Learning Compact Geometric Features

Marc Khoury
UC Berkeley

Qian-Yi Zhou
Intel Labs

Vladlen Koltun
Intel Labs

Abstract

We present an approach to learning features that represent the local geometry around a point in an unstructured point cloud. Such features play a central role in geometric registration, which supports diverse applications in robotics and 3D vision. Current state-of-the-art local features for unstructured point clouds have been manually crafted and none combines the desirable properties of precision, compactness, and robustness. We show that features with these properties can be learned from data, by optimizing deep networks that map high-dimensional histograms into low-dimensional Euclidean spaces. The presented approach yields a family of features, parameterized by dimension, that are both more compact and more accurate than existing descriptors.

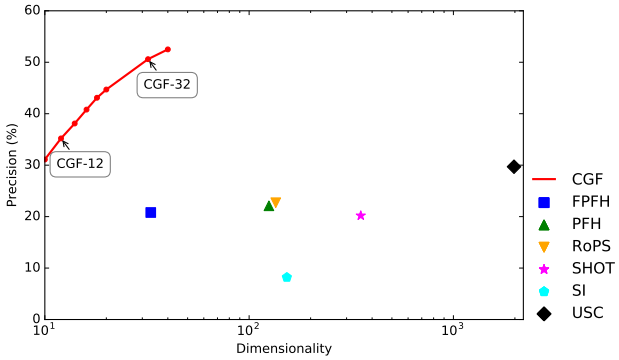


Figure 1. Our approach yields a family of Compact Geometric Features (CGF), parameterized by dimension. This figure illustrates the performance of CGF on the SceneNN test set. Our features are both more compact and more precise than the baselines. The horizontal axis (dimensionality) is on a logarithmic scale.

1. Introduction

Local geometric descriptors represent the local geometry around a point in a point cloud. They play a central role in geometric registration, which supports diverse applications in robotics and 3D vision [16] and underpins modern 3D reconstruction pipelines [42]. To enable accurate and efficient registration, the descriptor must possess a number of properties [12]. First, it should map the local geometry to a vector in a Euclidean space $\mathbb{R}^n$; such Euclidean representations support efficient geometric search structures and nearest-neighbor queries. Second, the descriptor should be discriminative: nearest neighbors in feature space should correspond to points with genuinely similar local neighborhoods. Third, the representation should be compact, with a small dimensionality n: this supports fast spatial search. Finally, the representation should be robust to artifacts that are commonly encountered in real data, such as noise and missing regions.

The design of local geometric descriptors has been the subject of intensive study for the past two decades. Many hand-crafted descriptors have been designed and evaluated [19, 11, 25, 27]. Nevertheless, no existing descriptor jointly satisfies the desiderata of high discriminative ability, compactness, and robustness [12]. Part of the challenge is the difficulty of optimizing the parameters of a high-dimensional feature representation by hand.

In this paper, we present an approach to learning local geometric features from data. Our descriptor applies directly to unstructured point clouds and does not require a clean and consistent surface parameterization [5], a volumetric representation [41], or the synthesis of auxiliary depth images [35]. Our features support nearest-neighbor queries in a Euclidean space, which allows establishing dense correspondences across point sets in near-linear time, in contrast to the quadratic complexity required by pairwise matching networks. We thus obtain the first learned geometric feature that can serve as a drop-in replacement for state-of-the-art hand-crafted features in existing pipelines [16, 42].

We show that the presented approach yields descriptors that are both more discriminative and more compact than state-of-the-art hand-crafted features. An illustration is provided in Figure 1. Experiments demonstrate that our Compact Geometric Features (CGF) yield more accurate matches at lower query times. When CGF is used on the standard Redwood benchmark for geometric registration, with no training or fine-tuning on that dataset, it yields the highest recall reported on the benchmark to date.

2. Background

The development of geometric descriptors for rigid alignment of unstructured point clouds dates back to the 90s. Classic descriptors include Spin Images [19] and 3D Shape Context [11]. More recent work introduced Point Feature Histograms (PFH) [26], Fast Point Feature Histograms (FPFH) [25], Signature of Histogram Orientations (SHOT) [27], and Unique Shape Contexts (USC) [33]. A comprehensive evaluation of existing local geometric descriptors is reported by Guo et al. [12].

Significant work has also been conducted on descriptors for nonrigid registration of deformable surfaces [1, 32]. These descriptors tend to make stronger assumptions, such as the existence of a reasonably clean meshed surface, and are designed to be invariant to isometric deformations. In contrast, rigid registration requires sensitivity to isometric deformations – the opposite of invariance. And applications in robotics require handling noisy unstructured point sets. Our work is devoted to rigid registration of unstructured point clouds.

A number of recent works applied learning to the problem of matching corresponding points based on local geometry. Wei et al. [35] describe an approach that matches points on human body scans and operates on ensembles of depth images. Boscani et al. [5] extend convolutional networks to Riemannian manifolds and apply them to establish correspondences across compatible manifolds. The contemporaneous work of Zeng et al. [41] uses volumetric signed distance fields and develops learned descriptors that use such volumetric representations as input. Cosmo et al. [8] learn descriptors for isometry-invariant nonrigid matching. In contrast, our work is devoted to learning compact descriptors for local geometry in unstructured point clouds, which can be used as highly efficient drop-in replacements for prior such descriptors in existing rigid registration pipelines [16, 42].

Deep networks have been applied to matching image patches and learning local image descriptors [3, 14, 30, 38, 39, 40]. Our research is informed by this work and applies related techniques to a different domain: point cloud registration. In particular, we investigate the effect of output dimensionality on accuracy and show that extremely low-dimensional descriptors can effectively represent the local geometry in an unstructured point cloud, significantly accelerating correspondence search in point cloud registration.

Learning has also been applied to shape classification and retrieval. Researchers have considered volumetric [37, 22] and multi-view representations [31]. These works do not deal with local geometric features and do not address the challenge of obtaining a local feature that is both accurate and compact. The difference between learning local geometric features and shape classification/retrieval is analogous to the difference between learning local image features [30] and image classification/retrieval [15, 2].

3. Overview

Parameterization. We parameterize the input to our model using spherical histograms centered at each point. These spherical histograms capture the local geometry in a neighborhood around each point. To incorporate rotational invariance, each spherical histogram is oriented to the normal and tangent spaces at each point. The interior of these spheres is subdivided along the radial, elevation, and azimuth directions. All neighboring points in the sphere are accumulated into the bins of the subdivision. The input parameterization is described in Section 4.

Feature embedding. We train a deep network to map from the high-dimensional space of spherical histograms to a very low-dimensional Euclidean space. The network learns an embedding into a low-dimensional feature space that maps similar geometric neighborhoods to nearby points. The model is trained using the triplet embedding loss. This is described in Sections 5 and 6.

Correspondences. Given a mapping f from a point into our learned feature space, computing correspondences between two point clouds $\mathcal{P}_i$ and $\mathcal{P}_j$ reduces to performing nearest-neighbor queries. We compute the set of features $f(\mathcal{P}_i)$ and $f(\mathcal{P}_j)$, and construct a k-d tree $\mathcal{T}$ on the point set $f(\mathcal{P}_j)$. For each point in $f(\mathcal{P}_i)$, we compute its nearest neighbor in $f(\mathcal{P}_j)$ using $\mathcal{T}$. As demonstrated in Section 7, correspondences computed using our feature space are much more accurate than correspondences computed using prior geometric feature descriptors. The low dimensionality of our features enables nearest-neighbor queries that are much faster than the second most accurate feature descriptor on real-world data.

Applications. Our features can serve as drop-in replacements for existing descriptors. We demonstrate this by replacing widely used Fast Point Feature Histograms (FPFH) [25] in existing geometric registration pipelines. This yields higher registration accuracy with no other modifications. These experiments are reported in Section 7.

4. Input Parameterization

Our basic approach is to start with a very high-dimensional representation of the raw local geometry around a point and train a deep network to embed this initial representation into a compact Euclidean space. Forming the initial representation is not trivial. Unlike images, which are laid out on a regular grid with a clear parameterization, a point cloud constitutes a set of unorganized points in $\mathbb{R}^3$. Even the cardinality of the set of points within a given neighborhood is not fixed. One possibility is to discretize the input into a uniform voxel grid, but such representa-

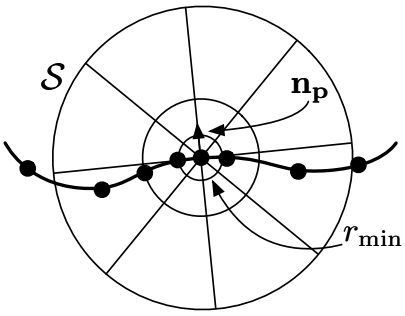


Figure 2. Our input parameterization, illustrated in two dimensions for clarity. The sphere $\mathcal{S}$ is centered at the point $\mathbf{p}$. In this two-dimensional illustration, the interior of $\mathcal{S}$ is subdivided into three bins along the radial direction and eight bins along the polar direction. This yields a 24-bin histogram into which the points in $\mathcal{S}$ are accumulated. The subdivision is aligned to the normal $\mathbf{n_p}$. In the real three-dimensional setting, the histogram has approximately two thousand bins.

tions are wasteful. For a 3-dimensional grid with C^3 cells, a smooth 2-dimensional surface will only intersect $O(C^2)$ cells: the rest are empty [18]. An alternative is to assume a clean parameterization of the underlying surface [5], but such a parameterization is not available in general.

Our initial representation is a histogram of the distribution of points in a local neighborhood, binned along a non-uniform radial grid [11]. Consider $\mathbf{p} \in \mathcal{P}$ and let $\mathcal{S}$ be a sphere centered at $\mathbf{p}$ with radius r. For rotational invariance, we estimate the normal $\mathbf{n_p}$ and a local reference frame based on this normal [27]. Consider the third vector $\mathbf{z_p}$ of the estimated local reference frame. If the dot product $\langle \mathbf{n_p}, \mathbf{z_p} \rangle < 0$, we flip the signs of all three vectors in the local reference frame.

The volume bounded by $\mathcal{S}$ can be subdivided into bins along the radial, elevation, and azimuth directions. These directions are defined in terms of the local reference frame. We subdivide the azimuth direction into A bins, each of extent $2\pi/A$. The elevation direction is subdivided into E bins, each of extent π/E. The radial direction, which has total span r, is logarithmically subdivided into R bins with the following thresholds:

$$r_i = \exp\left(\ln r_{\min} + \frac{i}{R} \ln\left(\frac{r}{r_{\min}}\right)\right). \qquad (1)$$

The first threshold r_0 evaluates to $r_{\min}$, which avoids excessive binning near the center. The thresholds grow exponentially, yielding an initial representation of multi-scale context. The result is a spherical histogram with $N = R \times E \times A$ bins. This is illustrated in Figure 2.

Let $\mathcal{N} \subset \mathcal{P}$ be the set of neighboring points that lie inside the sphere $\mathcal{S}$. The set $\mathcal{N}$ can be found efficiently using a k-d tree. For each point $\mathbf{q} \in \mathcal{N}$, we locate the histogram bin that contains $\mathbf{q}$ in constant time and increment the corresponding histogram value. After binning all the points in $\mathcal{N}$, we normalize the histogram by dividing each entry by $|\mathcal{N}|$. This yields a normalized N-dimensional feature vector that is used as input for a nonlinear embedding into a lower-dimensional Euclidean space.

5. Feature Embedding

We train a deep network $f : \mathbb{R}^N \to \mathbb{R}^n$ to map the space of input histograms into a lower-dimensional Euclidean space $\mathbb{R}^n$. This mapping serves two purposes. First, Euclidean distances between input histograms in $\mathbb{R}^N$ are to a significant extent arbitrary and do not appropriately reflect the similarity or dissimilarity of the geometric contexts represented by the histograms. Second, nearest-neighbor search in the lower-dimensional space $\mathbb{R}^n$ is much faster, which is important because nearest-neighbor search dominates the runtime of geometric registration pipelines [10, 42].

The mapping is trained to pull similar features together while pushing dissimilar features apart. To this end, we use the triplet loss [29, 36]. This objective has been used to optimize feature embeddings for a number of applications in computer vision [6, 28, 40].

Consider a set of triplets of input histograms $\mathcal{T} = \{(\mathbf{x}_i^a, \mathbf{x}_i^p, \mathbf{x}_i^n)\}_i$. Vector $\mathbf{x}_i^a$ is referred to as the anchor of triplet i, vector $\mathbf{x}_i^p$ is a positive example that is known to be similar to the anchor, and vector $\mathbf{x}_i^n$ is a negative example that is known to be dissimilar. Given such a set of triplets, we optimize the following objective:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{T}|} \sum_{i=1}^{|\mathcal{T}|} \big[\|f(\mathbf{x}_i^a;\boldsymbol{\theta}) - f(\mathbf{x}_i^p;\boldsymbol{\theta})\|^2 - \|f(\mathbf{x}_i^a;\boldsymbol{\theta}) - f(\mathbf{x}_i^n;\boldsymbol{\theta})\|^2 + 1\big]_+, \qquad (2)$$

where $\boldsymbol{\theta}$ are the parameters of the mapping f and $[\cdot]_+$ denotes $\max(\cdot, 0)$. Intuitively, f is optimized such that $\mathbf{x}_i^a$ is embedded closer to $\mathbf{x}_i^p$ than to $\mathbf{x}_i^n$, with a margin separating the distances.

We use a fully-connected network f with 5 hidden layers. Each hidden layer contains 512 nodes and is followed by an elementwise truncation $\max(\cdot, 0)$. We validated our model architecture with a controlled experiment reported in the supplement. At test time, computing the n-dimensional descriptor corresponding to an input histogram amounts to a sequence of matrix multiplications and elementwise operations.

6. Training

Consider a set of point clouds $\{\mathcal{P}_i\}_i$ that depict overlapping fragments of a scene. Let $\{\mathbf{T}_i\}_i$ be a set of rigid transformations that align the point clouds $\{\mathcal{P}_i\}_i$. Thus $\{\mathbf{T}_i\mathcal{P}_i\}_i$

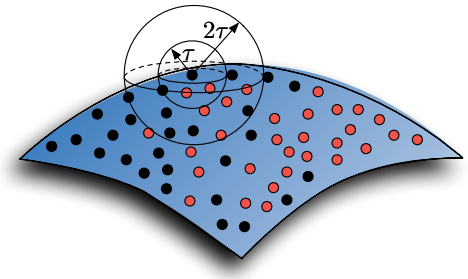


Figure 3. Two overlapping points clouds, shown here in red and black, are sampled from an underlying surface. We consider two concentric spheres of radius τ and 2τ around a black point $\mathbf{p}$. The red points in the innermost sphere, which form the set $\mathcal{N}_{\mathbf{p},\tau}$, are good correspondences for $\mathbf{p}$. The red points in the outermost sphere form the set $\mathcal{N}_{\mathbf{p},2\tau}$. We generate triplets for $\mathbf{p}$ by sampling $\mathbf{x}^p \in \mathcal{N}_{\mathbf{p},\tau}$ and $\mathbf{x}^n \in \mathcal{N}_{\mathbf{p},2\tau} \setminus \mathcal{N}_{\mathbf{p},\tau}$.

is a set of point clouds aligned to a common coordinate frame, in which distances between points $\mathbf{p} \in \mathbf{T}_i\mathcal{P}_i$ and $\mathbf{q} \in \mathbf{T}_j\mathcal{P}_j$ that depict nearby points in the latent scene are small. In this section we assume that the point clouds $\{\mathcal{P}_i\}_i$ and transformations $\{\mathbf{T}_i\}_i$ are given. Data in this form can be obtained from a variety of sources including scene reconstruction pipelines.

Consider a single point cloud $\mathcal{P}_i$ and a point $\mathbf{p} \in \mathcal{P}_i$. Let $\mathrm{nn}(\mathbf{p}, \mathcal{P}_i)$ denote the nearest neighbor of $\mathbf{p}$ in $\mathcal{P}_i \setminus \mathbf{p}$. Let ε_i be the median of the set of distances $\{\|\mathbf{p} - \mathrm{nn}(\mathbf{p}, \mathcal{P}_i)\| : \mathbf{p} \in \mathcal{P}_i\}$ and define $\varepsilon = \max_i \varepsilon_i$.

Now consider a pair of point clouds $(\mathcal{P}_i, \mathcal{P}_j)$. For each point $\mathbf{p} \in \mathbf{T}_i\mathcal{P}_i$ we can compute the nearest neighbor $\mathrm{nn}(\mathbf{p}, \mathbf{T}_j\mathcal{P}_j)$ of $\mathbf{p}$ in $\mathbf{T}_j\mathcal{P}_j$. Consider the fraction of such pairs that are within distance ε. Specifically, define

$$\alpha_{i,j} = \frac{|\{\mathbf{p} \in \mathbf{T}_i\mathcal{P}_i : \|\mathbf{p} - \mathrm{nn}(\mathbf{p}, \mathbf{T}_j\mathcal{P}_j)\| \leq \varepsilon\}|}{|\mathcal{P}_i|} \quad (3)$$

and similarly for $\alpha_{j,i}$. We say that $\mathcal{P}_i$ and $\mathcal{P}_j$ overlap if $\min(\alpha_{i,j}, \alpha_{j,i}) \geq 0.3$. This implies that the underlying surfaces from which $\mathcal{P}_i$ and P_j were sampled overlap by at least 30%.

Consider the set $\mathcal{O}$ of overlapping pairs of point clouds from $\{\mathcal{P}_i\}_i$. For each pair $(\mathcal{P}_i, \mathcal{P}_j) \in \mathcal{O}$ we examine each point $\mathbf{p} \in \mathcal{P}_i$. We compute the set of neighbors $\mathcal{N}^j_{\mathbf{p},\tau}$ in $\mathcal{P}_j$ that are at distance at most τ from $\mathbf{p}$. When τ is sufficiently small, the points in $\mathcal{N}^j_{\mathbf{p},\tau}$ are good correspondences for $\mathbf{p}$ in $\mathcal{P}_j$. Similarly consider $\mathcal{N}^j_{\mathbf{p},2\tau}$, the set of points in $\mathcal{P}_j$ that are at distance at most 2τ from $\mathbf{p}$. The set $\mathcal{N}^j_{\mathbf{p},2\tau} \setminus \mathcal{N}^j_{\mathbf{p},\tau}$ contains difficult negative examples for $\mathbf{p}$. These points have local geometries that are in general more similar to that of $\mathbf{p}$ than a randomly chosen point, but are not as close as those in $\mathcal{N}^j_{\mathbf{p},\tau}$. This is illustrated in Figure 3.

We generate training triplets $(\mathbf{x}^a, \mathbf{x}^p, \mathbf{x}^n)$ by sampling a pair of point clouds $(\mathcal{P}_i, \mathcal{P}_j) \in \mathcal{O}$, sampling a point $\mathbf{x}^a \in \mathcal{P}_i$ from the overlap region of $\mathcal{P}_i$ and $\mathcal{P}_j$, sampling $\mathbf{x}^p$ from the set $\mathcal{N}^j_{\mathbf{x}^a,\tau}$, and sampling $\mathbf{x}^n$ from the set $\mathcal{N}^j_{\mathbf{x}^a,2\tau} \setminus \mathcal{N}^j_{\mathbf{x}^a,\tau}$. This procedure is used to generate a large number of training triplets. The triplets are then permuted randomly and partitioned into minibatches.

We use minibatches of size 512 and train the mapping f using Adam [20]. The initial weights of the hidden nodes are drawn from a normal distribution with mean 0 and standard deviation 0.1. The learning rate is set to 10^{-4}. The parameters for the exponential decay of the first and second moment estimates are set to $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The network is trained for three epochs.

7. Experiments

7.1. Setup

Input parameterization. For the input parameterization, we use $R = 17$ subdivisions in the radial direction, $E = 11$ in the elevation direction, and $A = 12$ in the azimuth direction. The dimensionality of the input histogram is thus $N = 2{,}244$. We validate these choices via controlled experiments that are reported in the supplement.

Our approach yields a family of features parameterized by dimension. The primary setting of our feature space dimensionality is $n = 32$; the corresponding feature is referred to as CGF-32. The experimental results will demonstrate that this low dimensionality significantly outperforms prior, much larger descriptors.

On laser scan data, the radius r of the sphere $\mathcal{S}$ is set to 17% of the diameter of each model and $r_{\min}$ is set to 1.5% of the diameter. The value of the search radius is validated via controlled experiments reported in the supplement. The local reference frame at each point is computed using a search radius of 2% of the diameter.

On data from SceneNN [17], which has absolute metric scale, the radius r is set to 1.2 meters, which is approximately 17% of the diameter of the grid of each fragment, and $r_{\min}$ is set to 0.1 meters. The local reference frame is computed using a search radius of 0.25 meters.

Laser scan data. For experiments with laser scan data, we use a number of public-domain 3D models that are commonly used for this purpose. We use three models from the AIM@SHAPE repository (Bimba, Dancing Children, and Chinese Dragon), four models from the Stanford 3D Scanning Repository (Armadillo, Buddha, Bunny, and Stanford Dragon), and the Berkeley Angel [21]. Four of these models – Angel, Bimba, Bunny, and Chinese Dragon – were used as the training set, the Dancing Children model was used for validation, and the remaining three models – Armadillo, Buddha, and Stanford Dragon – were used as the test set.

For each model in the training and validation sets – Angel, Bimba, Bunny, Chinese Dragon, and Dancing Children – we synthesize depth images from 14 views uniformly distributed along the surface of an enclosing sphere. For each depth image we construct a point cloud that lies on the model. We compute the set of pairs of point clouds $\mathcal{O}$ that overlap in world space by at least 30%. Since some of these models do not have absolute scale, we set parameters and measure precision in relation to the diameter of the model. Synthesizing depth images allows us to automatically generate as much training data as we need and provides a controlled training environment in which we can validate our design choices. We found that descriptors trained on such synthetically scanned models successfully generalize to raw laser scans.

For testing we use the original raw laser scans of the models in our test set – Armadillo, Buddha, and Stanford Dragon. All three models were scanned with a Cyberware 3030 MS scanner. Armadillo has 114 scans, Buddha has 58 scans, and the Stanford Dragon has 71 scans. Using the provided alignments we compute a set of pairs of scans $\mathcal{O}$ that overlap in world space by at least 30%. These models demonstrate the ability of CGF to generalize to new domains, handle symmetric objects, and cope with noise encountered in laser scanned models.

SceneNN data. For experiments on real indoor scenes, we use SceneNN [17], a comprehensive recent dataset of indoor scenes scanned with consumer depth cameras. Starting from the raw SceneNN scans, we create fragments and register them using the pipeline of Choi et al. [7]. Each fragment is fused from 100 consecutive frames.

50% of the scenes are used for training, 25% for validation, and 25% as the test set, split randomly. We will publish our train/val/test split so that others can replicate our experiments. Let $\mathcal{O}$ be the set of pairs of overlapping fragments in the training scenes. For maximally precise alignment during training, we refined the registration of each pair $(\mathcal{P}_i, \mathcal{P}_j) \in \mathcal{O}$ using ICP [4]. We use the implementation of ICP provided in the Point Cloud Library [16].

Training. For each point cloud in the training set (synthetic depth image in the case of laser scan data, scene fragment in the case of SceneNN), we sample 40 triplets per point. Of these 40 triplets, 15 are constructed by sampling negatives from $\mathcal{N}_{\mathbf{p},2\tau} \setminus \mathcal{N}_{\mathbf{p},\tau}$, as described in Section 6. The remaining 25 are constructed by sampling negatives from the entire model. The threshold τ is set to 1% of the model's diameter in the case of laser scans and 7.5 cm in the case of SceneNN.

Baselines. We compare CGF to six well-known local descriptors: Point Feature Histograms (PFH) [26] (dimensionality 125), Fast Point Feature Histograms (FPFH) [25] (dimensionality 33), Rotational Projection Statistics (RoPS) [13] (dimensionality 135), Signature of Histogram Orientations (SHOT) [27] (dimensionality 352), Spin Images (SI) [19] (dimensionality 153), and Unique Shape Contexts (USC) [33] (dimensionality 1,980). For RoPS we use the implementation provided by the authors [13]. For all other baselines we use the implementations provided in the Point Cloud Library [16]. Each of these existing geometric feature descriptors has several parameters that need to be tuned to ensure good performance. We performed extensive hyperparameter sweeps to ensure that each baseline performed as well as possible in our experiments.

We have also applied Principal Components Analysis (PCA) to embed our input 2,244-dimensional histograms into $\mathbb{R}^n$, using our primary dimensionality $n = 32$. This evaluates the advantage of the presented nonlinear feature embedding over a linear embedding of the same input into the same space.

Additional baselines and controlled experiments are reported in the supplement.

Accuracy measure. Let $\{\mathcal{P}_i\}_i$, $\{\mathbf{T}_i\}_i$, and $\mathcal{O}$ be defined as in Section 6 and consider an overlapping pair $(\mathcal{P}_i, \mathcal{P}_j) \in \mathcal{O}$. Given a function f that maps points to geometric features, a set of correspondences between $\mathcal{P}_i$ and $\mathcal{P}_j$ can be found by first computing the sets of geometric features $f(\mathcal{P}_i)$ and $f(\mathcal{P}_j)$. Then we build a k-d tree $\mathcal{T}$ on the set $f(\mathcal{P}_j)$. For each point $\mathbf{p} \in \mathcal{P}_i$, we compute $\mathrm{nn}(f(\mathbf{p}), f(\mathcal{P}_j))$ by performing a nearest neighbor query in $\mathcal{T}$. Define

$$\mathcal{C}_f = \{(\mathbf{p}, \mathbf{q}) : \mathbf{p} \in \mathcal{P}_i, \mathbf{q} \in \mathcal{P}_j, f(\mathbf{q}) = \mathrm{nn}(f(\mathbf{p}), f(\mathcal{P}_j))\} \tag{4}$$

as the set of matches yielded by the feature f.

Since $\mathcal{P}_i$ only partially overlaps with $\mathcal{P}_j$, we first discard all correspondences $(\mathbf{p}, \mathbf{q})$ such that

$$\|\mathbf{T}_i\mathbf{p} - \mathrm{nn}(\mathbf{T}_i\mathbf{p}, \mathbf{T}_j\mathcal{P}_j)\| > \tau. \tag{5}$$

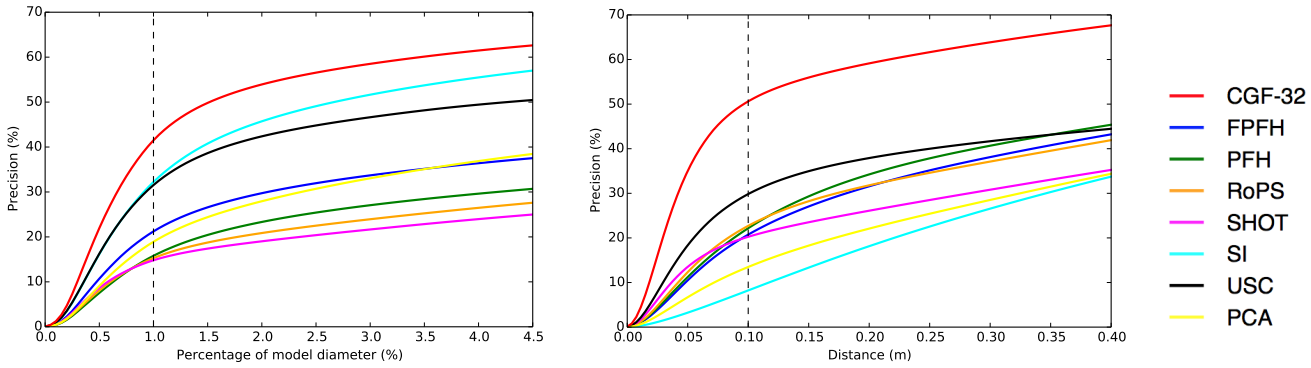
These points have no ground-truth correspondence in $\mathcal{P}_j$. Let $\mathcal{C}'_f$ denote the remaining set of correspondences.

For any distance threshold x, we can compute the fraction of matches that are within distance x of the ground truth:

$$\mathrm{precision}_f(x) = \frac{|\{\|\mathbf{T}_i\mathbf{p} - \mathbf{T}_j\mathbf{q}\| \leq x : (\mathbf{p}, \mathbf{q}) \in \mathcal{C}'_f\}|}{|\mathcal{C}'_f|}. \tag{6}$$

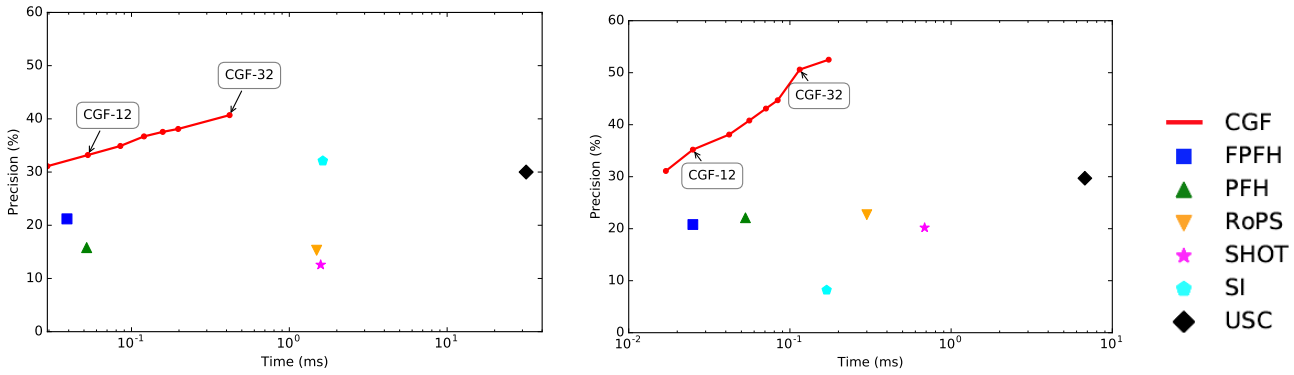
This will be our primary measure for evaluating the accuracy of different features f.

Timings. Average correspondence search times for different descriptors were benchmarked using a single thread on an Intel Xeon E7-8890 2.5 GHz CPU. We use FLANN [24] to perform nearest-neighbor queries.



(a) Precision on laser scan data test set (b) Precision on the SceneNN test set

Figure 4. (a) The precision of several geometric feature descriptors on laser scan data in the test set. For correspondences provided by CGF-32, 41.4% are precise to within 1% of the diameter. Prior feature descriptors are less accurate. (b) Precision of local geometric features on pairs of fragments from the SceneNN test set. CGF-32 yields the highest precision: 50.6% of the matches computed in the learned feature space are within 10 cm of the ground truth. USC (a 1,980-dimensional descriptor) comes in second at 29.8%.



(a) Query time and precision on laser scan data test set (b) Query time and precision on the SceneNN test set

Figure 5. (a) The query time and precision of several geometric feature descriptors on laser scan data in the test set. CGF-32 has an average query time of 0.42 ms, 3.9 times faster than the second most accurate feature (SI, 1.62 ms). (b) The query time and precision of local geometric features on pairs of fragments from the SceneNN test set. CGF-32 has an average query time of 0.1 ms, 67 times faster than the second most accurate feature (USC, 6.75 ms). The horizontal axis (time) is on a logarithmic scale.

7.2. Laser scan data

Precision of different features on the test set is shown in Figure 4(a). CGF-32 is much more accurate than existing descriptors. For example, 41.4% of the correspondences produced by CGF-32 lie within 1% of the model's diameter of the true match, whereas the most precise prior feature, SI, yields only 32.2% precision at this distance. CGF-32 improves over SI by 28.5% in relative precision while being 4.7 times more compact.

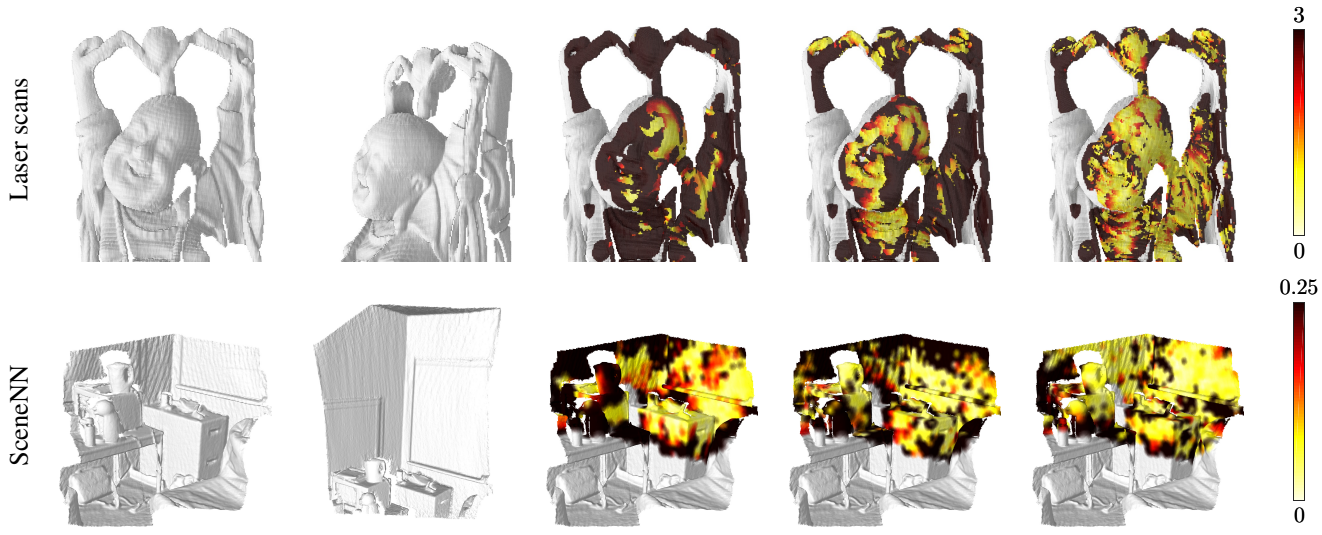
Timings. Query times for different features are presented in Figure 5. CGF-32 has an average query time of 0.42 ms, which is 3.9 times faster than the second most accurate feature (SI, 1.62 ms) and 75 times faster than USC (31.6 ms). CGF-12 has an average query time of 0.05 ms, slightly slower than the fastest feature (FPFH, 0.04 ms) while being more precise than all baselines (33.2% at 1% of the diameter).

Visualization. Figure 6 (top) shows error distributions of correspondences established in different feature spaces over two laser scans of the Buddha statue.

7.3. SceneNN data

Precision of different feature descriptors on real-world scene fragments from the SceneNN test set is shown in Figure 4(b). 50.6% of the matches established with CGF-32 are within 10 cm of the true match, much more than USC (29.8%), RoPS (22.7%), PFH (22.1%), FPFH (20.7%), SHOT (20.2%), and SI (8.2%). The baseline constructed by applying PCA to our 2,244-dimensional input parameterization yielded precision of 13.4%, far lower than the precision of the learned nonlinear embedding into the same space. Note that the second highest performing feature on laser scan data, SI, performed poorest on SceneNN.

Timings. Query times for different features are presented in



(a) source (b) target (c) FPFH (d) USC (e) CGF-32

Figure 6. **Top.** (a,b) Two laser scans of the Buddha statue. (c-e) Error magnitudes of matches established across the two scans in different feature spaces. CGF provides broad coverage of the surface with accurate matches. Units are in percentage of the model's diameter: black corresponds to error of 3% of the diameter or higher. **Bottom.** (a,b) Two fragments in the SceneNN test set. (c-e) Error magnitudes of correspondences established across these fragments in different feature spaces. Black corresponds to errors of 25 cm or higher. Correspondences established via CGF are more precise on average. Note the thin structure above the large hole in the middle of the fragment, along which all other feature spaces fail to establish good correspondences. Points shown in grey do not have a ground-truth correspondence on the other point cloud.

Figure 5. CGF-32 has an average query time of 0.1 ms, 67 times faster than the second most accurate feature (USC, 6.75 ms). CGF-12 has an average query time of 0.025 ms, matching the speed of FPFH. In addition to its speed, CGF-12 is more precise than all other features, with 31.5% of correspondences within 10 cm of the true match.

Visualization. Figure 6 (bottom) shows error distributions of correspondences established in different feature spaces over two fragments in the SceneNN test set.

7.4. Visualization

To get a qualitative sense of the learned representation, we can visualize the variation of the learned features over the surface of any model. Specifically, we can use PCA to project from the learned feature space into the 3-dimensional RGB color space. Given a point set, we can evaluate the learned feature for every point, use the learned linear mapping to obtain the corresponding color, and assign this color to the point. Figure 7 shows the result of this procedure for two synthesized views of the Dancing Children model. Note that the feature mapping appears stable, coherent, and discriminative. Corresponding points on the two views of the model tend to have similar color. Color varies more rapidly in regions of high-frequency geometric variation and is more stable in regions that are geometrically more uniform.

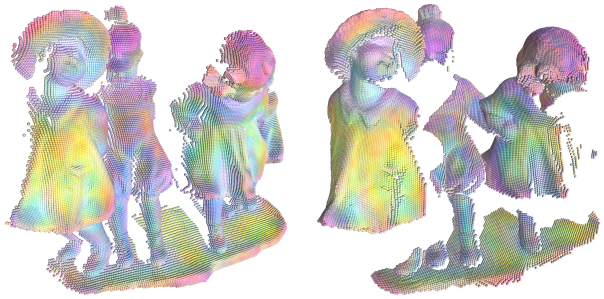


Figure 7. Visualization of local features over two views of the Dancing Children model. Features were projected from the learned feature space into the RGB color space.

7.5. Geometric registration

We now evaluate the utility of CGF in geometric registration. For this purpose, we use Fast Global Registration (FGR) [42], a state-of-the-art global registration algorithm that relies on feature matching. Since feature matching is the computational bottleneck of the algorithm, using a compact feature is important. The authors' implementation of FGR uses FPFH [42]. We use the published FGR pipeline as the baseline. To evaluate the utility of the learned CGF descriptor, we simply replace FPFH by CGF in the FGR pipeline.

To evaluate geometric registration accuracy, we follow the evaluation protocol of Choi et al. [7], which was also

	OpenCV [9]	Super 4PCS [23]	PCL [25, 16]	FGR [42]	CZK [7]	3DMatch [41]	FGR with CGF-32	CZK with CGF-32
Recall (%)	5.3	17.8	44.9	51.1	59.2	65.1	60.7	**72.0**
Precision (%)	1.6	10.4	14.0	23.2	19.6	**25.2**	9.4	14.6

Table 1. Evaluation on the Redwood benchmark. Plugging our learned descriptor into a pre-existing registration pipeline (CZK) yields the highest recall reported on the benchmark to date, with no training or fine-tuning on this dataset.

used by Zhou et al. [42] and Zeng et al. [41] .

Laser scans. We compare the accuracy of FGR when FPFH is used to the accuracy of FGR when CGF-32 is used. On the laser scan test set, FGR with FPFH correctly aligns 82.96% of the pairs while FGR with CGF-32 correctly aligns 92.27%. The average RMSE of FGR with FPFH is 13.8% of the diameter, while the average RMSE of FGR with CGF-32 is 9.2% of the diameter.

SceneNN. On the SceneNN test set, FGR with FPFH correctly aligns 88.54% of the fragment pairs, while FGR with CGF-32 correctly aligns 91.19%. The average RMSE of FGR with FPFH is 14.86 cm, while the average RMSE of FGR with CGF-32 is 11.83 cm.

If we focus on the correctly aligned pairs and evaluate the average RMSE only across those, FGR with FPFH yields an RMSE of 4.68 cm and FGR with CGF-32 yields an average RMSE of 4.07 cm. This in effect evaluates the tightness of the alignment produced by global registration. CGF-32 provides more precise correspondence pairs, which yield tighter alignment.

Cross-dataset generalization: Redwood benchmark. We now evaluate on the global registration benchmark of Choi et al. [7]. This benchmark has four datasets, each containing tens of scene fragments. Geometric registration is performed on every pair of fragments, with no initialization. For this experiment, we use the feature embedding that was trained on the SceneNN dataset. We did not retrain or fine-tune the descriptor in any way. This demonstrates the learned descriptor's ability to generalize to new datasets, as well as its ability to serve as a drop-in replacement in preexisting pipelines that depend upon discriminative geometric features.

The results are reported in Table 1. We report all the baselines from the evaluation conducted on this dataset by Zhou et al. [42]. We plug CGF-32 into FGR [42] and CZK [7], the existing implementations of which use the FPFH feature. This yields the corresponding "FGR with CGF-32" and "CZK with CGF-32" conditions.

CGF improves the recall of each method by more than 9 percentage points. With CGF-32, the CZK pipeline achieves a recall of 72%, by far the highest reported on the benchmark. Note that this is 6.9 percentage points higher than the contemporaneous results of Zeng et al. [41].

Choi et al. [7] defined two evaluation measures: recall and precision. Recall is the primary measure. The importance of recall is driven by two factors. First, the maximal level of precision that can be achieved by pairwise registration methods is low due to symmetric structures and other sources of geometric aliasing. Second, there are known ways to raise precision. Given a set of pairwise alignments, robust optimization of all fragments can prune false positives, retaining a given level of recall but increasing precision dramatically [7].

The effect of robust optimization is demonstrated in Table 2. Given pairwise alignments produced by CZK with CGF-32 features, robust optimization removes false positives and yields a set of pairwise alignments with 71.1% recall and 95.1% precision. The accuracy of this final result is limited not by the precision of the input set of pairwise alignments – as the results demonstrate, the overall pipeline is robust to low precision – but by the level of recall. Similar precision can be achieved by applying the framework of Choi et al. [7] to any of the prior works in Table 1.

	Before pruning		After pruning	
	FGR with CGF-32	CZK with CGF-32	FGR with CGF-32	CZK with CGF-32
Recall (%)	60.7	72.0	60.7	71.1
Precision (%)	9.4	14.6	86.8	95.1

Table 2. After post-processing with robust global optimization [7], CZK with CGF-32 achieves 71.1% recall and 95.1% precision on the Redwood benchmark.

8. Conclusion

We presented an approach to obtaining discriminative features for local geometry in unstructured point clouds. We have shown that state-of-the-art accuracy can be achieved with a low-dimensional feature space. The learned descriptor is both more precise and more compact than handcrafted features. Due to its Euclidean structure, the learned descriptor can be used as a drop-in replacement for existing features in robotics, 3D vision, and computer graphics applications. We expect future work to further improve precision, compactness, and robustness, possibly using new approaches to optimizing feature embeddings [34].

References

[1] M. Aubry, U. Schlickewei, and D. Cremers. The wave kernel signature: A quantum mechanical approach to shape analysis. In *ICCV Workshops*, 2011. 2

[2] A. Babenko, A. Slesarev, A. Chigorin, and V. S. Lempitsky. Neural codes for image retrieval. In *ECCV*, 2014. 2

[3] V. Balntas, E. Johns, L. Tang, and K. Mikolajczyk. PN-Net: Conjoined triple deep network for learning local image descriptors. *arXiv:1601.05030*, 2016. 2

[4] P. J. Besl and N. D. McKay. A method for registration of 3-D shapes. *PAMI*, 14(2), 1992. 5

[5] D. Boscaini, J. Masci, E. Rodolà, and M. M. Bronstein. Learning shape correspondence with anisotropic convolutional neural networks. In *NIPS*, 2016. 1, 2, 3

[6] G. Chechik, V. Sharma, U. Shalit, and S. Bengio. Large scale online learning of image similarity through ranking. *JMLR*, 11, 2010. 3

[7] S. Choi, Q.-Y. Zhou, and V. Koltun. Robust reconstruction of indoor scenes. In *CVPR*, 2015. 5, 7, 8

[8] L. Cosmo, E. Rodolà, J. Masci, A. Torsello, and M. M. Bronstein. Matching deformable objects in clutter. In *3DV*, 2016. 2

[9] B. Drost, M. Ulrich, N. Navab, and S. Ilic. Model globally, match locally: Efficient and robust 3D object recognition. In *CVPR*, 2010. 8

[10] J. Elseberg, S. Magnenat, R. Siegwart, and A. Nüchter. Comparison of nearest-neighbor-search strategies and implementations for efficient shape registration. *Journal of Software Engineering for Robotics*, 3(1), 2012. 3

[11] A. Frome, D. Huber, R. Kolluri, T. Bülow, and J. Malik. Recognizing objects in range data using regional point descriptors. In *ECCV*, 2004. 1, 2, 3

[12] Y. Guo, M. Bennamoun, F. A. Sohel, M. Lu, J. Wan, and N. M. Kwok. A comprehensive performance evaluation of 3D local feature descriptors. *IJCV*, 116(1), 2016. 1, 2

[13] Y. Guo, F. A. Sohel, M. Bennamoun, M. Lu, and J. Wan. Rotational projection statistics for 3D local surface description and object recognition. *IJCV*, 105(1), 2013. 5

[14] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg. MatchNet: Unifying feature and metric learning for patch-based matching. In *CVPR*, 2015. 2

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2

[16] D. Holz, A. E. Ichim, F. Tombari, R. B. Rusu, and S. Behnke. Registration with the point cloud library: A modular framework for aligning in 3-D. *IEEE Robotics and Automation Magazine*, 22(4), 2015. 1, 2, 5, 8

[17] B.-S. Hua, Q.-H. Pham, D. T. Nguyen, M.-K. Tran, L.-F. Yu, and S.-K. Yeung. SceneNN: A scene meshes dataset with annotations. In *3DV*, 2016. 4, 5

[18] T. Itoh and K. Koyamada. Automatic isosurface propagation using an extrema graph and sorted boundary cell lists. *TVCG*, 1(4), 1995. 3

[19] A. E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *PAMI*, 21(5), 1999. 1, 2, 5

[20] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 4

[21] R. K. Kolluri, J. R. Shewchuk, and J. F. O'Brien. Spectral surface reconstruction from noisy point clouds. In *Symposium on Geometry Processing*, 2004. 4

[22] D. Maturana and S. Scherer. VoxNet: A 3D convolutional neural network for real-time object recognition. In *IROS*, 2015. 2

[23] N. Mellado, D. Aiger, and N. J. Mitra. Super 4PCS: Fast global pointcloud registration via smart indexing. *Computer Graphics Forum*, 33(5), 2014. 8

[24] M. Muja and D. G. Lowe. Scalable nearest neighbor algorithms for high dimensional data. *PAMI*, 36, 2014. 5

[25] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (FPFH) for 3D registration. In *ICRA*, 2009. 1, 2, 5, 8

[26] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz. Aligning point cloud views using persistent feature histograms. In *IROS*, 2008. 2, 5

[27] S. Salti, F. Tombari, and L. di Stefano. SHOT: Unique signatures of histograms for surface and texture description. *Computer Vision and Image Understanding*, 125, 2014. 1, 2, 3, 5

[28] F. Schroff, D. Kalenichenko, and J. Philbin. FaceNet: A unified embedding for face recognition and clustering. In *CVPR*, 2015. 3

[29] M. Schultz and T. Joachims. Learning a distance metric from relative comparisons. In *NIPS*, 2003. 3

[30] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *ICCV*, 2015. 2

[31] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller. Multi-view convolutional neural networks for 3D shape recognition. In *ICCV*, 2015. 2

[32] J. Sun, M. Ovsjanikov, and L. J. Guibas. A concise and provably informative multi-scale signature based on heat diffusion. *Computer Graphics Forum*, 28(5), 2009. 2

[33] F. Tombari, S. Salti, and L. Di Stefano. Unique shape context for 3D data description. In *ACM Workshop on 3D Object Retrieval*, 2010. 2, 5

[34] E. Ustinova and V. Lempitsky. Learning deep embeddings with histogram loss. In *NIPS*, 2016. 8

[35] L. Wei, Q. Huang, D. Ceylan, E. Vouga, and H. Li. Dense human body correspondences using convolutional networks. In *CVPR*, 2016. 1, 2

[36] K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *JMLR*, 10, 2009. 3

[37] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *CVPR*, 2015. 2

[38] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua. LIFT: Learned invariant feature transform. In *ECCV*, 2016. 2

[39] S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. In *CVPR*, 2015. 2

[40] J. Žbontar and Y. LeCun. Stereo matching by training a convolutional neural network to compare image patches. *JMLR*, 17, 2016. 2, 3

[41] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser. 3DMatch: Learning the matching of local 3D geometry in range scans. In *CVPR*, 2017. 1, 2, 8

[42] Q.-Y. Zhou, J. Park, and V. Koltun. Fast global registration. In *ECCV*, 2016. 1, 2, 3, 7, 8