Automatic Merging of Lidar Point-Clouds Using Data from Low-Cost GPS/IMU Systems

Scott E. Budge[a] and Kurt von Neiderhausern[b]

[a]Center for Advanced Imaging Ladar, Utah State University, Logan, UT 84322-4170, (435) 797-3433

[b]Ball Aerospace & Technologies Corp, 1600 Commerce, Boulder, CO 80301, (303)939-4555

ABSTRACT

Stationary lidar (Light Detection and Ranging) systems are often used to collect 3-D data (point clouds) that can be used for terrain modelling. The lidar gathers scans which are then merged together to map a terrain. Typically this is done using a variant of the well-known Iterated Closest Point (ICP) algorithm when position and pose of the lidar scanner is not accurately known. One difficulty with the ICP algorithms is that they can give poor results when points that are not common to both scans (outliers) are matched together.

With the advent of MEMS (microelectromechanical systems)-based GPS/IMU systems, it is possible to gather coarse position and pose information at a low cost. This information is not accurate enough to merge point clouds directly, but can be used to assist the ICP algorithm during the merging process.

This paper presents a method called Sphere Outlier Removal (SOR), which accurately identifies outliers and inliers, a necessary prerequisite to using the ICP algorithm. SOR incorporates the information from a low cost GPS/IMU to perform this identification. Examples are presented which illustrate the improvement in the accuracy of merged point clouds when the SOR algorithm is used.

Keywords: lidar, ladar, ICP, outlier

1. INTRODUCTION

Point-cloud data acquired from lidar (Light Detection and Ranging) systems is increasingly used to build 3-D models of objects such as vehicles, buildings, and terrain. These models can then be used for a wide variety of applications, including automatic target recognition (ATR), obstacle detection and avoidance for autonomous vehicles, landscaping, civil engineering projects and mining.

The main problem that each application shares is that a lidar that collects data from a single viewpoint can't create a complete 3-D point cloud that completely covers the object or terrain of interest. This requires several scans to be acquired from different viewpoints, and the scans must then be merged together. If accurate knowledge of the position and pose of the sensor at each viewpoint is available in a common coordinate system, the scans can be easily transformed to the common reference frame and merged to create the complete 3-D model.

Knowledge of the position and pose can be acquired from a highly accurate GPS/IMU system. Although available, systems with the accuracy necessary to allow direct merging of point clouds can be very expensive. Recent developments in MEMS (microelectromechanical systems)-based GPS/IMU sensors have made it possible to gather position and pose information at a low cost. Although these sensors are improving, current MEMS sensors do not allow for direct merging of lidar scans accurately enough for many applications.

Often the merging of lidar scans is accomplished by the well-known Iterated Closest Point (ICP) algorithm proposed by Besl and McKay when position and pose information is inaccurate or unavailable.[1] There are several variations of the ICP algorithm that have been proposed, all of which are derivations from the original.[2–7]

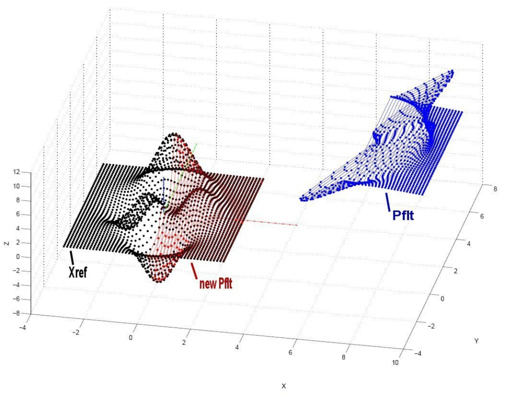
Originally, the ICP algorithm was designed to be used in a point-based approach. This algorithm sets the metric based on the Euclidean distance between two points in different scans. However, since a point to point correspondence does not take into account the surfaces of the object, it suffers from the inability to "slide" overlapping range images.[8] One method used to mitigate these slight misalignments is to use the point-plane Euclidean distance, which can be computed by evaluating the distance between the point and its neighboring plane.[9] The main advantage to using a point-plane distance calculation is that it is more accurate when registering two data sets.

One of the main disadvantages of ICP algorithms is that although they are able to correctly register points when they have a correlating match, when outliers are present the algorithm can fail. Besl and McKay mention that outlier removal is a necessary step in the pre-registration process, but do not discuss any method for outlier removal.

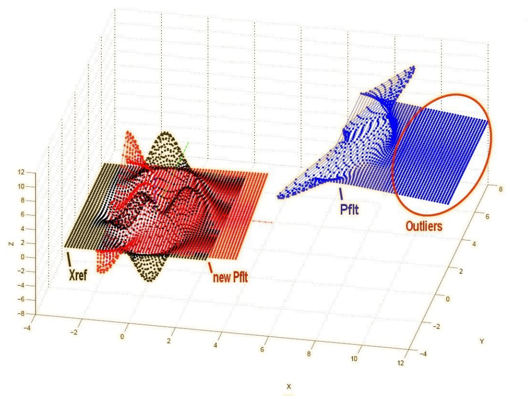
This paper presents a method of merging (registering) lidar scans when coarse position and pose information is available to help eliminate outliers. The paper proceeds as follows: in Section 2, an example of the problems with outliers is presented. Section 3 introduces the Sphere Outlier Removal approach to eliminating outliers, and Section 4 presents experimental results from using this method. Finally, Section 5 presents conclusions drawn from this research.

2. OUTLIER REMOVAL

An example of the outlier problem is given in Fig. 1. In this example, the ICP algorithm was used to register two data sets, X_{ref}, and P_{flt}. In Fig. 1a, X_{ref} (black) has more points than P_{flt} (blue), but because P_{flt} does not contain any outliers the ICP algorithm correctly registers the data newP_{flt} (red). The data set shown in Fig. 1b contains the same number of points in the reference frame (X_{ref}) (black) and the floating frame (P_{flt}) (blue), but half the points are considered outliers, and the ICP algorithm incorrectly registers the data newP_{flt} (red).



(a)



(b)

Figure 1. The effect of outliers on ICP merging. (a) P_{flt} (blue) does not contain any outliers, and the ICP algorithm correctly registers the data newP_{flt} (red). (b) P_{flt} (blue) contains outliers, and the ICP algorithm incorrectly registers the data newP_{flt} (red).

As can be seen, the removal of outliers is a necessary step that must be completed in order for the correct registration of data sets.

2.1 Current Outlier Removal Techniques

Even when most of the point correspondences are correct, one still has to deal with outliers resulting from mismatches and noise-corrupted data points.[8] The underlying problem here is how to robustly reject outliers. There are three main classes for the solution in the field of robust statistics. The first class, outlier thresholding, is the simplest method to implement and the fastest computationally.[8, 10] This method is is most widely used in vision applications because of its speed. Outlier thresholding estimates the standard deviation σ of the registration errors in the data set during each iteration and then removes points which have errors greater than

$|k\sigma|$ where $k \geq 3$. One problem with outlier thresholding is that an estimate of σ may be incorrect if there are more outliers than inliers. Another problem with outlier thresholding is that regardless of what value k is chosen, some points which are valid will be classified as outliers and some points which are outliers will be classified as valid.

The second class of robust estimators is the median/rank estimation method.[10] This method selects the median or kth value, for some percentile k with respect to errors for each observation and use that value as the error estimate. The median value is almost always guaranteed not to be an outlier as long as half the data is valid. An example of median estimator is the least-median-of-squares method (LMedS).[10] LMedS requires an exhaustive search of all the possible combinations of point correspondences with relation to the median value. While these median-based methods can be fairly robust, the time required to do an exhaustive search over a small data set can be very long.

The third class of outlier removal is called M-estimation.[10] M-estimation is a generalization of least squares, where "M" refers to maximum likelihood estimation. The general form of M-estimation defines a probability distribution which is maximized by minimizing a function of the form:

$$E(z) = \sum_i \rho(z_i), \tag{1}$$

where $\rho(z)$ is an arbitrary function of the errors z_i.

The disadvantage of these types of outlier removal techniques is they are computationally expensive and must be run during each iteration of the ICP algorithm, therefore slowing the speed of the registration process dramatically. Thresholding has the added disadvantage that it is not very robust. Median/Rank requires that 50% of the data be overlapping, and M-estimation requires an exhaustive search and is the most computationally expensive.[10]

3. SPHERE OUTLIER REMOVAL

The problem with the previous methods of outlier removal is that none of them are able to specify which points are outliers and which point are inliers before a registration is attempted. Instead they will compute inliers and outliers with each iteration of the ICP algorithm, thus drastically reducing the speed of the ICP algorithm. The Sphere Outlier Removal (SOR) algorithm identifies the inliers and outliers before the ICP algorithm is started, enabling this procedure to be done one time at the beginning.

The SOR algorithm is developed using the observation that if we have noisy information on the pose of the sensor, we can use that information to determine limits on the farthest two corresponding points in X_{ref} and P_{flt} can be separated. For example, if we know the measurements from a low-cost pose sensor have a 1-σ error of $\pm 0.5°$, we can find a sphere of appropriate radius centered at each point in X_{ref} in which we would expect corresponding points from P_{flt} to fall. Those points in P_{flt} that are not within the appropriate distance from corresponding points in X_{ref} can be rejected as outliers and discarded from the data set before the ICP algorithm is run.

The radius of the sphere containing inliers can be found by finding the maximum distance a point will move when errors are applied to the three pose angles, given by yaw (ψ), pitch (θ), and roll (ϕ). Note that this movement, from the viewpoint of the lidar scanner, arises from a combination of both the pose of the scanner and the angles that describe the direction the scanner is pointed for a particular measurement. This movement is given in homogeneous coordinates by the relationship

$$\begin{bmatrix} x_q \\ y_q \\ z_q \\ 1 \end{bmatrix} = R_{\psi\theta\phi} \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix}, \tag{2}$$

where

$$R_{\psi\theta\phi} = \begin{bmatrix} c\psi c\theta, & -s\psi c\phi + c\psi s\theta s\phi, & s\psi s\phi + c\psi s\theta c\phi, & 0 \\ s\psi c\theta, & c\psi c\phi + s\psi s\theta s\phi, & -c\psi s\phi + s\psi s\theta c\phi, & 0 \\ -s\theta, & c\theta s\phi, & c\theta c\phi, & 0 \\ 0, & 0, & 0, & 1 \end{bmatrix}, \tag{3}$$

and $c\psi$, $s\psi$ are abbreviations for $\cos(\psi)$ and $\sin(\psi)$, respectively, and similarly for the other terms. The squared Euclidean distance is thus given by

$$d^2(x_p, y_p, z_p, \psi, \theta, \phi) = \|R_{\psi\theta\phi}\boldsymbol{p} - \boldsymbol{p}\|_2^2, \tag{4}$$

or

$$\begin{aligned} d^2(\cdot) = 2[&x_p^2(1 - c\psi c\theta) + y_p^2(1 - c\psi c\phi - s\psi s\theta s\phi) + z_p^2(1 - c\theta c\phi) + \\ &x_p y_p(s\psi c\phi - s\psi c\theta - c\psi s\theta s\phi) + y_p z_p(c\psi s\phi - c\theta s\phi - s\psi s\theta c\phi) + \\ &x_p z_p(s\theta - s\psi s\phi - c\psi s\theta c\phi)]. \end{aligned} \tag{5}$$

We note that (5) is highly nonlinear, and there are several maxima in the error distance. Through numerical analysis of (5), we find that the maxima occur under four combinations of ψ, θ, ϕ:

$$\psi, \theta, -\phi, \tag{6}$$
$$\psi, -\theta, \phi, \tag{7}$$
$$-\psi, \theta, \phi, \tag{8}$$
$$-\psi, -\theta, -\phi. \tag{9}$$

These combinations arise because of the fact that $\cos(\psi)$, $\cos(\theta)$, $\cos(\phi)$ are all even functions and $\sin(\psi)$, $\sin(\theta)$, $\sin(\phi)$ are all odd functions. The maximum value of d is thus given as:

$$\begin{aligned} d_{max} = \max[&d(x_p, y_p, z_p, \psi, \theta, -\phi), d(x_p, y_p, z_p, \psi, -\theta, \phi), \\ &d(x_p, y_p, z_p, -\psi, \theta, \phi), d(x_p, y_p, z_p, -\psi, -\theta, -\phi)]. \end{aligned} \tag{10}$$

3.1 SOR Algorithm Implementation

The SOR algorithm uses a variable distance for the sphere radius depending on the (x_p, y_p, z_p) value of the points in P_{flt}. Once the radius is found using (10), all the points in X_{ref} must be searched for points within the sphere surrounding (x_p, y_p, z_p). This can be done by performing a radius search with a KD-Tree, where the KD-Tree is designed to efficiently find the points within a given radius of a point.[11] It is constructed from the points in X_{ref} before outlier removal is performed. The SOR pseudo code is shown in Algorithm 1.

Algorithm 1 uses as input the sets X_{ref} and P_{flt}, the scans to be merged. The points in X_{ref} are used to create a KD-Tree which is optimized for radius searches. It should be noted that the KD-Tree can either be made with the X_{ref} points or the P_{flt} points. If one point set is larger than the other, then creating the KD-Tree out of the larger point set would produce faster speeds, since it will only have to loop over all the points in the smaller set. The algorithm calculates (10) with the ψ, θ, ϕ values set to the 1-σ angle errors in the pose sensor. Figure 2 illustrates how the radius of the spheres would change based on the location of the point being checked, for the case where $\sigma = \pm 0.5°$. Points that are near the origin of the scanner will have a smaller radius to search for neighboring points, while points that are farther away from the origin will have a larger radius to search.

4. SOR EXPERIMENTAL RESULTS

We will now demonstrate how well the SOR algorithm works verses using a fixed radius to designate inliers and outliers. The IMU sensor selected for the experiments has an error for the three angles of $\sigma = \pm 0.5°$. Two lidar scans of the Old Main Building at Utah State University were acquired. Using the pose measurements from the IMU, the scans can be registered by transforming points from the second scan into the reference frame of the first scan. The result of this process is illustrated in Fig. 3. The error in the registration is obvious.

Algorithm 1 Sphere Outlier Removal (SOR)

```
Input:
        $P_{flt}$, %floating frame points
        $X_{ref}$ %reference frame points
        $\psi, \theta, \phi$ %$\sigma$ for Yaw, Pitch, and Roll
Output:
        $P_{in}$, %inlier points
        $P_{out}$, %outlier points
        $\mathrm{perc}_{overlap}$ %percent overlap
Local:
        $N_x$, %number of point in the reference frame
        $N_p$, %number of point in the floating frame
        $N_s$, %number of point in the range search
        $N_{in}$, %number of inliers points
        $p_i$, %the ith floating point
        $\mathrm{max}_{dist}$, %the maximum distance calculated by (10)
        $p_{sphere}$ %the points that are within the sphere radius search
begin
        KDTree ← MakeRadiusSearchKDTree($X_{ref}$) % create $X_{ref}$ KD-Tree
        for($i = 0 \ldots N_p$)
        {
            $p_i = P_{flt}(i)$
            $\mathrm{max}_{dist} = d_{max}(p_i.x, p_i.y, p_i.z, \psi, \theta, \phi)$ (10)
            %find points within radius of $p_i$
            $p_{sphere} = \mathrm{RadiusQuery}(KDTree, p_i, \mathrm{max}_{dist})$
            $N_s = \mathrm{length}(p_{sphere})$ %number of points within radius of $p_i$
            if($N_s > 0$)
                $P_{in} = [P_{in}, p_i]$ %add the point to the inliers
            else
                $P_{out} = [P_{out}, p_i]$ %add the point to the outliers
        }
        $N_{in} = \mathrm{length}(P_{in})$
        $\mathrm{perc}_{overlap} = 100\frac{N_{in}}{N_p}$
end
```

For points directly in front of the lidar, $(\psi, \theta, \phi) = (0, 0, 0)$, and a range of 30 meters, the maximum distance that a single point can be in error based on the $\pm 0.5°$ changes in yaw, pitch, or roll is 0.524 meters. One method for determining inliers and outliers is just using a worst case radius over all the points — in our case the fixed radius would be 0.524 meters. Figs. 4a and 4b illustrate the differences between the fixed radius and variable radius outlier removal algorithms. The images are highlighted to help observe the inliers within the two scans. The X_{ref} is highlighted yellow, the P_{flt} is highlighted red, and the inliers are highlighted cyan.

Figure 4a shows the results of using a fixed radius of 0.5 meter when removing the outliers within the point clouds of two of the scans taken with the lidar. Notice that the area where X_{ref} and P_{flt} start to overlap, the inliers (cyan) are marked about 0.5 meters away from the X_{ref} points marked in yellow. This is due to the fact that we used a fixed radius when determining inliers and outliers. This leads to a large number of points that are marked as inliers that should actually be outliers. Using a fixed radius for outlier removal fails to assign the inliers and outliers correctly, because some outliers are found within that 0.5 meter distance.

In comparison, Fig. 4b shows the removal of outliers using SOR algorithm which uses a variable radius. This figure shows that the excess points collected in the previous example are almost entirely removed by using smaller spheres when the points are close to the lidar origin (located at the center of the square cut out of the of the yellow points) and larger spheres when the points are located farther away from the lidar origin.

For further comparison, Table 1 shows the results of using the SOR algorithm before the ICP algorithm versus

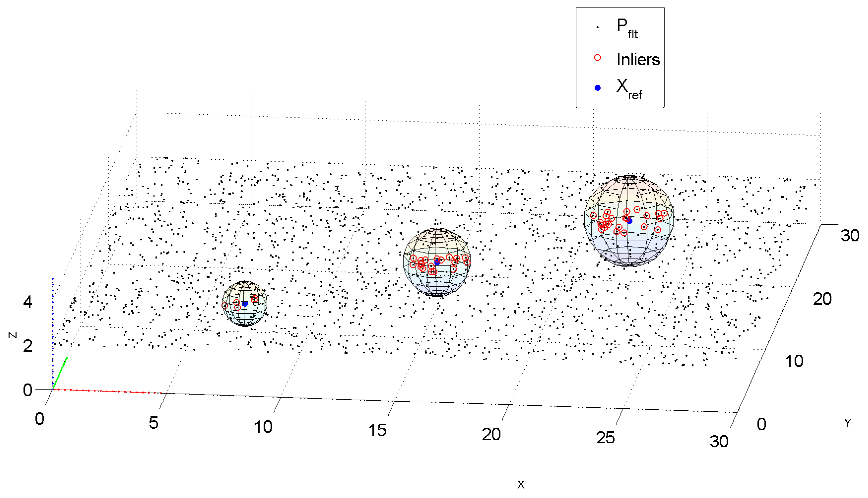


Figure 2. Inliers within a variable radius. In this example, the blue point are the reference points, the floating points are marked in black and the points that are within the sphere are considered inliers.

using a fixed radius for eight scans of Old Main taken by the lidar. These scans are adjacent and overlap the scans ordered before and after each scan. Notice that the Root Mean Squared Error (RMSE) between the scans after ICP registration is consistently less when the SOR algorithm is used. Although the fixed radius results in more reported inliers, some of these inliers are actually outliers and do not help in the registration process.

Table 1. Comparing SOR vs. fixed radius.

Scans	SOR $RMSE_{icp}$	Inliers	Fixed Radius 0.5 m $RMSE_{icp}$	Inliers
1 / 2	0.1445	24103	0.2248	40545
2 / 3	0.0945	33134	0.1280	68737
3 / 4	0.1106	6479	0.2759	36014
4 / 5	0.1244	11106	0.3176	57145
5 / 6	0.1228	20952	0.2223	61614
6 / 7	0.1112	2304	0.1869	3380
7 / 8	0.1826	3588	0.4390	19879

5. CONCLUSION

The removal of outliers is an important step to correctly registering 3D images with the ICP algorithm. Currently, the outlier removal techniques do not take into account the *a priori* orientation of the pose sensor and the measurement error of the sensor. Using the sensors pose error tolerances a new algorithm called the Sphere Outlier Removal algorithm was developed.

This algorithm is performed only once, at the start of the ICP registration process. It requires finding the points in one scan that are within a variable distance of points in the second scan. This distance is determined by the accuracy of the pose sensor and the point position relative to the lidar scanner. A search method based on KD-Trees is used to accelerate this search process.

The SOR algorithm was found to remove outliers better than a fixed radius approach. With outliers and inliers better identified, the ICP algorithm can more accurately register scans taken by the lidar.

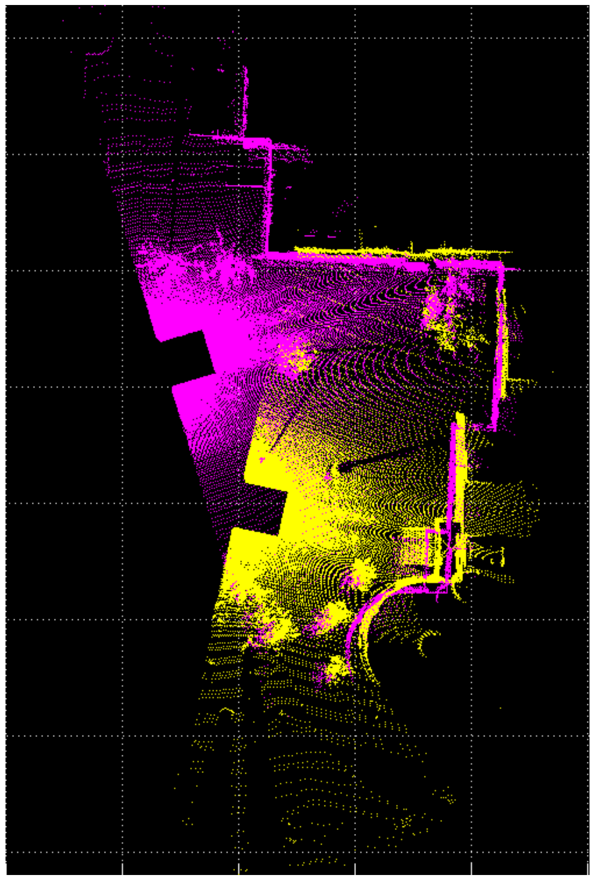


Figure 3. Scan 1 (yellow) and scan 2 (magenta) registered using coarse IMU measurements.

REFERENCES

[1] Besl, P. J. and McKay, N. D., "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.* **14**, 239–256 (Feb. 1992).

[2] Pajdla, T. and Gool, L. V., "Matching of 3D curves using semi-differential invariants," tech. rep., Computer Vision Laboratory Czech Technical University Karlovo namesti 13 121 35 Prague, Czech Republic (1995).

[3] Chetverikov, D. and Stepanov, D., "Robust Euclidean alignment of 3D point sets: The trimmed iterative closest point algorithm," tech. rep., Center for Applied Cybernetics, Faculty of Electrical Engineering, Czech Technical University (2002).

[4] Langis, C., Greenspan, M., and Godin, G., "The parallel iterative closest point algorithm," tech. rep., Institute for Information Technology, National Research Council Canada (2002).

[5] Stewart, C., Tsai, C.-L., and Roysam, B., "The dual-bootstrap iterative closest point algorithm with application to retinal image registration," *IEEE Trans. Med. Imag.* **22**, 1379–1394 (Nov. 2003).

[6] Estepar, R. S. J., Brun, A., and Westin, C. F., "Robust generalized total least squares iterative closest point registration," tech. rep., Harvard Medical School (2002).

[7] Louw, M. and Nicolls, F., "An approximate EM homographical iterative closest point algorithm," tech. rep., Department of Electrical Engineering University of Cape Town, South Africa (2004).

[8] Nishino, K. and Ikeuchi, K., "Robust simultaneous registration of multiple range images," in [*Proc. 5th Asian Conf. Computer Vision*], 454–461 (2002).

[9] Chen, Y. and Medioni, G., "Object modeling by registration of multiple range images," in [*Proc. IEEE Intl. Conf. Robotics and Automation*], 2724–2729 vol.3 (Apr. 1991).

[10] Wheeler, M. D., *Automatic Modeling and Localization for Object Recognition*, PhD thesis, Carnegie Mellon University, Pittsburgh (1996).

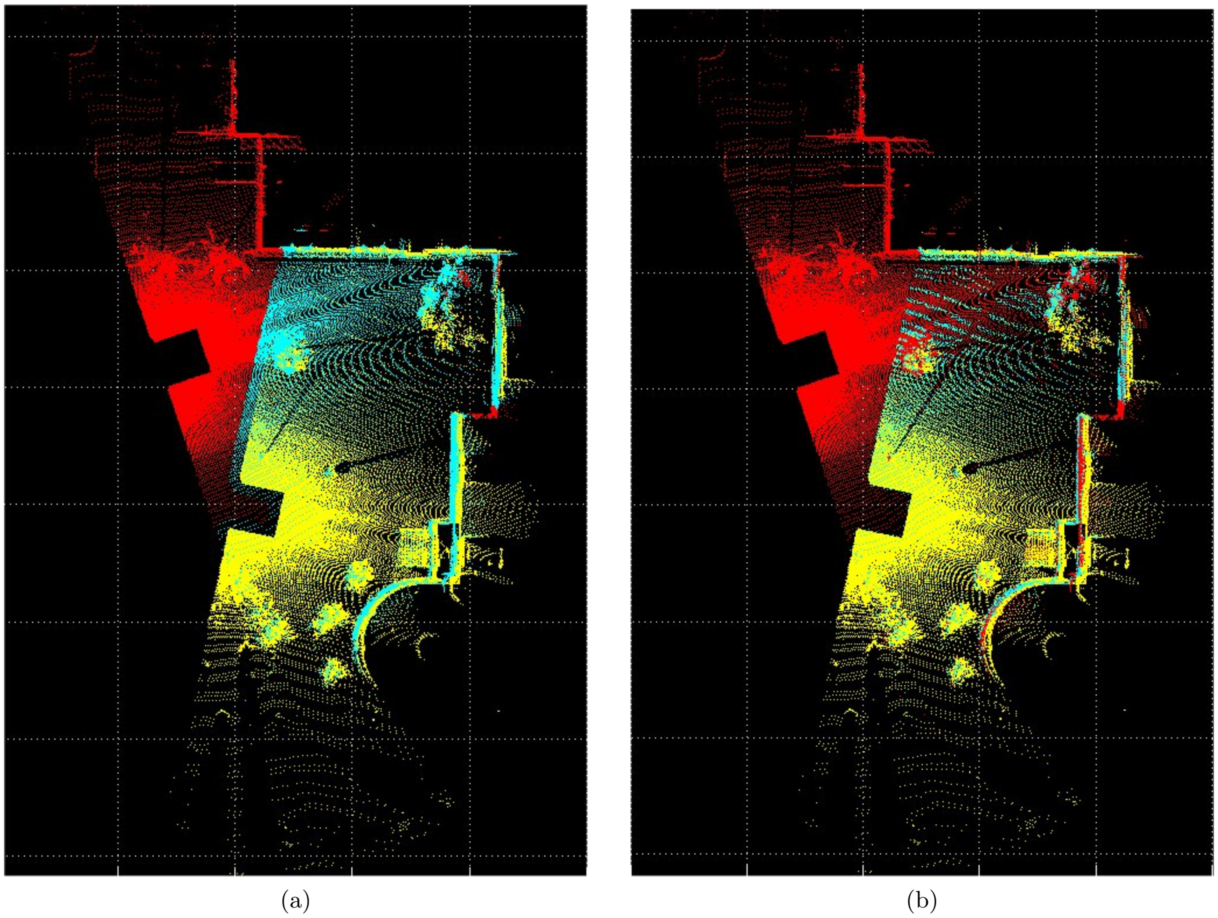


Figure 4. Comparison of inlier detection. X_{ref} (yellow), P_{flt} (red), Inliers (cyan). (a) Fixed radius of 0.5 m. (b) Variable radius using SOR.

[11] Bentely, J. L., "K-D trees for semidynamic point sets," tech. rep., AT&T Bell Laboratories (1990).