Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Image and Vision Computing 21 (2003) 1145–1153

www.elsevier.com/locate/imavis

Robust registration of 2D and 3D point sets

Andrew W. Fitzgibbon[1]

Department of Engineering Science, University of Oxford, 19 Parks Road, Oxford OX1 3PJ, UK

Abstract

This paper introduces a new method of registering point sets. The registration error is directly minimized using general-purpose non-linear optimization (the Levenberg–Marquardt algorithm). The surprising conclusion of the paper is that this technique is comparable in speed to the special-purpose Iterated Closest Point algorithm, which is most commonly used for this task. Because the routine directly minimizes an energy function, it is easy to extend it to incorporate robust estimation via a Huber kernel, yielding a basin of convergence that is many times wider than existing techniques. Finally, we introduce a data structure for the minimization based on the chamfer distance transform, which yields an algorithm that is both faster and more robust than previously described methods.

Keywords: Iterated Closest Point; Range image registration; Levenberg–Marquardt

1. Introduction

A common problem in computer vision is the registration of 2D and 3D point sets [1,5,7,8,21,28]. Applications include the integration of range datasets [14,25], and alignment of MRI/CAT scans [9,22]. Typically, a cloud of point samples from the surface of an object is obtained from two or more points of view, in different reference frames. The task of *registration* is to place the data into a common reference frame by estimating the transformations between the datasets. What makes the problem difficult is that correspondences between the point sets are unknown a priori. A popular approach to solving the problem is the class of algorithms based on the Iterated Closest Point (ICP) technique introduced by Besl [1] and Zhang [28]. ICP is attractive because of its simplicity and its performance. Although the initial estimate does need to be reasonably good, the algorithm converges relatively quickly.

This paper abandons one of the basic characteristics of ICP—its closed-form inner loop—and employs instead a standard iterative non-linear optimizer, the Levenberg–Marquardt (LM) algorithm [19]. This approach, perhaps surprisingly, incurs no significant loss of speed, but allows the extension of ICP to use truly robust statistics, with a concomitant reduction of dependence on the initial estimate. In contrast, existing ways of introducing robustness [4,24,28] are often many times slower than the proposal of this paper.

The paper is in several sections. Section 2 defines the problem and the notation used in the rest of the paper. In Section 3, we review existing work on point-set registration, ICP and otherwise, and also summarize the key ideas in non-linear optimization that will be required. Section 4 redefines registration as a non-linear optimization problem. This approach is compared with traditional ICP in 2D and 3D experiments. The second half of the paper, beginning in Section 5, exploits the simplicity of the new approach to develop a robust error function, which greatly increases the radius of convergence on supplied examples, again with no significant loss in speed.

In all the above examples, closest point computations are based on explicit Delaunay simplicization of the point set, allowing $O(\log n)$ closest point computations. In Section 6, we show how both the new and traditional procedures can be modified to use a fast lookup based on the distance transform. This speeds up both algorithms significantly, but the proposed technique benefits more, resulting in an algorithm which is faster and more accurate than traditional ICP, with a wider basin of convergence.

More generally, the message of the paper is that specialized 'home-grown' strategies for function minimization (of which ICP is one example) do not necessarily outperform more general—but more sophisticated—non-linear optimization algorithms.

E-mail address: awf@robots.ox.ac.uk (A.W. Fitzgibbon).
[1] http://www.robots.ox.ac.uk/~awf

0262-8856/$ - see front matter © 2003 Published by Elsevier B.V.
doi:10.1016/j.imavis.2003.09.004

2. Problem statement and definitions

The paper deals exclusively with the two-frame case, although multiple-frame approaches [8,21] should immediately benefit. We are given two sets of points in $\mathbb{R}^n$, which for convenience we shall denote by *model* and *data*, with their elements being denoted by $\{\mathbf{m}_i\}_{i=1}^{N_m}$ and $\{\mathbf{d}_i\}_{i=1}^{N_d}$. The task of registration is to determine the parameters of a transformation T which, when applied to the data points, best aligns model and data. The parameters of T are represented by a p-vector $\mathbf{a}$. Common transformations and corresponding values of p are listed in Table 1.

For 2D registration (see Fig. 1 for an example problem and Fig. 2 for an example of LM-ICP output) under Euclidean transformations, the number of parameters $p = 3$ comprising rotation angle θ and translation vector (t_x, t_y). Collecting the parameters into a parameter vector $\mathbf{a} = [\theta, t_x, t_y]$, we define

$$T_{2D}(\mathbf{a};\mathbf{x}) = T(\theta, t_x, t_y;\mathbf{x}) = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}\mathbf{x} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} \quad (1)$$

for $\mathbf{x} \in \mathbb{R}^2$

Alignment is measured by an error function $\epsilon^2(|\mathbf{x}|)$, and a typical choice is to define

$$\epsilon^2(|\mathbf{x}|) = \|\mathbf{x}\|^2$$

In order to measure alignment, we require that correspondence between the model and data points is specified. This correspondence is denoted by the function $\phi(i)$, which selects, for each data point, the corresponding model point. In order to cope with data points for which no correspondent is found, we also introduce *weights* w_i, which are set to zero for points with no match, and one otherwise. Thus, the error to be minimized is

$$E(\mathbf{a}, \phi) = \sum_{i=1}^{N_d} w_i \epsilon^2(|\mathbf{m}_{\phi(i)} - T(\mathbf{a};\mathbf{d}_i|) \quad (2)$$

In general, the function ϕ is considered part of the minimization process: in ICP-like algorithms $\phi(i)$ is chosen as the point which minimizes the distance between model and data, yielding the error function

$$E(\mathbf{a}) = \sum_{i=1}^{N_d} w_i \min_j \epsilon^2(|\mathbf{m}_j - T(\mathbf{a};\mathbf{d}_i)|) \quad (3)$$

and finally the estimate of the optimal registration is given by minimizing over $\mathbf{a}$:

$$\hat{\mathbf{a}} = \operatorname*{argmin}_{\mathbf{a}} \sum_{i=1}^{N_d} w_i \min_j \epsilon^2(|\mathbf{m}_j - T(\mathbf{a};\mathbf{d}_i)|)$$

Table 1
Transformations commonly occurring in registration problems

nD transformation	p	Parameters
2D Euclidean	3	Rotation θ, 2D translation (t_x, t_y)
2D Affine	6	2×3 matrix
2D Projective	8	3×3 homography matrix (defined up to scale)
3D Euclidean	6	3 rotation, 3 translation
3D Similarity	7	3 rotation, 3 translation, 1 scale
3D Affine	12	3×4 matrix
3D Projective	15	4×4 matrix (defined up to scale)

Fig. 1. 2D curves to be registered. The curves are subsets of the output of an edge detector applied to the image. Synthetic rotations and translations of the lower-right 'C' are registered to the upper left one.

2.1. The ICP algorithm

In its simplest form, the ICP algorithm iterates two steps. Beginning with an initial estimate of the registration parameters, $\mathbf{a}_0$, the algorithm forms a sequence of estimates

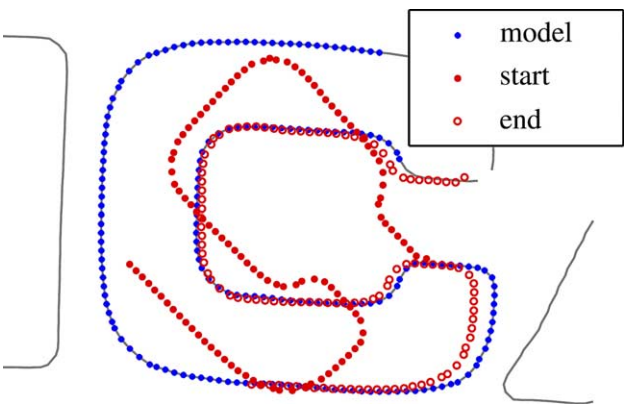

Fig. 2. Convergence of LM-ICP with Lorentzian kernel $\epsilon(r) = \log(1 + r^2)$ from a 40° rotation.

$\mathbf{a}_k$, which progressively reduce the error $E(\mathbf{a})$. Each iteration of the algorithm comprises the following two steps, labelled C and T:

C: Compute *correspondences*, ϕ : Set

$$\phi(i) = \underset{j\in\{1,\dots,N_m\}}{\operatorname{argmin}} \epsilon^2(|\mathbf{m}_j - T(\mathbf{a}_k;\mathbf{d}_i)|) \qquad i = 1,\dots,N_d$$

so that $\mathbf{m}_{\phi(i)}$ is the closest model point to the datum $\mathbf{d}_i$ transformed by the current estimate $\mathbf{a}_k$.

T: Update *transformation*, $\mathbf{a}$: Set

$$\mathbf{a}_{k+1} = \underset{\mathbf{a}}{\operatorname{argmin}} \sum_{i=1}^{N_d} \epsilon^2(|\mathbf{m}_{\phi(i)} - T(\mathbf{a};\mathbf{d}_i)|)$$

In many common cases, step T can be performed in closed form, or via well understood, numerically stable, procedures such as singular value decomposition.

It is easy to see that both steps must reduce the error, and that the error is bounded below. Thus convergence to a local minimum is guaranteed. Furthermore, it is straightforward to discern a termination criterion: when the set of correspondences does not change in step C, the value of $\mathbf{a}_{k+1}$ will be set equal to $\mathbf{a}_k$ in the T step, so no further change is possible.

2.2. The Levenberg–Marquardt algorithm

The proposal of this paper is to directly minimize the model-data fitting error (3) via non-linear minimization. The LM algorithm is an optimization procedure, that is particularly suited to functions such as E which are expressed as a sum of squared *residuals*, but alternative procedures such as conjugate gradients or even a pure Gauss–Newton algorithm could be used, and similar results would be expected. In this paper, only LM was tested.

2.2.1. Computing derivatives of E

The question that will immediately arise regarding the application of LM to the ICP problem is how the derivatives of E may be computed. Indeed the requirement for first derivatives might be seen as immediately disqualifying E from LM optimization, given the discrete minimization over j within the summation. However, it will be seen in Section 6 that in fact the derivatives of E may be easily and efficiently obtained, and that they are, or can be made, smooth.

For the moment, we shall assume they behave smoothly, and compute them via finite differencing [13,19], at a cost of p extra function evaluations per inner loop. This means that each iteration's cost is increased by a factor of $1 + p$, for the simplest form of LM-ICP. However, in typical cases, where LM requires fewer iterations to achieve a certain accuracy, this factor is an upper bound. In many cases, the reduction in number of iterations will exceed the increase in per-iteration cost. Finally, it must be emphasized that an implementer concerned with speed should read Section 6. If the techniques of that section are impossible to implement in a given application, there are better ways to use function evaluations than to compute finite-difference derivatives as suggested above. See Brent. [3] for a detailed discussion of derivative-free minimization strategies. One further note: if derivatives are computed via finite differences, calculations of the form $(E(\mathbf{a}+\delta) - E(\mathbf{a}))/\|\delta\|$ are required. It is important that the closest point computations are repeated when making these calculations. It might be thought more efficient to compute closest points once, when calculating $E(\mathbf{a})$, and to use the same values of ϕ to compute $E(\mathbf{a}+\delta)$. However, this will compute the derivatives with fixed correspondences, and the gradient descent direction will then be towards the same optimum as would be attained by the basic ICP algorithm. It is better, particularly with a robust kernel, to allow the correspondences to change for each δ.

2.2.2. The LM algorithm in detail

There now follows a derivation of the LM algorithm. Readers who are familiar with its operation may wish to skim this description in order to synchronize notation, and proceed to Section 3. The error function $E(\mathbf{a})$ can be written as the sum of N_d residuals as follows

$$E(\mathbf{a}) = \sum_{i=1}^{N_d} E_i^2(\mathbf{a})$$

where the residual for the ith data point is given by

$$E_i(\mathbf{a}) = \sqrt{w_i} \min_j \epsilon(|\mathbf{m}_j - T(\mathbf{a};\mathbf{d}_i)|)$$

An important concept in the derivation of LM will be the vector of residuals

$$\mathbf{e}(\mathbf{a}) = \{E_i(\mathbf{a})\}_{i=1}^{N_d}$$

in terms of which the error function becomes $E(\mathbf{a}) = \|\mathbf{e}(\mathbf{a})\|^2$.

The LM algorithm combines the gradient descent and Gauss–Newton approaches to function minimization. Using the above notation, the goal at each iteration is to choose an update to the current estimate $\mathbf{a}_k$, say $\mathbf{x}$, so that setting $\mathbf{a}_{k+1} = \mathbf{a}_k + \mathbf{x}$ reduces the error $E(\mathbf{a})$. Expanding $E(\mathbf{a}+\mathbf{x})$ around $\mathbf{a}$, we obtain

$$E(\mathbf{a}+\mathbf{x}) = E(\mathbf{a}) + (\nabla E(\mathbf{a})\cdot\mathbf{x}) + \frac{1}{2!}((\nabla^2 E(\mathbf{a})\cdot\mathbf{x})\cdot\mathbf{x}) + \text{h.o.t.}$$

Expressing this in terms of $\mathbf{e}$, we have

$$E(\mathbf{a}) = \mathbf{e}^{\mathrm{T}}\mathbf{e} \tag{4}$$

$$\nabla E(\mathbf{a}) = 2(\nabla\mathbf{e})^{\mathrm{T}}\mathbf{e} \tag{5}$$

$$\nabla^2 E(\mathbf{a}) = 2(\nabla^2\mathbf{e})\mathbf{e} + 2(\nabla\mathbf{e})^{\mathrm{T}}\nabla\mathbf{e} \tag{6}$$

We shall denote the $N_d \times p$ *Jacobian* matrix $\nabla\mathbf{e}$ by $\mathbf{J}$, with ijth entry $J_{ij} = \partial E_i/\partial a_j$. Introducing the Gauss–Newton approximation [19], i.e. $(\nabla^2\mathbf{e})\mathbf{e} \approx 0$, we arrive at

$$E(\mathbf{a}+\mathbf{x}) \approx \mathbf{e}^{\mathrm{T}}\mathbf{e} + 2\mathbf{x}^{\mathrm{T}}\mathbf{J}^{\mathrm{T}}\mathbf{e} + \mathbf{x}^{\mathrm{T}}\mathbf{J}^{\mathrm{T}}\mathbf{J}\mathbf{x}$$

The task at each iteration is to determine a step $\mathbf{x}$, which will minimize $E(\mathbf{a}+\mathbf{x})$. Using the approximation to E that we have just derived, we differentiate with respect to $\mathbf{x}$ and equate with zero, yielding

$$\nabla_{\mathbf{x}} E(\mathbf{a}+\mathbf{x}) = 2\mathcal{J}^{\mathrm{T}}\mathbf{e} + 2\mathcal{J}^{\mathrm{T}}\mathcal{J}\mathbf{x} = 0$$

Solving this equation for $\mathbf{x}$ yields the Gauss–Newton update, and gives the algorithm for one iteration of Gauss–Newton ICP:

1. Compute the vector of residuals $\mathbf{e}(\mathbf{a}_k)$, and its $N_d \times p$ matrix of derivatives $\mathcal{J}$ with respect to the components of $\mathbf{a}$. (For a 2D rigid-body transformation, $\mathbf{a}$ has three components, and $\mathcal{J}$ is $N_d \times 3$).
2. Compute the update $\mathbf{x} = -(\mathcal{J}^{\mathrm{T}}\mathcal{J})^{-1}\mathcal{J}^{\mathrm{T}}\mathbf{e}$.
3. Set $\mathbf{a}_{k+1} = \mathbf{a}_k + \mathbf{x}$.

Of course, the above strategy does not guarantee that the step taken will result in a reduced error at $E(\mathbf{a}_{k+1})$. Whether or not it does so depends on the accuracy of the second-order Taylor series expansion at $\mathbf{a}_k$, and on the validity of the Gauss–Newton approximation. However, it can be shown that when these approximations are good, as they tend to be when near the minimum, convergence is rapid and reliable.

By comparison, an accelerated gradient descent approach as used by some previous registration algorithms [1,8,20] is obtained by replacing step 2 with

2. Compute the update $\mathbf{x} = -\lambda^{-1}\mathcal{J}^{\mathrm{T}}\mathbf{e}$, where the value of λ controls the distance travelled along the gradient direction. For small λ, the iteration moves a long way along the downhill direction; large λ implies a short step. In contrast to Gauss–Newton, gradient descent does guarantee to reduce E, providing λ is sufficiently large. However, its convergence near the optimum is dismally slow.

The LM algorithm combines both updates in a relatively simple way in order to achieve good performance in all regions. Step 2 is replaced by

2′. Compute the update $\mathbf{x} = -(\mathcal{J}^{\mathrm{T}}\mathcal{J} + \lambda\mathcal{I})^{-1}\mathcal{J}^{\mathrm{T}}\mathbf{e}$.

Now large λ corresponds to small, safe, gradient descent steps, while small λ allows fast convergence near the minimum. The art of a good LM implementation is in tuning λ after each iteration to ensure rapid progress even where the Gauss–Newton approximations are poor. Details of such strategies may be found in Refs. [16,19].

We have therefore derived this paper's first proposal, the LM-ICP algorithm, summarized in Fig. 3.

function $\mathbf{a}$ = **lmicp** ($\{\mathbf{m}_j\}_{j=1}^{N_\mathbf{m}}$, $\{\mathbf{d}_i\}_{i=1}^{N_\mathbf{d}}$, $\mathbf{a}_0$)
Set λ to an initial value —*see Press et al[19], p.684*
Set $\mathbf{a} = \mathbf{a}_0$
repeat
  Compute $\mathbf{e}_k = \mathbf{e}(\mathbf{a})$ —*One closest-point computation*
  Compute $\mathcal{J}$ —*p closest-point computations (see §2.2.1)*
  Modify λ until $\mathbf{a}_k = \mathbf{a} - (\mathcal{J}^{\top}\mathcal{J} + \lambda\mathcal{I})^{-1}\mathcal{J}^{\top}\mathbf{e}_k$ reduces the error $\|\mathbf{e}(\mathbf{a}_k)\|^2$.
    —*One or more closest-point computations*
  Set $\mathbf{a} = \mathbf{a}_k$
until λ is large —*So only a small gradient descent step reduced the error.*

Fig. 3. The basic LM-ICP algorithm. No data structure is implied for the closest point computations, see Section 6 for a fast LM-ICP.

3. Previous work

Having defined the problem and thus introduced our notation (Table 2), we are in a position to compare existing work on point-set registration. Besl [1] introduced the name ICP, and provided enough examples of the performance of the basic algorithm to ensure its enduring popularity.

Zhang [28] enhanced the basic technique by replacing the error function $\epsilon^2(|x|) = x^2$ by a robust kernel [12,19]. Chen and Medioni [5] assume the model points $\mathbf{m}_i$ can be supplied with surface normals $\mathbf{n}_i$, which allows the point-to-point distance to be replaced by a point-to-tangent plane. Both of these extensions confer improved convergence properties on the basic algorithm without a significant increase in computational cost.

The problem of multiple local minima has been addressed by using colour [15] and curvature [9] properties to improve, the correspondence step—points are allowed to match based not just on proximity, but also on similarity of surface shape or texture. These extensions are as easily included in this paper's algorithm as in the original, but are not investigated here, as they would be expected to favour both old and new approaches equally. Extensions to non-Euclidean and non-rigid registration [9,22], replace the rigid-body computation in the T step with more general parameterized transformations. Again, the modifications to LM-ICP which would incorporate these extensions are straightforward.

The extensions to multiple-view reconstructions [8,21] necessitated the introduction of non-linear optimization strategies, although these were essentially limited to variations on gradient descent. Grimson [11] explicitly minimizes $E(\mathbf{a})$ again using a non-linear optimizer. However, the important distinction between these efforts and the current work is that they retain *lock-step*: the separation of the correspondence and update steps, and the concomitant sluggish convergence near the minimum.

The separation of correspondence and update remains true of EM-based approaches [6,17,27]. Although

Table 2
Notation summary for the paper

Notation summary
$N_\mathbf{m}$ model points $\mathbf{m}_j$
$N_\mathbf{d}$ data points $\mathbf{d}_j$
Data point $\leftrightarrow$ closest model point: $i \leftrightarrow \phi(i)$
Transformation $T(\mathbf{a};\mathbf{x})$, parameters $\mathbf{a} \in \mathbb{R}^p$
Error $\Delta_i(\mathbf{a}) = \mathbf{m}_{\phi(i)} - T(\mathbf{a};\mathbf{d}_i)$
Residuals $\mathbf{e} \in \mathbb{R}^{N_d}$, elements $E_i = \epsilon^2(\|\Delta_i(\mathbf{a})\|)$
Jacobian $\mathcal{J}$, size $N_d \times p$, ijth element $= \partial E_i/\partial a_j$

the correspondence is 'soft', it is still computed in a separate step, meaning that there is no opportunity to simultaneously adjust correspondence and transform parameters. In medical imaging problems [22,26,27], particularly the registration of MRI or CT scans, explicit minimization over registration parameters is commonplace, as the problems solved there (maximization of mutual information for example), do not admit a closed-form optimization step. However, there the problems do not generally allow for the computation of derivatives, so simple adaptation to 3D point-set registration would not yield the improvements shown in this paper.

The closest work to that reported here is the paper of Champleboux et al. [4], who use LM and the distance transform. However, the work assumes no outliers in the data, reverting to an explicit lock-step in order to reject gross errors. However, for the non-robust case, part of this paper's contribution is the comparison of the Champleboux and Besl approaches with the result that the former is found to be significantly faster and more accurate.

4. Experiments: LM-ICP

In this section, we compare the simplest case of point registration: minimizing the distance between two pointsets where the data are almost entirely a subset of the model. This is the case to which the simplest forms of ICP and LM-ICP apply. In this simple case, LM derivatives are computed using finite differences, which means each LM iteration is a factor of p more expensive than the ICP iteration. However, in Section 6, this cost will disappear, so that iteration counts are indeed the correct abscissae in this section.

4.1. 2D curve matching

The first experiment investigates the registration of 2D curves under Euclidean transformations. Fig. 1 shows a section of an image of a book cover, with overlaid edge-detected curves. The task is to register two curves. The curves to be registered were chosen from different parts of the image in order to obtain a realistic variation in arc-length and sampling artefacts. The curves were subjected to synthetic rotations in order to test the algorithms over a range of initial conditions. Results are displayed graphically in Figs. 4 and 5. The summary of these results is that LM-ICP has a slightly larger basin of convergence, and can find an optimum with slightly reduced error. The LM algorithm requires 50% fewer iterations on average, but unless speedups such as those in Section 6 are employed, this advantage will be lost in the computation of finite-difference derivatives. Even in this case, it might be thought surprising that the general-purpose LM approach is

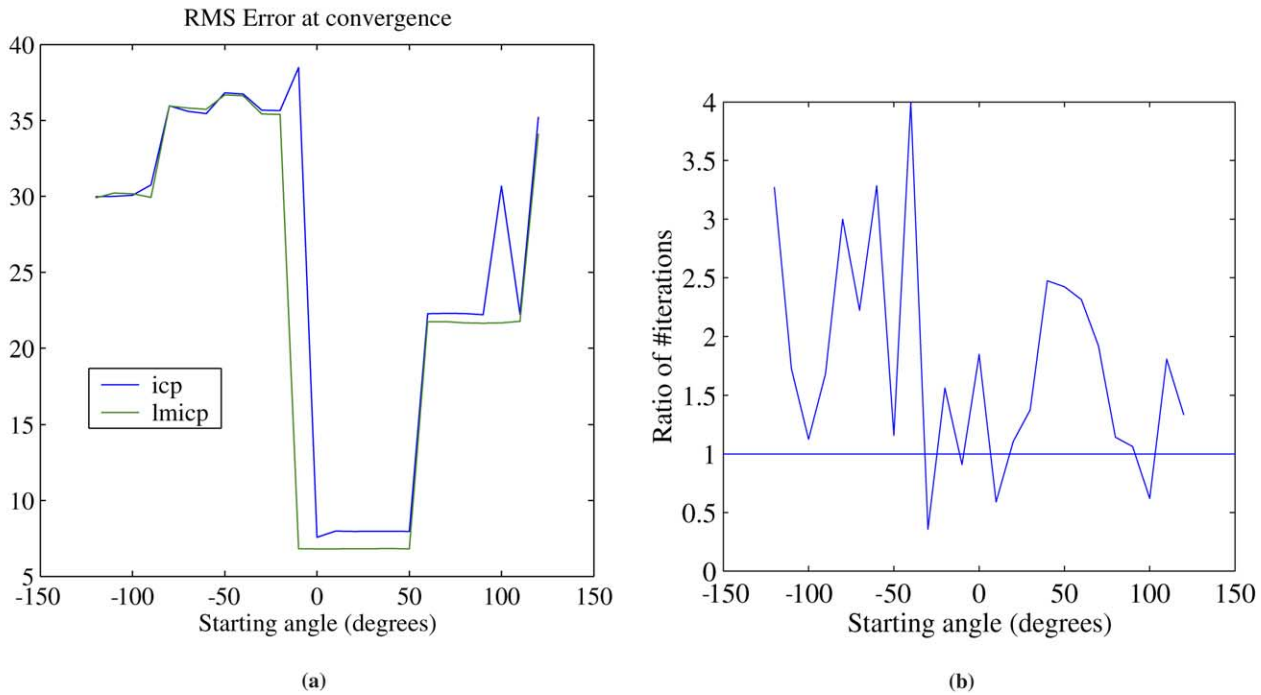

Fig. 4. (a) Basins of convergence of basic ICP and basic LM-ICP are similar. The graph is as a function of starting position, so the error at $X = 0$ is near to that corresponding to the global optimum. All solutions with this error were visually confirmed to be at the global optimum. LM-ICP has a slightly wider basin of convergence, and achieves a slightly lower error (because ICP has more local minima near the solution). (b) Ratio of number of iterations. Numbers greater than 1 imply LM-ICP required fewer iterations, less than 1 implies fewer ICP iterations. The algorithms are comparable, with LM-ICP slightly better. The ratio of the means is 1.52. This is before the speedups in Section 6 are applied, and before robust kernels are introduced, both of which accentuate the difference.

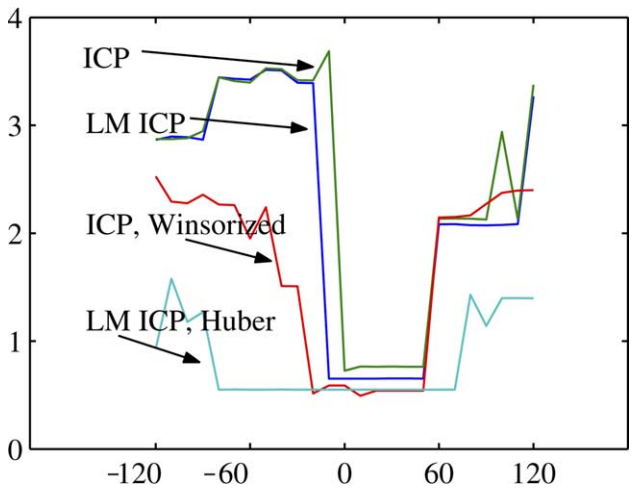

Fig. 5. Basins of convergence: robust kernels. Algorithms are initialized at 1° intervals between ±120° of the true solution, and the value of the minimum is plotted as a function of initial guess. LM–Huber is significantly wider than all others, at a cost of a factor of 2 in number of iterations.

even comparable with ICP's closed-form inner optimization, and this is really the key message of this paper.

4.2. 3D point matching

The extension of the algorithm to 3D is extremely straightforward, and an example is shown in Fig. 6b. Frames `bun000` and `bun045` of the Stanford 'bunny' dataset [25] were presented to LM–Huber (see Section 5), with the identity transformation as initial guess. ICP failed to converge, while LM–Huber produced the visually correct solution in the figure.

5. Robust estimation

Many attempts have been made to widen the basin of convergence of the ICP algorithm, and these largely amount to introducing robust estimation. This is difficult with standard ICP, as no closed-form robust estimate of the T step is known, so authors have used either a non-linear or RANSAC-based estimator [20,24,28], or exclude (*Winsorise*) points with large errors at the C step [4]. With LM-ICP, it is trivial to modify the error function to include a robust kernel. One must take a little care to ensure that the LM algorithm behaves well if the kernel chosen is not smooth, but without going into the details, the examples here use either of the following kernels

$$\text{Lorentzian}: \quad \epsilon(r) = \log\left(1 + \frac{r^2}{\sigma}\right)$$

or

$$\text{Huber}: \quad \epsilon(r) = \begin{cases} r^2 & r < \sigma \\ 2\sigma|r| - \sigma^2 & \text{otherwise} \end{cases}$$

LM-ICP using these kernels is compared against ICP with Winsorised residuals [4], as this is the most common way of robustifying ICP. Fig. 5 shows that LM-ICP with the Huber kernel has a basin of convergence twice as large as that of Winsorised ICP.

6. Fast ICP using the distance transform

The Euclidean distance transform [2] of a set of points $\mathcal{M} = \{\mathbf{m}_j\}_{j=1}^{N_m}$ is defined as

$$D_E(\mathbf{x}) = \min_j \|\mathbf{m}_j - \mathbf{x}\| \tag{7}$$

Algorithms exist for its computation on a discrete grid [2], which are extremely efficient, taking time which is a small constant number of machine instructions times the number of points plus the resolution of the grid. In 3D, the signed distance transform is the preeminent data structure for merging 3D models [14], meaning that it is readily available for the registration step.

We can analogously define the ϵ-distance transform D_ϵ by

$$D_\epsilon(\mathbf{x}) = \min_j \epsilon^2(|\mathbf{m}_j - \mathbf{x}|) \tag{8}$$

and if the mapping $\|\mathbf{x}\| \mapsto \epsilon^2(|\mathbf{x}|)$ is monotonic, we obtain that $D_\epsilon(\mathbf{x}) = \epsilon^2(|D_E(\mathbf{x})|)$, so that existing algorithms to compute D_E may be used to compute D_ϵ, without requiring knowledge of the form of ϵ. In this section, we write D for D_ϵ.

Now, the relationship between the distance transform and the registration problem is easily noted. Combining the definitions of D (Eqs. (7) and (8)) and of the error (3) we obtain

$$E(\mathbf{a}) = \sum_{i=1}^{N_d} w_i D(T(\mathbf{a}; \mathbf{d}_i)) \tag{9}$$

Furthermore, we are now in a position to compute derivatives of E. We compute a discretization of D (Fig. 7a), from which we can immediately compute finite-difference derivatives (Fig. 7b and c). For example, the x derivative image D_x can be given by convolution of D with the discrete kernel [1,0, − 1]. Thus, we have ready access to $\nabla_{\mathbf{x}} D$, which remains constant throughout the minimization. Differentiating Eq. (9) and applying the chain rule, we obtain

$$\nabla_{\mathbf{a}} E(\mathbf{a}) = \sum_{i=1}^{N_d} w_i \nabla_{\mathbf{a}} D(T(\mathbf{a}; \mathbf{d}_i)) \tag{10}$$

$$\nabla_{\mathbf{a}} E(\mathbf{a}) = \sum_{i=1}^{N_d} w_i \nabla_{\mathbf{a}} T(\mathbf{a}; \mathbf{d}_i) \cdot \nabla_{\mathbf{x}} D(T(\mathbf{a}; \mathbf{d}_i)) \tag{11}$$

As a specific example, consider the case of 2D Euclidean registration. The transformation $T(\mathbf{a}; (x, y))$ is given by

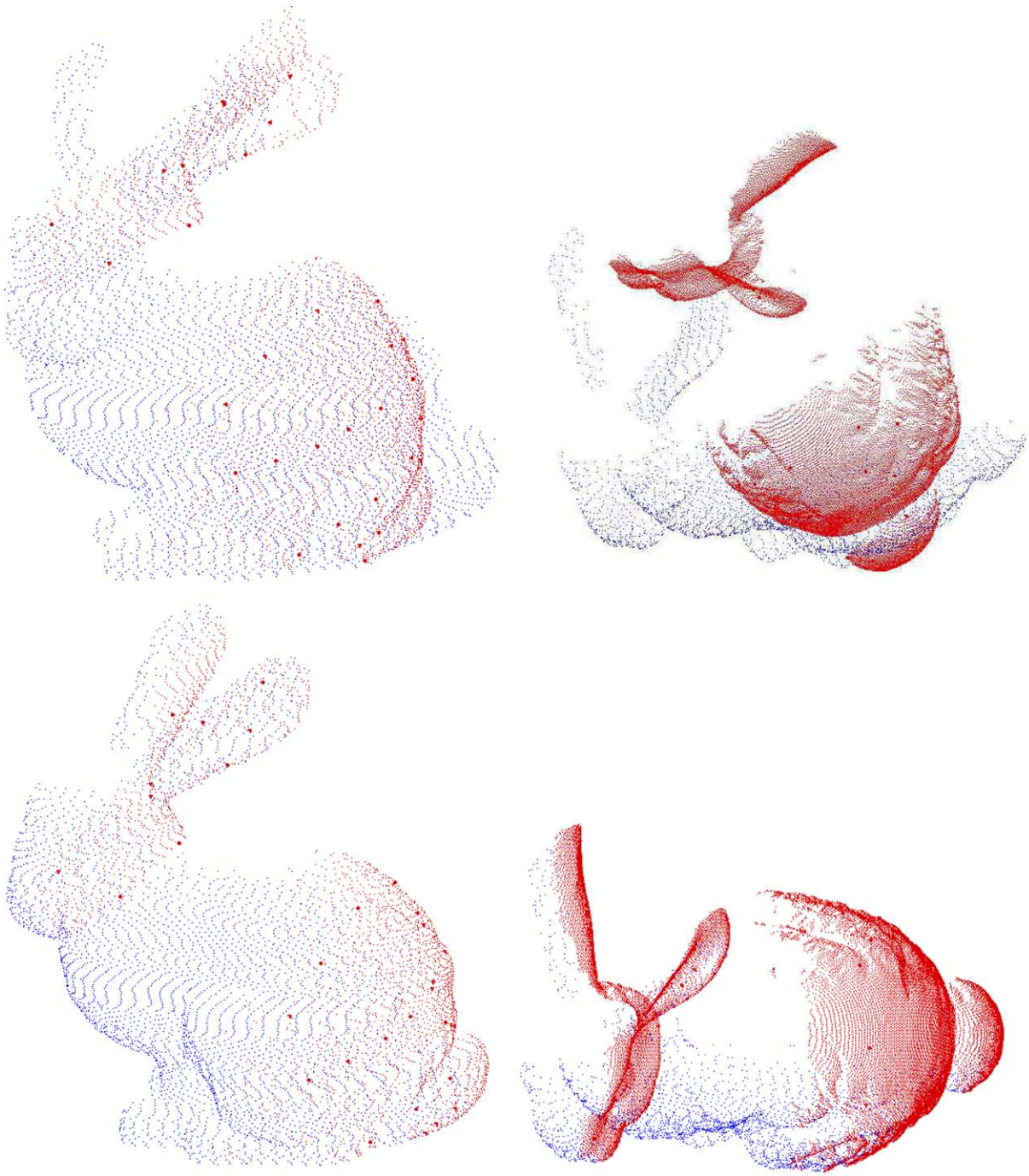

Fig. 6. 3D example. Top: initial alignment for 3D registration. Bottom: LM–Huber optimum.

$T(\theta, t_x, t_y; x, y)$ from Eq. (1). Then $\nabla_{\mathbf{a}} T(\mathbf{a}; (x_i, y_i))$ is the 3×2 matrix

$$\mathbf{M} = \begin{pmatrix} -x_i \sin\theta + y_i \cos\theta & -x_i \cos\theta - y_i \sin\theta \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{12}$$

and the gradient vector $\nabla_{\mathbf{a}} E(\mathbf{a})$ becomes (with $\mathbf{d}_i = (x_i, y_i)$)

$$\nabla_{\mathbf{a}} E(\theta, t_x, t_y) = \sum_{i=1}^{N_d} w_i \mathbf{M} \begin{pmatrix} D_x(x_i \cos\theta + y_i \sin\theta + t_x, -x_i \sin\theta + y_i \cos\theta + t_y) \\ D_y(x_i \cos\theta + y_i \sin\theta + t_x, -x_i \sin\theta + y_i \cos\theta + t_y) \end{pmatrix}$$

Implementation of 2D LM-ICP bilinearly interpolating a discrete distance transform reduced the elapsed time for

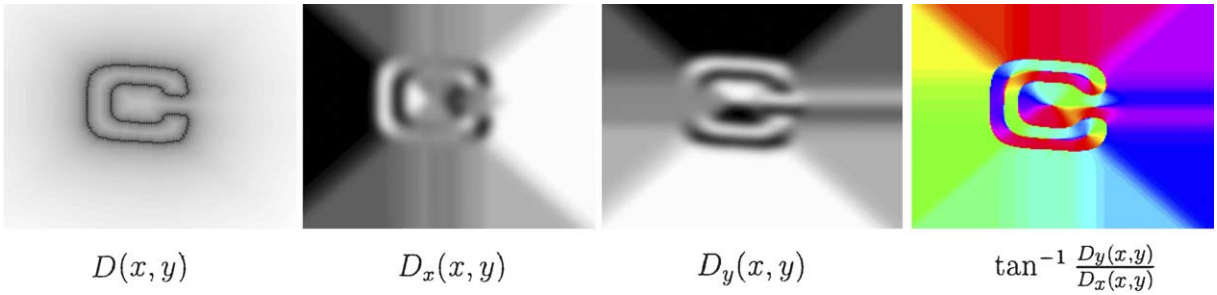

Fig. 7. Distance transform $D(x,y)$ and derivatives. Precomputing D and $\nabla_{\mathbf{x}}D$ allows fast computation of the error $E(\mathbf{a})$ and its derivatives $\nabla_{\mathbf{a}}E$ within the Levenberg–Marquardt iteration.

registration (the average described in Fig. 4) from 13 to 0.37 s (MATLAB on a 650 MHz Pentium III). Traditional ICP can benefit from the distance transform by storing, at each point, the integer label of the closest point rather than the point itself. This makes each iteration of ICP almost exactly the same cost as those of LM-ICP—the time taken to compute T is similar to the cost of the LM update. However, in this case, the LM algorithm's superior convergence means the overall runtime is reduced.

7. Discussion

I propose that point-set registration is better performed using a general-purpose non-linear optimization procedure than via the popular ICP algorithm. The general-purpose routine is faster, and much simpler to program. Because it is simpler to program, it may be enhanced to incorporate robust estimation, without loss in speed. In contrast, standard ICP suffers a significant speed penalty when robust metrics are introduced [24]. In fact, standard ICP cannot minimize a robust kernel unless an iterative approach is used in the T step. This paper shows that pulling the iteration outside both C and T steps leads to a faster algorithm.

It can be shown, although it is omitted here, that LM-ICP must require, at worst, p times as many function evaluations as regular ICP. In practice this limit was never met.

The more general conclusion of this work is that specialized algorithms such as ICP are not always to be preferred to general-purpose techniques. This is true in this paper, and concurs with similar observations which have been made in neural network learning [18], curve fitting [10] and photogrammetry [23]. MATLAB source code for the algorithm is available from the author's website [29].

Acknowledgements

I would like to thank David Capel, Frederik Schaffalitzky, and Andrew Zisserman for discussions relating to this work, and the Royal Society for its generous funding. The reviewers also made some excellent suggestions.

References

[1] P.J. Besl, N. McKay, A method for registration of 3-D shapes, IEEE Transactions on Pattern Analysis and Machine Intelligence 14 (2) (1992) 239–256.
[2] G. Borgefors, Hierarchical chamfer matching: a parametric edge matching algorithm, IEEE Transactions on Pattern Analysis and Machine Intelligence 10 (6) (1988) 849–865.
[3] R.P. Brent, Algorithms For Minimization Without Derivatives, Dover, New York, 2002.
[4] G. Champleboux, S. Lavallée, R. Szeliski, L. Brunie, From accurate range imaging sensor calibration to accurate model-based 3D object localization, In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1992, pp. 83–89.
[5] Y. Chen, G. Medioni, Object modelling by registration of multiple range images, Image and Vision Computing 10 (3) (1992) 145–155.
[6] H. Chui, A. Rangarajan, A new algorithm for non-rigid point matching, In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2000, pp. 40–51.
[7] S.J. Cunnington, A.J. Stoddart, N-view point set registration: a comparison, In: Proceedings of the British Machine Vision Conference, 1999, pp. 234–244.
[8] D. Eggert, A.W. Fitzgibbon, R.B. Fisher, Simultaneous registration of multiple range views satisfying global consistency constraints for use in reverse engineering, Computer Vision and Image Understanding 69 (3) (1998) 253–272.
[9] J. Feldmar, N. Ayache, Rigid and affine registration of smooth surfaces using differential properties, In: Proceedings of Third European Conference on Computer Vision, 1994, pp. 397–406.
[10] W. Gander, G.H. Golub, R. Strebel, Least-square fitting of circles and ellipses, BIT 43 (1994) 558–578.
[11] W.E.L. Grimson, T. Lozano-Pérez, W. Wells, G. Ettinger, S. White, R. Kikinis, An automatic registration method for frameless stereotaxy, image-guided surgery, and enhanced reality visualization, In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1994, pp. 430–436.
[12] J.P. Hampel, E.M. Ronchetti, P.J. Rousseeuw, W.A. Stahel, Robust Statistics: The Approach Based on Influence Functions, Wiley, New York, 1986.
[13] R.I. Hartley, Euclidean reconstruction from uncalibrated views, in: J. Mundy, A. Zisserman, D. Forsyth (Eds.), Applications of Invariance in Computer Vision, LNCS 825, Springer, Berlin, 1994, pp. 237–256.
[14] A. Hilton, A.J. Stoddart, J. Illingworth, T. Windeatt, Marching triangles: range image fusion from complex object modelling, In: Proceedings of the International Conference on Pattern Recognition, vol. 2, 1996, pp. 381–384.
[15] A. Johnson, S. Kange, Registration and Integration of textured 3D data, In: Proceedings of 3DIM'97, Ottawa, 1997, pp. 234–241.
[16] J.E. Dennis Jr., D.M. Gay, R.E. Welsch, Algorithm 573: NL2SOL—an adaptive nonlinear least-squares algorithm, ACM TOMS 7 (1981) 369–383.

[17] B. Luo, E.R. Hancock, Matching point-sets using procrustes alignment and the EM algorithm, In: Proceedings of the 10th British Machine Vision Conference, Nottingham, 1999, pp. 43–52.
[18] T. Masters, Advanced Algorithms for Neural Networks: A C++ Sourcebook, Wiley, New York, 1995.
[19] W. Press, S. Teukolsky, W. Vetterling, B. Flannery, Numerical Recipes in C, Second ed., Cambridge University Press, Cambridge, 1992.
[20] D.A. Simon, M. Hebert, T. Kanade, Techniques for fast and accurate intra-surgical registration, Journal of Image Guided Surgery 1 (1) (1995) 17–29.
[21] A.J. Stoddart, A. Hilton, Registration of multiple point sets, In: Proceedings of the International Conference on Pattern Recognition, 1996, pp. 40–44.
[22] C. Studholme, D. Hill, D. Hawkes, Automated 3D registration of truncated MR and CT images of the head, In: Proceedings of the Sixth British Machine Vision Conference, Birmingham, 1995, pp. 27–36.
[23] W. Triggs, P. McLauchlan, R. Hartley, A. Fitzgibbon, Bundle adjustment: a modern synthesis, in: W. Triggs, A. Zisserman, R. Szeliski (Eds.), Vision Algorithms: Theory and Practice, LNCS, Springer, Berlin, 2000.
[24] E. Trucco, A. Fusiello, V. Roberto, Robust motion and correspondence of noisy 3D point sets with missing data, Pattern Recognition Letters 20 (1999) 889–898.
[25] G. Turk, M. Levoy, Zippered polygon meshes from range images, In: Proceedings of the ACM SIGGRAPH Conference on Computer Graphics, 1994, pp. 311–318.
[26] P. Viola. Alignment by maximization of mutual information. PhD Thesis, MIT AI Lab, Technical Report AITR 1548, June 1995.
[27] W. Wells, Statistical approaches to feature-based object recognition, International Journal of Computer Vision 21 (1997) 63–98.
[28] Z. Zhang, On local matching of free-form curves, In: Proceedings of the British Machine Vision Conferences, 1992, pp. 347–356.
[29] http://www.robots.ox.ac.uk/~awf/lmicp