EUROGRAPHICS 2014 / Thomas Funkhouser and Shi-Min Hu
(Guest Editors)

Volume 33 (2014), Number 5

SUPER 4PCS
Fast Global Pointcloud Registration via Smart Indexing

Nicolas Mellado[1] Dror Aiger[2] Niloy J. Mitra[1]

[1]University College London [2]Google Inc.

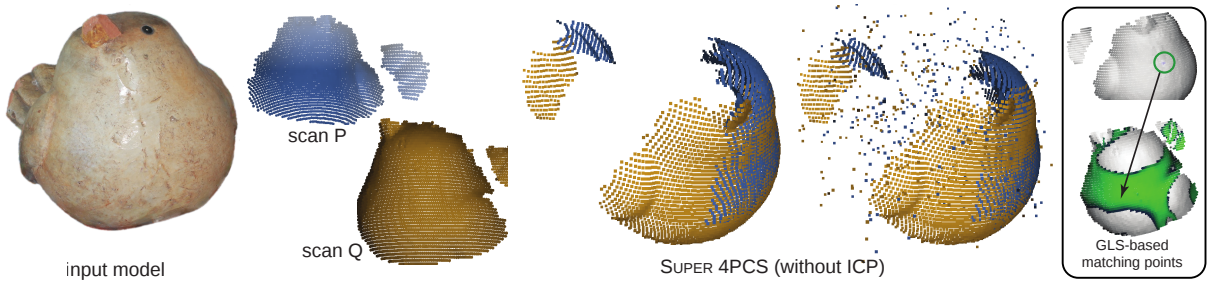

Figure 1: *We present* SUPER 4PCS*, an optimal linear time output-sensitive global alignment algorithm that registers a pair of raw pointclouds in arbitrary initial poses. This example is particularly challenging as no distinctive geometric features are available (see right inset) and color cues (not used) are confusing due to object shininess.* SUPER 4PCS *works even with low overlap (∼25%), and 20% outlier margin. Results are shown* without *ICP refinement. The proposed method has linear complexity over the state-of-the-art 4PCS, which has a quadratic time complexity.*

Abstract

Data acquisition in large-scale scenes regularly involves accumulating information across multiple scans. A common approach is to locally align scan pairs using Iterative Closest Point (ICP) algorithm (or its variants), but requires static scenes and small motion between scan pairs. This prevents accumulating data across multiple scan sessions and/or different acquisition modalities (e.g., stereo, depth scans). Alternatively, one can use a global registration algorithm allowing scans to be in arbitrary initial poses. The state-of-the-art global registration algorithm, 4PCS, however has a quadratic time complexity in the number of data points. This vastly limits its applicability to acquisition of large environments. We present SUPER 4PCS *for global pointcloud registration that is optimal, i.e., runs in linear time (in the number of data points) and is also output sensitive in the complexity of the alignment problem based on the (unknown) overlap across scan pairs. Technically, we map the algorithm as an 'instance problem' and solve it efficiently using a smart indexing data organization. The algorithm is simple, memory-efficient, and fast. We demonstrate that* SUPER 4PCS *results in significant speedup over alternative approaches and allows unstructured efficient acquisition of scenes at scales previously not possible. Complete source code and datasets are available for research use at http://geometry.cs.ucl.ac.uk/projects/2014/super4PCS/.*

1. Introduction

With rapid advances in sensor technologies, it is now easy to capture depth data at high-frame rates. This opens new opportunities in a range of topics including scene understanding, data collection, autonomous navigation, etc. High acquisition rate, however, comes at the cost of lower data quality. The scans are often sparse, noisy, and incomplete, and necessitate a data consolidation stage.

A common solution is to accumulate multiple scans across time (e.g., using Kinect Fusion). Such an approach performs *local* alignment using ICP (Iterative Closest Points) and assumes the scene to be static and the subsequent scans to be nearly aligned. Hence, for predictable accumulation results the acquisition device has to be moved slowly, or rigged with reliable GPS tracking to assist in initial positioning of the scans. Unfortunately, such a local registration approach inherently discourages data collection across multiple ses-

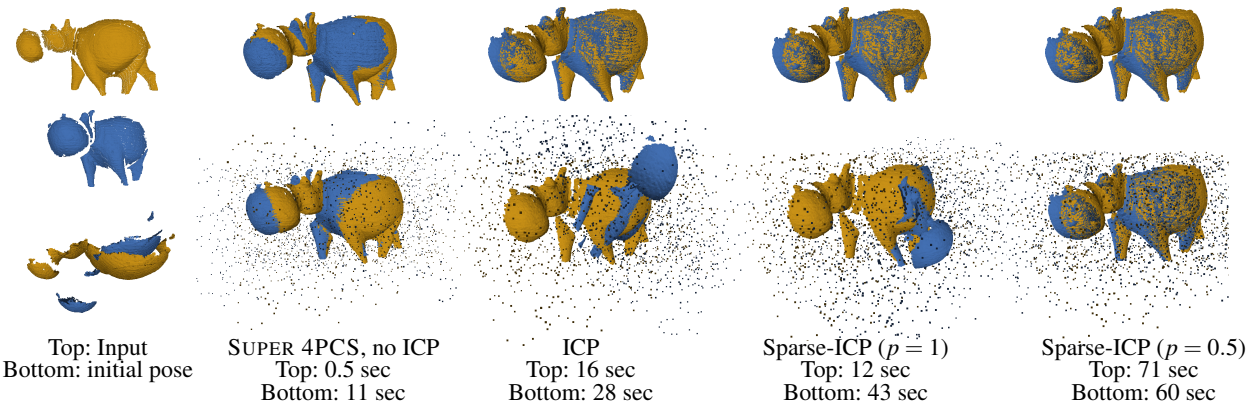

Figure 2: SUPER PCS, *being a global registration algorithm, is oblivious to initial model poses. In contrast, in case of low overlap and high outlier volumes, local alignment algorithms such as ICP or more advanced versions Sparse ICP [BTP13] can get caught in local minima, even when starting from nearly aligned initial poses.*

sions, or different acquisition modalities (e.g., stereo reconstruction, laser scan, SfM reconstruction).

Hence, in many scenarios *global* registration is desirable. Specifically, given a pair of scans, a source pointcloud P and a target pointcloud Q, in *arbitrary* initial poses, the problem is to find the best aligning rigid transformation T, such that $T(P) \approx Q$ under suitable distance measure. The state-of-the-art algorithm is 4PCS [AMCO08] that utilizes particular sets of congruent points for global registration. The algorithm, however, has a complexity of $O(n^2 + k)$ where n denotes the size of the pointclouds and k the set of candidate congruent 4-points. For large to very large point sets with low overlap the quadratic complexity quickly becomes a bottleneck (see Figure 1). This can be particularly limiting when multiple scans are to be stitched, e.g., reconstructing an indoor scene.

We propose SUPER 4PCS, a fast global registration for pointsets, which runs in optimal linear time and is output sensitive. The key insight is to remove the quadratic complexity in the original 4PCS algorithm by using an efficient yet practical data structure to solve the core *instance problem*, i.e., finding all point pairs that are within a distance range $(r - \varepsilon, r + \varepsilon)$. Specifically, SUPER 4PCS runs in $O(n + k_1 + k_2)$ time where k_1 is the number of pairs in Q at a given distance r and k_2 is the number of congruent sets. The proposed datastructure naturally extends to higher dimensions and allows a unified treatment of spatial and angular proximity queries. Hence, when auxiliary local surface information (e.g., normals, color) are available, then SUPER 4PCS can directly integrate the information. The proposed construction is adaptive and can be efficiently constructed at run-time with very low memory overheard.

We evaluate SUPER 4PCS on a range of real scans with varying amounts of noise, missing data, and overlap across scan pairs. We report significant speedups over the original 4PCS algorithm (3-10x in most cases). Finally, as an end application, we use SUPER 4PCS for large scale acquisition of indoor environments by simply waving a Kinect scanner at scenes without requiring smooth/slow motion paths (see Figure 15 and supplementary video). We believe this is the first demonstration of such large scale unstructured acquisition without specific scene assumptions.

2. Related Work

Scan registration is a fundamental task in geometry processing. Various surveys have explored aspects of the problem including surface descriptors [TM08], local registration [RL01], rigid and non-rigid registration [TCL*13]. Here, we summarize the papers most relevant to our method.

Local registration. When scan pairs start in close alignment, local registration algorithms are used to refine their alignment. The most popular local approach is Iterative Closest Points (ICP), both point-to-point [BM92] and point-to-plane [CM92] along with their many variants [RL01], and optimization using distance field formulation [MGPG04]. Recently, [SG07] analyze local minima of multiple ICP runs to find the best overall match between a known object and a range image, [BTP13] propose a sparse ICP formulation to robustly handle data with large amounts of outlier, or Kinect fusion [IKH*11] uses ICP with GPU optimization for real time scanning. These methods have convergence guarantees only when the scans pairs are roughly aligned to start with.

Global registration. When scan pairs start in arbitrary initial poses, then registration amounts to solving a global problem to find the best aligning rigid transform over the 6DOF space of all possible rigid transforms comprising of translations and rotations. Since aligning rigid transforms are uniquely determined by 3 pairs of (non-degenerate) corresponding points, one popular strategy is to invoke RANSAC to find such aligning triplets of point pairs [FB81, IR96,

CHC99]. The approach, however, regularly degrades to its worst case $O(n^3)$ complexity in the number n of data samples in presence of partial matching with low overlap.

Various alternatives have been proposed to encounter the cubic complexity: branch-and-bound using pairwise distance invariants [GMGP05]; hierarchical representation in the normal space [DMS12]; stochastic non-linear optimization to reduce distance between scan pairs [PB09, PB11]; super-symmetric tensors to represent the constraints between the tuples [CCM*13]; or evolutionary game theoretic matching [ART10, RABT13] as an alternative to RANSAC.

Earlier, the 4PCS algorithm [AMCO08], proposed a $O(n^2)$ algorithm using special four point basis instead of triplets as basis in RANSAC (see Section 3). Until this work, 4PCS remained the global registration algorithm of choice with the best time complexity.

Local descriptors. In presence of auxiliary information (e.g., unique feature points, color, texture, etc.) the Euclidean points can be appended with shape descriptors. A very large volume of such methods exists using local polynomial fits [TG11], local volume estimators in the form of integral features [PWHY09, ART10], multi-scale feature estimates [LG05, MGB*12], etc. Note that such descriptors are complementary to any registration algorithm and provide improvements only in objects with distinctive features or robust estimates, which are hard to compute under noise, missing data, color variations, reflections and highlights, etc.

3. The 4PCS Algorithm

The 4PCS algorithm is a global registration method for $3D$ point sets even with small overlap. The method makes no assumption about their starting scan poses. The approach is based on a novel technique to extract all sets of coplanar 4-points from a $3D$ point set that are approximately congruent, i.e., related by rigid transforms, to a given planar 4-points in $O(n^2+k)$ time, where n is the number of points and k is the number of reported 4-points sets. The 4PCS is widely used and has been also extended to take into account uniform scale variations [CDG*13].

The method relies on the following key fact: certain ratios defined on a planar congruent set remain invariant under affine transformations, and hence under rigid motion. Hence they propose a *generate and test* paradigm, i.e., pick a base from source P of minimum number of points. For each potential base from target Q, verify the corresponding aligning transform, and retain the best transform according to some similarity score. The efficiency of the algorithm depends on quickly extracting only a small set of candidates from Q that have to be verified. See [AMCO08], Sections 3 and 4, for details of the Algorithm.

The algorithm has two bottlenecks: (i) Finding all points at a given distance in a pointset in the congruent sets finding

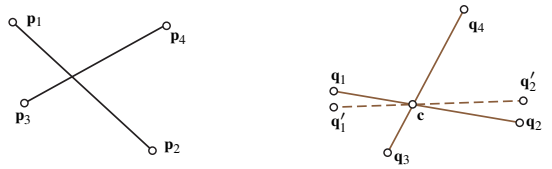

Figure 3: *The 4-point set $B := \{(\mathbf{p}_1,\mathbf{p}_2),(\mathbf{p}_3,\mathbf{p}_4)\}$ is congruent to $\{(\mathbf{q}_1,\mathbf{q}_2),(\mathbf{q}_3,\mathbf{q}_4)\}$, but also affine invariant to non-congruent set $\{(\mathbf{q}_1',\mathbf{q}_2'),(\mathbf{q}_3,\mathbf{q}_4)\}$. Such affine invariant yet non-congruent sets result in significant inefficiency and wasted validation steps in the original 4PCS algorithm.*

stage. For a given basis in P and pairs at distance r_1, r_2, the goal is to find all pairs in Q in distance r_1 and all pairs in Q at distance r_2. (ii) Removing the redundant 4-points candidates in the set of reported 4-points that arise due to considering affine invariants, even when the distances are constrained is a superset of all congruent sets (see Section 4.2).

The 4PCS takes roughly $O(n^2+k)$ time where k is the number of reported pairs. On one hand, when k is $\Omega(n^2)$, there is little motivation to improve the first term (at least asymptotically). Hence, the original 4PCS exhaustively enumerates the options in $O(n^2)$. So, if we cannot improve the n^2 term, there is no motivation to remove the redundant 4-points candidates. In this paper, we simultaneously address *both* problems, i.e., propose an efficient algorithm to solve the pairing problem for the first stage, and a smart indexing scheme to filter all the redundant pairs in the second stage. Both algorithms are optimal up to constant factor and take $O(n+k)$ time where k is the number of reported sets.

4. Smart Indexing

In this section, we make two key changes to the original 4PCS algorithm: first, lowering the quadratic complexity to optimal linear complexity; and second, lowering the number of candidate conjugate pairs generated.

4.1. Reporting incidences of points and spheres in $\mathbb{R}^3$

The 4PCS requires to report all point pairs in Q that are approximately r distance apart, for a given r. This is essentially a classical incidence problem between spheres and points in $3D$ (draw a sphere of radius r centered at each point and report all points intersect it), which has been long studied in computational geometry [PS04]. Specifically, we draw a sphere of radius of r centered at each point in Q and find the (approximate) incidences between all the n spheres and the n points in Q. The number of such incidences in the exact case (no approximation) is bounded [AKS05] and roughly given by the combinatorial bound $O(n^{1.5})$. Efficient algorithms to compute all incidences exist, which are based on cutting [Cha93], but are neither very suited for practical implementation, nor are output sensitive. Combinatorial bounds for the approximate case, assuming the minimum

distance between any two points in the set is bounded, is given in [GIMV04].

We are, however, interested in an output sensitive algorithm, linear in the number of (unknown) incidences, k. Further we relax the distances to be only approximately r, i.e., in $[r-\varepsilon, r+\varepsilon]$ for a given margin ε. As a key enabler we propose how to efficiently compute such approximate incidences. We propose two algorithms: The simple algorithm is essentially a rasterization approach (c.f., [GIMV04]). One puts all the points in a grid, and then simply rasterize (i.e., scan convert) a sphere of radius r in this grid and enumerate all the points in the cells (and their neighbors) encountered by the sphere. This leads to a runtime of $O(n \cdot (\Delta/\varepsilon)^2) + k)$ for a point set of diameter Δ and a grid of cell-size ε. Typically, for surface captured by a depth sensor n is $\Omega((\Delta/\varepsilon)^2)$ and thus the runtime is to $O((\Delta/\varepsilon)^4)$. For this kind of data sets (i.e., for sufficiently large n), we can do better. The second algorithm is described next:

Lemma 1 Let $\varepsilon > 0$ be a constant and let G_ε be a grid with cell-size ε. Let C be a set of distinct spheres of radius r, the center of each coincides with the center of a grid cell, and let U be a set of distinct grid cells, bounded by a square of side-length Δ. Then, all incidences (u,c), $u \in U$, $c \in C$ can be computed in time $O((\Delta/\varepsilon)^3 \log \Delta/\varepsilon)$.

Proof 1 The proof is similar to the one given in [AK10] for 2D. Let $\mathbf{B}$ be the bounding box of U. We divide $\mathbf{B}$ into 8 identical boxes. Recursively we compute the incidences for each subdivision with the subset of C that intersects it, using the cell-size as the minimum box size. For each box of this size, we report all the spheres that intersect the box. Since the number of (distinct) spheres that intersect a box of size Λ is $O((\Lambda/\varepsilon)^3)$ (the reason for this is quite involved and we give the intuition bellow), the runtime $T(\Lambda)$ for a box of diameter Λ satisfies, for $\Lambda = \varepsilon$, $T(\varepsilon) = O(1)$. The recursion for $T(\Lambda)$ becomes $T(\Lambda) = O((\Lambda/\varepsilon)^3) + 8T(\Lambda/2) = O((\Lambda/\varepsilon)^3 \log \Lambda/\varepsilon)$, which concludes the proof when $\Lambda = \Delta$.

NOTE: In order to show that the set of spheres intersecting a sub box is bounded, we need to use some non trivial analysis. The key is to note that among the set of all spheres, some become very close to each other within a small sub box. This suggests that we can approximate them by a small ($O(\Lambda/\varepsilon)^3$ for sub box of size Λ) set of other distinct spheres. Specifically, any circle of sufficiently large radius (relative to the box size) can be approximated by a sphere defined by three points on the edges of the box and the fixed radius (for smaller radius the bound is trivial by the fact that all circles are centered in grid cells).

We can therefore create a small set of spheres, defined by points of distance ε on the edges of the box such that every input sphere is closer than ε to one of them (we quantize the input spheres). In the algorithm, we first quantize every input sphere to the closest sphere in the small set, record all input spheres corresponding to each approximate sphere and give only the approximate set to the recursion applied to any sub box. We back trace the original spheres once the recursion reports all approximate incidences. During this process, we accumulate error, therefore we must take care of this by using actually some smaller ε' such that the overall error accumulation is $< \varepsilon$.

4.2. Eliminating redundant 4-points congruent sets

The 4PCS uses affine invariants to report similar 4-points. For a given 4-point in P, the algorithm finds all *affine* equivalent 4-points in Q where the distance between points in two pairs is (approximately) fixed. Although this results in a set much smaller than all the affine equivalent 4-points, the set can still be significantly larger compared to the exact congruent set (rigidly). This is inefficient as the obtained superset has to be filtered to get the exact congruent sets. By eliminating such redundant candidates as described next, SUPER 4PCS produces significant speeds up compared to 4PCS in most practical cases.

As shown in Figure 3, the pairs of points $(\mathbf{p}_1, \mathbf{p}_2)$ and $(\mathbf{p}_3, \mathbf{p}_4)$, all planar, define a 4-point set with two invariants (see Figure 5 in [AMCO08]). Exactly the same invariants correspond to the pairs $(\mathbf{q}_1', \mathbf{q}_2')$ and $(\mathbf{q}_3, \mathbf{q}_4)$ by arbitrarily rotating one line around the intersection point $\mathbf{c}$. This means that the set of 4-points extracted from Q is a superset of the set of *congruent* 4-points. We address this by extracting *only* the 4-planar points from Q that correspond to the *same angle* between the line segments in addition of being at the same distance and the same invariants.

The challenge is to extract this set in linear time with the number of reported set (i.e., without traveling the superset and filtering). Let $B := \{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4\}$ be a planar base in P picked randomly in the RANSAC loop. Let $r_1 = \|\mathbf{p}_1 - \mathbf{p}_2\|$ and $r_2 = \|\mathbf{p}_3 - \mathbf{p}_4\|$. Let f_1 and f_2 be the invariants computed from the pairs. At this stage, we assume that two sets of pairs, S_1 and S_2 have already been extracted from the set Q such that every pair in S_1 is of approximately at distance

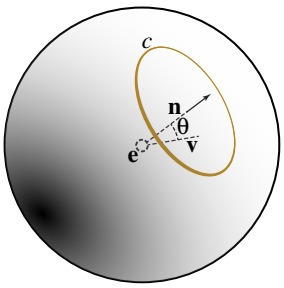

Figure 4: *Searching for all (unit) directions* $\mathbf{v}$ *such that* $\angle(\mathbf{n}, \mathbf{v}) \approx \theta$ *can be considered as an incidence problem, and solved efficiently. This allows up to generate a set of candidate 4-point sets, which are later verified, as a tight approximation to the set of congruent pairs.*

r_1 and every pair in S_2 is of approximately at distance r_2 (see Section 4.1).

Matching angle queries

We map the angle matching problem into indexing a set of normals, such that for any given query normal $\mathbf{n}$ and some angle θ, we can report quickly all normals that have angle θ with $\mathbf{n}$. Considering normals as a point on the unit sphere (Gauss sphere) it is exactly the same incidence problem as before, namely, to report all the points that lie (approximately) on a circle, say c, on the unit sphere (Figure 4). The circle is defined by the angle θ with respect to the query normal direction $\mathbf{n}$. We use the first algorithm to compute incidences because the radius of the circles is bounded and small. Again for approximation parameter ε, we rasterize a circle on an unit volume for a cell-size ε. The query normal $\mathbf{n}$, indicted by an arrow, defines the circle on the unit sphere that contains all points having angle θ with $\mathbf{n}$.

We now proceed with the overall 4-point congruent sets extraction. We impose a grid of cell-size ε, and in each grid we store normal indexing as described above. We describe the process for direction as per the one ordered pair $(\mathbf{p}_1, \mathbf{p}_2)$, while we repeat the same for the other direction. We loop over all pairs in S_1 and compute a new point $\mathbf{e}$, corresponding to the invariant f_1. We add $\mathbf{e}$ to the cell mapped to the grid and attach a vector in $\mathbb{R}^3$ representing the normalized direction from $\mathbf{p}_1 \rightarrow \mathbf{p}_2$. We insert the vector in the augmented normal index. At the end of this stage, all points $\mathbf{e}_i$ correspond to the invariant r_1 are stored in the grid cells and their normals are stored in the normal index at every cell. For a given base B, we have the angle θ between the crossing lines, using which we extract only the pairs in S_2 that agree with this angle, i.e., such that the angle between them and the corresponding pair in S_1 is approximately θ. At the end of this stage, we have a set of 4-points from Q that are *congruent* to the base B and now, it is the real set, not a superset. The efficiency of the indexing is output sensitive (it has some fixed runtime overhead to render the circles).

5. Implementation Detail

Based on Section 4, we now present two indexing schemes to achieve linear time output-sensitive global alignment: a pair extraction procedure (see Section 5.1) and congruent set extraction procedure (see Section 5.2). We explain the constructions for n-d spaces, but illustrate only in 2D.

5.1. Extracting approximately r-distant pairs

Overview. Starting from two pointclouds P and Q, and a planar 4-point set $B := \{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4\} \in P$, our goal is to extract congruent sets $\{\mathbf{q}_i, \mathbf{q}_j\} \in Q$. Let the candidate basis B define two distances r_1 and r_2. For any point $\mathbf{q}_i \in Q$, we want to find and index all the points at distance $r_1 \pm \varepsilon$ and $r_2 \pm \varepsilon$, in other words all the points close enough to the hyper-spheres

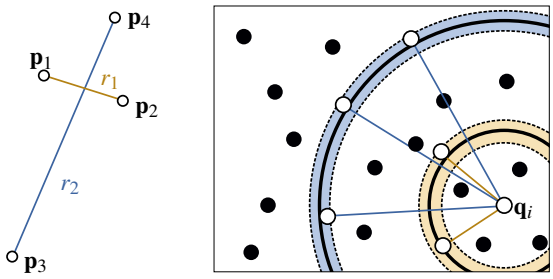

Figure 5: *Left: Example of a basis in P (for sake of simplicity we show only the distances defined by one pair). Right: For any point $\mathbf{q}_i \in Q$, pairs are generated by finding the other points close to the hyperspheres centered in $\mathbf{q}_i$ and with radius $r_1 \pm \varepsilon$ and $r_2 \pm \varepsilon$.*

S_i^r with radius $r = r_1$ or $r = r_2$ and centered at $\mathbf{q}_j$. Effectively, we perform simultaneous and adaptive rasterization of nD circles on to a nD grid, as illustrated in Figure 6 in 2D. The main idea is to subdivide recursively the space enclosing Q, and compute the intersection between a set of hyper-spheres and the subdivided volumes. Pairs can then be built between the points lying in the intersected volumes and the hyper-sphere centers (see Figure 5).

Hence, we use a regular splitting strategy, like those used for construction of quadtree and octrees [FB74]. A naive solution is to build such Q-tree and compute the intersections with the tree cells, represented as hyper-cubes, and the hyper-spheres. However, this is suboptimal as potentially large parts of the tree are unnecessarily subdivided even when they do not intersect hyper-spheres.

Instead, as shown in Proof 1, we recursively render quantified spheres (aligned on the ε grid with ε-quantified radius) resulting in optimal complexity to extract cells containing the potential pairs candidates. In practice, this requires to store and manipulate information (recursion tree, relations

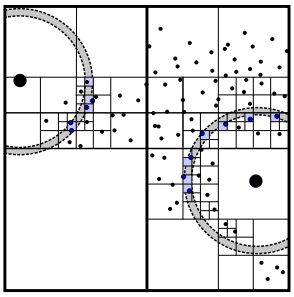

Figure 6: *Illustration of the simultaneous rasterization of two hyper-spheres of thickness ε and the extraction of points for pairing (in blue), using the Procedure ExtractPairs. The size of the final cells is equal to ε. Note that in 3D cells get split into 8 cells.*

between cells and hyper-spheres) that slow down the process and makes it inefficient. Next, we provide a tractable practical approach with low memory footprint without sacrificing running complexity.

Efficient hyper-sphere rasterization A pseudo-code of our implementation is presented in Procedure ExtractPairs. For simplicity, we normalize the data to fit in an unit bounding box. Any point at a distance $r \pm \varepsilon$ from $\mathbf{q}_i$ lying in that space is included in one of the cells of size 2ε intersecting the hyper-sphere S_i^r. Since we work in unit space and we split by two the cells size between each subdivision level, the number of recursion levels L can be deduced directly from ε at the beginning of the process. There are three key aspects of the procedure we describe next.

First, we use a sequential implementation to avoid unbounded recursive runs that could cause instabilities during the process. For that we need to store two lists of cells ($\mathbf{B}_1$ and $\mathbf{B}_2$), i.e., the candidates we want to try to intersect at the current level, and the candidates for the next level, e.g., the children of cells who actually intersected the hyper-spheres.

Second, we conservatively detect the intersections between the cells and the spheres, with a varying radius margin $\pm\varepsilon$. One option is to visit the cells that are neighbors with those intersecting the sphere, but this requires to build and explore the tree associated to the recursive grid (see Figure 7a). Another non-efficient option is to intersect explicitly the cells with $S_i^{r+\varepsilon}$ and $S_i^{r-\varepsilon}$ (see Figure 7b). Instead we grow the cells size by ε for the intersection, leading to the same amount of intersection test than solution (a), but without requiring to store the recursion tree (see Figure 7c).

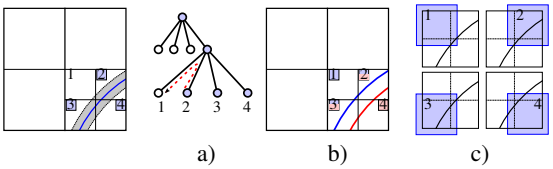

Figure 7: *Conservative intersection recursion. a) Non-intersecting cells are retrieved by visiting the neighboring cells thanks to the recursion tree. b) One can compute the intersection with the two hyper-spheres $S_i^{r+\varepsilon}$ and $S_i^{r-\varepsilon}$. c) Growing cells by ε for intersection test permits selecting appropriate cells without explicitly storing a recursion tree.*

Procedure ExtractPairs Given two point sets P and Q, extract the set of pairs of points with distance r within an approximation margin $\varepsilon > 0$.

```
Normalize P to fit in the Unit Box and transform Q to the same frame ;
B_1 ← { Unit Box} ;
L ← −log_2(ε) ;
for l ← 0 to L − 1 do
    B_2 ← ∅ ;
    forall the cells b_i ∈ B_1 do
        forall the points q_j ∈ Q do
            if Intersect(b_i, ε-alignedSphere(q_i, r)) [Arv90]
            then
                Subdivide(b_i) and re-index points contained
                in b_i for each child node.;
                B_2 ← { B_2 + childs(b_i) } ;
                break;
    B_1 ← B_2 ;
R ← ∅ ;
forall the points q_j ∈ Q do
    forall the cells b_i ∈ B_1 do
        if Intersect(b_i, ε-alignedSphere(q_i, r)) then
            Generate pairs R_i between Q_i and all points
            stored in b_i and R ← { R, R_j }
return R
```

Third, explicitly storing the information "which cell intersects which hyper-sphere" increases the memory footprint, and manipulating such information reduces the efficiency of the approach. In practice, we found that iterating over the cells and selecting them for the next level, if they intersect at least one hyper-sphere, is largely more efficient, even if the number of intersection test is globally more important.

From cells to pair extraction. Our objective is to use the cells to access efficiently input points and create pairs. This requires to associate each cell to all its enclosed points.

We achieve this efficiently and with a minimal memory footprint by using an index array shared between all cells and associated to the input point cloud Q. Each cell also stores a working range in this array. At the beginning, we have only one cell, working on the whole array. Each time we split, we partially reorder the indices in order to have a continuous range of indices for each of the child cells. Finally, for all the selected cells, we iterate over the associated index range, test if the points are valid, and create the validated pairs.

Performance. We tested our approach with increasing number of input points and compared the processing time to the reference solution in $O(n^2)$ that enumerate all the pairs and then filter them (see Figure 8a). The quadratic behavior of the naive filtering approach is clearly visible on the curves. Note that our result is guaranteed to be the same as the reference solution.

We also compare our method with or without quantifying the input hyper-spheres. This quantification is required to ensure to have a bounded complexity (see Proof 1) and reduce the number of hyperspheres, and thus the number of intersection tests. In practice, the observed gain is usually lower than the time spent to quantify the spheres, so we work directly with the original spheres.

We early abort recursions for cells containing a minimum

number of points. Such cells are stored and used at the end of the process to create the pairs like the other cells. Similarly, we skip over void cells and reject them directly after the splitting process. Empirically, our approach is faster than explicit quering even with small point sets (∼20-30).

5.2. Congruent set extraction

Overview. We again use a geometric approach to extract the congruent set from the previously computed set of points. This time it is exactly the procedure described in Section 4.2, the constants in the complexity analysis being practical and do not require further optimization.

Starting from two set of pairs P_{r_1} and P_{r_2} formed by a 4-points basis P and points $\mathbf{q}_i \in Q$, our goal is to extract the set of (planar) quadrilaterals that are congruent to the basis. A quadrilateral is congruent to the basis if it is composed of pairs with the right length, and if the angle θ between these pairs is similar to the angle formed by the two basis pair.

The idea is to represent a pair by a couple of vectors, i.e., the position and the orientation of the pair, considered in the following as a normalized vector (see Figure 4). The query can be mapped to another instancing problem with two main conditions: First, ensure that the retrieved points lie in the same cell as the query in a grid defined in the n-dimensional ambient space. We hash a static grid with constant size for linear time. Second, we rasterize circles formed by the intersection between the unit sphere and a right angular cone with an aperture of 2θ around direction $\mathbf{n}$.

Data representation and query. Our data structure stores the pairs indices, indexed with respect to the Euclidean coordinates and then by the normal coordinates (see Figure 9). We use a sparse representation in Euclidean space, i.e., cells are built only if we need to store an index or more. Each such cell contains a spherical map (c.f., [GHB*05]) where we store the pair index with respect to its orientation.

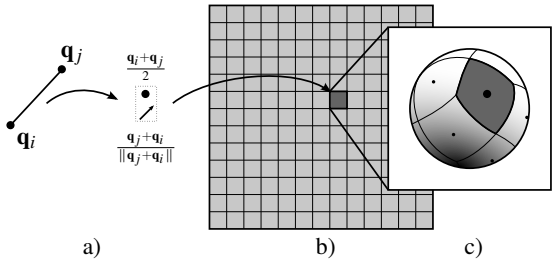

Figure 9: *Structure used to hash pairs first by their position in the ambient space, and then by their orientation. a) Starting from a pair, we compute the centroid and its direction, represented as a normalized vector. b) The position of the centroid is used to access cells in a regular grid. c) The orientation is then hashed and the index of the pair is stored in orientation space.*

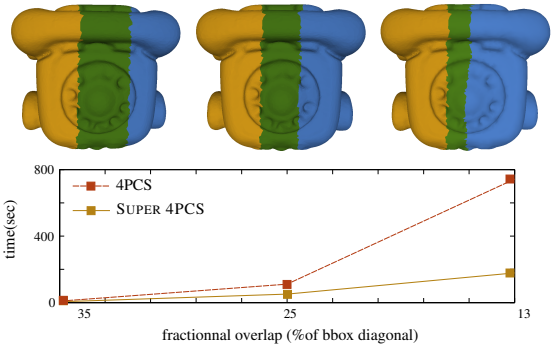

Figure 10: *Computation time comparison between* SUPER 4PCS *and 4PCS for decreasing overlap margin. Notice the linear versus quadratic complexity for the respective methods. For visualization, we indicate the overlap regions in green. The models start in arbitrary initial poses.*

One could use a hyper-circle rasterization to extract indices with respect to the normal criterion. In practice, however, the normal spaces contain only few elements, for simplicity we just sample the hyper-circles in angular space, and collect points in the cells where the samples are lying in the normal grid. Another option would be to work with spherical coordinates, and render the circle in that space.

Performances. In this step, we used an approximated query based only on hashing functions, and we may miss retrieving all the pairs. In practice we simply use an increasing factor to reduce the probability to miss configuration, and did not observe any significant impact on the quality of the matching. As shown in Figure 8b, the speed-up obtained is significant and outperform the filtering previously used.

6. Results

We implemented SUPER 4PCS in C++ (available on the project page). We used Eigen [GJ*10] to for filtering in arbitrary dimensions $n > 2$, [Arv90] to efficiently intersect hyper-spheres and hyper-cubes, and [GHB*05] to sample orientation space.

We tested our approach on point-clouds from several datasets [BTP13, CDG*13], acquired with different techniques (range scans, end-user depth cams, stereo reconstruction), and with a large variety of geometric properties: smooth natural shapes, man-made objects, buildings with flat surfaces, featureless surfaces. The registrations presented in this paper are between pairs with low overlap (mostly <30%) and in arbitrary initial poses. Normals are used only when available in the source data. Please refer to Table 1 for performance on the different datasets.

We first evaluate our approach in term of computation time improvement, then compare it to state-of-the-art techniques, and finally show a direct application of our approach for the large scale registration of unstructured data.

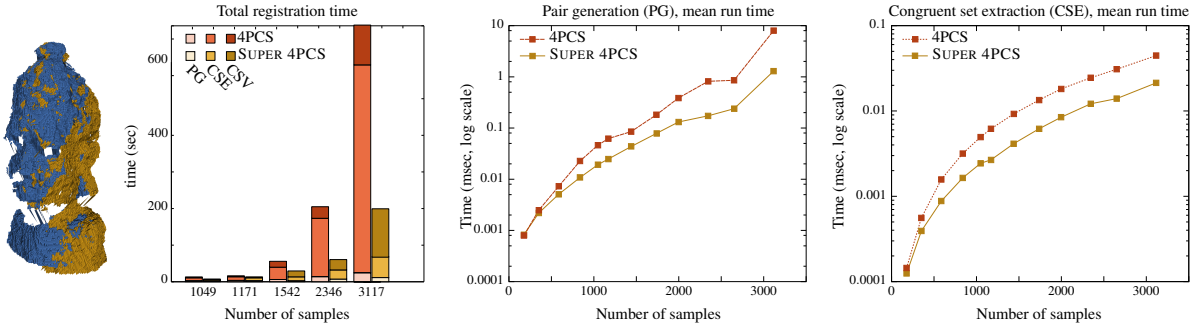

Figure 8: *Timings of* SUPER 4PCS *to 4PCS to match two point clouds with normals and 30% overlap. Left graph: total time (sec) needed for matching, with details on the time spend for Pair Generation (PG), Congruent Set Extraction (GSE), and Congruent Set Verification (CSV). Middle and right graphs: individual timing (msec) for the PG and GSE subroutines. Note that the timings are in* log*-scale.*

6.1. Evaluation

Since our approach compute the same transformations than 4PCS, we focus on timing improvements in various conditions. Figure 8 shows the impact of the number of samples by increasing the sampling rate of Q and reports the total time spent in each subroutine to extract the pairs, build the congruent set and analyze it, both with 4PCS and SUPER 4PCS. Note that the verification step remains the same in both approaches. The change from a quadratic to linear behavior is clearly visible and allows registering object pairs a bit faster with 3117 samples using SUPER 4PCS compared to 2346 samples with 4PCS on that example. Note that our subroutines are always faster than the naive approaches for this object, with a significant difference when increasing the number of samples (we use a log-scale to emphasis improvement at all sampling rates).
We show in Figures 10 and 12 practical examples where the overlap between P and Q is low and requires to increase the number of samples, making our approach much faster than 4PCS. Our approach remains robust in such low-overlap cases as shown in Figure 1 and 13. We also tested our approach on noisy pointclouds (see Figures 11 and 15).

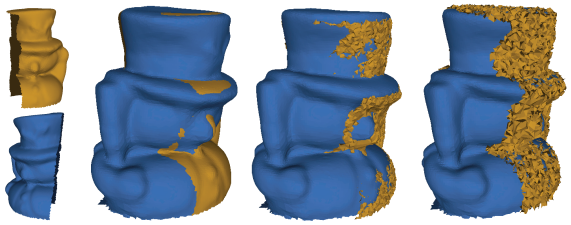

Figure 11: SUPER 4PCS *remains robust under effect of increasing noise even under small over without any noticeable impact on running time.*

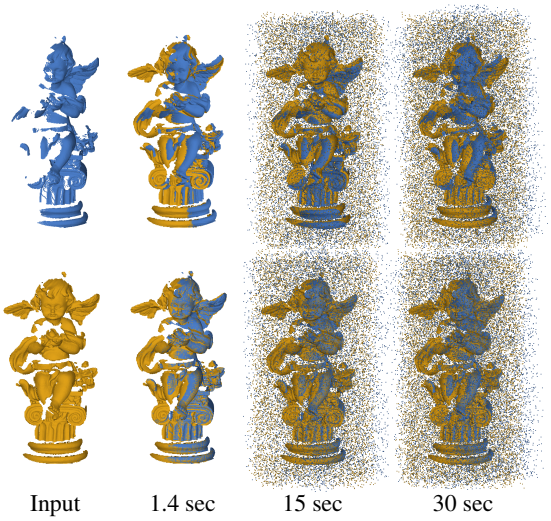

Figure 12: *Effect of increasing outliers on* SUPER 4PCS *global alignment performance. Note that outliers effectively amounts to lower* overlap *between the two input models. For each result set, top and bottom images, respectively show result before and after ICP.*

6.2. Comparison

We compared our approach with state-of-the-art Sparse-ICP [BTP13] on two range scans with overlap (>50%) and artificial outliers. We keep the orientation of the acquired scans to define the input pose, since the device motion between the two frames was not too big. Our approach performs well w/ and w/o outliers. In fact, it is faster to first run SUPER 4PCS and then ICP; instead of running it directly (even if it converges). Note SUPER 4PCS ignores initial pose information (see Figure 2).

SUPER 4PCS is very useful for featureless point-clouds (see Figure 1), where descriptors are not discriminative. In

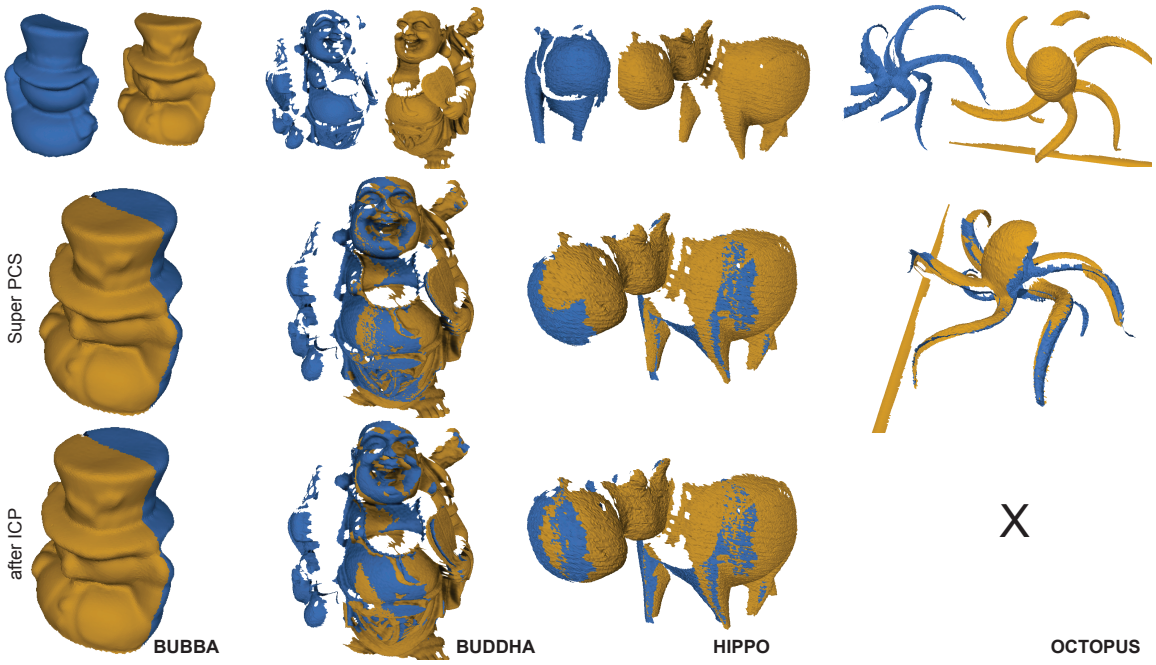

Figure 13: SUPER 4PCS *results on various models. Note that the bubba model has very little overlap, and for the octopus the overlap was too thin for ICP to work. In all these cases, overlap ranged from 5-25%.*

the right inset we show that given the slippable surface, feature extraction is ineffective, even using a robust multi-scale descriptor like Growing Least Squares [MGB*12]. SUPER 4PCS remains oblivious to such challenges.

6.3. Application

Aligning heterogeneous acquisition data. We aligned two point-clouds of the Cathedral of Pisa, one acquired using LIDAR and the other using Multi-View Stereo (MVS) reconstruction, and each of them formed by more the 2.5 millions of points. The matching required 15 seconds using the full MVS point-cloud and 1000 samples on the other one.

Table 1: *Time taken for aligning various input datasets as measured on a 3.00GHz Xeon E5-1607 with 8GB RAM. In all examples, the models start in arbitrary poses, preprocessing times if any are included, and the reported times are average over a multiple runs of the algorithm.*

model	#points (in 1000)	4PCS (in sec)	SUPER 4PCS (in sec)
Bird	2.5	48	26
Hippo	32	11	0.5
Phone	32	8.7	4.7
Bubba	10	5	2.5
Buddha	37	63	37
Octopus	50	131	60
Angel	35	2.47	1.4
Pisa	2500	22.6	13

Aligning heterogeneous acquisition data. SUPER 4PCS can also directly align multiple Kinect scans (color information ignored) without any volexiation or smoothing. Figure 15 shows results *without* any ICP refinement for 3 different indoor scenes (see supplementary video).

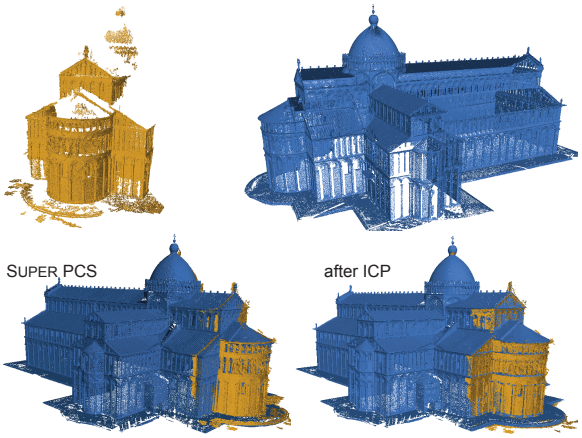

Figure 14: SUPER 4PCS, *being a global alignment algorithm, is well suited to align data from different acquisition sessions (here LIDAR and structure-from-motion outputs), where no initial alignment is available.*

Figure 15: SUPER 4PCS *based alignment of 5-6 scans for different scenes captured using Kinect.*

7. Conclusion

We presented SUPER 4PCS, an optimal linear time output sensitive algorithm for global registration, i.e., aligning scan pairs in arbitrary initial poses. The algorithm is particularly effective in case of low overlap across scans and/or presence of outliers. The algorithm, for the first time, allows us to consolidate data acquired by different devices or across multiple sessions. Finally, even when the starting scans are nearly aligned, prior methods (like ICP or sparse ICP) fail in case of low overlap but SUPER 4PCS continues to work. We tested the algorithm in various scenarios and compared with competing alternatives, both in terms of speed and accuracy.

Although SUPER 4PCS is optimal (for single core) with very practical runtime constants, we believe a parallel implementation is possible. This is particularly attractive for real-time acquisition in order to remove assumption of controlled acquisition paths, visual verification to fill in holes [RHHL02], or integration over voxel volumes (e.g., Microsoft Kinect) leading to loss of fine details. In the future, we plan to explore this direction along with realtime integration of color and texture.

Acknowledgements

We thank the reviewers for their comments and suggestions for improving the paper. We thank Duygu Ceylan for useful feedback and Aron Monszpart for discussion. We thank Sofien Bouaziz and Andrea Tagliasacchi for useful help on SparseICP comparison and for providing us the Hippo, Angel, Buddha and Octopus models, and Matteo Dellepiane for the Cathedral of Pisa dataset. This work was supported in part by the Marie Curie Career Integration Grant 303541, the ERC Starting Grant SmartGeometry (StG-2013-335373), and gifts from Adobe Research.

References

[AK10] AIGER D., KEDEM K.: Approximate input sensitive algorithms for point pattern matching. *Pattern Recogn. 43*, 1 (Jan. 2010), 153–159. 4

[AKS05] ARONOV B., KOLTUN V., SHARIR M.: Incidences between points and circles in three and higher dimensions. In *Discrete Comput. Geom.* (2005), pp. 185–2006. 3

[AMCO08] AIGER D., MITRA N. J., COHEN-OR D.: 4-points congruent sets for robust surface registration. *ACM TOG (SIGGRAPH) 27*, 3 (2008), 85, 1–10. 2, 3, 4

[ART10] ALBARELLI A., RODOLÀ E., TORSELLO A.: Loosely distinctive features for robust surface alignment. In *ECCV* (2010), pp. 519–532. 3

[Arv90] ARVO J.: A simple method for box-sphere intersection testing. In *Graphics Gems*, Glassner A. S., (Ed.). 1990. 6, 7

[BM92] BESL P. J., MCKAY N. D.: A method for registration of 3-d shapes. *IEEE PAMI 14*, 2 (Feb. 1992), 239–256. 2

[BTP13] BOUAZIZ S., TAGLIASACCHI A., PAULY M.: Sparse iterative closest point. *CGF (SGP) 32*, 5 (2013), 1–11. 2, 7, 8

[CCM*13] CHENG Z.-Q., CHEN Y., MARTIN R., LAI Y.-K., WANG A.: Supermatching: Feature matching using supersymmetric geometric constraints. *IEEE TVCG 19*, 11 (2013). 3

[CDG*13] CORSINI M., DELLEPIANE M., GANOVELLI F., GHERARDI R., FUSIELLO A., SCOPIGNO R.: Fully automatic registration of image sets on approximate geometry. *Int. J. Comput. Vision 102*, 1-3 (Mar. 2013), 91–111. 3, 7

[Cha93] CHAZELLE B.: Cutting hyperplanes for divide-and-conquer. *Discrete Comput. Geom. 9*, 2 (Apr. 1993), 145–158. 3

[CHC99] CHEN C.-S., HUNG Y.-P., CHENG J.-B.: RANSAC-based DARCES: A new approach to fast automatic registration of partially overlapping range images. *IEEE PAMI 21*, 11 (Nov. 1999), 1229–1234. 2

[CM92] CHEN Y., MEDIONI G.: Object modelling by registration of multiple range images. *Image Vision Comput. 10*, 3 (Apr. 1992), 145–155. 2

[DMS12] DIEZ Y., MARTA J., SALVI J.: Hierarchical normal space sampling to speed up point cloud coarse matching. *Pattern Recognition Letters 33*, 16 (2012), 2127 – 2133. 3

[FB74] FINKEL R., BENTLEY J.: Quad trees a data structure for retrieval on composite keys. *Acta Informatica 4*, 1 (1974), 1–9. 5

[FB81] FISCHLER M. A., BOLLES R. C.: Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM 24*, 6 (June 1981), 381–395. 2

[GHB*05] GÓRSKI K. M., HIVON E., BANDAY A. J., WANDELT B. D., HANSEN F. K., REINECKE M., BARTELMANN M.: Healpix: A framework for high-resolution discretization and fast analysis of data distributed on the sphere. *The Astrophysical Journal 622*, 2 (2005), 759. 7

[GIMV04] GAVRILOV M., INDYK P., MOTWANI R., VENKATASUBRAMANIAN S.: Combinatorial and experimental methods for approximate point pattern matching. *Algorithmica 38*, 1 (2004), 59–90. 4

[GJ*10] GUENNEBAUD G., JACOB B., ET AL.: Eigen v3. http://eigen.tuxfamily.org, 2010. 7

[GMGP05] GELFAND N., MITRA N. J., GUIBAS L. J., POTTMANN H.: Robust global registration. In *Proc. SGP* (2005), pp. 197–206. 3

[IKH*11] IZADI S., KIM D., HILLIGES O., MOLYNEAUX D., NEWCOMBE R., KOHLI P., SHOTTON J., HODGES S., FREEMAN D., DAVISON A., FITZGIBBON A.: Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *Proc. UIST* (2011), pp. 559–568. 2

[IR96] IRANI S., RAGHAVAN P.: Combinatorial and experimental results for randomized point matching algorithms. In *Proc. Symposium on Computational Geometry* (1996), pp. 68–77. 2

[LG05] LI X., GUSKOV I.: Multi-scale features for approximate alignment of point-based surfaces. In *Proc. SGP* (2005). 3

[MGB*12] MELLADO N., GUENNEBAUD G., BARLA P., REUTER P., SCHLICK C.: Growing least squares for the analysis of manifolds in scale-space. *Comp. Graph. Forum 31*, 5 (Aug. 2012), 1691–1701. 3, 9

[MGPG04] MITRA N. J., GELFAND N., POTTMANN H., GUIBAS L.: Registration of point cloud data from a geometric optimization perspective. In *Proc. SGP* (2004), pp. 23–31. 2

[PB09] PAPAZOV C., BURSCHKA D.: Stochastic optimization for rigid point set registration. In *Proc. Intern. Symposium on Advances in Visual Computing* (2009), pp. 1043–1054. 3

[PB11] PAPAZOV C., BURSCHKA D.: Stochastic global optimization for robust point set registration. *Comput. Vis. Image Underst. 115*, 12 (Dec. 2011), 1598–1609. 3

[PS04] PACH J., SHARIR M.: Geometric incidences. *Towards a Theory of Geometric Graphs (J. Pach, ed.), Contemporary Mathematics, Amer. Math. Soc., Providence 342* (2004), 185–223. 3

[PWHY09] POTTMANN H., WALLNER J., HUANG Q.-X., YANG Y.-L.: Integral invariants for robust geometry processing. *Comput. Aided Geom. Des. 26*, 1 (Jan. 2009), 37–60. 3

[RABT13] RODOLÁ E., ALBARELLI A., BERGAMASCO F., TORSELLO A.: A scale independent selection process for 3d object recognition in cluttered scenes. *International Journal of Computer Vision 102*, 1-3 (2013), 129–145. 3

[RHHL02] RUSINKIEWICZ S., HALL-HOLT O., LEVOY M.: Real-time 3D model acquisition. *Proc. SIGGRAPH 21*, 3 (July 2002), 438–446. 10

[RL01] RUSINKIEWICZ S., LEVOY M.: Efficient variants of the icp algorithm. In *Proc. 3DIM* (2001), IEEE, pp. 145–152. 2

[SG07] SHANG L., GREENSPAN M.: Pose determination by potentialwell space embedding. In *Proc. 3DIM* (2007), pp. 297–304. 2

[TCL*13] TAM G., CHENG Z.-Q., LAI Y.-K., LANGBEIN F., LIU Y., MARSHALL D., MARTIN R., SUN X.-F., ROSIN P.: Registration of 3d point clouds and meshes: A survey from rigid to nonrigid. *IEEE TVCG 19*, 7 (July 2013), 1199–1217. 2

[TG11] TAATI B., GREENSPAN M.: Local shape descriptor selection for object recognition in range data. *Comput. Vis. Image Underst. 115*, 5 (May 2011), 681–694. 3

[TM08] TUYTELAARS T., MIKOLAJCZYK K.: Local invariant feature detectors: A survey. *Found. Trends. Comput. Graph. Vis. 3*, 3 (July 2008), 177–280. 2