster of workstations and took approximately 33 computer days. At each run, the algorithm's output and the CPU time taken were recorded.

Table 5.3: Average rank of each algorithm on each data set over all noise and outlier settings according to N-strongest-to-1 scoring. The method with minimum average rank in each row is highlighted.

	SeqRS	MulRS	JL	MJL	RHA	KF
Stairs4	3.84	1.71	**1.68**	4.38	4.49	3.00
Stairs5	5.14	4.21	**1.43**	3.94	3.56	2.24
Star5	2.56	4.17	2.51	3.90	5.44	**1.38**
Star7	2.14	4.33	3.00	3.60	5.52	**2.02**
Circles5	**3.06**	3.38	3.49	3.63	4.29	3.10
Planes2	3.92	3.11	4.03	**1.11**	4.94	2.97
Planes3	3.64	1.92	3.14	**1.47**	5.36	5.28

5.3 Results

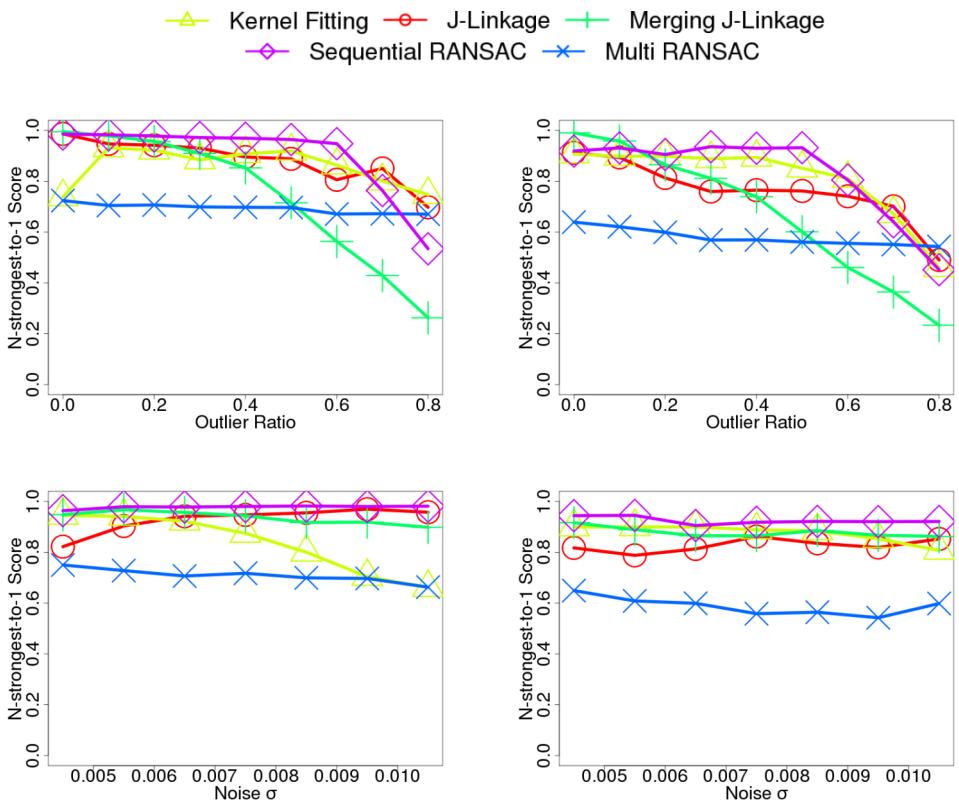
We now analyze the results of our evaluation study. In addition to manually examining all of the results to assess performance and prevent inaccurate automated evaluation, we plotted average N-strongest-to-1 scores against outlier composition and noise levels for each data set. When plotting against outliers, we limit the outlier composition to 20%; when plotting against noise, we limit the noise level to the second-smallest for geometric figure fitting and use all levels for plane fitting (since the former's noise range is significantly larger than the latter's). To summarize performance, we also present the average rank of each algorithm over all test sets for each data set in Table 5.3. Note that the numbers are ranks, and thus are not informative about an algorithm's performance taken in isolation; for this, one must consult the score graphs presented later in this section. We will discuss each data set in turn, and then summarize our findings for each algorithm at the end. To avoid clutter, we do not present RHA's results: although it was critical for the development of later algorithms and can often detect many models, as can be seen in Table 5.3, RHA is not competitive.

5.3.1 Lines

The lines data sets each pose a different challenge to algorithms. As mentioned earlier, the stairs sets are ambiguous, and a line across every ground-truth model may contain more data points than a line across a ground-truth model. This effect is more pronounced in the 5-model case than 4-model case: in the 4 model case, the ground-truth interpretation barely has more points than an incorrect one (36 vs. 35); in the 5 model case, a correct interpretation has significantly fewer points (40 vs. 49). In stars, although there is no ambiguity, models intersect cyclically, posing issues for algorithms that assume disjointness. We plot average N-strongest-to-1 score against outlier and noise in Fig. 5.10 and 5.11 .

J-linkage performs consistently well on all line-fitting examples, and overall outperforms the other algorithms on the stairs examples. This performance is not emulated by Merging J-linkage, which quickly hallucinates models from the outliers in all line-fitting examples (recall the mess of lines that was not correctly penalized by N-strongest-to-1-inliers in Fig. 5.9). The merging step enables outlier clusters to merge into seemingly valid models: the outliers can create best-fitting models, which will produce more spurious models than in Sequential RANSAC, in which a strongly-supported spurious model must be encountered by chance.

Kernel Fitting performs similarly well to J-linkage, although issues with selecting an appropriate model-selection scheme arise in the stairs example. The scheme in the accompanying code performs significantly worse that the paper's, and so we used our own implementation, which results in competitive performance on both stairs data sets. The code's merging scheme performs very well on both stars data sets, outperforming the other algorithms overall. We observed that when Kernel Fitting failed on Star 7, it tended to under estimate the number of models by 1 or 2. We suspect that this is related to the dimensionality (6) of the spaces used for

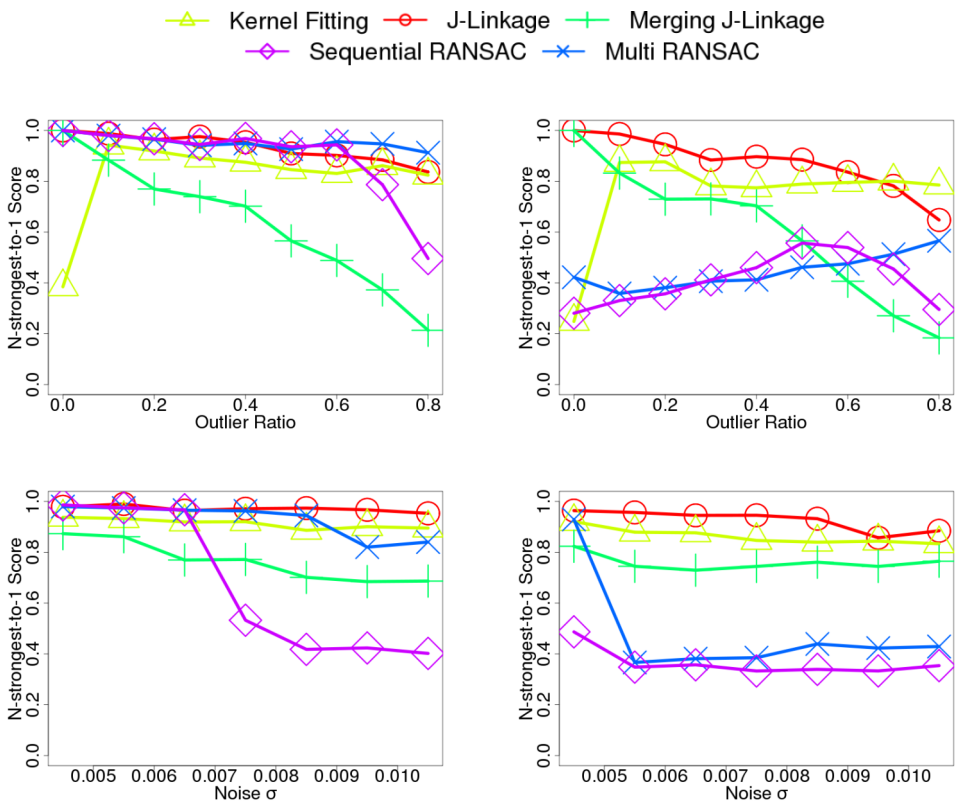


(a) Star5 (b) Star7

Figure 5.10: The performance of each algorithm on the star data set as a function of outliers (top row) and noise (bottom row)

clustering and model-selection, which might not be able to accurately represent data for 7 models. We further observed that Kernel Fitting tended to do poorly in the presence of no outliers.

The consensus-set-oriented approaches have significant trouble on the stairs data sets. This was observed for Sequential RANSAC in [44, 32]. However, our use of Stairs5 reveals issues with MultiRANSAC: MultiRANSAC's performance dramatically drops (from an average rank of 1.71 to 4.21) when the number of models is changed from 4 to 5 (and thus the consensus set size of a diagonal model increases). One problem is the disjointness assumption used in UpdateCS: when UpdateCS merges two collections of consensus sets, it takes the consensus sets in decreasing order. Therefore, if there is a diagonal model in either consensus set



(a) Stairs4 (b) Stairs5

Figure 5.11: The performance of each algorithm on the stairs data set as a function of outliers (top row) and noise (bottom row)

collection, it is selected first; when the correct models are considered for inclusion, they are rejected as they intersect the diagonal model. Thus, in addition to failing in the presence of intersecting models, MultiRANSAC's disjointness assumption ensures that it does poorly on stairs as well. Further, as the performance of the consensus-set-oriented algorithms depend on how many diagonal models are produced when sampling, increasing the noise (which induces more diagonal models) dramatically affects performance, which may be observed by the "breaking points" seen as a function of noise in Figs. 5.11a and 5.11b.

Sequential RANSAC does surprisingly well on Stars, and MultiRANSAC does poorly since the models intersect. MultiRANSAC's poor performance on the star data sets was noted in [32]. This ability to cope with intersecting models sug-

gests that although it is more simple, Sequential RANSAC's method of enforcing disjointness (removal of inliers) is perhaps more effective than MultiRANSAC's (rejection of intersecting models): when models intersect, Sequential RANSAC must overcome the challenge of detecting the second model without common points; on the other hand, MultiRANSAC must estimate both models accurately while not letting them intersect (an impossible challenge in many cases). Thus, MultiRANSAC tends to get the approximate idea of each model correct, while estimating the parameters incorrectly to avoid intersection.

5.3.2 Circles

Circles5 is by far the most challenging of the data sets. Contributing to the difficulties are our noise-levels: past evaluations have used 0.375% [32] and 0.1% [9], well below our levels (as high as 1.05%. However, the difficulty is also intrinsic to the task of circle-fitting: in short, in circle-fitting, finding a coherent model is significantly more difficult than in line fitting, and coherent models are not very distinctive compared to outlier models. The former is relatively easy to understand: finding a coherent circle model requires finding one extra data point, and is thus $P(C)$ less likely than finding a coherent line model, using the notation from our discussion of RANSAC. The latter is far more subtle, and is described numerically in Table 5.4. We took three exemplar data sets with 40% outliers from each model class. We sampled 2000 outlier-only and 2000 inlier-only models from each data set and recorded their consensus set sizes (which approximately indicates the strength of that model). We then examined what percentage of outlier (i.e., spurious) models have consensus set sizes larger than a series of consensus-set size percentiles of the inlier-models. For example, according to Table 5.4, if the 10th percentile of consensus set sizes for lines occurs at x (i.e. 10% of the inlier

Table 5.4: An indication of the overlap in consensus set sizes between models formed from outliers and models formed from inliers. We list what fraction of the outlier-only models have consensus sets sizes bigger than the given percentile of the inlier-only models.

	10th	25th	50th	75th	90th
Lines	5.35%	0.05%	0%	0%	0%
Circles	51.25%	22.8%	3.85%	3.05%	2.75%
Planes	72.65%	2.7%	0.25%	0%	0%

models have consensus set sizes smaller than x), then 5.35% of the outlier models have consensus sets with size larger than x. The difficulty of our circle-fitting data set is revealed by the significant overlap of outlier models and inlier models with respect to consensus set sizes: for the inlier models to be detectable, they have to be distinguishable from outlier models.

Sequential RANSAC does the best of all the algorithms on Circles5, with MultiRANSAC a close second, despite the circles' intersection. J-linkage will often fail to form sufficiently strong clusters and detect the underlying models, and Merging J-linkage has the opposite problem, and quickly hallucinates circles from the outliers. Kernel Fitting works best with the paper's merging scheme, although its performance is hit-or-miss and it often misgroups points; with the code's merging scheme, it frequently merges all points together.

5.3.3 Planes

Merging J-linkage, unsurprisingly, outperforms the other algorithms overall. As predicted by the motivation for its merging scheme, it does particularly well on Planes2. The large support regions of the planes in Planes2 challenge J-linkage, Sequential RANSAC, and MultiRANSAC: as was argued in Section 4.5, J-linkage's requirement of one commonly accepted model becomes problematic since Kanazawa sampling makes it difficult to find a minimum sample set that

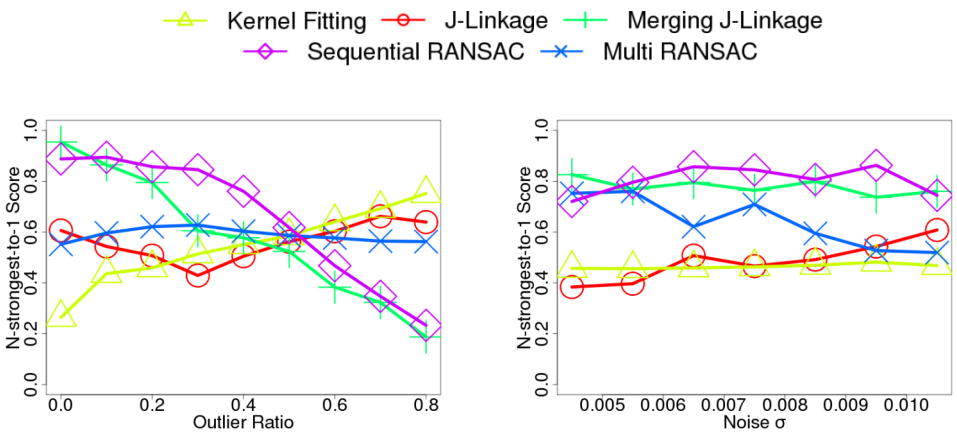
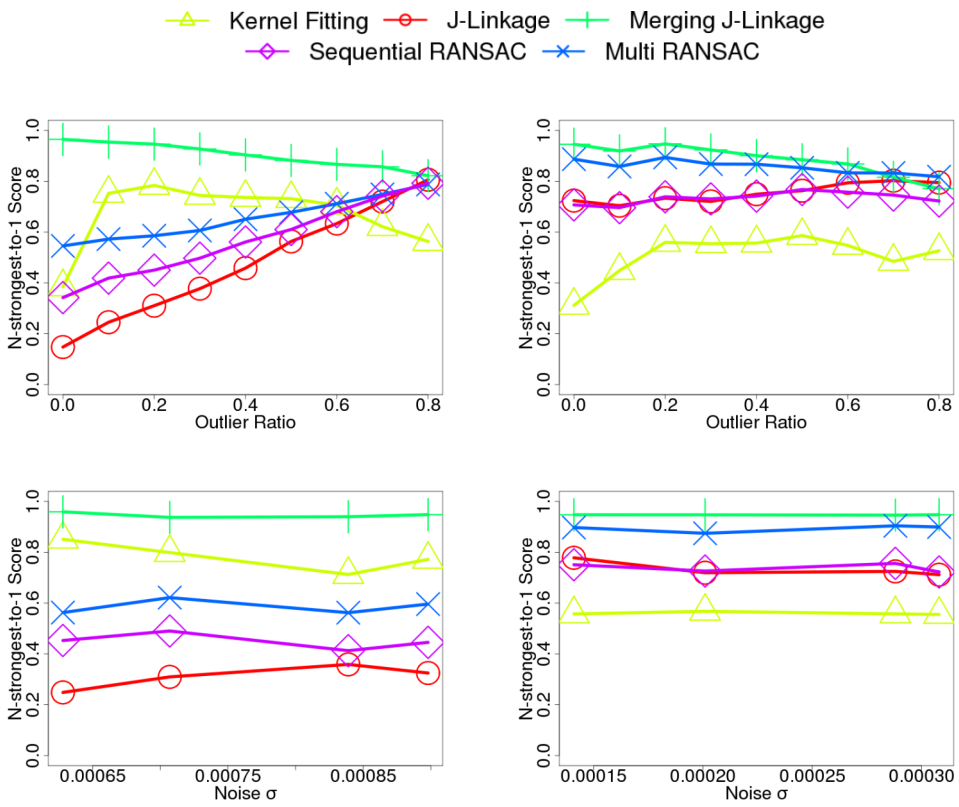


Figure 5.12: The performance of each algorithm on circle-fitting data as a function of noise (left) and outliers (right). Note that the seeming increase in performance as a function of outliers is an artifact of the algorithms failing to estimate many models, and the increase in outliers.

is both coherent and capable of estimating the global aspects of the perspective transformation. Sequential RANSAC and MultiRANSAC both fundamentally depend on consensus sets of minimum sample models, and so they similarly have issues finding an accurate model. Kernel-Fitting does very well on Planes2, apart from poor behavior on outlier-free data, as similarly observed in line-fitting.

Although Merging J-linkage retains its lead in Planes3, a large number of algorithms do significantly better with the smaller planar surfaces. MultiRANSAC achieves nearly the same performance as Merging J-linkage, and J-linkage and Sequential RANSAC perform much better as well. Mysteriously, Kernel Fitting does worse on Planes3: with the code's merging scheme (used for the graphs), it struggles with disambiguating each planar surface, and with the paper's merging scheme, it oversegments. The disparity in performance between Planes2 and Planes3 indicates that future evaluations should include a range of plane-fitting tasks, and that larger surfaces that cannot be modeled well by affine transformations may pose more of a challenge than having an additional model present in the data.



(a) Planes2 (b) Planes3

Figure 5.13: The performance of each algorithm on plane-fitting data as a function of outliers (top row) and noise (bottom row); note that the seeming increase in performance on Planes2 as outliers increase is the aforementioned artifact of increasing outlier levels.

5.3.4 Discussion

We now provide high-level observations about the algorithms evaluated. Average runtimes for a series of data set sizes (number of data points) are presented in Table 5.5. It would appear that RHA, Sequential RANSAC, and MultiRANSAC are empirically linear in the number of input points, and that J-linkage variants and Kernel Fitting are quadratic. This empirical assessment may be confirmed by considering the approaches. The RANSAC methods and RHA scan through the data points roughly once per minimum sample model, and both Kernel Fitting and J-linkage construct a N^2 matrix: J-linkage's is the Jaccard distance matrix

Table 5.5: Average runtime in seconds for each method for three dataset sizes, which roughly double between rows.

# of points	SeqRS	MulRS	JL	MJL	RHA	KF
411	16.2	88.2	19.9	22.8	10.5	22.8
822	22.1	122.8	87.7	98.4	15.0	213.9
1645	37.3	202.2	559.7	616.3	24.5	529.0

between clusters (which is $N \times N$ initially), and Kernel Fitting's is the kernel matrix K. Thus, the RANSAC approaches and RHA are favored with respect to time.

We now discuss the overall performance of each algorithm:

- Although **Sequential RANSAC** has been treated as a baseline method by most evaluation studies, it is one of the clear winners in this evaluation: it is capable of performing acceptably to strongly on most data sets (except for stairs), and is both simple to implement and extremely fast. As articulated earlier, its approach to disambiguating models in the consensus-set space is superior to MultiRANSAC's.

- **MultiRANSAC**'s disjoint consensus set assumption and parallelized approach consistently lead to problems. It does well on Stairs4 as well as Planes3. However, its poor performance on Stairs5 suggests that its performance on Stairs4 is not representative, and in the case of plane-detection, it is outperformed on both data sets by Merging J-linkage, which is equally easy to implement.

- **J-linkage** performs consistently across the data sets, although the setting of an appropriate threshold for cluster size is tricky: if it is too low, spurious models rapidly appear, and if it is too high, then J-linkage misses models. Although it is relatively simple to implement, it has quadratic time complexity with respect to the number of data points. This issue has recently been

addressed by the authors in [34], with another J-linkage variant that trades accuracy for computational efficiency.

- **Merging J-linkage** is highly effective for plane-fitting, but its tendency to hallucinate models in geometric figure fitting effectively renders it a domain-specific approach.
- Although **RHA** is historically important and often capable of detecting a number of the models, its performance is not competitive in comparison to the other algorithms.
- **Kernel Fitting** performs very well, and has the added benefit of requiring no a-priori knowledge about the input data. However, it has a number of drawbacks: like J-linkage, it is quadratic in the number of data points; its implementation is difficult; it seems to perform poorly when there are no outliers present; and no one model selection scheme works best in any problem domain. Future work in outlier detection and model selection, as in [7], might enable Kernel Fitting to more consistently outperform the other algorithms. Since it requires no a-priori knowledge, consistently effective outlier removal and model selection schemes would be a crucial step in the development of multi-model estimation algorithms.

Before we summarize the evaluation methodology and offer future research directions, we offer concrete advice about the selection of an algorithm. Sequential RANSAC should be a first choice due to its speed and general effectiveness. Although we anticipate that Sequential RANSAC will be sufficient in many applications, we expect that there will be situations in which it is inadequate for either performance reasons or its need for a-priori knowledge of the data. In the case of performance issues, either J-linkage or Kernel Fitting seem to be a suitable next approach, and in the case of the latter, Kernel Fitting is the only effective option

for the estimation of structures without a-priori knowledge. If neither algorithm's performance is satisfactory, then one can begin integrating domain knowledge into the algorithm. For J-linkage, this will be somewhat difficult, but in some cases, there may be a way to use domain-specific information to either introduce an alternative model-selection biasing scheme to Kanazawa sampling, or adjust or modify the output. In Kernel Fitting, if one can encode information about the likelihood of sharing a model as a kernel function (as is done with the Gaussian kernel), then one can simply add some positive fraction of that kernel to the ORK and Gaussian kernels.

In this chapter, we have introduced an evaluation methodology for multi-model estimation algorithms and analyzed its application to a number of state-of-the-art methods. In addition to the usual inclusion of more algorithms (in particular, [7, 34, 13]), future studies should include motion segmentation, and real-world geometric figure-fitting data to assess the reality of the synthetic data sets commonly in use. The former is frequently used in demonstrating the effectiveness of a new algorithm, and its inclusion with a range of outliers and noise levels would augment our overall understanding of the effectiveness of the algorithms. The latter would ensure the applicability of the evaluation to real-world model estimation problems.

CHAPTER 6

CONCLUSIONS

This thesis has presented an overview of contemporary solutions to the problem of estimating multiple models from outlier contaminated data in the context of computer vision. After introducing earlier techniques that are effective only without outliers, a classic probabilistic approach to handling outliers in the single-model case, RANSAC, was introduced. In addition to describing the basic approach, we described a sampling strategy that enables rapid detection, even of high parameter models in data containing an overwhelming number of outliers. To motivate the multi-model case, and to demonstrate a practical application of these techniques in vision, we demonstrated the detection of planar surfaces in image pairs using feature correspondences. After introducing the natural extension of RANSAC to the multi-model case, we described five other contemporary multi-model estimation algorithms: MultiRANSAC, J-linkage, Merging J-linkage, Residual Histogram Analysis, and Kernel Fitting. The variety of approaches raises the practical question of which algorithm is most effective. To answer this question, we introduced a novel evaluation methodology and applied it to the algorithms described. We now present a number of high-level concluding remarks about the multi-model estimation task.

Although they were introduced as the inheritors of approaches such as least-squares or LMedS, there is a significant disconnect between multi-model algorithms for vision and their purported statistical heritage. This is perhaps best illustrated by the stairs data set, and perhaps line-fitting in general. Immediately, our mathematical formulation is suspect as human vision is never concerned with finding lines as mathematical objects, but instead with detecting line-segments. Putting this aside, there seems to be no mathematically valid reason that the diagonal

interpretation is incorrect: if we use consensus set size as a criterion, the diagonal models (even as line-segments) in Stairs5 have far more points than the straight ones; and if we use goodness of fit for inliers, neither interpretation of the stairs data set is a clear winner. The argument that no human would possibly argue for the diagonal interpretation only reinforces this point: our perception of the data set is presumably not driven by lines as a mathematical entities, but instead by the an interpretation of the entire scene, including the spatial relationship between the dots. Looking for a line, we immediately rule out the diagonal interpretation since it crosses enormous gaps of white space. Although this spatial information is highly effective in vision – both Kanazawa sampling and Kernel Fitting's inclusion of the Gaussian improve the results of algorithms – it has no mathematical basis if the model estimation task is thought of in the same fashion as least-squares.

This disconnect is driven by the nature of the problem, and perhaps of vision in general. There is no mathematical basis for Kanazawa sampling or for deciding which stairs model is correct because the problem is poorly posed, and thus the only definitions of correctness are manufactured: unlike a travelling salesman problem, we do not have an immediately accessible natural definition of what constitutes optimal or correct; and unlike ordinary least-squares, our task may not be posed as finding the maximum-likelihood solution to a problem that corresponds to real-world processes. We can invent and describe any number of criteria (goodness-of-fit, consensus size, etc.) to determine what constitutes correct, but in the end, one can only argue for their suitability by invoking their empirical effectiveness; this is in turn driven largely by how well the results agree with our visual system. If this is the case, then it seems that manufactured criteria for determining correctness are merely a heuristic that frequently agrees with our intuition, and that the poorly-defined approach of whether the results agree with our visual system is perhaps, in

the end, the best. The lack of mathematical justification and proofs of correctness are then not an issue: if we cannot even determine or justify a mathematical function to optimize, then one can have no hope of mathematically justifying the algorithm. We should not, however, be disappointed at not having an elegant mathematical foundation: the human visual system is similarly justified not by mathematical formalism, but instead empirical effectiveness.

BIBLIOGRAPHY

[1] D. Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2011. In press.

[2] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[3] C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[4] H. Chen and P. Meer. Robust regression with projection based m-estimators. *Proc. IEEE Intl. Conf. Computer Vision*, 2:878 – 885, 2003.

[5] N. Chernov and C. Lesort. Least squares fitting of circles. *Journal of Mathematical Imaging and Vision*, 23:239–252, 2005.

[6] T.-J. Chin. Kernel fitting code. `http://cs.adelaide.edu.au/~tjchin/doku.php?id=code_page`, last accessed: April 2011.

[7] T.-J. Chin, D. Suter, and H. Wang. Multi-structure model selection via kernel optimisation. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 3586 – 3593, 2010.

[8] T.-J. Chin, H. Wang, and D. Suter. The ordered residual kernel for robust motion subspace clustering. In *Advances in Neural Information Processing Systems (NIPS)*, pages 333–341, 2009.

[9] T.-J. Chin, H. Wang, and D. Suter. Robust fitting of multiple structures: The statistical learning approach. In *Proc. IEEE Intl. Conf. Computer Vision*, pages 413–420, 2009.

[10] S. Choi, T. Kim, and W. Yu. Performance evaluation of RANSAC family. In *Proc. British Machine Vision Conf.*, 2009.

[11] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. In *IEEE Trans. Pattern Analysis and Machine Intelligence*, volume 24, pages 603–619, 2002.

[12] A. Criminisi, I. D. Reid, and A. Zisserman. A plane measuring device. *Image and Vision Computing*, 17(8):625–634, 1999.

[13] A. Delong, A. Osokin, H. Isack, and Y. Boykov. Fast approximate energy minimization with label costs. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 2173–2180, 2010.

[14] R. Duda and P. Hart. Use of the Hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15:11–15, 1972.

[15] M. Fischler and R. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. of the ACM*, 24(6):381–395, June 1981.

[16] D. Fouhey, D. Scharstein, and A. Briggs. Multiple plane detection in image pairs using J-linkage. In *Proc. IEEE International Conf. on Pattern Recognition (ICPR)*, pages 336–339, 2010.

[17] D. Gallup, J. Frahm, and M. Pollefeys. Piecewise planar and non-planar stereo for urban scene reconstruction. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 1418–1425, 2010.

[18] R. I. Hartley and A. Zisserman. *Multiple View Geometry*. Cambridge University Press, Cambridge, UK, 2004.

[19] Y. Kanazawa and H. Kawakami. Detection of planar regions with uncalibrated stereo using distributions of feature points. In *Proc. British Machine Vision Conf.*, pages 247–256, 2004.

[20] I. Kasa. A curve fitting procedure and its error analysis. *IEEE Trans. Inst. Meas.*, 25:8–14, 1976.

[21] R. Larsen and M. Marx. *An Introduction to Mathematical Statistics and Its Applications*. Pearson, 2006.

[22] D.C. Lay. *Linear Algebra and Its Applications*. Addison Wesley, 2003.

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

[24] C. Papalazarou, P.M.J. Rongen, and P. H. N. de With. Multiple model estimation for the detection of curvilinear segments in medical x-ray images using sparse-plus-dense-RANSAC. In *Proc. IEEE International Conf. on Pattern Recognition (ICPR)*, pages 2484–2487, 2010.

[25] J. Prankl, M. Zillich, B. Leibe, and M. Vincze. Incremental model selection for detection and tracking of planar surfaces. In *Proc. BMVC*, pages 87.1–12, 2010.

[26] P.J. Rousseeuw. Least median of squares regression. *Journal of the American Statistical Association*, 79:871–880, 1984.

[27] S. S. Shapiro and M. B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):pp. 591–611, 1965.

[28] C.V. Stewart. Bias in robust estimation caused by discontinuities and multiple structures. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19:818–833, 1997.

[29] R. Subbarao and P. Meer. Nonlinear mean shift for clustering over analytic manifolds. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 1168–1175, 2006.

[30] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer London, 2010.

[31] J.P. Tardif. Non-iterative approach for fast and accurate vanishing point detection. In *Proc. IEEE Intl. Conf. Computer Vision*, pages 1250–1257, 2009.

[32] R. Toldo and A. Fusiello. Robust multiple structures estimation with J-linkage. In *Proc. European Conf. Computer Vision*, pages 537–547, 2008.

[33] R. Toldo and A. Fusiello. Automatic estimation of the inlier threshold in robust multiple structures fitting. In *Proc. of the 15th International Conf. on Image Analysis and Processing*, pages 123–131, 2009.

[34] R. Toldo and A. Fusiello. Real-time incremental J-linkage for robust multiple structures estimation. In *3DPVT10*, 2010.

[35] P. H. S. Torr and A. Zisserman. MLESAC: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78:2000, 2000.

[36] P.H.S. Torr. Bayesian model estimation and selection for epipolar geometry and generic manifold fitting. *International Journal of Computer Vision*, 50:35–61, 2002.

[37] R. Tron and R. Vidal. A benchmark for the comparison of 3-d motion segmentation algorithms. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 1 – 8, 2007.

[38] E. Vincent and R. Laganiere. Detecting planar homographies in an image pair. In *Proceedings of the 2nd International Symposium on Image and Signal Processing and Analysis*, pages 182–187, 2001.

[39] H. Wang, D. Mirota, and G. Hager. A generalized kernel consensus based robust estimator. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 32(1):178–184, 2010.

[40] H. Wang and D. Suter. MDPE: A very robust estimator for model fitting and range image segmentation. In *Intl. Journal of Computer Vision*, volume 59, pages 139–166, 2004.

[41] C. Wren. Perspective transform estimation. `http://alumni.media.mit.edu/~cwren/interpolator/`, last accessed: April 2011.

[42] L. Xu, E. Oja, and P. Kultanen. A new curve detection method: Randomized Hough transform (RHT). *Pattern Recognition Letters*, 11(5):331 – 338, 1990.

[43] W. Zhang and J. Kosecká. Nonparametric estimation of multiple structures with outliers. In *Dynamical Vision, ICCV 2005 and ECCV 2006 Workshops, Lecture Notes in Computer Science vol. 4358*, pages 60,74, 2006.

[44] M. Zuliani, C.S. Kenney, and B.S. Manjunath. The multiRANSAC algorithm and its application to detect planar homographies. In *Proc. IEEE Intl. Conf. Image Processing*, pages 153–156, 2005.