A Resampling Method for Computer Vision

Carlo Tomasi and John Zhang
Computer Science Department, Stanford University
Stanford, CA 94305
{tomasi,zhang}@cs.stanford.edu

Abstract

A resampling procedure based on Efron's bootstrap method is proposed for the robust estimation of parameters from redundant data. The procedure handles a substantial fraction of outliers, has linear complexity even for superlinear estimation problems, can be applied to any parameter estimation algorithm without modification, and is easily parallelized. The problem of estimating camera motion from instantaneous image velocities is used to illustrate the method. Simulations and results show robust and accurate results.

1 Introduction

Many problems in computer vision involve the computation of unknown quantities from a set of noisy and usually redundant measurements. This computation is called a *parameter estimation* problem in the statistics literature. For instance, the rotation and direction of translation of a camera moving in a static environment can be computed from measurements of the motion of five or more image points. The three-dimensional coordinates of the corresponding world points are a side-result of the computation as well.

In parameter estimation, the noise corrupting the measurements is assumed to have a known probability distribution. In addition, a small fraction of the data, called *outliers*, are often entirely wrong, in the sense that the errors affecting them are not modeled well by this distribution. To address outliers, so-called *robust* parameter estimation procedures have been introduced in the past, both in computer vision and in the statistics literature.

Robust methods are based on the fact that from a candidate solution one can compute the measurements that would be obtained in the absence of noise if that solution were correct. Actual measurements that are in some sense too far from these predicted measurements are considered outliers, and either deemphasized or discarded from the computation altogether. Consider for instance the case of camera motion computation mentioned above. Given camera rotation and direction of translation, as well as three-dimensional point coordinates, that is, given a candidate solution to the problem, the ideal positions and velocities of the image points can be computed. The *residuals*, that is, the discrepancies between computed and actual measurements, can then be formed, and used as a basis for injecting robustness into the solution. Intuitively, a solution is accepted if a sufficient number of measurements has a sufficiently small residual, while measurements with excessive residuals are discarded. As summarized in section 4, several methods have been proposed for implementing this idea, both in computer vision and elsewhere.

In this paper, we propose an alternative to this basic idea, based on the notion of *resampling* from the statistical literature. Rather than achieving a robust solution by monitoring the size of the residuals between actual and computed *measurements*, we propose to reason about the discrepancies between *solutions*, that is, between parameter estimates computed from distinct subsets of the data. In other words, we propose to inject robustness by working in the space of the parameters, rather than in the space of the measurements. Random subsets of the redundant measurements are used to compute different parameter estimates through a nonrobust method, and a clustering method identifies solutions that agree with one another. The final solution is a representative for the best cluster. In the camera motion example, a standard least-squares method can be used to compute sets of motion parameters from random subsets of the image measurements. Solutions from outlier-free subsets will cluster, while subsets with outliers will usually yield parameter estimates that scatter in the space of all possible solutions. Any of a number of clustering methods will then return a good solution.

One of the main advantages of this resampling method is that it can be applied essentially without modification to any nonrobust parameter estimation procedure. For instance, a standard least-squares method can be used as a basic subroutine for estimating camera motion. By itself, this subroutine is not robust. However, as we will see in more detail in section 4, it usually has a well-understood behavior, and can be analyzed and made to converge more easily than, say, a method based on a robust residual norm. Resampling, then, makes the final results robust to outliers, at the cost of repeating the nonrobust computation several times. It must be noted, however, that performing the same computation several times on subsets of the data can be comparable to, or even faster than, doing it once on all of the data. In fact, as we show in section 3, the resampling method makes the complexity of the overall estimation procedure *linear* in the number of data points, even if the original, nonrobust method has superlinear complexity. In addition, different runs of the basic subroutine can proceed from different starting points, thereby lessening the dependence of local optimization algorithms on initial guesses. Finally, the resampling method is parallelizable in a very straight-forward way.

In the following, we introduce the general idea of resampling in section 2. In section 3, we establish conditions for the method to work correctly and we examine its complexity. In section 4, we investigate the connections of the resampling method with other robust methods, as well as with the so-called "bootstrap" and "jackknife" methods in statistics. In section 5, we show simultions and experiments for the example of camera motion estimation from instantaneous flow field measurements.

2 The Resampling Method

Assume that a parametric estimation algorithm is given, which maps the m measurements in the input set $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$ to an estimate $\hat{\boldsymbol{\vartheta}}$ of the unknown, n-dimensional parameter vector $\boldsymbol{\vartheta}$:

$$\hat{\boldsymbol{\vartheta}} = s(X) \ . \tag{1}$$

The resampling procedure runs s a number r of times, on random subsets of j measurements each, and then clusters the results. More specifically, let j satisfy $0 < j \leq m$ (typically j is a constant or a small fraction of m), and let r be given (e.g., $r = 200$ or $r = m$). The resampling method has two stages.

Resampling. For each $b = 1, 2, \ldots, r$, randomly draw j elements from X, without replacement, to form the sample set $X_b = \{\boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_j}\} \subset X$, and apply the given estimation algorithm s to each X_b. This yields a set

$$\Theta = \{\hat{\boldsymbol{\vartheta}}_1, \ldots, \hat{\boldsymbol{\vartheta}}_r\}$$

of r estimates of the parameter.

Clustering. Use a clustering algorithm to find an estimate of the parameter from the r estimates in the set Θ:

$$\hat{\boldsymbol{\vartheta}} = \text{Cluster}(\Theta). \tag{2}$$

Conceptually, clustering is itself a parameter estimation problem, the data of which are however in parameter space, rather than in the original data space.

Thus, resampling essentially transfers the robust estimation problem from the original data space to the space of parameters. Note that the clustering stage only needs to identify the point with the highest density of neighbors in the solution set Θ. In many cases the median, applied componentwise, is a good clustering method, but any of a number of mode-seeking algorithms can be used here. For instance, one can also first compute a histogram to find a coarse estimation of the highest density point, and then perform the median in the appropriate bucket. Several mode-seeking methods are discussed in [27]. Even the mean within the selected cluster can be effective in many cases. For tight clusters, the difference between median, mode, or mean is often negligible.

3 Correctness and Complexity

The proposed resampling method can reduce or eliminate the effects of outliers altogether. Intuitively, for the method to work, the good solutions must have a substantial presence in the solution set Θ, so that the clustering algorithm can single out the desired solution. In this section we present some intuitive considerations concerning this point.

A standard least-squares approach would use all the m data points at once to find a solution to a given estimation problem. In the absence of outliers, this is typically only slightly better than using, say, q random subsets of j data points each and averaging all solutions, as long as qj is substantially greater than m, and j is substantially greater than the minimum number $j_{\min}$ necessary to solve the problem. In fact, $qj \gg m$ implies that with high probability all data are used in the solution, and $j \gg j_{\min}$ implies that the condition number of a size-j problem is comparable to that of the size-m problem.

To illustrate this point, consider an $m \times n$ linear system

$$A\boldsymbol{x} = \boldsymbol{b} \tag{3}$$

where the "measurement" vector $\boldsymbol{b}$ is corrupted by random Gaussian noise. Let e_m be the error in the solution,

$$e_m = \|\hat{\boldsymbol{x}} - \boldsymbol{x}\|$$

where $\hat{\boldsymbol{x}} = A^{\dagger}\boldsymbol{b}$ is the least-squares solution to the system ($A^{\dagger}$ is the pseudoinverse of A), and $\boldsymbol{x}$ is the true value of the solution. Furthermore, let e_{jq} be the error obtained by solving q subsystems of (3), each with j equations, and then averaging the q solutions together. Figure 1 shows the mean value, over 30 trials (each with a different random matrix A), of the ratio e_{jq}/e_m for $m = 200$, $n = 5$, $j = 15$, for $q = 1, \ldots, 50$, and in the case in which A is a random matrix with entries uniformly distributed between 0 and 1.

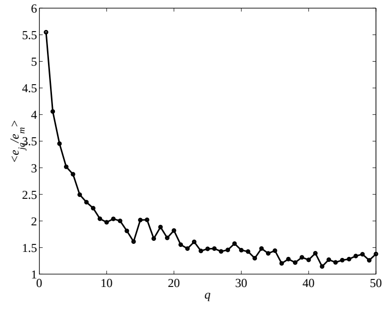


Figure 1: 30-trial average of the errors obtained by averaging the solutions to q subsystems of size 15×5, randomly drawn out of a 200×5 linear system $A\boldsymbol{x} = \boldsymbol{b}$ with a random matrix A and noisy right-hand side $\boldsymbol{b}$. The average errors are expressed as a multiple of the error obtained by solving the entire system at once.

From this figure, we see that when $qj = 15q$ becomes comparable to $m = 200$, that is, when q is of the order of ten or grater, the solution from averaging q subsystems of size $j = 15$ each is at most about twice as bad, on the average, as the one-shot solution from all the $m = 200$ data points. This result only holds for linear systems with random matrices, and it is hard to say anything quantitative about nonlinear systems with nonrandom matrices. However, the underlying intuition applies, and the linear-random case illustrates the principle.

The main consequence of this fact is that in the *presence* of outliers the resampling procedure will yield a solution whose quality is comparable to the one-shot solution in the absence of outliers provided that

- there are enough outlier-free solutions to make the clustering stage pick the correct cluster; and
- the selected cluster has a number q of solutions such that $qj \gg m$.

The first condition ensures that outliers have no effect, since they do not appear in the main cluster. The second condition implies that the cluster is tight enough that even the mean of the cluster solutions is of quality comparable to the one-shot solution that would be obtained in the absence of outliers.

Thus, we need to establish conditions for the correct cluster to be sufficiently populated. To this end, we assume that when we pick a subset of j data points, the solution from this set of data is either in the correct cluster, when all data points are good, or outside it when one or more outliers are present. In other words, we assume that outliers corrupt the solution very substantially, an assumption that is generally satisfied at least for the poorly conditioned problems that are typical of image motion analysis.

When there are n data points with k outliers, the probability for a solution to be good is the number $\binom{m-k}{j}$ of good subsets divided by the number $\binom{m}{j}$ of all possible subsets, that is,

$$p = \frac{(m-k)(m-k-1)\ldots(m-k-j+1)}{m(m-1)\ldots(m-j+1)}. \qquad (4)$$

Under very mild conditions on the number of outliers k and the subset size j, this probability is bounded away from zero as m and k are made to vary. For example, if j is fixed to be 20, and there are about 5% outliers ($k/m \approx .05$), then p is close to .35 when m is large. In the case $m = 100$ and $k = 5$, $p \approx .32, .44$, and .58 if the subset size $j = 20, 15$, and 10, respectively. Clearly p converges to 1 if $k/m \to 0$ for fixed j.

Thus, each subset yields a good solution with probability p. When we repeat this process r times, the number g of good solutions is a random variable with the binomial distribution

$$\binom{r}{g} p^g (1-p)^{r-g}.$$

The mean of g is

$$\bar{g} = rp$$

and its variance is

$$\sigma^2 = rp(1-p)\ .$$

For large r, this can be approximated by the normal distribution $N(rp, \sqrt{rp(1-p)})$ [14].

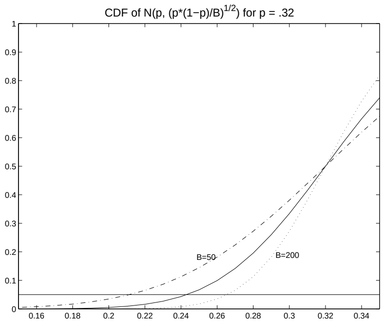


Figure 2: The cumulative distribution function of the normal approximation to g/r shows that with probability of $.95$, more than $21\%, 24\%, 26\%$ of the subsample solutions are good for $r = 50, 100, 200$ subsamples, respectively.

Figure 2 shows the graph of the cumulative distribution function of the normal approximation of the random variable g/r, with probability density $N(p, \sqrt{p(1-p)/r})$, for the case $p = .32$ and $r = 50, 100$ and 200. We see that with probability of .95, more than 21%, 24% and 26% of the subset solutions are good for a number $r = 50, 100$ and 200 of subsets, respectively.

Thus, under mild conditions, when the number r of subsets is large enough, a fixed percentage of them yield good solutions with high probability. With this probability (95% in the numerical example above), the proposed method is immune to outliers. In practice, we follow the recommendation of [13] to choose $r = 200$, or $r = \max\{200, m\}$. A validation process of selecting the solution with smallest remainder median, like that in neural networks (cf. [3, p. 373]), can be used to make sure that the solution is acceptable. Otherwise this process is repeated or r is increased. In [1], this validation process was used with $r = 1$.

Complexity

The proposed approach reduces the computational complexity in the following sense: assuming that the one-shot parametric estimation (1) from all the data has a complexity of $O(f(m))$ in the size of the input data set X, then a resampling approach with r subsets requires a complexity of $O(f(j)r)$. If r is $O(m)$, so that the number of data used is proportional to the number of data available, the complexity is $O(f(j)m)$. Since j is a constant, resampling is essentially a linear complexity algorithm, $O(m)$, even for problems whose one-shot complexity is superlinear. Clustering may add to the complexity, but not if median clustering is used, as we do in our experiments below. Note also that with the proposed method the solution from each sample can still be computed by least-squares, thereby avoiding the possible complications, such as spurious local minima or narrowed basins of attraction, that can derive from non-convex robust error functions. Finally, the resampling procedure can be very obviously parallelized, for example, by devoting each available processor to the computation of parameter estimates from one or a fixed number of sample subsets.

4 Related Methods

There is a rich literature on robust parameter estimation methods in general (see [19] or [31] for comprehensive treatments) and in computer vision ([27] is a good review). There is a long list of papers that use robust techniques for various computer vision problems, and [4, 9, 24, 23, 39] are but examples of the use of robust estimation for the computation of camera motion or structure-from-motion.

All these approaches work by somehow discounting *data* that are inconsistent with the final solution. In their simplest forms, these methods use *robust norms* to measure the discrepancy between predicted and actual measurements. Robust norms grow less than quadratically as a function of the discrepancy, and in some cases saturate to a finite value. As a result, the large discrepancies associated with outliers have a lesser effect than they have in standard least squares. This approach is used for instance in [39], where a p-norm $\|\cdot\|^p$ with $1 < p < 2$ is used instead of the quadratic norm $\|\cdot\|^2$. In addition to the greater computational complexity of robust norms, they often introduce spurious local minima because of their inflection points. Also, the basins of attraction of local minima with robust norm formulations can be narrower than those for the standard quadratic formulation, thereby making convergence to the correct minimum harder. Rank estimators [27] weigh residuals by their rank in an ordered list.

Different variations on the idea of *cross-validation* have appeared as well. In very broad lines, this works as follows. Part of the data, the validation set, is used to compute an estimate $\hat{\boldsymbol{\vartheta}}$ of the parameters. That estimate is then used to predict the ideal value that new data would have if $\hat{\boldsymbol{\vartheta}}$ were correct. Data whose actual values differ from this prediction too much are discarded as outliers. For instance, in camera motion estimation, one could determine tentative values for the epipolar geometry, and validate image flow measurements by how closely they satisfy the epipolar constraint [40].

Perhaps the most famous example of the cross-validation approach is RANSAC (RANdom SAmple Consensus, [15]). For an estimation problem, there is usually a minimum number of data points that determine the solution. For instance, camera motion estimation from two frames requires at least five image point with correspondences. RANSAC randomly selects a data set of this size, and adds to it points, one at a time, that are consistent, according to a threshold, and in the sense of cross-validation explained above, with this initial solution. If the initial set fails to grow (as determined by another threshold on the number of attempts made to add points to the initial set), the starting set is discarded, and the procedure is restarted. Thus, RANSAC performs explicit validation of the data: a data item is either in or out of the consensus set. This method has been used for many vision problems ever since its first appearance, and often with good results. However, the presence of several thresholds in its definition makes it sometimes hard to apply. Also, the use of a minimal number of data points as a seed can yield a statistically poor validation criterion. To address this, it has been proposed that the parameter estimate be recomputed every time a new data point is added. In this version, the complexity of RANSAC at least equals that of the one-shot solution.

In contrast with these methods, our resampling procedure uses whichever error norm is appropriate for the problem, rather than robust norms, and performs a cross-validation of sorts *in the space of parameters*, rather than in data space. In other words, *solutions* reinforce one another, rather than data points. As discussed in section 3, this approach leads to linear complexity for the robust procedure even for problems that have a superlinear one-shot solution; it allows using quadratic norms in the basic optimization; and it involves essentially no thresholds, since the only parameters r (number of sample sets) and j (size of each sample set) can be assigned generous values with no harm other than computational cost.

The idea underlying the resampling method has already appeared in the statistical literature under the name of *boostrap* or *jackknife*. In the following, we show that bootstrap-jackknife subsumes resampling. However, the former has always been used for *performance evaluation*, that is to evaluate statistical parameters, typically the mean and covariance, of a parameter estimate that is computed by some other method.

The classical bootstrap method was first introduced by Efron (cf. [10, 11, 12, 13]). Given the estimation problem (1), the original goal of the method was to estimate the standard deviation of the estimate by resampling, that is, by drawing m elements from X repeatedly and with replacement, say r times. Let the ith sample set be

$$X_i^* = \{x_1^*, \ldots, x_m^*\}, \qquad i = 1, \ldots, r,$$

where each x_l^* is sampled from the original data set X with replacement, so that the samples in X_i^* may not be distinct. Apply the algorithm to get r estimates of the parameter,

$$\theta_i^* = s(X_i^*), \qquad i = 1, \ldots, r,$$

and let the mean of these be

$$\bar{\theta}^* = \frac{1}{r}\sum_{i=1}^{r} \theta_i^*. \tag{5}$$

Then an estimation of the standard error is given by

$$\sigma^* = \sqrt{\frac{1}{r-1}\sum_{i=1}^{r}(\theta_i^* - \bar{\theta}^*)^2}. \tag{6}$$

The original jackknife method can be found in [30, 37, 13], where there are m jackknife sample sets, with each set being obtained from the original data set X by deleting one sample, that is,

$$X_i^* = \{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_m\}, \quad i = 1, \ldots, m.$$

The expressions for $\bar{\theta}^*$ and σ^* have the same form as in (5) and (6), with $r = m$. This is the same as in the cross–validation in many literatures (cf. [16, 17, 12]). To make the jackknife method work well for non–smooth functions $\theta = s(X)$, the delete–d jackknife methods were studied [33, 32, 13], where the sample sets are all the sets obtained by deleting d samples from X, thus leading to a total of

$$r = \binom{m}{d}$$

jackknife subsample sets. The deletion size d should be large (cf. [13, p.149]), preferably of the same order as m. In other words, most of the points are deleted, that is, the size j of the subsets is small. Note that the differences between delete–d jackknife and bootstrap methods are in their sampling replacement policy (without vs. with), sampling size (delete d vs. size m) and number of sample sets ($\binom{m}{d}$ vs. m).

Thus, our resampling method is essentially a jackknife method, with a large deletion factor d, and with a random rather than systematic selection of the deleted data. However, we do not use this method for performance evaluation, but we rather augment it with a clustering stage, and use it to achieve robustness in parameter estimation problems. We point out that even in vision the bootstrap method has always been used only to evaluate the covariance of solutions. For instance, in [8, 26], the bootstrap method is used to evaluate clustering algorithms and edge detection algorithms, and in [25] it is used to compute the covariance of 3D data inputs and confidence regions in stereo–heading computations.

5 A Case Study: Camera Motion

In this section we first reformulate the camera heading computation problem from visual angles as a parametric estimation problem, which we then use to illustrate the resampling method. In this problem, the direction of translation (heading) of a moving camera is to be determined from the instantaneous motion of a set of image points. A number of solutions [21, 29, 18, 36, 35, 28, 39, 7, 20, 38] have been proposed. Here we apply the resampling method to a simplified formulation of [35] which is introduced hereafter.

For every pair of feature points P and Q in the scene, as shown in Figure 3, the visual angle α is defined as the angle between the projection rays of P and Q. It satisfies

$$\cos\alpha = \boldsymbol{p}^T\boldsymbol{q} \tag{7}$$

where $\boldsymbol{p}$ and $\boldsymbol{q}$ are unit vectors corresponding to the image measurements. These can be computed from the feature point positions in the image and the camera internal parameters. By differentiating both sides of equation (7) with respect to time we obtain

$$\dot{\alpha} = t^T V, \qquad \text{with} \quad V = d_P u_{pq} + d_Q u_{qp} \qquad (8)$$

where $t = \dot{C}$ is the normalized camera velocity, that is, a three dimensional vector pointing along the direction of camera translation, $d_P = 1/|P - C|$ and $d_Q = 1/|Q - C|$ are the inverses of the distances of P and Q from the camera center, and $u_{pq} = (p - (\cos\alpha)q)/\sin\alpha$ and $u_{qp} = (q - (\cos\alpha)p)/\sin\alpha$ are the orthogonal projection vectors shown in Figure 3. For a set X of tracked feature points,

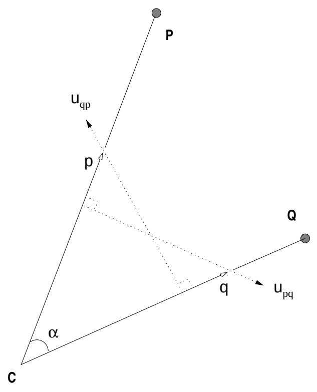


Figure 3: The visual angle α. Feature directions $\boldsymbol{p}$, $\boldsymbol{q}$, and orthogonal projections vectors u_{pq}, u_{qp}.

we can pick a set of point pairs, that is, a graph, to write a system of equations

$$F(\boldsymbol{t})\boldsymbol{d} = \dot{\alpha}.$$

In this system, every equation is of the form (8), and corresponds to an edge of a graph whose vertices are the feature points. The motion direction $\boldsymbol{t}$ is computed by the minimization

$$\boldsymbol{t} = s(X) = \arg\min_{\|\boldsymbol{t}\|=1} \|F(\boldsymbol{t})\boldsymbol{d} - \dot{\alpha}\|.$$

We now show simulations and experiments for the resampling method as applied to the camera heading problem.

Simulations

Figure 4 shows the results of two sets of simulations with $m = 100$ random points in 3D with a known motion direction of $\boldsymbol{t} = [1, -1, 3]/\sqrt{11}$. A rotation of $20°$ has been added, even though this does not affect visual angles. In all simulations, five outlier points were randomly selected, and an error of about ± 10 pixels was added to them. The plots in the left column are in the absence of noise. The solutions with sample sets $j = 10$ (top), 15 (middle) and 20 (bottom) cluster densely around the solution, with close to 60%, 50% and 40% of the sample sets in the correct bin. These values compare favorably with the .95 probability bounds of $58\%, 44\%$, and 32% discussed in section 3. In the right column of figure 4, noise is introduced into the data, in addition to outliers, with a normal distribution with zero mean and a standard deviation of .1 pixels. Although somewhat broadened, the correct clusters are still sharp.

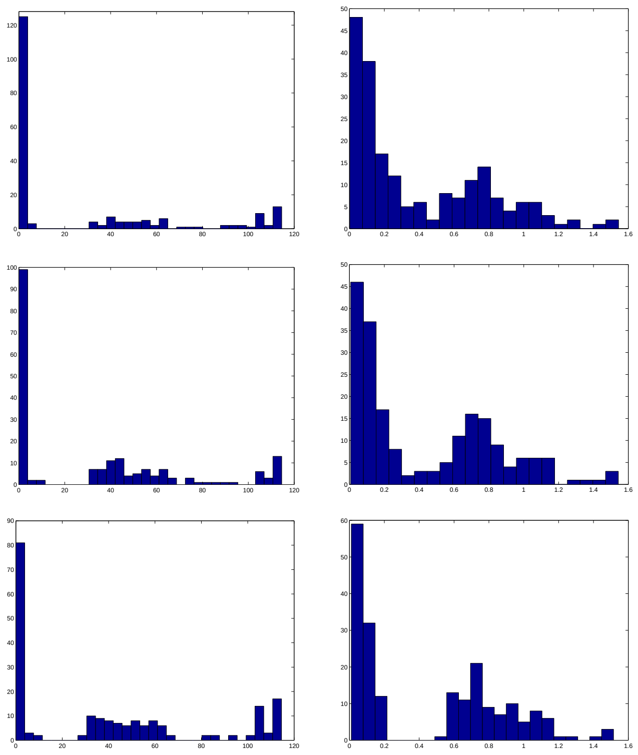


Figure 4: Histogram of the angle errors of 200 solutions from random subsets of size $j = 10$ (top), 15 (middle) and 20 (bottom) from simulation in the presence of 5 outliers out of 100 points. Plots are from both noise-free (left) and noisy (right) simulations.

Real Experiments

The scene is shown in Figure 5 (a). We experimented with 50 consecutive frames and computed the camera heading. Feature points are selected and tracked automatically with the tracker from the algorithms in [22, 34] and the implementations from [2]. The feature tracking results are also displayed in Figure 5 (b) from frame 15 to 16. At least 5 obvious outliers are visible in (b).

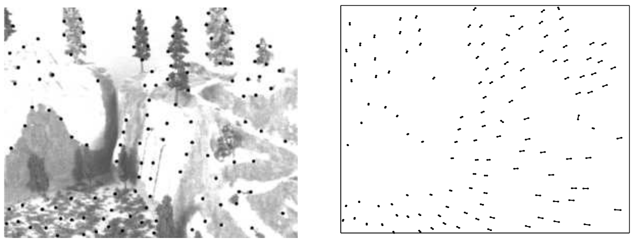

(a) (b)

Figure 5: (a) The scene and features; (b) Feature tracking from Frame 15 to 16 (note the outliers).

The camera heading computation from Frame 15 to 16 is challenging due to the existence of at least 5 outliers. The resampling method deals with this problem very nicely, even though the final direction of camera heading was obtained simply by computing the componentwise median. In our experiments, the camera motion was controlled by a precise positioning device. Although absolute directions cannot be measured precisely, *changes* in diretion can be determined accurately to within one hundredth of a degree with the positioning device, and provide therefore a reliable gold standard.

Table 1 records the resampling solution with $r = 200$ subsets for three choices of subset size j and with median clustering. As expected, the effect of increasing j rapidly tails off once the theoretical minimum $j = 5$ is comfortably exceeded, and $j = 20$ is a good choice. The results for competing methods are also shown in the table. For least squares (LS), the function $\Phi(t) = t^2$ is used for each residue error term. Because of the existence of outliers, the result is a disappointing $-51.3162°$, versus the true direction change of $8.1175°$. The result is better for $L^{1.2}$ where $\Phi(t) = |t|^{1.2}$ is used, but still far off. Similar error terms (H.01) are shown for Huber's robust function (cf.[19]) with $\tau = .01$

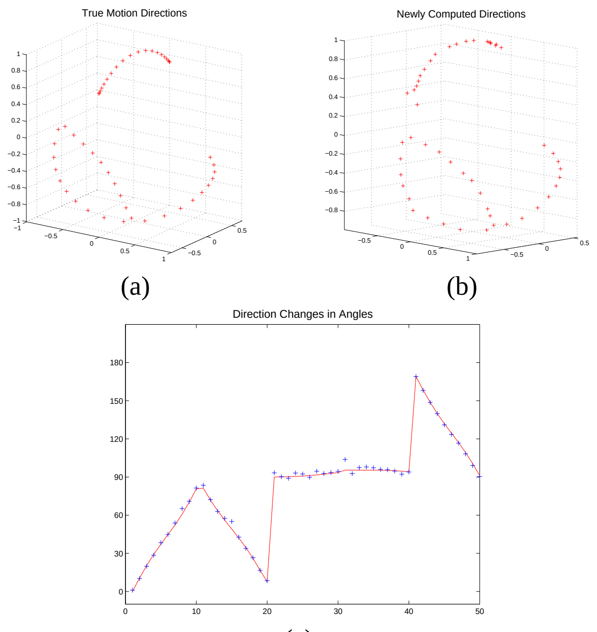
$$\Phi(t) = \begin{cases} t^2, & (|t| \le \tau) \\ \tau^2 + 2\tau * (|t| - \tau), & (|t| > \tau) \end{cases}$$

which uses L^2 around zero, and a linear function farther away. Even though, in theory, the effect of outliers can decrease as $\tau \longrightarrow 0$, negligible changes were observed in practice. For the error function $\Phi(t) = t^2/(t^2+\sigma^2)$, which is utilized in [4, 5, 6] the result for $\sigma = .01$ is recorded in column (B.01).

Figure 6 shows the camera heading results for 50 consecutive frames. The true and computed camera motion directions are shown in (a) and (b). In (c), the angular changes of direction relative to the first frame are depicted. The solid plot represents the true angular changes, as measured by the positioning device, and the crosses are the angular changes from resampling. The few visible errors occur for almost-vanishing camera translation, in which case the camera heading problem becomes nearly singular. Close to singularities, the signal to noise ratio is exceedingly small, and no method can return very accurate results.

Method	Computed Direction ° Change	Error °
True Value	8.1175	0
Resampling, $j = 10$	9.1332	1.0157
Resampling, $j = 15$	7.6672	-0.4503
Resampling, $j = 20$	8.6156	0.4981
LS	51.3162	43.1987
$L^{1.2}$	28.2116	20.0941
$H.01$	24.7685	16.6510
$B.01$	24.9054	16.7875

Table 1: The camera motion direction changes in degrees between Frames 15⟶16 and Frames 16⟶17, calculated by various methods.



(a) (b)

(c)

Figure 6: Experiments on 50 frames: (a) true and (b) computed camera motion directions; (c) angular changes of direction relative to the first frame.

6 Conclusions

The resampling method has shown to be very effective for handling outliers in camera motion estimation, achieving results that are more than an order of magnitude more accurate than competing robust methods. This method applies to any optimization procedure without requiring any modification, is computationally efficient, and can be easily implemented in parallel.

Conceivably, this method can be applied to many other problems in computer vision, and we have started to explore this in our current work.

Acknowledgements

This research was supported by NSF Grants IRI-9201751 and IRI-9496205. The authors thank Professors Brad Efron, Jim Dai, and Gene Golub for discussions.

References

[1] S. Avidan and A. Shashua. Novel view synthesis by cascading trilinear tensors. *IEEE Trans. on Visual. and Comp. Graphics*, 4:293–306, 1998.

[2] S. Birchfield. Klt: An implementation of the Kanade–Lucas–Tomasi feature tracker. *http://vision.stanford.edu/~birch/klt*, 1997.

[3] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[4] M. J. Black and P. Anandan. Robust dynamic motion estimation over time. In *CVPR*, 296–302, 1991.

[5] M. J. Black and A. Rangarajan. On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *IJCV*, 19(1):57–92, 1996.

[6] M. J. Black, G. Sapiro, D. Marimont, and D. Heeger. Robust anisotropic diffusion. *IEEE Trans. on Im. Proc.*, 7(3):421–432, 1998.

[7] A. R. Bruss and B. K. P. Horn. Passive navigation. *CVGIP*, 21(1):3–20, 1983.

[8] K. Cho, P. Meer, and J. Cabrera. Performance assessment through bootstrap. *PAMI*, 19(11):1185–1198, 1997.

[9] R. Cipolla, Y. Okamoto, and Y. Kuno. Robust structure from motion using motion parallax. In *ICCV*, 374–382, 1993.

[10] B. Efron. Bootstrap methods: another look at the jackknife. *Ann. Stat.*, 7:1–26, 1979.

[11] B. Efron. *The Jackknife, the Bootstrap, and Other Resampling Plans*. SIAM, Philadelphia, PA, 1982.

[12] B. Efron and G. Gong. A leisurely look at the bootstrap, the jackknife, and cross-validation. *J. Amer. Stat.*, 37(1):36–48, 1983.

[13] B. Efron and R. J. Tibshirani. *An introduction to the bootstrap*. Chapman and Hall, New York, NY, 1993.

[14] W. Feller. *An introduction to probability theory and its applications*. Wiley, New York, NY, 1968.

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

[16] G. H. Golub, M. T. Heath, and G. Wahba. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21:215–223, 1979.

[17] C. Gu, D. M. Bates, Z. H. Chen, and G. Wahba. The computation of generalized cross-validation functions through Householder tridiagonalization with applications to the fitting of interaction spline models. *SIAM J. Mat. Anal. Appl.*, 10(4):457–480, 1989.

[18] D. J. Heeger and A. D. Jepson. Subspace methods for recovering rigid motion i: Algorithm and implementation. *IJCV*, 7(2):95–118, 1992.

[19] P. J. Huber. *Robust Statistics*. John Wiley and Sons, New York, NY, 1981.

[20] K. Kanatani. 3-d interpretation of optical flow by renormalization. *IJCV*, 11(3):267–282, 1993.

[21] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981.

[22] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, 1981.

[23] C. B. Madsen. A comparative study of the robustness of two pose estimation techniques. *Machine Vision and Appl.*, 9(6):291–303, 1997.

[24] T. Masuda. Robust estimation of rigid motion parameters between a pair of range images. In *Proc. Scand. Conf. on Image Analysis*, 21–39, Tromso, Norway, 1993.

[25] B. Matei and P. Meer. Optimal rigid motion estimation and performance evaluation with bootstrap. *CVPR*, 339–345, 1999.

[26] K. Meena, V. Ganapathy, and A. Balasubramaniam. Bootstrap approach to validate clustering solution. In *2nd Int'l. Conf. on Automation, Robotics and Computer Vision*, 1992.

[27] P. Meer, D. Mintz, and A. Rosenfeld. Robust regression methods for computer vision: a review. *IJCV*, 6(1):59–70, 1991.

[28] J. Oliensis. Computing the camera heading from multiple frames. In *CVPR*, 203–210, 1998.

[29] K. Prazdny. On the information in optical flows. *CGIP*, 22:239–259, 1983.

[30] M. Quenouille. Approximate tests of correlation in time series. *J. Royal Statist. Soc.*, B11:18–44, 1949.

[31] G. A. F. Seber and C. J. Wild. *Nonlinear Regression*. John Wiley and Sons, New York, NY, 1989.

[32] J. Shao. Consistency of jackknife variance estimator. *Statistics*, 22:49–57, 1991.

[33] J. Shao and C.F.J. Wu. A general theory for jackknife variance estimation. *Ann. Stat.*, 17:1176–1197, 1989.

[34] J. Shi and C. Tomasi. Good features to track. In *CVPR*, 593–600, 1994.

[35] C. Tomasi and J. Shi. Image deformations are better than optical flow. *Math. and Comp. Modeling J.*, 24(5/6):165–175, 1996.

[36] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *IJCV*, 9(2):137–154, 1992.

[37] J. W. Tukey. Bias and confidence in not quite large samples. *Ann. Math. Stat.*, 29:614, 1958.

[38] J. Weng, N. Ahuja, and T. S. Huang. Optimal motion and structure estimation. *PAMI*, 15(9):864–884, 1993.

[39] T. Zhang and C. Tomasi. Fast, robust, and consistent camera motion estimation. In *CVPR*, (I):164–170, 1999.

[40] Z. Zhang. Determining the epipolar geometry and its uncertainty: a review. *IJCV*, 27(2):161–195, 1998.