2009 IEEE International Conference on Robotics and Automation
Kobe International Conference Center
Kobe, Japan, May 12-17, 2009

Fast Point Feature Histograms (FPFH) for 3D Registration

Radu Bogdan Rusu, Nico Blodow, Michael Beetz
Intelligent Autonomous Systems, Technische Universität München
{rusu,blodow,beetz}@cs.tum.edu

Abstract— **In our recent work [1], [2], we proposed Point Feature Histograms (PFH) as robust multi-dimensional features which describe the local geometry around a point p for 3D point cloud datasets. In this paper, we modify their mathematical expressions and perform a rigorous analysis on their robustness and complexity for the problem of 3D registration for overlapping point cloud views. More concretely, we present several optimizations that reduce their computation times drastically by either caching previously computed values or by revising their theoretical formulations. The latter results in a new type of local features, called Fast Point Feature Histograms (FPFH), which retain most of the discriminative power of the PFH. Moreover, we propose an algorithm for the online computation of FPFH features for realtime applications. To validate our results we demonstrate their efficiency for 3D registration and propose a new sample consensus based method for bringing two datasets into the convergence basin of a local non-linear optimizer: SAC-IA (SAmple Consensus Initial Alignment).**

I. Introduction

In this paper we tackle the problem of consistently aligning various overlapping 3D point cloud data views into a complete model (in a rigid sense), also known as 3D registration. A solution to this can be found by formulating it as an optimization problem, that is, by solving for the best rotation and translation (6 DOF) between the datasets such that the distance between the overlapping areas of the datasets is minimal, given an appropriate metric space. Without any information on their initial pose in space or the datasets' overlapping areas, the problem is even more difficult and most optimization techniques are susceptible to fail in finding the best possible solution. This is because the function to be optimized is multidimensional and has local optimum solutions possibly close to the global one.

A simple classification of 3D rigid registration methods can be made based on the type of the underlying optimization method used: global or local. Perhaps the most well known efforts in the first category are based on global stochastic optimization using Genetic Algorithms [3] or evolutionary techniques [4], with their major deficiency being the actual computation time. A lot of the work done in 3D registration however falls into the second category, and the most popular registration method to date is indubitably the Iterative Closest Point (ICP) algorithm [5], [6].

The ICP method has seen many improvements from its original form, from using non-linear optimization methods [7], [8], finding good initial guesses [9], [10], or estimating better point features [9], [11], [12], to addressing the problem of ICP's computational complexity [13], [14], to name a few.

Our present contributions fall within the area of feature estimation and selection for point correspondence search, used in geometrical-based alignment methods which bring the datasets into the convergence basin of non-linear optimization algorithms. The work present in this paper constitutes incremental work from [1], [2], [15], [16]. Due to space constraints and since we already covered related publications which treat initiatives similar to ours there, we will not address them again here. Instead, throughout the remaining of the paper we will reiterate their usage through brief discussions and refer the reader to the most appropriate reference. The key contributions of the research reported in this paper include the following ones:

- optimizations of the PFH computations that reduce run time drastically by reordering the dataset and caching previously computed values;
- a revised set of features to form Fast Point Feature Histograms (FPFH) that can be computed online and which have a computational complexity of $O(k)$ (as opposed to $O(k^2)$ for PFH) while still retaining most of the descriptive power of the PFH.
- a Sample Consensus based method for the initial alignment of two datasets to fall into the convergence basin of a local non-linear optimizer (SAC-IA).

The remainder of this paper is organized as follows. The next section (Section II) introduces the Point Feature Histograms (PFH). In Section III we revise the PFH theoretical formulations and create the Fast Point Feature Histogram (FPFH). Section IV presents the applications of FPFH for the 3D registration problem using a new sample consensus based initial alignment algorithm, and to test the FPFH efficiency on noisy scanned datasets, we perform several experiments and discuss the results in Section V. Finally, we conclude and give insight on our future work in section VI.

II. Point Feature Histograms (PFH)

As we previously proposed in [1], [16], Point Feature Histograms (PFH) are informative pose-invariant local features which represent the underlying surface model properties at a point p. Their computation is based on the combination of certain geometrical relations between p's nearest k neighbors. They incorporate $\langle x, y, z \rangle$ 3D point coordinates and estimated surface normals $\langle nx, ny, nz \rangle$, but are extensible to the use of other properties such as curvature, 2^{nd} order moment invariants, etc.

In this section we perform a brief analysis on the computational model of the features, discuss their persistence over

multiple scales (i.e. different number of k neighbors), and give insight on how points on certain geometric surfaces are represented in our feature space. Furthermore, we propose an optimized algorithm that can be used to drastically decrease the features' computation time by caching previously computed values and re-using them on reordered datasets.

To illustrate and exemplify the theoretical aspects of our proposed methods, we will use the Stanford *bunny* model, because it is one of the best known and widely spread datasets in the 3D registration community.

A. Theoretical aspects

In its most basic form, the computation of a PFH at a point p relies on the presence of 3D coordinates and estimated surface normals, and is computed as follows: i) for each point p, all of p's neighbors enclosed in the sphere with a given radius r are selected (k-neighborhood); ii) for every pair of points p_i and p_j ($i \neq j$) in the k-neighborhood of p and their estimated normals n_i and n_j (p_i being the point with a smaller angle between its associated normal and the line connecting the points), we define a Darboux uvn frame ($u = n_i,\ v = (p_j - p_i) \times u,\ w = u \times v$) and compute the angular variations of n_i and n_j as follows:

$$\begin{aligned} \alpha &= v \cdot n_j \\ \phi &= (u \cdot (p_j - p_i))/||p_j - p_i|| \\ \theta &= \arctan(w \cdot n_j, u \cdot n_j) \end{aligned} \tag{1}$$

In our previous work [2], [15], besides the three features mentioned above, we used a fourth one characterizing the Euclidean distance from p_i to p_j. However, recent experiments showed that its exclusion from the PFH presents no significant decrease in robustness, especially when computed in 2.5D datasets where the distance between neighboring points increases as we move away from the viewpoint. For these scans, where the local point density influences this feature dimension, omitting the fourth feature value proved beneficial.

Figure 1 presents an influence region diagram of the PFH computation for a query point (p_q). p_q is marked with red and placed in the middle of a circle (sphere in 3D) with radius r, and all its k neighbors (points with distances smaller than the radius r) are fully interconnected in a mesh.

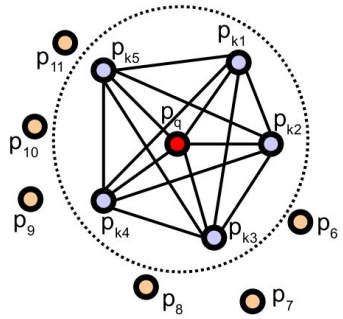

Fig. 1. The influence region diagram for a Point Feature Histogram. The query point (red) and its k-neighbors (blue) are fully interconnected in a mesh.

B. Persistence analysis

In large datasets, the number of points with the same or similar PFH might be very big and could lead to the so called "sliding" problem in registration, where points on certain surfaces do not contribute positively to the global distance metric due to ambiguous correspondences [17]. A solution would be to either segment out these surfaces at an a priori step, or to neglect all points with features which are considerably dominant in the dataset and thus concentrate on more prominent points. The latter can be achieved by performing a so called persistence analysis, that is observing which histograms are salient at which scale (k-neighborhood).

The PFH selection criterion at a given scale is motivated by the fact that in a given metric space, one can compute the distances from the mean PFH of a dataset to all the features of that dataset. As shown in [2], this distance distribution can be approximated with a Gaussian distribution, and using simple statistical heuristics, features whose distances are outside the $\mu \pm \beta \cdot \sigma$ interval can be selected as less common (therefore *unique*), where μ represents the mean PFH of the dataset and σ represents the standard deviation of the distance distribution. The parameter β controls the width of the interval and acts as a band-stop filter cut-off parameter.

To account for density variations but also different scales, the above is repeated over a discrete scaling interval (i.e. each point is enclosed in spheres with varying radii and its PFH values recomputed), and points which are marked as *unique* over the entire interval are marked as *persistent*. In particular, a point p is persistent if: i) its PFH is selected as *unique* with respect to a given radius; and ii) its PFH is selected in both r_i and r_{i+1}, that is:

$$P_f = \bigcup_{i=1}^{n-1} [P_{f_i} \cap P_{f_{i+1}}] \tag{2}$$

where P_{f_i} represents the set of points which are selected as unique for a given radius r_i.

The values of the r_i radii set are selected based on the size of the features that need to be detected. Based on the sensor's resolution, the neighborhood can contain anything from a few points to thousands or even more. For most datasets, fixing the value of α between 1..2 will give satisfactory results. Figure 2 presents the individually selected points for 3 different radii (from left to right) and the overall persistent points (right) for the Stanford bunny00 dataset.

Fig. 2. From left to right: PFH persistence over multiple scales ($r_1 = 0.003$, $r_2 = 0.004$, $r_3 = 0.005$) and the set of overall persistent points for the bunny00 dataset.

C. Geometric surface primitives signatures

To analyze the discriminating power of the PFH space, we need to look at how features computed for different geometric surfaces resemble or differ from each other. In our previous work [1], we analyzed a set of primitive 3D geometric surfaces including planes, cylinders, spheres, cones, tori, as well as edges and corners, for the purpose of scene segmentation in indoor environments. Since the PFH computation is based on normal information, the features can be separately computed and grouped for both two cases of convex and concave shapes (with the exception of the plane). The results showed that if the computation parameters are chosen carefully (i.e. the scale), the features are informative enough to differentiate between points lying on different surfaces.

Figure 3 presents the PFH signatures for points lying on 5 different convex surfaces, namely a sphere and a cylinder with a radius of 5 cm, an edge, a corner, and finally a plane. To illustrate that the features are discriminative, in the left part of the figure we assembled a confusion matrix with gray values representing the distances between the mean histograms of the different shapes, obtained using the Histogram Intersection Kernel [18]:

$$d(PFH_{\mu 1}, PFH_{\mu 2}) = \sum_{i=1}^{nr_{bins}} \min(PFH^i_{\mu 1}, PFH^i_{\mu 2}) \quad (3)$$

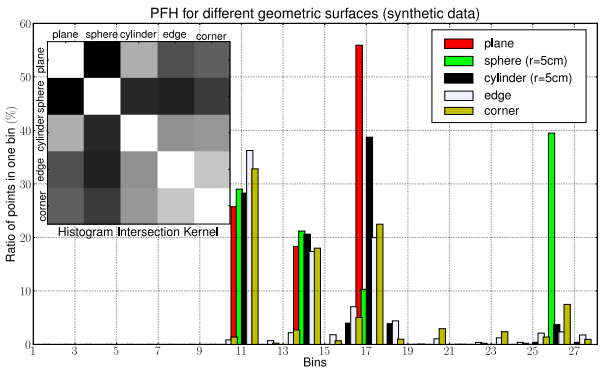

Fig. 3. Example of Point Feature Histograms for points lying on primitive 3D geometric surfaces.

D. Caching and Point Ordering

An interesting aspect regarding the computational complexity of Point Feature Histograms is given by analyzing the number of features that need to be recomputed in the neighborhoods of two query points p and q if p and q are each other's neighbors. In this case, many points of p's neighborhood will also be part of q's neighborhood, so if their respective histograms are being computed, the temporal locality of data accesses can be exploited using a cache.

The cache being employed in our implementation is growing as needed. If the size of the cache exceeds a certain limit, elements get replaced on a FIFO basis. This doesn't impact the performance too heavily, since a cache occupying 2GB can hold the feature values of more than $1.3 \cdot 10^8$ point pairs, and in our application, the FIFO replacement policy behaves quite similar to a least-recently used policy while imposing less overhead.

Note that the theoretical runtime doesn't improve with this algorithm, since still all point pairs within a neighborhood need to be considered. However, the lookups are considerably faster than the feature computations. This is reflected in the results presented in Figure 4. We conducted experiments computing PFHs for varying radii for the bunny00 dataset with unsorted (randomized) point indices and with the same dataset where the points were resorted to optimize temporal locality in the cache. That is, points that are close in the point cloud should have indices which are close together.

The reordering is being performed using a growing algorithm in Euclidean distance space using an octree to achieve similar results as minimum spanning trees in graph theory. The point order is represented in color, ranging from red for low indices to blue for high indices. Note how the cache impacts the PFH computation time considerably for the ordered dataset with a reduced runtime of about 75% compared to the standard computation method. The speedup is lower for the unordered dataset due to the random point order which renders the FIFO replacement method suboptimal.

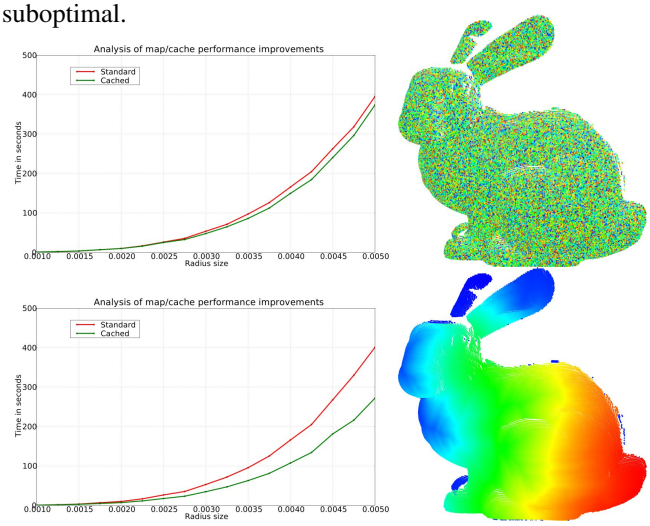

Fig. 4. Complexity Analysis on Point Feature Histograms computations for the bunny00 dataset: unordered (top), and reordered (bottom).

III. Fast Point Feature Histograms (FPFH)

The theoretical computational complexity of the Point Feature Histogram for a given point cloud P with n points is $O(n \cdot k^2)$, where k is the number of neighbors for each point p in P. A straightforward optimization is to cache feature values and reorder the point cloud dataset so that performing lookups in a data container becomes faster than recomputing the values (see Section II-D).

For realtime or near realtime applications however, the computation of Point Feature Histograms in dense point neighborhoods can represent one of the major bottlenecks in the registration framework. Therefore, in the following, we

propose a simplified version called Fast Point Feature Histograms (FPFH) that reduces the computational complexity of the algorithm to $O(n \cdot k)$, while still retaining most of the discriminative power of the PFH.

A. Theoretical aspects

To simplify the histogram feature computation, we proceed as follows: i) in a first step, for each query point p we compute only the relationships (see Equation 1) between itself and its neighbors – we will call this the Simplified Point Feature Histogram (SPFH); ii) then in a second step, for each point we re-determine its k neighbors and use the neighboring SPFH values to weight the final histogram of p (called FPFH):

$$FPFH(p) = SPF(p) + \frac{1}{k} \sum_{i=1}^{k} \frac{1}{\omega_k} \cdot SPF(p_k) \qquad (4)$$

where the weight ω_k represents the distance between query point p and a neighbor point p_k in a given metric space.

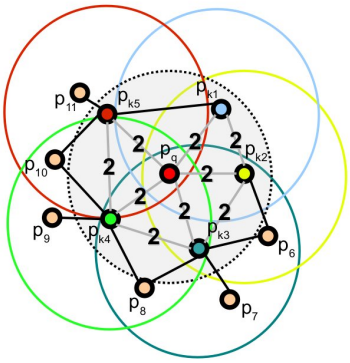

Fig. 5. The influence region diagram for a Fast Point Feature Histogram. Each query point (red) is connected only to its direct k-neighbors (enclosed by the gray circle). Each direct neighbor is connected to its own neighbors and the resulted histograms are weighted together with the histogram of the query point to form the FPFH. The connections marked with *2* will contribute to the FPFH twice.

A influence region diagram illustrating the FPFH computation is presented in Figure 5. For a given query point p_q, we first estimate its SPFH values by creating pairs between itself and its neighbors. We repeat this for all the points in the dataset, and then we re-weight the SPFH values of p_k using the SPFH values of its neighbors, thus creating the FPFH for p_q. As the diagram shows, some of the value pairs will be counted twice (marked with 2 in the figure). The differences between PFH and FPFH are: i) the FPFH does not fully interconnect all neighbors of p_q as it can be seen from Figures 1 and 5, and is thus missing some value pairs which might contribute to capture the geometry around p_q; ii) the PFH models a precisely determined surface around p_q, while the FPFH includes additional point pairs outside the r radius sphere (though at most $2r$ away); and finally iii) because of the re-weighting scheme, the FPFH combines SPFH values and re-captures some of the point neighboring value pairs.

A further PFH optimization can be pursued if we tackle the correlation problem in the feature histogram space. So far, the resulting number of histogram bins was given by q^d, where q is the number of *quantums* (i.e. subdivision intervals in a feature's value range) and d the number of features selected (in our case: $5^3 = 125$ bins). This can be described as a subdivided $5 \times 5 \times 5$ cube in 3 dimensions, where one subdivision cell corresponds to a point having certain values for its 3 features, hence leading to a fully correlated feature space. Because of this however, our resulting histogram will contain a lot of zero values (see Figure 3), and can thus contribute to a certain degree of information redundancy in the histogram space, as some of the subdivision cells of the cube will never contain any values. A simplification of the above is to decorrelate the values, that is to simply create d separate feature histograms, one for each feature dimension, and concatenate them together.

B. Persistence analysis

Using the formulation presented in Section II-B, a set of persistent features in the FPFH space can be determined. Figure 6 presents the individually selected points for 3 different radii (from left to right) and the overall persistent points (right) for the bunny00 dataset. A direct comparison to the results presented in Figure 2 clarifies that most of the discriminative power of the PFH has been retained, but indubitably some fine details are lost, especially in the areas of the bunny face and the front leg. This arises the question whether the FPFH uncorrelated space will still retain the necessary information for finding good correspondences for registration, as the PFH does [2].

Fig. 6. FPFH persistence over multiple scales for the bunny00 dataset.

C. Geometric surface primitives signatures

Similar to Figure 3, Figure 7 presents the FPFH signatures for points lying on the above mentioned geometric surfaces. Because the feature values are no longer correlated and the FPFH space does not capture precisely all the information included in the PFH, the resulted histograms are no longer as informative. However, the assembled confusion matrix using the formulation in Equation 3 shows that the FPFH signatures are still discriminative with regards to the underlying surface they characterize.

D. Online implementation

In most applications, the acquisition of 3D point cloud data is performed through the movement of a sensor (e.g. laser) beam using actuation elements. For situations when *single sweep* scans are possible, e.g. actuating a 2D laser sensor with a pan-tilt unit or a robotic arm, the FPFH formulation allows online, incremental implementations. Algorithm 1 presents a summary of the stages for our online implementation of FPFH. The presented method processes

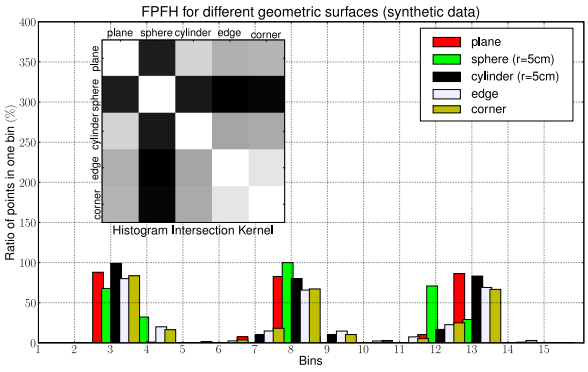

Fig. 7. Example of Fast Point Feature Histograms for points lying on primitive 3D geometric surfaces.

scanlines and maintains a list of nearest neighbors for all points in the queue. Once a new scan line doesn't affect this list for a certain point, the point in question can be processed. This introduces a small delay before a scanned point can be used, but the algorithm can estimate features close to real-time.

Algorithm 1 Online calculus of FPFH for 1-sweep scans

```
k                    // desired size of neighborhoods
Q ← {}               // queue of pts with uncompleted neighborhoods
// queue entrys also hold a list n_{q_i} of k neighbors and a sid_{q_i} entry – see below
for all scanline S = {s_i} do
  P ← {}             // list of points ready for processing
  increase(sid)      // current scan line id number
  for all points q_j ∈ Q do
    for all points s_i ∈ S do
      if s_i is closer to q_i than q_i's current neighbors then
        n_{q_i} ← n_{q_i} ∪ s_i
        if |n_{q_i}| > k then
          delete most distant point from n_{q_i}
        update sid_{q_j} ← sid
    if sid_{q_j} < sid then
      // q_i's neighborhood unaffected by cur. scan line
      P ← P ∪ q_j
      Q ← Q\q_j
  for all points s_i ∈ S do
    Q ← Q ∪ s_i
    Initialize n_{s_i} by using the same algorithm backwards
    // go back through last scanlines until sid_{s_i} stays unchanged
  for all points p_i ∈ P do
    compute FPFH on p_i's neighborhood n_{p_i}
```

IV. Sample Consensus Initial Alignment: SAC-IA

The Greedy Initial Alignment method described in our previous work [2] is very robust since it uses point cloud intrinsic, rotation invariant features. However, the computational complexity is relatively high since the merging step requires to look at all possible correspondence pairs. Also, since it is a greedy method, there are situations where it might get trapped in a local minimum. Therefore, we implemented a SAmple Consensus method that tries to maintain the same geometric relations of the correspondences without having to try all combinations of a limited set of correspondences. Instead, we sample large numbers of correspondence candidates and can rank each of them very quickly by employing the following scheme:

1) Select s sample points from P while making sure that their pairwise distances are greater than a user-defined minimum distance $d_{\min}$.
2) For each of the sample points, find a list of points in Q whose histograms are similar to the sample points' histogram. From these, select one randomly which will be considered that sample points' correspondence.
3) Compute the rigid transformation defined by the sample points and their correspondences and compute an error metric for the point cloud that computes the quality of the transformation.

Our scheme finds a good transformation fast by looking at a very large number of different correspondence triplets[1]. The error metric for the third step is determined using a Huber penalty measure L_h:

$$L_h(e_i) = \begin{cases} \frac{1}{2}e_i^2 & ||e_i|| \leq t_e \\ \frac{1}{2}t_e(2||e_i|| - t_e) & ||e_i|| > t_e \end{cases} \quad (5)$$

These three steps are repeated, and the transformation that yielded the best error metric is stored and used to roughly align the partial views. Finally, a non-linear local optimization is applied using a Levenberg-Marquardt algorithm [8]. Figure 8 presents the results obtained after registration with SAC-IA on two partial views (bunny00 and bunny90) of the Stanford bunny model.

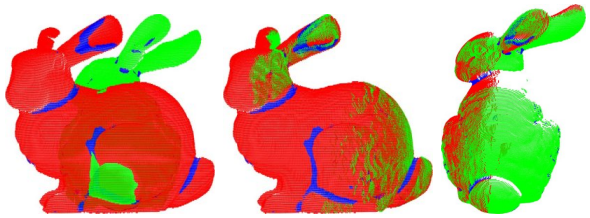

Fig. 8. From left to right: two partial views of the bunny model before alignment; results obtained after applying the solution found using SAC-IA.

V. Experimental Results on Noisy Data

To validate the FPFH space on data representing real-world scenes encountered in mobile robotics applications, we performed several experiments using the outdoor datasets from [2]. Table I presents a comparison between the two initial alignment methods, namely our previously proposed Greedy Initial Alignment (GIA) and the Sample Consensus Initial Alignment (SAC-IA), for determining the best registration solution between the two datasets presented in the left part of Figure 9. The combinatorial nature of GIA makes it extremely slow for large datasets, and thus a workaround is to use downsampled versions of the data. However this results in FPFH features being "averaged", and most of the fine details can be lost. The sample consensus based method does not suffer from these shortcomings.

The datasets have an estimated overlap of $\approx 45\%$. As shown in Table I, SAC-IA clearly outperforms Greedy IA with respect to registration speed. Note that for GIA, the number of tested combinations is directly dependent on the

[1]See attached video for details.

TABLE I

INITIAL ALIGNMENT RESULTS FOR THE LJUBLJANA OUTDOOR DATASET.

	GIA - run 1	GIA - run 2	SAC-IA
run time	$>$ 17 min	$>$ 43 min	34 sec
# points considered	200	250	10462
# of combinations	37186	58300	1000

number of points considered (p_c), since we generate all 2-point correspondences and hierarchically merge them to form 16-point correspondences (totaling $\approx p_c(p_c-1)(1-1/16)$ combinations). Considering only 200 points is already close to intractable, running for 17 minutes, or 43 minutes for 250 points. On the other hand, SAC-IA ran for 1000 iterations, and found the best transformation in iteration 476. Even though it created far less combinations, it incorporated over 10000 points in its search. To refine the registration

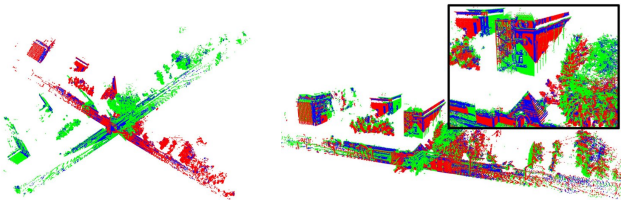

Fig. 9. Left: two overlapping urban datasets before registration (shown with red and green colors, persistent FPFH points are shown in blue); right: the alignment results obtained after applying SAC-IA.

solution we applied a local non-linear Levenberg-Marquardt optimization method as proposed in [8]. The results are presented in Figure 10.

VI. CONCLUSIONS AND FUTURE WORK

In this paper we presented two novel 3D robust features which characterize the local geometry around a point, namely the Point Feature Histogram (PFH) and its fast variant, the FPFH. The features are pose invariant and their discriminative power makes them good candidates for point correspondence search in 3D registration. To illustrate their properties, we tested their persistence over multiple scales, showed that they are informative enough to differentiate between the underlying primitive geometric surfaces they represent, performed an analysis of their computational time requirements, and presented methods for optimization. We have also presented a sample consensus based initial alignment algorithm (SAC-IA) which performs fast searches in an exhaustive FPFH correspondence space to find a good alignment solution which can be further refined using a non-linear optimization method.

Our future plans are to investigate the robustness of the feature histogram spaces for noisier point cloud data, coming from stereo or Time Of Flight cameras. Another direction of future research is to learn classifiers in the FPFH space that could be applied for fast scene segmentation, similar to our previous work in [1].

Acknowledgements This work is supported by the CoTeSys (Cognition for Technical Systems) cluster of excellence.

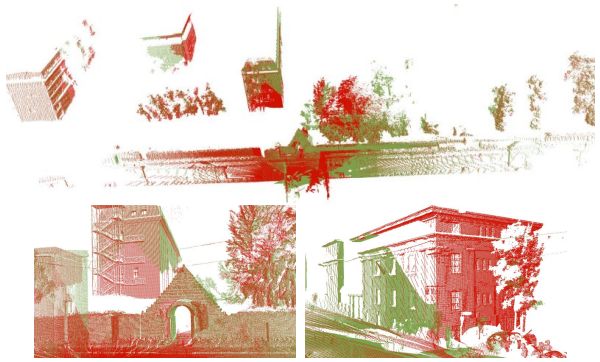

Fig. 10. Fine registration after SAC-IA, using a non-linear Levenberg-Marquardt optimization method. The two individual registered datasets are shown with red and green respectively.

REFERENCES

[1] R. B. Rusu, Z. C. Marton, N. Blodow, and M. Beetz, "Learning Informative Point Classes for the Acquisition of Object Model Maps," in *Proceedings of the 10th International Conference on Control, Automation, Robotics and Vision (ICARCV), Hanoi, Vietnam, December 17-20*, 2008.

[2] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Aligning Point Cloud Views using Persistent Feature Histograms," in *Proceedings of the 21st IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Nice, France, September 22-26*, 2008.

[3] L. Silva, O. R. P. Bellon, and K. L. Boyer, "Precision Range Image Registration Using a Robust Surface Interpenetration Measure and Enhanced Genetic Algorithms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 5, pp. 762–776, 2005.

[4] O. Cordón, S. Damas, and J. Santamaría, "A fast and accurate approach for 3D image registration using the scatter search evolutionary algorithm," *Pattern Recogn. Lett.*, vol. 27, no. 11, 2006.

[5] P. J. Besl and N. D. McKay, "A Method for Registration of 3-D Shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, 1992.

[6] Z. Zhang, "Iterative Point Matching for Registration of Free-Form Curves, Tech. Rep. RR-1658.

[7] A. Gruen and D. Akca, "Least squares 3D surface and curve matching," *International Journal of Photogrammetry and Remote Sensing*, vol. 59, pp. 151–174, May 2005.

[8] A. W. Fitzgibbon, "Robust Registration of 2D and 3D Point Sets," in *Proceedings of the British Machine Vision Conference*, 2001.

[9] N. Gelfand, N. J. Mitra, L. J. Guibas, and H. Pottmann, "Robust Global Registration," in *Proc. Symp. Geom. Processing*, 2005.

[10] A. Makadia, A. I. Patterson, and K. Daniilidis, "Fully Automatic Registration of 3D Point Clouds," in *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, pp. 1297–1304.

[11] A. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3D scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, May 1999.

[12] G. Sharp, S. Lee, and D. Wehe, "ICP registration using invariant features," 2002.

[13] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pp. 145–152, 2001.

[14] A. Nüchter, K. Lingemann, and J. Hertzberg, "Cached k-d tree search for ICP algorithms," in *Proceedings of the Sixth International Conference on 3-D Digital Imaging and Modeling (3DIM)*, 2007.

[15] R. B. Rusu, Z. C. Marton, N. Blodow, and M. Beetz, "Persistent Point Feature Histograms for 3D Point Clouds," in *Proceedings of the 10th International Conference on Intelligent Autonomous Systems*, 2008.

[16] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz, "Towards 3D Point Cloud Based Object Maps for Household Environments," *Robotics and Autonomous Systems Journal (Special Issue on Semantic Knowledge)*, 2008.

[17] N. Gelfand, L. Ikemoto, S. Rusinkiewicz, and M. Levoy, "Geometrically stable sampling for the ICP algorithm," in *Fourth International Conference on 3D Digital Imaging and Modeling (3DIM)*, oct 2003.

[18] A. Barla, F. Odone, and A. Verri, "Histogram intersection kernel for image classification," in *Proceedings of International Conference on Image Processing (ICIP)*, 2003.