15th IAPR International Conference on Machine Vision Applications (MVA)
Nagoya University, Nagoya, Japan, May 8-12, 2017.

Multi-resolution ICP for the Efficient Registration of Point Clouds based on Octrees

Michiel Vlaminck, Hiep Luong, Wilfried Philips
Image Processing and Interpretation (IPI), Ghent University, imec, Belgium
michiel.vlaminck@ugent.be

Abstract

In this paper we propose a multiresolution scheme based on hierarchical octrees for the registration of point clouds acquired by lidar scanners. The point density of these point clouds is generally sparse and inhomogeneous, a property that can yield a risk for correct alignment. Experiments demonstrate that our multiresolution technique is a lot faster than the traditional iterative closest point (ICP) algorithm while it is more robust, e.g. in case of abrupt movements of the sensor. We can report a speed-up factor of more than 30, without jeopardizing the level of accuracy. In scenarios for which the level of detail is less critical, e.g. in case of navigation for autonomous robots, we can even achieve a larger speed-up by trading speed for quality.

1 Introduction

Nowadays, the alignment or registration of point clouds, also referred to as *scan matching*, plays a crucial role in many localization and mapping solutions, often denoted as SLAM. On its turn, SLAM is a critical part for applications on autonomous vehicles or applications on critical infrastructure or disaster management. In this paper we focus on the 3D mapping of indoor 'man-made' environments. Our proposed technique can be used for both online, e.g. autonomous robots, and offline applications, e.g. mobile mapping. The main issue with a lot of existing systems is that they are still computation intensive. Many solutions have therefore been presented in literature that try to improve the speed of the *iterative closest point* (ICP) algorithm, the most popular scan matching technique. This algorithm iteratively tries to find closest point pairs that are then used to compute the transformation by minimizing the total distance between them. The most time intensive part of this algorithm is the closest point computation. Most speed-up improvements are thus based on one of the following ideas: 1) reduce the number of points, 2) reduce the number of iterations or 3) speeding up the corresponding point computation. In this work, we will tackle them all at once by incorporating a hierarchical data structure, i.e. an octree, to perform multi-resolution spatial partitioning. In each next iteration, we consider a higher resolution and based on that, we use a specific subset of points that we use to compute the transformation, denoted as *control* points. Using this technique we can not only reduce the number of iterations, we are also able to speed up the closest point computation since nearest neighbor searches can be easily found using the octree. Moreover, we show how approximate nearest neighbors can be sufficient to lead to an accurate alignment. We apply our technique to reconstruct *man-made* indoor environments, typically consisting of large planar surfaces. Examples are conference rooms or university campuses.

2 Related work

Regarding the speed-up of correspondence estimation, some studies rely on the fact that point clouds are often derived from range images which have an organized structure, i.e. a fixed width and height. In case we are interested in the transformation between two consecutive scans of the same sensor we can exploit this structure to quickly find neighbouring points. In [6], Newcombe *et al.* used this idea to perform *projective data association* (PDA), a technique that projects the points of one scan to the camera view of the previous scan. In addition, the authors focused on a hardware-oriented speed-up, exploiting the parallel nature of the PDA process by offloading it to the GPU. However, in case we have to cope with data from different modalities, this exploitation is not longer applicable. Also, in case of incremental alignment of successive scans, we often want to keep track of a global 3D map which does not have the initial structure anymore, because we want to filter *redundant* points to keep the size of the map tractable. Therefore, other 3D partitioning data structures are more appropriate, such as hierarchical bounding volumes or trees. Using these hierarchies, it is possible to reduce the processing time for the closest point computation. For example, using a k-d tree, the time complexity of the process can be decreased from $\mathcal{O}(N_s N_t)$ to $\mathcal{O}(N_s \log N_t)$ in which N_s and N_t are the number of points of respectively the source and target point cloud. The problem with k-d trees however is that it is often complicated to update them as it is very likely that the tree has to be rebuild when you insert new points to the scene. For this reason we adopt the octree data structure in this work. In [2], Hornung *et al.* presented OctoMap, an open source framework for 3D mapping using octrees. This framework was used by Beno *et al.* [5] to map the environment and subsequently help in navigating autonomous robots. However, they do not use the octree representation to help in performing the actual scan matching. Therefore, the authors of [1] proposed to combine a hierarchical searching scheme, similar to the one presented by Jost *et al.* [3], with an octree-based ICP algorithm.

3 Approach

In this work we adopt a similar approach as in [5] by changing the standard single resolution ICP to act as a multiresolution registration algorithm. To this end we use an octree to serve as a hierarchical data structure and to filter the data up to the desired resolution. In

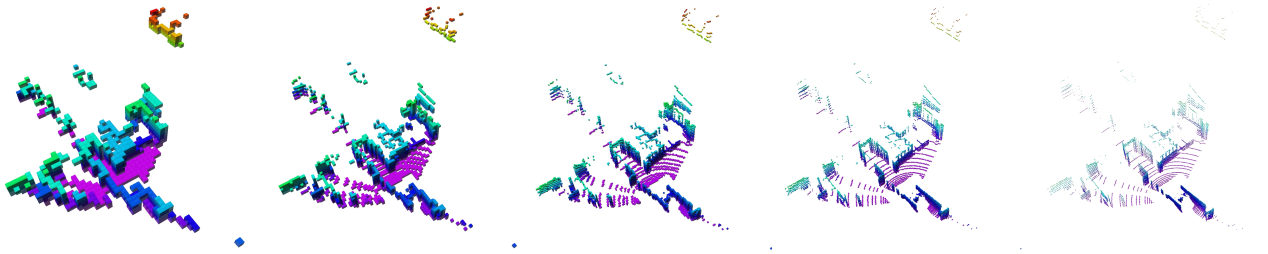

Figure 1. Picture of the multiresolution scheme. From left to right, the voxel size is respectively 0.64, 0.32, 0.16, 0.08 and 0.04 meter corresponding with resolution level 10 to 14.

the following we will briefly discuss the ICP algorithm and subsequently explain how we extended it using our octree-based multi-resolution scheme.

3.1 Iterative closest point (ICP)

In a nutshell, the ICP algorithm iteratively tries to find closest point pairs in two point clouds and subsequently computes the transformation that minimizes a distance metric and that yields the *perfect* alignment of both point clouds. Several metrics have been proposed in literature including the *point-to-point* and *point-to-plane* distance. We will use the latter one as it has been proven that it is more robust and leads to faster convergence. The *point-to-plane* distance metric is given by the following equation:

$$E(\mathcal{P}_s, \mathcal{P}_t; \mathrm{T}) = \sum_{i=1}^{N} \mathbf{w}^i ((\mathrm{T}\mathbf{p}_s^i - \mathbf{p}_t^{\mathbf{c}(i)}) \cdot \mathbf{n}_t^{\mathbf{c}(i)})^2. \quad (1)$$

In this equation, $\mathcal{P}_s$ and $\mathcal{P}_t$ are respectively the *source* and *target* point cloud that we want to align, $\mathbf{p}_s^i$ is the source point corresponding with the target point $\mathbf{p}_t^{\mathbf{c}(i)}$ with surface normal $\mathbf{n}_t^{\mathbf{c}(i)}$. The vector $\mathbf{c}$ contains the indices of the N corresponding points and $\mathbf{w}^i$ is the weight corresponding to the i-th corresponding pair. In order to solve this optimization problem, we adopt the method proposed by Low *et al.* in [4]. Regarding the estimation of the surface normals in the point cloud we adopt the same strategy as was presented in [7]. More specifically, we build a topological space of the point cloud yielding for every point its optimal neighbourhood. Using this neighbourhood, we compute for every point the eigenvectors and eigenvalues of its neighbouring points. The surface normal is then approximated by the eigenvector corresponding to the smallest eigenvalue.

3.2 Octree-based multiresolution ICP

Most of the computation time for the ICP algorithm is spent to the corresponding pair computation. The largest speed-up can hence be obtained by lowering the number of points used to estimate the transformation or by speeding up the nearest neighbour search. Motivated by this idea, we convert both the source and the target point cloud (also denoted as model point cloud) to an octree data structure. As can be seen in Figure 2, an octree is a hierarchical data structure that represents the environment as a cube which is iteratively subdivided by 8 smaller sub-cubes. The cuboid

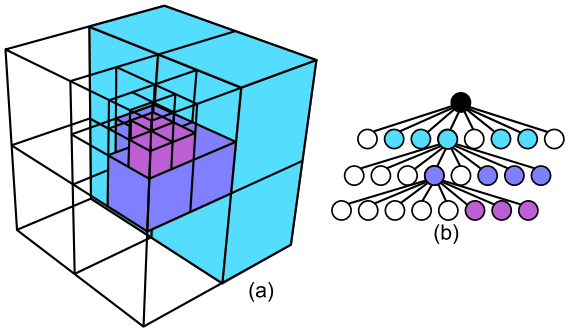

Figure 2. The octree data structure (b) and its representation in 3D (a). This hierarchical space partitioning is the core of our registration approach.

at the root level serves as a bounding box of the environment. The main idea is now to perform registration using different resolutions, going from the lower to the higher resolution, hence adopting a *coarse-to-fine* registration or alignment. For every iteration, the result of the previous one serves as an initial guess and doing so we iteratively refine the transformation estimation. In every iteration we conduct the main steps of the traditional ICP algorithm, but in contrast to existing solutions, we change the resolution every time. This hierarchical octree-based ICP algorithm has many advantages. First, it helps finding nearest neighbours in a quick way. Second, since the transformation estimated at a low resolution can serve as an initial guess for the transformation estimation at higher resolutions, it is not needed to perform multiple iterations at the same resolution level. This way we can lower the total number of iterations. Third, the displacement between two point clouds can be larger, hence leading to a higher robustness of the system. Figure 1 illustrates how the several resolution levels look like.

3.2.1 Building the tree

As mentioned in the introduction, the main environments we have in mind are indoor scenes with large planar surfaces such as conference rooms or university campuses. For these environments, the octree cells at some point in the hierarchy will contain only points that are lying on a planar surface and can hence be easily summarized by one *surface* feature, cfr. [7]. Among other things, it incorporates the centroid of the cell and the *average* normal vector and it will be updated every time a new point is added to a certain cell. We refer the reader to [7] for an entire description of this

Algorithm 1 Alignment of two point clouds

1: **function** ALIGN($\mathcal{P}_s$, $\mathcal{P}_t$, R, $\mathbf{t}$)
2: create octrees $\mathcal{T}_s$ and $\mathcal{T}_t$ for $\mathcal{P}_s$ en $\mathcal{P}_t$
3: **for** each resolution r_i in $\{r_{min}, \ldots, r_{max}\}$ **do**
4: $\mathcal{P}_{s,i} \leftarrow \mathcal{T}_s(r_i)$
5: $\mathcal{P}_{t,i} \leftarrow \mathcal{T}_t(r_i)$
6: $\mathbf{c} \leftarrow$ APPROXCLOSESTPOINTS($\mathcal{P}_{s,i}\mathcal{P}_{t,i}$)
7: $\mathbf{c} \leftarrow$ REJECT($\mathbf{c}$)
8: $\mathbf{w} \leftarrow$ WEIGHT($\mathbf{c}$)
9: $(R_i, \mathbf{t}_i) = \underset{R_{i-1},\mathbf{t}_{i-1}}{\operatorname{argmin}} E(\mathcal{P}_{s,i}, \mathcal{P}_{t,i}; T_{i-1})$
10: **end for**
11: **end function**

feature. When we want to align two consecutive scans, both the source octree $\mathcal{T}_s$ and the model octree $\mathcal{T}_t$ must be built from scratch. However, in the case we want to perform incremental registration for use in 3D reconstruction or SLAM applications, the model point cloud can be updated every time a new point cloud is added to the *map*.

3.2.2 Finding approximate closest points

In order to determine corresponding pairs, we adopt the principle of approximate closest points. Moreover, we use *mutual* closest point pairs, i.e. we consider two points matching as both the source point is the closest point to the target point and vice versa. To find the approximate closest point correspondences at resolution level r_i, we adopt the following procedure:

1. Extract the point clouds $\mathcal{P}_{s,i}$ and $\mathcal{P}_{t,i}$ by selecting for each cell at resolution r_i the centroid and the average normal vector of the cell.

2. For every point in $\mathcal{P}_{s,i}$ and $\mathcal{P}_{t,i}$, traverse the octree $\mathcal{T}_s$ or $\mathcal{T}_t$ corresponding to respectively the source and target point cloud until cell C_i at resolution r_i has been reached. This operation takes $\mathcal{O}(1)$ processing time.

The total time complexity to find corresponding points at resolution r_i is therefore given by $\mathcal{O}(\log N_{s,i})$. Hence, the computation time is only dependent on the resolution level and the size of the bounding box and not on the actual number of points of the model point cloud.

3.2.3 Octree-based registration algorithm

The idea of the algorithm is to minimize the number of points we consider to guide the minimization process. To this end, we use the octree representation of the point cloud to *filter* it spatially and to keep a summarized version. Each octree cell at all levels contains the centroid point of all points that fall within this cell as well as the average surface normal. When we want to align two point clouds, we start with processing a low resolution representation of the point cloud, obtained by considering all *nodes* up to the level with a certain voxel size. Thus, we can start with a voxel size of 64 centimeter and end up with a voxel size of 1 centimeter. Algorithm 1 summarizes this idea. Note again that in case of incremental registration, it is not needed to rebuild the octree for the target point cloud every time (line 2). Instead, the octree can be updated every time new points are discovered. On line 4 and 5, $\mathcal{T}_s(r_i)$ and $\mathcal{T}_t(r_i)$ are representing the functions that extract the centroid point cloud using resolution level r_i. Next, using these two *new* point clouds, the closest point pairs are determined as explained in the previous section. However, it can still happen that some of these closest points are bad corresponding pairs and are hence acting as *outliers*. Therefore, the REJECT function will check if the surface normal is *sufficiently* close, i.e. if the angle between the two vectors is small. Based on the feature representations (including the normal vectors), a weight is assigned for each corresponding pair based on the distance in feature space. Finally, the error metric presented in 1 is minimized using these weights.

Using this multi-resolution technique, it is (no longer) needed that for each resolution the iterations should continue until convergence. In fact, only one iteration seems to be sufficient for the intermediate resolutions. Only at the lowest resolution the iterations can continue until convergence, but even there we see that often times, only one iteration is needed.

4 Experimental results

We conducted some experiments using data that was captured at a campus of Ghent University. The data was acquired using a Velodyne HDL-32e lidar scanner that was put horizontally. The goal was to perform sequential registration of consecutive point clouds and hence to obtain large-scale 3D reconstructions of indoor environments. In total 14 sequences were evaluated, all of them containing 30 seconds to 3 minutes of lidar data. Table 1 gives an overview of the results of our octree-based multiresolution alignment process. These results summarize the processing times as well as the residuals of the different resolution levels. The term residual denotes the final cost of the ICP procedure after convergence, i.e. the Euclidean distance of all corresponding point pairs. All of the experiments were conducted on a laptop computer with an Intel Core i7-4712HQ, 2.30Ghz CPU inside and 16GB RAM. The experiments were conducted using OpenMP using the 4 cores of the CPU at the same time for the correspondence estimation step. As can be seen, the process converges quite quickly to the optimal *residual*. In other words, between the iterations at resolution level 12 and 13, the total gain is less than one millimeter. Thus, for some applications, e.g. local map building for autonomous vehicles, the level of accuracy achieved at resolution level 13 is sufficient and can be achieved in ± 31 ms. Even if we pursue the best possible accuracy we can state that our multi-resolution is beneficial in terms of processing speed. After resolution 16, corresponding to a leaf size of 1 cm, we could not get any better residual. This is logical as the Velodyne lidar data itself has an accuracy of ± 2 cm, independently of errors made in the pose estimation process. Still we can report that on the local level, i.e. without taking into account the error propagation in case of subsequent point cloud registration, we achieve sub-centimeter accuracy. More precisely, the average distance of corresponding pairs in consecutive point clouds approximates 0.27. This can be denoted

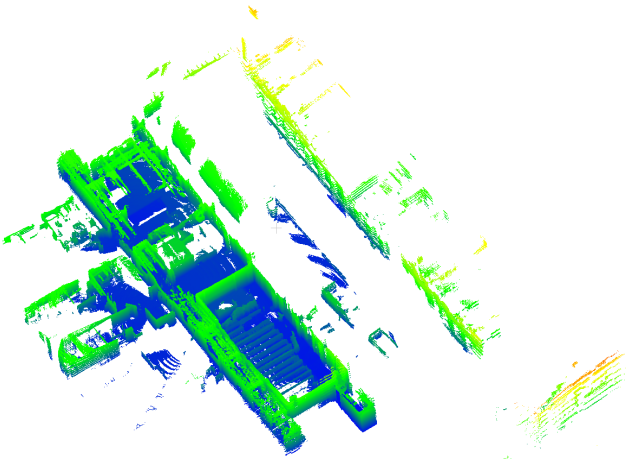

Figure 4. Examples of a 3D reconstruction. The blue color means that the points are located at a lower height, whereas the orange color means that the points are located at a greater height.

Table 1. Results of the octree-based alignment process. In this table, the term residual denotes the final cost of the ICP procedure after convergence, i.e. the Euclidean distance of all corresponding point pairs. The time t denotes the time that is needed to compute the transformation using the respective resolution.

level	residual (cm)	t(ms)
initial	3.1907	
11	0.6309	5
12	0.3027	11
13	0.2799	16
14	0.2725	32
15	0.2719	51
16	0.2716	77
total	0.2716	**192**
non-MR	0.2730	**7212**

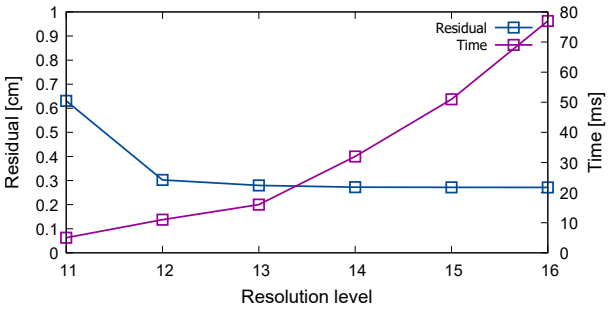

Figure 3. Residual and processing time for each resolution level.

as very accurate. Moreover, this entire process takes only 192ms whereas our non multi-resolution technique takes slightly more than 7 seconds, hence leading to a speed-up factor of 37. Figure 4 finally shows an example of a 3D reconstruction result we obtained using this approach on data that was recorded in an auditorium of Ghent University. Visually, we can not see any discrepancies in the 3D reconstruction as the walls are straight and no *ghost* objects can be noticed.

5 Conclusion

In this paper a new algorithm was presented for the registration of point clouds acquired by lidar scanners. The solution adopts a coarse-to-fine approach by using an octree-based ICP algorithm. Our solution is not only faster than traditional approaches, it is also more robust against abrupt movements of the sensor. Experiments demonstrate that we achieve a speed-up factor of more then 30 while the quality remains equal. We can state that we can achieve a sub-centimeter accuracy on the local level.

Acknowledgement

This research is part of the ARIA project, an ICON project co-funded by imec, digital research institute founded by the Flemish Government.

References

[1] Jianda Han, Peng Yin, Yuqing He, and Feng Gu. Enhanced icp for the registration of large-scale 3d environment models: An experimental study. *Sensors*, 16(2):228, 2016.

[2] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 2013.

[3] Timothée Jost and Heinz Hügli. A multi-resolution ICP with heuristic closest point search for fast and robust 3d registration of range images. In *4th International Conference on 3D Digital Imaging and Modeling (3DIM 2003), 6-10 October 2003, Banff, Canada*, pages 427–433, 2003.

[4] Kok-Lim Low. Linear least-squares optimization for point-to plane icp surface registration. Technical report, Chapel Hill, University of North Carolina, 2004.

[5] Beno P., Pavelka V., Duchon F., and Dekan M.i. Using octree maps and rgbd cameras to perform mapping and a* navigation. In *International Conference on Intelligent Networking and Collaborative Systems (INCoS)*. IEEE, Sep 2016.

[6] Otmar Hilliges David Molyneaux David Kim Andrew J. Davison Pushmeet Kohli Jamie Shotton Steve Hodges Andrew Fitzgibbon Richard A. Newcombe, Shahram Izadi. Kinectfusion: Real-time dense surface mapping and tracking. In *IEEE ISMAR*. IEEE, October 2011.

[7] Michiel Vlaminck, Hiep Luong, Werner Goeman, and Wilfried Philips. 3d scene reconstruction using omnidirectional vision and lidar: A hybrid approach. *Sensors*, 16(11):1923, 2016.

[8] Jinkun Wang and Brendan Englot. Fast, accurate gaussian process occupancy maps via test-data octrees and nested bayesian fusion. In *2016 IEEE International Conference on Robotics and Automation, ICRA 2016, Stockholm, Sweden, May 16-21, 2016*, pages 1003–1010, 2016.

[9] T. Whelan, M. Kaess, M.F. Fallon, H. Johannsson, J.J. Leonard, and J. McDonald. Kintinuous: Spatially extended kinectfusion. In *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, Sydney, Australia, Jul 2012.