Rapid Analytic Optimization of Quadratic ICP Algorithms

Leonid German[1(✉)], Jens R. Ziehn[1,2], and Bodo Rosenhahn[1]

[1] Institut für Informationsverarbeitung (TNT),
Leibniz Universität Hannover, Hannover, Germany
german@tnt.uni-hannover.de
[2] Fraunhofer IOSB, 76131 Karlsruhe, Germany

Abstract. This paper discusses the efficient optimization of iterative closest points (ICP) algorithms. While many algorithms formulate the optimization problem in terms of quadratic error functionals, the discontinuities introduced by varying changing correspondences usually motivate the optimization by quasi-Newton or Gauss-Newton methods. These disregard the fact that the Hessian matrix in these cases is constant, and can thus be precomputed analytically and inverted a-priori. We demonstrate on the example of Allen et al.'s seminal paper "The space of human body shapes", that all relevant quantities for a full Newton method can be derived easily, and lead to an optimization process that reduces computation time by around 98% while achieving results of almost equal quality (about 1% difference). Along the way, the paper proposes minor improvements to the original problem formulation by Allen et al., aimed at making the results more reproducible.

1 Introduction

ICP (iterative closest point) algorithms are simple, locally optimal algorithms that, given a distance measure, compute correspondences between two sets of points (registration). By establishing this correspondence, it is possible to estimate a rigid or non-rigid transformation, that will align the points. This approach has been successfully applied to morph a known shape towards an unknown shape, for example moving a labeled mesh of a generic face onto a stereo depth surface showing the face of a person, to detect face parts and expressions. Another common application is morphing a labeled mesh of a human body onto a laser scan to detect body parts, assign animations, or estimate pose or body shape. ICP is also used to register 2D (retinal images [1]) as well as 3D (CT, ultrasound [2]) medical image data. Registrations can produce a large image by stitching separate snapshots, reveal connections between images of different modalities and help in diagnosis by exposing change in images taken over time.

[3] gives an overview of various registration methods and categorizes them by different properties, among others by techniques chosen for optimization. Among these, many relevant ICP approaches (such as [4–8]) express the matching error between two shapes (which is to be minimized) exclusively in terms of quadratic

© Springer International Publishing AG 2017
C.-S. Chen et al. (Eds.): ACCV 2016 Workshops, Part III, LNCS 10118, pp. 61–75, 2017.
DOI: 10.1007/978-3-319-54526-4_5

differences, which means that the optimization problem has a constant Hessian matrix. This could be exploited by optimizing via Newton's method, using gradients and the inverted Hessian directly. Instead however, the common optimization approach is to *approximate* the relevant quantities (such as the Hessian) numerically at each iteration, for example through the L-BFGS method. This introduces an extreme computational overhead; whether it is justified has (to the knowledge of the authors) not been studied conclusively. [7] remarks "Because the cost function is evaluated based on the control points in the test scan and their closest counterparts in the deformable model, and the closest counterparts may change due to the adjustment [...], the optimization problem is highly nonlinear", which is true: The problem is a discontinuous piecewise combination of parabolae of identical shape but different offset. It is, however, not evaluated, whether approximation algorithms such as BFGS necessarily perform better in this case. [6] computes correspondences in faces by minimizing an objective function built from squares of distances and squares of scalar products of vectors. A minimum is found via L-BFGS by performing at most 1000 iterations. [9] prefers Newton's method over BFGS for its more analytic and transparent formulation, and notes improvements in solution quality for face registration; however there is no acknowledgment of the fact that the Hessian is constant and need not be recomputed in the process, and no record of any significant improvement in computation speed. A simplification step is considered to produce a minimizable quadratic function. We will show, however, that the original function is already locally quadratic almost everywhere with a constant Hessian, so that the proposed repeated solution of a linear system [9] is unnecessary and can be reduced to a matrix multiplication. This leads to a speed-up that can be motivated by the problem understanding alone.

This paper will demonstrate on the example of [5], that the analytic derivation of gradients and of the constant Hessian can dramatically reduce computational effort in applicable ICP algorithms, while at the same time achieving results of very similar quality, despite the fact that the optimization problem is highly discontinuous. To do so, Sect. 2 will introduce the original problem formulation in [5]; Sect. 3 will discuss the resulting problem structure from an optimizational perspective, and derive the relevant analytic quantities to perform Newton's method. Section 4 compares the proposed analytic solution to several variants of the original BFGS solution, to determine their respective performance in terms of result quality and computation time.

2 Problem Statement

This section introduces the notation used to describe the problem, as well as the formulae that define the optimization goals. As previously stated, the goal is to transform vertices of a mesh M (here considered to be a generic low-poly model of a human body) by a vector of unknown homogeneous transformation matrices $\mathbf{x}$ to fit it to a surface S (here a *finite* high-resolution 3D point cloud from a laser scanner, without a specific topology but including normals).

Definition 1 (Surface, s, S). The *surface* is considered to be a set[1] of points S, whose elements $s \in S$ are assigned the following quantities:

- $\boldsymbol{\psi}_s$, an $\mathbb{R}^4$ vector of homogeneous coordinates that describe where s lies on the surface.
- $\boldsymbol{\nu}_s$, the $\mathbb{R}^3$ normal vector of the surface at s.

Definition 2 (Mesh, m, M). The *mesh* is considered to be a set of points M, whose elements $m \in M$ are assigned the following quantities:

- $\mathbf{p}^m$, an $\mathbb{R}^4$ vector of homogeneous coordinates of m in the input mesh.
- $\mathbf{X}^m$, an $\mathbb{R}^{4\times4}$ homogeneous transformation matrix; $\mathbf{X}^m\mathbf{p}^m$ is the position of m in the deformed mesh. These matrices are the optimization variables.
- $\mathcal{N}^m \subset M$, the neighborhood of m in the mesh's topology. It holds that $n \in \mathcal{N}^m \Leftrightarrow m \in \mathcal{N}^n$, but $m \notin \mathcal{N}^m$.
- $\mathbf{n}^m$, the $\mathbb{R}^3$ mesh normal at m *after* the transformation. It must be recomputed several times during an optimization from the current $\mathbf{X}^n\mathbf{p}^n, n \in \mathcal{N}^m$.

Definition 3 (Vector of all unknowns, $\mathbf{x}$, $\mathbf{x}^m$). The concatenation of all $\mathbf{X}^m$ into a vector $\mathbf{x} \in \mathbb{R}^{12|M|}$ is referred to as *vector of all unknowns* (the actual ordering of the indices is of no concern here). $\mathbf{x}^m$ denotes the part of $\mathbf{x}$ containing exactly the elements of $\mathbf{X}^m$.

2.1 Optimization Goal

A given vector $\mathbf{x} \in \mathbb{R}^{12|M|}$ is evaluated by an *energy function* of the form

$$E_{\text{total}}(\mathbf{x}) = \delta \cdot E_{\text{data}}(\mathbf{x}) + \mu \cdot E_{\text{marker}}(\mathbf{x}) + \sigma \cdot E_{\text{smooth}}(\mathbf{x}), \tag{1}$$

using weights $\delta, \mu, \sigma \in [0, \infty)$, with the goal of finding $\mathbf{x}^* \in \arg\min_{\mathbf{x}} E_{\text{total}}(\mathbf{x})$. The corresponding energy terms will be defined in this section, along with a brief discussion of their relevant properties.

Data Error. The similarity between the transformed mesh M and the surface S is determined via

$$E_{\text{data}}(\mathbf{x}) = \textstyle\sum_{m\in M} \left\|\mathbf{X}^m\mathbf{p}^m - \boldsymbol{\psi}_{s(m)}\right\|_2^2, \tag{2}$$

where $s(m)$ picks the *closest compatible surface point* to m out of S.

Definition 4 (Closest Compatible Point, $s(m)$). A point $s(m)) \in S$ is called the *closest compatible point of* m, if it is the closest to m among all surface points $S(m)$ whose normal $\boldsymbol{\nu}_s$ is within an angle of $\pi/2$ of the transformed mesh normal $\mathbf{n}^m$. Formally

$$S(m) = \{s \in S | \boldsymbol{\nu}_s^\mathsf{T}\mathbf{n}^m \geqslant 0\} \quad \text{and} \quad s(m) = \underset{s\in S(m)}{\arg\min} \left\|\mathbf{X}^m\mathbf{p}^m - \boldsymbol{\psi}_s\right\|_2. \tag{3}$$

E_{data} thus accumulates the squared distances between all mesh points and their closest partners on the surface.

[1] While [5] allows for an infinite $|S|$, all known practical applications (including the ones in [5]) use a finite $|S|$. For this reason, we focus on the finite case, which brings about peculiar effects (discussed in Sect. 3) absent in an infinitely fine S.

Marker Error. To stabilize the optimization by use of limited manual preprocessing, a (usually small) number of *markers* can be specified, which are assigned fixed corresponding surface points (in lieu of the closest compatible point function $s(m)$), for example for the head, hands and feet.

Definition 5 (Markers, M', $\check{s}(m)$). A subset $M' \subset M$ is called *markers*, for which there exists an a-priori function $\check{s} : M' \to S$ which assigns marker points $m' \in M'$ definite correspondence $\check{s}(m')$ in the surface.

The distance of marker points to their corresponding (according to $\check{s}$) surface points is evaluated by the marker error:

$$E_{\text{marker}}(\mathbf{x}) = \sum_{m' \in M'} \left\| \mathbf{X}^{m'} \mathbf{p}^{m'} - \boldsymbol{\psi}_{\check{s}(m')} \right\|_2^2. \tag{4}$$

In the formulation in [5], markers are thus evaluated twice, once in (4) and once in (2), which does not exclude M', but possibly with respect to different points. This formulation is optional; (2) may include or exclude M'.

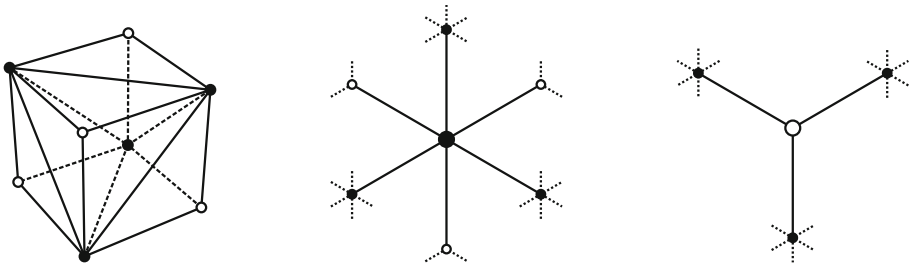
Smoothness Error. The use of homogeneous matrices for transformation, instead of mere translations, means that transformations such as scaling or rotation can be described as a uniform transformation of several points (such as an arm), as opposed to a diverse variety of individual translations. By asking that the matrices be as uniform as possible over the entire mesh (thus for the first time using the mesh's topology $\mathcal{N}$), it is possible to express that the original mesh shape should be locally preserved as well as possible. This is achieved by stating that the matrices $\mathbf{X}^m$ and $\mathbf{X}^n$ of neighboring points should be as similar as possible. The term given in [5] as a *smoothness error*[2] is

$$E_{\text{smooth}}(\mathbf{x}) = \sum_{m \in M} \sum_{n \in \mathcal{N}^m} \left\| \mathbf{X}^m - \mathbf{X}^n \right\|_{\text{fro}}^2, \tag{5}$$

using the Frobenius norm $\left\|\mathbf{M}\right\|_{\text{fro}}^2 = \sum_{i,j} M_{ij}^2$ as a measure of similarity. Although not the main focus of this paper, (5) as proposed in [5] leaves two potential pitfalls, which may be worth resolving:

– **Unit Invariance.** First, it is risky to sum over all (squared) matrix elements regardless of their unit: While all $X_{i,j}$ with $i, j \in \{1, 3\}$ are ratios and thus dimensionless, the translation components $X_{i,4}$ with $i \in \{1, 3\}$ represent units of length. Therefore, two identical models in different units (e.g. inches and centimeters) optimized with the same parameters could have different optima. For this reason we propose to include another ratio weight ρ in addition to

[2] It should be noted that "smoothness" here refers to the matrix elements, not the mesh: Each matrix element should change smoothly between neighboring points. The resulting mesh need not be smooth in shape at all, but neighboring points should be transformed similarly.



(a) complete mesh (b) black nodes' $\mathcal{N}$ (c) white nodes' $\mathcal{N}$

Fig. 1. Unevenly distributed neighbor counts on an isotropic 3D model: (a) shows a triangulated cube, which has two types of vertices (shown black and white) with different numbers of neighbors: The black vertices (b) have 6 neighbors each, the white vertices (c) only three. Depending on the smoothness error formulation, black and white vertices behave very differently, even though in the untriangulated cube, all corners are equal.

σ, that turns units into dimensionless values, e.g. $\rho = 2.3/\text{mm}$. Equation (5) thus becomes

$$E_{\text{smooth}} = \sum_{m\in M} \sum_{n\in\mathcal{N}^m} \Big\| \underbrace{\begin{bmatrix} 1 & 1 & 1 & \sqrt{\rho} \\ 1 & 1 & 1 & \sqrt{\rho} \\ 1 & 1 & 1 & \sqrt{\rho} \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix}}_{\mathbf{P}} \circ (\mathbf{X}^m - \mathbf{X}^n) \Big\|^2_{\text{fro}}, \tag{6}$$

where $(\mathbf{M} \circ \mathbf{N})_{ij} = m_{ij} \cdot n_{ij}$ defines the element-wise (or Hadamard) product. The last row of the weighting matrix is irrelevant because it is identical for all $\mathbf{X}^m$. The square roots represent the fact that all other weights are applied to squared values.

– **Triangulation Invariance.** In the original definition a node with many neighbors will generally be affected more strongly by smoothness than a node with few neighbors, because the errors are summed over the entire neighborhood. This means that the result of the optimization may depend heavily on the topology, but the topology of triangulation does not necessarily reflect the topology of the underlying shape. Figure 1 gives an example: A regular cube is triangulated, but the number of neighbors is very unevenly distributed and varies by a factor of 2. In general this is not avoidable, and thus not a mark of a bad triangulation. For the cube it means that even though a cube in theory has no preferred vertices, its triangulated representation does, and so if this cube is fit to a given surface, its points will have to satisfy very different smoothness requirements and the transformation will likely not be uniform. This can be countered in various ways; the one proposed here is normalizing the smoothness term over the number of neighbors, to obtain (along with the previously introduced weighting of (6))

$$E_{\text{smooth}} = \sum_{m\in M} \frac{1}{|\mathcal{N}^m|} \sum_{n\in\mathcal{N}^m} \big\| \mathbf{P} \circ (\mathbf{X}^m - \mathbf{X}^n) \big\|^2_{\text{fro}}. \tag{7}$$

In this paper, Eq. (7) was used as a smoothness error, and all following derivations rely on this instead of the original error (5) as given in [5]. The changes necessary to adapt the following considerations to the original formulation are, however, straightforward.

3 Optimization

To (approximately) minimize (1) for $\mathbf{x}^*$, [5] proposes using the limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) method, an approximation of Newton's method. We will provide a very brief overview of the key ideas behind both methods in this section, to motivate why for the particular problem structure proposed in [5] (as described in Sect. 2) Newton's method is both simpler and better suited. The reader is referred to [10] for detailed yet accessible accounts of all methods mentioned here.

3.1 Newton's Method and L-BFGS

In optimization, Newton's method takes an objective function $E(x)$ (often assumed to be at least twice continuously differentiable) and an "initial guess" x_i, and improves $x_i \mapsto x_1$ to minimize $E(x_1)$ by only considering local derivatives of E up to second order. The process is usually iterated over $x_i \mapsto x_{i+1}$. It approximates $E(x)$ based on its second-order Taylor expansion

$$E(x_i + s) \approx \tilde{E}_{x_i}(s) = E(x_i) + s \cdot E'(x_i) + s^2 \cdot 1/2 \cdot E''(x_i), \tag{8}$$

where $E' = \mathrm{d}E/\mathrm{d}x$ and $E'' = \mathrm{d}^2E/(\mathrm{d}x)^2$. The optimum of the parabola $\tilde{E}_{x_i}(s)$ lies at $\hat{s} = -E'(x_i)/E''(x_i)$. If $E'' > 0$, this optimum is a minimum, and the update $x_1 = x_i - \hat{s}$ can be assumed to approximate the (or a) minimum of E. When applied iteratively, the process is repeated until, for example, $E(x_i)$, $|E(x_i) - E(x_{i-1})|$ or $E'(x_i)$ are sufficiently small, or i becomes too large. In n dimensions, Newton's method chooses steps $\hat{s}$ according to

$$\mathbf{H}\,\hat{\mathbf{s}} = -\mathbf{g}, \tag{9}$$

where $\mathbf{H}$ is the local *Hessian* $H_{ij} = \mathrm{d}^2/\mathrm{d}x_i\mathrm{d}x_j E$ and $\mathbf{g} = \nabla E$ is the local *gradient* $g_i = \mathrm{d}E/\mathrm{d}x_i$.

Line Search, Wolfe Conditions and BFGS. Optimization using a *line search* method departs from Newton's method in that it does not compute the step length along with the step direction (as $\hat{\mathbf{s}}$), but in a second step and by potentially separate criteria. It is common to use the direction of $\hat{\mathbf{s}}$ as determined in (9), and then search along the *line* $\Lambda(\alpha) = E(\mathbf{x} + \alpha\hat{\mathbf{s}})$ for a step scale α.

In practice, it is often considered inefficient to attempt to locally or globally minimize even the one-dimensional Λ for α (in particular due to E usually being not well-understood); instead, *Wolfe conditions* are used which provide criteria

for so-called *sufficient decrease*. A step that satisfies these conditions is proven to exist, and within these bounds assures progress in minimization. To find the optimal scale α, $\Lambda(\alpha)$ and thereby E usually has to be evaluated several times.

The Broyden–Fletcher–Goldfarb–Shanno (BFGS) method can be applied when the Hessian is not available or too difficult to compute. It approximates the Hessian in this process by starting as a line search along the gradient from a given point (assuming the Hessian e.g. as an identity matrix) and estimating a new Hessian from the *change in gradients* encountered after each iteration. Since the underlying Newton model of quadratic approximation can be used to predict the gradient at the next step, any deviation in the actually computed gradient hints at a defect in the approximated Hessian (either because the approximation was insufficient, or because the Hessian has changed between steps). In this case, the true gradient provides a one-dimensional information to perform a rank-one update on the approximated Hessian. This process is well defined if BFGS is used in combination with *line search* and *Wolfe conditions*.

L-BFGS. In cases where the Hessian is further very large[3], it may be considered inefficient to store and update the full approximation. Therefore, limited-memory BFGS (L-BFGS) stores just a fixed number of previous rank-one updates and expresses the approximated Hessian exclusively in terms of these.

We conclude that the aptness of the three approaches depends on properties of the optimization problem, namely whether the Hessian can and should be computed analytically, and whether function evaluation is so computationally inexpensive that line search and Wolfe conditions should be applied.

Furthermore, the actual performance of an optimization method cannot be predicted exclusively from these metrics; therefore the considerations in the following Sect. 3.2 merely provide a theoretical basis to be validated in Sect. 4.

3.2 Optimizational Properties of the Error Functional

This section will illustrate the properties of E_{total} in terms of optimization, and motivate the choice of Newton's method over L-BFGS. Analytical representations of the gradient and the Hessian will be given. Due to the sum rule, it holds for the gradient that $\nabla E_{\text{total}} = \nabla E_{\text{data}} + \nabla E_{\text{marker}} + \nabla E_{\text{smooth}}$ and respectively for the Hessian that

$$\underbrace{\nabla\nabla^{\mathsf{T}} E_{\text{total}}}_{\mathbf{H}} = \underbrace{\nabla\nabla^{\mathsf{T}} E_{\text{data}}}_{\mathbf{H}_{\text{data}}} + \underbrace{\nabla\nabla^{\mathsf{T}} E_{\text{marker}}}_{\mathbf{H}_{\text{marker}}} + \underbrace{\nabla\nabla^{\mathsf{T}} E_{\text{smooth}}}_{\mathbf{H}_{\text{smooth}}}, \tag{10}$$

so that the gradients and Hessians can be considered for each error term E_{data}, E_{marker} and E_{smooth} independently. It will be shown that the Hessian matrix $\mathbf{H} = \nabla\nabla^{\mathsf{T}} E_{\text{total}}$ is constant over the optimization process for a given application, and depends exclusively on the topology of the input mesh. The reason for this lies in the fact that the energy function E_{total} is piecewise quadratic in the

[3] In the present example, there are $\mathcal{O}(|M|^2)$ entries in $\mathbf{H}$, which can be considerable.

variables x_{ij}, and any quadratic function has a constant $\mathbf{H}$. There are, however, discontinuities in $\mathbf{g}$ that propagate into $\mathbf{H}$, when the association between a mesh point and its closest surface point changes. In these places $\mathbf{H}$ is not defined.[4]

Data Error. The term E_{data} of (2) is not everywhere continuously differentiable due to the fact that $\boldsymbol{\psi}_{s(m)}$ is not continuous. The associated closest points may jump at infinitesimally small changes in $\mathbf{X}^m$. It is, however, continuously differentiable *almost* everywhere, since the space of parameters where a mesh point switches the closest surface point has measure zero. Furthermore the term is everywhere *continuous* as even if $\boldsymbol{\psi}_{s(m)}$ jumps from $\boldsymbol{\psi}_a$ to $\boldsymbol{\psi}_b$ when $\mathbf{X}^m$ is changed infinitesimally, the jump occurs just when both $\boldsymbol{\psi}_a$ and $\boldsymbol{\psi}_b$ have the same distance to $\mathbf{X}^m\mathbf{p}^m$, so the limits from both sides match.

Also, using a homogeneous matrix with 12 real parameters instead of an $\mathbb{R}^3$ translation vector to transform individual points means that E_{data} has a huge overhead of redundant parameters. Therefore there is no single minimum but a space of minima with $|M|\cdot 9$ parameters. In general there are $|S|^{|M|}$ disconnected minima, since any point in M can be located directly at any surface point.

Data Gradient. From Eq. (2) it follows that the derivative of E_{data} by the m-th unknown transformation matrix $\mathbf{X}^m$ is:

$$\frac{\mathrm{d}E_{\text{data}}}{\mathrm{d}\mathbf{X}^m} = \frac{\mathrm{d}}{\mathrm{d}\mathbf{X}^m} \sum_{\bar{m}\in M} \left\| \mathbf{X}^{\bar{m}}\mathbf{p}^{\bar{m}} - \boldsymbol{\psi}_{s(\bar{m})} \right\|_2^2 \tag{11}$$

$$= \frac{\mathrm{d}}{\mathrm{d}\mathbf{X}^m} \left\| \mathbf{X}^m\mathbf{p}^m - \boldsymbol{\psi}_{s(m)} \right\|_2^2 = 2\left(\mathbf{X}^m\mathbf{p}^m - \boldsymbol{\psi}_{s(m)}\right)\mathbf{p}^{m\mathsf{T}}, \tag{12}$$

where step (11)–(12) makes use of the fact that each matrix is independent of all others in the sum.[5] As previously discussed, the gradient is not defined where $s(m)$ is not uniquely defined, i.e. when two different surface points are equally close to $\mathbf{X}^m\mathbf{p}^m$. These discontinuities lead to a non-convex problem.

Data Hessian. The derivation of Eq. (12) by another matrix $\mathbf{X}^o$ (because "n" is still reserved for neighbors in the smoothness error) gives

$$\frac{\mathrm{d}^2E_{\text{data}}}{\mathrm{d}\mathbf{X}^o\mathrm{d}\mathbf{X}^m} = \frac{\mathrm{d}}{\mathrm{d}\mathbf{X}^o} 2\left(\mathbf{X}^m\mathbf{p}^m - \boldsymbol{\psi}_{s(m)}\right)\mathbf{p}^{m\mathsf{T}} = \begin{cases} 2\,\mathbf{p}^m\mathbf{p}^{m\mathsf{T}} & \text{for } m = o \\ 0 & \text{else,} \end{cases} \tag{13}$$

which is, as can be seen, independent of all variable properties, and exclusively depends on the untransformed coordinates of the initial mesh. Nevertheless the Hessian is not formally defined at those $\mathbf{x}$ where the closest point associations

[4] In the theoretical limit of a continuous surface, this happens almost permanently. In this case, $\mathbf{g}$ is almost always continuous and $\mathbf{H}$ cannot be given as shown here (i.e. in terms of a polygon topology). In general, $\mathbf{H}$ is not constant then. As initially stated, this case is not considered.

[5] The aforementioned overhead in parameters (Sect. 2.1) is mirrored here in the fact that the gradient is of the form $\mathbf{u}\mathbf{v}^\mathsf{T}$, which means it has rank 1 (dimension 3), corresponding to the expected 3° of freedom out of 12 parameters.

$s(m)$ change, because the gradient is not defined there—in spite of the fact that these places constitute a *removable* singularity on behalf of the Hessian, due to the fact that the Hessian is the same everywhere.

Marker Error. The term E_{marker} of (4) is continuous, convex (but not *strictly* convex), quadratic and has a global minimum value of zero, attained when all $\mathbf{X}^m$ (where $m \in M'$) place their corresponding $\mathbf{p}^m$ exactly at their markers $\boldsymbol{\psi}_{\check{s}(m)}$. Again there is no single minimum, once due to the homogeneous coordinates as for the data error, but additionally because once all points in M' are set, all other points $M \setminus M'$ can be assigned arbitrary matrices. This leads to a single connected space of minima with $|M \setminus M'| \cdot 12 + |M'| \cdot 9$ parameters.

Marker Gradient and Hessian. The marker gradient and Hessian are identical to the data gradient and Hessian, only that the corresponding surface points are fixed. This also includes that the marker gradient and Hessian are defined everywhere without exception.

Smoothness Error. The term E_{smooth} of (5) or (7) is continuous and has a global minimum value of zero, attained when all $\mathbf{X}^m$ are identical. It is no single minimum either: $\mathbf{X}^1 = \cdots = \mathbf{X}^{|M|}$ can be arbitrary. Thus the space of minima has 12 free parameters. This is, by the way, independent of which error metric (of the three alternatives given in Sect. 2) is used.

Smoothness Gradient. The derivation of Eq. (7) gives

$$\frac{\mathrm{d}E_{\text{smooth}}}{\mathrm{d}\mathbf{X}^m} = \frac{\mathrm{d}}{\mathrm{d}\mathbf{X}^m} \sum_{\bar{m} \in M} \frac{1}{|\mathcal{N}^{\bar{m}}|} \sum_{\bar{n} \in \mathcal{N}^{\bar{m}}} \|\mathbf{P} \circ (\mathbf{X}^{\bar{m}} - \mathbf{X}^{\bar{n}})\|_{\text{fro}}^2 \tag{14}$$

$$= \mathbf{P}^2 \circ \left(\frac{2}{|\mathcal{N}^m|} \sum_{\bar{n} \in \mathcal{N}^m} (\mathbf{X}^m - \mathbf{X}^{\bar{n}}) + \sum_{\bar{n} \in \mathcal{N}^m} \frac{2}{|\mathcal{N}^{\bar{n}}|} (\mathbf{X}^{\bar{n}} - \mathbf{X}^m) \right), \tag{15}$$

where the first addend of Eq. (15) represents the case where $\mathbf{X}^m$ is "here" (in the role of m in Eq. (7)), the second addend is where $\mathbf{X}^m$ is one of the neighbors (in the role of n in Eq. (7)), and $\mathbf{P}^2 = \mathbf{P} \circ \mathbf{P}$, not $\mathbf{PP}$.

Smoothness Hessian. The smoothness Hessian is the derivation of Eq. (15) by another matrix $\mathbf{X}^o$

$$\frac{\mathrm{d}^2 E_{\text{smooth}}}{\mathrm{d}\mathbf{X}^o \mathrm{d}\mathbf{X}^m} = \frac{\mathrm{d}}{\mathrm{d}\mathbf{X}^o} \mathbf{P}^2 \circ \left(\frac{2}{|\mathcal{N}^m|} \sum_{\bar{n} \in \mathcal{N}^m} (\mathbf{X}^m - \mathbf{X}^{\bar{n}}) + \sum_{\bar{n} \in \mathcal{N}^m} \frac{2}{|\mathcal{N}^{\bar{n}}|} (\mathbf{X}^{\bar{n}} - \mathbf{X}^m) \right), \tag{16}$$

which can be simplified as all X_{ij}^m in one $\mathbf{X}^m$ share the same smoothness Hessian entries. These entries can be denoted by a scale factor η^{om}, which depends only on the mesh points o and m (namely their topology), but not indices i and j. It can be defined via the formula

$$P_{ij}^2 \cdot \eta^{om} = \frac{\mathrm{d}^2}{\mathrm{d}X_{ij}^o \mathrm{d}X_{ij}^m} E_{\text{smooth}}, \tag{17}$$

where P_{ij}^2 is either 1 or ρ (cf. Eq. (7)). If, as above, this is factored out, η^{om} is fully independent of ij.

The above equation only differentiates twice by the index ij. It must also be considered that the gradient ij could be differentiated by different indices kl. However in this case it holds that

$$\frac{\mathrm{d}^2}{\mathrm{d}X_{ij}^o \mathrm{d}X_{kl}^m} E_{\text{smooth}} = 0 \quad \text{for ij} \neq kl, \tag{18}$$

because in E_{smooth} each matrix entry ij only depends on the entries ij of its neighbors, not on any other indices kl. Also, if o and m are not identical or neighbors, η^{om} will always be 0, because distinct, non-neighboring matrices do not depend on each other in terms of smoothness.

Using these considerations, Eq. (16) can be solved for η^{om} to obtain

$$\eta^{om} = \begin{cases} 2 - \sum_{n \in \mathcal{N}^m} \frac{2}{|\mathcal{N}^n|} & \text{if } o = m \\ \frac{2}{|\mathcal{N}^o|} - \frac{2}{|\mathcal{N}^m|} & \text{if } o \in \mathcal{N}^m \\ 0 & \text{else,} \end{cases} \tag{19}$$

which in turn can be used to set up the smoothness Hessian (using $\mathbf{D}$, which is a $\mathbb{R}^{12\times12}$ diagonal matrix whose diagonal elements consist of the $P_{ij} \in \{1, \rho\}$ in the order fitting how $\mathbf{x}^m$ is obtained from $\mathbf{X}^m$):

$$\mathbf{H}_{\text{smooth}} = \begin{bmatrix} \eta^{11}\mathbf{D} & \cdots & \eta^{1n}\mathbf{D} \\ \vdots & \ddots & \vdots \\ \eta^{n1}\mathbf{D} & \cdots & \eta^{nn}\mathbf{D} \end{bmatrix}. \tag{20}$$

Again it can be found that, as with the data Hessian, the smoothness Hessian does not depend on any variables, just on the topology of the input mesh. Taken together this proves the expected result that the complete $\mathbf{H}$ be independent of $\mathbf{x}$, and thus constant throughout the process, because E_{total} is piecewise quadratic.

Combined Error Term. The combined error term E_{total} is the sum of the previously described individual errors (as given in Eq. (1)). Unless some weights are set to zero, it is influenced by all previously described properties. The first important property is that while none of the previous terms defined exact local minima (each had a whole space of minima due to an overhead of parameters) the total error in general requires the full set of parameters, and thus local minima are points, not spaces. The smoothness error makes use of the full homogeneous matrices, and the data and marker errors assure that smoothness is related to an optimal data fit, and thus prefers some transformation matrices over others. Furthermore due to the fact that E_{data} (2) is not continuously differentiable (due to reassignment of closest surface points), while both other errors are, the sum of them must have a discontinuous derivative as well.

3.3 Summary of Optimization Considerations

The previous sections have shown not only that for a given problem the analytic prerequisites for Newton's method, the gradient and the Hessian, can be

computed analytically—but that they can be computed easily and efficiently, particularly due to the constant Hessian which need not be recomputed during iterations. The repeated numerical approximation of these values via BFGS is not necessary: The simple problem formulation makes it ideally suited for Newton's method using analytic gradients and Hessians.

Furthermore, it was seen that an evaluation of E requires a considerable number of processing steps for large M and S, due to the effect of normals and point associations. Therefore, line search must be regarded sceptically, since the assumption that function evaluations are less costly than the computation of an exact Newton step is not necessarily satisfied.

However, as initially stated, L-BFGS could still outperform the results of Newton's method, depending on the actual shape of M, S and thus E. To shed light on this, the following Sect. 4 will compare both approaches based on their practical performance on realistic data.

4 Practical Application

To evaluate the practical performance of the proposed analytic optimization, we compare it to the original algorithm in [5] on realistic data. We match three human body meshes of equal topology but different pose, $|M_1| = |M_2| = |M_3| = 1002$ to three laser-scanned human body surfaces, $|S_1| = 347\,644$, $|S_2| = 361\,026$, $|S_3| = 367\,093$. This provides 9 combinations, for each of which we compare Newton's method and different variants of L-BFGS.

[5] proposes a two-pass approach: First minimizing the error using only markers and smoothness, with weights $\delta = 0, \sigma = 1, \mu = 10$, then including the data term with $\delta = 1, \sigma = 1, \mu = 10$.[6] This is meant to place the mesh in a roughly correct position before transforming separate mesh points. We test this method with BFGS, and additionally the "one-pass" approach of using data error right from the start. For BFGS, testing both approaches is valid because (as seen in Fig. 3), the one-pass approach takes only about 2/3 of the computation time, but provides results with about 200% the error. Therefore, quality and speed could be traded off. For Newton's method, omitting the marker-only pass is unnecessary: As for $\delta = 0$ point associations do not change, Newton's method converges after only one iteration, so that the additional effort is negligible.

Additionally, in Newton's method, knowing the Hessian to be constant provides another choice: It is possible to invert the Hessian completely after computing it, and using it to solve (9) via $\hat{\mathbf{s}} = -\mathbf{H}^{-1}\mathbf{g}$, or applying Cholesky decomposition to separate it into two triangular matrices $\mathbf{H} = \mathbf{C}\mathbf{C}^{\mathsf{T}}$, and using substitution in each iteration steps to solve (9) for $\hat{\mathbf{s}}$. The former option, which we will denote Newton-INV, features a slower inversion step, but a faster iteration step only involving matrix multiplication; the latter option, denoted Newton-CD, features

[6] It should be noted at this point that here the issues discussed in Sect. 2 become apparent, where a lack of specified units will lead to different optimization goals for models in meters, feet or inches, for example. In this case, the unit is considered to be meters.

a faster initial decomposition step, but requires slower substitution operations at each iteration. In general it can be assumed that Newton-INV is faster in cases where many iterations are required, such that $\mathbf{H}^{-1}$ can be reused many times, while Newton-CD is faster for fewer iterations.

Figure 2 shows the comparison between Newton-CD and Newton-INV. In all evaluated cases, Newton-INV is faster, suggesting that already at 13 iterations

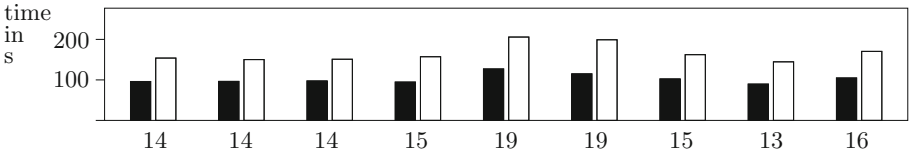


Fig. 2. Comparison of computation times for Newton-INV (black) and Newton-CD (white) on the 9 examples; the required iteration steps until convergence are given below the bars. In all cases, Newton-INV outperforms Newton-CD, at an average 62% of the latter's computation time.

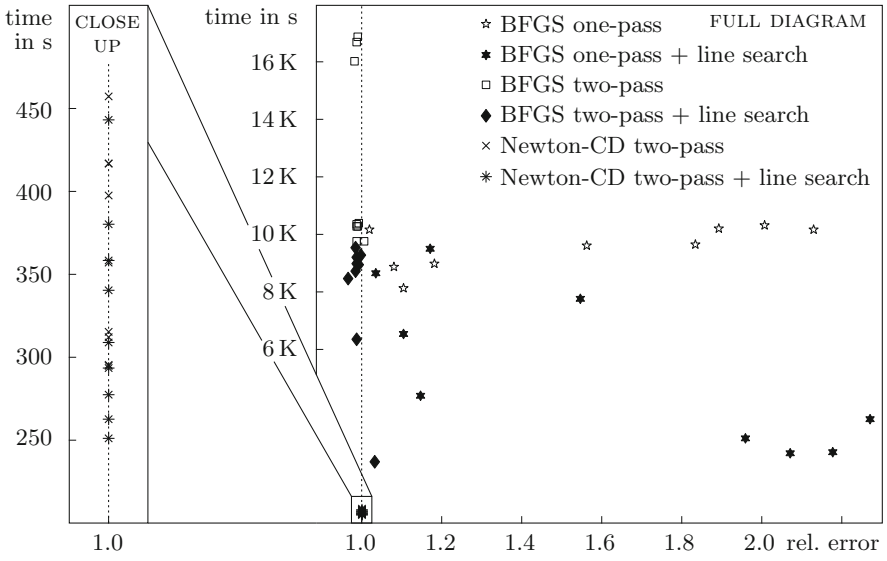


Fig. 3. Performance of Newton's method with Cholesky decomposition and on-the-fly substitution, and different variants of BFGS, computation time vs. result optimality (in error points normalized per mesh/shape combination such that the result of Newton's method is exactly 1. It can be seen that the two-pass BFGS methods usually achieve slightly better results, however at approximately 31× the computation time. One-pass method can match neither the speed nor the quality of Newton's method. Line search almost always reduces computation time considerably. A further acceleration with respect to the Cholesky decomposition, the full a-priori inversion of the Hessian, is given in Fig. 2.

the effort of once inverting **H** outweighs the effort of repeatedly substituting via the Cholesky-decomposed **H** (although implementation details may affect this). In all cases, the same minimum was found in the same number of steps by Newton-CD and Newton-INV, which appears obvious analytically, but indicates that numerical stability is not an issue when choosing between the two approaches.

Figure 3 shows the evaluation of L-BFGS with/without line search, and with one-pass and two-pass, while for Newton's method we only depict the slower Newton-CD and only differentiate between with and without line search. It can be seen that even Newton-CD vastly outperforms the L-BFGS methods in terms of computational effort, which are on average 31 times slower than the Newton-CD approach (and 50 times slower than the Newton-INV approach). The result quality varies; one-pass approaches perform poorly, while the two-pass approaches suggested in [5] usually slightly outperform Newton's method

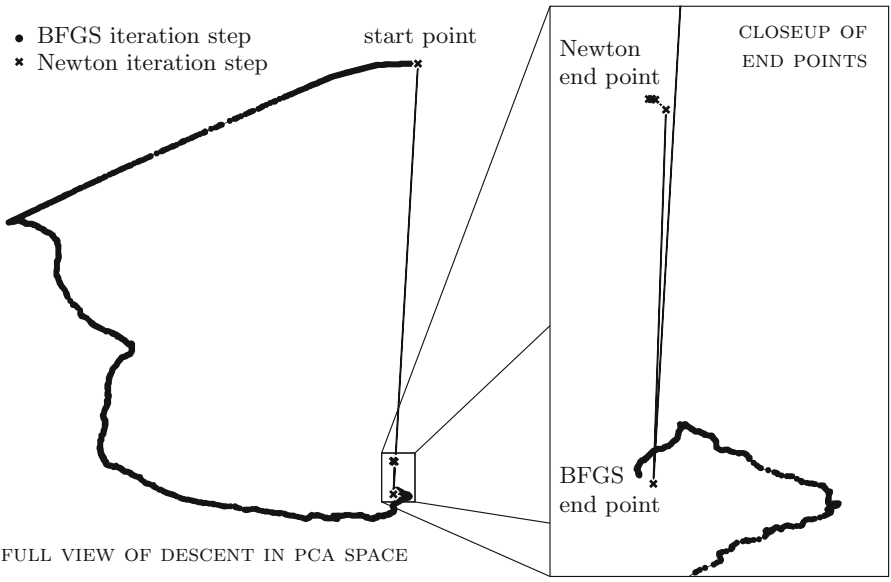


Fig. 4. To visually compare the paths that L-BFGS and Newton's method take, the respective paths of a given transformation optimization were reduced from $\mathbb{R}^{12\cdot|M|} = \mathbb{R}^{12\,024}$ to $\mathbb{R}^2$. It can be seen that the original L-BFGS approach takes a considerable detour involving many iteration steps, before even reaching the proximity of the first Newton step. L-BFGS terminates approximately there for lack of progress, while Newton's method descends several steps further after this, reaching a lower minimum after just 7 steps.

by an average of 1.06%[7]. Figure 4 visualizes the fundamentally different descent approaches by showing the paths taken by L-BFGS and Newton's method for one of the examples in a projection onto an abstract 2-dimensional space, achieved by computing a PCA on the paths and plotting the two most significant axes. It can be seen that L-BFGS requires many iterations before reaching the vicinity of the optimum; Newton's method on the other hand reaches this region after only one step, but settles in a different local optimum.

5 Conclusion and Outlook

In this paper, we have proposed an analytic solution to ICP algorithms with quadratic error terms, and demonstrated the approach on the example of [5]. It was shown that in such cases, the Hessian matrix is constant and can easily be computed analytically, instead of numerically approximating it during each optimization step, for example via L-BFGS, as is commonly done in ICP methods.

In the given example, an optimization descent with Newton's method using precomputed Hessians was able to reduce computation times on average to (not *by*) 2% of the computation times of BFGS when applied to the same problem. The quality of the optimization result of BFGS was on average slightly better, by around 1.06%, depending on the choice of parameters. Still it can be concluded that Newton's method can achieve very similar result qualities at a fraction of the computation time, rendering quadratic ICP approaches fit for real-time settings or to run on simpler hardware.

On the example of [5], all necessary parameters to adapt the original algorithm to the proposed analytic version were derived comprehensibly and can be readily implemented. Optimizational properties of the problem formulation were analyzed in detail to provide an understanding of the original and the newly proposed optimization process, and facilitate potential further adjustments.

In addition to the improvements in solution efficiency, we have proposed minor improvements to the original problem formulation of Allen et al. that make the formulation more mathematically sound, and allow to reuse weight parameters for models with different mesh triangulation or different units of measurement, without affecting the optimization goal. The original algorithm implicitly required to redetermine weights for different mesh triangulations, and for a different unit system.

[7] The approximative algorithm outperforming the exact one in terms of result quality may seem counter-intuitive, but is immanent to the problem structure: Newton's method is exact at finding the closest local optimum where point associations do not change. L-BFGS instead is more likely to miss this "direct" minimum. As its approximation combines information across different point associations, it can thereby "learn" the overall shape of the surface, which is, for high-density surface points, more accurate than the purely local quadratic view. In turn, for sparse surface models, Newtons method is more accurate, as it better captures the dominant structure.

The vast reduction of computation times in quadratic ICPs opens up new optimizational possibilities. As was noted in Sect. 4, Newton's method usually achieves a local minimum that is analytically exact, but slightly greater than the minimum found by the L-BFGS approach. However, as Newton's method required only 1/50 of the computation time, there is plenty of time to search the vicinity and match the result quality of BFGS, and still outperform it in terms of computation time.

References

1. Stewart, C.V., Tsai, C.L., Roysam, B.: The dual-bootstrap iterative closest point algorithm with application to retinal image registration. IEEE Trans. Med. Imaging **22**, 1379–1394 (2003)
2. Almhdie, A., Lger, C., Deriche, M., Lde, R.: 3D registration using a new implementation of the ICP algorithm based on a comprehensive lookup matrix: application to medical imaging. Pattern Recogn. Lett. **28**, 1523–1533 (2007)
3. Tam, G.K.L., Cheng, Z.Q., Lai, Y.K., Langbein, F.C., Liu, Y., Marshall, D., Martin, R.R., Sun, X.F., Rosin, P.L.: Registration of 3D point clouds and meshes: a survey from rigid to nonrigid. IEEE Trans. Vis. Comput. Graph. **19**, 1199–1217 (2013)
4. Allen, B., Curless, B., Popović, Z.: Articulated body deformation from range scan data. ACM Trans. Graph. **21**, 612–619 (2002)
5. Allen, B., Curless, B., Popović, Z.: The space of human body shapes: reconstruction and parameterization from range scans. ACM Trans. Graph. (TOG) **22**, 587–594 (2003)
6. Salazar, A., Wuhrer, S., Shu, C., Prieto, F.: Fully automatic expression-invariant face correspondence. Mach. Vis. Appl. **25**, 859–879 (2014)
7. Lu, X., Jain, A.: Deformation modeling for robust 3D face matching. IEEE Trans. Pattern Anal. Mach. Intell. **30**, 1346–1357 (2008)
8. Sumner, R.W., Popović, J.: Deformation transfer for triangle meshes. ACM Trans. Graph. **23**, 399–405 (2004)
9. Amberg, B., Romdhani, S., Vetter, T.: Optimal step nonrigid ICP algorithms for surface registration. In: 2007 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8 (2007)
10. Nocedal, J., Wright, S.J.: Numerical Optimization, 2nd edn. Springer, Berlin (2006)