Acceleration of Non-Rigid Point Set Registration With Downsampling and Gaussian Process Regression

Osamu Hirose, *Member, IEEE*

Abstract—Non-rigid point set registration is the process of transforming a shape represented as a point set into a shape matching another shape. In this paper, we propose an acceleration method for solving non-rigid point set registration problems. We accelerate non-rigid registration by dividing it into three steps: i) downsampling of point sets; ii) non-rigid registration of downsampled point sets; and iii) interpolation of shape deformation vectors corresponding to points removed during downsampling. To register downsampled point sets, we use a registration algorithm based on a prior distribution, called motion coherence prior. Using the same prior, we derive an interpolation method interpreted as Gaussian process regression. Through numerical experiments, we demonstrate that our algorithm registers point sets containing over ten million points. We also show that our algorithm reduces computing time more radically than a state-of-the-art acceleration algorithm.

Index Terms—Non-rigid point set registration, motion coherence prior, soft matching, downsampling, displacement field interpolation, Bayesian coherent point drift, Gaussian process regression

1 Introduction

Non-rigid point set re gistration is the process of finding deformation between shapes represented as point sets so that the deformed shape matches another shape. Various registration algorithms have been proposed in the field of computer vision and graphics because registered shapes have become a foundation for organizing, reconstructing, and synthesizing a broader class of shapes [1], [2], [3]. Point set registration algorithms can be classified into hard and soft matching on the basis of the decision procedure for point-to-point correspondences between two shapes. The former group includes registration algorithms based on iterative closest point (ICP) [4]. In such methods, point-to-point correspondences are assumed to be one-to-one. The main advantage of ICP-based methods is their scalability to large point sets, and variants of the ICP algorithm have been developed in the field of mesh registration [5], [6], [7], [8], [9]. The latter group comprises the registration methods in which point-to-point correspondences are assumed to be one-to-many, and this characteristic often contributes to their robustness [10], [11], [12], [13], [14], [15], [16], [17], [18].

One issue with the soft-matching methods is their computational costs; they require affinity computations involving all pairs of points between two shapes. To reduce these computational costs, Myronenko *et al*. [15] proposed a method based on fast Gauss transform (FGT) [19]; FGT accelerates summation of Gaussian functions, which is the bottleneck computation during soft matching. Golyanik *et al*. mentioned that their acceleration scheme requires approximately the same computational cost as a naive approach because it switches from FGT to truncated Gauss computation near convergence [20]. Recently, they proposed a registration method based on physical laws [21], [22]. They replaced the soft matching computation with a "gravity" computation between points included in two shapes. The calculation was accelerated by the Barnes–Hut octree [23], in which the gravity from distant points is roughly computed. Hirose proposed accelerating the soft-matching computation by selecting a suitable approximation technique during optimization [18]. The method switches from the Nyström method to a neighbor-search method with a KD tree near convergence, which helps preserve efficiency throughout the optimization. However, the tree-based, soft-matching methods can be computationally expensive for dense point sets because the number of neighboring points rapidly increases as point sets become dense. Apart from the soft and hard matching methods, several methods reduce the computational cost by initially selecting candidates of corresponding points using feature descriptors [24], [25], [26], [27]. These methods with feature descriptors sometimes produce shape surface regions where candidate corresponding points are sparse and reduce the registration quality for such regions.

Non-rigid registration algorithms involve an additional bottleneck computation apart from the soft-matching computation. The displacement field characterizing a nonlinear shape deformation is typically represented as a linear combination of kernel functions, the exact evaluation of which requires a Gram matrix for all points in a shape. Myronenko *et al*. proposed an acceleration scheme that avoids the exact Gram matrix evaluation by using an accelerated eigendecomposition with FGT [15]. Dupej *et al*. mentioned that the FGT-based eigendecomposition was time-consuming, and they proposed an approximate eigendecomposition [28] based on the Nyström method [29] and the improved fast Gauss transform (IFGT) [30]. One drawback of these methods is that the acceleration is limited to the case of Gaussian functions. Hirose proposed a method that can be applied to non-Gaussian functions by approximating eigendecomposition through the Nyström method alone [18]. Other approaches to avoiding the bottleneck computation used random subsampling of control points, which reduces the number of basis functions that define a shape deformation [24], [25], [26], [27], [31]. Computational costs of the approximation techniques based on the eigendecomposition or the random subsampling are proportional to the number of points in a point set. However, the approximate computations can still be costly for large point sets because the corresponding proportionality constants are relatively large.

Despite the efforts to accelerate non-rigid registration algorithms based on soft matching, their scalability to large point sets is still limited. To address this issue, we propose an acceleration method for non-rigid point set registration. Our method, summarized in Fig. 1, accelerates non-rigid registration via i) downsampling of point sets, ii) non-rigid registration of the downsampled point sets, and iii) interpolation of shape deformation vectors corresponding to the removed points. To register downsampled point sets while preserving the merits of soft matching, we use Bayesian coherent point drift (BCPD) [18], a non-rigid registration algorithm based on soft matching. To interpolate the shape deformation vectors, we derive a predictive distribution of an arbitrary shape deformation vector from the motion coherence prior [18], i.e., a Gaussian process prior, employed as a basis of BCPD. We demonstrate that our algorithm is scalable to more than ten million points through numerical experiments. We evaluate its registration accuracy and runtime using datasets with synthetic and non-artificial deformations.

2 Method

In non-rigid point set registration, a non-linear map $\mathcal{T}$ is found that transforms the shape represented as a point set $Y = \{y_1, \ldots, y_M\}$ into a shape matching the shape represented as another point set

• *The author is with the Institute of Science and Engineering, Kanazawa University, Kakuma, Kanazawa, Ishikawa 920-1192, Japan. E-mail: hirose@se.kanazawa-u.ac.jp.*

Manuscript received 20 Aug. 2020; revised 17 Nov. 2020; accepted 7 Dec. 2020. Date of publication 10 Dec. 2020; date of current version 1 July 2021.
(Corresponding author: Osamu Hirose.)
Recommended for acceptance by V. Koltun.
Digital Object Identifier no. 10.1109/TPAMI.2020.3043769

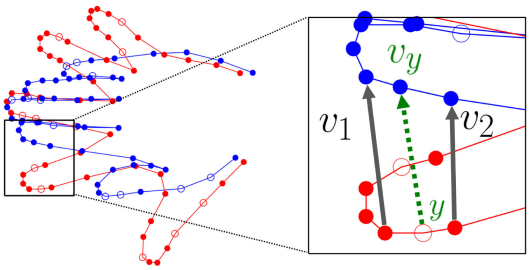


Fig. 1. Our approach to accelerating non-rigid registration. The target and source point sets are colored blue and red, respectively. Blank circles represent the points removed during downsampling. Gray arrows v_1 and v_2 represent displacement vectors estimated from the downsampled point sets. After downsampling and registration, we interpolate the displacement vectors corresponding to the removed source points using Gaussian process regression.

$X = \{x_1, \ldots, x_N\}$. In this section, we propose an acceleration method for solving non-rigid registration problems.

2.1 Notation

Here, we define the notation used throughout this paper. We refer to the point set to be deformed as a source point set and refer to the other point set that remains fixed as a target point set. We denote the target and source point sets by X and Y, respectively. For convenience, we do not differentiate the symbol for a point set and that for its matrix representation; e.g., we use the same symbol Y for the point set $\{y_1 \ldots, y_M\}$ and its matrix representation $(y_1, \ldots, y_M)^T$. Throughout this paper, we use the following notation:

- D – the dimensionality of the space in which a point set is embedded.
- N, M – the numbers of points in target and source point sets.
- $X = (x_1, \ldots, x_N)^T \in \mathbb{R}^{N \times D}$ – a target point set.
- $Y = (y_1, \ldots, y_M)^T \in \mathbb{R}^{M \times D}$ – a source point set.
- $V = (v_1, \ldots, v_M)^T \in \mathbb{R}^{M \times D}$ – a matrix collecting displacement vectors that deform a source shape Y.
- $\rho(\cdot)$ – a similarity transformation.
- N', M' – the numbers of points after downsampling X and Y.
- $X' = (x'_1, \ldots, x'_{N'})^T \in \mathbb{R}^{N' \times D}$ – a point set obtained by downsampling X.
- $Z = (z_1, \ldots, z_{M'})^T \in \mathbb{R}^{M' \times D}$ – a point set obtained by downsampling Y.
- $V_Z = (v'_1, \ldots, v'_{M'})^T \in \mathbb{R}^{M' \times D}$ – a matrix collecting vectors that deform a downsampled source shape Z.
- $\varphi(\cdot) : \mathbb{R}^{M' \times D} \to \mathbb{R}^{M \times D}$ – a function that interpolates V from V_Z.
- $\mathcal{K}(\cdot, \cdot) : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ – a positive definite kernel that defines motion coherence.
- $G_{AB} = (\mathcal{K}(a_i, b_j)) \in \mathbb{R}^{N_A \times N_B}$ – a Gram matrix for arbitrary point sets $A \in \mathbb{R}^{N_A \times D}$ and $B \in \mathbb{R}^{N_B \times D}$, where $a_i \in \mathbb{R}^D$ and $b_j \in \mathbb{R}^D$ represent the ith point in A and the jth point in B, respectively.

2.2 Problem Definition

We define a point set registration problem as an optimization problem of finding a similarity transformation ρ and a displacement matrix $V \in \mathbb{R}^{M \times D}$ corresponding to non-rigid deformation as follows:

$$(\hat{V}, \hat{\rho}) = \arg\min_{(V,\rho)} \mathcal{L}(V, \rho, X, Y), \tag{1}$$

where $\mathcal{L}$ is a loss function evaluating the dissimilarity between $X \in \mathbb{R}^{N \times D}$ and $T = \rho(Y + V) \in \mathbb{R}^{M \times D}$. Some formulations of non-rigid registration include soft-matching weights as additional arguments to be optimized. We omitted the weights because they can typically be represented as a function of (V, ρ, X, Y).

2.3 Motivation

One issue with non-rigid registration methods based on soft matching is that the computational cost can be prohibitive because a naive computation of soft-matching weights requires $O(MN)$ computations. Even if we use acceleration techniques such as FGT [19] and IFGT [30], non-rigid registration of point sets with more than a million points will still be time-consuming.

A naive approach to reducing the computational cost is to use a downsampling technique, which reduces the number of points in a point set. Applying a non-rigid registration method to downsampled point sets, $X' \in \mathbb{R}^{N' \times D}$ and $Z \in \mathbb{R}^{M' \times D}$, we obtain a point set $T_Z \in \mathbb{R}^{M' \times D}$ that matches X', defined as follows:

$$T_Z = \hat{\rho}(Z + \hat{V}_Z),$$

where $\hat{V}_Z \in \mathbb{R}^{M' \times D}$ is a matrix comprising displacement vectors corresponding to Z. However, this approach creates another issue; the displacement vectors corresponding to the points removed during downsampling are unknown. Even if we interpolate a resulting deformed shape T_Z using a smoothing technique, the displacement vectors corresponding to the removed points cannot be recovered, and the original registration problem, Eq. (1), remains unsolved.

2.4 Our Approach

To address the issue originating from downsampling, we approximately solve the optimization problem, Eq. (1), through the following subproblems:

$$(\hat{V}_Z, \hat{\rho}) = \arg\min_{(V_Z,\rho)} \mathcal{L}(V_Z, \rho, X', Z), \tag{2}$$

$$\hat{V} = \arg\min_{V=\varphi(\hat{V}_Z)} \mathcal{L}(V, \hat{\rho}, X, Y), \tag{3}$$

where $\varphi : \mathbb{R}^{M' \times D} \to \mathbb{R}^{M \times D}$ is an interpolating function. The former problem is to solve a non-rigid registration problem for downsampled point sets, whereas the latter problem is to interpolate V from $\hat{V}_Z$ such that V minimizes the loss function $\mathcal{L}(V, \hat{\rho}, X, Y)$. We use the BCPD algorithm to solve the former problem, and the computational cost of the optimization becomes $O((N' + M')\log(N' + M'))$ owing to its acceleration scheme [18]. Hereafter, we focus on the latter problem.

2.4.1 Motion Coherence Prior

We indirectly solve the optimization subproblem, Eq. (3), by directly defining a reasonable interpolating function φ. The basis for the interpolation is the motion coherence prior [18], defined as a Gaussian distribution as follows:

$$p(\mathrm{vec}(V)|Y) = \mathcal{N}(0, \lambda^{-1} G_{YY} \otimes I_D), \tag{4}$$

where $\mathrm{vec}(V)$ is the vector representation of the displacement matrix V in row-major order, I_D is the identity matrix of size D, the symbol $\otimes$ is the Kronecker product, and $\lambda > 0$ is a positive constant that controls the expected length of displacement vectors. This distribution is a basis of BCPD and is interpreted as a Gaussian process prior [32]. Intuitively, this assumption implies that the displacement vectors of neighboring points correlate with each other, and therefore, the displacement vector of a source point can be estimated from those of neighboring points after downsampling and registration.

2.4.2 Interpolation With Gaussian Process Regression

We define an interpolating function φ using the predictive distribution derived from the motion coherence prior. Suppose $v_y \in \mathbb{R}^D$ is the displacement vector corresponding to an arbitrary point $y \in \mathbb{R}^D$. The motion coherence prior, Eq. (4), implies that the expectation of

v_y regarding the predictive distribution $p(v_y|y, V_Z, Z)$ is represented as follows:

$$\mathrm{E}[v_y|y, V_Z, Z] = V_Z^T G_{ZZ}^{-1} g_Z(y), \tag{5}$$

where $g_Z(y) : \mathbb{R}^D \to \mathbb{R}^{M'}$ is an M'-valued function, the m'th value of which is defined as $\mathcal{K}(y, z_{m'})$. Because the motion coherence prior is a Gaussian process prior, the estimation of displacement vectors based on Eq. (5) is interpreted as Gaussian process regression [32]. Augmenting the expected displacement vectors corresponding to all points in Y using Eq. (5), we define the interpolating function φ as follows:

$$\varphi(V_Z) = G_{YZ} G_{ZZ}^{-1} V_Z. \tag{6}$$

As the ground truth of V_Z is unknown, we replace V_Z with an estimate $\hat{V}_Z$, i.e., a solution for the subproblem, Eq. (2). We use the BCPD algorithm to obtain $\hat{V}_Z$ because the algorithm is based on the same assumption, i.e., the motion coherence prior.

2.4.3 Reformulation of BCPD Displacement Vectors

To simplify the resulting displacement vectors $\hat{V}$ and reduce the computational cost of the interpolating function φ, we reformulate displacement vectors $\hat{V}_Z$, the original formula of which is available in Proposition 1 in [18], as follows:

$$\hat{V}_Z = G_{ZZ}(G_{ZZ} + \Psi)^{-1} E, \tag{7}$$

where Ψ is a diagonal matrix of size $M' \times M'$ that controls the smoothness of $\hat{V}_Z$ and E is a matrix of size $M' \times D$ that collects non-smooth displacement vectors. These matrices are defined as follows:

$$\Psi = \frac{\lambda\sigma^2}{s^2}\mathrm{diag}(P1_{M'})^{-1},$$
$$E = \hat{\rho}^{-1}(\mathrm{diag}(P1_{M'})^{-1} PX') - Z,$$

where $1_{M'}$ is the vectors of all 1s of size M', and s, σ^2, and $P \in \mathbb{R}^{M' \times N'}$ are variables output by BCPD; s is a scale factor constituting the similarity transformation $\hat{\rho}$, σ^2 is the estimated residual variance, and P is the matching probability matrix between Z and X'.

2.4.4 Resulting Formula for Shape Deformation Vectors

We use displacement vectors $\hat{V} = \varphi(\hat{V}_Z)$ resulting from the Gaussian process regression as an approximate optimizer for the subproblem, Eq. (3). Substituting V_Z in Eq. (6) with $\hat{V}_Z$ in Eq. (7), we obtain an estimate of the displacement vectors corresponding to source points before downsampling as follows:

$$\hat{V} = \varphi(\hat{V}_Z) = G_{YZ}(G_{ZZ} + \Psi)^{-1} E. \tag{8}$$

We see that Eq. (8) is closely related to a well-known smoothing technique [33]. We also note that this smoothing technique works even if no points are shared by Y and Z, and therefore, any downsampling techniques can be applied, e.g., the voxel grid filter. The computational cost of evaluating this equation is $O(MM' + M'^3)$, becoming an efficient computation if M' is sufficiently small, e.g., $M' = 10^3$.

2.4.5 Further Acceleration With Low-Rank Approximation

The direct evaluation of Eq. (8) is still computationally expensive for a moderately large M', e.g., $M' = 10^4$, owing to the $O(MM' + M'^3)$ computation. We accelerate the evaluation of Eq. (8) through the following approximations:

- The approximation of G_{YZ} using the Nyström method.
- The approximation of $\Phi = (G_{ZZ} + \Psi)^{-1} E$ through the eigendecomposition of G_{ZZ} based on the Nyström method.

We begin by accelerating the computation of G_{YZ}. Suppose $U \in \mathbb{R}^{L \times D}$ is a subset of $Y \cup Z$, typically extracted by the random sampling of L points in the union. Then, we approximate G_{YZ} using the Nyström method as follows:

$$G_{YZ} \approx G_{YU} G_{UU}^{-1} G_{UZ}.$$

The sizes of G_{YZ}, G_{UU}, and G_{UY} are $M \times L$, $L \times L$, and $L \times M'$, respectively. Therefore, the computational cost of evaluating the product of G_{YZ} and Φ decreases from $O(MM')$ to $O(M + M')$.

Next, we proceed to the evaluation of Φ, which can also be a bottleneck because the inversion of the matrix $(G_{ZZ} + \Psi) \in \mathbb{R}^{M' \times M'}$ requires $O(M'^3)$ time. To accelerate it, we approximate G_{ZZ} using the eigendecomposition as follows:

$$G_{ZZ} \approx Q \Lambda Q^T,$$

where $Q \in \mathbb{R}^{M' \times L}$ is a column-orthonormal matrix comprising L eigenvectors, and $\Lambda \in \mathbb{R}^{L \times L}$ is a diagonal matrix comprising the corresponding eigenvalues. Because the eigendecomposition itself is a bottleneck, we accelerate it using the Nyström method [29], which performs approximate eigendecomposition with L leading eigenvectors in $O(M')$ time. Then, we calculate Φ in $O(M')$ time using the Woodbury identity as follows:

$$\Phi \approx \Psi^{-1}\{I - Q(\Lambda^{-1} + Q^T \Psi^{-1} Q)^{-1} Q^T \Psi^{-1}\} E.$$

Therefore, the computational cost of evaluating Eq. (8) becomes $O(M + M')$, which is much smaller than that of the exact evaluation of Eq. (8), i.e., $O(MM' + M'^3)$. We call the resulting algorithm BCPD++ and summarize it in Algorithm 1.

Algorithm 1. BCPD++

Input: $X \in \mathbb{R}^{N \times D}$, $Y \in \mathbb{R}^{M \times D}$, N', M', L.
Output: $T = \hat{\rho}(Y + \hat{V})$.
a) **Downsampling**:
 · Generate X' and Z of sizes $N' \times D$ and $M' \times D$ by applying a downsampling method to X and Y, respectively.
b) **Registration**:
 · Find $T_Z = \hat{\rho}(Z + \hat{V}_Z)$ that matches X' by applying the BCPD algorithm to X' and Z, and compute Ψ and E that satisfy $\hat{V}_Z = G_{ZZ}(G_{ZZ} + \Psi)^{-1} E$.
c) **Interpolation**:
 · Decompose G_{ZZ} into $Q\Lambda Q^T$ s.t. the number of eigenvalues is L using the Nyström method.
 · $\Phi = \Psi^{-1}\{I - Q(\Lambda^{-1} + Q^T\Psi^{-1}Q)^{-1}Q^T\Psi^{-1}\}E$.
 · Generate $U \subset Y \cup Z$ s.t. the number of points in U is L, and compute G_{YU}, G_{UU}, and G_{UZ}.
 · $T = \hat{\rho}(Y + \hat{V})$, where $\hat{V} = G_{YU} G_{UU}^{-1} G_{UZ} \Phi$.

3 Experiments

In this section, we evaluate the registration performance of BCPD++ using datasets with synthetic and non-artificial deformations.

3.1 Computational Environment

We used a Mac Mini (2018, macOS 10.14.6) with a 3.2 GHz Intel Core i7 CPU (six cores) and 64 GB RAM. We implemented BCPD and BCPD++ in C and used Apple clang 11.0.0 as a C compiler. We parallelized BCPD and BCPD++ using OpenMP and measured the wall-clock time.

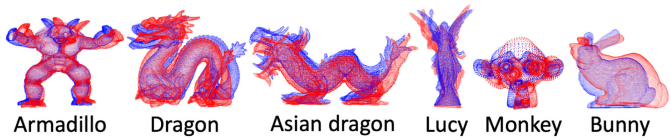


Fig. 2. Synthetic datasets. Each shape colored red was created by deforming the shape colored blue non-linearly. Asian Dragon and Lucy were downsampled for visualization. In experiments, we used red and blue shapes as the source and target shapes, respectively.

3.2 Evaluation Measure

For datasets containing the ground truth, we evaluate registration accuracy using a scale-invariant accuracy measure based on the root-mean-squared distance (RMSD). For point sets $A \in \mathbb{R}^{M\times D}$ and $B \in \mathbb{R}^{M\times D}$ with point-to-point correspondences, we define the RMSD, denoted by $r(A, B)$, as follows:

$$r(A, B) = \sqrt{\mathrm{Tr}\{(A - B)^T(A - B)\}/M}.$$

Then, we define an accuracy measure as follows:

$$\text{Accuracy} = 1 - r(X, T)/r(X, Y),$$

where $T \in \mathbb{R}^{M\times D}$ is a deformed shape. The accuracy becomes one for perfect registration, and it becomes zero if the source shape Y and the deformed shape T are identical. The accuracy becomes negative if $r(X, T) \geq r(X, Y)$.

3.3 BCPD Parameters

BCPD++ requires the parameters of BCPD along with the type of a kernel function because BCPD++ uses BCPD at the registration step. For all experiments, we used the Gaussian kernel. The set of BCPD parameters consists of ω, λ, β, γ, and κ, which control outlier probability, the expected length of deformation vectors, the degree of motion coherence, the randomness regarding the initial guess of point matching, and the uniformity of the number of target points matching with a source point, respectively. We used $(\omega, \lambda, \beta, \gamma, \kappa) = (0, 50, 2, 10, \infty)$ unless otherwise noted. We heuristically chose these parameters because we had no sophisticated parameter selection algorithm.

BCPD has its internal acceleration scheme, which is irrelevant to the downsampling and interpolation outside BCPD. This internal acceleration usually reduces computing time without sacrificing registration accuracy [18]. Therefore, we always activated the internal acceleration with the acceleration parameters $(J, K) = (300, 70)$, which we heuristically selected.

3.4 Downsampling Method

We implemented a resampling algorithm motivated by the voxel-grid filter. The algorithm divides the space embedding a point set into voxels of the same size, as in the voxel-grid filter. Then, it resamples points so that the expected numbers of resampled points are equivalent in all voxels containing at least a point. We call the method voxel-grid resampling. A merit of the resampling algorithm is that the number of resampled points can be specified, and therefore, we can use it as a downsampling technique. We defined a voxel as a cube with an edge length of 0.08. A performance evaluation of this algorithm is available in Appendix A , which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2020.3043769.

3.5 Performance Evaluation With Large Data

In this section, we evaluate the performance of BCPD++ using large point sets that contain at least 100,000 points.

3.5.1 Data

We used four shapes, namely Armadillo, Dragon, Asian Dragon, and Lucy, downloaded from the Stanford 3D scanning repository (https://graphics.stanford.edu). The numbers of points in the shapes are 106,289, 437,645, 3,609,446, and 14,027,872, respectively. For Asian Dragon, we randomly extracted 1,000,000 points.

3.5.2 Synthetic Deformation

For a shape denoted by $A \in \mathbb{R}^{M\times D}$, we created a deformed shape $B = A + V \in \mathbb{R}^{M\times D}$ preserving point-to-point correspondences, where $V \in \mathbb{R}^{M\times D}$ is a deformation matrix sampled from the motion coherence prior with the covariance matrix $\lambda^{-1}G_{AA}$. Because the motion coherence prior is a multivariate normal distribution, we can sample a deformation matrix $V \in \mathbb{R}^{M\times D}$ via the eigendecomposition of the covariance matrix and the generation of random numbers following the standard normal distribution [34].

To reduce the computational cost of the eigendecomposition, we performed approximate eigendecomposition based on the Nyström method [29]. We defined G_{AA} based on the Gaussian kernel with the radius parameter 2.0, and we randomly chose λ within the range of 10 to 100. We set the number of Nyström samples to 100. This procedure creates a smoothly deformed shape because the motion coherence prior implies that deformation vectors of neighboring points correlate with each other. Fig. 2 shows the original and corresponding deformed shapes.

3.5.3 Demonstration

Fig. 3 shows an application of BCPD++ to Lucy data containing more than ten million points under the acceleration parameters $M' = N' = 50,000$ and $L = 100$. We see that BCPD++ appears to succeed in registering the point sets containing more than 10^7 points. We report the registration accuracy for this dataset in Section 3.5.5.

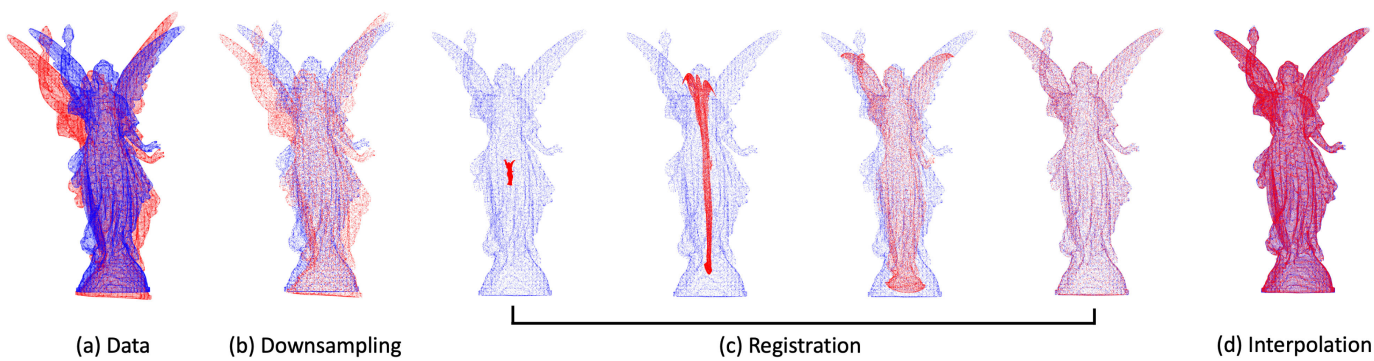


Fig. 3. Application to the Lucy data containing more than ten million points. (a) Input point sets. (b) Point sets downsampled by voxel grid resampling. (c) Registration of the downsampled point sets by the BCPD algorithm. (d) Interpolation of the displacement vectors corresponding to removed source points. For visualization, the dataset (a) and the result (d) were downsampled, although BCPD++ was applied to the original dataset.

TABLE 1
Comparison Using Datasets Containing at Least 100,000 Points

	Data	#points	BCPD	BCPD++	1NN
Down-sampling	Armadillo	106K	-	0.008	0.009
	Dragon	438K	-	0.042	0.042
	A. Dragon	1M	-	0.067	0.086
	Lucy	14M	-	0.856	0.918
Registration	Armadillo	106K	60.7	29.9	30.3
	Dragon	438K	378.4	34.0	33.8
	A. Dragon	1M	3834	24.8	28.5
	Lucy	14M	-	43.6	45.5
Interpolation	Armadillo	106K	-	1.82	0.17
	Dragon	438K	-	3.34	0.47
	A. Dragon	1M	-	5.60	1.09
	Lucy	14M	-	63.6	12.7
Total time	Armadillo	106K	60.8	31.8	30.6
	Dragon	438K	379.3	38.4	35.3
	A. Dragon	1M	3835	31.3	30.5
	Lucy	14M	-	119.1	70.9
Accuracy	Armadillo	106K	0.956	0.945	0.937
	Dragon	438K	0.982	0.955	0.946
	A. Dragon	1M	0.994	0.967	0.949
	Lucy	14M	-	0.942	0.932

Values in the first four rows and in the last row represent the average computing time (s) and the average registration accuracy for ten trials, respectively. The total time includes I/O, downsampling, registration, and interpolation time. The '1NN' column indicates the results of a method comprising voxel-grid resampling, BCPD registration, and one-nearest neighbor interpolation, i.e., a baseline interpolation technique. For Lucy, BCPD did not converge within 24 hours.

3.5.4 Evaluation

We evaluated the registration performance of BCPD++ by comparing it with BCPD. For BCPD++, we set the acceleration parameters to $N' = M' = 50,000$ and $L = 100$. As a baseline, we added the results of one-nearest-neighbor (1NN) interpolation. The 1NN method estimates the displacement vector of a removed point as the displacement vector of the nearest point in a source shape.

3.5.5 Results

Table 1 presents the result of the evaluation. From the table, we see that BCPD++ registered point sets more accurately than the 1NN interpolation. We also see that, although the registration accuracies of BCPD++ moderately decreased, the computing times of BCPD++ were noticeably smaller than those of BCPD. In particular, for Lucy data, the average computing time of BCPD++ was under two minutes, while BCPD did not converge within 24 hours. These results suggest an advantage of BCPD++ over BCPD for point sets containing more than 10^7 points.

3.5.6 Effect of Acceleration Parameters

We evaluated the effect of acceleration parameters N', M', and L. First, we measured computing time and evaluated registration accuracy under $L = 100$, changing M' and N' from 10^4 to 10^5 at intervals of 10^4. For each M', we repeated the experiment 10 times and calculated the average registration accuracy and the average computing time. Here, we defined computing time as the elapsed time during I/O, downsampling, BCPD execution, and displacement field interpolation. Fig. 4a depicts the result of the evaluation. We see that the registration accuracies and computing times increased as the numbers of subsampled points N' and M' increased, except for Armadillo, suggesting a trade-off between registration accuracy and computing time.

Then, we measured computing times and evaluated registration accuracies under $M' = N' = 50,000$, changing L from 40 to 150 at intervals of 10. Fig. 4b shows the result. We see that computing times increased as L increased. In contrast, the registration accuracies were approximately the same for $L \geq 90$, suggesting that $L = 100$ is sufficient for these datasets.

3.5.7 Effect of Low-Rank Approximation

To investigate the effect of the low-rank approximations described in Section 2.4.5, we directly evaluated Eq. (8) without the low-rank approximations. We compared the direct evaluation with BCPD++ and BCPD using the Armadillo dataset. For convenience, we call this direct evaluation method BCPD+. For BCPD+ and BCPD++, we changed the acceleration parameters M' and N' from 10,000 to 50,000 at intervals of 5,000. We used $L = 100$ for BCPD++.

Fig. 4c shows the result of the evaluation. For all $M' = N'$, the computing times of BCPD++ were smaller than those of BCPD+, and the difference in registration accuracy was slight. The computing time of BCPD+ rapidly increased as $M' = N'$ increased and was greater than that of BCPD at $M' = N' \geq 30,000$. BCPD+ execution failed at $M' = N' = 50,000$ because our computational environment did not allow $O(M'^2)$ memory allocation required for G_{ZZ}. These results suggest the need for the low-rank approximations under relatively large M' and N', e.g., $M' = N' \geq 30,000$.

3.6 Performance Evaluation With Small Data

In this section, we compare BCPD++ with BCPD [18], CPD [15], GMM-REG [16], and TPS-RPM [13]. We used point sets with artificial disturbance to investigate the robustness of the registration methods. We synthetically generated point sets containing at most 1,200 points so that registration methods without an acceleration scheme could handle them.

3.6.1 Data

We used two shapes called Monkey and Bunny, shown in Fig. 2. We downloaded Bunny from the Stanford scanning repository and

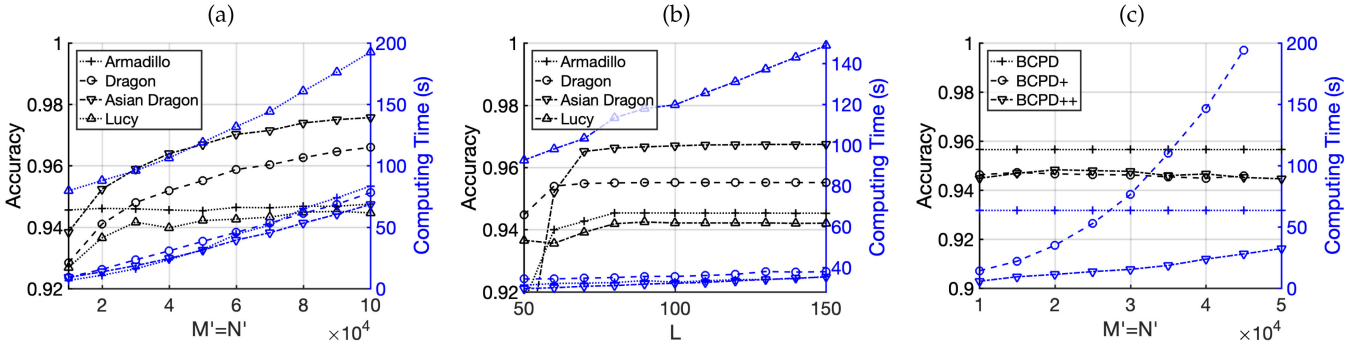


Fig. 4. (a) Registration accuracy and computing time versus the number of downsampled points M' and N' under $L = 100$. (b) Registration accuracy and computing time versus the number of Nyström points L under $M' = N' = 50,000$. (c) Effect of low-rank approximations for the Armadillo dataset. BCPD+ indicates the acceleration without low-rank approximations. BCPD+ execution failed at $M' = N' = 50,000$ because of excessive memory consumption.

TABLE 2
Comparison Using Small Data With Artificial Disturbance

Method	Outlier 20%	Noise 20%	Hole r=0.2
CPD [15]	0.576	0.942	0.945
GMM-REG [16]	0.880	0.186	0.883
TPS-RPM [13]	0.970	0.969	0.935
BCPD [18]	0.980	0.977	0.979
BCPD++	0.849	0.805	0.930

A value in the table represents the median registration accuracy for 20 shapes comprising Monkey and Bunny shapes with artificial disturbance.

extracted Monkey from Blender. The numbers of points in Monkey and Bunny were 7,958 and 35,947. We created deformed shapes that preserve point-to-point correspondences via the same procedure described in Section 3.5.2 and randomly extracted 1,000 points for each shape.

3.6.2 Addition of Artificial Disturbance

For the target shapes in the Monkey and Bunny datasets, we added three types of artificial disturbances: outliers simulating isolated outliers, noise simulating clustered outliers, and a hole simulating occlusion and partial overlaps. For each shape and type of artificial disturbance, we created ten shapes by repeating the following procedure ten times.

a) *Outliers*. We synthesized 200 outliers that follow a uniform distribution with the bounding box surrounding a target point set. We set the edge length of the bounding box along the dth axis to $\max_n\{x_{nd}\} - \min_n\{x_{nd}\}$, where x_{nd} is the dth element of x_n.

b) *Noise*. We synthesized 200 clustered outliers that follow an isotropic Gaussian distribution with standard deviation 0.1 after normalizing a target point set. We randomly selected a point in the target point set as the center of the distribution.

c) *Hole*. We removed points inside a ball with a radius $r = 0.2$ from a target point set after normalizing a target point set. We randomly selected a point in the target point set as the center of the ball.

3.6.3 Evaluation

For each type of artificial disturbance, we created a dataset comprising Monkey and Bunny shapes with the artificial disturbance. Using the dataset, we calculated the median accuracy for each method.

3.6.4 Parameters

For parameters shared by BCPD and BCPD++, we used the parameters described in Section 3.3 except ω and γ. We used $\omega = 0.1$, which specifies outlier probability 0.1, and $\gamma = 1.0$, which specifies the same initialization performed by CPD. For BCPD++, we set the acceleration parameters $M' = N' = 500$ and $L = 100$. For CPD, we used $(\omega, \lambda, \beta) = (0.1, 50, 2)$, which were the same as those used for BCPD and BCPD++. For TPS-RPM, we used `T_finalfac=500`, `frac=1`, and `T_init=1.5`. For GMM-REG, we used the following parameters:

```
method='TPS_L2', normalize=1, level=3,
sigma=[0.5,0.1,0.02], lambda=0,
max_function_evals=[20,50,50].
```

3.6.5 Results

Table 2 shows the results of the evaluation. We see that the median registration accuracies of BCPD++ were over 0.8 for all artificial disturbances. We also see that BCPD++ was relatively sensitive to the artificial disturbance in comparison with BCPD, which achieved the highest registration accuracy among the five methods for all data types.

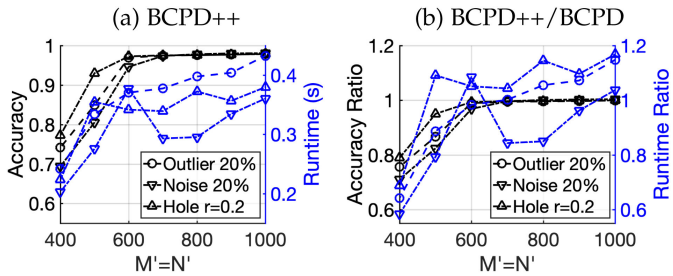


Fig. 5. Effect of acceleration parameters M' and N' on BCPD++ performance for small, noisy data. (a) Registration accuracy and runtime. (b) Comparison between BCPD++ and BCPD. The left and right vertical axes represent the ratio of BCPD++ accuracy to BCPD accuracy and the ratio of BCPD++ runtime to BCPD runtime, respectively.

3.6.6 Effect of Acceleration Parameters

We evaluated the registration accuracy and runtime for BCPD++, changing the acceleration parameters $M' = N'$ from 400 to 1,000 at intervals of 100 under $L = 100$. Fig. 5a shows the registration accuracies and runtimes of BCPD++. It required $M' = N' = 600$ to achieve a registration accuracy of 0.9 for all disturbances. Fig. 5b presents the accuracy ratio and runtime ratio of BCPD++ to BCPD. In this figure, a runtime ratio of 1.0 corresponds to 0.38 s, 0.35 s, and 0.33 s for datasets with outliers, noise, and holes, respectively. Additionally, an accuracy ratio of 1.0 corresponds to 0.980, 0.977, and 0.979 for datasets with outliers, noise, and holes, respectively. We see that the runtime ratios of BCPD++ to BCPD were within the range of 0.5 to 1.2, suggesting that the computing times of BCPD++ were not always smaller than those of BCPD. These results suggest the superiority of BCPD over BCPD++ for small, noisy data.

3.7 Performance Evaluation With SHREC'19 Data

In this section, we demonstrate that BCPD++ can handle non-artificial deformations.

3.7.1 Data and Pre-Processing

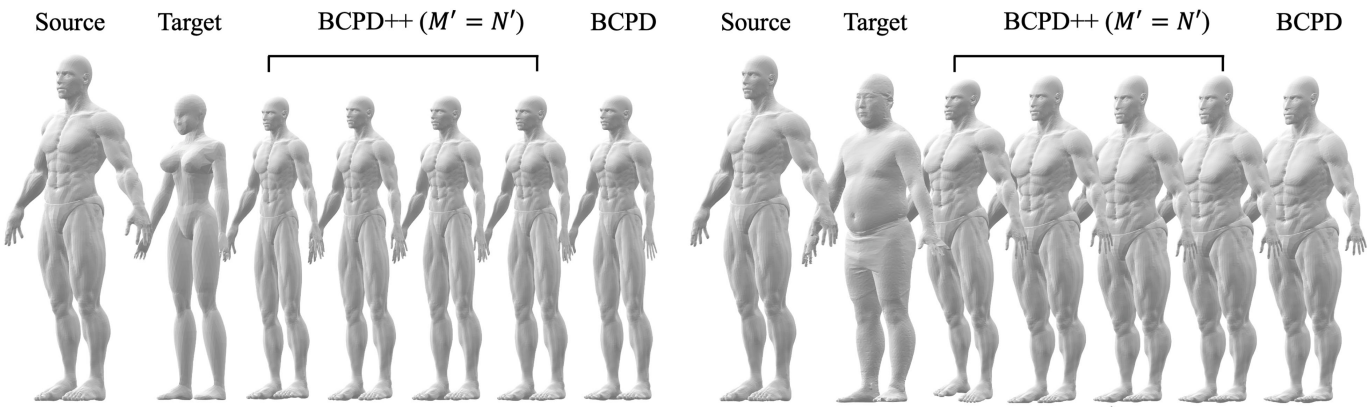
We used human body data taken from a SHREC'19 track, called "matching humans with different connectivity" [35]. Among them, we selected ten shapes with an upright posture. Using these shapes, we created two datasets for qualitative and quantitative evaluations. The first dataset contains two pairs of shapes without the ground truth, and the second contains ten pairs of shapes with the ground truth. We manually rotated these shapes and roughly embedded them into the global coordinate system before registration because each shape was located in a local coordinate system.

3.7.2 Qualitative Evaluation

We used shape no. 20 as the source shape and shapes no. 1 and 9 as target point sets. For BCPD++, we changed acceleration parameters M' and N' to 500, 1,000, 2,000, and 4,000 under $L = 100$. Fig. 6 shows a source shape, target shapes, and the resulting shapes deformed by BCPD++ and BCPD. From the figure, we see that all resulting shapes approached the corresponding target shapes. We also see that the shapes deformed by BCPD++ approached those deformed by BCPD as M' and N' increased. The RMSDs listed in the tables attached to the figure support this observation. The tables also show that BCPD++ runtimes decreased compared with BCPD.

3.7.3 Quantitative Evaluation

We quantitatively evaluated BCPD++ performance using the ten pairs of shapes with the ground-truth correspondences. For BCPD



M=25,875 N=4,706	(500)	(1,000)	(2,000)	(4,000)	
Computing time (s)	0.246	0.704	2.645	11.13	89.63
#VB loops	148	90	404	832	1,000
RMSD	0.0311	0.0189	0.0098	0.0086	—

M=25,875 N=15,154	(500)	(1,000)	(2,000)	(4,000)	
Computing time (s)	0.450	0.665	1.330	2.227	17.80
#VB loops	57	53	109	79	132
RMSD	0.0560	0.0325	0.0140	0.0097	—

Fig. 6. Application to human body data. The numbers in parentheses represent acceleration parameters M' and N' for BCPD++. The computing times listed in the attached tables include I/O, downsampling, registration, and interpolation time. The row '#VB loops' indicates the number of iterations required for convergence in the registration step. RMSD represents the root-mean-squared distance between the shapes deformed by BCPD++ and BCPD.

++, we used the acceleration parameters $M' = N' = 2,000$ and $L = 100$. Table 3 shows the result of the evaluation. BCPD++ registered all the ten pairs of the shapes in less computing time than BCPD did. BCPD++ outperformed BCPD for six of the ten pairs in terms of registration accuracy, although the difference in registration accuracy was small.

3.8 Performance Evaluation With D-FAUST Data

To test the registration performance of BCPD++ for more realistic shapes, we used the D-FAUST data containing 4D human body shapes constructed from human motions scanned at intervals of approximately 4 milliseconds [36]. Among them, we used the dataset whose subject ID was 50004 (female) and motion type was 'jiggle on toes.' The dataset also contains the ground-truth correspondences for each shape comprising 6,890 points.

By using Non-rigid ICP (NICP) [6], CPD [15], BCPD [18], and BCPD++, we registered the shape at frame t and the other shape at frame $t + \Delta t$ for $t = 1, \ldots, 50$. We set $\Delta t = 2$ and $\Delta t = 10$, indicating small and moderately large deformations, respectively. For CPD, BCPD, and BCPD++, we used the same parameters described in Section 3.6.4. Additionally, we accelerated CPD using FGT and the low-rank approximation of the Gram matrix. For NICP, we scaled point sets to be embedded into the cube $[-1, 1]^3$ and used $\gamma = 1$ by following the paper [6].

TABLE 3
Performance Evaluation With SHREC'19 Data

Source		Target		BCPD		BCPD++	
ID	#points	ID	#points	Acc.	Time	Acc.	Time
1	4,706	11	43,102	0.994	9.9	0.993	0.89
1	4,706	24	12,500	0.738	6.1	0.750	0.93
1	4,706	29	12,500	0.768	3.3	0.783	1.07
12	43,102	20	25,875	0.978	56.9	0.977	1.25
20	25,875	23	12,500	0.947	105.9	0.944	1.94
20	25,875	29	12,500	0.933	77.1	0.936	1.16
11	43,102	9	15,154	0.917	15.7	0.927	1.18
9	15,154	13	43,102	0.996	32.4	0.995	0.96
9	15,154	16	6,449	0.927	33.8	0.931	1.23
9	15,154	26	10,050	0.220	23.2	0.342	0.91

The columns 'ID', 'Acc.', and 'Time' indicate shape ID in the SHREC'19 dataset, registration accuracy, and computing time (s), respectively.

Fig. 7 shows the results of the comparison. From the figure, we see that the accuracies of BCPD++ and BCPD were roughly the same for both $\Delta t = 2$ and $\Delta t = 10$. We also see that registration accuracies of BCPD++ for $\Delta t = 10$ were higher than those for $\Delta t = 2$, suggesting that BCPD++ is suitable for moderately large deformation rather than small deformation.

4 Conclusion

In this paper, we proposed a fast non-rigid point set registration algorithm called BCPD++. The algorithm accelerates non-rigid registration via downsampling point sets and interpolating the deformation vectors of the points removed during downsampling. The rationale behind the interpolation is the motion coherence prior, i.e., a Gaussian process prior, which derives an efficient interpolation method interpreted as Gaussian process regression.

Through numerical experiments, we demonstrated that our registration algorithm was scalable to more than ten million points. Additionally, we compared the registration performance of BCPD++, BCPD, and other prevalent algorithms. The comparisons showed that the computing times of BCPD++ were noticeably shorter than those of BCPD for point sets containing more than a million points. They also showed that BCPD++ was relatively sensitive to artificial disturbances in small data compared to BCPD. These results suggest that BCPD++ is suitable for dense point sets rather than small, noisy data.

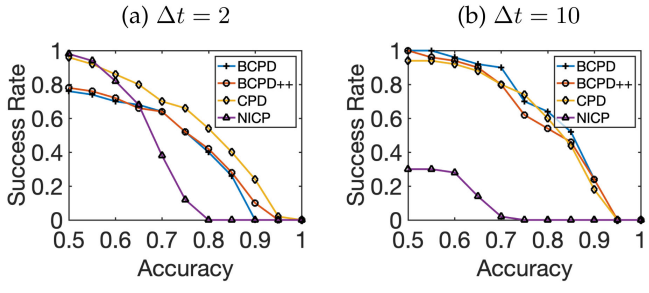


Fig. 7. Comparison of registration methods using D-FAUST data. The y-axis represents the rate of successful registrations for 50 shape pairs. Each shape pair comprises shapes at frame t and $t + \Delta t$. We defined a registration as being successful if the corresponding accuracy was less than a value specified by the x-axis.

ACKNOWLEDGMENTS

The author would like to thank the anonymous reviewers who provided constructive, detailed comments on the earlier version of the manuscript.

REFERENCES

[1] J. W. Tangelder and R. C. Veltkamp, "A survey of content based 3D shape retrieval methods," *Multimedia Tools Appl.*, vol. 39, no. 3, pp. 441–471, 2008.
[2] M. Berger *et al.*, "A survey of surface reconstruction from point clouds," *Comput. Graph. Forum*, vol. 36, no. 1, pp. 301–329, 2017.
[3] Y. Sahillioğlu, "Recent advances in shape correspondence," *Vis. Comput.*, vol. 36, no. 8, pp. 1705–1721, 2020.
[4] P. Besl and N. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, Feb. 1992.
[5] B. Allen, B. Curless, and Z. Popović, "The space of human body shapes: Reconstruction and parameterization from range scans," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 587–594, 2003.
[6] B. Amberg, S. Romdhani, and T. Vetter, "Optimal step nonrigid ICP algorithms for surface registration," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2007, pp. 1–8.
[7] H. Li, R. W. Sumner, and M. Pauly, "Global correspondence optimization for non-rigid registration of depth scans," *Comput. Graph. Forum*, vol. 27, no. 5, pp. 1421–1430, 2008.
[8] D. A. Hirshberg, M. Loper, E. Rachlin, and M. J. Black, "Coregistration: Simultaneous alignment and modeling of articulated 3D shape," *Lecture Notes Comput. Sci.*, vol. 7577, pp. 242–255, 2012.
[9] F. Bogo, J. Romero, M. Loper, and M. J. Black, "FAUST: Dataset and evaluation for 3D mesh registration," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 3794–3801.
[10] A. Rangarajan, H. Chui, and F. L. Bookstein, "The softassign procrustes matching algorithm," *Lecture Notes Comput. Sci.*, vol. 1230, pp. 29–42, 1997.
[11] S. Gold, C. P. Lu, A. Rangarajan, S. Pappu, and E. Mjolsness, "New algorithms for 2D and 3D point matching: Pose estimation and correspondence," *Neural Inf. Process. Syst.*, vol. 31, no. 8, pp. 1019–1031, 1998.
[12] S. Granger and X. Pennec, "Multi-scale EM-ICP: A fast and robust approach for surface registration," in *Proc. Eur. Conf. Comput. Vis.*, 2002, pp. 418–432.
[13] H. Chui and A. Rangarajan, "A new point matching algorithm for non-rigid registration," *Comput. Vis. Image Understanding*, vol. 89, no. 2–3, pp. 114–141, 2003.
[14] A. Myronenko, X. Song, and M. Á. Carreira-Perpiñán, "Non-rigid point set registration: Coherent point drift," in *Proc. Int. Conf. Neural Inf. Process. Syst.*., 2006, pp. 1009–1016.
[15] A. Myronenko and X. Song, "Point set registration: Coherent point drift," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 12, pp. 2262–2275, Dec. 2010.
[16] B. Jian and B. C. Vemuri, "Robust point set registration using Gaussian mixture models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 1633–1645, Aug. 2011.
[17] M. Saval-Calvo, J. Azorin-Lopez, A. Fuster-Guillo, V. Villena-Martinez, and R. B. Fisher, "3D non-rigid registration using color: Color coherent point drift," *Comput. Vis. Image Understanding*, vol. 169, pp. 119–135, 2018.
[18] O. Hirose, "A Bayesian formulation of coherent point drift," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Feb. 06, 2020, doi: 10.1109/TPAMI.2020.2971687
[19] L. Greengard and J. Strain, "The fast Gauss transform," *SIAM J. Sci. Statist. Comput.*, vol. 12, no. 1, pp. 79–94, 1991.
[20] V. Golyanik, B. Taetz, G. Reis, and D. Stricker, "Extended coherent point drift algorithm with correspondence priors and optimal subsampling," in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, 2016, pp. 1–9.
[21] V. Golyanik, S. A. Ali, and D. Stricker, "Gravitational approach for point set registration," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 5802–5810.
[22] V. Golyanik and C. Theobalt, "Optimising for scale in globally multiply-linked gravitational point set registration leads to singularities," in *Proc. Int. Conf. 3D Vis.*, 2019, pp. 164–172.
[23] J. Barnes and P. Hut, "A hierarchical O(N log N) force-calculation algorithm," *Nature*, vol. 324, no. 6096, pp. 446–449, 1986.
[24] G. Wang, Q. Zhou, and Y. Chen, "Robust non-rigid point set registration using spatially constrained Gaussian fields," *IEEE Trans. Image Process.*, vol. 26, no. 4, pp. 1759–1769, Apr. 2017.
[25] G. Wang, Y. Chen, and X. Zheng, "Gaussian field consensus: A robust nonparametric matching method for outlier rejection," *Pattern Recognit.*, vol. 74, pp. 305–316, 2018.
[26] G. Wang and Y. Chen, "SCM: Spatially coherent matching with Gaussian field learning for nonrigid point set registration," *IEEE Trans. Neural Netw. and Learning Syst.*, early access, Apr. 07, 2020, doi: 10.1109/TNNLS.2020.2978031
[27] J. Ma, J. Wu, J. Zhao, J. Jiang, H. Zhou, and Q. Z. Sheng, "Nonrigid point set registration with robust transformation learning under manifold regularization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 12, pp. 3584–3597, Dec. 2019.
[28] J. Dupej, V. Krajíček, and J. Pelikán, "Low-rank matrix approximations for coherent point drift," *Pattern Recognit. Lett.*, vol. 52, pp. 53–58, 2014.
[29] C. K. I. Williams and M. W. Seeger, "Using the Nyström method to speed up kernel machines," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2001, vol. 13, pp. 682–688.
[30] C. Yang, R. Duraiswami, N. A. Gumerov, and L. Davis, "Improved fast Gauss transform and efficient kernel density estimation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2003, pp. 664–671.
[31] J. Ma, J. Zhao, and A. L. Yuille, "Non-rigid point set registration by preserving global and local structures," *IEEE Trans. Image Process.*, vol. 25, no. 1, pp. 53–64, Jan. 2016.
[32] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 2006.
[33] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural network architectures," *Neural Comput.*, vol. 7, pp. 219–269, 1995.
[34] C. M. Bishop, *Pattern Recognition and Machine Learning*. Berlin, Germany: Springer, 2006.
[35] R. Marin, S. Melzi, E. Rodolà, and U. Castellani, "FARM: Functional automatic registration method for 3D human bodies," *Comput. Graph. Forum*, vol. 39, no. 1, pp. 160–173, 2019.
[36] F. Bogo, J. Romero, G. Pons-Moll, and M. J. Black, "Dynamic FAUST: Registering human bodies in motion," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5573–5582.