RANSAC for Dummies

With examples using the RANSAC toolbox for Matlab™ and more...

Marco Zuliani

marco.zuliani@gmail.com

November 25, 2008

To all the free thinkers,

who freely share their ideas.

Contents

List of Figures

Chapter 1

RANdom Sample And Consensus

This chapter is structured as follows. First we will briefly introduce the RANSAC algorithm in Section 1.1. In Section 1.2 we will present the notation and provide some definitions that will facilitate the discussion of the technical details of the algorithm (Section 1.3 and 1.4). Finally we will present related works in the literature in Section 1.5.

1.1 Introduction

The RANSAC algorithm (*RANdom Sample And Consensus*) was first introduced by Fischler and Bolles [5] in 1981 as a method to estimate the parameters of a certain model[1] starting from a set of data contaminated by large amounts of *outliers*. In this tutorial a datum is considered to be an outlier if it will not fit a model instantiated by a given set of parameters (assuming that both the model and the parameters are the "true" ones) within some error threshold that defines the maximum deviation attributable to the effect of noise. The percentage of outliers which can be handled by RANSAC can be larger than 50% of the entire data set. Such a percentage, know also as the *breakdown point*, is commonly assumed to be the practical limit for many other commonly used techniques for parameter estimation (such as all the

[1] Fischler and Bolles used RANSAC to solve the Location Determination Problem (LDP), where the goal is to determine the points in the space that project onto an image into a set of landmarks with known locations.

least squares flavours or robust techniques like M-estimators and least median of squares [10, 17, 13, 26, 19]). We want to mention here a robust estimator proposed by Stewart called MINPRAN [18] capable of estimating the parameters of a model in data sets containing more than 50% of outliers.

Despite many modifications, the RANSAC algorithm is essentially composed of two steps that are repeated in an iterative fashion (*hypothesize-and-test framework*):

- **Hypothesize**. First *minimal sample sets* (MSSs) are randomly selected from the input dataset and the model parameters are computed using *only* the elements of the MSS. The cardinality of the MSS is the smallest sufficient to determine the model parameters (as opposed to other approaches, such as *least squares*, where the parameters are estimated using *all* the data available, possibly with appropriate weights).

- **Test**. In the second step RANSAC checks which elements of the entire dataset are consistent with the model instantiated with the parameters estimated in the first step. The set of such elements is called *consensus set* (CS).

RANSAC terminates when the probability of finding a better ranked CS drops below a certain threshold. In the original formulation the ranking of the CS was its cardinality (i.e. CSs that contain more elements are ranked better than CSs that contain fewer elements).

1.2 Preliminaries

To facilitate the discussion that follows, it is convenient to introduce a suitable formalism to describe the steps for the estimation of the model parameters and for the construction of the CS. As usual we will denote vectors with boldface letters and the superscript $^{(h)}$ will indicate the h^{th} iteration. The symbol $\hat{x}$ indicates the estimated value of the quantity x. The input dataset which is composed of N elements is indicated by $D = \{\boldsymbol{d}_1, \ldots, \boldsymbol{d}_N\}$ and we will indicate a MSS with the letter s. Let $\boldsymbol{\theta}(\{\boldsymbol{d}_1, \ldots, \boldsymbol{d}_h\})$ be the parameter vector estimated

using the set of data $\{\boldsymbol{d}_1, \ldots, \boldsymbol{d}_h\}$, where $h \geq k$ and k is the cardinality of the MSS.[2] The *model manifold* $\mathcal{M}$ is defined as:

$$\mathcal{M}(\boldsymbol{\theta}) \stackrel{\text{def}}{=} \left\{\boldsymbol{d} \in \mathbb{R}^d : f_{\mathcal{M}}(\boldsymbol{d}; \boldsymbol{\theta}) = 0\right\}$$

where $\boldsymbol{\theta}$ is a parameter vector and $f_{\mathcal{M}}$ is a smooth function whose zero level set contains all the points that fit the model $\mathcal{M}$ instantiated with the parameter vector $\boldsymbol{\theta}$. We define the error associated with the datum $\boldsymbol{d}$ with respect to the manifold $\mathcal{M}(\boldsymbol{\theta})$ as the distance from $\boldsymbol{d}$ to $\mathcal{M}(\boldsymbol{\theta})$:

$$e(\boldsymbol{d}, \mathcal{M}(\boldsymbol{\theta})) \stackrel{\text{def}}{=} \min_{\boldsymbol{d}' \in \mathcal{M}(\boldsymbol{\theta})} \text{dist}(\boldsymbol{d}, \boldsymbol{d}')$$

where $\text{dist}(\cdot, \cdot)$ is an appropriate distance function. Using this error metric, we define the CS as:

$$S(\boldsymbol{\theta}) \stackrel{\text{def}}{=} \{\boldsymbol{d} \in D : e(\boldsymbol{d}, \mathcal{M}(\boldsymbol{\theta})) \leq \delta\} \tag{1.1}$$

where δ is a threshold which can either be inferred from the nature of the problem or, under certain hypothesis, estimated automatically [25]. In the former case, if we want to relate the value of δ to the statistics of the noise that affects the data and the distance function is the Euclidean norm, we can write:

$$e(\boldsymbol{d}, \mathcal{M}(\boldsymbol{\theta})) = \min_{\boldsymbol{d}' \in \mathcal{M}(\boldsymbol{\theta})} \sqrt{\sum_{i=1}^{n} (d_i - d_i')^2} = \sqrt{\sum_{i=1}^{n} (d_i - d_i^*)^2}$$

where $\boldsymbol{d}^*$ is the orthogonal projection of $\boldsymbol{d}$ onto the manifold $\mathcal{M}(\boldsymbol{\theta})$. Now suppose that the datum $\boldsymbol{d}$ is affected by Gaussian noise $\boldsymbol{\eta} \sim \mathcal{N}(\boldsymbol{0}, \sigma_\eta I)$ so that $\boldsymbol{\eta} = \boldsymbol{d} - \boldsymbol{d}^*$. Our goal is to calculate the value of δ that bounds, with a given probability P_{inlier}, the error generated by a true inlier contaminated with Gaussian noise. More formally we want to find the value δ

[2] Suppose we want to estimate a line: in this case the cardinality of the MSS is 2, since at least two distinct points are needed to uniquely define a line.

such that:

$$P[e(\boldsymbol{d}, \mathcal{M}(\boldsymbol{\theta})) \leq \delta] = P_{inlier} \tag{1.2}$$

Following [7], p. 118, we can write the following chain of equations:

$$P[e(\boldsymbol{d}, \mathcal{M}(\boldsymbol{\theta})) \leq \delta] = P\left[\sum_{i=1}^{n} \eta_i^2 \leq \delta^2\right] = P\left[\sum_{i=1}^{n} \left(\frac{\eta_i}{\sigma_\eta}\right)^2 \leq \frac{\delta^2}{\sigma_\eta^2}\right]$$

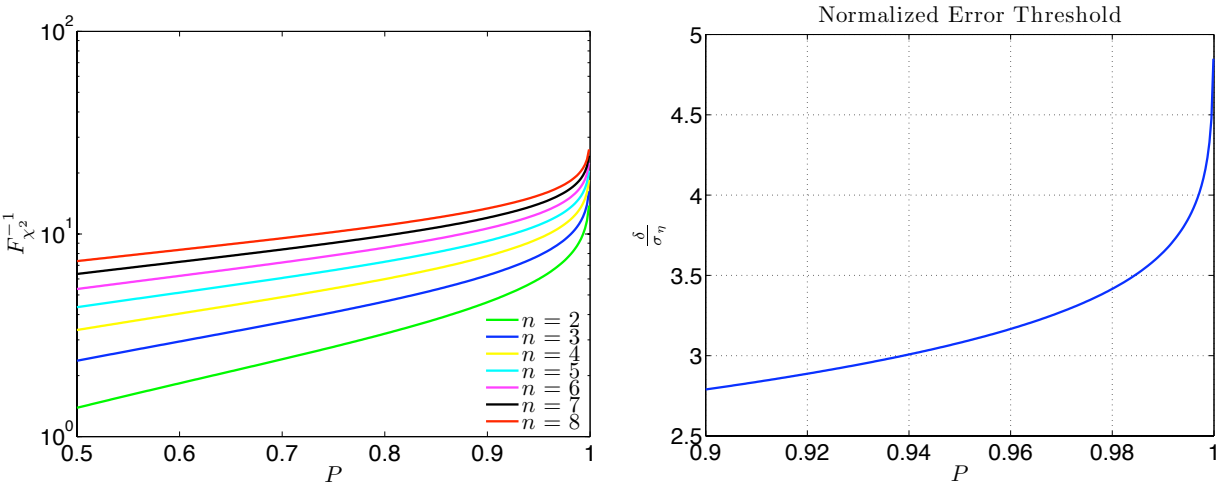
and since $\eta_i/\sigma_\eta \sim \mathcal{N}(0,1)$, the random variable $\sum_{i=1}^{n} \left(\frac{\eta_i}{\sigma_\eta}\right)^2$ has a χ^2_n distribution. Hence:

$$\delta = \sigma_\eta \sqrt{F^{-1}_{\chi^2_n}(P_{inlier})} \tag{1.3}$$

where $F^{-1}_{\chi^2_n}$ is the inverse cumulative distribution function associated with a χ^2_n random variable. Figure 1.1(a) displays the function $F^{-1}_{\chi^2_n}$ for different values of n. Note that when P_{inlier} tends to one (i.e. we want to pick an error threshold such that all the inliers will be considered) the value of $F^{-1}_{\chi^2_n}$ diverges to infinity. Values of P_{inlier} close to one will return a large threshold with the risk of including some outliers as well. On the other hand, too small values of P_{inlier} will generate a value for δ which is too tight, and possibly some inliers will be discarded.

1.3 RANSAC Overview

A pictorial representation of the RANSAC fundamental iteration together with the notation just introduced is shown in Figure 1.2. As mentioned before, the RANSAC algorithm is composed of two steps that are repeated in an iterative fashion (hypothesize-and-test framework). First a MSS $s^{(h)}$ is selected from the input dataset and the model parameters $\boldsymbol{\theta}^{(h)}$ are computed using *only* the elements of the selected MSS. Then, in the second step, RANSAC checks which elements in the dataset D are consistent with the model instantiated with the estimated parameters and, if it is the case, it updates the current best CS S^* (which, in



(a) (b)

Figure 1.1: Figure (a) shows the function $F^{-1}_{\chi^2_n}$ for different values of n. Note that the vertical axis values are logarithmically spaced). Figure (b) displays the normalized error threshold for the symmetric transfer error of planar feature points ($n = 2 + 2 = 4$).

the original Fischler and Bolles formulation, is the CS with the largest cardinality). The algorithm terminates when the probability of finding a better CS drops below a certain threshold. In the next paragraphs we will discuss how to estimate the number of iterations that RANSAC is supposed to perform and other approaches to rank the CSs.

1.3.1 How many iterations?

Let q be the probability of sampling from the dataset D a MSS s that produces an accurate estimate of the model parameters. Consequently, the probability of picking a MSS at least one outlier (i.e. a MSS that produces a biased estimate of the true model parameter vector) is $1-q$. If we construct h different MSS, then the probability that *all of them are contaminated by outliers* is $(1-q)^h$ (this quantity tends to zero for h going to infinity: sooner or later we will pick something good!). We would like to pick h (i.e. the number of iterations) large enough so that the probability $(1-q)^h$ is smaller or equal than a certain probability threshold ε (often called *alarm rate*), i.e. $(1-q)^h \leq \varepsilon$. The previous relation can be inverted so that

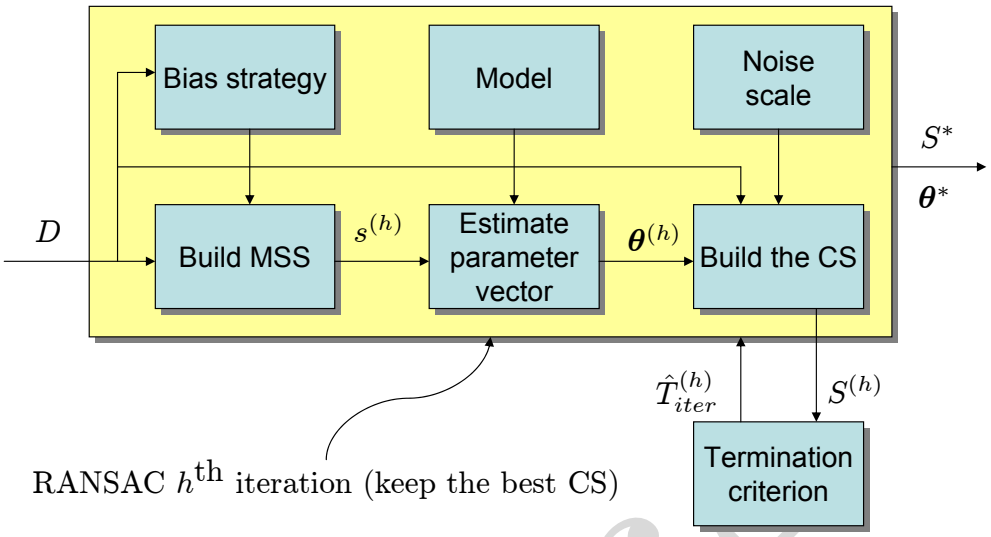


Figure 1.2: Pictorial representation of the fundamental RANSAC iteration.

we can write:

$$h \geq \left\lceil \frac{\log \varepsilon}{\log (1-q)} \right\rceil \tag{1.4}$$

where $\lceil x \rceil$ denotes the smallest integer larger than x. Therefore we can set:

$$\hat{T}_{iter} = \left\lceil \frac{\log \varepsilon}{\log (1-q)} \right\rceil \tag{1.5}$$

1.3.2 Constructing the MSSs and Calculating q

If we imagine that the inliers inside the dataset D are noise free, then any MSS entirely composed of inliers will generate the "true" value of the parameter vector.[3] If all the elements in the dataset have *the same probability of being selected*, then the probability of obtaining a MSS composed only of inliers is:

$$q = \frac{\binom{N_I}{k}}{\binom{N}{k}} = \frac{N_I!(N-k)!}{N!(N_I-k)!} = \prod_{i=0}^{k-1} \frac{N_I - i}{N - i} \tag{1.6}$$

[3] As long as we disregard numerical approximations and we pick the noise threshold δ to be an arbitrarily small positive number.

where N_I is the total number of inliers. Note that if $N, N_I \gg k$, then q is approximately equivalent to the probability of picking for k times an inlier from the data set (with re-insertion). In fact:

$$q = \prod_{i=0}^{k-1} \frac{N_I - i}{N - i} \approx \left(\frac{N_I}{N} \right)^k \tag{1.7}$$

Unfortunately, to compute q we should know N_I which is generally not known a priori. However it is easy to verify that for any $\hat{N}_I \leq N_I$ we have $q(\hat{N}_I) \leq q(N_I)$ and consequently $(1 - q(N_I))^h \geq \left(1 - q(\hat{N}_I)\right)^h$ (where we made explicit the dependency of q on the number of inliers). Therefore we can estimate the maximum number of iterations using the cardinality of the *largest set of inliers found so far* (call this $\hat{N}_I$), which can be regarded as a conservative estimate of N_I. Hence, the iteration threshold can be fixed to:

$$\hat{T}_{iter} = \left\lceil \frac{\log \varepsilon}{\log \left(1 - q(\hat{N}_I)\right)} \right\rceil \tag{1.8}$$

Note however that some researchers (for example Tordoff et al. [22]) consider this threshold on the number of the iterations to be *over-optimistic*, since in presence of noisy data it is not enough to generate a MSS composed only of inliers to obtain a reliable estimate of the parameters of the model. This observation motivates some of the considerations that will be outlined in the next section.

Biased Sampling

Sometimes there exists some a priori information regarding the probability that a datum is an inlier or an outlier. This information can be used to guide the sampling procedure in a more principled manner (see also [4, 22]). In the biased sampling case the probability of picking the element $\boldsymbol{d}_j$ at the l^{th} draw will be denoted as:

$$P(\boldsymbol{d}_j | D \setminus \{\boldsymbol{d}_{i(1)}, \ldots, \boldsymbol{d}_{i(l)}\}) \tag{1.9}$$

where $i^{(l)}$ denotes the index of the element in the data set D obtained at the l^{th} draw. The probability (1.9) is modified by the *bias* $\omega(\boldsymbol{d}) \in \mathbb{R}_+$, which incorporates the a priori information regarding the likelihood of an element to be an inlier. In an ideal case the weight of any inlier is larger than the weight of any outlier, but in real life scenarios this hypothesis does not necessarily hold true (otherwise the detection of the inliers versus the outlier would be *greatly* simplified). Clearly, the computation of the probability of obtaining a MSS composed only of inliers is complicated by the fact that the order in which the elements are sampled *matters*: the element picked at the l^{th} draw modifies the probability of picking a certain element at the $(l+1)^{\text{th}}$ draw. Therefore, in general, the probability of sampling k inliers in the order specified by the indices $i_1, \ldots, i_k$ is:

$$\begin{aligned} P(\boldsymbol{d}_{i_1}, \ldots, \boldsymbol{d}_{i_k}) &= \\ (1^{\text{st}} \text{ draw}) \quad & P(\boldsymbol{d}_{i_1} \in D_I) \cdot \\ (2^{\text{nd}} \text{ draw}) \quad & P(\boldsymbol{d}_{i_2} \in D_I \setminus \{\boldsymbol{d}_{i_1}\}) \cdot \\ & \vdots \\ (k^{\text{th}} \text{ draw}) \quad & P(\boldsymbol{d}_{i_k} \in D_I \setminus \{\boldsymbol{d}_{i_1}, \ldots, \boldsymbol{d}_{i_{k-1}}\}) \end{aligned} \tag{1.10}$$

The previous expression confirms what was anticipated earlier, i.e. that the computation of q is complicated by the fact that the order of selection of the elements matters. To compute q we need to enumerate all the possible permutations of k inliers (recall that k is the cardinality of a MSS), which turn out to be:

$$\text{Number of permutations} = \frac{N_I!}{(N_I - k)!} = \prod_{i=0}^{k-1} (N_I - i)$$

If $\mathcal{I}$ is the set of all the permutations of k indices of inliers then we can write:

$$q = \sum_{\{i_1, \ldots, i_k\} \in \mathcal{I}} P(\boldsymbol{d}_{i_1}, \ldots, \boldsymbol{d}_{i_k}) \tag{1.11}$$

This expression can be used to design a function that updates the probability every time the number of inliers is updated. However this is not a trivial task, since the number of the terms of the summation can be extremely large[4].

1.3.3 Ranking the Consensus Set

In the original formulation of RANSAC, the ranking r of a consensus set was nothing but its cardinality:

$$r(CS) \stackrel{\text{def}}{=} |CS|$$

In other words CSs that are larger are ranked higher. Thus RANSAC can be seen as an *optimization* algorithm [23] that *minimizes* the cost function:

$$C_{\mathcal{M}}(D;\boldsymbol{\theta}) = \sum_{i=1}^{N} \rho(\boldsymbol{d}_i, \mathcal{M}(\boldsymbol{\theta}))$$

where:

$$\rho(\boldsymbol{d}, \mathcal{M}(\boldsymbol{\theta})) = \begin{cases} 0 & e(\boldsymbol{d}, \mathcal{M}(\boldsymbol{\theta})) \leq \delta \\ 1 & \text{otherwise} \end{cases}$$

This observation leads immediately to an approach based on *M-estimators* [10, 17, 26], the idea being to substitute the function ρ with a more sensible one. The first function that we consider is the following:

$$\rho(\boldsymbol{d}, \mathcal{M}(\boldsymbol{\theta})) = \begin{cases} e(\boldsymbol{d}, \mathcal{M}(\boldsymbol{\theta})) & e(\boldsymbol{d}, \mathcal{M}(\boldsymbol{\theta})) \leq \delta \\ \delta & \text{otherwise} \end{cases}$$

Using this re-descending M-estimator, the inliers are scored according to their *fitness to the model*, while the outilers are still given a constant weight. Torr et al. refer to this modification of the original RANSAC algorithm with the name MSAC, i.e. M-estimator SAmple and

[4] If the number of inliers is $N_I = 100$ and the cardinality of the MMS is $k = 4$ the number of terms in the summation (1.11) is almost four millions.

Consensus. We agree with their claim:

> The implementation of this new method yields a modest to hefty benefit to all robust estimations with absolutely no additional computational burden. Once this is understood there is no reason to use RANSAC in preference to this method.

A further improvement can be obtained modifying RANSAC in order to maximize the likelihood of the solution. This is the approach implemented by MLESAC [23], a variation of RANSAC that evaluates the *likelihood* of the hypothesis by representing the error distribution as a *mixture model*. More precisely, the probability distribution of the error for entire dataset (comprising both the iniers and outliers) can be modeled as the mixture of two distributions (one taking into account the inliers, tho other the outliers) so that the likelihood can be expressed as:

$$p[e(D,\mathcal{M}(\boldsymbol{\theta}))|\boldsymbol{\theta}] = \prod_{i=1}^{N}\Big(\gamma p\left[e(\boldsymbol{d}_i,\mathcal{M}(\boldsymbol{\theta})|\text{the } i^{\text{th}} \text{ element is an inlier}\right] + (1-\gamma)p\left[e(\boldsymbol{d}_i,\mathcal{M}(\boldsymbol{\theta})|\text{the } i^{\text{th}} \text{ element is an outlier}\right]\Big)$$

A common practice is to maximize the log-likelihood, which is given by:

$$L[e(D,\mathcal{M}(\boldsymbol{\theta}))|\boldsymbol{\theta}] = \sum_{i=1}^{N}\log\Big(\gamma p\left[e(\boldsymbol{d}_i,\mathcal{M}(\boldsymbol{\theta}))|\text{the } i^{\text{th}} \text{ element is an inlier}\right] + (1-\gamma)p\left[e(\boldsymbol{d}_i,\mathcal{M}(\boldsymbol{\theta}))|\text{the } i^{\text{th}} \text{ element is an outlier}\right]\Big)$$

A common practice is to model the error distributions for the inliers with a Gaussian model:

$$p\left[e(\boldsymbol{d}_i,\mathcal{M}(\boldsymbol{\theta}))|\text{the } i^{\text{th}} \text{ element is an inlier}\right] = \frac{1}{Z}\exp\left(-\frac{e(\boldsymbol{d}_i,\mathcal{M}(\boldsymbol{\theta})^2}{2\sigma_\eta^2}\right)$$

where Z is the appropriate normalization constant and σ_η indicates the noise standard deviation (see also equation (1.3)). The error statistics for the outliers is described by a uniform distribution:

$$p\left[e(\boldsymbol{d}_i, \mathcal{M}(\boldsymbol{\theta}))|\text{the } i^{\text{th}} \text{ element is an outlier}\right] = \begin{cases} \frac{1}{2e_{max}} & |e(\boldsymbol{d}_i, \mathcal{M}(\boldsymbol{\theta}))| \leq e_{max} \\ 0 & \text{otherwise} \end{cases}$$

where e_{max} represents the largest error induced by the outliers (an estimate of such quantity can be obtained from the context the data are drawn from). Note that in this case we need to estimate two quantities: the parameter $\boldsymbol{\theta}$ that maximizes the likelihood and the mixture coefficient γ. This is traditionally done using the *expectation maximization approach*[5].

1.4 Computational Complexity

In this section we will briefly discuss the computational complexity associated to RANSAC.

1.4.1 Hypothesize Step

At each iteration we need to compute the parameters of the model starting from the MSS. Let's define the cost of this operation to be $C_{estimate}(m)$.

1.4.2 Test Step

Once the parameters have been estimated we need to evaluate how many data fit the model. If the cost associated to compute the fitting of each element is $C_{fitting}$, then the overall complexity is $O(NC_{fitting})$. Note that the approach by Chum et al. [3] aims at reducing the complexity of this stage.

[5] A good practical reference with Matlab™ code to perform this task can be found in [15]

1.4.3 Overall Complexity

Putting together the cost of the hypothesize and test steps, the overall complexity of the algorithm in the worst case scenario is:

$$\text{Complexity} = O\left(\hat{T}_{iter}(C_{estimate}(m) + NC_{fitting})\right) \tag{1.12}$$

1.5 Other RANSAC Flavors

Since 1981 RANSAC has become a fundamental tool in the computer vision and image processing community. In 2006, for the 25^{th} anniversary of the algorithm, a workshop was organized at the International Conference on Computer Vision and Pattern Recognition (CVPR) to summarize the most recent contributions and variations to the original algorithm, mostly meant to improve the speed of the algorithm, the robustness and accuracy of the estimated solution and to decrease the dependency from user defined constants. We regard with amusement the effort of the researchers in finding colorful variations to the name of the original algorithm. Some of these approaches (with their relative names) will be described in the following paragraphs.

As pointed out by Torr et al. , RANSAC can be sensitive to the choice of the correct noise threshold that defines which data points fit a model instantiated with a certain set of parameters. If such threshold is too large, then all the hypotheses tend to be ranked equally (good). On the other hand, when the noise threshold is too small, the estimated parameters tend to be unstable (i.e. by simply adding or removing a datum to the set of inliers, the estimate of the parameters may fluctuate). To partially compensate for this undesirable effect, Torr et al. proposed two modification of RANSAC called MSAC (M-estimator SAmple and Consensus) and MLESAC (Maximum Likelihood Estimation SAmple and Consensus) [23]. The main idea is to evaluate the quality of the consensus set (i.e. the data that fit a model and a certain set of parameters) calculating its likelihood (whereas in the original

formulation by Fischler and Bolles the rank was the cardinality of such set). More details have been provided in Section 1.3.3. An extension to MLESAC which keeps into account the prior probabilities associated to the input data set is proposed by Tordoff et al. in [22]. The resulting algorithm is dubbed Guided-MLESAC. Along similar lines, Chum proposed to guide the sampling procedure if some a priori information regarding the input data is known, i.e. whether a datum is likely to be an inlier or an outlier. The proposed approach is called PROSAC, PROgressive SAmple Consensus [4].

Chum et al. also proposed a randomized version of RANSAC called R-RANSAC [3] to reduce the computational burden to identify a good CS. The basic idea is to initially evaluate the goodness of the currently instantiated model using only a reduced set of points instead of the entire data set. A sound strategy will tell with high confidence when it is the case to evaluate the fitting of the entire data set or when the model can be readily discarded. It is reasonable to think that the impact of this approach is more relevant in cases where the percentage of inliers is large. The type of strategy proposed by Chum et al. is called *preemption scheme*. In [16], Nistér proposed a paradigm called Preemptive RANSAC that allows real time robust estimation of the structure of a scene and of the motion of the camera. The core idea of the approach consists in generating a fixed number of hypothesis so that the comparison happens with respect to the quality of the generated hypothesis rather than against some absolute quality metric.

Other researchers tried to cope with difficult situations where the noise scale is not know and/or multiple model instances are present. The first problem has been tackled in the the work by Wang and Suter [25]. Some solutions for the second problem where instead proposed in [27] and more recently in [21]. More specifically, Toldo et al. represent each datum with the characteristic function of the set of random models that fit the point. Then multiple models are revealed as clusters which group the points supporting the same model. The clustering algorithm, called J-linkage, does not require prior specification of the number of models, nor it necessitate manual parameters tuning.

RANSAC has also been tailored for recursive state estimation applications, where the input measurements are corrupted by outliers and Kalman filter approaches, which rely on a Gaussian distribution of the measurement error, are doomed to fail. Such an approach, dubbed KALMANSAC, is presented in [24].

Chapter 2

RANSAC at Work

In this chapter we will describe some simple applications of RANSAC. To facilitate the discussion we will utilize the RANSAC Toolbox for Matlab™.

2.1 The RANSAC Toolbox for Matlab™

In this section we briefly introduce the RANSAC Toolbox for Matlab™. This toolbox is highly customizable and it is designed to be a flexible research/didactic resource.

To install the package first download the `.zip` archive from the Mathworks file exchange website. After unzipping the archive run the script `SetPathLocal.m` to set the local paths and some support global variables. To check if updates are available run the script `RANSAC_update.m`. The sub-directory `Examples` contains some example scripts to estimate lines, planes and homographies. Each example will be described in some more detail in the next sections.

2.1.1 RANSAC.m

This is the driver function that implements the RANSAC algorithm. In this section we will describe in detail the options that are accessible to the user and that can be listed issuing the usual command `help RANSAC`.

Input Parameters

- `X`: this matrix collects the input data set. Its dimensions are $d \times N$ and the i^{th} column contains the datum $\boldsymbol{d}_i$.

- `options`: this structure collects the options for the algorithm. If an option is not specified the function will either use a default value or issue an error message (if a default value for the desired option does not exist).

  - `sigma`: scalar value of the noise standard deviation under the assumption that the components of the input are corrupted by Gaussian noise with covariance matrix $\sigma_\eta I$. See the discussion in Section 1.2 and Equation (1.3).

  - `P_inlier`: the probability that a point whose fitting error is less or equal than δ is actually an inlier (see Equation (1.2)). Default value $P_{inlier} = 0.99$.

  - `T_noise_squared`: when this value is provided, it will force the error threshold to be equal to δ. If this value is provided, the calculation of the threshold using the value of σ_η (see Equation (1.3)) is *overridden*.

  - `epsilon`: False alarm rate, i.e. the probability that the algorithm throughout all the iterations will *never* sample a MSS containing *only* inliers (see Section 1.3.1). Default value $\varepsilon = 0.001$.

  - `Ps`: Probability mass distribution to bias the sampling: `Ps(i)` $= P(\boldsymbol{d}_i \text{ is sampled})$. Note that $P_s \in \mathbb{R}^N_+$ must be normalized so that $\sum_{i=1}^N P(\boldsymbol{d}_i \text{ is sampled}) = 1$. The default value is a uniform distribution, i.e. $P(\boldsymbol{d}_i \text{ is sampled}) = \frac{1}{N}$.

  - `ind_tabu`: logical array that identifies the elements that *should not* be selected to construct the MSSs during the sampling (default is empty).

  - `validateMSS_fun`: function that validates the MSS sampled from the dataset (see 2.2.4). A typical template for this function is found in Appendix D.1.1.

- `est_fun`: function that returns the estimate of the parameter vector starting from a set of data. A typical template for this function is found in Appendix D.1.2.
- `validateTheta_fun`: function that validates the parameter vector estimated from the MSS. A typical template for this function is found in Appendix D.1.3.
- `man_fun`: function that returns the fitting error of the data. A typical template for this function is found in Appendix D.1.4.
- `mode`: A string that specifies the algorithm flavour:
  - `'RANSAC'`: original Fischler and Bolles formulation [5]
  - `'MSAC'`: Torr and Zisserman formulation [23]
- `max_iters`: maximum number of iterations allowed (overrides the threshold (1.5), default = ∞)
- `min_iters`: minimum number of iterations required (overrides the threshold (1.5), default = 0)
- `max_no_updates`: maximum number of iterations with no updates (default = 0)
- `reestimate`: true to resestimate the parameter vector using all the detected inliers (default = false)
- `fix_seed`: set to true to fix the seed of the random number generator so that the results on the same data set are *repeatable* (default = false)
- `verbose`: true for verbose output (default = true)
- `notify_iters`: if the verbose output is enabled this parameter specifies the maximim number of iterations that will occur before a progress messages is displayed (default is empty).

Output Parameters

- `results`: a structure containing the results of the estimation. Its fields are the following:

  - `Theta`: The vector of the estimates of the parameters of the model (see Section 1.2).
  - `E`: A $1 \times N$ vector containing the fitting error for each data point calculated using the function `man_fun`.
  - `CS`: A $1 \times N$ logical vector whose entries are true for the data points that have been labelled as inliers.
  - `J`: Overall ranking of the solution (see Section 1.3.3).
  - `iter`: Number of iterations performed by RANSAC before converging.

- `results`: this structure collects the options for the algorithm, after the default values have been set.

2.2 Some Examples Using the RANSAC Toolbox

2.2.1 Estimating Lines

In this section we will describe how to perform linear regression in presence of outliers utilizing RANSAC.

Parameter Estimation

The implicit model of a line is $\theta_1 x_1 + \theta x_2 + \theta_3 = 0$. The scaling of the parameter vector $\boldsymbol{\theta}$ is immaterial, therefore we just need to points two equations (in other words two points will suffice to estimate a line, i.e. the cardinality of the MSS is 2). In case we are dealing with N points the following equation must hold:

$$\sum_{j=1}^{N} \left(\theta_1 \boldsymbol{x}_1^{(j)} + \theta_2 \boldsymbol{x}_2^{(j)} + \theta_3 \right) = 0$$

This implies that:

$$\theta_3 = \frac{1}{N} \sum_{j=1}^{N} \left(\theta_1 \boldsymbol{x}_1^{(j)} + \theta_2 \boldsymbol{x}_2^{(j)} \right) = \theta_1 \bar{x} + \theta_2 \bar{y} \tag{2.1}$$

where the barred quantities indicate the sample means. If we define:

$$\Delta^{(j)} \stackrel{\text{def}}{=} \sum_{j=1}^{N} \left(\theta_1 \boldsymbol{x}_1^{(j)} + \theta_2 \boldsymbol{x}_2^{(j)} \right) - \theta_3 = \sum_{j=1}^{N} \left(\theta_1 \tilde{\boldsymbol{x}}_1^{(j)} + \theta_2 \tilde{\boldsymbol{x}}_2^{(j)} \right)$$

(where the tilde denotes the corresponding quantities after the mean value has been removed) in order to estimate the line of interest we can minimize the cost function:

$$J = \sum_{j=1}^{N} \left(\Delta^{(j)} \right)^2 \tag{2.2}$$

To avoid the trivial solution given by $\theta_1 = \theta_2 = 0$ we can perform the minimization subject to the constraint $\theta_1^2 + \theta_2 = 1$. If we define the following matrix:

$$A \stackrel{\text{def}}{=} \begin{bmatrix} \tilde{\boldsymbol{x}}_1^{(1)} & \tilde{\boldsymbol{x}}_2^{(1)} \\ \vdots & \vdots \\ \tilde{\boldsymbol{x}}_1^{(N)} & \tilde{\boldsymbol{x}}_2^{(N)} \end{bmatrix} \tag{2.3}$$

the minimization problem can be rewritten in matrix form as:

$$\boldsymbol{\psi}^* = \underset{\substack{\boldsymbol{\psi}\in\mathbb{R}^2 \\ \|\boldsymbol{\psi}\|=1}}{\operatorname{argmin}} \|A\boldsymbol{\psi}\|^2 \tag{2.4}$$

where:

$$\boldsymbol{\psi} \stackrel{\text{def}}{=} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$$

This type of problems can be readily solved using the SVD decomposition of A (see Appendix B.4.2). Note that once we estimate $\boldsymbol{\psi}$ we can immediately recover θ_3 from (2.1). If we are willing to sacrifice some accuracy in the estimation of the parameters, we can solve the previous problem in closed form. The SVD decomposition of the matrix A is related to the eigen decomposition of $A^T A$ as stated in Lemma B.2.1. Therefore the solution of the problem is obtained calculating the eigenvector corresponding to the smallest eigenvalue. A quick way to compute these quantities is described in Appendix B.3. The source code that implements the line parameter estimation is listed in Appendix D.2.1.

Remark 2.2.1. *The least square problem to estimate the parameters of the line can be set up in a different manner. If we define the matrix A to be:*

$$A \stackrel{\text{def}}{=} \begin{bmatrix} \tilde{\boldsymbol{x}}_1^{(1)} & \tilde{\boldsymbol{x}}_2^{(1)} & 1 \\ \vdots & \vdots & \\ \tilde{\boldsymbol{x}}_1^{(N)} & \tilde{\boldsymbol{x}}_2^{(N)} & 1 \end{bmatrix}$$

then we need to solve the new constrained minimization problem:

$$\boldsymbol{\theta}^* = \underset{\substack{\boldsymbol{\theta}\in\mathbb{R}^3 \\ \|\boldsymbol{\theta}\|=1}}{\operatorname{argmin}} \|A\boldsymbol{\theta}\|^2$$

This formulation is thoretically equivalent to (2.4), however practically it is not numerically sound. Without digging into too many analytical details, the reason is because the quantities that appear in A are not well scaled, more specifically the presence of ones in the last column (a quantity known with no error uncertainty) contrast with the remaining components of the matrix which are measurements affected by noise.

Error Estimation

The fitting error is defined as the distance between the point $\boldsymbol{x}$ and the line $\mathcal{M}(\boldsymbol{\theta})$ which is expressed by the formula:

$$e(\boldsymbol{x}, \mathcal{M}(\boldsymbol{\theta}))^2 = \frac{(\theta_1 x_1 + \theta_2 x_2 + \theta_3)^2}{\theta_1^2 + \theta_2^2}$$

If we assume that the point $\boldsymbol{x}$ is affected by Gaussian noise, then $e(\boldsymbol{x}, \mathcal{M}(\boldsymbol{\theta}))^2$ is χ^2_2 distributed. The function that computes the fitting error of the data is listed in Appendix D.2.1.

2.2.2 Estimating Planes

We will describe how it is possible to utilize RANSAC to identify points in $\mathbb{R}^3$ that belong to a plane in the space (i.e. to an affine linear manifold in $\mathbb{R}^3$) and simultaneously estimate the parameters of such manifold.

Parameter Estimation

The implicit model of the plane containing the points $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N_I)}$ can be described by a system of linear equations:

$$\begin{cases} \theta_1 x_1^{(1)} + \theta_2 x_2^{(1)} + \theta_3 x_3^{(1)} + \theta_4 = 0 \\ \vdots \quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \vdots \quad \vdots \\ \theta_1 x_1^{(N)} + \theta_2 x_2^{(N)} + \theta_3 x_3^{(N)} + \theta_4 = 0 \end{cases}$$

It is convenient to group these equations in matrix form, so that we can write:

$$\begin{bmatrix} \left(\boldsymbol{x}^{(1)}\right)^T & 1 \\ \vdots & \vdots \\ \left(\boldsymbol{x}^{(N)}\right)^T & 1 \end{bmatrix} \boldsymbol{\theta} = A\boldsymbol{\theta} = 0 \tag{2.5}$$

Therefore the parameter vector that instantiates the plane containing the points $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N_I)}$ is given by:

$$\boldsymbol{\theta}^* = \underset{\substack{\boldsymbol{\theta}\in\mathbb{R}^4 \\ \|\boldsymbol{\theta}\|=1}}{\operatorname{argmin}} \|A\boldsymbol{\theta}\|^2 \tag{2.6}$$

Note that the model has only *three* degrees of freedom, since the length of the parameter vector is immaterial. The solution of (2.6) can be readily obtained using the SVD decomposition of A (see Appendix B.4.2). Since a plane is uniquely defined at least by three points, the cardinality of the MSS is $k = 3$. Therefore, using the notation introduced in Section 1.2, the model manifold (i.e. an affine space) can be expressed as:

$$\mathcal{M}(\boldsymbol{\theta}) \stackrel{\text{def}}{=} \left\{ \boldsymbol{x} \in \mathbb{R}^3 : \begin{bmatrix} \boldsymbol{x}^T & 1 \end{bmatrix} \boldsymbol{\theta} = 0 \right\}$$

The Matlab™function that implements the parameter estimation routine is listed in Appendix D.2.2.

Error Estimation

The fitting error is defined as the distance between the point $\boldsymbol{x}$ and the manifold $\mathcal{M}(\boldsymbol{\theta})$. For each data point $\boldsymbol{x}$ the unique solution of $\text{argmin}_{\boldsymbol{x}' \in \mathcal{M}(\boldsymbol{\theta})} \|\boldsymbol{x} - \boldsymbol{x}'\|^2$ can be obtained using the method of the Lagrange multipliers. The squared distance between $\boldsymbol{x}$ and its orthogonal projection onto $\mathcal{M}(\boldsymbol{\theta})$ (i.e. the plane) is given by the well know Line expression:

$$e(\boldsymbol{x}, \mathcal{M}(\boldsymbol{\theta}))^2 = \frac{\left(\begin{bmatrix} \boldsymbol{x}^T & 1 \end{bmatrix} \boldsymbol{\theta} \right)^2}{\|\boldsymbol{\theta}_{1:3}\|^2}$$

If we assume that the point $\boldsymbol{x}$ is affected by Gaussian noise, then $e(\boldsymbol{x}, \mathcal{M}(\boldsymbol{\theta}))^2$ is χ^2_3 distributed. The function that computes the fitting error of the data is listed in Appendix D.2.2.

2.2.3 Estimating A Rotation Scaling and Translation

We will describe how it is possible to utilize RANSAC to identify points in $\mathbb{R}^2$ that are related via a rotation, a scaling and a translation (RST) and simultaneously estimate the parameters of such transformation. The functional form of the RST transformation (in Euclidean coordinates) is given by:

$$\boldsymbol{T}_{\boldsymbol{\theta}}(\boldsymbol{y}) = s \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \boldsymbol{y} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} \tag{2.7}$$

The parameter vector is $\boldsymbol{\theta} = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \theta_4 \end{bmatrix}^T = \begin{bmatrix} s\cos\phi & s\sin\phi & t_1 & t_2 \end{bmatrix}^T$.

Parameter Estimation

The parameters of an RST transformation can be estimated in a least square sense after reordering (2.7) as:

$$\boldsymbol{T}_{\boldsymbol{\theta}}(\boldsymbol{y}) = \begin{bmatrix} y_1 & -y_2 & 1 & 0 \\ y_2 & y_1 & 0 & 1 \end{bmatrix} \boldsymbol{\theta} = \begin{bmatrix} S & I \end{bmatrix} \boldsymbol{\theta} \tag{2.8}$$

where S is skew symmetric and I is the identity. Each point correspondence contributes for two equations and since the total number of parameters is four we need at least two point correspondences. Thus the cardinality of the MSS is $k = 2$. To estimate the parameter models starting from the MSS we need to solve a system in four equations and four unknowns. In principle we could use the same least square approach[1] used to solve the overdetermined linear system obtained from more than two corresponding point pairs. However the parameter estimation using solely the elements in the MSS should be *as fast as possible* since it needs to be performed in every iteration of RANSAC, as shown by equation (1.12) (whereas the estimation starting from the CS is in general performed only at the end). We will demonstrate how a more careful analysis of the previous equations can lead to a relevant improvement in the performance. Grouping the equations (2.8) obtained using two point correspondences we can write:

$$\begin{bmatrix} \boldsymbol{T}_{\boldsymbol{\theta}}(\boldsymbol{y}^{(1)}) \\ \boldsymbol{T}_{\boldsymbol{\theta}}(\boldsymbol{y}^{(2)}) \end{bmatrix} = \begin{bmatrix} S^{(1)} & I \\ S^{(2)} & I \end{bmatrix} \boldsymbol{\theta}$$

If we define the matrix $M \stackrel{\text{def}}{=} S^{(1)} - S^{(2)}$ (which is nothing but the Schur complement of the bottom left identity), we can decompose the previous equation as:

$$\begin{bmatrix} \boldsymbol{T}_{\boldsymbol{\theta}}(\boldsymbol{y}^{(1)}) \\ \boldsymbol{T}_{\boldsymbol{\theta}}(\boldsymbol{y}^{(2)}) \end{bmatrix} = \begin{bmatrix} I & I \\ 0 & I \end{bmatrix} \begin{bmatrix} M & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} I & 0 \\ S^{(2)} & I \end{bmatrix} \boldsymbol{\theta} = \begin{bmatrix} M & I \\ 0 & I \end{bmatrix} \begin{bmatrix} I & 0 \\ S^{(2)} & I \end{bmatrix} \boldsymbol{\theta}$$

[1] Least squares can be solved efficiently via Gaussian elimination. This is implemented in Matlab™ using the compact notation `x=A\b;`. See also Appendix B.4.1.

Bringing the matrices that multiply the parameter vector on the left hand side of the previous equality we get:

$$\begin{bmatrix} I & 0 \\ -S^{(2)} & I \end{bmatrix} \begin{bmatrix} M^{-1} & -M^{-1} \\ 0 & I \end{bmatrix} \begin{bmatrix} \boldsymbol{T}_{\boldsymbol{\theta}}(\boldsymbol{y}^{(1)}) \\ \boldsymbol{T}_{\boldsymbol{\theta}}(\boldsymbol{y}^{(2)}) \end{bmatrix} =$$

$$= \begin{bmatrix} I & 0 \\ -S^{(2)} & I \end{bmatrix} \begin{bmatrix} M^{-1}\left(\boldsymbol{T}_{\boldsymbol{\theta}}(\boldsymbol{y}^{(1)}) - \boldsymbol{T}_{\boldsymbol{\theta}}(\boldsymbol{y}^{(2)})\right) \\ \boldsymbol{T}_{\boldsymbol{\theta}}(\boldsymbol{y}^{(2)}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\theta}_{1:2} \\ \boldsymbol{\theta}_{3:4} \end{bmatrix}$$

Thus we can recover the parameter vector via simple algebraic operations and back substitutions, first computing $\boldsymbol{\theta}_{1:2} = M^{-1}\left(\boldsymbol{T}_{\boldsymbol{\theta}}(\boldsymbol{y}^{(1)}) - \boldsymbol{T}_{\boldsymbol{\theta}}(\boldsymbol{y}^{(2)})\right)$ and then $\boldsymbol{\theta}_{3:4} = -S^{(2)}\boldsymbol{\theta}_{1:2} + \boldsymbol{T}_{\boldsymbol{\theta}}(\boldsymbol{y}^{(2)})$. Note that $M^{-1} = \frac{1}{|M|}M^T$ and $|M| = M_{1,1}^2 + M_{2,2}^2$. The total count of basic algebraic operations is listed in Table 2.1. The parameter vector can be estimated performing 12 multiplications and 11 additions. The *time unit* column shows at which stage the operation can be performed (assuming that the operations for each single task can be executed in parallel). The overall estimation will require just 6 clock cycles (on modern CPUs capable of streamed operations). The Matlab™function that computes the parameter vector (`estimate_RST.m`) is listed in Appendix D.2.3. Note that the point coordinates are normalized to improve the accuracy of the estimate. For more details refer to Section C.2.

Table 2.1: Counting operations for the RST parameter estimation using the MSS

Task	Multiplications	Additions	Time Unit
M	0	2	1
$\lvert M\rvert$	2	1	2
M^{-1}	2	0	3
$\boldsymbol{T}_{\boldsymbol{\theta}}(\boldsymbol{y}^{(1)}) - \boldsymbol{T}_{\boldsymbol{\theta}}(\boldsymbol{y}^{(2)})$	0	2	1
$\boldsymbol{\theta}_{1:2}$	4	2	4
$-S^{(2)}\boldsymbol{\theta}_{1:2}$	4	2	5
$\boldsymbol{\theta}_{3:4}$	0	2	6
$\boldsymbol{\theta}$	12	11	

2.2.4 Estimating Homographies

In this section we will study in more detail the design of a RANSAC algorithm to estimate the parameters of an homography given a set of point correspondences (of course, possibly contaminated by outliers). Before continuing we recall that an homography is a linear transformation in the projective space that relates two views of a planar scene obtained via an ideal pin-hole camera. The functional form of the homographic transformation (in Euclidean coordinates) is given by the *nonlinear* relation:

$$\boldsymbol{T}_{\boldsymbol{\theta}}(\boldsymbol{y}) = \begin{bmatrix} \frac{\theta_1 y_1 + \theta_4 y_2 + \theta_7}{\theta_3 y_1 + \theta_6 y_2 + \theta_9} \\ \frac{\theta_2 y_1 + \theta_5 y_2 + \theta_8}{\theta_3 y_1 + \theta_6 y_2 + \theta_9} \end{bmatrix}$$

We also recall that an homography is parameterized by nine numbers but it only has *eight* degrees of freedom, since scaling in the projective space is immaterial.

Parameter Estimation

The parameters of the homography can be estimated via the normalized DLT algorithm, which is discussed in some more detail in Appendix C. The Matlab™function that implements the parameter estimation routine is listed in Appendix D.2.4.

The Sideness Constraint

As we saw in Section 1.4 the complexity of RANSAC depends also on the effort needed to estimate the parameters starting from the MSS (see Equation(1.12)). It should be noticed that certain configurations of the points that compose the MSS are not "physically" meaningful even though they would yield mathematically acceptable estimates of the parameters. Sometimes detecting this pathological configurations is computationally cheap if compared to the effort of estimating the parameters from the MSS. In this section we will present a

constraint that must be satisfied in order to produce meaningful homographies[2].

Consider a MSS formed by the following 4 point correspondences:

$$MSS = \left\{ \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}'_1 \end{bmatrix}, \ldots, \begin{bmatrix} \boldsymbol{x}_4 \\ \boldsymbol{x}'_4 \end{bmatrix} \right\}$$

Consider the line passing through the points $\boldsymbol{x}_1$ and $\boldsymbol{x}_l$ such that the points $\boldsymbol{x}_h$ and $\boldsymbol{x}_k$ lie in different semi-planes (as long as there do not exist collinear triplets it is always possible to find an arrangement of the distinct indices l, h and k in $\{2, 3, 4\}$ such that the previous condition is satisfied). The fundamental observation is that the points $\boldsymbol{x}'_h$ and $\boldsymbol{x}'_k$ must lie with respect to the line passing trough $\boldsymbol{x}'_1$ and $\boldsymbol{x}'_l$ on the corresponding semi-planes containing the points $\boldsymbol{x}_h$ and $\boldsymbol{x}_k$. This also implies that the polygon formed by the properly oriented sequence of vertexes must remain convex. Figure 2.1 describes pictorially the previous condition and displays cases when it is violated. The test for the sideness constraint is quite simple. First we need to identify the line passing through the points $\boldsymbol{x}_1$ and $\boldsymbol{x}_l$ so that $\boldsymbol{x}_h$ and $\boldsymbol{x}_k$ belong to two different semi-planes. Then we compute a vector orthogonal to such line. A vector that will serve to this purpose is given by $\boldsymbol{n}_{1,l} = \begin{bmatrix} x_{l,2} - x_{1,2} & x_{1,2} - x_{l,2} \end{bmatrix}^T$. To check on which semi-plane the points $\boldsymbol{x}_h$ and $\boldsymbol{x}_k$ lie we just need to evaluate the sign of the expressions $\boldsymbol{n}_{1,l}^T(\boldsymbol{x}_h - \boldsymbol{x}_1)$ and $\boldsymbol{n}_{1,l}^T(\boldsymbol{x}_k - \boldsymbol{x}_1)$ and the analogous expressions for the corresponding points. *If the signs remain consistent then the MSS is acceptable.* The function that implements the sideness validation is listed below. Note that in the worst case scenario 32 summations and 16 multiplications are needed. In general the cost of these operations is less than the cost of estimating the homography via the nDLT algorithm. Again, we want to emphasize that this approach can save a lot of computations if it possible to develop a validation test which is (much) faster than the estimation of the parameters utilizing the elements in the MSS. The code that validates the MSS based on the sidenes constraint can be found in Appendix D.2.4.

[2]Note that the only requirement for an homography is to be non-singular. Here we refer to the fact that an homography should represent a valid transformation between images.

Error Estimation

The correspondence error between the i^{th} pair of points related by an homography can be expressed using the *symmetric transfer error*:

$$e_i^2 \stackrel{\text{def}}{=} \|\boldsymbol{x}_i' - \boldsymbol{T}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\|^2 + \|\boldsymbol{x}_i - \boldsymbol{T}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{x}_i')\|^2 \tag{2.9}$$

Assuming that the errors $\boldsymbol{\eta} = \|\boldsymbol{x}_i - \boldsymbol{T}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{x}_i')\|$ and $\boldsymbol{\eta}' = \|\boldsymbol{x}_i' - \boldsymbol{T}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\|$ can be modeled as Gaussian random vectors whose components are independent and identically distributed $\mathcal{N}(\mathbf{0}, \sigma_\eta I)$, then the distribution of e_i^2 is a Chi-square distribution with 4 degrees of freedom (each error contributing for two degrees of freedom). Therefore, from (1.3), we have that $\delta = \sigma_\eta \sqrt{F_{\chi_4^2}^{-1}(P_{inlier})}$. Figure 1.1(b) displays the error threshold for different values of P_{inlier}. As expected from the previous discussion, the larger is the probability to find all the inliers, the larger is the error threshold. The function that computes the fitting error of the data is listed in Appendix D.2.4:

Maximum Number of Iterations

As we saw in equation (1.8), the total number of iterations is essentially proportional to the ratio of the number of inliers over the number of outliers, once the cardinality of the minimum sample set is fixed. For the nDLT algorithm, the caridnality of the MSS is 4: i.e. we need at least four good point correspondences to estimate the homography. Figure **??** displays the number of iterations for differnt value of the false alarm rate ε and different values of the ratio $\frac{N_I}{N}$. To Be Finished: `Add figure`.

2.3 Frequently Asked Questions

2.3.1 What is the "right" value of σ?

There is no general answer for this question. The value of σ is used to derive the threshold that discriminates between inliers and outliers if the noise affecting the inliers is Gaussian. Thus the "right" value of σ depends on the nature of the data you are dealing with. Sometimes you may want to set directly the noise threshold: to this purpose you may want to use the option `T_noise_squared`.

2.3.2 How do I use the toolbox for image registration purposes?

The toolbox and more specifically the homography estimation routines can be used for registration purposes in order to refine a set of image correspondences. The toolbox itself *does not* provide the routines to detect/describe the features and to establish preliminary matches. This can be done using two packages which can be freely downloaded and which are based on the SIFT framework developed by Dr. Lowe. The original implementation can be found at http://www.cs.ubc.ca/~lowe/keypoints/. An excellent implementation with source code developed by Dr. Vedaldi can be obtained from http://vision.ucla.edu/~vedaldi/code/sift/sift.html.

2.3.3 Why the behaviour of RANSAC is not repeatable?

Because of the intrinsic nature of the algorithm itself. RANSAC draws the elements composing the MSS randomly (with or without the a bias) from the entire dataset. Therefore at each run the behaviour might change. To obtain repeatable performance you should fix the seed of the internal random number generator. To accomplish this task set the option `fix_seed` to `true`.

2.3.4 What should I do if I find a bug in the toolbox?

Please contact me at `marco.zuliani@gmail.com` sending me an email with a brief description of the bug, the options that you are using and the data that generates the bug. I'll try to get back to you as soon as possible.

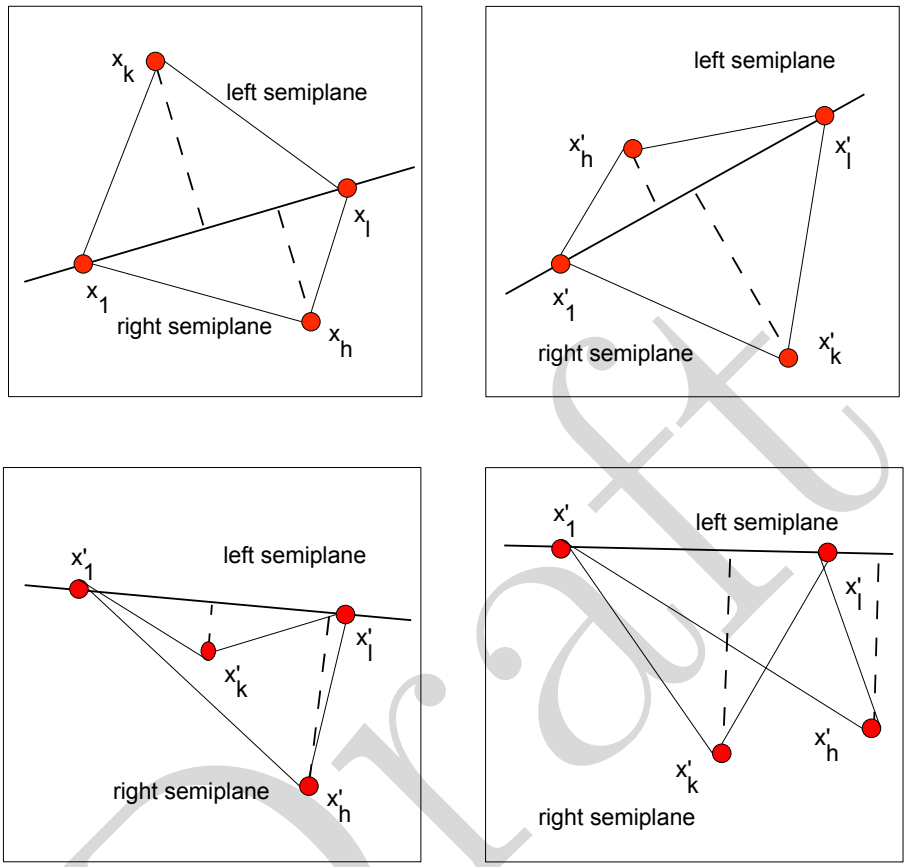


Figure 2.1: The top left figure displays the arrangements of the points composing the MSS in the first image and the line used to compute the sideness constraint. The remaining figures show possible configurations of the corresponding points where the sidness constraint is violated.

Appendix A

Notation

- Bold letters indicate vectors, e.g. $\boldsymbol{x} \in \mathbb{R}^n$. Unless otherwise stated, vector components are indicated with the same non bold letter indexed starting from 1, i.e. $\boldsymbol{x} = \begin{bmatrix} x_1 & \dots & x_n \end{bmatrix}^T$. By default vectors are column vectors.
- The hat over a variable indicates the estimate of such variable, e.g. $\hat{\boldsymbol{\theta}}$ is the estimate of the vector $\boldsymbol{\theta}$.
- To indicate portions of matrices and/or vectors Matlab™is used. For example $A_{1:p,1:q}$ indicates the sub-matrix obtained retaining the firs p rows and the first q columns of A.

Appendix B

Some Linear Algebra Facts

The results contained in this section are presented in a very accessible way in [20, 14] and analyzed in greater details in [6, 9]. Good references for optimization are [12, 1] and [11, 2] which can be downloaded respectively from http://www.siam.org/books/kelley/fr18/index.php and http://www.stanford.edu/~boyd/cvxbook/. Here we will provide a list of some basic results that have been tailored for the the applications described in the previous chapters.

B.1 The Singular Value Decomposition

Theorem B.1.1 (SVD Decomposition). *Let $A \in \mathbb{R}^{m\times n}$ be a matrix of rank r. Then there exist matrices $U \in \mathbb{R}^{m\times m}$, $V \in \mathbb{R}^{n\times n}$ and $\Sigma_1 \in \mathbb{R}^{r\times r}$ such that:*

- *V is a* unitary *matrix whose columns form a complete orthonormal basis of eigenvectors of A^TA. V can be partitioned as $\begin{bmatrix} V_1 & V_2 \end{bmatrix}$, where $V_1 \in \mathbb{R}^{n\times r}$, so that:*
  - *$\mathcal{R}(V_1) = \mathcal{R}(A^T)$ and the columns of V_1 form an orthonormal basis of $\mathcal{R}(A^T)$*
  - *$\mathcal{R}(V_2) = \mathcal{N}(A)$ and the columns of V_2 form an orthonormal basis of $\mathcal{N}(A)$*
- *U is a* unitary *matrix whose columns form a complete orthonormal basis of eigenvectors of AA^T. U can be partitioned as $\begin{bmatrix} U_1 & U_2 \end{bmatrix}$, where $U_1 \in \mathbb{R}^{m\times r}$, so that:*

  - $\mathcal{R}(U_1) = \mathcal{R}(A)$ *and the columns of* U_1 *form an orthonormal basis of* $\mathcal{R}(A)$
  - $\mathcal{R}(U_2) = \mathcal{N}(A^T)$ *and the columns of* U_2 *form an orthonormal basis of* $\mathcal{N}(A^T)$

- $\Sigma_r = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r) \in \mathbb{R}^{r \times r}$ *such that* $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$. *The scalars* σ_i *are called* singular values *of* A.

  - A *has a* dyadic expansion*:*

$$A = U_1 \Sigma_1 V_1 = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$$

  - A *has a* singular value decomposition*:*

$$A = U \Sigma V^T = U \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix} V^T$$

B.2 Relation Between the SVD Decomposition and the Eigen Decomposition

Lemma B.2.1. *Consider a matrix* $A \in \mathbb{R}^{m \times n}$ *with* $m \geq n$. *Then the squared singular values of* A *coincide with the eigenvalues of* $A^T A$ *and the right singular vectors of* A *coincide with the eigenvectors of* $A^T A$.

Proof. Consider the SVD decomposition $A = U \Sigma V^T$. Then, because of the orthogonality of U, the following chain of equations hold:

$$A^T A = \left(U \Sigma V^T\right)^T U \Sigma V^T = V \Sigma^T \Sigma V^T$$

The previous chain of equations proves the claim regarding the relation between eigenvectors and singular vectors, whereas the relation between eignevalues and singular values is established observing that $\Sigma \Sigma^T = \text{diag}\{\sigma_1^2, \ldots, \sigma_n^2\}$. □

B.3 Fast Diagonalization of Symmetric 2×2 Matrices

Consider a real symmetric matrix $A = \begin{bmatrix} a & b \\ b & c \end{bmatrix}$. We want to diagonalize A using the least number of operations, so that:

$$A = \begin{bmatrix} C & -S \\ S & C \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} C & S \\ -S & C \end{bmatrix}$$

where $C = \cos\phi$ and $S = \sin\phi$. The eigenvalues can be computed first defining:

$$\begin{aligned} \alpha &\stackrel{\text{def}}{=} a + c \\ \beta &\stackrel{\text{def}}{=} (a - c)^2 \\ \gamma &\stackrel{\text{def}}{=} 4b^2 \\ \delta &\stackrel{\text{def}}{=} \sqrt{\beta + \gamma} \end{aligned}$$

and then letting:

$$\lambda_1 = 0.5(\alpha + \delta)$$

$$\lambda_2 = 0.5(\alpha - \delta)$$

for a total number of 5 additions, 5 multiplications and the extraction of a square root. Finally the angle ϕ that defines the eigenvectors univocally (modulo a reflection about the origin) can be obtained recalling the definition of eigenvector, i.e. :

$$(A - \lambda_i I)\boldsymbol{v} = \begin{bmatrix} a - \lambda_i & b \\ b & c - \lambda_i \end{bmatrix} \begin{bmatrix} C \\ S \end{bmatrix} = 0$$

Since $(\lambda_i - a)C = bS$ then $\phi = \arctan\frac{\lambda_1 - a}{b}$. As a side note, we observe that we can easily compute the 2-norm condition number of A, which is defined as the ratio of the largest

singular value of A to the smallest:

$$\kappa(A) = \frac{\alpha + \delta}{\alpha - \delta}$$

Large condition numbers indicate a nearly singular matrix.

B.4 Least Square Problems Solved via SVD

As we saw in the examples in Section 2.2, RANSAC depends on the solution of least square problems of the form $A\boldsymbol{\theta} = \boldsymbol{b}$, where the matrix A and the vector $\boldsymbol{b}$ are obtained from noisy measurements. If $\boldsymbol{b}$ happens to be the null vector it is a common practice to constrain the norm of $\boldsymbol{\theta}$ in order to avoid trivial solutions. In the next sections we will informally discuss how both problems can be solved using the SVD decomposition described in Section B.1.

B.4.1 Solving $A\boldsymbol{\theta} = \boldsymbol{b}$

Let's assume that the matrix $A \in \mathbb{R}^{m \times n}$, with $m \geq n$, is full rank. Since the number of equations in larger than the number of unknowns the solution must be intended in a least square sense, i.e. we are seeking for the vector $\boldsymbol{\theta}^*$ such that:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta} \in \mathbb{R}^n}{\operatorname{argmin}} \|A\boldsymbol{\theta} - \boldsymbol{b}\|^2 \tag{B.2}$$

The function $\|A\boldsymbol{\theta} - \boldsymbol{b}\|^2$ is convex [2] and since $\|\boldsymbol{x}\|^2 = \boldsymbol{x}^T\boldsymbol{x}$, we need to find the point where the gradient of $\boldsymbol{\theta}^T A^T A\boldsymbol{\theta} - 2\boldsymbol{\theta}^T A^T \boldsymbol{b} + \boldsymbol{b}^T\boldsymbol{b}$ is zero. This happens when $\boldsymbol{\theta}$ satisfies the expression $A^T A\boldsymbol{\theta} = A^T\boldsymbol{b}$. Since we assumed that the matrix A is full rank, $A^T A$ is full rank too, its inverse exists and we can write:

$$\boldsymbol{\theta}^* = (A^T A)^{-1} A^T \boldsymbol{b} = A^\dagger \boldsymbol{b} \tag{B.3}$$

The matrix $A^\dagger$ is called pseudo-inverse (or Moore-Penrose inverse) of A for obvious reasons. If we plug in (B.3) the SVD decomposition of A and recall that the inverse of a unitary matrix coincides with its transpose, we obtain that:

$$\boldsymbol{\theta}^* = V\Sigma^{-1}_{1:n,1:n}U^T\boldsymbol{b}$$

In Matlab™the previous expression can be quickly computed either using the function `pinv` (i.e. `theta = pinv(A)*b;`) or the operator \ (i.e. `theta = A\b;`). When $m \geq n$ and A is full rank the solution is the same (but only in this case).

B.4.2 Solving $A\boldsymbol{\theta} = \mathbf{0}$ subject to $\|\boldsymbol{\theta}\| = 1$

Let's assume that the matrix $A \in \mathbb{R}^{m\times n}$, with $m \geq n$, has rank $n-1$ (if the rank is smaller similar considerations hold). In this case we are seeking for the vector that solves the optimization problem:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}\in\mathbb{R}^n \text{ such that } \|\boldsymbol{\theta}\|=1}{\operatorname{argmin}} \|A\boldsymbol{\theta}\|_2 \tag{B.4}$$

Since we assumed that the rank of A is $n-1$, the right null space of the matrix has at most dimension 1. Therefore the SVD of such matrix can be written as:

$$A = U\Sigma V^T = \sum_{i=1}^{n}\sigma_i\boldsymbol{u}_i\boldsymbol{v}_j^T = \begin{bmatrix} \boldsymbol{u}_1 & \dots & \boldsymbol{u}_{n-1} & \boldsymbol{u}_n \end{bmatrix} \begin{bmatrix} \sigma_1 & \dots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & \sigma_{n-1} & 0 \\ 0 & \dots & 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_1^T \\ \vdots \\ \boldsymbol{v}_{n-1}^T \\ \boldsymbol{v}_n^T \end{bmatrix}$$

where σ_i denotes the i^{th} singular value and $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ are the corresponding left and right singular vectors. We conclude that the non-trivial solution is given by $\boldsymbol{\theta}^* = \boldsymbol{v}_n$ (i.e. the basis for the right null space of A), in fact $\|A\boldsymbol{\theta}^*\|_2 = \|A\boldsymbol{v}_n\|_2 = 0$.

But what happens if the matrix A is obtained from noisy measurements? Most likely the matrix will become full rank and then we should expect $\|A\boldsymbol{\theta}^*\| > 0$. However the solution still coincides with the right singular vector corresponding to the smallest singular value (in fact now $\sigma_n > 0$). To understand why let's consider again the minimization problem (B.4). We need to minimize $\|U\Sigma V^T\boldsymbol{\theta}\|_2$ or equivalently $\|\Sigma V^T\boldsymbol{\theta}\|_2$, since orthogonal matrices do not modify the 2-norm of a vector (intuitively rotating or reflecting a vector does not modify its length). Because of this we can also write that $\|V^T\boldsymbol{\phi}\|_2 = \|\boldsymbol{\phi}\|_2$. Hence if we let $\boldsymbol{\phi} = V^T\boldsymbol{\theta}$, the minimization problem in (C.2) can be rewritten as:

$$\boldsymbol{\phi}^* = \underset{\boldsymbol{\phi}\in\mathbb{R}^n \text{ such that } \|\boldsymbol{\phi}\|=1}{\operatorname{argmin}} \|\Sigma\boldsymbol{\phi}\|_2$$

But Σ is a diagonal matrix with its entries sorted in decreasing order. Hence the solution is $\boldsymbol{\phi} = \begin{bmatrix} 0 & \dots & 0 & 1 \end{bmatrix}^T$. Since $\boldsymbol{\phi} = V^T\boldsymbol{\theta}$, this is equivalent to pick again the last column of the matrix V, i.e. the right singular vector corresponding to the smallest singular value.

Appendix C

The Normalized Direct Linear Transform (nDLT) Algorithm

C.1 Introduction

In the ideal case we would like to estimate the parameter vector $\hat{\boldsymbol{\theta}}$ such that for every point i the following homographic relation (expressed in Euclidean coordinates) holds:

$$\boldsymbol{x}'_i = \boldsymbol{T}_{\hat{\boldsymbol{\theta}}}(\boldsymbol{x}_i) = \begin{bmatrix} \frac{\theta_1 x_{1,i} + \theta_4 x_{2,i} + \theta_7}{\theta_3 x_{1,i} + \theta_6 x_{2,i} + \theta_9} \\ \frac{\theta_2 x_{1,i} + \theta_5 x_{2,i} + \theta_8}{\theta_3 x_{1,i} + \theta_6 x_{2,i} + \theta_9} \end{bmatrix}$$

Expanding the previous equation we obtain two equations that are *linear* in the parameter components:

$$\begin{aligned} (\theta_3 x_{1,i} + \theta_6 x_{2,i} + \theta_9)\, x'_{1,i} &= \theta_1 x_{1,i} + \theta_4 x_{2,i} + \theta_7 \\ (\theta_3 x_{1,i} + \theta_6 x_{2,i} + \theta_9)\, x'_{2,i} &= \theta_2 x_{1,i} + \theta_5 x_{2,i} + \theta_8 \end{aligned}$$

The previous equations can be rearranged in matrix form as:

$$\begin{bmatrix} x_{1,i} & 0 & -x_{1,i}x'_{1,i} & x_{2,i} & 0 & -x_{2,i}x'_{1,i} & 1 & 0 & -x'_{1,i} \\ 0 & x_{1,i} & -x_{1,i}x'_{2,i} & 0 & x_{2,i} & -x_{2,i}x'_{2,i} & 0 & 1 & -x'_{2,i} \end{bmatrix} \boldsymbol{\theta} = A(\boldsymbol{x}_i, \boldsymbol{x}'_i)\boldsymbol{\theta} = \mathbf{0} \tag{C.1}$$

Thus we have two equations for nine unknowns (which have just 8 degrees of freedom since the scaling for an homography is immaterial). If we stack one upon the other the matrices $A(\boldsymbol{x}_i, \boldsymbol{x}'_i)$ we obtain the over-determined (i.e. more equations than unknowns) homogeneous (i.e. the right hand side is zero) linear system:

$$\begin{bmatrix} A(\boldsymbol{x}_1, \boldsymbol{x}'_1) \\ \vdots \\ A(\boldsymbol{x}_N, \boldsymbol{x}'_N) \end{bmatrix} \boldsymbol{\theta} = A\boldsymbol{\theta} = \mathbf{0}$$

Hence one valid solution is obtained solving:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\mathbb{R}^9 \text{ such that } \|\boldsymbol{\theta}\|=1}{\operatorname{argmin}} \|A\boldsymbol{\theta}\|_2 \tag{C.2}$$

This previous minimization problem can be solved using the singular value decomposition (SVD) as explained in Section B.4.2. The algorithm just illustrated is know as *Direct Linear Transform*.

Remark C.1.1. *The cross product of a vector with itself is zero. In a noise free case we can write* $\boldsymbol{x} \times \boldsymbol{T}_{\boldsymbol{\theta}}(\boldsymbol{x}) = 0$. *Starting from this observation, we can develop an alternative version of the DLT algorithm*[1]. *The solution boils down again to the minimization of an homogeneous over-determined linear system:* $B(\boldsymbol{x}_i, \boldsymbol{x}'_i)\boldsymbol{\theta} = \mathbf{0}$. *The computation of the entries of the matrix* B *is trivial, however the details can be found in [7].*

[1]These two different formulations are selected in the function `HomographyDLT` by appropriately setting the parameter `mode`.

C.2 Point Normalization

In this paragraph we will try to intuitively assess the problem of numerical stability of the DLT algorithm. The interested reader is referred to [8] for a more thorough treatment of the subject, which requires some advanced concepts of numerical linear algebra[2]. The numerical stability of the algorithm refers to its behavior in presence of noise affecting the point position.

Quantitatively, the numerical stability of a problem (or, equivalently, of the algorithm used to solve the problem) is measured by the *condition number*. Such quantity is a measure of that problem's suitability to digital computation. A problem with a small condition number is said to be well-conditioned, while a problem with a large condition number is said to be ill-conditioned. In the case of the homography estimation, the condition number associated to the problem corresponds to the condition number of the matrix A, which can be defined as the ratio between the largest eigenvalue to the smallest one.

$$\kappa(A) = \frac{\sigma_{max}(A)}{\sigma_{min}(A)}$$

The reason behind this fact is summarized in a well known result from perturbation theory that examines how the solution of a least square problem is affected by perturbations in the coefficient matrix (and possibly in the vector of measurements). For a more general treatment of the subject refer to [6], p. 242.

The major reason for the poor condition of the matrix A is the fact that its entries may differ of several orders of magnitude. For our simple analysis it is convenient to consider the matrix $A^T A \in \mathbb{R}^{9\times 9}$. It is easy to prove that the eigenvectors of $A^T A$ coincide with the right singular vectors of A and the relation between eigenvalues and singular values is given by $\lambda_i = \sigma_i^2$. Thus we can write:

$$\kappa(A) = \sqrt{\frac{\lambda_{max}(A^T A)}{\lambda_{min}(A^T A)}}$$

[2] An excellent overview of what every computer scientist should know regarding floating-point arithmetic can be found at the following link: http://docs.sun.com/source/806-3568/ncg_goldberg.html.

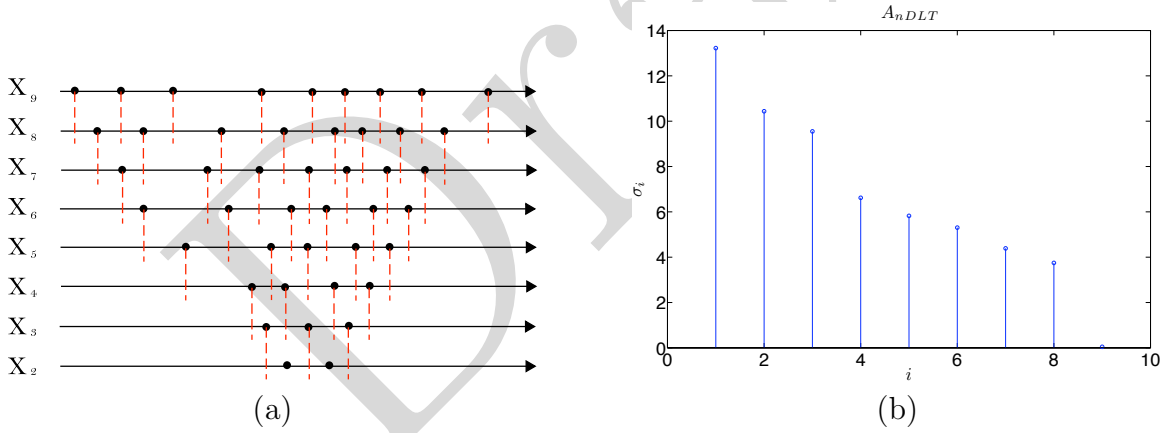
Our goal is now to obtain a rough estimate on the bounds for the condition number. Following [8], let's denote by X_r the the $r \times r$ principal submatrix (i.e. the matrix obtained retaining only the first r rows and columns) of $A^T A$. Since $X_9 = A^T A$, then:

$$\kappa(A) = \sqrt{\frac{\lambda_{max}(X_9)}{\lambda_{min}(X_9)}}$$

A very important result of matrix analysis is the *interlacing property of the eigenvalues* (see [6], p. 396). This theorem states that the when we remove a row/column from X, the eigenvalues of the original matrix X and of the reduced matrix X_r interlace, i.e. :

$$\lambda_9(X_9) \leq \lambda_8(X_8) \leq \lambda_8(X_9) \leq \ldots \leq \lambda_2(X_9) \leq \lambda_1(A_8) \leq \lambda_1(X_9)$$

If we iterate the removal process the interlacing pattern of the eigenvalues is depicted in



(a) (b)

Figure C.1: Figure (a) shows the interlacing pattern of the eigenvalues for a symmetric matrix. The black dots represent the position of the eigenvalues on the real line and the red vertical lines indicate the bounds on their location when we progressively remove one row and one column from the initial matrix. Figure (b) shows the singular values of the matrix A_{nDLT} for the example in Section C.3. Note that the dynamic range of the singular values is reasonably small and that the ninth one is much smaller than the eighth one ($\sigma_8 \approx 3.7457$ vs. $\sigma_9 \approx 0.0448$, as expected since the matrix rank is 8).

Figure C.1(a). Now let's consider the coordinates of a point in a typical digital image: they have an order of magnitude of 10^2. Thus the order of magnitude of the components of the

diagonal of the matrix $A(\boldsymbol{x}_i, \boldsymbol{x}_i')^T A(\boldsymbol{x}_i, \boldsymbol{x}_i')$ is:

$$\begin{bmatrix} 10^2 & 10^2 & 10^4 & 10^2 & 10^2 & 10^4 & 1 & 1 & 10^2 \end{bmatrix}$$

Note that the matrix $A^T A$ is obtained summing all the matrices for the different point correspondences (thus the order of magnitude of the coefficients on the diagonal of $A^T A$ does not vary significantly for the purpose of our analysis). It follows that the order of magnitude of the coefficients of the diagonal of X_2 is 10^2. Since the trace of a matrix is the sum of the eigenvalues we conclude that $\text{trace}(X_2) = \lambda_1(X_2) + \lambda_2(X_2)$ has approximately an order of magnitude of 10^2 and since the eigenvalues are always positive (the matrix $A^T A$ is positive semidefinite by construction) because of the interlacing property we can write:

$$\lambda_9(X_9) \leq \lambda_2(X_2) \leq 10^2$$

Moreover the largest eigenvalue of $A^T A$ must be not less than the largest diagonal entry of $A^T A$. This immediately follows from the interlacing properties of the singular values. We just need to notice that there exist a permutation matrix that leaves the spectrum of $A^T A$ unaltered and moves the largest diagonal entry at the beginning of the diagonal. If, with a little abuse of notation, we consider X_1 to be the 1×1 matrix containing the largest diagonal entry of $A^T A$, indeed (from Figure C.1) the largest eigenvalue of $A^T A$ (i.e. of X_9) must be not less than the largest diagonal entry of $A^T A$. The order of magnitude of the latter is 10^4 and therefore:

$$\lambda_1(X_9) \geq 10^4$$

Therefore the bound that we can set for the condition number is:

$$\kappa(A) = \sqrt{\frac{\lambda_{max}(X_9)}{\lambda_{min}(X_9)}} \geq \sqrt{\frac{10^4}{10^2}} = 10$$

which is approximately the square root of the order of magnitude of the coordinates of the

points.

To make the DLT algorithm more stable we will lower the condition number by a *normalization procedure* (also known as *preconditioning* by the numerical analysis community). The idea consist in transforming the point coordinates *before applying the DLT algorithm* (i.e. before constructing the matrix A) by centering them about the centroid $\boldsymbol{x}_c \stackrel{\text{def}}{=} \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i$ and then scaling them so that their average distance from the origin is approximately equal to a small constant (in [7] it is suggested to pick such constant equal to $\sqrt{2}$). This can be done by computing the average distance:

$$d_{av} \stackrel{\text{def}}{=} \frac{1}{N}\sum_{i=1}^{N}\|\boldsymbol{x}_i - \boldsymbol{x}_c\|$$

and finally setting $s = \frac{\sqrt{2}}{d_{av}}$ so that $\bar{\boldsymbol{x}}_i \stackrel{\text{def}}{=} s\,(\boldsymbol{x} - \boldsymbol{x}_c)$. Once the points have been normalized the DLT algorithm is applied and finally the resulting homography is denormalized (in order to go establish a correct mapping between the non normalized original points). More specifically, if $\bar{H}$ is the homography estimated using the normalized points, the denormalization procedure is implemented by the following cascade of linear transformations in the projective space:

$$H = T'^{-1}\bar{H}T = \begin{bmatrix} \frac{1}{s'} & 0 & x'_{c,1} \\ 0 & \frac{1}{s'} & x'_{c,2} \\ 0 & 0 & 1 \end{bmatrix} \bar{H} \begin{bmatrix} s & 0 & -sx_{c,1} \\ 0 & s & -sx_{c,2} \\ 0 & 0 & 1 \end{bmatrix} \tag{C.3}$$

Once again note that the normalization procedure is meant to reduce the dynamic range of the entries of the matrix A and consequently, as we discussed before, to improve the conditioning of such matrix. The DLT algorithm plus the normalization procedure is known as *normalizaed DLT algorithm*.

C.3 A Numerical Example

We will illustrate what has been discussed in the previous sections using a numerical example. Suppose that the ground truth homography that describes the mapping between a set of points is given by:

$$H = \begin{bmatrix} 0.57458 & -0.00125 & -3.16022 \\ 0.09153 & 0.65266 & -2.48508 \\ -0.00014 & -0.00025 & 0.52426 \end{bmatrix}$$

We now generate a set of $N = 32$ points uniformly randomly distributed in the interval $[-500, 500] \times [-500, 500]$. The point correspondences are formed by mapping the first set via the homography H and then by perturbing the mapped set with Gaussian noise with zero mean and standard deviation $\sigma_\eta = 3$. We now estimate the homographies using first the DLT algorithm:

$$H_{DLT} = \begin{bmatrix} 0.09147 & -0.00036 & -0.55467 \\ 0.01463 & 0.10341 & -0.81616 \\ -0.00002 & -0.00004 & 0.08338 \end{bmatrix}$$

and then the normalized DLT algorithm, obtaining:

$$H_{nDLT} = \begin{bmatrix} -0.63883 & 0.00216 & 2.78957 \\ -0.10245 & -0.72486 & 3.53780 \\ 0.00016 & 0.00028 & -0.58282 \end{bmatrix}$$

The condition number of the matrices A is respectively $\kappa(A_{DLT}) \approx 412940$ and $\kappa(A_{nDLT}) \approx 295$. As expected the condition number after the normalization is much smaller. The singular values of A_{nDLT} are shown in Figure C.1(b). Note that the dynamic range of the singular values is reasonably small and that the ninth one is much smaller than the eighth one ($\sigma_8 \approx 3.7457$ vs. $\sigma_9 \approx 0.0448$, as expected since the matrix rank is 8). The better performance obtained via the normalization is also confirmed by the value of the symmetric transfer error, which is $e_{DLT} \approx 1749$ in the first case and $e_{nDLT} \approx 1098$ in the second. The forward and

backward mapping is illustrated in Figure C.2.

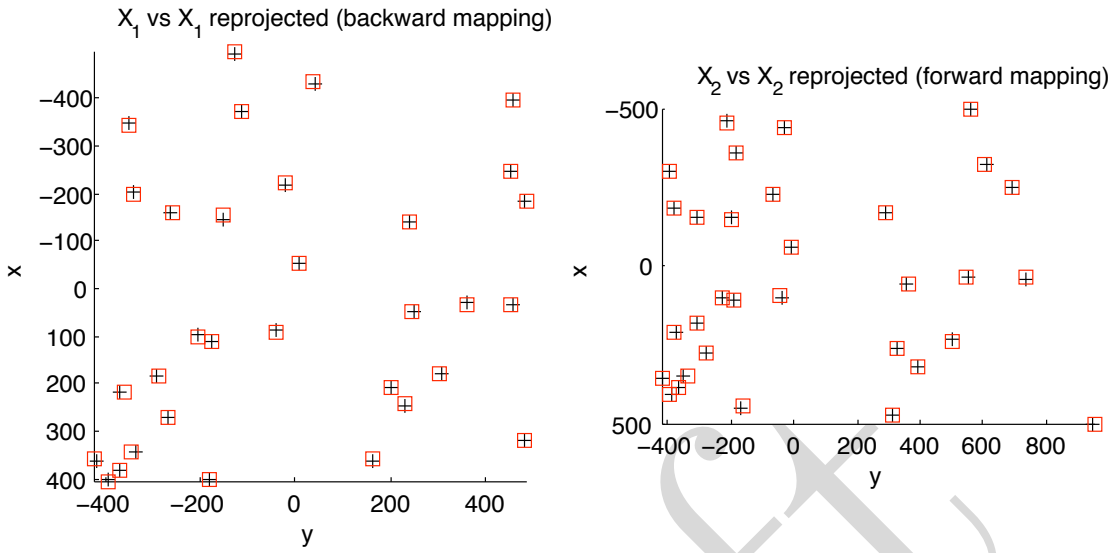


Figure C.2: The figure displays the forward and backward mapping via the homography H_{nDLT}. The crosses represent the original points whereas the squares represent the mapped points.

As a concluding remark, we want to emphasize that the normalization procedure *does not always have such a big impact* in the accuracy of the estimation of the homography, but for sure it will never cause any harm, so it is very important to normalize the points in any case (given also the fact that the additional computational complexity is not too relevant).

C.4 Concluding Remarks About the Normalized DLT Algorithm

The (normalized) DLT algorithm minimizes an algebraic quantity (C.2) that does not have an immediate correspondence with the geometry of the problem. It can be shown that this happens when the homography represents a 2D affine transformation. In the general case a common practice is to minimize the symmetric transfer error (2.9) utilizing iterative descent techniques once an initial estimate of the homography is obtained using the normalized

DLT algorithm. In general this approach provides an increased accuracy (especially if the assumption regarding the contamination of Gaussian noise for the point coordinates holds true) at the expense of an increased computational complexity, of the necessity of having an initial estimate of the optimum and of the selection of a stopping criterion that is not always trivial.

A desirable property of the algorithm used to estimate the parameters of the homographic transformation is *to be invariant* with respect to certain class of transformations of the point correspondences. More specifically consider two linear transformations in the projective space, such that:

$$\begin{aligned} \bar{\boldsymbol{p}} &= T\boldsymbol{p} \\ \bar{\boldsymbol{p}}' &= T'\boldsymbol{p}' \end{aligned}$$

and suppose that the points $\boldsymbol{p}$ and $\boldsymbol{p}'$ are related via an homography: $\boldsymbol{p}' \sim H\boldsymbol{p}$. It follows immediately that the transformed points are related via a new homography $\bar{H}$:

$$\bar{\boldsymbol{p}}' \sim \underbrace{T'^{-1}HT^{-1}}_{\bar{H}}\bar{\boldsymbol{p}}$$

We are now interested in answering the following question: if we estimate both the homography H and the homography $\bar{H}$ (for the transformed points) using the DLT algorithm, will we obtain the same result? Unfortunately the answer is no. To show why this is not true is a little bit involved: more details can be found in [7], p. 105. However the good news is that the minimizer of the symmetric transfer error is *invariant under similarity transformations*. To minimize the symmetric transfer error usually people resort to some iterative descent algorithm. The interest reader is once again referred to [7] for more information. Here we suggest the reader to visit the site of Dr. Lourakis for a fast implementation (which can also be called from Matlab™) of the non linear refinement procedure based on the Levenberg-Marquardt algorithm.

A Matlab™function that implements the normalized DLT algorithm is listed below:

```
function [H A] = HomographyDLT(X1, X2, mode, normalization)

% [H A] = HomographyDLT(X1, X2, mode)
%
% DESC:
% computes the homography between the point pairs X1, X2
%
% AUTHOR
% Marco Zuliani - zuliani@ece.ucsb.edu
%
% VERSION:
% 1.0.1
%
% INPUT:
% X1, X2         = point matches (cartesian coordinates)
% mode           = 0 -> Hartley Zisserman formulation
%                  1 -> Zuliani formulation (default)
% normalization = true (default) or false to enable/disable point
%                  normalzation
%
% OUTPUT:
% H              = homography
% A              = homogenous linear system matrix

% HISTORY
% 1.0.0          ??/??/04 - intial version
% 1.0.1          07/22/04 - small improvements

if (nargin < 3)
    mode = 'MZ';
```

```
31 end;
32
33 if (nargin < 4)
34     normalization = true;
35 end;
36
37 N = size(X1, 2);
38
39 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
40 % checks
41 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
42 if (size(X2, 2) ~= N)
43     error('HomographyDLT:inputError', ...
44         'The set of input points should have the same cardinality')
45 end;
46 if N < 4
47     error('HomographyDLT:inputError', ...
48         'At least 4 point correspondences are needed')
49 end;
50
51 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
52 % normalize the input
53 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
54 if normalization
55     % fprintf('\nNormalizing...')
56     [X1, T1] = normalize_points(X1);
57     [X2, T2] = normalize_points(X2);
58 end;
59
60 % compute h
61 switch mode
62     case 'HZ'
63     A = get_A_HZ(X1, X2);
```

```
64      case 'MZ'
65      A = get_A_MZ(X1, X2);
66  end;
67  try
68  [U S V] = svd(A);
69  catch
70      keyboard
71  end;
72  h = V(:, 9);
73
74  % reshape the output
75  switch mode
76      case 'HZ'
77      H = [h(1:3)'; h(4:6)'; h(7:9)'];
78      case 'MZ'
79      H = reshape(h, 3, 3);
80  end;
81
82  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
83  % de-normalize the parameters
84  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
85  if normalization
86      H = T2\H*T1;
87  end;
88
89  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
90  % re-normalize the homography
91  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
92  H = H/norm(H(:));
93
94  return
95
96  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
97  % Matrix construction routine
98  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
99
100 % Hartley Zisserman formulation
101 function A = get_A_HZ(X1, X2)
102
103 X1 = cart2homo(X1);
104 X2 = cart2homo(X2);
105
106 N = size(X1, 2);
107
108 A = zeros(2*N, 9);
109 zero = [0; 0; 0];
110
111 row = 1;
112 for h = 1:N
113
114     a = X2(3,h)*X1(:,h)';
115     b = X2(2,h)*X1(:,h)';
116     c = X2(1,h)*X1(:,h)';
117     A(row, :) = [zero' -a b];
118     A(row+1, :) = [a zero' -c];
119
120     row = row + 2;
121
122 end;
123
124 % Zuliani's formulation
125 function A = get_A_MZ(X1, X2)
126
127 N = size(X1, 2);
128
129 A = zeros(2*N, 9);
```

```
130
131 row = 1;
132 for h = 1:N
133
134     A(row, :)   = [...
135         X1(1,h) 0 -X1(1,h)*X2(1,h) ...
136         X1(2,h) 0 -X1(2,h)*X2(1,h) ...
137         1 0 -X2(1,h) ...
138         ];
139     A(row+1, :) = [...
140         0 X1(1,h) -X1(1,h)*X2(2,h) ...
141         0 X1(2,h) -X1(2,h)*X2(2,h) ...
142         0 1 -X2(2,h) ...
143         ];
144
145     row = row + 2;
146
147 end;
148
149 return
```

Appendix D

Some RANSAC Toolbox Source Code

D.1 Function Templates

D.1.1 MSS Validation

```
1  function flag = validateMSS_foo(X, s)
2
3  % flag = validateMSS_foo(X, s)
4  %
5  % DESC:
6  % Validates the MSS obtained via the sampling of the data before computing
7  % the parameter vector Theta
8  %
9  % INPUT:
10 % X                 = samples on the manifold
11 % s                 = indices of the MSS
12 %
13 % OUTPUT:
14 % flag              = true if the MSS is valid
15
```

```
16 % perform here the check on the MSS
17
18 return;
```

D.1.2 Parameter Estimation

```
1  function [Theta, k] = estimate_foo(X, s)
2
3  % [Theta k] = estimate_foo(X, s)
4  %
5  % DESC:
6  % Template for the estimation function to be used inside RANSAC
7  %
8  % INPUT:
9  % X                 = 2D point correspondences
10 % s                 = indices of the points used to estimate the
11 %                     parameter vector. If empty all the points
12 %                     are used
13 %
14 % OUTPUT:
15 % Theta             = estimated parameter vector Theta = H(:). If
16 %                     the estimation is not successful return an
17 %                     empty vector. i.e. Theta = [];
18 % k                 = dimension of the minimal subset
19
20 % here we define the size of the MSS
21 k = ;
22
23 % check if the input parameters are valid
24 if (nargin == 0) || isempty(X)
25     Theta = [];
26     return;
27 end;
28
29 % select the points to estimate the parameter vector
30 if (nargin == 2) && ¬isempty(s)
```

```
31      X = X(:, s);
32  end;
33
34  % check if we have enough points
35  N = size(X, 2);
36  if (N < k)
37      error('estimate_foo:inputError', ...
38          'At least k point correspondences are required');
39  end;
40
41  % call the estimation function foo
42  Theta = foo(X);
43
44  % here you may want to check if the results are meaningful.
45  % If not return an empty vector
46
47  return;
```

D.1.3 Parameter Validation

```
function flag = validateTheta_foo(X, Theta, s)

% flag = validateMSS_foo(X, Theta, s)
%
% DESC:
% Validates the parameter vector
%
% INPUT:
% X                 = samples on the manifold
% Theta             = parameter vector
% s                 = indices of the MSS
%
% OUTPUT:
% flag              = true if the MSS is valid

% perform here the check on the parameter vector Theta


return
```

D.1.4 Fitting Error

```
function [E T_noise] = error_foo(Theta, X, sigma, P_inlier)

% [E T_noise] = error_foo(Theta, X, sigma, P_inlier)
%
% DESC:
% Template to estimate the error due to the foo constraint. To
% return only the error threshold the function call should be:
%
% [dummy T_noise] = error_foo([], [], sigma, P_inlier);
%
% INPUT:
% Theta             = foo parameter vector
% X                 = samples on the manifold
% sigma             = noise std
% P_inlier          = Chi squared probability threshold for inliers
%                     If 0 then use directly sigma.
%
% OUTPUT:
% E                 = squared error
% T_noise           = noise threshold

% compute the error obtained by the orthogonal projection of
% the data points X onto the model manifold instantiated with the
% parameters Theta
E = [];
if ¬isempty(Theta) && ¬isempty(X)

    % error computation

end;
```

```
31
32 % compute the error threshold
33 if (nargout > 1)
34
35     if (P_inlier == 0)
36     % in this case the parameter sigma coincides with the noise
37     % threshold
38         T_noise = sigma;
39     else
40     % otherwise we compute the error threshold given the standard
41     % deviation of the noise assuming that the errors are normally
42     % distributed. Hence the sum of their squares is Chi2
43     % distributed with d degrees of freedom
44         d = ;
45
46         % compute the inverse probability
47         T_noise = sigma^2 * chi2inv_LUT(P_inlier, d);
48
49     end;
50
51 end;
52
53 return;
```

D.2 Source Code for the Examples

D.2.1 Line Estimation

Parameter Estimation

```
1  function [Theta, k] = estimate_line(X, s)
2
3  % [Theta k] = estimate_line(X)
4  %
5  % DESC:
6  % estimate the parameters of a 2D line given the pairs [x, y]^T
7  % Theta = [a; b; c] where a*x+b*yc = 0
8  %
9  % AUTHOR
10 % Marco Zuliani - zuliani@ece.ucsb.edu
11 %
12 % VERSION:
13 % 1.0.0
14 %
15 % INPUT:
16 % X                 = 2D points
17 % s                 = indices of the points used to estimate the parameter
18 %                     vector. If empty all the points are used
19 %
20 % OUTPUT:
21 % Theta             = estimated parameter vector Theta = [a; b; c]
22 % k                 = dimension of the minimal subset
23
24 % HISTORY:
25 % 1.0.0             = 01/26/08 - initial version
26
```

```
27 % cardinality of the MSS
28 k = 2;
29
30 if (nargin == 0) || isempty(X)
31     Theta = [];
32     return;
33 end;
34
35 if (nargin == 2) && ¬isempty(s)
36     X = X(:, s);
37 end;
38
39 % check if we have enough points
40 N = size(X, 2);
41 if (N < k)
42     error('estimate_line:inputError', ...
43         'At least 2 points are required');
44 end;
45
46 % normalize the points
47 mu = mean(X, 2);
48 Xn = X - repmat(mu, 1, N);
49
50 ATA1 = dot(Xn(1, :), Xn(1, :));
51 ATA2 = dot(Xn(1, :), Xn(2, :));
52 ATA4 = dot(Xn(2, :), Xn(2, :));
53
54
55 % sacrifice accuracy for speed
56 alpha   = ATA1+ATA4;
57 temp    = ATA1-ATA4;
58 beta    = temp*temp;
59 gamma   = 4*ATA2*ATA2;
```

```
60 Δ    = sqrt(beta + gamma);
61 lambda  = 0.5*(alpha-Δ);
62 phi     = atan2(ATA2, lambda-ATA1);
63
64 Theta(1) = sin(phi);
65 Theta(2) = cos(phi);
66 Theta(3) = Theta(1)*mu(1) + Theta(2)*mu(2);
67
68 return;
```

Error Estimation

```
1  function [E T_noise] = error_line(Theta, X, sigma, P_inlier)
2
3  % [E T_noise] = error_line(Theta, X, sigma, P_inlier)
4  %
5  % DESC:
6  % estimate the squared fitting error for a line expresed in cartesian form
7  % ax + by + c =0
8  %
9  % AUTHOR
10 % Marco Zuliani - zuliani@ece.ucsb.edu
11 %
12 % VERSION:
13 % 1.0.0
14 %
15 % INPUT:
16 % Theta             = line parameter vector
17 % X                 = samples on the manifold
18 % sigma             = noise std
19 % P_inlier          = Chi squared probability threshold for inliers
20 %                     If 0 then use directly sigma.
21 %
22 % OUTPUT:
23 % E                 = squared symmetric reprojection error
24 % T_noise           = noise threshold
25
26 % HISTORY
27 %
28 % 1.0.0             - 01/26/08 initial version
29
30 % compute the squared error
```

```
31 E = [];
32 if ¬isempty(Theta) && ¬isempty(X)
33
34     den = Theta(1)^2 + Theta(2)^2;
35
36     E = ( Theta(1)*X(1,:) + Theta(2)*X(2,:) + Theta(3) ).^2 / den;
37
38 end;
39
40 % compute the error threshold
41 if (nargout > 1)
42
43     if (P_inlier == 0)
44         T_noise = sigma;
45     else
46         % Assumes the errors are normally distributed. Hence the sum of
47         % their squares is Chi distributed (with 2 DOF since we are
48         % computing the distance of a 2D point to a line)
49
50         % compute the inverse probability
51         T_noise = sigma^2 * chi2inv_LUT(P_inlier, 2);
52
53     end;
54
55 end;
56
57 return;
```

D.2.2 Plane Estimation

Parameter Estimation

```
1  function [Theta, k] = estimate_plane(X, s)
2
3  % [Theta k] = estimate_plane(X)
4  %
5  % DESC:
6  % estimate the parameters of a 3D plane given the pairs [x, y, z]^T
7  % Theta = [a; b; c; d] where:
8  %
9  % a*x1+b*y1+c*z1+d = 0
10 % a*x2+b*y2+c*z2+d = 0
11 % a*x3+b*y3+c*z3+d = 0
12 %
13 % AUTHOR
14 % Marco Zuliani - zuliani@ece.ucsb.edu
15 %
16 % VERSION:
17 % 1.0.0
18 %
19 % INPUT:
20 % X                 = 3D points
21 % s                 = indices of the points used to estimate the parameter
22 %                     vector. If empty all the points are used
23 %
24 % OUTPUT:
25 % Theta             = estimated parameter vector Theta = [a; b; c; d]
26 % k                 = dimension of the minimal subset
27
28 % HISTORY:
29 % 1.0.0             = 07/05/08 - initial version
30
31 % cardinality of the MSS
32 k = 3;
```

```
33
34 if (nargin == 0) || isempty(X)
35     Theta = [];
36     return;
37 end;
38
39 if (nargin == 2) && ¬isempty(s)
40     X = X(:, s);
41 end;
42
43 % check if we have enough points
44 N = size(X, 2);
45 if (N < k)
46     error('estimate_plane:inputError', ...
47         'At least 3 points are required');
48 end;
49
50 A = [transpose(X(1, :)) transpose(X(2, :)) transpose(X(3, :)) ones(N, 1)];
51 [U S V] = svd(A);
52 Theta = V(:, 4);
53
54 return;
```

Error Estimation

```
1  function [E T_noise] = error_plane(Theta, X, sigma, P_inlier)
2
3  % [E T_noise] = error_plane(Theta, X, sigma, P_inlier)
4  %
5  % DESC:
6  % estimate the squared fitting error for a plane expresed in cartesian form
7  %
8  % a*x1+b*y1+c*z1+d = 0
9  % a*x2+b*y2+c*z2+d = 0
10 % a*x3+b*y3+c*z3+d = 0
11 %
12 % AUTHOR
13 % Marco Zuliani - zuliani@ece.ucsb.edu
14 %
15 % VERSION:
16 % 1.0.0
17 %
18 % INPUT:
19 % Theta              = plane parameter vector
20 % X                  = samples on the manifold
21 % sigma              = noise std
22 % P_inlier           = Chi squared probability threshold for inliers
23 %                      If 0 then use directly sigma.
24 %
25 % OUTPUT:
26 % E                  = squared symmetric reprojection error
27 % T_noise            = noise threshold
28
29 % HISTORY
30 %
```

```
31 % 1.0.0             – 07/05/08 initial version
32
33 % compute the squared error
34 E = [];
35 if ¬isempty(Theta) && ¬isempty(X)
36
37     den = Theta(1)^2 + Theta(2)^2 + Theta(3)^2;
38
39     E = ( ...
40         Theta(1)*X(1,:) + ...
41         Theta(2)*X(2,:) + ...
42         Theta(3)*X(3,:) + ...
43         Theta(4)...
44         ).^2 / den;
45
46 end;
47
48 % compute the error threshold
49 if (nargout > 1)
50
51     if (P_inlier == 0)
52         T_noise = sigma;
53     else
54         % Assumes the errors are normally distributed. Hence the sum of
55         % their squares is Chi distributed (with 3 DOF since we are
56         % computing the distance of a 3D point to a plane)
57
58         % compute the inverse probability
59         T_noise = sigma^2 * chi2inv_LUT(P_inlier, 3);
60
61     end;
62
63 end;
```

```
64
65  return;
```

D.2.3 RST Estimation

Parameter Estimation

```
function [H s phi T] = RSTLS(X1, X2, normalization)

% [H s phi T] = RSTLS(X1, X2, normalization)
%
% DESC:
% computes the RST transformation between the point pairs X1, X2
%
% AUTHOR
% Marco Zuliani - zuliani@ece.ucsb.edu
%
% VERSION:
% 1.0.0
%
% INPUT:
% X1, X2         = point matches (cartesian coordinates)
% normalization = true (default) or false to enable/disable point
%                 normalzation
%
% OUTPUT:
% H              = homography representing the RST transformation
% s              = scaling
% phi            = rotation angle
% T              = translation vector

% HISTORY
% 1.0.0          08/27/08 - intial version

if (nargin < 3)
```

```
29      normalization = true;
30  end;
31
32  N = size(X1, 2);
33
34  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
35  % checks
36  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
37  if (size(X2, 2) ≠ N)
38      error('RSTLS:inputError', ...
39          'The set of input points should have the same cardinality')
40  end;
41  if N < 2
42      error('RSTLS:inputError', ...
43          'At least 2 point correspondences are needed')
44  end;
45
46  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
47  % normalize the input
48  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
49  if normalization
50      % fprintf('\nNormalizing...')
51      [X1, T1] = normalize_points(X1);
52      [X2, T2] = normalize_points(X2);
53  end;
54
55  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
56  % estimation
57  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
58  if (N == 2)
59
60      % fast estimation
61      Theta = zeros(4,1);
```

```
62
63      % MM = M_{:,1} = y^(1) - y^(2) = [ y_1^(1) - y_1^(2) ; y_2^(1) - y_2^(2) ]
64      % 2 additions
65      MM = X1(:,1) - X1(:,2);
66      % detMM = |M|
67      % 1 additions, 2 multiplication
68      detMM = MM(1)*MM(1) + MM(2)*MM(2);
69      % MMi = [ [M^-1]_{1,1} ; -[M^-1]_{2,1} ]
70      % 2 multiplications
71      MMi = MM / detMM;
72
73      % Delta = T_theta(y^(1)) - T_theta(y^(2))
74      % 2 additions
75      Delta = X2(:,1) - X2(:,2);
76
77      % Theta(1:2) = M^-1 (T_theta(y^(1)) - T_theta(y^(2)))
78      % 1 additions, 2 multiplications
79      Theta(1) = MMi(1)*Delta(1) + MMi(2)*Delta(2);
80      % 1 additions, 2 multiplications
81      Theta(2) = MMi(1)*Delta(2) - MMi(2)*Delta(1);
82      % Theta(3:4) = -S^(2) theta_{1:2} + T_theta(y^(2))
83      % 2 additions, 2 multiplications
84
85      Theta(3) = X2(1,2) - Theta(1)*X1(1,2) + Theta(2)*X1(2,2);
86      % 2 additions, 2 multiplications
87      Theta(4) = X2(2,2) - Theta(1)*X1(2,2) - Theta(2)*X1(1,2);
88
89      % total: 11 additions, 12 multiplications
90  else
91
92      A = zeros(2*N, 4);
```

```
93      b = zeros(2*N, 1);
94
95      ind = 1:2;
96      for n = 1:N
97
98          A(ind, 1:2) = [X1(1,n) -X1(2,n); X1(2,n) X1(1,n)];
99          A(ind, 3:4) = eye(2);
100
101         b(ind) = X2(1:2, n);
102
103         ind = ind + 2;
104
105     end;
106
107     % solve the linear system in a least square sense
108     Theta = A\b;
109
110 end;
111
112 % compute the corresponding homography
113 H = [Theta(1) -Theta(2) Theta(3); Theta(2) Theta(1) Theta(4); 0 0 1];
114
115 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
116 % de-normalize the parameters
117 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
118 if normalization
119     H = T2\H*T1;
120 end;
121 H = H/H(9);
122
123 % prepare the output
124 if nargout > 1
125
```

```
126       s         = sqrt(H(1,1)*H(1,1) + H(2,1)*H(2,1));
127       phi       = atan2(H(2,1), H(1,1));
128       T         = H(1:2, 3);
129
130   end;
131
132   return
```

D.2.4 Homography Estimation

Parameter Estimation

```
function [Theta, k] = estimate_homography(X, s)

% [Theta k] = estimate_homography(X, s)
%
% DESC:
% estimate the parameters of an homography using the normalized
% DLT algorithm. Note that Theta = H(:)
%
% AUTHOR
% Marco Zuliani - zuliani@ece.ucsb.edu
%
% VERSION:
% 1.0.1
%
% INPUT:
% X                 = 2D point correspondences
% s                 = indices of the points used to estimate the parameter
%                     vector. If empty all the points are used
%
% OUTPUT:
% Theta             = estimated parameter vector Theta = H(:)
% k                 = dimension of the minimal subset

% HISTORY:
% 1.0.0             = ??/??/05 - initial version
% 1.0.1             = 27/08/08 - minor improvements

% cardinality of the MSS
```

```
29 k = 4;
30
31 if (nargin == 0) || isempty(X)
32     Theta = [];
33     return;
34 end;
35
36 if (nargin == 2) && ¬isempty(s)
37     X = X(:, s);
38 end;
39
40 % check if we have enough points
41 N = size(X, 2);
42 if (N < k)
43     error('estimate_homography:inputError', ...
44         'At least 4 point correspondences are required');
45 end;
46
47 H = HomographyDLT(X(1:2, :), X(3:4, :));
48 Theta = H(:);
49
50 return;
```

MSS Validation

```
1  function flag = validateMSS_homography(X, s)
2
3  % flag = validateMSS_homography(X, s)
4  %
5  % DESC:
6  % Validates the MSS obtained sampling the data using the sideness
7  % constraintbefore computing the parameter vector Theta
8  %
9  % INPUT:
10 % X                 = samples on the manifold
11 % s                 = indices of the MSS
12 %
13 % OUTPUT:
14 % flag              = true if the MSS is valid
15
16 % HISTORY:
17 %
18 % 1.1.0             - 10/12/08 - Initial version
19
20
21 % set this to true to display invalid MSS (just for debug/didactic
22 % purposes)
23 graphic = false;
24
25 % Check if the points are in pathological configurations. Compute the
26 % covariance matrix and see if the determinant is too small (which implies
27 % the point are collinear)
28 ind = [1 2];
29 for h = 1:2
30     mu = mean(X(ind, s), 2);
```

```
31      Xzm = X(ind, s) - repmat(mu, 1, length(s));
32      C1 = Xzm(1, :)*transpose(Xzm(1, :));
33      C2 = Xzm(1, :)*transpose(Xzm(2, :));
34      C3 = Xzm(2, :)*transpose(Xzm(2, :));
35      % compute the condition number
36      alpha = C1+C3;
37      temp  = C1-C3;
38      beta  = temp*temp;
39      gamma = 4*C2*C2;
40      Δ = sqrt(beta+gamma);
41      kappa = (alpha+Δ)/(alpha-Δ);
42      if ( kappa > 1e9 )
43          flag = false;
44          return;
45      end;
46      ind = ind + 2;
47  end;
48
49  % Generate all the possible pairings for the line that determines the
50  % semi-planes. The anchor point is the first one, i.e. the one with index
51  % 1. Thus the line that defines the semiplanes can be the line passing
52  % through the points:
53  %
54  % (1,2) or
55  % (1,3) or
56  % (1,4)
57  %
58  % The remaining rows define the points that should lie in different
59  % semiplanes
60  ind = s([...
61      2 3 4; ...
62      3 2 2; ...
63      4 4 3]);
```

```
64
65 % Setting the flag to false should guard against collinearity
66 flag = false;
67 for l = 1:3
68
69     % compute the normal vector n_{1,l}
70     % 2 summations
71     n(1) = X(2, ind(1,l))-X(2, s(1));
72     n(2) = X(1, s(1))-X(1, ind(1,l));
73
74     % get the projection of the other two points
75     % 6 summations, 4 multiplications
76     p1 = n(1)*( X(1,ind(2,l)) - X(1, s(1)) ) +...
77         n(2)*( X(2,ind(2,l)) - X(2, s(1)) );
78     p2 = n(1)*( X(1,ind(3,l)) - X(1, s(1)) ) +...
79         n(2)*( X(2,ind(3,l)) - X(2, s(1)) );
80
81     % if they lie on the same side then select next arrangement
82     if sign(p1) == sign(p2)
83         continue;
84     end;
85
86     % compute the normal vector n'_{1,l} for the corresponding
87     % points
88     % 2 summations
89     np(1) = X(4, ind(1,l))-X(4, s(1));
90     np(2) = X(3, s(1))-X(3, ind(1,l));
91
92     % get the projection of the other two corresponding points
93     % 6 summations, 4 multiplications
94     pp1 = np(1)*( X(3,ind(2,l)) - X(3, s(1)) ) +...
95         np(2)*( X(4,ind(2,l)) - X(4, s(1)) );
96     pp2 = np(1)*( X(3,ind(3,l)) - X(3, s(1)) ) +...
```

```
97          np(2)*( X(4,ind(3,l)) - X(4, s(1)) );
98
99      % verify the sideness
100     flag = (sign(p1) == sign(pp1)) && (sign(p2)==sign(pp2));
101
102     if (graphic) && (flag == false)
103
104         color = 'gr';
105         d = 16;
106
107         figure;
108
109         offset = 0;
110         for kk = 1:2
111             subplot(1,2,kk)
112             hold on
113             plot(X(1+offset, s), X(2+offset, s), ...
114                 'o','MarkerSize', 8, ...
115                 'MarkerEdgeColor', 'k', ...
116                 'MarkerFaceColor', color(kk))
117             % draw the line that separates the planes
118             plot([X(1+offset, s(1)) X(1+offset, ind(1, l))], ...
119                 [X(2+offset, s(1)) X(2+offset, ind(1, l))], '--k');
120
121             for hh = 1:4
122                 text(X(1+offset, s(hh))+d, ...
123                     X(2+offset, s(hh))+d, num2str(hh))
124             end;
125             xlabel('x');
126             ylabel('y');
127             axis equal
128             offset = offset + 2;
129         end;
```

```
130
131             pause
132         end;
133
134         break;
135
136     end;
137
138
139     return;
```

Error Estimation

```
1  function [E T_noise] = error_homography(Theta, X, sigma, P_inlier)
2
3  % [E T_noise] = error_homography(Theta, X, sigma, P_inlier)
4  %
5  % DESC:
6  % estimate the squared symmetric transfer error due to the homographic
7  % constraint
8  %
9  % AUTHOR
10 % Marco Zuliani - zuliani@ece.ucsb.edu
11 %
12 % VERSION:
13 % 1.0.0
14 %
15 % INPUT:
16 % Theta             = homography parameter vector
17 % X                 = samples on the manifold
18 % sigma             = noise std
19 % P_inlier          = Chi squared probability threshold for inliers
20 %                     If 0 then use directly sigma.
21 %
22 % OUTPUT:
23 % E                 = squared symmetric reprojection error
24 % T_noise           = noise threshold
25
26 % HISTORY
27 %
28 % 1.0.0             - 11/18/06 initial version
29
30 % compute the squared symmetric reprojection error
```

```
31 E = [];
32 if ¬isempty(Theta) && ¬isempty(X)
33
34     H = reshape(Theta, 3, 3);
35
36     X12 = homo2cart(H*cart2homo(X(1:2, :)));
37     X21 = homo2cart(H\cart2homo(X(3:4, :)));
38
39     E1 = sum((X(1:2, :)-X21).^2, 1);
40     E2 = sum((X(3:4, :)-X12).^2, 1);
41
42     E = E1 + E2;
43 end;
44
45 % compute the error threshold
46 if (nargout > 1)
47
48     if (P_inlier == 0)
49         T_noise = sigma;
50     else
51         % Assumes the errors are normally distributed. Hence the sum of
52         % their squares is Chi distributed (with 4 DOF since the symmetric
53         % distance contributes for two terms and the dimensionality is 2)
54
55         % compute the inverse probability
56         T_noise = sigma^2 * chi2inv_LUT(P_inlier, 4);
57
58     end;
59
60 end;
61
62 return;
```

Appendix E

GNU Free Documentation License

Version 1.2, November 2002

51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the

same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "**Document**", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "**you**". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "**Modified Version**" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "**Secondary Section**" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "**Invariant Sections**" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections.

If the Document does not identify any Invariant Sections then there are none.

The "**Cover Texts**" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "**Transparent**" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "**Opaque**".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "**Title Page**" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "**Entitled XYZ**" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "**Acknowledgements**", "**Dedications**", "**Endorsements**", or "**History**".) To "**Preserve the Title**" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or non-commercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover

Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you

must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its

Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer

review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all

sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it,

you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with ... Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Bibliography

[1] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, second edition, 1999.

[2] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, sixth edition, 2008.

[3] O. Chum and J. Matas. Randomized RANSAC with $T_{d,d}$ test. In *13th British Machine Vision Conference*, September 2002.

[4] O. Chum and J. Matas. Matching with PROSAC - progressive sample consensus. In *Proceedings of Conference on Computer Vision and Pattern Recognition*, volume 1, pages 220–226, San Diego, June 2005.

[5] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24:381–395, 1981.

[6] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The John Hopkins University Press, 1996.

[7] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2003.

[8] R. I. Hartley. In defence of the eight-point algorithm. *IEEE Transaction on Pattern Recognition and Machine Intelligence*, 16(6):580–593, 1997.

[9] R. Horn and C. R. Johnson. *Matrix Analisys*. Cambridge University Press, 1999.

[10] P. J. Huber. *Robust Statistics*. Wiley, 1981.

[11] C. T. Kelley. *Iterative Methods for Optimization*. SIAM, 1999.

[12] D. G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley, second edition, 2003.

[13] J. Mendel. *Lessons in Estimation Theory for Signal Processing, Communication and Control*. Prentice-Hall, Englewood-Cliffs, 1995.

[14] C. D. Meyer. *Matrix Analysis and Applied Linear Algebra*. SIAM, 2001.

[15] I. Nabney. *Netlab: Algorithms for Pattern Recognition*. Springer, 2003.

[16] D. Nistér. Preemptive RANSAC for live structure and motion estimation. In *IEEE International Conference on Computer Vision*, pages 199–206, Nice, France, October 2003.

[17] P. J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection*. Wiley, 1987.

[18] C. V. Stewart. MINPRAN: A new robust estimator for computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(10):925–938, October 1995.

[19] C. V. Stewart. Robust parameter estimation in computer vision. *SIAM Review*, 41(3):513–537, September 1999.

[20] G. Strang. *Linear Algebra and Its Applications*. Academic Press, 1988.

[21] R. Toldo and A. Fusiello. Robust multiple structures estimation with j-linkage. In *European Conference on Computer Vision*, pages 537–547, Marseille, France, October 2008.

[22] B. J. Tordoff and D. W. Murray. Guided-MLESAC: Faster image transform estimation by using matching priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1523–1535, October 2005.

[23] P.H.S. Torr and A. Zisserman. MLESAC: A new robust estimator with application to estimating image geometry. *Journal of Computer Vision and Image Understanding*, 78(1):138–156, 2000.

[24] A. Vedaldi, H. Jin, P. Favaro, and S. Soatto. KALMANSAC: Robust filtering by consensus. In *Proceedings of the International Conference on Computer Vision (ICCV)*, volume 1, pages 633–640, 2005.

[25] H. Wang and D. Suter. Robust adaptive-scale parametric model estimation for computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1459–1474, 2004.

[26] Z. Zhang. Parameter estimation techniques: A tutorial with application to conic fitting. *Image and Vision Computing Journal*, 25(1):59–76, 1997.

[27] M. Zuliani, C. S. Kenney, and B. S. Manjunath. The MultiRANSAC algorithm and its application to detect planar homographies. In *IEEE International Conference on Image Processing*, September 2005.