DOI: 10.1111/cgf.13587

COMPUTER GRAPHICS *forum*
*Volume 00 (2018), number 00 pp. 1–25*

# A Variational Approach to Registration with Local Exponential Coordinates

Ashish Paman and Ramsharan Rangarajan

Department of Mechanical Engineering, Indian Institute of Science Bangalore, Bengaluru, India
ashish.paman@gmail.com, rram@iisc.ac.in

**Abstract**

*We identify a novel parameterization for the group of finite rotations* ($SO_3$)*, consisting of an atlas of exponential maps defined over local tangent planes, for the purpose of computing isometric transformations in registration problems that arise in machine vision applications. Together with a simple representation for translations, the resulting system of coordinates for rigid body motions is proper, free from singularities, is unrestricted in the magnitude of motions that can be represented and poses no difficulties in computer implementations despite their multi-chart nature. Crucially, such a parameterization helps to admit varied types of data sets, to adopt data-dependent error functionals for registration, seamlessly bridges correspondence and pose calculations, and inspires systematic variational procedures for computing optimal solutions. As a representative problem, we consider that of registering point clouds onto implicit surfaces without introducing any discretization of the latter. We derive coordinate-free stationarity conditions, compute consistent linearizations, provide algorithms to compute optimal solutions and examine their performance with detailed examples. The algorithm generalizes naturally to registering curves and surfaces onto implicit manifolds, is directly adaptable to handle the familiar problem of pairwise registration of point clouds and allows for incorporating scale factors during registration.*

**Keywords:** computer vision - tracking, methods and applications, implicit surfaces, modelling, computational geometry

**ACM CCS:** • Computing methodologies → Computer vision

## 1. Introduction

The registration problem in machine vision consists of placing data sets in a common frame of reference by estimating the transformations between them. It arises in varied contexts, including photogrammetry, robotics, motion sensing and object detection. It is perhaps most commonly encountered as the 'scan matching' problem, wherein clouds of point samples from a surface are obtained from multiple views, each in its own system of coordinates. This is a recurring issue in surface scanners since multiple partial reconstructions are required to compensate for the limited field of view. The scan matching problem also manifests when merging point clouds determined using different scanning modalities. In general though, data sets to be registered need not consist of just point clouds; they may also include curves, surfaces and depth maps. During component inspection, for example, the computed aided design (CAD) geometry and the measured data (points, curves and surfaces) need to be mapped to a common coordinate system for comparison and to detect manufacturing errors. Just as with data sets, the class of transformations permitted during registration can be varied as well. Throughout this paper, we will be exclusively concerned with transformations that are isometries, i.e. motions comprised of a rigid body rotation, a translation and possibly an isotropic scale factor.

**Key ideas.** A principal unknown in the rigid registration problem is a rotation matrix. The challenge in computing it is made apparent by the observation that the set of rotations ($SO_3$) lacks a linear structure due to the constraints imposed by orthogonality. Noting that $SO_3$ is a three-dimensional manifold, the lack of linearity is not a hindrance since computations can be pulled back to a parametric domain instead. Rather, the challenge lies in choosing a proper system of coordinates [Stu64, SA91, ELF97]. This question of how to parameterize rotations in registration problems is indeed one of our main concerns here. Our task is confounded by the fact that it is impossible to identify a singularity-free three

degree of freedom(dof) parameterization for $SO_3$ . The singularities encountered at the poles when representing rotations using Euler angles demonstrate precisely this fact. If an improper set of coordinates can be admitted, the task is drastically simplified. For instance, 4-parameter quaternions furnish a robust parameterization, facilitate simple rules for composing rotations and enable compact storage for rotation fields in computer implementations [FH86, Hor87]. Alternately, viewing $SO_3$ as a constrained subset of the vector space of affine transformations, we can simply pick the nine components of a rotation matrix (say in a Cartesian basis) as its degrees of freedom and augment these with Lagrange multipliers to enforce the orthogonality conditions. This was in fact the approach followed in [Kab76] to lay the foundations for the iterative closest point (ICP) algorithm. Unlike the parameterizations mentioned above, exponential coordinates represent a canonical choice for $SO_3$. The idea behind their definition lies in *locally* parameterizing the manifold over tangent planes using the exponential map (Exp). Recall that the tangent plane to $SO_3$ at an arbitrary rotation is isomorphic to the set of skew-symmetric matrices, and hence to $\mathbb{R}^3$ itself. Geometrically speaking, the three dofs of a local exponential coordinate system over $SO_3$ are the Cartesian coordinates of an axial vector defining an infinitesimal rotation at a given point on the manifold. Then, the exponential map naturally defines an association from the tangent plane to the manifold, which in this case, is from the Lie algebra $\mathfrak{so}_3$ to the Lie group $SO_3$ [GX03, Gal11]. Unlike with general manifolds, we are also afforded the convenience of being able to easily compute this exponential map. At the identity rotation, for example, Exp is simply the matrix exponential defined over the set of skew matrices and is given by what is commonly referred to as the Rodriguez formula [Gra98, Ead17]. It is, however, important to recognize that the chart $\text{Exp} : \mathfrak{so}_3 \to SO_3$ is only local, since the restriction of Exp to $\mathfrak{so}_3$ fails to be injective when the magnitude of incremental rotations exceeds $180^\circ$. For this reason perhaps, exponential coordinates are often discarded in machine vision and robotics applications in favour of parameterizations using Euler angles and quaternions [SA91].

It may appear that the restricted domain for the chart defined by Exp effectively limits rotation angles to $180^\circ$. This limitation is only illusory and can be overcome by explicitly constructing an atlas for $SO_3$ using local charts induced at each rotation. In contrast, such atlas constructions appear to be impractical with other 3-dof parameterizations for $SO_3$. Specifically in the context of resolving registration problems, injectivity of the local exponential map only stipulates that rotation increments be smaller than $180^\circ$ at each solution iteration. Crucially, there is no restriction on the cumulative rotation computed over multiple iterations — rotations exceeding $180^\circ$ can certainly be computed, but just not over a single iteration/increment. For these reasons, local exponential coordinates are an ideal choice for resolving registration problems. In the exposition that follows, we find that the choice of these coordinates proffers benefits beyond just parameterizing rigid body transformations — they help to derive *coordinate-free* stationarity conditions satisfied by the solution using variational procedures and afford convenient algorithms for numerically approximating these stationary points.

**Related literature.** The point of departure when discussing methods for resolving registration problems is of course the ICP algorithm. Although numerous alternatives have been proposed, ICP is the method of choice for registering pairs of point cloud data sets [BSB*15]. As its name suggests, the algorithm is iterative and consists of computing point correspondences between data sets, followed by a pose estimation step. At each iteration, closest pairs of points in the two data sets are deemed to correspond, based on either point-to-point or point-to-plane distance metrics [CM92]. With the error functional defined as the sum of squared distances between corresponding pairs of points in centroid-aligned clouds, the optimal rotation is computed using the singular value decomposition of the cross-correlation matrix [AHB87, Ume91, BM*92, Kan94]. Perhaps the most appealing feature of ICP is the fact that the optimal pose at each iteration is explicitly computable. This aspect of the algorithm is routinely exploited in applications demanding fast and efficient registrations [RL01].

Shortcomings of the ICP algorithm are well documented in the literature. As a first among these, we mention one that also happens to be a defining feature of the algorithm — the two-step paradigm in the algorithm explicitly precludes the possibility of simultaneously altering correspondences and pose. This deficiency is also common to its numerous variants [RL01, SHT09], and is responsible for these algorithms occasionally identifying incorrect solutions (suboptimal local minimizers). The algorithms we propose in Sections 3 and 4 for registration onto implicit surfaces directly address this shortcoming by seamlessly bridging correspondence and pose calculations. Direct optimization-based approaches are also a promising alternative to ICP-based algorithms in this regard [Fit03, MGPG04].

Recent work on registration algorithms overwhelmingly falls in the category of probabilistic methods, which help to address uncertainties in the registration process not considered in ICP — both in measurements (e.g. noise) as well as in the computed transformation parameters and correspondences. A common theme underlying many such methods consists of replacing point clouds by Gaussian mixture models (GMMs) and computing transformations that fit GMM centroids to the target data set by maximizing the likelihood [JV05]. Notably, the binary assignment of correspondences in ICP-type algorithms is replaced by the idea of assigning correspondences with probability density functions. Various refinements of this idea meant towards improving robustness of the registration procedure have been proposed. The coherent point drift algorithm constrains GMM centroids to transform as a group in order to preserve the topological structure of point sets [MS10]. Permitting the variances of GMMs to be an additional parameter during registration leads to a multiscale scheme [GP02], since variances can be interpreted as scale/blurring factors. The kernel correlation algorithm constructs probability distributions for each data set and measures the dissimilarity between them to achieve registration [TK04]. The assumption of isotropic covariances is relaxed in [HFY*11] by permitting the use of general covariance matrices for mixture model components. Robust point matching algorithms [CRZL04] consider all possible correspondences and iterate towards uniquating matches, and can therefore succeed in registering data sets for which ICP fails. The covariance-driven correspondence algorithm of [SYS07] estimates the uncertainty in point correspondences to improve the robustness of the registration. We mention that many of the ideas mentioned above are also applicable in the general context of non-rigid registration, where a suitable model for the type of admissible deformations is additionally required [CR03, MZJZ17, TCL*13, MZY16].

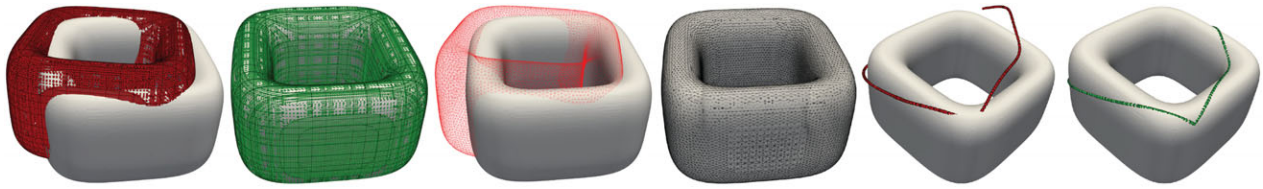


**Figure 1:** *We propose novel registration algorithms that adopt a variational approach and local exponential map-based coordinates to achieve a unified framework for registration of point clouds, (triangulated) surfaces and curves onto implicitly defined manifolds.*

The fuzzy nature of correspondences inherent in the probabilistic methods mentioned above is also common to soft assignment methods, which construct a correspondence matrix while imposing row and column constraints [RCB97, CR03, Liu07, KSS*17]. Incorporating structural information is generally recognized to help improve the robustness and accuracy of registration. The notion of a shape context proposed in [BMP02, MQZ*15], while directly relevant for registration, is more generally useful in applications involving shape recognition. In [MQZ*15], local structures in point clouds are exploited to improve probabilities assigned in mixture models. A point-matching algorithm based on preserving local neighbourhood structures with a graph-theoretic interpretation can be found in [ZD06]. The recent work in [MZJZ17] illustrates the possibility of using intrinsic geometric information in mixture models to achieve robust non-rigid registration despite the presence of outliers.

Integral to the success of most registration algorithms, including ones proposed here and those mentioned above, is the requirement that data sets being registered be sufficiently close (in an appropriate distance metric). Hence, coarse registration based on landmarks, feature points or background markers is generally a prerequisite for successful fine registration procedures. In this context, schemes designed to sample the solution space play a crucial role in automating coarse registration. For instance, the idea of extended Gaussian images in [MPD06] and the branch-and-bound search scheme in [YLJ13] search the space of isometric transformations and therefore help to identify globally optimal solutions despite large discrepancies in initial alignment. Accounting for partial overlaps and outliers is also crucial during registration. Rather than relying on tolerance-sensitive and heuristic criteria for rejecting point-pairings, methods that sample numerous possible correspondences help to identify outliers and partial overlaps more robustly [TFR99, CSK05].

Perhaps the closest approach to ours, which exploits the structure of the constraint manifold of rigid body motions for registration, is the work in [KLMV05]. Therein, the problem of simultaneously registering multiple point clouds with *known* correspondences is considered (cf. [WB01]), and a construction of local charts using exponential coordinates for $SO_3$ plays a key role. The algorithm proposed there consists of first lifting the registration problem to the affine tangent space at the current configuration (a set of poses), identifying a solution iterate in the tangent space and subsequently projecting back to the constraint manifold with the local exponential map. In contrast, we resolve the registration problem directly on the constraint manifold by explicitly constructing paths on it and hence bypass the lift-optimize-project paradigm, while conspicuously avoiding any exiguous algebraic additions/subtractions of rotations. Besides, the variational approach proposed and consistent linearizations derived here yield unambiguously more general algorithms, as evidenced by the algorithms for registering varied data sets (point clouds, curves and surface) to implicit surfaces without assuming any correspondences *a priori*, see Figure 1.

A loose analogy between rigid body motions and the registration problem appears in the work in [PLH04], where the notion of 'instantaneous kinematics' is introduced to compute incremental helical motions to achieve registration. The explicit requirement of the squared distance to the target surface, the ensuing quadratic approximation of squared distances and the linearized kinematics yielding helical motions are all key aspects of the algorithm in [PLH04] which are completely irrelevant in our approach. In fact, none of the linearizations appearing in [PLH04] are employed here. Instead, we directly pose the registration problem on the manifold of isometric transformations.

The group theoretic and differential geometric ideas underlying our work here are also generally well adopted in computer graphics applications related to constructing paths interpolating rigid body transformations [Ale02]. Indeed, numerous algorithms for constructing interpolants in Lie groups have been proposed, with applications ranging from synthesizing animations in computer graphics or path planning in robotics [ŽK98, Agr06], to designing integrators for ordinary differential equations [Mar99]. A recent application exploiting the recognition of rigid body motions as a Lie group to detect symmetries in data sets can be found in [SAD*16]. As a particular example of the potency of adopting canonical coordinates for problems posed in Lie groups, we highlight an analysis of the motion and structure recovery problem arising in computer vision [MKS01], where the parameterization of the essential manifold (closely related to $SO_3$) plays a key role. The multi-chart atlas for $SO_3$ adopted here shares many features with methods used for describing motions in rigid body mechanics [SW91] and for describing the kinematics of director fields in rod and shell theories [SVQ88, SF89].

Besides the choice of coordinates, an important ingredient in our work are algorithms for optimization problems posed on manifolds [AMS10]. Steepest descent and Newton-type algorithms are particularly relevant to the registration problem we study, as are algorithms designed for Stiefel manifolds [EAS98] since the orthogonal group can be interpreted as a special case of it.

**Contributions.** We concisely list our key contributions in this paper.

(i) We formulate the problem of pairwise rigid registration of data sets as an optimization problem posed over the constrained manifold of rigid body motions.

(ii) We adopt a proper system (6-dof) of local exponential coordinates for parameterizing the space of admissible solutions. Such a parameterization for isometries is independent of the representation chosen for data sets themselves, which enables us to register varied data sets (point clouds, curves and surfaces as depicted in Figure 1), facilitates flexible choices for the error metric being minimized and helps to naturally incorporate higher order information in the target data set (e.g. normals and curvatures). It is also straightforward to include scaling factors during registration by introducing an additional dof.

(iii) We exploit variational principles to derive concise stationarity conditions satisfied by the solution and detail consistent linearization procedures for iteratively computing optimizers using a general class of Newton-type nonlinear solvers. In particular, we demonstrate that realizing the idea of using an exponential map-based multi-chart atlas for $SO_3$ in the proposed variational framework poses no algorithmic or numerical difficulties.

(iv) Remarkably, in all the variants of registration problems studied here, stationarity conditions satisfied by solutions turn out to be coordinate-free expressions that completely disguise the underlying parameterizations for isometries chosen to derive them in the first place. These stationarity conditions can serve as a starting point in alternative registration algorithms using a completely different set of coordinates if desired. Furthermore, the stationarity conditions derived here do not seem to appear frequently in the literature presumably because they are obscured by the choice of coordinates and the presence of extraneous coordinate-specific constraints.

(v) It is worth noting that none of the registration problems considered in Sections 3 and 4 can be resolved with ICP, at least not without introducing some approximation/discretization of the data sets being registered.

(vi) We include numerous examples demonstrating the robustness of the algorithms proposed, provide detailed accounts of their performance and compare it with state-of-the-art algorithms for a representative set of examples.

**Organization.** We begin in Section 2 by providing a concise description of the exponential map-based atlas constructed over $SO_3$. Such a parameterization facilitates constructing 1-parameter families of rotations, i.e. curves on the constraint manifold which help to define variational procedures for resolving registration problems. Through the problem of registering a point cloud onto an implicit surface that is considered in detail in Section 3, we explain the key ingredients in our approach — defining an objective functional, deriving stationarity conditions satisfied by the optimal pose, computing consistent linearization of these conditions and employing them in a Newton algorithm to numerically approximate the solution. We provide numerous examples with the algorithm, examine its performance in detail and compare it with alternative registration algorithms. Generalizations of the algorithm to registering curves and surfaces onto implicit manifolds and a novel procedure to include scale factors during registration are discussed and illustrated with examples in Section 4. Section 5 is devoted to the case of pairwise registration of point clouds. We provide two approaches to this end. The first consists of approximating the target point cloud with an implicit surface, while the second is an ICP-type algorithm. Finally, we end with a few concluding remarks in Section 6.

## 2. A Parameterization for Finite Rotations

We begin by recalling a few geometric preliminaries that are directly relevant for our purposes and in the process, introduce the notation required in the remainder of this paper. The group of orthogonal transformations, given by

$$\mathrm{SO}_3 \triangleq \left\{ \mathbf{R} : \mathbb{R}^3 \to \mathbb{R}^3 \,|\, \mathbf{R}^t = \mathbf{R}^{-1} \quad \text{and} \quad \det \mathbf{R} = +1 \right\}, \tag{1}$$

is a set endowed with a manifold structure. Its tangent space at the identity element ($\mathbf{I}$) is the set of infinitesimal rotations

$$\mathrm{T}_{\mathbf{I}}\mathrm{SO}_3 \triangleq \left\{ \mathbf{W} : \mathbb{R}^3 \to \mathbb{R}^3 \,|\, \mathbf{W}^t = -\mathbf{W} \right\}, \tag{2}$$

also identified as the Lie algebra $\mathfrak{so}_3$ corresponding to the Lie group $\mathrm{SO}_3$. At an arbitrary element $\mathbf{R} \in \mathrm{SO}_3$, the tangent space is simply a rotated version of $\mathrm{T}_{\mathbf{I}}\mathrm{SO}_3$, i.e.

$$\mathrm{T}_{\mathbf{R}}\mathrm{SO}_3 \triangleq \left\{ \mathbf{W}\mathbf{R} \,|\, \mathbf{R}\mathbf{W}\mathbf{R}^t \in \mathrm{T}_{\mathbf{I}}\mathrm{SO}_3 \right\}. \tag{3}$$

Equation (3) is often referred to as the right translation of $\mathfrak{so}_3$.

Recognizing the isomorphism between $\mathfrak{so}_3$ and $\mathbb{R}^3$, we define the hat operator $\hat{}\,: \mathbb{R}^3 \to \mathfrak{so}_3$ as

$$\boldsymbol{\Theta} \in \mathbb{R}^3, \;\; \hat{\boldsymbol{\Theta}}\,\mathbf{v} \triangleq \boldsymbol{\Theta} \times \mathbf{v} \qquad \text{for each } \mathbf{v} \in \mathbb{R}^3, \tag{4}$$

where $\times$ is the usual cross product. In Equation (4), we interpret $\boldsymbol{\Theta}$ to be the axial vector of the infinitesimal rotation $\hat{\boldsymbol{\Theta}}$. Throughout the remainder of this paper, we will use this identification between $\boldsymbol{\Theta}$ and $\hat{\boldsymbol{\Theta}}$ without explicit mention, and for convenience, also set $\theta \triangleq \|\boldsymbol{\Theta}\|$. Ignoring the distinction between vectors/transformations and their corresponding matrix representations in a Cartesian basis, Equation (4) can be expressed in component form as

$$\boldsymbol{\Theta} \in \mathbb{R}^3, \boldsymbol{\Theta} = \begin{bmatrix} \Theta_1 \\ \Theta_2 \\ \Theta_3 \end{bmatrix} \;\Rightarrow\; \hat{\boldsymbol{\Theta}} \triangleq \begin{bmatrix} 0 & -\Theta_3 & \Theta_2 \\ \Theta_3 & 0 & -\Theta_1 \\ -\Theta_2 & \Theta_1 & 0 \end{bmatrix}. \tag{5}$$

The exponential of a skew matrix will be central to our presentation. Given $\hat{\boldsymbol{\Theta}} \in \mathfrak{so}_3$, we have the Rodriguez formula

$$\mathrm{Exp}[\hat{\boldsymbol{\Theta}}] \triangleq \mathbf{I} + \frac{\sin\theta}{\theta}\hat{\boldsymbol{\Theta}} + \frac{(1-\cos\theta)}{\theta^2}\boldsymbol{\Theta} \otimes \boldsymbol{\Theta}, \tag{6}$$

where $\otimes$ denotes the dyadic product. Equation (6) is straightforward to prove, for instance, by using a series expansion for the matrix exponential and noting the relationships

$$\hat{\boldsymbol{\Theta}}^2 = \boldsymbol{\Theta} \otimes \boldsymbol{\Theta} - \theta^2\mathbf{I} \quad \text{and} \quad \hat{\boldsymbol{\Theta}}^3 = -\theta^2\hat{\boldsymbol{\Theta}}$$

when computing higher powers of $\hat{\boldsymbol{\Theta}}$, see [MLSS94]. While Equation (6) defines the exponential map for $\mathrm{SO}_3$ at the identity, the mapping $\mathrm{Exp}_{\mathbf{R}} : \mathrm{T}_{\mathbf{R}}\mathrm{SO}_3 \to \mathrm{SO}_3$ over the tangent space at an arbitrary $\mathbf{R} \in \mathrm{SO}_3$ now follows as

$$\mathrm{Exp}_{\mathbf{R}}[\hat{\boldsymbol{\Theta}}] \triangleq \mathrm{Exp}[\hat{\boldsymbol{\Theta}}]\mathbf{R} \qquad \text{for each } \hat{\boldsymbol{\Theta}} \in \mathfrak{so}_3. \tag{7}$$

Strictly speaking, the argument for $\mathrm{Exp}_{\mathbf{R}}[\cdot]$ should be an element in $\mathrm{T}_{\mathbf{R}}\mathrm{SO}_3$; we nevertheless permit the abuse of notation in Equation (7)

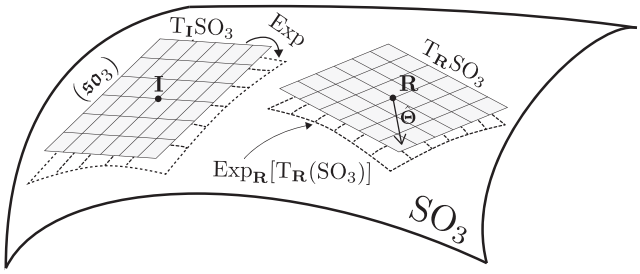



**Figure 2:** *A conceptual illustration of the multi-chart parameterization for* $SO_3$ *over local tangent planes with the exponential map.*

owing to the simple relationship between $\mathfrak{so}_3$ and $T_{\mathbf{R}}SO_3$ evident in Equation (3). It is straightforward to verify that the mapping in Equation (7) is surjective and that it is injective only for $\theta$ smaller than $\pi$. For this reason, $\mathrm{Exp}_{\mathbf{R}}$ only defines a local chart for $SO_3$ at $\mathbf{R}$ over $T_{\mathbf{R}}SO_3$, see Figure 2. The local parameterization for $SO_3$ induced by Equation (7) will be the foundation of our registration algorithms.

With the local parameterization $\hat{\mathbf{\Theta}} \mapsto \mathrm{Exp}_{\mathbf{R}}[\hat{\mathbf{\Theta}}]$ at hand, we can now define 1-parameter families of rotations for each $\mathbf{R} \in SO_3$ and $\hat{\mathbf{\Theta}} \in \mathfrak{so}_3$, as

$$\mathbb{R} \ni \varepsilon \mapsto \mathbf{R}_\varepsilon \triangleq \mathrm{Exp}_{\mathbf{R}}[\varepsilon\hat{\mathbf{\Theta}}]. \tag{8}$$

Note that $\mathbf{R}_\varepsilon$ satisfies

$$\lim_{\varepsilon\to 0} \mathbf{R}_\varepsilon = \mathbf{R} \quad \text{and} \quad \lim_{\varepsilon\to 0} \frac{d\mathbf{R}_\varepsilon}{d\varepsilon} = \hat{\mathbf{\Theta}}\mathbf{R}. \tag{9}$$

In particular, Equation (9) shows that the curve $\varepsilon \mapsto \mathbf{R}_\varepsilon$ passes through $\mathbf{R}$ at $\varepsilon = 0$ and its tangent there coincides with the infinitesimal rotation $\hat{\mathbf{\Theta}}\mathbf{R} \in T_{\mathbf{R}}SO_3$. We will see next that this observation enables us to explore admissible configurations in the immediate neighbourhood of $\mathbf{R}$ and hence to systematically construct variational procedures to solve registration problems.

## 3. Registering Point Clouds to Implicit Surfaces

As a first problem, we consider registering a point cloud $\mathbf{U} = \{\mathbf{u}_i\}_{i=1}^n$ to an implicitly defined surface $\mathcal{S}$. Such a scenario is commonly encountered, for instance, when registering scanned point clouds from manufactured products to their model CAD surfaces during quality inspections. We will discuss this registration problem in some detail since it serves as a prototype for further generalizations examined in subsequent sections.

By saying that $\mathcal{S}$ is implicitly defined, we identify $\mathcal{S}$ as the zero-level set of a twice continuously differentiable function $\psi : \mathbb{R}^3 \to \mathbb{R}$, so that

$$\mathcal{S} \triangleq \{\mathbf{x} \in \mathbb{R}^3 \,:\, \psi(\mathbf{x}) = 0,\ \|\nabla\psi\| \neq 0\}. \tag{10}$$

Registering $\mathbf{U}$ to $\mathcal{S}$ consists of finding $(\mathbf{R}, \mathbf{t}) \in SO_3 \times \mathbb{R}^3$ that maps $\mathbf{U}$ onto $\mathcal{S}$ in some optimal sense. Here, we pose this question as a minimization problem:

$$\text{Find} \quad (\mathbf{R}, \mathbf{t}) \triangleq arg\,min_{(\mathbf{Q},\mathbf{s})\in SO_3\times\mathbb{R}^3} \mathrm{J_p}(\mathbf{Q}, \mathbf{s}), \tag{11}$$

where $\mathrm{J_p} : SO_3 \times \mathbb{R}^3 \to \mathbb{R}$ is the functional

$$\mathrm{J_p}(\mathbf{Q}, \mathbf{s}) \triangleq \frac{1}{2}\sum_{i=1}^{n} \psi^2(\mathbf{Q}\mathbf{u}_i + \mathbf{s}).$$

The choice of the error functional $\mathrm{J_p}$ is quite natural. It is motivated by the fact that when $\mathbf{U}$ is perfectly registered, then $\psi(\mathbf{R}\mathbf{u}_i + \mathbf{t}) = 0$ for each $i = 1, \ldots, n$ and consequently, $\mathrm{J_p} = 0$ as well.

### 3.1. Stationarity conditions

In general, it is not possible to compute closed form solutions for $(\mathbf{R}, \mathbf{t})$ in Equation (11). Instead, our first objective is to deduce conditions satisfied by a minimizer of $\mathrm{J_p}$. For this purpose, we adopt a variational approach and request stationarity of $\mathrm{J_p}$ with respect to all admissible variations of the solution $(\mathbf{R}, \mathbf{t})$:

$$\begin{aligned}\langle \delta\mathrm{J_p}(\mathbf{R}, \mathbf{t}), (\hat{\mathbf{\Theta}}, \mathbf{w})\rangle &\triangleq \lim_{\varepsilon\to 0} \frac{d}{d\varepsilon}\mathrm{J_p}(\mathrm{Exp}_{\mathbf{R}}[\varepsilon\hat{\mathbf{\Theta}}], \mathbf{t} + \varepsilon\mathbf{w}) \\ &= 0 \quad \forall(\hat{\mathbf{\Theta}}, \mathbf{w}) \in \mathfrak{so}_3 \times \mathbb{R}^3. \end{aligned} \tag{12}$$

In stating Equation (12), we have exploited the fact that since $\mathfrak{so}_3 \ni \hat{\mathbf{\Theta}} \mapsto \mathrm{Exp}_{\mathbf{R}}[\hat{\mathbf{\Theta}}]$ is local chart for $SO_3$ at $\mathbf{R}$, any admissible configuration in the neighbourhood of $\mathbf{R}$ is necessarily of the form $\mathrm{Exp}_{\mathbf{R}}[\varepsilon\hat{\mathbf{\Theta}}]$ for some $\mathbf{\Theta} \in \mathfrak{so}_3$ and some $\varepsilon \in \mathbb{R}$ that is presumed to be small.

Since variations $\hat{\mathbf{\Theta}}$ and $\mathbf{w}$ can be chosen independently of each other, Equation (12) is equivalent to the pair of conditions

$$\langle \delta\mathrm{J}(\mathbf{R}, \mathbf{t}), \hat{\mathbf{\Theta}}\rangle \triangleq \lim_{\varepsilon\to 0} \frac{d}{d\varepsilon}\mathrm{J}(\mathrm{Exp}_{\mathbf{R}}[\varepsilon\hat{\mathbf{\Theta}}], \mathbf{t}) = 0 \quad \forall\,\hat{\mathbf{\Theta}} \in \mathfrak{so}_3, \tag{13a}$$

$$\langle \delta\mathrm{J}(\mathbf{R}, \mathbf{t}), \mathbf{w}\rangle \triangleq \lim_{\varepsilon\to 0} \frac{d}{d\varepsilon}\mathrm{J}(\mathbf{R}, \mathbf{t} + \varepsilon\mathbf{w}) = 0 \quad \forall\mathbf{w} \in \mathbb{R}^3, \tag{13b}$$

which result from setting $\mathbf{w} = 0$ and $\hat{\mathbf{\Theta}} = 0$, respectively, in Equation (12). We understand Equations (13a) and (13b) as conditions defining stationarity of the error functional with respect to incremental rotations and incremental translations, respectively.

Following the calculations outlined in Proposition 1 that is stated and proved in Appendix A, we find that Equation (13) simplifies to the condition

$$\mathbf{r}(\mathbf{R}, \mathbf{t}) \triangleq \begin{bmatrix} \mathbf{r}_\theta(\mathbf{R}, \mathbf{t}) \\ \mathbf{r}_\mathbf{t}(\mathbf{R}, \mathbf{t}) \end{bmatrix} = \sum_{i=1}^{n} \begin{bmatrix} (\mathbf{R}\mathbf{u}_i) \times \psi\nabla\psi(\mathbf{R}\mathbf{u}_i + \mathbf{t}) \\ \psi\nabla\psi(\mathbf{R}\mathbf{u}_i + \mathbf{t}) \end{bmatrix} = 0. \tag{14}$$

A few remarks concerning Equation (14) are in order.

(i) Equation (14) is a set of six independent equations that suffice to determine the stationary point $(\mathbf{R}, \mathbf{t})$, provided that the data sets $\mathbf{U}$ and $\mathcal{S}$ satisfy certain non-degeneracy conditions.

(ii) Equation (14) is coordinate-free — it completely conceals the specific parameterization for $SO_3 \times \mathbb{R}^3$ used to derive it.

(iii) In addition to the functional form of Equation (14), nonlinearity in the stationarity conditions also stems from the underlying solution space $SO_3 \times \mathbb{R}^3$ being a manifold lacking a vector space structure. As we discuss next, it is

necessary to resort to a Newton algorithm to resolve these equations.

(iv) The form of the stationarity conditions in Equation (14) will be common to registration problems considered in subsequent sections as well, with differences only stemming from the specific definitions for the objective functional.

(v) In deriving Equation (14) in Proposition 1, we do not impose or compute any correspondences between the point cloud $\mathbf{U}$ and the surface $\mathcal{S}$.

(vi) It is in general not possible to condense out the translational dofs from Equation (14). This is in contrast to the scenario that arises in pairwise registration of point clouds, where the translation has the simple interpretation of serving to align the centroids of the two point clouds. In particular, Equation (14) is a fully coupled system of nonlinear algebraic equations for $(\mathbf{R}, \mathbf{t})$.

### 3.2. Consistent linearization of stationarity conditions

While the set of six conditions in Equation (14) can be resolved using any set of coordinates for $\mathbf{R}$, we adopt exponential coordinates for this purpose as well. This choice, together with the consistent linearization of the residual computed next, will be the basis for iterative nonlinear solvers to compute numerical approximations of the solution.

We linearize the residual $\mathbf{r}(\mathbf{R}, \mathbf{t})$ at the pose $(\mathbf{R}, \mathbf{t})$ along $(\hat{\boldsymbol{\Theta}}, \mathbf{w})$ as

$$\mathrm{L}_{(\mathbf{R},\mathbf{t})}\mathbf{r}(\hat{\boldsymbol{\Theta}}, \mathbf{w}) \triangleq \mathbf{r}(\mathbf{R}, \mathbf{t}) + \mathbf{K}(\mathbf{R}, \mathbf{t}) \begin{bmatrix} \hat{\boldsymbol{\Theta}} \\ \mathbf{w} \end{bmatrix}, \tag{15a}$$

$$\text{where} \quad \mathbf{K}(\mathbf{R}, \mathbf{t}) \begin{bmatrix} \hat{\boldsymbol{\Theta}} \\ \mathbf{w} \end{bmatrix} \triangleq \lim_{\varepsilon \to 0} \frac{d}{d\varepsilon} \mathbf{r}(\mathbf{R}_\varepsilon, \mathbf{t} + \varepsilon \mathbf{w}). \tag{15b}$$

We understand $\mathbf{K}(\mathbf{R}, \mathbf{t})\,[\hat{\boldsymbol{\Theta}}; \mathbf{w}]^t$ to be the directional derivative of $\mathbf{r}$ at $(\mathbf{R}, \mathbf{t})$ along $(\hat{\boldsymbol{\Theta}}, \mathbf{w}) \in \mathfrak{so}_3 \times \mathbb{R}^3$. The notation in Equation (15), which closely follows [MH94], ch. 4], emphasizes the fact that the directional derivative $\mathbf{K}(\mathbf{R}, \mathbf{t})\,[\hat{\boldsymbol{\Theta}}; \mathbf{w}]^t$ depends linearly on $\hat{\boldsymbol{\Theta}}$ and $\mathbf{w}$, and that $(\hat{\boldsymbol{\Theta}}, \mathbf{w}) \mapsto \mathrm{L}_{(\mathbf{R},\mathbf{t})}\mathbf{r}(\hat{\boldsymbol{\Theta}}, \mathbf{w})$ is an affine map over $\mathfrak{so}_3 \times \mathbb{R}^3$. Unlike the stationarity conditions, we find that the choice of coordinates for $\mathrm{SO}_3$ clearly manifests in the linearizations of $\mathbf{r}$. Following Proposition 2 proved in Appendix A, we find that $\mathbf{K}(\mathbf{R}, \mathbf{t})$ in Equation (15) is given by

$$\mathbf{K}(\mathbf{R}, \mathbf{t}) = \sum_{i=1}^{n} \begin{bmatrix} \mathbf{K}_i^{\theta\theta} & \mathbf{K}_i^{\theta\mathbf{t}} \\ \mathbf{K}_i^{\mathbf{t}\theta} & \mathbf{K}_i^{\mathbf{tt}} \end{bmatrix}_{6\times 6} \Bigg|_{\mathbf{R}\mathbf{u}_i + \mathbf{t}}, \tag{16}$$

$$\begin{aligned} \text{with:} \quad \mathbf{K}_i^{\mathbf{tt}} &= \nabla\psi \otimes \nabla\psi + \psi \nabla\nabla\psi, \\ \mathbf{K}_i^{\theta\mathbf{t}} &= \widehat{(\mathbf{R}\mathbf{u}_i)}\, \mathbf{K}_i^{\mathbf{tt}} \\ \mathbf{K}_i^{\mathbf{t}\theta} &= \left(\mathbf{K}_i^{\theta\mathbf{t}}\right)^t, \\ \text{and} \quad \mathbf{K}_i^{\theta\theta} &= \left(\widehat{(\psi \nabla\psi)} - \widehat{(\mathbf{R}\mathbf{u}_i)} \mathbf{K}_i^{\mathbf{tt}}\right) \widehat{(\mathbf{R}\mathbf{u}_i)}. \end{aligned}$$

The form of the residual $\mathbf{r}$ and the Jacobian $\mathbf{K}$ in Equations (14) and (16) reveals that they can be conveniently assembled by summing contributions from each point in $\mathbf{U}$, which can also be done in parallel if desired.

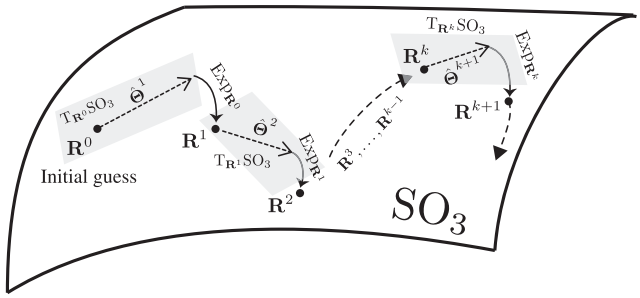



**Figure 3:** *An illustration depicting the evolution of solution iterates in Algorithm 1. At the $(k+1)^{th}$ iteration, the algorithm computes the increment $\boldsymbol{\Theta}^{k+1}$ belonging to the tangent plane at $\mathbf{R}^k$. The solution guess is then updated using the exponential map at $\mathbf{R}^k$.*

### 3.3. Iterative solution with a Newton algorithm

Let us suppose that we have an initial guess $(\mathbf{R}^0, \mathbf{t}^0)$ for the solution of Equation (14). Following a Newton algorithm, we set the linearized residual $\mathrm{L}_{(\mathbf{R}^0,\mathbf{t}^0)}(\hat{\boldsymbol{\Theta}}^1, \mathbf{w}^1)$ to zero, which yields a linear system of equations for the increment $(\boldsymbol{\Theta}^1, \mathbf{w}^1) \in \mathbb{R}^3 \times \mathbb{R}^3$. Specifically from Equation (15), we get

$$\underbrace{\mathbf{K}(\mathbf{R}^0, \mathbf{t}^0)}_{\mathbf{K}^0} \begin{bmatrix} \boldsymbol{\Theta}^1 \\ \mathbf{w}^1 \end{bmatrix} = -\underbrace{\mathbf{r}(\mathbf{R}^0, \mathbf{t}^0)}_{\mathbf{r}^0}, \tag{17}$$

where the Jacobian $\mathbf{K}^0$ is a $6 \times 6$ matrix and the residual $\mathbf{r}^0$ is a $6 \times 1$ column vector. With the increment $(\boldsymbol{\Theta}^1, \mathbf{w}^1)$ computed from Equation (17), we improve the guess for the solution as

$$\mathbf{R}^1 = \mathrm{Exp}_{\mathbf{R}^0}[\hat{\boldsymbol{\Theta}}^1] = \mathrm{Exp}[\hat{\boldsymbol{\Theta}}^1]\mathbf{R}^0 \quad \text{and} \quad \mathbf{t}^1 = \mathbf{t}^0 + \mathbf{w}^1.$$

Next, setting $\mathrm{L}_{(\mathbf{R}^1,\mathbf{t}^1)}\mathbf{r}(\hat{\boldsymbol{\Theta}}^2, \mathbf{w}^2) = 0$ yields the increment $(\boldsymbol{\Theta}^2, \mathbf{w}^2)$, and so on, see Figure 3. At the $(k+1)^{th}$ iteration, we do:

**Algorithm 1**

(i) *Assemble*: $\mathbf{K}^k = \mathbf{K}(\mathbf{R}^k, \mathbf{t}^k)$ and $\mathbf{r}^k = \mathbf{r}(\mathbf{R}^k, \mathbf{t}^k)$ using eqs. (14) and (16).

(ii) *Solve*: compute the increments $(\boldsymbol{\Theta}^{k+1}, \mathbf{w}^{k+1})$ satisfying

$$\mathbf{K}^k \begin{bmatrix} \boldsymbol{\Theta}^{k+1} \\ \mathbf{w}^{k+1} \end{bmatrix}_{6\times 1} = -\mathbf{r}^k.$$

(iii) *Update*: the solution to

$$\mathbf{R}^{k+1} = \mathrm{Exp}_{\mathbf{R}^k}[\boldsymbol{\Theta}^{k+1}] \text{ and } \mathbf{t}^{k+1} = \mathbf{t}^k + \mathbf{w}^{k+1}.$$

The above linearize-solve-update recipe can be repeated indefinitely, at least until a termination criterion is satisfied. In practice, we terminate the algorithm when the norm of the residual falls below a predefined tolerance.

### 3.4. A unified perspective on pose updates

The pose update

$$\mathbf{R}^{k+1} = \mathrm{Exp}_{\mathbf{R}^k}[\boldsymbol{\Theta}^{k+1}] \quad \text{and} \quad \mathbf{t}^{k+1} = \mathbf{t}^k + \mathbf{w}^{k+1} \tag{18}$$

at a generic $(k+1)^{\text{th}}$ iteration in Algorithm 1 with the computed incremental rotation $\boldsymbol{\Theta}^{k+1}$ and translation $\mathbf{w}^{k+1}$ provides a clear juxtaposition of the parameterizations adopted for rotations and translations. The vector space structure of $\mathbb{R}^3$ and the lack of such linearity in $SO_3$ is the main reason behind the different update formulas for rotations and translations appearing in Equation (18).

A unified perspective, however, emerges from identifying the space of admissible translations $\mathbb{R}^3$ endowed with the addition operation $(+)$ as a Lie group. In such a trivial case, the Lie algebra coincides with the Lie group itself and the exponential map is simply the identity. Hence, for a given $\mathbf{t} \in \mathbb{R}^3$, the exponential map $\mathrm{Exp}_{\mathbf{t}} : \mathbb{R}^3 \to \mathbb{R}^3$ for the additive group is given by

$$\mathrm{Exp}_{\mathbf{t}}[\mathbf{w}] \triangleq \mathbf{t} + \mathbf{w}.$$

We can now view isometries as being locally parameterized over the Lie algebra $\mathfrak{so}_3 \times \mathbb{R}^3$ of the tensor product Lie group $SO_3 \times \mathbb{R}^3$. The exponential map at the element $(\mathbf{R}, \mathbf{t})$ is denoted by $\mathrm{Exp}_{(\mathbf{R},\mathbf{t})} : \mathfrak{so}_3 \times \mathbb{R}^3 \to SO_3 \times \mathbb{R}^3$ and is defined as

$$\mathrm{Exp}_{(\mathbf{R},\mathbf{t})}[\hat{\boldsymbol{\Theta}}, \mathbf{w}] = (\mathrm{Exp}_{\mathbf{R}}[\hat{\boldsymbol{\Theta}}], \mathrm{Exp}_{\mathbf{t}}[\mathbf{w}]),$$

so that the update in Equation (18) is given by

$$(\mathbf{R}^{k+1}, \mathbf{t}^{k+1}) = \mathrm{Exp}_{(\mathbf{R}^k,\mathbf{t}^k)}[\hat{\boldsymbol{\Theta}}^{k+1}, \mathbf{w}^{k+1}], \tag{19}$$

revealing a uniform treatment of rotations and translations in the algorithm. The unified perspective apparent from Equation (19) fully justifies the title of this article as well. Indeed, we could have adopted the perspective in Equation (19) throughout this section without altering the stationarity conditions, their linearization or Algorithm 1. We have avoided such an abstract presentation in favour of simplicity.

### 3.5. Illustrative examples

We demonstrate the performance of Algorithm 1 with a few examples. In the first example depicted in Figure 4, the target surface $\mathcal{S}$ is the zero-level set of the function

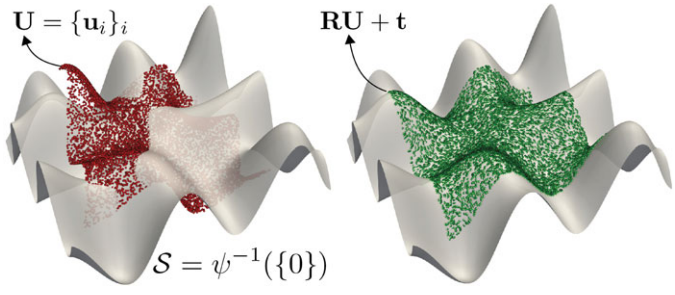
$$\psi(x, y, z) \triangleq y \sin x - x \cos y - 10z/3, \tag{20}$$

and is shown in gray in the figure. The cloud $\mathbf{U}$ to be registered consists of $10^4$ points and is shown in red. It is defined by superimposing a randomly generated rigid body transformation on an irregular sampling of $\mathcal{S}$. We start Algorithm 1 with the trivial initial guess $(\mathbf{R}^0, \mathbf{t}^0) = (\mathbf{I}, 0)$, where $\mathbf{I}$ is the identity matrix. The registered data set is shown in green in the figure. In this example, we find that with the tolerance for the norm of the residual set to $10^{-10}$, the algorithm converges in seven iterations. Details of the iterations are reported in Figure 4(b). Note that at the converged solution, the functional $\mathrm{J_p}$ being minimized is practically reduced to zero. Since $\mathrm{J_p} \geq 0$, we infer that the algorithm has indeed found the exact solution. Furthermore, as generally expected in Newton iterations, we find that once the residual is reduced to a small value, it converges to zero very rapidly.

Using the surfaces defined by the level set functions listed in Figure 5, Figure 6 shows more examples demonstrating the application of Algorithm 1. As we did in Figure 4(a), the implicitly defined surface is shown in gray in each case. The nomenclature chosen for the surfaces is based on the terminology used in the literature on algebraic surfaces [KI15], and in visualization and surface polygonalization softwares [Con, Tea, Ren]. The unregistered point clouds are shown in red, while their registered counterparts computed by Algorithm 1 are shown in green in Figure 6. In each example, we start the algorithm with the trivial initial guess $(\mathbf{R}^0, \mathbf{t}^0) = (\mathbf{I}, 0)$ for the solution. The tables in the last row of the figure report additional details on the the number of points registered, the reduction in the value of the objective after registration, the number of iterations required for convergence and the time taken for the computation. We note that for the sake of better visualization, the images in the figure show registration of sparser clouds, i.e. with fewer number of points, while the table reports details of the calculation for registering denser point clouds.

### 3.6. Implementation and performance

We provide code snippets closely resembling our C++ implementation of Algorithm 1 as Supplementary Information supporting this article. It includes outlines of routines for assembling the matrix-vector system to be resolved at each iteration, for calculating the




**(a)** Unregistered and registered point clouds.

| Iter # | $\mathrm{J_p}$ | $\|\mathbf{r}\|$ | $\|(\Theta, \mathbf{w})\|$ |
|---|---|---|---|
| 0 | $5.72 \times 10^{4}$ | - | - |
| 1 | $1.73 \times 10^{4}$ | $3.60 \times 10^{5}$ | 0.60 |
| 2 | $6.16 \times 10^{3}$ | $1.81 \times 10^{5}$ | 0.65 |
| 3 | $6.63 \times 10^{2}$ | $1.11 \times 10^{5}$ | 0.28 |
| 4 | $3.62 \times 10^{-3}$ | $1.05 \times 10^{4}$ | $3.43 \times 10^{-2}$ |
| 5 | $1.65 \times 10^{-11}$ | $6.26 \times 10^{1}$ | $2.93 \times 10^{-4}$ |
| 6 | $1.83 \times 10^{-26}$ | $5.74 \times 10^{-3}$ | $1.01 \times 10^{-8}$ |
| 7 | $1.77 \times 10^{-26}$ | $3.89 \times 10^{-11}$ | $1.19 \times 10^{-16}$ |

**(b)** Details of the iterations to convergence.

**Figure 4:** *An illustrative example demonstrating the application of Algorithm 1 to register a point cloud* $\mathbf{U}$ *(in red) onto the surface* $\mathcal{S}$ *that is specified as the zero-level set of the function in Equation* (20)*. The algorithm converges within seven iterations, whose details are provided in the table on the right.*

| Surface | Implicit function $\psi(x,y,z)$ |
|---|---|
| Blob | $x^2+y^2+z^2+\sin 4x+\sin 4y+\sin 4z-1$ |
| Bretzel | $(x^2(1-x^2)-y^2)^2+z^2-0.01$ |
| Chair | $(x^2+y^2+z^2-ak^2)^2-b((z-k)^2-2x^2)((z+k)^2-2y^2), a=0.95, b=0.8, k=5$ |
| Goursat | $x^4+y^4+z^4-5(x^2+y^2+z^2)+11.8$ |
| Mullen | $(1+x^2)(1+y^2)(1+z^2)-8xyz-2$ |
| Pilz | $((x^2+y^2-1)^2+(z-1)^2)((x^2/a^2+(z-b)^2-1)+y^2)-0.1, a=1.4, b=0.3$ |
| Squared-torus | $(\sqrt{x^4+y^4}-2)^2+z^4-1$ |
| T4 | $8(x^4+y^4+z^4)-8(x^2+y^2+z^2)+3$ |
| T6 | $32(x^6+y^6+z^6)-48(x^4+y^4+z^4)+18(x^2+y^2+z^2)-3$ |
| Tangled | $x^4+y^4+z^4-5(x^2+y^2+z^2)+11.8$ |
| Rings | $((x^2+y^2-0.8^2)^2+(z^2-1)^2)((x^2+z^2-0.8^2)^2+(y^2-1)^2)((z^2+y^2-0.8^2)^2+(x^2-1)^2)-0.01$ |
| Monge | $y\sin x-x\cos y-10z/3$ |

**Figure 5:** *Level set functions defining the implicit surfaces appearing in the examples in Sections 3 and 4.*

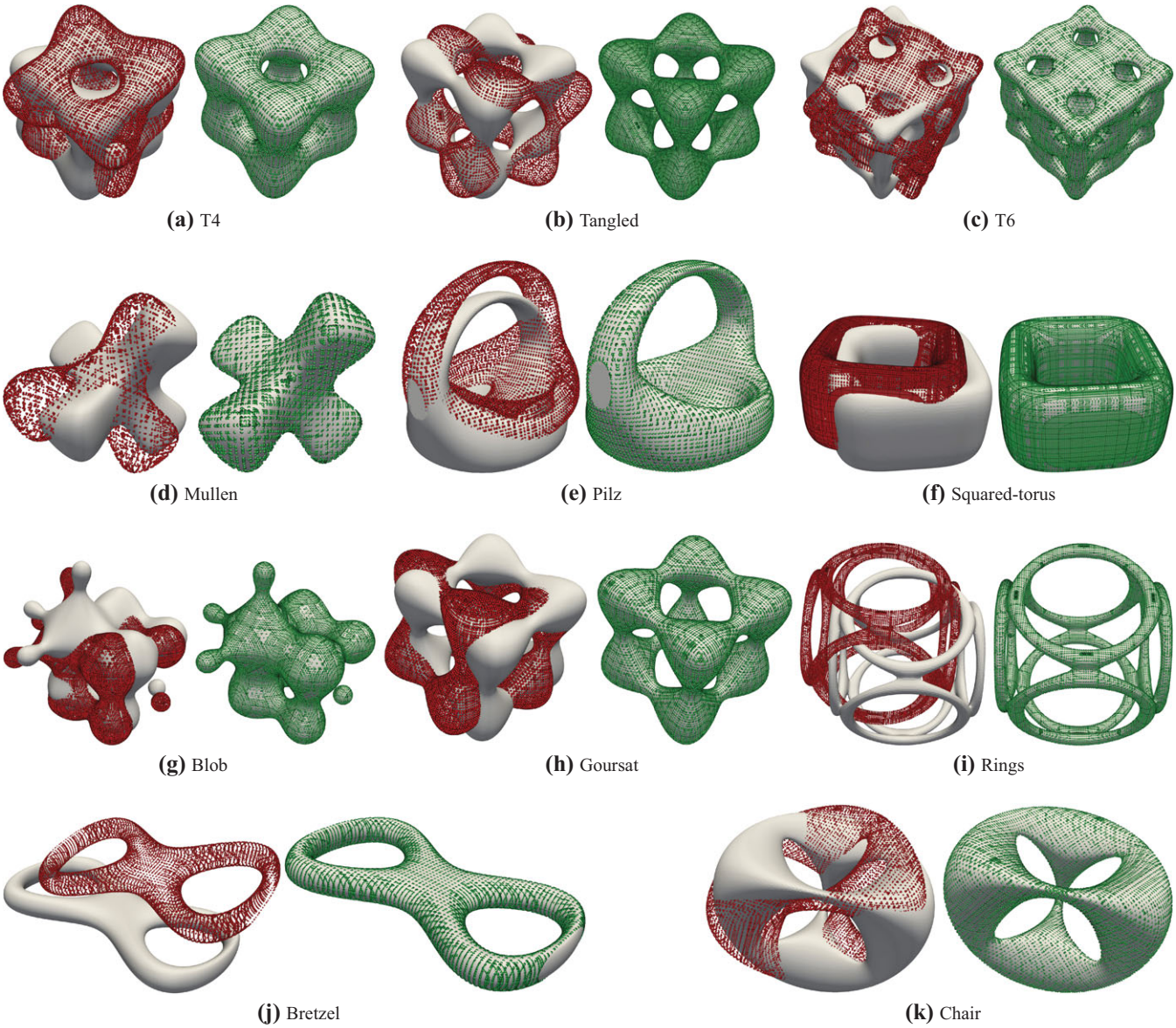
exponential map and for computing pose updates. We have omitted details of the routines we use for solving the system of linear equations at each iteration. In our implementation, we use the GNU GSL open source library [Gou09] to compute the LU-decomposition of the matrix **K**, following which, computing the solution vector is straightforward. Alternative linear solvers can be used without significantly altering performance since the size of the system being resolved is small (**K** is a $6 \times 6$ matrix).

All the examples shown and run times reported here correspond to executing our serial implementation on a Mac Pro workstation (3.5GHz, Intel Xeon E5 processor). We use the GNU g++ compiler with the optimization flag set to '-O2'.

Referring to the tables in Figures 6(l) and (m), we first draw attention to the size of the point cloud data sets registered across the different examples. The number of points ranges from $10^4$ for the Monge surface, to about $8.2 \times 10^5$ in the case of the Blob surface. While it is guaranteed that the solutions found in the examples are stationary points of $J_p$ since the residual is reduced to zero (smaller than a tolerance), the fact that the objective functional is reduced by factors exceeding $10^{10}$ in all the examples suggests that a minimum has indeed been found by the algorithm. We consistently find that the algorithm converges within a dozen iterations. The only exception appears to be the example of the 'Rings' surface, where 17 iterations were required. We suspect the reason behind this to be the presence of intricate features in the surface (note the complicated topology of the surface in Figure 6(i)), which in turn causes its level set function to have steep gradients.Owing to the complexity of the level set functions defining the surfaces in these examples, we rely on a symbolic manipulator (Mathematica [Inc]) to compute gradients and Hessians, and export the expressions found directly to our routines. In particular, we did not make any attempt to efficiently implement these expressions. We mention this point because the execution time of the algorithm depends directly on the implementation of the level set function $\psi$ and its derivatives. We highlight a few examples that substantiate this observation. In the case of the Monge surface, where the level set function given by Equation (20) and its derivatives assume very simple forms, the execution time of 0.02 s to register $10^4$ points is very fast. Similarly, registering $4.5 \times 10^5$ points to the Goursat surface takes only 1.18 s. In contrast, registering $3.5 \times 10^5$ points to the Squared-torus surface takes 5.29 s while requiring a similar number of iterations. This is a direct consequence of the inefficient evaluation of $\nabla\psi$ and $\nabla\nabla\psi$ for the Squared-torus surface.

In the next set of performance tests, we register point clouds of varying sizes to each of the 12 surfaces. Figure 7 reports our findings. The plot on the left shows the execution time required per iteration for the different examples. We find an approximately linear scaling between the number of points registered (spanning at least two decades) and the execution time per iteration for multiple surfaces. The spread of data in the plot is directly related to differences in the cost of evaluating the level set function and their derivatives for each surface.

An alternative interpretation of the timing data is provided in the plot on the right in Figure 7. Therein, we have normalized the execution times and the data set sizes by corresponding values from the smallest data set registered to a surface. For example, we register point clouds of sizes 59 367, 191 045 and 822 537 to the Blob surface, which requires 0.0195, 0.0625 and 0.273 s per iteration. Labelling these data sets as DS-1, DS-2 and DS-3, the problem scale factors of DS-2 and DS-3 are $191\,045/59\,367 \approx 3.22$ and $822\,537/59\,367 \approx 13.86$ and the scaled execution times are $0.0625/0.0195 \approx 3.2$ and $0.273/0.0195 \approx 14.02$. Hence, we find that to register data sets DS-2 and DS-3 which are 3.22 and 13.86 times as large as DS-1, the algorithm takes 3.2 and 14.02 times as much time as registering DS-1. Such a normalization serves to factor out the differences in the cost of evaluating the implicit function and their derivatives for the different surfaces. The plot shows that the execution time of the algorithm scales linearly with the size of the data set being registered, and in particular, reveals that there are no significant overheads in the implementation of the algorithm.



**(a)** T4 **(b)** Tangled **(c)** T6

**(d)** Mullen **(e)** Pilz **(f)** Squared-torus

**(g)** Blob **(h)** Goursat **(i)** Rings

**(j)** Bretzel **(k)** Chair

| Surface | # points | $J_{final}/J_{initial}$ | # iterations | Exec. time (sec) |
|---|---|---|---|---|
| Blob | 822537 | $1.96\times 10^{-12}$ | 6 | 1.64 |
| Bretzel | 246600 | $2.87\times 10^{-15}$ | 8 | 0.64 |
| Chair | 280206 | $1.13\times 10^{-12}$ | 7 | 0.49 |
| Goursat | 456504 | $2.19\times 10^{-12}$ | 6 | 1.18 |
| Mullen | 22938 | $3.78\times 10^{-13}$ | 11 | 0.053 |
| Pilz | 223732 | $1.51\times 10^{-13}$ | 10 | 0.52 |

**(l)**

| Surface | # points | $J_{final}/J_{initial}$ | # iterations | Exec. time (sec) |
|---|---|---|---|---|
| Squared-torus | 350396 | $1.33\times 10^{-12}$ | 7 | 5.29 |
| T4 | 322928 | $3.93\times 10^{-13}$ | 7 | 0.71 |
| T6 | 443450 | $7.95\times 10^{-14}$ | 11 | 3.46 |
| Tangled | 326628 | $1.26\times 10^{-12}$ | 6 | 0.85 |
| Rings | 175733 | $4.35\times 10^{-16}$ | 17 | 0.77 |
| Monge | 10000 | $4.53\times 10^{-31}$ | 7 | 0.02 |

**(m)**

**Figure 6:** *More example demonstrating the performance of Algorithm 1 for registering point clouds onto implicit surfaces. The level set functions defining the surfaces shown are provided in Figure 5.*

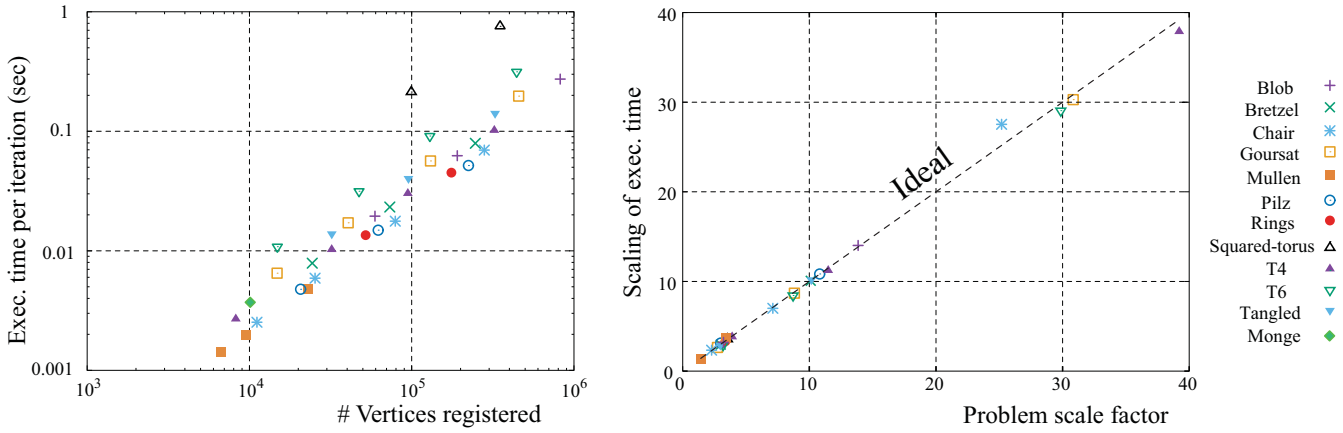



**Figure 7:** *Plots examining the performance of our implementation of Algorithm 1 when registering data sets of varying sizes to each of the 12 surfaces depicted in Figure 6. The plot on the left shows that the execution time per iteration scales approximately linearly with the size of the data set being registered. Normalizing the axes by the size and execution time for the smallest data set registered with each surface yields the plot on the right. The close to ideal scaling reveals that the overheads involved in implementing the algorithm are negligible and that it is straightforward to parallelize.*

This observation suggests that it is straightforward to parallelize the implementation of the algorithm. For instance, the assembly of the matrix-vector system, which is the main contributor to the algorithm's execution time, can be distributed among multiple cores in a shared memory machine using OpenMP directives.

Finally in Figure 8, we illustrate the performance of the algorithm with noise added to the point cloud being registered. The noise added in Figure 8(a) follows a uniform distribution, while that in the remaining two examples follows a Gaussian distribution. We mention that there are no special steps included in the algorithm to account for the presence of noise — the algorithm is used as is. While the results appear to be encouraging, special techniques that model the noise distribution to achieve better registration will inevitably be required.

### 3.7. Comparisons with alternative algorithms

We compare the performance of Algorithm 1 with alternative rigid registration algorithms, namely point-to-point ICP, point-to-plane ICP, coherent point drift (CPD) [MS10], 4PCS [AMCO08] and Super4PCS [MAM14]. Before discussing our findings, we provide details of the data sets and the implementations used for each algorithm.

***Target data sets.*** The point cloud $\mathbf{U}$ to be registered by Algorithm 1, ICP (point-to-point and point-to-plane), CPD, 4PCS and Super4PCS are all identical. However, the target data sets are not. Algorithm 1 requires the target to be an implicit surface, while ICP, CPD, 4PCS and Super4PCS require the target to be a point cloud. To this end, we use a sampling of the surface provided to Algorithm 1 as the target in the remaining algorithms. In the process, we ensure that the sampling is relatively uniform and that the number of sample points is at least 2–3 times larger than the number of points in $\mathbf{U}$. Hence, the target point cloud provided to ICP, CPD, 4PCS and Super4PCS is at least twice as dense as the data set $\mathbf{U}$.

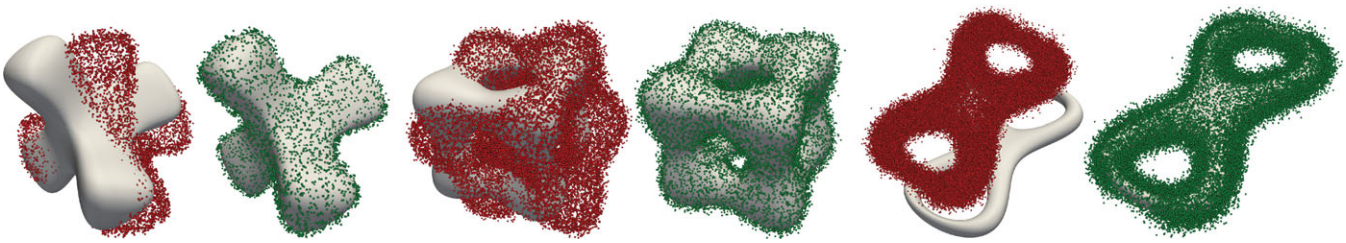
***Implementations used.*** Numerous softwares, including open source ones [CCC*08, BC11, clo18], provide efficient implementations of ICP. Here, we use the implementation provided in the *pcregistericp()* routine as part of the Computer Vision System Toolbox in Matlab [mat18]. The choice of either the point-to-point or the point-to-plane distance metric is specified through an argument passed to the function. We do not employ any downsampling for the target point cloud. In the case of the point-to-plane metric, we precompute the normals to local tangent planes using 15 points with the *pcnormals()* routine before invoking the registration function call. The convergence tolerances required in the routine are set to $10^{-8}$.

We use the Matlab implementation of CPD accessed from [Myr] provided by the authors of [MS10]. All algorithmic parameters are set to default values. With the intention of using the most accurate version of the algorithm, we do not employ the option for accelerating the algorithm using fast Gauss transforms.

For 4PCS and Super4PCS, we use the C++ implementation accessed from [Mel] provided by the authors of [MAM14]. We set the command line parameter for the estimated overlap to be 95% (`-o 0.95`) and for the number of samples to be used for matching to be 2000 (`-n 2000`). All other algorithmic parameters are used as provided in the implementation.

***Error measures.*** Noting that all the experiments shown here use synthetic examples, we measure the accuracy of each algorithm in terms of the errors in the computed transformations:

$$\mathrm{E} = \left( \sum_{i,j=1}^{3} \left( \mathbf{R}_{ij} - \mathbf{R}^{\star}_{ij} \right)^2 + \sum_{i=1}^{3} \left( \mathbf{t}_i - \mathbf{t}^{\star}_i \right)^2 \right)^{1/2}, \qquad (21)$$



**(a)** Mullen **(b)** T4 **(c)** Bretzel

**Figure 8:** *Examples demonstrating the performance of Algorithm 1 with noise present in the point cloud data set. The noise added follows a uniform distribution in (a) and a Gaussian distribution in (b) and (c).*

where $(\mathbf{R}^\star, \mathbf{t}^\star)$ represents the correct solution. Evidently, E is the Frobenius norm of the difference between the homogeneous transformation matrices representing the correct and computed solutions.

***Timing.*** The run time for Algorithm 1 consists of the time required for executing Newton iterations. This includes the time taken to assemble the matrix-vector system, to solve the resulting system of equations and to update the pose for the next iteration. The execution time required for ICP is measured using the *timeit()* routine in Matlab, which triggers various run-time optimizations and calls the routine multiple times to report the median of the measurements. When timing point-to-plane ICP, we do not include the time required to compute normals. The run times for CPD, 4PCS and Super4PCS are reported by the implementations themselves.

Figure 9(a) reports the errors in the transformations computed by each one of the six algorithms tested in five different examples. Without exception, we find that Algorithm 1 is more accurate than the alternatives. We attribute this to the fact that Algorithm 1 naturally utilizes detailed information about the target data set (normals and curvatures) without introducing a discretization for it. Figure 9(b) reports the execution times for the different algorithms in each example. Point-to-plane ICP and Algorithm 1 easily outperform the remaining algorithms. We also find that for a fixed number of iterations, Algorithm 1 is generally about 10 times faster than point-to-plane ICP. We have omitted the timing data for point-to-point ICP in Figure 9(b) because it requires a large number of iterations ($> 500$) to converge in the examples shown.

Based on estimates for their complexity, both CPD and 4PCS are expected to be slow. The CPD algorithm can be sped up by resorting to fast Gauss transforms; we have not used this option because the resulting solutions are less accurate. Super4PCS is an accelerated version of 4PCS. While it can be run faster by using fewer matching points, we found that using fewer than 2000 points resulted in incorrect solutions in some of the examples.

We conclude this discussion emphasizing that the comparisons shown should be interpreted with caution. Unlike Algorithm 1, we note that ICP, CPD, 4PCS and Super4PCS all presume the target data set to be point clouds and therefore introduce some discretization of the target surface. The notion of correspondence between the model and target data sets is also very different among these algorithms, which again, should be contrasted with Algorithm 1 that does not introduce any notion of correspondence. There are various aspects of CPD, 4PCS and Super4PCS that our examples here did not test, including for instance, partial overlap between data sets or the presence of noise/outliers. For these reasons, the comparisons we have shown are necessarily subjective.

## 4. Generalizations

We devote this section to presenting a few generalizations of Algorithm 1. In Sections 4.1 and 4.2, we discuss registering triangulations and curves to implicit surfaces. The adaptations of Algorithm 1 required to this end are quite straightforward and essentially amount to replacing summation signs with integrals. A unified interpretation of these variants of Algorithm 1 is in fact possible, as we highlight in Section 4.3. Finally, in Section 4.4, we present a modification of Algorithm 1 that incorporates scale factors during registration.

### 4.1. Surface-to-surface registration

We consider the problem of registering a surface $\Gamma$ to a target surface $\mathcal{S}$. As we did in Algorithm 1, we assume $\mathcal{S}$ to be specified implicitly according to Equation (10). We compute the pose $(\mathbf{R}, \mathbf{t})$ registering $\Gamma$ onto $\mathcal{S}$ as a minimizer of the functional

$$\mathrm{J_s}(\mathbf{Q}, \mathbf{s}) \triangleq \frac{1}{2} \int_{\mathbf{u} \in \Gamma} \psi^2(\mathbf{Q}\mathbf{u} + \mathbf{s})\, d\Gamma, \tag{22}$$

where $d\Gamma$ represents the surface area measure of $\Gamma$. Assuming $\Gamma$ to be specified parametrically over a domain $\mathbf{A} \subset \mathbb{R}^2$, Equation (22) can be written more explicitly as

$$\mathrm{J_s}(\mathbf{Q}, \mathbf{s}) = \frac{1}{2} \int_{\xi \in \mathbf{A}} \psi^2(\mathbf{Q}\Gamma(\xi) + \mathbf{s})\, |\Gamma_{\xi_1} \times \Gamma_{\xi_2}|\, d\xi, \tag{23}$$

where we have denoted the parameterization of the surface also by $\Gamma$, $(\xi_1, \xi_2)$ denote the coordinates of $\xi \in \mathbf{A}$ and $|\Gamma_{\xi_1} \times \Gamma_{\xi_2}| d\xi$ is the surface area measure at $\xi$.

Stationarity conditions and their linearization follow from calculations analogous to those in Sections 3.1 and 3.2, and Propositions 1 and 2. We only record the final expressions here and omit detailed arguments. At the pose $(\mathbf{R}, \mathbf{t})$, the residual is given by

$$\mathbf{r}(\mathbf{R}, \mathbf{t}) = \int_{\mathbf{A}} \begin{bmatrix} (\mathbf{R}\Gamma(\xi)) \times \psi \nabla \psi(\mathbf{R}\Gamma(\xi) + \mathbf{t}) \\ \psi \nabla \psi(\mathbf{R}\Gamma(\xi) + \mathbf{t}) \end{bmatrix} |\Gamma_{\xi_1} \times \Gamma_{\xi_2}(\xi)|\, d\xi, \tag{24}$$

| Surface | Error in the computed transformation | | | | | |
|---|---|---|---|---|---|---|
| | Algo A | ICP (Point-to-Point) | ICP (Point to Plane) | CPD | 4PCS | Super4PCS |
| T4 | $6.67\times 10^{-7}$ | $2.13\times 10^{-4}$ | $1.67\times 10^{-5}$ | $1.50\times 10^{-4}$ | $2.24\times 10^{-2}$ | $1.79\times 10^{-2}$ |
| Mullen | $7.58\times 10^{-7}$ | $2.44\times 10^{-4}$ | $1.19\times 10^{-5}$ | $1.90\times 10^{-4}$ | $1.06\times 10^{-1}$ | $1.90\times 10^{-1}$ |
| T6 | $9.89\times 10^{-7}$ | $2.12\times 10^{-4}$ | $2.12\times 10^{-5}$ | $6.49\times 10^{-4}$ | $7.78\times 10^{-2}$ | $8.31\times 10^{-2}$ |
| Rings | $7.41\times 10^{-7}$ | $1.89\times 10^{-4}$ | $7.91\times 10^{-6}$ | $1.24\times 10^{-4}$ | $3.24\times 10^{-2}$ | $2.80\times 10^{-2}$ |
| Pilz | $9.02\times 10^{-7}$ | $7.22\times 10^{-5}$ | $1.44\times 10^{-6}$ | $6.90\times 10^{-3}$ | $2.08\times 10^{-2}$ | $5.13\times 10^{-2}$ |

**(a)** Errors in transformations computed by each algorithm measured using the metric E defined in eq. (21).

| Surface | # model points | # scene points (ICP, CPD, 4PCS) | Execution time (seconds) | | | | |
|---|---|---|---|---|---|---|---|
| | | | Algo A | ICP (Point-to-Plane) | CPD | 4PCS | Super4PCS |
| T4 | 8236 | 32147 | 0.02 | 0.15 | 272.25 | 47.11 | 4.45 |
| Mullen | 9507 | 22938 | 0.18 | 0.16 | 149.99 | 9.71 | 3.45 |
| T6 | 14852 | 47344 | 0.12 | 0.27 | 845.43 | 61.87 | 5.02 |
| Rings | 20133 | 52049 | 0.51 | 0.28 | 912.25 | 598.13 | 44.49 |
| Pilz | 20679 | 64400 | 0.44 | 0.31 | 1006.34 | 83.41 | 6.77 |

**(b)** Sizes of data sets and execution times for each algorithm.

**Figure 9:** *A comparison of the performance of Algorithm 1 with alternatives from the literature, see Section 3.7 for details. We find that Algorithm 1 easily outperforms the alternatives in terms of accuracy, while its run time is comparable to that of point-to-plane ICP.*

and $(\mathbf{R}, \mathbf{t})$ is a stationary point of $\mathrm{J_s}$ if $\mathbf{r}(\mathbf{R}, \mathbf{t}) = 0$. The Jacobian matrix, required for linearizing $\mathbf{r}$, is given by

$$\mathbf{K}(\mathbf{R}, \mathbf{t}) = \int_{\mathbf{A}} \begin{bmatrix} \mathbf{K}^{\theta\theta} & \mathbf{K}^{\theta\mathbf{t}} \\ \mathbf{K}^{\mathbf{t}\theta} & \mathbf{K}^{\mathbf{tt}} \end{bmatrix}_{6\times 6} \Bigg|_{\mathbf{R}\Gamma(\xi)+\mathbf{t}} |\Gamma_{\xi_1} \times \Gamma_{\xi_2}|\, d\xi, \quad (25)$$

with: $\mathbf{K}^{\mathbf{tt}} = \nabla\psi \otimes \nabla\psi + \psi\nabla\nabla\psi,$

$$\mathbf{K}^{\theta\mathbf{t}} = (\widehat{\mathbf{R}\Gamma(\xi)})\,\mathbf{K}^{\mathbf{tt}}$$

$$\mathbf{K}^{\mathbf{t}\theta} = \left(\mathbf{K}^{\theta\mathbf{t}}\right)^{t},$$

and $\mathbf{K}^{\theta\theta} = (\widehat{(\psi\nabla\psi)} - (\widehat{\mathbf{R}\Gamma(\xi)})\mathbf{K}^{\mathbf{tt}})(\widehat{\mathbf{R}\Gamma(\xi)}).$

The algorithm for registering $\Gamma$ to $\mathcal{S}$ is now identical to Algorithm 1, with the expressions for the residual and Jacobian required at each iteration now furnished by Equations (24) and (25). The integrals appearing in these equations are evaluated using standard quadrature rules over the parametric domain $\mathbf{A}$.

Figure 10 shows representative examples registering triangulated surfaces ($\Gamma$) onto implicit surfaces ($\mathcal{S}$). The unregistered surface $\Gamma$ is shown in red, while the registered counterpart $\mathbf{R}\Gamma + \mathbf{t}$ is shown in black. Virtually all details, including convergence criteria, are identical to those used for registering point clouds to surfaces in Algorithm 1. The last row of images in the figure shows registration of noisy surfaces. For brevity sake, we omit providing further details but mention that the number of iterations required for computing a converged solution and the execution times are comparable to those reported for Algorithm 1. We note that there is no restriction that $\Gamma$ be a triangulated surface; Equations (24) and (25) are applicable for registering general parametric surfaces. For instance, spline surfaces can be registered in this way. We have chosen triangulated surfaces in our examples solely for the sake of convenience.

### 4.2. Curve-to-surface registration

Next, we consider the problem of registering a curve $\gamma$ to an implicit surface $\mathcal{S}$ and compute the required pose $(\mathbf{R}, \mathbf{t})$ as a minimizer of the functional

$$\mathrm{J_c}(\mathbf{Q}, \mathbf{s}) \triangleq \frac{1}{2}\int_{\eta\in\mathbf{I}} \psi^2(\mathbf{Q}\gamma(\eta) + \mathbf{s})\,|\gamma'(\eta)|\,d\eta, \quad (26)$$

where we have assumed the curve to be parameterized as $\gamma : \mathbf{I} \to \mathbb{R}^3$. Notice that we recover Equation (26) from Equation (23) by simply replacing the integral over a surface with that over a curve, and suitably replacing the measure used for integration. Indeed, the expressions for the residual and Jacobian for this problem are virtually identical to Equations (24) and (25), with surface integrals replaced by line integrals. For this reason, we do not repeat the expressions here. Representative examples demonstrating the registration of piecewise linear curves onto implicit surfaces are shown in Figure 11.

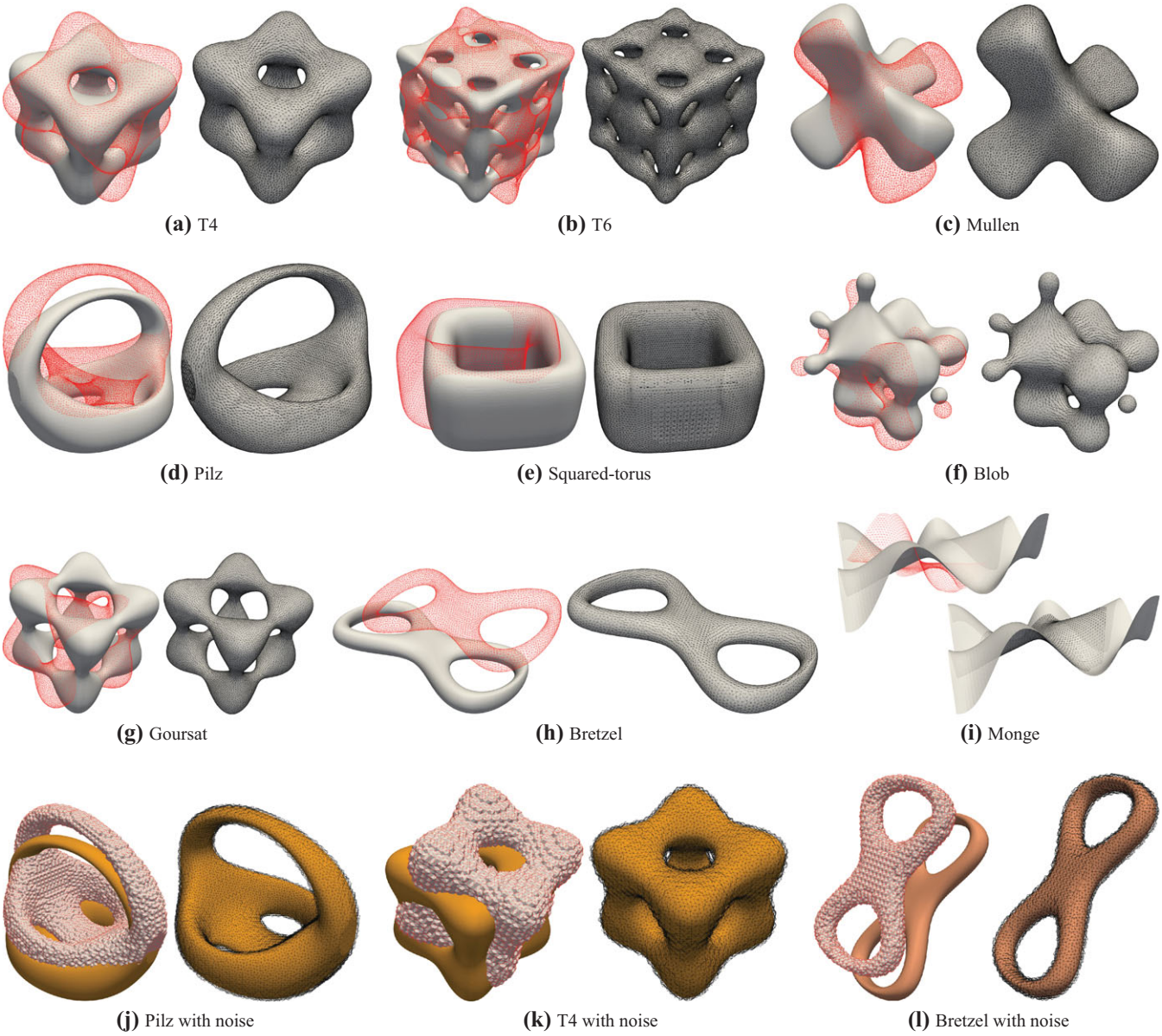



**Figure 10:** *Examples demonstrating pairwise registration of surfaces using an adaptation of Algorithm 1 that is discussed in Section 4.1. In each case, the moving data set is a triangulated surface, while the corresponding target is an implicit surfaces. The unregistered surface is shown in red, while the registered counterpart is depicted in black. The last row of images shows instances of registrations achieved with noise added to the triangulations.*

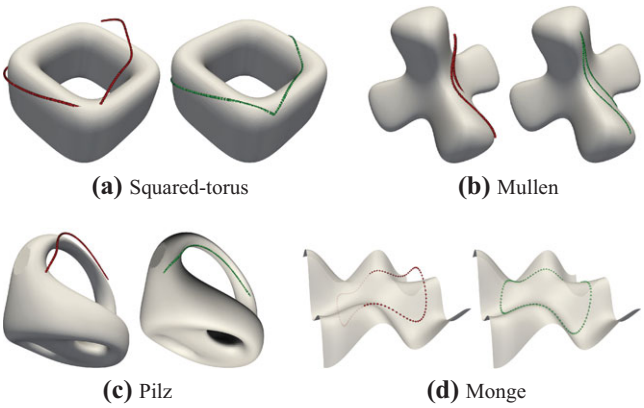
### 4.3. A unified algorithm for varied data sets

The resemblance of the objective functionals $J_p$, $J_s$ and $J_c$ in Equations (12), (23) and (26), respectively, is difficult to miss. Evidently, the differences in their definitions and in the residuals and Jacobians arise solely because of the different nature of the data sets being registered, namely point clouds, surfaces and curves. A unified perspective emerges by choosing a suitable measure for integration over the data set being registered. Specifically, denoting the data set to be registered by $\mathcal{M}$, we set the objective functional at a pose $(\mathbf{Q}, \mathbf{s})$ to equal

$$J(\mathbf{Q}, \mathbf{s}) \triangleq \frac{1}{2} \int_{\mathbf{x} \in \mathcal{M}} \psi^2(\mathbf{Q}\mathbf{x} + \mathbf{t})\, d\mu, \tag{27}$$

where $d\mu$ represents a measure on $\mathcal{M}$. Then, notice that identifying

- $\mathcal{M}$ with the point cloud $\mathbf{U}$ and $d\mu$ with Dirac measures yields Equation (12),
- $\mathcal{M}$ with the surface $\Gamma$ and $d\mu$ with the surface measure yields Equation (23) and
- $\mathcal{M}$ with the curve $\gamma$ and $d\mu$ with the line measure yields Equation (26).



**(a)** Squared-torus **(b)** Mullen **(c)** Pilz **(d)** Monge

**Figure 11:** *Examples showing the registration of (piecewise linear) curves to implicit surfaces.*

In this sense, Equation (27) represents a unified framework for registering point clouds, curves and surfaces onto implicit surfaces. Such a perspective has useful consequences for the implementation of our algorithms as well. In our implementation for instance, we overload assembly operations based on the type of data set being registered, while remaining aspects of the algorithm remain common, thereby enabling extensive code reuse.

### 4.4. Registration with scaling

It is often necessary to include a scale factor during registration. Here, we discuss an adaptation of Algorithm 1 to include a scale factor when registering a point cloud $\mathbf{U}$ onto an implicitly defined surface $\mathcal{S}$. We mention that the ideas discussed below extend verbatim to the case of registering surfaces and curves as well.

Specifically, we seek a transformation of the form $\mathbf{U} \mapsto c\mathbf{R}\mathbf{U} + \mathbf{t}$ that registers $\mathbf{U}$ onto $\mathcal{S}$, with the scalar $c > 0$ understood to be an (isotropic) scale factor. The inclusion of a scale factor hence provides an additional degree of freedom, which can help achieve better registration. We consider a natural generalization of Equation (11), wherein we seek a minimizer $(\mathbf{R}, \mathbf{t}, c)$ of the functional

$$\hat{\mathrm{J}}_{\mathrm{p}}(\mathbf{R}, \mathbf{t}, c) \triangleq \frac{1}{2} \sum_{i=1}^{n} \psi^2(c\mathbf{R}\mathbf{u}_i + \mathbf{t}). \tag{28}$$

In principle, it is possible to adopt a monolithic solution procedure that computes $(\mathbf{R}, \mathbf{t}, c)$ simultaneously as a set of seven unknowns. We favour a staggered scheme instead, which sequentially approximates the pose $(\mathbf{R}, \mathbf{t})$ and then the scale $c$ at each iteration. Hence, the pose update at each iteration is identical to that in Algorithm 1, while the scale factor at a given pose $(\mathbf{R}, \mathbf{t})$ is computed as a stationary point of the function

$$c \mapsto \frac{1}{2} \sum_{i=1}^{n} \psi^2(c\mathbf{R}\mathbf{u}_i + \mathbf{t}). \tag{29}$$

The stationarity condition satisfied by the scale $c$ in Equation (29) is

$$\hat{\mathrm{r}}_{(\mathbf{R},\mathbf{t})}(c) = \sum_{i=1}^{n} \psi(c\mathbf{R}\mathbf{u}_i + \mathbf{t}) \nabla\psi(c\mathbf{R}\mathbf{u}_i + \mathbf{t}) \cdot (\mathbf{R}\mathbf{u}_i) = 0. \tag{30}$$

We resolve the nonlinear equation defining $c$ in Equation (30) using a Newton method, with the Jacobian corresponding to the residual in Equation (30) given by

$$\begin{aligned}\hat{\mathrm{K}}_{(\mathbf{R},\mathbf{t})}(c) = \sum_{i=1}^{n} &(\nabla\psi(c\mathbf{R}\mathbf{u}_i + \mathbf{t}) \cdot (\mathbf{R}\mathbf{u}_i))^2 \\ &+ \psi(c\mathbf{R}\mathbf{u}_i + \mathbf{t}) \left((\mathbf{R}\mathbf{u}_i) \cdot \nabla\nabla\psi(c\mathbf{R}\mathbf{u}_i + \mathbf{t}) \cdot (\mathbf{R}\mathbf{u}_i)\right).\end{aligned} \tag{31}$$

The generalization of Algorithm 1 now follows. Starting with the initial guess $(\mathbf{R}^0, \mathbf{t}^0, c^0)$, at the $(k+1)^{\text{th}}$ iteration, we do:

**Algorithm 2**

(i) *Apply scaling*: For each $\mathbf{u}_i \in \mathbf{U}$, do $\mathbf{u}_i \mapsto c\mathbf{u}_i$.
(ii) *Assemble*: $\mathbf{K}^k = \mathbf{K}(\mathbf{R}^k, \mathbf{t}^k)$ and $\mathbf{r}^k = \mathbf{r}(\mathbf{R}^k, \mathbf{t}^k)$ using eqs. (14) and (16).
(iii) *Solve*: Compute the increments $(\boldsymbol{\Theta}^{k+1}, \mathbf{w}^{k+1})$ satisfying

$$\mathbf{K}^k \begin{bmatrix} \boldsymbol{\Theta}^{k+1} \\ \mathbf{w}^{k+1} \end{bmatrix}_{6\times 1} = -\mathbf{r}^k.$$

(iv) *Pose update*: Update the solution to

$$\mathbf{R}^{k+1} = \mathrm{Exp}_{\mathbf{R}^k}[\boldsymbol{\Theta}^{k+1}] \text{ and } \mathbf{t}^{k+1} = \mathbf{t}^k + \mathbf{w}^{k+1}.$$

(v) *Scale estimate*: Set $c = 1$.
(*Newton iterations to compute c*)
  (a) Compute $\hat{\mathrm{r}} = \hat{\mathrm{r}}_{(\mathbf{R}^{k+1},\mathbf{t}^{k+1})}(c)$ using eq. (30).
  (b) Compute $\hat{\mathrm{K}} = \hat{\mathrm{K}}_{(\mathbf{R}^{k+1},\mathbf{t}^{k+1})}(c)$ using eq. (31).
  (c) Compute $\Delta c = -\hat{r}/\hat{\mathrm{K}}$.
  (d) Update $c = c + \Delta c$.
  (e) If $|\Delta c| <$ tolerance, break. Else go to (a).

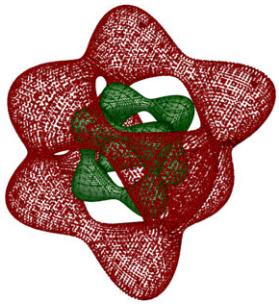
Arguably, the nested Newton loop to compute the scale in step (v) of the algorithm is unnecessary during the initial few registration iterations when the pose estimate is likely to be far away from the correct solution. As a more efficient alternative, we replace the nonlinear problem defining $c$ in Equation (29) by the minimizer of the quadratic function

$$c \mapsto \frac{1}{2} \sum_{i=1}^{n} \|c\mathbf{R}\mathbf{u}_i + \mathbf{t} - \Pi(\mathbf{R}\mathbf{u}_i + \mathbf{t})\|^2, \tag{32}$$

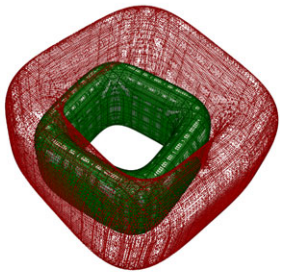
where $\Pi$ is the closest point projection onto $\mathcal{S}$. The minimizer is in fact explicitly computable:

$$c = \frac{\sum_{i=1}^{n} (\Pi(\mathbf{R}\mathbf{u}_i + \mathbf{t}) - \mathbf{t}) \cdot \mathbf{R}\mathbf{u}_i}{\sum_{i=1}^{n} \|\mathbf{R}\mathbf{u}_i\|^2}. \tag{33}$$

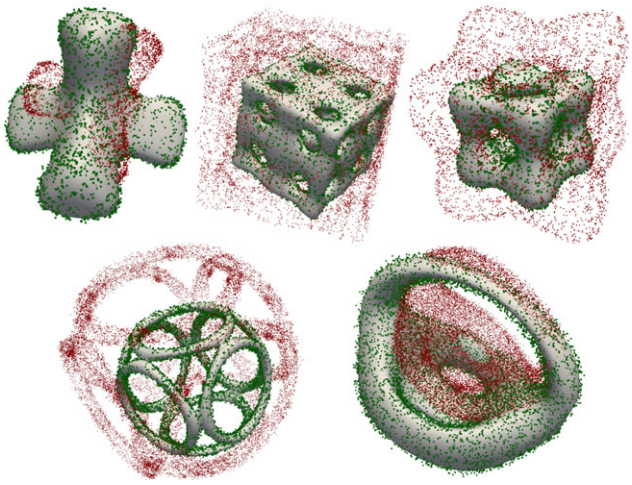
The estimate in Equation (33) is reminiscent of formulas used to compute scale factors in ICP algorithms, cf. [Hor87, Ume91, DZX*10]. In fact, Equation (33) can be interpreted as computing the scale to register the cloud $\{\mathbf{R}\mathbf{u}_i + \mathbf{t}\}_i$ to the cloud of closest point projections $\{\Pi(\mathbf{R}\mathbf{u}_i + \mathbf{t})\}_i$ lying on the surface $\mathcal{S}$.



**(a)** Goursat



**(b)** Squared-torus



**(c)** Registration of noisy point clouds to implicit surfaces using Algorithm 2.

| Surface | Error in (R, t) | | Error in scale factor | | Run time (seconds) | |
|---|---|---|---|---|---|---|
| | Algo 2 | CPD | Algo 2 | CPD | Algo 2 | CPD |
| T4 | 0.028 | 0.029 | 0.0018 | 0.00025 | 0.12 | 280.42 |
| Mullen | 0.059 | 0.059 | 0.0052 | 0.0074 | 0.07 | 138.60 |
| T6 | 0.041 | 0.029 | 0.0036 | 0.0017 | 0.62 | 907.42 |
| Rings | 0.029 | 0.029 | 0.0060 | 0.00026 | 0.36 | 1071.74 |
| Pilz | 0.056 | 0.057 | 0.018 | 0.0031 | 0.22 | 1584.16 |

**(d)** Comparison of Algorithm 2 with CPD for the examples shown in (c).

**Figure 12:** *Illustrative examples demonstrating the performance of Algorithm 2. The table in (d) compares the performance of Algorithm 2 with CPD.*

Equation (33) is, however, a heuristic estimate for the simple reason that unlike Equation (29), the solution given by Equation (33) has little to do with the objective functional $\hat{\mathrm{J}}_{\mathrm{p}}$ being minimized to compute the registered solution. In our implementation, we use Equation (33) to compute $c$ in step (v) of Algorithm 2 for the first few iterations after which we switch to the calculation provided in the algorithm.

Representative examples demonstrating the performance of Algorithm 2 are shown in Figures 12(a) and (b). Figure 12(c) shows additional examples with normally distrbuted noise added to the model data set. In these images, the unregistered data sets are shown in red, while their registered counterparts are shown in green. Figure 12(d) compares the accuracy and run times of Algorithm 2 with CPD for the examples shown in Figure 12(c). The table in Figure 12(d) compares the errors in the computed transformations (measured using Equation (21)) and the errors in the computed scale factors. We find that the accuracy of the two algorithms are comparable for the examples considered. This is despite not including any special procedures to account for noise in Algorithm 2. We also highlight that CPD is many orders of magnitude slower than Algorithm 2. We refer to Section 3.7 for details of the data sets used in Figure 12(c), and for information on the implementation/options used for CPD.

## 5. Extension to Pairwise Registration of Point Clouds

Next, we consider the problem of pairwise registration of point clouds, a problem that is routinely solved by ICP as well as other algorithms mentioned in Section 1. We revisit this problem mainly to emphasize the generality of the proposed system of exponential coordinates, besides demonstrating its applicability in resolving what is perhaps the most ubiquitous version of the registration problem.

In the following, we denote the moving point cloud by $\mathbf{U}$ and the target by $\mathbf{V}$. We register $\mathbf{U}$ onto $\mathbf{V}$ in two ways: (i) by approximating $\mathbf{V}$ with an implicit surface and invoking Algorithm 1, and (ii) by resorting to an ICP-type algorithm while still using exponential coordinates. We pursue these two approaches in Sections 5.1 and 5.2, respectively, and show illustrative examples along the way.

### 5.1. Registration with SSD functions

Let us assume that $\mathbf{V}$ consists of a set of points $\{\mathbf{v}_i\}_{i=1}^m$ and a corresponding set of orientations (unit normals) $\{\mathbf{n}_i\}_{i=1}^m$. In constructing an implicit surface $\mathcal{S}$ that approximates $\mathbf{V}$, we seek a function $\psi$ such that in an approximate sense, the points $\{\mathbf{v}_i\}_i$ lie on the surface $\mathcal{S} = \psi^{-1}(\{0\})$ and the unit normal to $\mathcal{S}$ at $\mathbf{v}_i$ coincides with $\mathbf{n}_i$. This can be viewed as a data fitting exercise, with the function $\psi$ sought in the span of a chosen set of basis functions. Varied definitions of such regression problems and constructions for smooth bases have been proposed in the literature, see [CBC*01] for a representative example.

Here, we define $\psi$ to be a *smoothed signed distance* function [CT11] by setting it to be the minimizer of the quadratic functional

$$\mathrm{E}[\psi] = \sum_{i=1}^{m} \psi^2(\mathbf{x}_i) + \sum_{i=1}^{m} \|\nabla\psi(\mathbf{x}_i) - \mathbf{n}_i\|^2 + \alpha \int_{\Omega} |\nabla\nabla\psi|^2 \, d\Omega, \tag{34}$$

where $\Omega$ is the domain of definition for $\psi$ and $\alpha > 0$ is a user defined parameter. The first term in Equation (34) enforces the requirement that the zero-level set of $\psi$ lies close to the points $\{\mathbf{v}_i\}_i$, while the second term constrains the normal to $\mathcal{S}$ at $\mathbf{v}_i$ to approximate the given orientation $\mathbf{n}_i$. The third term, roughly speaking, controls the smoothness of $\psi$ by penalizing the appearance of large curvatures. Larger choices for $\alpha$ result in smoother surfaces $\mathcal{S}$ but at the expense of larger deviations from the input data $\mathbf{V}$. The minimizer $\psi$ of Equation (34) is *not* a signed distance function in general. For instance, its gradient on $\mathcal{S}$ may not have unit magnitude. This is,

however, not a concern for us since we do not require $\psi$ to be a signed distance function in the first place.

In contrast to the algorithm proposed in [CT11] for resolving Equation (34), we seek the minimizer $\psi$ that is of the form

$$\psi(\mathbf{x}) = \sum_{a=1}^{\ell} \psi_a N_a(\mathbf{x}), \tag{35}$$

where $N_a : \Omega \to \mathbb{R}$ are local *max-ent* shape functions [AO06]. It can be shown that Equation (34) has a unique solution of the form Equation (35) for any $\alpha > 0$. Local maximum entropy shape functions are well suited for mesh-free approximation schemes, with the notable feature of representing a Pareto optimality between maximizing the entropy of approximation as well as maximizing the locality of shape functions. Given an unorganized point cloud $\{\xi_a\}_{a=1}^{\ell}$ and a parameter $0 < \beta < \infty$, the value of the shape functions $\{N_a(\xi)\}_a$ at a point $\xi$ in the interior of the convex hull of $\{\xi_a\}_a$ is given by

$$N_a(\xi) = \frac{\exp[-\beta\|\xi - \xi_a\|^2 + \boldsymbol{\lambda}^\star(\xi)\cdot(\xi - \xi_a)]}{Z(\xi, \boldsymbol{\lambda}^\star(\xi))},$$

where the partition function $Z$ is

$$Z(\xi, \boldsymbol{\lambda}) \triangleq \sum_{a=1}^{\ell} \exp[-\beta\|\xi - \xi_a\|^2 + \boldsymbol{\lambda}\cdot(\xi - \xi_a)],$$

and $\quad \lambda^\star(\xi) \triangleq \mathit{arg\ min}_{\boldsymbol{\lambda}\in\mathbb{R}^3} \log Z(\xi, \boldsymbol{\lambda})$.

Evidently, computing shape function values at a point itself requires resolving a minimization problem. These are, however, small (three-dimensional) convex problems and are therefore easily solved. Of direct consequence to our purpose of computing $\psi$ is the fact that local max-ent functions are smooth, and in particular, twice continuously differentiable. This is in contrast to conventional finite element shape functions, which are only continuous and therefore ill-suited for minimizing the functional in Equation (34). The radius of support of max-ent shape functions is approximately $\sqrt{-\log(\varepsilon_0)/\beta}$, where $\varepsilon_0$ is a given cut-off value for the shape functions. Consequently, the parameter $\beta$ directly controls the sparsity of the matrix realized when resolving the linear system resulting from the minimization of Equation (34).

Figure 13 shows an example computing an implicit representation using SSD functions for a subset of the *Happy Buddha* data set retrieved from [TL]. The point cloud in black in Figure 13(a) represents the input data set $\mathbf{V}$. It is shown immersed in a uniform grid of $21 \times 21 \times 21$ nodes that constitute the set $\{\xi_a\}_a$ defining the max-ent nodes. The parameter $\beta$ is set to $0.8/h^2$, where $h$ equals the node spacing. The minimizer $\psi$ of Equation (34) with the smoothing parameter $\alpha = 0.01$ is computed by resolving a sparse linear system of $21^3$ equations. The zero-level set $\mathcal{S}$ of the computed solution is shown in Figures 13(b) and (c). The point clouds shown in red, green and blue in Figure 13(d) are partial scans from the *Happy Buddha* data set. The result of registering them onto the zero-level set of the SSD function using Algorithm 1 is shown in Figure 13(e).

Figure 14 shows a similar example in which the reference data set $\mathcal{S}$ is the leaf pictured in Figure 14(a). A point cloud sampling of the surface is measured using a laser-based scanner (Romer Absolute Arm 7320) and is shown in black in Figure 14(b). Following a procedure similar to that in Figures 13(a)–(c), we compute an implicit representation for the leaf surface as depicted in Figures 14(c) and (d). Figures 14(e) and (f) show examples registering a point cloud and a triangulation to the implicit surface using Algorithm 1.

### 5.2. An ICP-type algorithm with exponential coordinates

Rather than approximating the target point cloud $\mathbf{V}$ by an implicit surface, we consider an ICP-like algorithm next to register $\mathbf{U}$ onto $\mathbf{V}$. Specifically, we seek the pose $(\mathbf{R}, \mathbf{t})$ that minimizes the functional

$$\mathrm{J}(\mathbf{Q}, \mathbf{s}) \triangleq \frac{1}{2}\sum_{i=1}^{n} \|\mathbf{Q}\mathbf{u}_i + \mathbf{s} - \mathbf{v}_{i\star}\|^2, \tag{36}$$

where $\mathbf{v}_{i\star}$ is the point in $\mathbf{V}$ closest to $\mathbf{u}_i \in \mathbf{U}$. Equation (36) is identical to the problem considered in ICP algorithms [Kab76]. A closed form solution for Equation (36) is in fact furnished by the singular value decomposition of the cross-correlation matrix of the data sets $\mathbf{U}$ and $\mathbf{V}$ [Ume91]. Here, we show that it can be resolved using local exponential coordinates as well.

**(a)** Input point cloud (in black) from the Happy Buddha data set is immersed in a grid of nodes defining local max-ent shape functions.

**(b)** Zero level set of the computed SSD function.

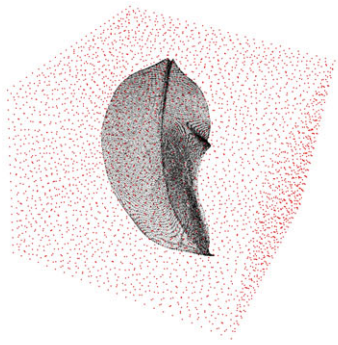
**(c)** Comparison of the input point cloud and its implicit surface representation.

**(d)** Implicit surface and partial scans for the Happy Buddha data set.

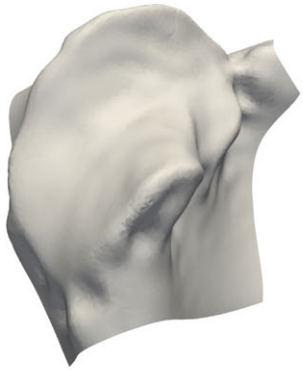
**(e)** Registration of partial scans achieved using Algorithm 1.

**Figure 13:** *Images (a)–(c) illustrate the approximation of an oriented point cloud by an implicit surface specified using an SSD function. Images (d) and (e) show point clouds before and after registration to the surface.*

**(a)** Target surface: a leaf

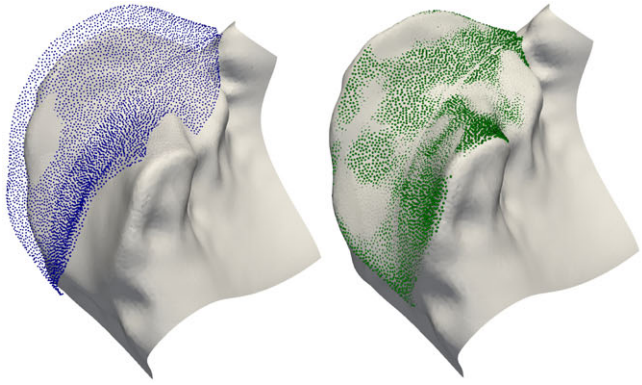


**(b)** Point cloud representation of the leaf measured with a laser scanner is embedded in a grid of max-ent nodes.

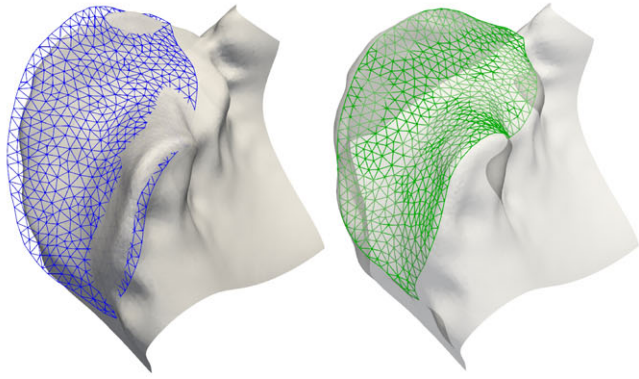


**(c)** Computed implicit surface representation of (d).

**(d)** Comparison of the point sampling and its implicitization.



**(e)** Registration of a point cloud to the leaf surface.



**(f)** Registration of a triangulation to the leaf surface.

**Figure 14:** *The top row of images shows the implicitization of a point cloud sampling of a leaf surface measured with a laser scanner. The bottom row shows the registration of a point cloud and a triangulation onto the computed surface using Algorithm 1.*

With the set of closest point associations between $\mathbf{U}$ and $\mathbf{V}$ held fixed, the stationarity condition satisfied by the pose $(\mathbf{R}, \mathbf{t})$ follows from Proposition 3 as

$$\mathbf{r}(\mathbf{R}, \mathbf{t}) = \sum_{i=1}^{n} \begin{bmatrix} (\mathbf{v}_{i\star} - \mathbf{t}) \times (\mathbf{R}\mathbf{u}_i) \\ (\mathbf{R}\mathbf{u}_i + \mathbf{t} - \mathbf{v}_{i\star}) \end{bmatrix} = 0. \tag{37}$$

It is interesting to note that Equation (37) provides an alternate characterization of the pose update in ICP using a variational approach rather than by introducing cross-correlation matrices and their singular value decompositions. The Jacobian matrix corresponding to the residual in Equation (37) follows from Proposition 4 as

$$\mathbf{K}(\mathbf{R}, \mathbf{t}) = \sum_{i=1}^{n} \begin{bmatrix} -\widehat{(\mathbf{v}_i - \mathbf{t})}\widehat{(\mathbf{R}\mathbf{u}_i)} & \widehat{(\mathbf{R}\mathbf{u}_i)} \\ -\widehat{(\mathbf{R}\mathbf{u}_i)} & \mathbf{I} \end{bmatrix}. \tag{38}$$

It is important to recognize that Equations (37) and (38) only hold with the correspondence between $\mathbf{U}$ and $\mathbf{V}$ held fixed. For this reason, it is necessary to recompute correspondences between $\mathbf{U}$ and $\mathbf{V}$ after each pose update. An ICP-like algorithm now follows. Starting with an initial guess $(\mathbf{R}^0, \mathbf{t}^0)$ for the solution, at the $(k+1)^{\text{th}}$ iteration, we do:

**Algorithm 3**

(i) *Update correspondences*: For each $\mathbf{u}_i \in \mathbf{U}$, find $\mathbf{v}_{i\star} \in \mathbf{V}$ closest to $\mathbf{R}^k\mathbf{u}_i + \mathbf{t}^k$.

(ii) *Assemble*: $\mathbf{K}^k = \mathbf{K}(\mathbf{R}^k, \mathbf{t}^k)$ and $\mathbf{r}^k = \mathbf{r}(\mathbf{R}^k, \mathbf{t}^k)$ using eqs. (37) and (38).

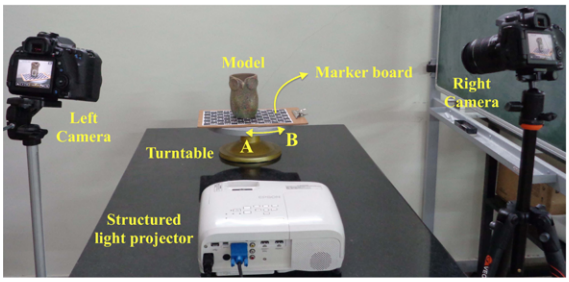
(iii) *Solve*: compute the increments $(\boldsymbol{\Theta}^{k+1}, \mathbf{w}^{k+1})$ satisfying

$$\mathbf{K}^k \begin{bmatrix} \boldsymbol{\Theta}^{k+1} \\ \mathbf{w}^{k+1} \end{bmatrix}_{6\times 1} = -\mathbf{r}^k.$$

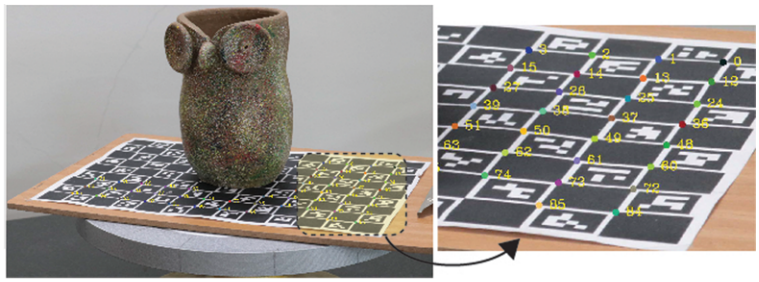
(iv) *Update*: the solution to

$$\mathbf{R}^{k+1} = \mathrm{Exp}_{\mathbf{R}^k}[\boldsymbol{\Theta}^{k+1}] \text{ and } \mathbf{t}^{k+1} = \mathbf{t}^k + \mathbf{w}^{k+1}.$$

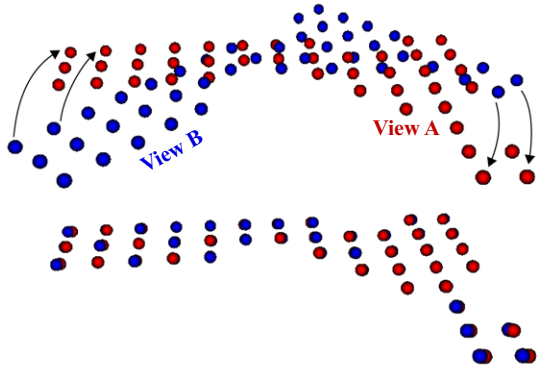
**An example using Algorithm 3 for coarse registration:** In this example, we register point clouds from partial scans produced with a structured light-based stereo imaging setup [LT09]. The scanner, which is shown in Figure 15(a), consists of a pair of calibrated digital cameras and a digital light projector. With the cameras held fixed, the clay vase to be reconstructed is placed on a turntable. By rotating the turntable in small increments and imaging portions of the vase surface falling within the field of view of the scanner (i.e.



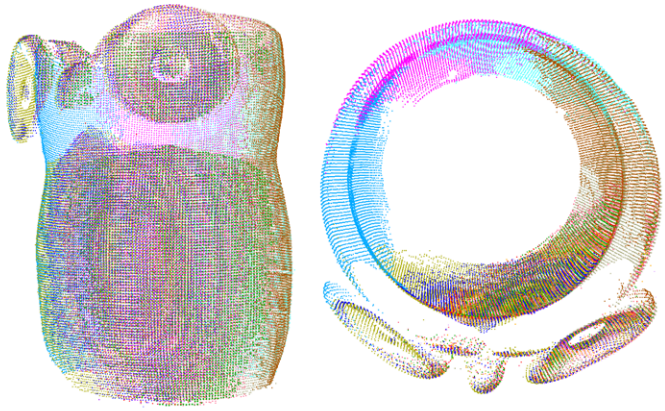

**(a)** Structured light optical scanner.



**(b)** Marker detection for coarse registration.




**(c)** Registering corresponding markers from a pair of views



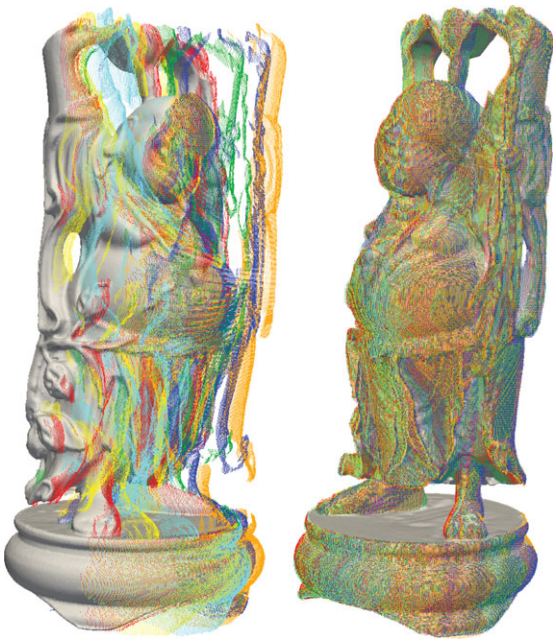
**(d)** Coarse registration of 12 point cloud data sets.

**Figure 15:** *A structured light optical scanner consisting of a pair of digital cameras and a projector is shown in (a). The model to be scanned (a vase) is placed on a chessboard embedded with fiduciary markers shown in (b) to help estimate the motion of the turntable during scanning. We use Algorithm 3 to register marker locations from multiple partial scans; an example is shown in (c). By applying the transformations that align common sets of markers in different pairs of views to these partial scans, we get an approximately registered sampling of the surface. The resulting coarsely registered point clouds are shown in distinct colours in (d).*

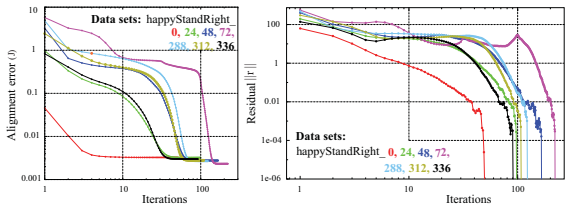
cameras and projector) at each increment, we compute a sequence of point clouds that each sample the object surface. A key requirement for registering these partial scans is a coarse alignment step that maps the measured point clouds to approximately correct locations. In this example, we rely on special fiduciary markers developed in [GJMSMCMJ14] and implemented in [Bra00] to compute coordinate transformations that help to align partial scans. The resulting registration procedure is automatic and does not rely on identifying any features of the object. We also mention that the structured lighting from the projector only serves to define correspondences in images of the scanned surface and is entirely irrelevant for marker identification and triangulation, and hence for the coarse registration problem.

Figure 15(b) shows a representative image recorded by the left camera with the turntable set at orientation B. Notice that the vase sits on a chessboard pattern that rotates together with the turntable. The white squares in the board are embedded with distinct markers that help to robustly identify and enumerate corners in the chessboard. Of course, due to occlusions, only few of the corners in the board can be identified in each view. In this experiment, 55 corners are detected in view A, while 52 are identified in view B. Among these, 43 corners are common to both views A and B. We now triangulate the three-dimensional locations of these 43 corners in view A, which serves as the data set $\mathbf{V}$ for the problem in Equation (37), while the corresponding corner coordinates in view B defines the data set $\mathbf{U}$. The two sets of reconstructed corner locations are shown in Figure 15(c). The distinct identities of the markers automatically define the correspondences between the data sets $\mathbf{U}$ and $\mathbf{V}$. Hence, the correspondence update in Algorithm 3 is performed just once and not at each iteration. Then, starting with the initial guess $(\mathbf{R}^0, \mathbf{t}^0) = (\mathbf{I}, 0)$, Algorithm 3 computes the transformation $(\mathbf{R}, \mathbf{t})$ mapping marker locations in view B to their corresponding locations in view A. Figure 15(c) also shows the final alignment achieved. Applying the coordinate transformation computed by registering markers to the point cloud sampling of the object surface coarsely registers the partial scan from view B to that in view A. Figure 15(d) shows a coarse reconstruction of the vase surface achieved by registering markers from 12 different views and applying the computed transformations to the respective point cloud samples measured in each view. In the figure, point clouds from the different views are shown in distinct colours.

**An example using Algorithm 3 for fine registration:** Figure 16 shows an example using Algorithm 3 to register benchmark data sets for the *Happy Buddha* retrieved from [TL]. We use data sets as is from the repository without any post processing (e.g. outlier removal or smoothing) and register partial scans with a full



**(a)** The data sets in gray, red, green, blue, orange, yellow and cyan correspond to files happy_vrip, happyStandRight_{0,24,48,72,312,336} in the repository at [TL]. The target data set is shown in gray.




**(b)** Details of the iterations to convergence for the error functional and the residual realized in registering the data sets shown in (a).

**Figure 16:** *Fine registration of partial scans for the benchmark example of the 'Happy Buddha' using Algorithm 3.*

reconstruction [CL96]. The specifics chosen for this example are intended to help easily reproduce the calculations reported here without the need for any user defined parameters. Figure 16(a) shows the target data set in gray consisting of approximately 540 k vertices. This serves as the point cloud $\mathbf{V}$ in problem (36). The figure also shows five point clouds in distinct colours representing partial scans — each of these data sets will serve as $\mathbf{U}$. These partial scans are in independent coordinate systems, which is the reason why they appear offset from the target cloud. Figure 16(a) shows the result of registering the partial scans using Algorithm 3. It is worth noting that the registration is successful (i.e. avoids local minima/maxima) despite large errors in the initial poses. Details of the iterations to convergence are reported in Figure 16(b). We find that the error functional is reduced appreciably in each case.

## 6. Concluding Remarks

We introduced the idea of parameterizing isometries over local tangent planes with exponential maps as a potent tool in a variational framework for resolving registration problems. The algorithms proposed here represent clear departures from existing registration methods both in theory and practice. They are capable of handling varied data sets in a unified manner (point clouds, curves and surfaces) and enable adopting flexible error metrics during registration. They seamlessly couple correspondence and pose estimations, and naturally exploit detailed information about target manifolds (normals and curvatures) to achieve accurate registrations. They are straightforward to implement and besides requiring tolerances to detect convergence, do not rely on any heuristic user defined parameters.

A prospective practical application of Algorithm 1 is in the context of automated quality inspection during manufacturing. A point cloud scan of a finished product, measured with a laser scanner for instance, can be registered with its CAD model to identify deviations from prescribed tolerances. Algorithm 1 is particularly appropriate in such applications because it avoids introducing any discretizations of the CAD surface and therefore avoids introducing errors which may be comparable with the geometric defects to be detected in the first place.

We envision the ideas proposed here to be applicable in the more general context of non-rigid registration problems as well. This is because it is algorithmically convenient and practically meaningful to compute non-rigid maps as a composition of simpler transformations, including rigid transformations. As a particular example, we refer to [DZYL10] where affine transformations are computed in such a manner. We also expect that the flexibility in choosing objective functionals afforded by the proposed algorithms will help to incorporate additional information such as textures during registration.

One of the features common to all the algorithms we have discussed is that they achieve registration by resolving a set of stationarity conditions, which are a set of nonlinear equations. The success of these algorithms is therefore subject to the usual intricacies inherent in solving nonlinear systems. For instance, Newton iterations may fail to converge if the initial guess is far away from the correct solution. For the same reason, it is also possible that the algorithms identify solutions that are stationary points but not the global minimizer. These solutions are characterized by the residual being approximately zero despite the error functional remaining large. In such scenarios, it is necessary to either interactively improve the initial guess for the solution or resort to schemes that systematically sample the solution space [MPD06, YLJ13].

A clear benefit of computing consistent linearizations of stationarity conditions is the rapid convergence of Newton iterations close to the optimal solution. Almost without exception, all the examples presented here required a dozen or fewer iterations to compute the optimal solution. Jacobian calculations are also useful for an entirely different reason — when singular, its eigenvectors corresponding to null eigenvalues reveal symmetries in data sets [SAD*16]. This may include translational periodicity, rotational invariances or a combination of both. A consequence of the specific parameterization we have adopted for rotations is that the Jacobian matrices computed during Newton iterations are not expected to be symmetric except at the converged solution. Specifically, the $3 \times 3$ block denoted by $\mathbf{K}^{\theta\theta}$ is not symmetric in general. For our purposes requiring the

resolution of small linear systems (6 dofs in the case of isometries), such a lack of symmetry poses no difficulties in practice.

There are of course numerous aspects of registration problems which are important in practice but have been neglected here. This includes issues of incorporating models for noise and measurement errors into the problem definition. A detailed comparison of the algorithms introduced here with more alternatives in the literature is also important [SMFF07]. Finally, we note that our discussions and examples have all been limited to pairwise registration of data sets. Multi-view registration requires additional considerations to avoid accumulation of errors as discussed in [Mas02].

## Appendix A: Stationarity Conditions and their Linearizations

**Proposition 1** (Registering point cloud $\mathbf{U}$ to the implicit surface $\mathcal{S}$)**.** *A minimizer* $(\mathbf{R}, \mathbf{t})$ *of the problem*

$$\text{Find} \quad (\mathbf{R}, \mathbf{t}) \triangleq \textit{arg min}_{(\mathbf{Q},\mathbf{s})\in \mathrm{SO}_3\times\mathbb{R}^3} \mathrm{J_p}(\mathbf{Q}, \mathbf{s}), \tag{A.1}$$

*where* $\mathrm{J_p} : \mathrm{SO}_3 \times \mathbb{R}^3 \to \mathbb{R}$ *is the functional*

$$\mathrm{J_p}(\mathbf{Q}, \mathbf{s}) \triangleq \frac{1}{2}\sum_{i=1}^{n} \psi^2(\mathbf{Q}\mathbf{u}_i + \mathbf{s})$$

*satisfies the stationarity conditions*

$$\sum_{i=1}^{n} \begin{bmatrix} (\mathbf{R}\mathbf{u}_i) \times \psi\nabla\psi(\mathbf{R}\mathbf{u}_i + \mathbf{t}) \\ \psi\nabla\psi(\mathbf{R}\mathbf{u}_i + \mathbf{t}) \end{bmatrix} = 0. \tag{A.2}$$

*Proof.* Recalling that $\mathbf{R}_\varepsilon = \mathrm{Exp}_{\mathbf{R}}[\varepsilon\hat{\mathbf{\Theta}}]$ and using the shorthand $d\mathbf{R}_\varepsilon/d\varepsilon = \mathbf{R}'_\varepsilon$, we evaluate the variation of $\mathrm{J_p}$ with respect to admissible variations of $\mathbf{R}$ as

$$\begin{aligned}
\langle \delta \mathrm{J_p}(\mathbf{R}, \mathbf{t}), \hat{\mathbf{\Theta}}\rangle &\triangleq \lim_{\varepsilon\to 0} \frac{d}{d\varepsilon}\frac{1}{2}\sum_{i=1}^{n} \psi^2(\mathbf{R}_\varepsilon\mathbf{u}_i + \mathbf{t}), \\
&= \lim_{\varepsilon\to 0} \sum_{i=1}^{n} \psi\nabla\psi(\mathbf{R}_\varepsilon\mathbf{u}_i + \mathbf{t}) \cdot \mathbf{R}'_\varepsilon\mathbf{u}_i && \text{(A.3a)} \\
&= \sum_{i=1}^{n} \psi\nabla\psi(\mathbf{R}\mathbf{u}_i + \mathbf{t}) \cdot \left(\hat{\mathbf{\Theta}}\mathbf{R}\mathbf{u}_i\right) && \text{(A.3b)} \\
&= \sum_{i=1}^{n} \psi\nabla\psi(\mathbf{R}\mathbf{u}_i + \mathbf{t}) \cdot (\mathbf{\Theta} \times (\mathbf{R}\mathbf{u}_i)) && \text{(A.3c)} \\
&= \sum_{i=1}^{n} (\mathbf{R}\mathbf{u}_i \times (\psi\nabla\psi(\mathbf{R}\mathbf{u}_i + \mathbf{t}))) \cdot \mathbf{\Theta}, && \text{(A.3d)}
\end{aligned}$$

where we have used Equation (9) in Equation (A.3a), the definition from Equation (4) in Equation (A.3b) and the invariance of scalar triple products under circular shifts in Equation (A.3c). Similarly, the sensitivity of $\mathrm{J_p}$ to admissible variations of the translation $\mathbf{t}$ is computed as

$$\begin{aligned}
\langle \delta \mathrm{J_p}(\mathbf{R}, \mathbf{t}), \mathbf{w}\rangle &\triangleq \lim_{\varepsilon\to 0} \frac{d}{d\varepsilon}\frac{1}{2}\sum_{i=1}^{n} \psi^2(\mathbf{R}\mathbf{u}_i + \mathbf{t} + \varepsilon\mathbf{w}) \\
&= \sum_{i=1}^{n} \psi\nabla\psi(\mathbf{R}\mathbf{u}_i + \mathbf{t}) \cdot \mathbf{w}. && \text{(A.4)}
\end{aligned}$$

As expected, Equations (A.3) and (A.4) are linear in the arbitrarily chosen vectors $\mathbf{w}, \mathbf{\Theta} \in \mathbb{R}^3$. Setting the computed variations to zero and invoking the arbitrariness of $\mathbf{w}$ and $\mathbf{\Theta}$ yields Equation (A.2). ∎

**Proposition 2** (Linearization of stationarity conditions in Proposition 1)**.** *The linearization of*

$$\mathbf{r}(\mathbf{R}, \mathbf{t}) \triangleq \begin{bmatrix} \mathbf{r}_\theta(\mathbf{R}, \mathbf{t}) \\ \mathbf{r}_\mathbf{t}(\mathbf{R}, \mathbf{t}) \end{bmatrix} = \sum_{i=1}^{n} \begin{bmatrix} (\mathbf{R}\mathbf{u}_i) \times \psi\nabla\psi(\mathbf{R}\mathbf{u}_i + \mathbf{t}) \\ \psi\nabla\psi(\mathbf{R}\mathbf{u}_i + \mathbf{t}) \end{bmatrix} \tag{A.5}$$

*at the configuration* $(\mathbf{R}, \mathbf{t})$ *along* $(\hat{\mathbf{\Theta}}, \mathbf{w})$ *is given by*

$$\mathrm{L}_{(\mathbf{R},\mathbf{t})}\mathbf{r}(\hat{\mathbf{\Theta}}, \mathbf{w}) = \mathbf{r}(\mathbf{R}, \mathbf{t}) + \mathbf{K}(\mathbf{R}, \mathbf{t})\begin{bmatrix} \mathbf{\Theta} \\ \mathbf{w} \end{bmatrix}, \tag{A.6}$$

*where*

$$\mathbf{K}(\mathbf{R}, \mathbf{t}) = \sum_{i=1}^{n} \begin{bmatrix} \mathbf{K}_i^{\theta\theta} & \mathbf{K}_i^{\theta\mathbf{t}} \\ \mathbf{K}_i^{\mathbf{t}\theta} & \mathbf{K}_i^{\mathbf{t}\mathbf{t}} \end{bmatrix}_{6\times 6}\Bigg|_{\mathbf{R}\mathbf{u}_i+\mathbf{t}}, \tag{A.7}$$

$$\begin{aligned}
\textit{with:} \quad \mathbf{K}_i^{\mathbf{t}\mathbf{t}} &= \nabla\psi \otimes \nabla\psi + \psi\nabla\nabla\psi, \\
\mathbf{K}_i^{\theta\mathbf{t}} &= \widehat{(\mathbf{R}\mathbf{u}_i)}\,\mathbf{K}_i^{\mathbf{t}\mathbf{t}} \\
\mathbf{K}_i^{\mathbf{t}\theta} &= \left(\mathbf{K}_i^{\theta\mathbf{t}}\right)^t, \\
\textit{and} \quad \mathbf{K}_i^{\theta\theta} &= \left(\widehat{(\psi\nabla\psi)} - \widehat{(\mathbf{R}\mathbf{u}_i)}\mathbf{K}_i^{\mathbf{t}\mathbf{t}}\right)\widehat{(\mathbf{R}\mathbf{u}_i)}.
\end{aligned}$$

*Proof.* For the sake of notational simplicity, we omit the sum appearing in the expressions for $\mathbf{r}$ and $\mathbf{K}$ since the index $i$ is irrelevant in the proof. Instead we will denote a generic point $\mathbf{u}_i$ as $\mathbf{u}$ and assume $\mathbf{r}$ to be simply given by

$$\mathbf{r}(\mathbf{R}, \mathbf{t}) = \begin{bmatrix} \mathbf{r}_\theta \\ \mathbf{r}_\mathbf{t} \end{bmatrix} = \begin{bmatrix} (\mathbf{R}\mathbf{u}) \times \psi\nabla\psi \\ \psi\nabla\psi \end{bmatrix}\Bigg|_{\mathbf{R}\mathbf{u}+\mathbf{t}}.$$

Computing the directional derivatives of $\mathbf{r}$ along admissible variations $\mathbf{w}$ of $\mathbf{t}$, we get

$$\begin{aligned}
\mathrm{D}\mathbf{r}_\mathbf{t}(\mathbf{R}, \mathbf{t}) \cdot \mathbf{w} &= \lim_{\varepsilon\to 0} \frac{d}{d\varepsilon}\left(\psi\nabla\psi(\mathbf{R}\mathbf{u} + \mathbf{t} + \varepsilon\mathbf{w})\right) \\
&= \left(\nabla\psi(\nabla\psi \cdot \mathbf{w}) + \psi(\nabla\nabla\psi\ \mathbf{w})\right)\Big|_{\mathbf{R}\mathbf{u}+\mathbf{t}} \\
&= \underbrace{\left(\nabla\psi \otimes \nabla\psi + \psi\nabla\nabla\psi\right)\Big|_{\mathbf{R}\mathbf{u}+\mathbf{t}}}_{\mathbf{K}^{\mathbf{t}\mathbf{t}}}\mathbf{w},
\end{aligned}$$

which proves the required expression for $\mathbf{K}^{\mathbf{t}\mathbf{t}}$. Similarly,

$$\begin{aligned}
\mathrm{D}\mathbf{r}_\theta(\mathbf{R}, \mathbf{t}) \cdot \mathbf{w} &= \lim_{\varepsilon\to 0} \frac{d}{d\varepsilon}\left((\mathbf{R}\mathbf{u}) \times \psi\nabla\psi(\mathbf{R}\mathbf{u} + \mathbf{t} + \varepsilon\mathbf{w})\right) \\
&= (\mathbf{R}\mathbf{u}) \times \left(\lim_{\varepsilon\to 0} \frac{d}{d\varepsilon}\left(\psi\nabla\psi(\mathbf{R}\mathbf{u} + \mathbf{t} + \varepsilon\mathbf{w})\right)\right) \\
&= (\mathbf{R}\mathbf{u}) \times \left(\mathbf{K}^{\mathbf{t}\mathbf{t}}\mathbf{w}\right)\Big|_{\mathbf{R}\mathbf{u}+\mathbf{t}} \\
&= \underbrace{\widehat{(\mathbf{R}\mathbf{u})}\,\mathbf{K}^{\mathbf{t}\mathbf{t}}}_{\mathbf{K}^{\theta\mathbf{t}}}\Big|_{\mathbf{R}\mathbf{u}+\mathbf{t}}\mathbf{w},
\end{aligned}$$

proving the expression for $\mathbf{K}^{\theta\mathbf{t}}$. Next, computing derivatives along admissible variations of the rotation $\mathbf{R}$, we have

$$\begin{aligned}
\mathrm{D}\mathbf{r}_{\mathbf{t}}(\mathbf{R},\mathbf{t})\cdot\boldsymbol{\Theta} &= \lim_{\varepsilon\to 0}\frac{d}{d\varepsilon}\left(\psi\nabla\psi(\mathbf{R}_\varepsilon\mathbf{u}+\mathbf{t})\right)\\
&= \left(\nabla\psi(\nabla\psi\cdot\hat{\boldsymbol{\Theta}}\mathbf{R}\mathbf{u})+\psi\nabla\nabla\psi(\hat{\boldsymbol{\Theta}}\mathbf{R}\mathbf{u})\right)\Big|_{\mathbf{R}\mathbf{u}+\mathbf{t}}\\
&= \underbrace{\left(\nabla\psi\otimes\nabla\psi+\psi\nabla\nabla\psi\right)}_{\mathbf{K}^{\mathbf{tt}}}\Big|_{\mathbf{R}\mathbf{u}+\mathbf{t}}(\hat{\boldsymbol{\Theta}}\mathbf{R}\mathbf{u})\\
&= \mathbf{K}^{\mathbf{tt}}(\boldsymbol{\Theta}\times\mathbf{R}\mathbf{u})\Big|_{\mathbf{R}\mathbf{u}+\mathbf{t}}\\
&= -\mathbf{K}^{\mathbf{tt}}(\widehat{\mathbf{R}\mathbf{u}})\Big|_{\mathbf{R}\mathbf{u}+\mathbf{t}}\boldsymbol{\Theta}.\\
&= \underbrace{(\widehat{\mathbf{R}\mathbf{u}})\,\mathbf{K}^{\mathbf{tt}})^t}_{(\mathbf{K}^{\theta\mathbf{t}})^t=\mathbf{K}^{\mathbf{t}\theta}}\Big|_{\mathbf{R}\mathbf{u}+\mathbf{t}}\boldsymbol{\Theta},
\end{aligned}$$

where in the last step we have used the facts that $\mathbf{K}^{\mathbf{tt}}$ is symmetric and $(\widehat{\mathbf{R}\mathbf{u}})$ is skew. Finally,

$$\begin{aligned}
\mathrm{D}\mathbf{r}_{\theta}(\mathbf{R},\mathbf{t})\cdot\boldsymbol{\Theta} &= \lim_{\varepsilon\to 0}\frac{d}{d\varepsilon}(\mathbf{R}_\varepsilon\mathbf{u}\times\psi\nabla\psi(\mathbf{R}_\varepsilon\mathbf{u}+\mathbf{t}))\\
&= (\hat{\boldsymbol{\Theta}}\mathbf{R}\mathbf{u}\times\psi\nabla\psi+\mathbf{R}\mathbf{u}\times\mathbf{K}^{\mathbf{t}\theta}\boldsymbol{\Theta})\Big|_{\mathbf{R}\mathbf{u}+\mathbf{t}}\\
&= ((\boldsymbol{\Theta}\times\mathbf{R}\mathbf{u})\times\psi\nabla\psi-\mathbf{R}\mathbf{u}\times\mathbf{K}^{\mathbf{tt}}(\widehat{\mathbf{R}\mathbf{u}})\boldsymbol{\Theta})\Big|_{\mathbf{R}\mathbf{u}+\mathbf{t}}\\
&= (\psi\nabla\psi\times(\mathbf{R}\mathbf{u}\times\boldsymbol{\Theta})-(\widehat{\mathbf{R}\mathbf{u}})\mathbf{K}^{\mathbf{tt}}(\widehat{\mathbf{R}\mathbf{u}}))\Big|_{\mathbf{R}\mathbf{u}+\mathbf{t}}\\
&= ((\widehat{\psi\nabla\psi})(\widehat{\mathbf{R}\mathbf{u}})-(\widehat{\mathbf{R}\mathbf{u}})\mathbf{K}^{\mathbf{tt}}(\widehat{\mathbf{R}\mathbf{u}}))\Big|_{\mathbf{R}\mathbf{u}+\mathbf{t}}\boldsymbol{\Theta}\\
&= \underbrace{((\widehat{\psi\nabla\psi})-(\widehat{\mathbf{R}\mathbf{u}})\mathbf{K}^{\mathbf{tt}})(\widehat{\mathbf{R}\mathbf{u}})}_{\mathbf{K}^{\theta\theta}}\Big|_{\mathbf{R}\mathbf{u}+\mathbf{t}}\boldsymbol{\Theta},
\end{aligned}$$

yielding the required expression for $\mathbf{K}^{\theta\theta}$. ■

**Proposition 3** (Pairwise registration of point clouds $\mathbf{U}$ to $\mathbf{V}$)**.** *Let $\mathbf{U}=\{\mathbf{u}_i\}_{i=1}^n$ and $\mathbf{V}=\{\mathbf{v}_i\}_{i=1}^m$ be a pair of point clouds. For each $\mathbf{u}_i\in\mathbf{U}$, denote the point in $\mathbf{V}$ closest to $\mathbf{u}_i$ by $\mathbf{v}_{i\star}$. Then, a stationary point $(\mathbf{R},\mathbf{t})$ of the functional*

$$\mathrm{J}(\mathbf{Q},\mathbf{s})\triangleq\frac{1}{2}\sum_{i=1}^{n}\|\mathbf{Q}\mathbf{u}_i+\mathbf{s}-\mathbf{v}_{i\star}\|^2$$

*satisfies the stationarity conditions*

$$\mathbf{r}(\mathbf{R},\mathbf{t})\triangleq\begin{bmatrix}\mathbf{r}_\theta(\mathbf{R},\mathbf{t})\\ \mathbf{r}_t(\mathbf{R},\mathbf{t})\end{bmatrix}=\sum_{i=1}^{n}\begin{bmatrix}(\mathbf{v}_{i\star}-\mathbf{t})\times(\mathbf{R}\mathbf{u}_i)\\ (\mathbf{R}\mathbf{u}_i+\mathbf{t}-\mathbf{v}_{i\star})\end{bmatrix}=0. \quad \text{(A.8)}$$

*Proof.* Adopting the shorthands $\mathbf{R}_\varepsilon=\mathrm{Exp}_{\mathbf{R}}[\varepsilon\hat{\boldsymbol{\Theta}}]$, $\mathbf{R}'_\varepsilon=d\mathbf{R}_\varepsilon/d\varepsilon$, we have

$$\begin{aligned}
\langle\delta\mathrm{J}(\mathbf{R},\mathbf{t}),\hat{\boldsymbol{\Theta}}\rangle &= \lim_{\varepsilon\to 0}\frac{d}{d\varepsilon}\sum_{i=1}^{n}\frac{1}{2}\|\mathbf{R}_\varepsilon\mathbf{u}_i+\mathbf{t}-\mathbf{v}_{i\star}\|^2\\
&= \sum_{i=1}^{n}\lim_{\varepsilon\to 0}(\mathbf{R}_\varepsilon\mathbf{u}_i+\mathbf{t}-\mathbf{v}_{i\star})\cdot(\mathbf{R}'_\varepsilon\mathbf{u}_i)\\
&= \sum_{i=1}^{n}(\mathbf{R}\mathbf{u}_i+\mathbf{t}-\mathbf{v}_{i\star})\cdot(\hat{\boldsymbol{\Theta}}\mathbf{R}\mathbf{u}_i) \quad \text{(A.9a)}\\
&= \sum_{i=1}^{n}(\mathbf{R}\mathbf{u}_i+\mathbf{t}-\mathbf{v}_{i\star})\cdot(\boldsymbol{\Theta}\times(\mathbf{R}\mathbf{u}_i)) \quad \text{(A.9b)}\\
&= \sum_{i=1}^{n}\boldsymbol{\Theta}\cdot((\mathbf{R}\mathbf{u}_i)\times(\mathbf{R}\mathbf{u}_i+\mathbf{t}-\mathbf{v}_{i\star})), \quad \text{(A.9c)}\\
&= \sum_{i=1}^{n}\boldsymbol{\Theta}\cdot((\mathbf{v}_{i\star}-\mathbf{t})\times(\mathbf{R}\mathbf{u}_i)),
\end{aligned}$$

where we have invoked Equation (9) in Equation (A.9a), the definition from Equation (4) in Equation (A.9b) and the invariance of scalar triple products under circular shifts in Equation (A.9c).

Evaluating the variation of J with respect to translational dofs is more straightforward. We have

$$\begin{aligned}
\langle\delta\mathrm{J}(\mathbf{R},\mathbf{t}),\mathbf{w}\rangle &= \lim_{\varepsilon\to 0}\frac{d}{d\varepsilon}\frac{1}{2}\sum_{i=1}^{n}\|\mathbf{R}\mathbf{u}_i+\mathbf{t}+\varepsilon\mathbf{w}-\mathbf{v}_{i\star}\|^2\\
&= \sum_{i=1}^{n}(\mathbf{R}\mathbf{u}_i+\mathbf{t}-\mathbf{v}_{i\star})\cdot\mathbf{w}. \quad \text{(A.10)}
\end{aligned}$$

Setting the variations in Equations (A.9) and (A.10) to zero and invoking the arbitrariness of $\boldsymbol{\Theta},\mathbf{w}\in\mathbb{R}^3$ yields Equation (A.8). ■

**Proposition 4** (Linearization of stationarity conditions in Proposition 3)**.** *The linearization of*

$$\mathbf{r}(\mathbf{R},\mathbf{t})\triangleq\begin{bmatrix}\mathbf{r}_\theta(\mathbf{R},\mathbf{t})\\ \mathbf{r}_\mathbf{t}(\mathbf{R},\mathbf{t})\end{bmatrix}=\sum_{i=1}^{n}\begin{bmatrix}(\mathbf{v}_{i\star}-\mathbf{t})\times(\mathbf{R}\mathbf{u}_i)\\ (\mathbf{R}\mathbf{u}_i+\mathbf{t}-\mathbf{v}_{i\star})\end{bmatrix} \quad \text{(A.11)}$$

*at the pose $(\mathbf{R},\mathbf{t})$ along $(\hat{\boldsymbol{\Theta}},\mathbf{w})$ is given by*

$$\mathrm{L}_{(\mathbf{R},\mathbf{t})}\mathbf{r}(\hat{\boldsymbol{\Theta}},\mathbf{w})=\mathbf{r}(\mathbf{R},\mathbf{t})+\mathbf{K}(\mathbf{R},\mathbf{t})\begin{bmatrix}\boldsymbol{\Theta}\\ \mathbf{w}\end{bmatrix},$$

$$\textit{where}\quad \mathbf{K}(\mathbf{R},\mathbf{t})=\sum_{i=1}^{n}\begin{bmatrix}-(\widehat{\mathbf{v}_{i\star}-\mathbf{t}})(\widehat{\mathbf{R}\mathbf{u}_i}) & (\widehat{\mathbf{R}\mathbf{u}_i})\\ -(\widehat{\mathbf{R}\mathbf{u}_i}) & \mathbf{I}\end{bmatrix}. \quad \text{(A.12)}$$

*Proof.* For the sake of notational simplicity, we omit the sum appearing in the expressions for $\mathbf{r}$ and $\mathbf{K}$ since the index $i$ is irrelevant in the proof. Instead we will denote a generic corresponding point pair $\mathbf{u}_i,\mathbf{v}_{i\star}$ simply as $\mathbf{u},\mathbf{v}_\star$ and assume that

$$\mathbf{r}(\mathbf{R},\mathbf{t})\triangleq\begin{bmatrix}\mathbf{r}_\theta(\mathbf{R},\mathbf{t})\\ \mathbf{r}_\mathbf{t}(\mathbf{R},\mathbf{t})\end{bmatrix}=\begin{bmatrix}(\mathbf{v}_\star-\mathbf{t})\times(\mathbf{R}\mathbf{u})\\ (\mathbf{R}\mathbf{u}+\mathbf{t}-\mathbf{v}_\star)\end{bmatrix}.$$

Proceeding to compute the block matrices in Equation (A.12) constituting $\mathbf{K}$, we have:

$$\begin{aligned}\mathbf{K}^{\theta\theta}\boldsymbol{\Theta} &= \lim_{\varepsilon\to 0}\frac{d}{d\varepsilon}\left((\mathbf{v}_\star-\mathbf{t})\times(\mathbf{R}_\varepsilon\mathbf{u})\right)\\ &= \lim_{\varepsilon\to 0}(\mathbf{v}_\star-\mathbf{t})\times(\mathbf{R}'_\varepsilon\mathbf{u})\\ &= (\mathbf{v}_\star-\mathbf{t})\times(\hat{\boldsymbol{\Theta}}\mathbf{R}\mathbf{u})\\ &= -\widehat{(\mathbf{v}_\star-\mathbf{t})}\widehat{(\mathbf{R}\mathbf{u})}\boldsymbol{\Theta}. \end{aligned} \tag{A.13a}$$

$$\begin{aligned}\mathbf{K}^{\theta\mathbf{t}}\mathbf{w} &= \lim_{\varepsilon\to 0}\frac{d}{d\varepsilon}\left((\mathbf{v}_\star-\mathbf{t}-\varepsilon\mathbf{w})\times(\mathbf{R}\mathbf{u})\right)\\ &= -\mathbf{w}\times(\mathbf{R}\mathbf{u}) = \widehat{(\mathbf{R}\mathbf{u})}\mathbf{w}. \end{aligned} \tag{A.13b}$$

$$\begin{aligned}\mathbf{K}^{\mathbf{t}\theta}\boldsymbol{\Theta} &= \lim_{\varepsilon\to 0}\frac{d}{d\varepsilon}\left(\mathbf{R}_\varepsilon\mathbf{u}+\mathbf{t}-\mathbf{v}_\star\right)\\ &= \lim_{\varepsilon\to 0}\mathbf{R}'_\varepsilon\mathbf{u} = \hat{\boldsymbol{\Theta}}\mathbf{R}\mathbf{u} = -\widehat{(\mathbf{R}\mathbf{u})}\boldsymbol{\Theta}, \end{aligned} \tag{A.13c}$$

$$\text{and}\quad \mathbf{K}^{\mathbf{tt}}\mathbf{w} = \lim_{\varepsilon\to 0}\frac{d}{d\varepsilon}(\mathbf{R}\mathbf{u}+\mathbf{t}+\varepsilon\mathbf{w}-\mathbf{v}_\star) = \mathbf{I}\,\mathbf{w}. \tag{A.13d}$$

Equation (A.12) now follows from Equation (A.13). ■

## References


[Agr06] AGRAWAL M.: A lie algebraic approach for consistent pose registration for general Euclidean motion. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2006* (NJ, USA, 2006), IEEE, pp. 1891–1897.

[AHB87] ARUN K. S., HUANG T. S., BLOSTEIN S. D.: Least-squares fitting of two 3-D point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *5* (1987), 698–700.

[Ale02] ALEXA M.: Linear combination of transformations. In *ACM Transactions on Graphics (TOG)* (2002), ACM, New York, vol. 21, pp. 380–387.

[AMCO08] AIGER D., MITRA N. J., COHEN-OR D.: 4-points congruent sets for robust surface registration. *ACM Transactions on Graphics 27*, 3 (2008), #85, 1–10.

[AMS10] ABSIL P.-A., MAHONY R., SEPULCHRE R.: Optimization on manifolds: Methods and applications. In *Recent Advances in Optimization and its Applications in Engineering*. M. Diehl, F. Glineur, E. Jarlebring and W. Michiels (Eds.). Springer, Berlin (2010), pp. 125–144.

[AO06] ARROYO M., ORTIZ M.: Local maximum-entropy approximation schemes: A seamless bridge between finite elements and meshfree methods. *International Journal for Numerical Methods in Engineering 65*, 13 (2006), 2167–2202.

[BC11] BOGDAN R., COUSINS S.: 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)* (Shanghai, China, 2011).

[BM*92] BESL P. J., MCKAY N. D.: A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence 14*, 2 (1992), 239–256.

[BMP02] BELONGIE S., MALIK J., PUZICHA J.: Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 4 (2002), 509–522.

[Bra00] BRADSKI G.: The OpenCV Library. *Dr. Dobb's Journal of Software Tools 25*, (2000).

[BSB*15] BELLEKENS B., SPRUYT V., BERKVENS R., PENNE R., WEYN M.: A benchmark survey of rigid 3-D point cloud registration algorithm. *International Journal of Advanced Intelligent Systems 8* (2015), 118–127.

[CBC*01] CARR J. C., BEATSON R. K., CHERRIE J. B., MITCHELL T. J., FRIGHT W. R., MCCALLUM B. C., EVANS T. R.: Reconstruction and representation of 3-D objects with radial basis functions. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (New York, 2001), ACM, pp. 67–76.

[CCC*08] CIGNONI P., CALLIERI M., CORSINI M., DELLEPIANE M., GANOVELLI F., RANZUGLIA G.: MeshLab: An open-source mesh processing tool. In *Eurographics Italian Chapter Conference*. S. Vittorio, C R., E U. (Eds.), The Eurographics Association (2008), pp. 129–136.

[CL96] CURLESS B., LEVOY M.: A volumetric method for building complex models from range images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (New York, USA, 1996), ACM, pp. 303–312.

[clo18] Cloudcompare (ver. 2.8.1). http://www.cloudcompare.org/ (2018). Last accessed on August 25, 2018.

[CM92] CHEN Y., MEDIONI G.: Object modelling by registration of multiple range images. *Image and Vision Computing 10*, 3 (1992), 145–155.

[Con] CONSORTIUM D.: Virtual math museum. http://virtualmathmuseum.org/index.html. Last accessed on June 13, 2018.

[CR03] CHUI H., RANGARAJAN A.: A new point matching algorithm for non-rigid registration. *Computer Vision and Image Understanding 89*, 2–3 (2003), 114–141.

[CRZL04] CHUI H., RANGARAJAN A., ZHANG J., LEONARD C.: Unsupervised learning of an atlas from unlabeled point-sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence 26*, 2 (2004), 160–172.

[CSK05] CHETVERIKOV D., STEPANOV D., KRSEK P.: Robust Euclidean alignment of 3D point sets: The trimmed iterative closest point algorithm. *Image and Vision Computing 23*, 3 (2005), 299–309.

[CT11] CALAKLI F., TAUBIN G.: SSD: Smooth signed distance surface reconstruction. In *Computer Graphics Forum* (2011), Wiley Online Library, vol. *30*, pp. 1993–2002.

[DZX*10] Du S., Zheng N., Xiong L., Ying S., Xue J.: Scaling iterative closest point algorithm for registration of m–d point sets. *Journal of Visual Communication and Image Representation 21*, 5–6 (2010), 442–452.

[DZYL10] Du S., Zheng N., Ying S., Liu J.: Affine iterative closest point algorithm for point set registration. *Pattern Recognition Letters 31*, 9 (2010), 791–799.

[Ead17] Eade E.: Lie groups for 2D and 3D transformations. http://ethaneade.com/lie.pdf, revised May (2017). Last accessed on August 25, 2018.

[EAS98] Edelman A., Arias T. A., Smith S. T.: The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications 20*, 2 (1998), 303–353.

[ELF97] Eggert D. W., Lorusso A., Fisher R. B.: Estimating 3-D rigid body transformations: A comparison of four major algorithms. *Machine Vision and Applications 9*, 5–6 (1997), 272–290.

[FH86] Faugeras O. D., Hebert M.: The representation, recognition, and locating of 3-D objects. *International Journal of Robotics Research 5*, 3 (1986), 27–52.

[Fit03] Fitzgibbon A. W.: Robust registration of 2-D and 3-D point sets. *Image and Vision Computing 21*, 13 (2003), 1145–1153.

[Gal11] Gallier J.: *Geometric Methods and Applications: For Computer Science and Engineering*. Vol. *38*. Springer Science & Business Media, New York, 2011.

[GJMSMCMJ14] Garrido-Jurado S., Muñoz-Salinas R., Madrid-Cuevas F. J., Marín-Jiménez M. J.: Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition 47*, 6 (2014), 2280–2292.

[Gou09] Gough B.: *GNU scientific library reference manual*. Network Theory Ltd. http://www.gnu.org/software/gsl/ (2009). Last accessed on August 25, 2018.

[GP02] Granger S., Pennec X.: Multi-scale EM-ICP: A fast and robust approach for surface registration. In *European Conference on Computer Vision* (2002), Springer, Berlin, Heidelberg, pp. 418–432.

[Gra98] Grassia F. S.: Practical parameterization of rotations using the exponential map. *Journal of Graphics Tools 3*, 3 (1998), 29–48.

[GX03] Gallier J., Xu D.: Computing exponentials of skew-symmetric matrices and logarithms of orthogonal matrices. *International Journal of Robotics and Automation 18*, 1 (2003), 10–20.

[HFY*11] Horaud R., Forbes F., Yguel M., Dewaele G., Zhang J.: Rigid and articulated point registration with expectation conditional maximization. *IEEE Transactions on Pattern Analysis and Machine Intelligence 33*, 3 (2011), 587–602.

[Hor87] Horn B. K.: Closed-form solution of absolute orientation using unit quaternions. *JOSA A 4*, 4 (1987), 629–642.

[Inc] Inc. W. R.: Mathematica, Version 11.3. Champaign, IL, 2018.

[JV05] Jian B., Vemuri B.: A robust algorithm for point set registration using mixture of Gaussians. In *10th IEEE International Conference on Computer Vision, 2005*. (2005), IEEE, IEEE Computer Society, NJ, vol. *2*, pp. 1246–1251.

[Kab76] Kabsch W.: A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography 32*, 5 (1976), 922–923.

[Kan94] Kanatani K.: Analysis of 3-D rotation fitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence 16*, 5 (1994), 543–549.

[KI15] Krivoshapko S., Ivanov V.: *Encyclopedia of Analytical Surfaces*. Springer International Publishing, Switzerland, 2015.

[KLMV05] Krishnan S., Lee P. Y., Moore J. B., Venkatasubramanian S.: Global registration of multiple 3D point sets via optimization-on-a-manifold. In *Symposium on Geometry Processing* (2005), pp. 187–196.

[KSS*17] Kolagunda A., Sorensen S., Saponaro P., Treible W., Kambhamettu C.: Robust shape registration using fuzzy correspondences. arXiv preprint arXiv:1702.05664 (2017).

[Liu07] Liu Y.: A mean field annealing approach to accurate free form shape matching. *Pattern Recognition 40*, 9 (2007), 2418–2436.

[LT09] Lanman D., Taubin G.: Build your own 3-D scanner: 3-D photography for beginners. In *ACM SIGGRAPH 2009 Courses* (2009), ACM, New York, p. 8.

[MAM14] Mellado N., Aiger D., Mitra N.: Super 4PCS fast global pointcloud registration via smart indexing. *Computer Graphics Forum 33*, 5 (2014), 205–215.

[Mar99] Marthinsen A.: Interpolation in lie groups. *SIAM Journal on Numerical Analysis 37*, 1 (1999), 269–285.

[Mas02] Masuda T.: Registration and integration of multiple range images by matching signed distance fields for object shape modeling. *Computer Vision and Image Understanding 87*, 1–3 (2002), 51–65.

[mat18] Matlab computer vision system toolbox (ver. r2018a). https://in.mathworks.com/help/vision/ (2018). Last accessed on August 25, 2018.

[Mel] Mellado N.: 4PCS and super4PCS (super4pcs-v1.1.3-osx-clang-703.zip). https://github.com/nmellado/Super4PCS/releases. Accessed on August 25, 2018.

[MGPG04] Mitra N., Gelfand N., Pottmann H., Guibas L.: Registration of point cloud data from a geometric optimization perspective. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (2004), ACM, New York, pp. 22–31.

[MH94] Marsden J. E., Hughes T. J.: *Mathematical Foundations of Elasticity*. Dover Publications, New York, 2012.

[MKS01] Ma Y., Košecká J., Sastry S.: Optimization criteria and geometric algorithms for motion and structure estimation. *International Journal of Computer Vision 44*, 3 (2001), 219–249.

[MLSS94] Murray R. M., Li Z., Sastry S. S., Sastry S. S.: *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Boca Raton, FL 1994.

[MPD06] Makadia A., Patterson A., Daniilidis K.: Fully automatic registration of 3D point clouds. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2006* (2006), IEEE, IEEE Computer Society, NJ, vol. *1*, pp. 1297–1304.

[MQZ*15] Ma J., Qiu W., Zhao J., Ma Y., Yuille A., Tu Z.: Robust L2E estimation of transformation for non-rigid registration. *IEEE Transactions on Signal Processing 63*, 5 (2015), 1115–1129.

[MS10] Myronenko A., Song X.: Point set registration: Coherent point drift. *IEEE Transactions on Pattern Analysis and Machine Intelligence 32*, 12 (2010), 2262–2275.

[Myr] Myronenko A.: Coherent point drift (cpd2.zip). https://sites.google.com/site/myronenko/research/cpd. Accessed on August 25, 2018.

[MZJZ17] Ma J., Zhao J., Jiang J., Zhou H.: Non-rigid point set registration with robust transformation estimation under manifold regularization. In *AAAI* (2017), pp. 4218–4224.

[MZY16] Ma J., Zhao J., Yuille A.: Non-rigid point set registration by preserving global and local structures. *IEEE Transactions on Image Processing 25*, 1 (2016), 53–64.

[PLH04] Pottmann H., Leopoldseder S., Hofer M.: Registration without ICP. *Computer Vision and Image Understanding 95*, 1 (2004), 54–71.

[RCB97] Rangarajan A., Chui H., Bookstein F.: The softassign procrustes matching algorithm. In *Biennial International Conference on Information Processing in Medical Imaging* (Berlin, Heidelberg, 1997), Springer, pp. 29–42.

[Ren] Renderer X.: Implicit surfaces. http://xrt.wikidot.com/gallery:implicit. Accessed on June 13, 2018.

[RL01] Rusinkiewicz S., Levoy M.: Efficient variants of the ICP algorithm. In *Proceedings of the 3rd International Conference on 3-D Digital Imaging and Modeling* (IEEE Computer Society, NJ, 2001), IEEE, pp. 145–152.

[SA91] Sabata B., Aggarwal J.: Estimation of motion from a pair of range images: A review. *CVGIP: Image Understanding 54*, 3 (1991), 309–324.

[SAD*16] Shi Z., Alliez P., Desbrun M., Bao H., Huang J.: Symmetry and orbit detection via lie-algebra voting. In *Computer Graphics Forum* (2016), Wiley Online Library, Goslar Germany, Germany, vol. *35*, pp. 217–227.

[SF89] Simo J., Fox D.: On a stress resultant geometrically exact shell model. Part I: Formulation and optimal parametrization. *Computer Methods in Applied Mechanics and Engineering 72*, 3 (1989), 267–304.

[SHT09] Segal A., Haehnel D., Thrun S.: Generalized-ICP. In *Robotics: Science and Systems* (2009), vol. *2*, p. 435.

[SMFF07] Salvi J., Matabosch C., Fofi D., Forest J.: A review of recent range image registration methods with accuracy evaluation. *Image and Vision Computing 25*, 5 (2007), 578–596.

[Stu64] Stuelpnagel J.: On the parametrization of the three-dimensional rotation group. *SIAM Review 6*, 4 (1964), 422–430.

[SVQ88] Simo J., Vu-Quoc L.: On the dynamics in space of rods undergoing large motions— A geometrically exact approach. *Computer Methods in Applied Mechanics and Engineering 66*, 2 (1988), 125–161.

[SW91] Simo J., Wong K.: Unconditionally stable algorithms for rigid body dynamics that exactly preserve energy and momentum. *International Journal for Numerical Methods in Engineering 31*, 1 (1991), 19–52.

[SYS07] Sofka M., Yang G., Stewart C.: Simultaneous covariance driven correspondence (CDC) and transformation estimation in the expectation maximization framework. In *IEEE Conference on Computer Vision and Pattern Recognition, 2007* (IEEE Computer Society, NJ, 2007), IEEE, pp. 1–8.

[TCL*13] Tam G., Cheng Z., Lai Y., Langbein F., Liu Y., Marshall D., Martin R., Sun X., Rosin P.: Registration of 3-D point clouds and meshes: A survey from rigid to nonrigid. *IEEE Transactions on Visualization and Computer Graphics 19*, 7 (2013), 1199–1217.

[Tea] Team G.-P.: Implicit algebraic surfaces. http://www-sop.inria.fr/galaad/surface/. Accessed on June 13, 2018.

[TFR99] Trucco E., Fusiello A., Roberto V.: Robust motion and correspondence of noisy 3-D point sets with missing data. *Pattern Recognition Letters 20*, 9 (1999), 889–898.

[TK04] Tsin Y., Kanade T.: A correlation-based approach to robust point set registration. In *European Conference on Computer Vision* (2004), Springer, Berlin, Heidelberg, pp. 558–569.

[TL] Turk G., Levoy M.: The Stanford 3-D Scanning Repository. Stanford Computer Graphics Laboratory. http://graphics.stanford.edu/data/3Dscanrep. Accessed on June 30, 2017.

[Ume91] UMEYAMA S.: Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence 4* (1991), 376–380.

[WB01] WILLIAMS J., BENNAMOUN M.: Simultaneous registration of multiple corresponding point sets. *Computer Vision and Image Understanding 81*, 1 (2001), 117–142.

[YLJ13] YANG J., LI H., JIA Y.: Go-ICP: Solving 3D registration efficiently and globally optimally. In *IEEE International Conference on Computer Vision (ICCV), 2013* (IEEE Computer Society, NJ, 2013), IEEE, pp. 1457–1464.

[ZD06] ZHENG Y., DOERMANN D.: Robust point matching for non-rigid shapes by preserving local neighborhood structures. *IEEE Transactions on Pattern Analysis & Machine Intelligence 28*, 4 (2006), 643–649.

[ŽK98] ŽEFRAN M., KUMAR V.: Interpolation schemes for rigid body motions. *Computer-Aided Design 30*, 3 (1998), 179–189.

**Supporting Information**

Additional supporting information may be found online in the Supporting Information section at the end of the article.

**Code snippet 1**: Exponential map of a skew matrix

**Code snippet 2**: Level set function $\Psi$ and its derivatives

**Code snippet 3**: Assemble matrix-vector system

**Code snippet 4**: Pose update

**Code snippet 5**: Implementation of Algorithm 1