Simultaneous Multi-Line-Segment Merging for Robot Mapping using Mean Shift Clustering

Rolf Lakaemper

***Abstract*— Line segment based representation of 2D robot maps is known to have advantages over raw point data or grid based representation gained from laser range scans. It significantly reduces the size of the data set. It also contains higher geometric information, which is necessary for robust post processing. The paper describes an algorithm to convert global 2D robot maps to line segment representation, using a pre-aligned set of point-based single scans as input. Mean-shift clustering on the set of all line segments is utilized to merge perceptually similar segments to single instances: locally linear features in the environment are unambiguously represented by single line segments in the final global map. Apart from a scaling parameter, the approach is parameter free. Experiments on real world data sets prove its applicability in the field of robot mapping.**

I. Introduction

Robot mapping based on laser range scans is a major field of research in robotics in the recent years. The basic task of mapping is to combine spatial data usually gained from laser range devices, called 'scans', to a single data set, the 'global map'. The global map represents the environment scanned from different locations. Although most of the current approaches [6], [13], [16] use data points to create and to represent the global map, it becomes increasingly popular to use geometric mid level data representation, like line segments, basic geometric objects or splines, or, in 3D, linear patches [17]. The main advantages of such a representation are

- High data compression rate. Representing e.g. a wall as a single segment means storing its two endpoints only.
- Higher geometric information level. Storing data using mid level geometric elements simplifies further processing steps. For example, finding rectangles in a map is significantly simpler if the map data consists of line segments, not of raw data points, see e.g. [9].
- High level of accuracy: compared to grid based map representation, segment based maps are "*...more accurate since they provide floating point resolution and do not suffer from discretization problems*" [18].

This paper aims at representation of 2D laser data using line segments. More specific, it processes a simplification of *already segment based* robot maps. This is not a limitation, since in general it is simple to compute a non optimal segment-based pre-representation of a global map: in the typical case, global robot maps consist of aligned single scans. (Non optimal) segment detection in point-based single scans is a solved task, see e.g. [14], [8] or Figure 5a,b. The problem with such single scan line segment representation is, that a single object might be present in multiple scans; in the aligned global map, a single object might be represented by multiple line segments. This representation is not only highly redundant, but errors in data acquisition and alignment lead to inconsistencies. Figure 5a,b shows an example.

However, having advanced point-based alignment algorithms and simple and fast line segment fitting for single scans at hand, the presented approach offers fast and reliable conversion of aligned point-based global maps to segment-based global maps. The approach merges perceptually similar segments to a single segment and solves the problem of inconsistency and redundancy.

The approach utilizes a classic clustering approach, 'mean shift clustering' [5], [3], [4], which simultaneously and iteratively moves all segments in a parameter space to certain density modes. Segments ending in the same mode belong to the same cluster, and can be further processed to determine a single representative segment, see Figure 5c,d.

In practice, the mean shift merging approach needs a single parameter only. It defines the scale of the map, or, more intuitive, the minimum size of a significant detail (strictly spoken, there are three further parameters, the bandwith 'h', and kernel widths σ_x, σ_y. However, these are directly derived from the scaling parameter and turned out to be constant over all experiments). Since the scale of objects in robot mapping is known, this parameter can be determined once; the algorithm stays parameter free from there on. The length, location and direction of the line segments is automatically adjusted. Since mean shift clustering is a 'soft approach', no bin sizes or spatial/directional resolution must be defined.

The presented approach works in two phases, both utilizing mean shift clustering. Phase one detects clusters C_α of segments of similar direction α with the additional condition of mutual spatial nearness. Phase two computes sub-clusters of each C_α, joining nearly collinear segments.

A short introduction to mean shift clustering is given in section III. Section IV explains the merging process, experiments and results are reported in section V.

II. Related Work

Segment extraction as well as segment merging are classic problems in computer vision. [18] gives a thorough overview, especially examining three different methods. The first method is closely related to our method in the sense that it computes line segments in a single scan, and then merges it with a set of previously computed segments. In contrast to

Rolf Lakaemper is with the Department of Computer and Information Science, Temple University, Philadelphia, USA lakamper@temple.edu

our method, this method is incremental, while we perform simultaneous off-line merging. As [18] reports for incremental methods: "*The disadvantage of such approaches is that it is harder to detect and filter out dynamic aspects such as beams reflected by people walking by and outliers*". The advantage of incremental methods is, that they can be used on-line to assist e.g. the alignment process. The incremental method in [18] has a rather simple merging step, which traces back the point-data of segments to merge, it is therefore not entirely segment based. In contrast, in our approach the point data can be omitted as soon as the pre-segments in single scans are computed. The second, off-line approach in [18] is an advanced version of the Hough Transform [7]. It operates on the entire point set, and therefore does not take advantage of the partially ordered data points. The third technique in [18] is a widely accepted line fitting approach, Expectation Maximization (EM). EM suffers from multiple problems: it tends to converge to local minima if insufficiently initialized. Additionally, the model has to be known to a certain extent, i.e. the number of line segments. Additionally, it operates on the entire point set, which reduces the runtime performance. A modified EM version for line fitting is described in [12]. It is able to find line segments in unordered point sets with high robustness with respect to the initial model. It is a more general approach, with the drawback of multiple parameters and slower performance.

[14] compares different line extraction algorithms for 2D. In contrast to the presented approaches, we only extract lines in a single scan (any of the presented methods in [14] could be used for that); the main contribution of this paper is the clustering and merging of segments. All of the methods in [14] aim to extract line segments from arbitrary point sets, which in an off-line setting can be arbitrarily large. Using the information of point-order in single scans, we drastically reduce the runtime, since we convert each single scan to segments immediately. Hence the raw data points are processed in $O(n)$, the follow up processes work on the set of segments.

[1] presents a method for line segment reduction in maps. The technique is based on the notion of a matching chain, which describes a set of segments connected by a geometrically simple 'matching' relation. This method is critical, since dissimilarities accumulate and transfer through the set of the matching chain. In data sets with ambiguous matching relation, like the 'spiral' example (see Figure 2), such an approach would lead to over-merging, i.e. merging segments belonging to different underlying subsequent sections of the spiral. The merging methods looked at in [1] are additionally closely linked with the scan alignment process, i.e. they are designed for online merging purposes.

Our approach of mean shift clustering is simple, robust and fast: mean shift is simple to implement and takes no input parameters. Robust: since the mean shift clustering process works on all data points simultaneously, the effect of outliers is reduced. Clustering is done with respect to modes of data density, which is an averaged property of a certain neighborhood. Fast: the approach takes advantage of the setting of robot mapping; in each single scan the order of points is known. This makes the pre-processing step, i.e. detecting segments in single scans, a fast $O(n)$ task. The runtime order of mean shift is $O(k\ log(k))$, k=number of segments. k is usually significantly smaller than n.

In contrast to online processes, our approach is meant as a post-processing step after scan alignment, it is an off-line process which does not aid e.g. in the alignment process. It is designed to translate data of successful yet point based alignment algorithms like ICP-based approaches [16], [2], [13] or Particle Filter based approaches [6]; especially it can be used to simplify the result of already segment based alignment approaches like FFS [11].

III. Mean Shift

Mean shift, first described in [5], generalized in [3] and made further approachable for Computer Vision applications by [4], is a gradient descent procedure to find multiple modes of density distributions. It iteratively shifts data points to their mean in a certain neighborhood. Using mean shift as a clustering method, data points converging to the same mode belong to one cluster. The main advantage and, at the same time a drawback of mean shift is that, except for the definition of the neighborhood, it is parameter free. This includes that neither the number of clusters nor the step-width of the gradient steps have to be determined. The drawback is, that mean shift is known to be very sensitive to the selection of the kernel, which determines the degree of local vs. global operation. Therefore, using mean shift must be a careful consideration. In our case, domain knowledge is available to determine the kernel for robust operation. The domain knowledge consists of the size of significant elements in the environment.

[3] broke the ground for mean shift, showing its mathematical foundation, proving convergence and showing example applications. The following will give a short introduction to mean shift.

For data points $p \in P \subset \mathbb{R}^n$, the sample mean $m(x)$ with kernel K at $x \in \mathbb{R}^n$ is defined as

$$m(x) = \frac{\sum_P K(p-x)p}{\sum_P K(p-x)} \tag{1}$$

In our approach, we utilize a multivariate Gauss kernel, see section IV. Mean shift denotes the difference $\Delta m = m(x) - x$. For a finite set $\{t_i\} = T \subset \mathbb{R}^n$, the *cluster centers*, the algorithm iteratively performs steps $t_i \rightarrow m(t_i)$ simultaneously for all t_i until $max(\Delta m(t_i))$ is sufficiently small, i.e. the algorithm converged. For each t_i the mean $m(t_i)$ (equation 1 with $x = t_i$) is computed simultaneously, before all t_i are replaced by $m(t_i)$.

[3] describes two versions of mean shift, *blurring* and *non blurring*. In both versions, the point set T^0 ($= T$ at iteration 0) is initialized to $T^0 = P$. In the blurring version, P is iteratively set (blurred) to $P \leftarrow T^{j-1}$ at iteration j, i.e. the computation of $\Delta m(t_i)$ is based on the previous iteration result T^{j-1}. In contrast, the non-blurring version keeps the set P fixed, i.e. $\Delta m(t_i)$ is computed using the

original data set P. Our merging procedure follows the non-blurring approach. In this approach and with a Gauss kernel K, the mean shift Δm is in the gradient direction of the *density estimate* $q(x) = \sum_P K(p - x)$ using the same kernel K. Hence mean shift is steepest gradient ascent to the modes of the density distribution of P (with K) *without explicit computation of the density*. [3] proves that the step size is well-adjusted, i.e. no overshoots are possible. Under the non-blurring assumption (P does not change, and therefore its density does not change either), this leads directly to convergence of the algorithm: each individual point $t_i \in T$ climbs until it reaches a mode. The ordered set $\mathcal{T}_i = (t_i^0, .., t_i^r)$ point t_i at iterations $0..r$ is called the *trajectory* of t_i. Figure 2,a,b, shows examples of trajectories.

As mentioned before, mean shift mode seeking can be interpreted as clustering: since T was initialized using P, it is possible to assign all points p_i the same label if the corresponding t_i converged to the same mode. The mode represents the cluster center, or the location of locally highest density in P. The design of the kernel, i.e. the standard deviation σ_n for each dimension n is an important factor. It designs the neighborhood for the mean shift process and is responsible for the number of modes. The two extremes are a broad kernel ($\sigma_n = \infty \, \forall n$), which leads to a single mode, and the zero-radius kernel ($\sigma_n \to 0 \, \forall n$), which makes each data point p_i a density mode. Careful choice of the kernel is of paramount importance for successful application of mean shift. Section IV will show the kernel definition for segment merging.

The runtime for mean shift is (brute force implementation) $O(n^2)$ in the number n of line segments. It can be easily reduced to $O(nlog(n))$ using appropriate data structures (e.g. KD-trees). Pre processing single scans to gain line segments is an $O(n)$ process, n=number of data points.

IV. Segment Merging using Mean Shift

The basic motivation of segment merging using clustering is the simple idea to combine spatially close segments with similar directions in a single cluster. The clustering process is performed in two separate phases, determining locally common directions (phase 1) and collinearity of segments (phase 2). The clustering is followed by the actual merging process, combining segments of a single cluster.

A. Data Representation: Center-Direction Joint Space

Let $S = \{s_1, .., s_m\}$ be a set of m 2-dimensional line segments. We represent each s_i in the 3-dimensional *center-direction joint space*,

$$s_i \to p_i = (x, y, \alpha) \in \mathbb{R} \times \mathbb{R} \times [0..\pi] \qquad (2)$$

where (x, y) is the center of s_i. α is the direction of s_i, i.e the angle between s_i and the x-axis (mod π, i.e. heading independent). In the following we denote for $p = (x, y, \alpha)$: $p^{[1:2]} = (x, y)$ and $p^{[3]} = \alpha$.

The segments are gained from a pre-processing step: in each single scan, segments are created using the approach of [8]. Finding segments in single scans is a simple task, since the order of scan points is known. S is the set union of all segments gained from all single scans.

Segments s_i exceeding a certain length are first split into smaller subsegments (for ease of notation we denote this set also with $S = \{s_i\}$). The center-points and directions of these subsegments form the initial data set $P = T^0$ for mean shift clustering process. The length constraint is directly derived from the kernel design (related to σ) and does not impose a new parameter. This step guarantees that the center point is an appropriate location description for s_i and that short and long segments are likewise represented by locally near-uniform distributed center points. The locally near uniform distribution justifies the geometric meaning of the density modes to be cluster centers. Additionally, segment shortening renders unnecessary the introduction of weights to represent significant differences in length. Figure 1 shows a segment set and its representation in center-direction joint space. Please keep in mind that α is to interpret mod π, i.e. $0 = \pi$, hence the illustration is topologically inaccurate in the third dimension $p^{[3]}$. The center-direction joint space combines perceptually different segment features, spatial location and direction. In our approach, we account for this difference by linear scaling of the direction value relative to the spatial ($p^{[1:2]}$) values: for robot mapping in real world scenes, we assume that a spatial difference of $||p^{[1:2]}|| = 20$cm relates to a directional difference of 10 degrees. This means, we are using a certain constant bandwidth h to scale the direction dimension, see equation 4.

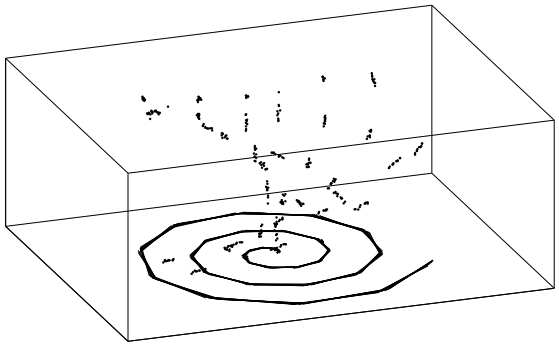


Fig. 1. Center-direction joint space of data set 'Spiral', with 300 points representing the 300 segments (at bottom). The data set 'Spiral', shown in Figure 2c,d, consists of 10 superimposed spirals with 30 segments each. The 10 spirals are slightly rotated and translated relative to each other.

B. Phase 1: Detection of Locally Common Directions

Goal of this phase is to cluster segments by their direction and spatial region. Emphasis in this phase is on separation of segments of sufficiently different direction, and consolidation of segments of similar direction. This phase is meant to learn sets of common directions of spatially near segments. We use a symmetric Gauss kernel (same $\sigma_n = \sigma \in \mathbb{R}^+$ for all dimensions n). σ is the *only* parameter in the entire system. It depends on the scale of the data. Defining σ once based on the scale of the data

(i.e., based on the minimal unit-size of a significant feature, in our case 20 cm), σ can be determined once: the system stays parameter free for environments of identical scale. Looking at the spatial dimensions $p^{[1:2]}$ only, we define symmetric neighborhoods which are not biased towards any direction in the spatial distribution of the segments' center-points. This approach is geometrically motivated as follows: since we do not assume any expected directions a priori, we do not prefer collinearity to the weaker condition of parallelism. This is the main difference between the first and second phase. The influence of spatial distance and distance in direction is governed by the aforementioned bandwidth h. Taking into account the kernel symmetry, the bandwidth h and the mod computation in α, we define the kernel $K_1(x)$ (index 1 for phase 1) for equation 1 by:

$$K_1(x) = K_1(x^{[1:2]}, x^{[3]}) = e^{\frac{d(x)^2}{2\sigma^2}} \tag{3}$$

with distance d

$$d(x) = \sqrt{||x^{[1:2]}||^2 + \left(\frac{\min(|x^{[3]}|, \pi - |x^{[3]}|)}{h}\right)^2} \tag{4}$$

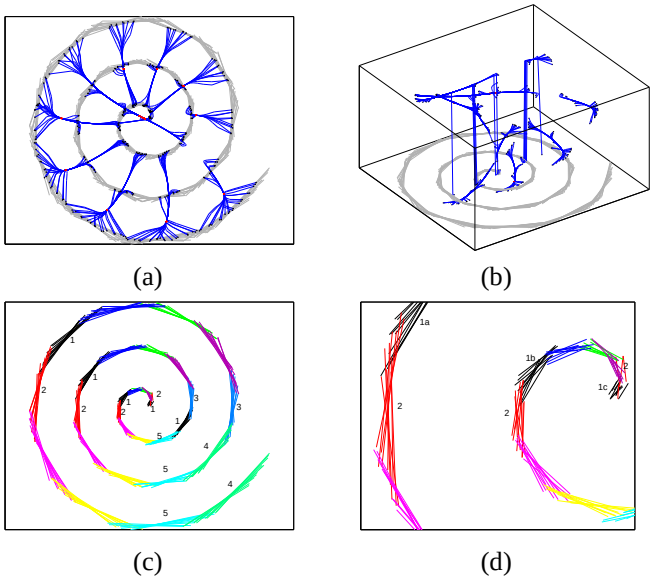


Fig. 2. Result of phase 1. a) 2D (x,y) view. blue: Trajectories $\mathcal{T}_i$, black points: T^0, red points: $\hat{T}$. b) same as (a), but 3D x, y, α view. The vertical jumps in the display of the trajectories are a result of the topology (mod π) of the center-direction space. c) labeled segment clusters. Same color displays same cluster (some example clusters are numbered). d) magnified view of center of (c). Segments 1a, 1b, 1c still belong to the same cluster. They will be separated in phase 2.

Starting with the set of points $\{t_i\} = T^0 = P$ representing the (split) segments s_i, we apply the mean shift clustering using kernel K_1 until it converges, forming the final point set $\hat{T}$. A cluster C_l in $\hat{T}$ consists of points at the same mode. We assign the label $l(s_i)$ to the corresponding segments, i.e. $l(s_i) = l(s_j) \Leftrightarrow d(t_i - t_j) < \epsilon$, with $d(.)$ being the distance measure defined in equation 4. Figure 2,a,b, shows the initial point set, the trajectories $\mathcal{T}_i$ and the final point set of the 'spiral' example. Figure 2,c,d, shows the resulting segment clusters. Segments of similar direction and spatial neighborhood are clustered. Please note that clustering by spatial neighborhood and direction is different than just (1-dimensional) clustering by direction alone: it allows segments to differ more strongly by direction if they are spatially close. Vice versa, it allows for distinction of smaller directional distances if segments are located in different neighborhoods, which is in accord with human perception. As an example, see Figure 2,d: a few segments in cluster 1 and 2 (black/red) share approximately the same direction, but are part of different clusters.

C. Phase 2: Clustering by Collinearity

Phase 1 determines locally common directions, but can not distinguish between collinear and non-collinear (both approximately), e.g. segments $1a, 1b, 1c$ in Figure 2,d. Phase 2 aims to sub-cluster each phase 1-cluster using the information of the cluster's direction: only collinear segments are merged. For simplicity of notation, we denote clusters in $\hat{T}$ and their corresponding clusters in S by the same symbol C_l.

We define the *direction of a cluster* as the average direction $\theta = t_i^{[3]}$ of any corresponding point $t_i \in C_l$ (as a result of phase 1, all points have approx. the same direction). The cluster's direction is the only information we keep from phase 1, the clusters' locations $\hat{T}^{[1:2]}$ are omitted. Omitting the location information is the price we pay for higher robustness: in the second phase, the locations are re-computed but with given direction. Re-computation does not increase the order of magnitude of the algorithm, but doubles (in the worst case of a single cluster) the processing time. Since all segments in one cluster are now assigned the same direction, it is sufficient to represent the segments s_i in a 2-dimensional space: phase 2 mean shift clustering operates in a 2-dimensional space, defining a single cluster's points' location. We initialize the process by $T^0 = P^{[1:2]}$.

We design an anisotropic kernel K_2. Elongated in the cluster's direction, and narrowed in the perpendicular direction, it aims to merge collinear segments only. It emphasizes the density in direction θ; each mean shift gradient ascent therefore favors gravitation between center points belonging to collinear segments. We define the standard deviation σ_x, σ_y for the kernel based on σ defined in phase 1. In our implementation, we fix $\sigma_x = 2\sigma$ and $\sigma_y = \sigma/10$. With cluster direction $\theta = t_i^{[3]}$, the 2-dimensional Gauss kernel K_2 is defined by

$$K_2(x) = e^{-\frac{1}{2}x^T \Sigma x} \tag{5}$$

with covariance matrix

$$\Sigma = \begin{bmatrix} a & b \\ b & c \end{bmatrix} \tag{6}$$

with $a = \frac{\cos^2(\theta)}{2\sigma_x^2} + \frac{\sin^2(\theta)}{2\sigma_y^2}$, $b = \frac{-\sin(2\theta)}{4\sigma_x^2} + \frac{\sin(2\theta)}{4\sigma_y^2}$, $c = \frac{\sin^2(\theta)}{2\sigma_x^2} + \frac{\cos^2(\theta)}{2\sigma_y^2}$

Remark: Splitting the process into 2 phases makes the approach more robust. However, for 'simple' data sets the

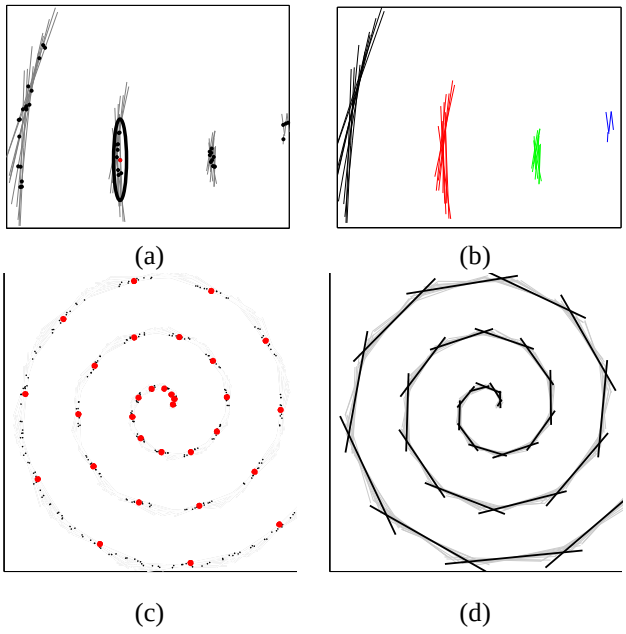


Fig. 3. Phase 2 and segment merging. a) Segments of single cluster from first phase with center points. The ellipse illustrates the anisotropic kernel K_2, which is elongated in the average cluster segment direction. b) result of phase 2: sub-clustered segments. Different colors are different sub-clusters. c) Cluster centers (density modes) of all sub clusters. d) Result set $\mathcal{M}$ of merging process. The 30 segments correspond to the 30 modes of (c). The 'overshooting' at segment intersections is caused by the original segment distribution, see Figure 2d. It can easily be removed by a post processing step.

phases can be combined: in order to compute $\Delta m(t_i)$, replace the average direction θ with the ith segment's direction $p_i^{[3]}$ at iteration 0, which is $p_i^{[3]}$. Use a kernel with covariance matrix $\Sigma = \begin{bmatrix} a & b & 0 \\ b & c & 0 \\ 0 & 0 & \frac{1}{\sigma^2 h^2} \end{bmatrix}$ (a, b, c, h, σ defined as above) in phase 1 (phase 2 is then omitted). In this case, the neighborhood of each point t_i is defined by a different kernel $K(t_i)$. Since we use the non-blurring approach, $K(t_i)$ stays constant over the iterations. Hence the convergence is still guaranteed. This single phase approach is less robust since it uses directions of single segments, instead of average directions of sets of segments. Hence, if phase 1 of the 2-phase clustering would lead to clusters with low intra cluster direction deviation of segments, the data set is 'simple', a one phase approach could be performed. The 2 phase approach is more robust since in the second phase, parallelism can be assumed. This strong assumption makes it easier to model a 'slimmer' directed kernel K_2, i.e. $\sigma_x >> \sigma_y$.

D. Segment Merging

Having the clusters computed, the actual merging is a simple, straightforward geometric procedure. Phase 2 splits each cluster C_l into sub-clusters $C_l^{[1]}, .., C_l^{[k_l]}$. The corresponding modes $M_l^{[1]}, .., M_l^{[k_l]}$ are points of single lines $R_l^{[1]}, .., R_l^{[k_l]}$, the cluster representative lines: $R_l^{[j]}$ is defined by $M_l^{[j]}$ and the cluster's direction θ. To compute the final result of the merging process, we project all segments s_i of a single sub-cluster $C_l^{[j]}$ onto $R_l^{[j]}$ and merge overlapping segments to a single segment. Figure 4 illustrates this process. The set $\mathcal{M}$ of all (merged) projected segments is the result of the segment merging phase, see e.g. Figure 3c,d and results in section V.

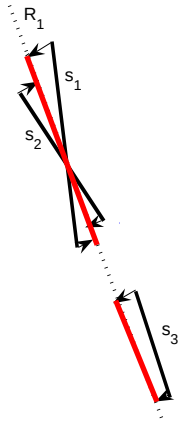


Fig. 4. Merging process. The segments s_1, s_2, s_3 are projected onto the cluster-representative line R_1. Overlapping segments (s_1, s_2) are merged. The result of the merging process are the (merged) projected red segments.

V. Experiments and Results

A. Hallway

The data set, originating from [14], consists of 100 scans of 722 scan points each. After removal of invalid scan points, the data set contains 59245 points, see Figure 6a. The algorithm of [8] is utilized to compute segments for each scan. It connects the first and last point in a sequence of sufficiently collinear points. To improve the result, we additionally perform a least square fit with each segment. Figure 5 shows an example of segments extracted from a single scan and segments being superimposed. The segment set S gained from single scan segment extraction consists of 1654 segments, see Figure 6a, or the detail in Figure 5. S is the input to our mean shift process. Phase 1 identifies 495 clusters. Phase 2 splits these clusters to a total of 525. The merging result $\mathcal{M}$, 525 merged segments, is shown in Figure 6b. In a post processing step we erase small segments with insufficient scan point support (density too low). This cleans $\mathcal{M}$ to a final result of 255 segments, Figure 6c. To evaluate the quality of the segment merging, we compare the average distance of the scan point set X to the input S and the output $\mathcal{M}$ of the merging algorithm. We define the distance $D(x, S)$ between a point x and a set S of segments by $\min_{s \in S}(d(x, s))$, with $d(x, s)$ denoting the shortest distance between point x and a single segment $s \in S$. Since the process of extracting segments from single scans has a filter which erased small segments, certain outlier points in X have a high distance to the nearest segment. We therefore erase points outside of 2 standard deviations of the minimum distances and recalculate the mean distance on the inlier point set X'. The average distance $D_S = \text{mean}_{x \in X'}(D(x, S))$ for the input set S is $D_S = 0.0050$ meters. For the output $\mathcal{M}$,

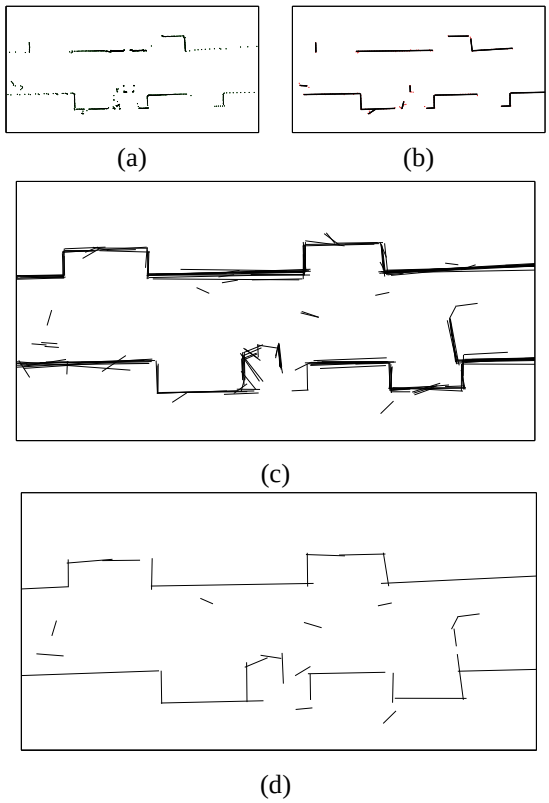


Fig. 5. Detail from data set 'Hallway' (see Figure 6). a) points from single scan b) extracted segments, single scan c) segments of superimposed scans d) result of scan merging in same area.

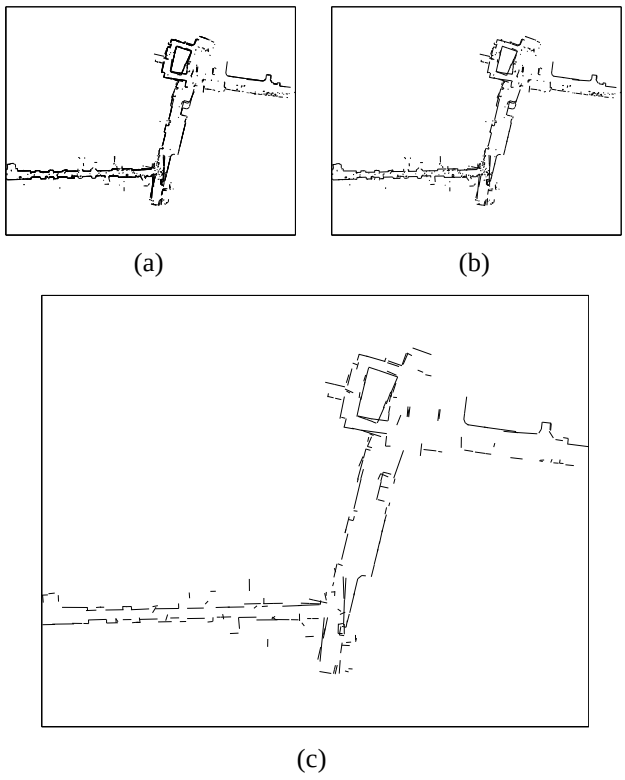


Fig. 6. Data set 'Hallway'(100 scans). a)Input to merging process: 1654 segments. These segments were generated from single scans. Gray: underlying points (59245 points) b) Result of merging process $\mathcal{M}$: 525 Segments. c) post processed, short segments removed: 255 segments. Compression rate between a and c is 1:116. Compare also Figure 5

the average distance $D_{\mathcal{M}} = \text{mean}_{x \in X'}(D(x, \mathcal{M}))$ increases to $D_{\mathcal{M}} = 0.0064$ meters. As Figure 5 shows, details like small corners of door entrances are preserved. Comparison of Figure 6 (a) and (d) illustrate that the overall appearance of the map did not change significantly. However, a data compression of 1:116 is achieved. In addition, the segment data structure allows for simpler computation of higher level geometric features.

To compare different line extraction algorithms, [14] uses a set of manually created "truth lines". Unfortunately, we did not have the data set to compare our results. However, [14] reports 679 truth-lines. Our merging step (before cleaning) results in 525 segments. The (better) lower number of our approach results from creation of truth lines in single scans in [14]. Since visual inspection (e.g. Figure 5,(d) or the accompanying video) does not show over-merging, the resulting 525 lines seem consistent with the underlying data set. With an order of magnitude of $O(n) + O(k\ log(k))$ our approach is faster than all compared algorithms in [14]. However, such a comparison has to be taken with a grain of salt, since, as mentioned before, the 6 algorithms in [14] are differently motivated than ours.

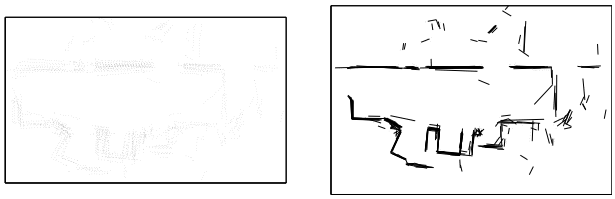
Please have in mind that the merging step is neither responsible for the alignment errors in the underlying data set, nor is it designed to solve the alignment problems.

B. Laser on a Stick: Low Cost Low Effort Scanning - NIST Response Robot Evaluation Exercise

The data for this experiment was collected during the organized Response Robot Evaluation Exercise organized by NIST (National Institute of Standards and Technology) in Texas, November 2009 ([15]). The exercise was located in 'Disaster City'. Disaster City is a TEEX (Texas Engineering Extension Service, a part of Texas A&M University, College Station) facility. It offers multiple sites, for example collapsed buildings, trains etc. for simulated disaster environment training. During the exercise, different sites were scanned with different kinds of equipment (2D scanners, 3D scanners, cameras etc.) to evaluate the performance of sensors and algorithms and to demonstrate their usability to first responders. The purpose of this specific experiment was to gain experiences with extremely cost efficient equipment. The equipment consisted of a single Hokuyo URG-04LX 2D laser scanner (< 2500 US$) mounted on a stick, see Fig. 7. This device was carried through the environment by a human first-responder to retrieve scans in a 'step and shoot' manner. No additional sensors were used. The executing responder was instructed to keep a constant step-length, and to trigger three single horizontal scans ($90°$ left, walking direction, $90°$ right) at each step at knee height. A photo of the scanned area (Disaster city location 'House of Pancakes') can be seen in Figure 7. The result of the superimposed 27 raw scans

Fig. 7. Retrieving data in Disaster City. Photo of the scanned environment (the wall to the left appears at the top in Figure 8, the drawer (right) appears on the bottom of 8). The foreground shows the simple scan-tool, a stick-mounted laser.

is shown in Figure 8. The superimposition assumes a predetermined constant step length ($93cm$) of the responder. The 27 scans were aligned using a technique described in [10]. Expectedly, the alignment can not show a consistent map of the environment: since angular errors are unavoidable in the retrieval process (human estimated, error-prone pitch, roll and yaw of the laser), the imprecise and inconsistent raw data lead to representation of single features by multiple line segments. Figure 9 shows the result of the segment merging



(a) (b)

Fig. 8. Result of scanning the environment shown in photo Figure 7(c). a) 27 scans superimposed using the estimated odometry (human step length). b) Result of alignment

algorithm, based on the input shown in Figure 8b. The result consists of < 25 line segments. The compact representation of the environment is extremely useful in disaster scenarios: data like this can be reliably transmitted even to hand held devices for the responders; high redundancy, which might be necessary in disaster scenarios, can be added due to the low data volume.

VI. Conclusion and Outlook

We presented a merging technique based on mean shift clustering to represent robot maps by line segments. Being scaled once, the approach is parameter free. It takes advantage of the simplicity of generating line segments from single scans, and to merge these in global maps based on spatial nearness and collinearity. It is sufficiently direction sensitive to represent approximations to round structures (spiral). Experiments with real data showed its applicability to robot mapping.

VII. Acknowledgements

We would like to thank Raj Madhavan, Oak Ridge National Labs, Chris Scrapper, Adam Jacoff and Elena Messina (all NIST, Gaithersburg, MD) for their great and hard work of organizing and building the 2008 Response-Robot-Exercise experience. We also want to thank Steve Richards, Acroname inc., for helping to collect the data set Figure 8.

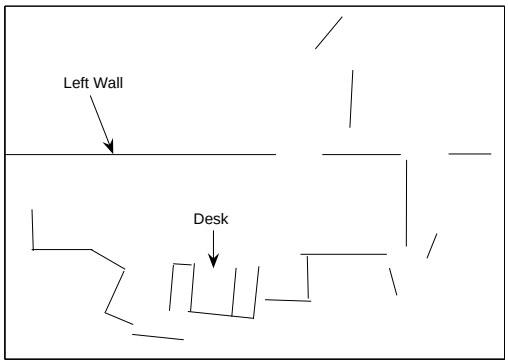


Fig. 9. Result $\mathcal{M}$ of merging segments S of Figure 8b. Compare the image with photo Figure 7. The wall on the left and the desk to the right are easily detectable. The result consists of 23 segments and can easily be submitted even with limited equipment, e.g. hand held devices.

References

[1] F. Amigoni and S. Gasparini. Analysis of methods for reducing line segments in maps: Towards a general approach. In *Proceedings of IEEE/RSJ 2008 International Conference on Robotics and Systems (IROS 2008)*, pages 2896–2901. IEEE Press, 2008.

[2] P. Besl and N. McKay. A method for registration of 3.d shapes. *IEEE PAMI, 14(2)*, 1992.

[3] Y. Cheng. Mean shift, mode seeking, and clustering. *Pattern Analysis and Machine Intelligence*, 17(8):790–799, August 1995.

[4] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[5] K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *Information Theory, IEEE Transactions on*, 21(1):32–40, Jan 1975.

[6] G. Grisetti, C. Stachniss, and W. Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. *ICRA*, 2005.

[7] P. V. C. Hough. Methods and means for recognizing complex patterns. *US patent 3,069,654*, 1962.

[8] P. D. Kovesi. MATLAB and Octave functions for computer vision and image processing. School of Computer Science & Software Engineering, The University of Western Australia. Available from: <http://www.csse.uwa.edu.au/∼pk/research/matlabfns/>.

[9] D. Lagunovsky. Ablameyko s. fast line and rectangle detection by clustering and grouping. *Proc. of CAIP'97, Kiel, Germany*, 1997.

[10] R. Lakaemper and N. Adluru. Improving sparse laser scan alignment with virtual scans. In *IROS*, pages 2915–2921, 2008.

[11] R. Lakaemper, N. Adluru, L. Latecki, and R. Madhavan. Multi robot mapping using force field simulation. *Journal of Field Robotics, Special Issue on Quantitative Performance Evaluation of Robotic and Intelligent Systems*, 2007.

[12] R. Lakaemper and L. Latecki. Using extended em to segment planar structures in 3d. *18th Int. Conf. on Pattern Recognition (ICPR), Hong Kong*.

[13] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Auton. Robots, 4(4)*, pages 333–349, 1997.

[14] V. Nguyen, S. Gächter, A. Martinelli, N. Tomatis, and R. Siegwart. A comparison of line extraction algorithms using 2d range data for indoor mobile robotics. *Auton. Robots*, 23(2):97–111, 2007.

[15] NIST. Response robot evaluation exercise. Website: http://www.teex.com.

[16] A. Nuechter. *3D Robotic Mapping*. Springer Tracts in Advanced Robotics (STAR). Springer Verlag, 2009.

[17] J. Poppinga, N. Vaskevicius, A. Birk, and K. Pathak. Fast plane detection and polygonalization in noisy 3d range images. *International Conference on Intelligent Robots and Systems (IROS)*, 2008.

[18] D. Sack and W. Burgard. A comparison of methods for line extraction from range data. In *Proc. of the 5th IFAC Symposium on Intelligent Autonomous Vehicles (IAV)*, 2003.