M3RSM: Many-to-Many Multi-Resolution Scan Matching

Edwin Olson[1]

***Abstract*— We describe a new multi-resolution scan matching method that makes exhaustive (and thus local-minimum-proof) matching practical, even for large positional uncertainties. Unlike earlier multi-resolution methods, in which putative matches at low-resolutions can lead the matcher to an incorrect solution, our method generates *exactly* the same answer as a brute-force full-resolution method. We provide a proof of this. Novelly, our method allows decimation of both the look-up table *and* in the point cloud, yielding a 10x speedup versus contemporary correlative methods.**

When a robot closes a large-scale loop, it must often consider many loop-closure candidates. In this paper, we describe an approach for posing a scan matching query over these candidates *jointly*, finding the best match(es) between a particular pose and a set of candidate poses ("one-to-many"), or the best match between two sets of poses ("many-to-many"). This mode of operation finds the first loop closure as much as 45x faster than traditional "one-to-one" scan matching.

I. Introduction

A critical step in virtually all modern mapping systems is to identify places that a robot has visited more than once. Such a "loop closure" is the primary mechanism through which Simultaneous Localization and Mapping (SLAM) systems prevent the unbounded accumulation of error.

A typical approach with laser range-finder data is to compute the posterior uncertainty of the robot's current position (usually in the form of a covariance ellipse) and then to construct the set of previously-visited places that fall within that ellipse. A "scan matcher" is then used to attempt to align the robot's current laser scan with each historical scan in the set. If there are H historical poses, H scan matching operations are performed, and H possible alignments (or "loop closure hypotheses") result. Most of these are wrong: the scan matcher tries to find the best possible alignment between two laser scans, but it is often possible to find a plausible match between scans acquired at distinct places.

Additional validation of these scan-matcher loop closures candidates is part of virtually every practical system, either through an explicit validation system or through robust optimization methods that intrinsically reject incorrect loop closures [18], [13]. In this paper, however, we will focus on the *generation* of these hypotheses. Further, we will focus on the challenging case where the positional uncertainty is very large (e.g., large loop closures and kidnapped robots), though our approach also applies to scenarios with smaller uncertainties (e.g. "laser odometry" and environments in which loop closures are relatively common).

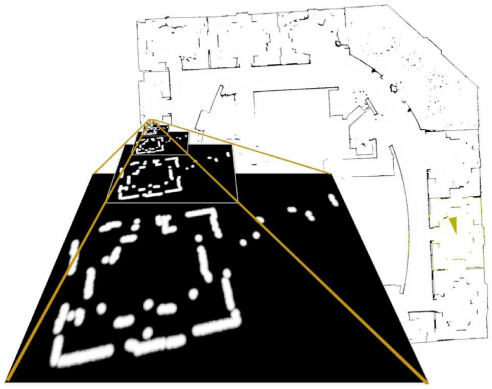

Fig. 1. Multi-resolution image pyramid. A high resolution cost function (bottom) is successively reduced in dimension forming an image pyramid. By constructing the pyramid carefully, alignment scores computed using low-resolution levels can be used to prune the search space, leading to large speedups. The region from which the cost function was generated is visible in the map behind (green).

This matching process is computationally expensive, and can be an even greater bottleneck than the SLAM optimization itself. In our multi-robot mapping system, over a dozen robots explored an area simultaneously [14]. The robots operated largely independently (so as to explore the environment as quickly as possible) but thus quickly accumulated large positional uncertainties with respect to their teammates. Each robot's position estimate could overlap dozens or hundreds of historical poses of its teammates, and new poses arrived about every second. In short, it is reasonable to want to examine hundreds of candidate alignments per second.

Because of the large positional uncertainties that arise in loop-closing scenarios, iterative local optimization methods like ICP often become stuck in local minima— failing to produce reasonable alignments even when the scans *do* overlap. Unfortunately, scan matching methods (which are desirable due to being immune to these local minima), can be impractically slow when the search windows are large.

This paper makes several contributions towards these problems:

- We describe a multi-resolution matcher (an improvement over earlier 2-level matchers) that produces identical results to brute-force methods; this method achieves significant speed-ups versus 2-level methods.
- We show how matching can be further accelerated by decimating not only the look-up table (as above), but also decimating the query points. This provides addi-

*This work was funded by the DARPA YFA program (D13AP00059).

[1]Edwin Olson is Associate Professor of Computer Science and Engineering at the University of Michigan, 2260 Hayward St, Ann Arbor MI ebolson@umich.edu

tional speedups, while still maintaining the correctness guarantee.
- We introduce the idea of "many-to-one" and "many-to-many" matching, where multiple queries are posed simultaneously, and show that our scan matching approach can solve these much faster than examining these queries individually.

II. Related Work

Many common approaches to scan matching are variants on Iterative Closest Point (ICP) [1], [11], [20] and Iterative Closest Line (ICL) [5], [2], [3]. In these approaches, correspondences between two scans are explicitly computed, allowing a rigid-body transformation to be computed [8]. A major disadvantage of these methods is their susceptibility to local minima; poor initial estimates lead to incorrect data associations and divergence [12].

Correlation based methods [10], [19] search for a rigid-body transformation (without computing correspondences) that projects one set of LIDAR points (the *query* scan) on top of a reference map. This reference map may be known in advance (a *localization* problem), or may be derived from a LIDAR scan earlier in the robot's trajectory (a *SLAM* problem). The reference map is generally implemented as a look-up table that assigns a cost for every projected query point. Commonly, LIDAR points from the reference scan are rendered into this cost map using a blurry kernel that approximates the uncertainty of the laser data. Our previous work [12] examined correlative matching, both with a Graphics Processing Unit (GPU) and with a two-level multi resolution method. We showed that scan matching could be made fast enough for use in a laser odometry system, but processing time for loop closures was still problematic.

2D laser range data, the focus of this paper, is often ambiguous: it is common to find distinct places whose laser range data appears to match. In contrast, vision-based methods like FAB-MAP [6] exploit the variation in appearance between places in order to better distinguish them. However, these methods generally do not produce the precise metrical positioning information needed for many applications. In this sense, the proposed method is synergistic with visual feature methods in that they benefit from their culling of hypotheses but produce more precise positioning information.

Scan matching methods can also be applied to visual data, though additional complications arise due to the effects of perspective projection [15].

A wide variety of other scan matching approaches have been considered, including the use of polar coordinates [7], histogram-based methods [16], feature-based methods [2], and Hough transforms [4].

III. Approach

A. Correlative Scan Matching Overview

The basic approach to correlative scan matching is to find the translation and rotation that project a set of points into a "good" configuration. Configurations are scored according to a cost map (see Fig. 1) that is commonly generated from another scan. These cost maps often have a probabilistic interpretation: they approximate the log likelihood of observing a point at every location.

More concretely, suppose we are matching two scans taken from two locations. Suppose that A and B are 3×3 rigid-body transformation matrices that project points from the robot's coordinate frame into global coordinates. Let T be the transformation that projects points from B's coordinate system into A's coordinate system. In this case, the LIDAR scan acquired at position A is the "reference" scan, and the scan acquired at B is the "query" scan. We then have:

$$T = A^{-1}B \tag{1}$$

Suppose $L()$ is a function which takes a point (in A's coordinate frame) and returns a cost associated with observing a point at that position. Let us assume that N points are observed from position B, and write them as 3×1 homogeneous vectors p_i. Assuming independence of individual points, the cost associated with a particular transformation matrix T is the sum of the costs of each transformed point. The best transformation T^* is then:

$$T^* = \text{argmin}_T \sum_{i \in N} L(Tp_i) \tag{2}$$

The function $L()$ is usually implemented by rasterizing the points observed in coordinate frame A into a 2D look-up table (effectively an image). Points are rasterized with a "blurry" kernel that approximates the noise in the observed points. When this kernel is quadratic as a function of the distance from the observed point, it corresponds to a Gaussian noise model. Further, because the look-up table has finite precision, costs saturate (at 255, for example), effectively making the cost function robust.

Finding the best alignment T^* is conceptually simple: one can simply try a large number of transformations, evaluating the cost for each one. But the search space is three-dimensional, including translations in x and y, and rotations. As a result, the computational complexity can grow very rapidly with larger search windows.

B. Multi-Resolution Search

The basic idea of a multi-resolution scan matcher is to begin searching for T^* using a coarser grid of candidate transformations, and to use the results of this low-resolution search to inform a search at higher resolutions. This is advantageous because the search at low-resolution is much faster than the search at high resolution. The challenge is to ensure that the results of the low-resolution search are consistent with the optimal answer T^*.

In previous work, we used a two-level multi-resolution scheme in which the two levels differed in resolution by a factor of 10 [12]. In this work, we generalize this to a multi-level pyramid and a more flexible search strategy oriented around a heap data structure.

The heap contains search tasks for specific rotations, a range of translations to consider, and a resolution level. For each task, we compute an optimistic bound on the maximum

likelihood. The main search loop simply extracts the most promising task from the heap: if that task is at the highest resolution, the search is complete; otherwise, the task is expanded into child tasks, each of which are evaluated at the next higher resolution level. The search is effectively a best-first search [17], though the "path" through the space of T is a purely artificial construct whose sole purpose is to provide a tree-like partitioning of the search space.

The novelty of our approach is in deriving a method for bounding the cost over a range of values of T, allowing cost estimates to be computed for non-terminal nodes in the search tree. Further, our method computes these bounds quickly using a multi-resolution lookup table.

C. Computing optimistic estimates

Given a set of points p_i and a function $L()$ with finite support that maps integer arguments to real values, let us consider the problem of finding an integer-valued 1D translation t^* such that:

$$t^* = \text{argmax}_t \sum_i L(p_i + t - o) \tag{3}$$

The function of parameter o is to simplify the implementation of L as a look-up table using an array. Most languages only support positive array indices, and so the value o is set to be index of the first non-zero value of L.

Assuming that we will search over a large range of t, it is convenient to decompose this search into sub-searches, each covering a range of D values. Let us consider the search within a sub-region defined by an integer j:

$$t \in T = [Dj, Dj + D - 1] \tag{4}$$

In other words, we are considering a set of translations of size D that is "aligned" to an even multiple of D. Obviously, any search range for t can be enclosed by one or more such regions of size D.

We now construct a "low resolution" version of L, picking an offset $o_d \leq o$ such that $o_d = Dk$ for some integer k:

$$\begin{gathered} L_D(y) = \max_{x \in X} L(x), \\ X = [Dy + o_d - o, Dy + 2D - 2 + o_d - o] \end{gathered} \tag{5}$$

Claim: A single lookup in L_D can conservatively replace D lookups in L. Specifically:

$$\begin{gathered} \textstyle\sum_i L_D(\lfloor p_i/D \rfloor + j - o_d/D) \geq \max_{t \in T} \sum_i L(p_i + t - o) \\ T = [Dj, Dj + D - 1] \end{gathered} \tag{6}$$

Proof: First, the claim is true if the sum and max on the right hand side are interchanged: picking a different t for each point will surely result in at least as good a score as picking a single t for every point.

$$\begin{gathered} \textstyle\sum_i L_D(\lfloor p_i/D \rfloor + j - o_d/D) \geq \sum_i \max_{t \in T} L(p_i + t - o) \\ T = [Dj, Dj + D - 1] \end{gathered} \tag{7}$$

This bound holds if we drop the $\sum$ from both sides, thus requiring *every* term on the right hand side to be at least as large as the one on the left. We will now consider an arbitrary point p:

$$\begin{gathered} L_D(\lfloor p/D \rfloor + j - o_d/D) \geq \max_{t \in T} L(p + t - o) \\ T = [Dj, Dj + D - 1] \end{gathered} \tag{8}$$

Two simplifications: first, recall that $o_d/D = k$. Second, let us move the o into the set of T:

$$\begin{gathered} L_D(\lfloor p/D \rfloor + j - k) \geq \max_{t \in T} L(p + t), \\ T = [Dj - o, Dj + D - 1 - o] \end{gathered} \tag{9}$$

We will now compare the set of values used to compute $L_D(\lfloor p/D \rfloor + j - k)$ to the values of $L()$ appearing on the right hand side, showing that the left-most value used in the construction of $L_D(...)$ is at least as far left as the left-most value of $p + T$. (We must show that the same is true from the right.)

First, the left-hand side:

$$\begin{gathered} D(\lfloor p/D \rfloor + j - k) + o_d - o \leq p + Dj - o \\ D\lfloor p/D \rfloor + Dj - Dk + o_d - o \leq p + Dj - o \\ D\lfloor p/D \rfloor \leq p \end{gathered} \tag{10}$$

The final line is true due to the definition of the floor function. Now, the right-hand side:

$$\begin{gathered} D(\lfloor p/D \rfloor + j - k) + 2D - 2 + o_d - o \geq p + Dj + D - 1 - o \\ D\lfloor p/D \rfloor + Dj - Dk + 2D - 2 + o_d - o \geq p + Dj + D - 1 - o \\ D\lfloor p/D \rfloor + D - 1 \geq p \end{gathered} \tag{11}$$

The final line is true due to the definition of the floor function and the fact that p is an integer.

A major advantage of this formulation (and an improvement over our previous multi-resolution scan matching method) is that the coordinates of query points p only appear as part of a $\lfloor p/D \rfloor$ term. This property is what allows query points to be decimated: if two points (p_2 and p_7, for example) have $\lfloor p_2/D \rfloor == \lfloor p_7/D \rfloor$, then these points can be combined into a single lookup with the result of that lookup multiplied by two. The larger the parameter D, the more likely it is that multiple query points will collapse onto the same value of $\lfloor p/D \rfloor$.

Decimating the points makes this scan matching approach scale very well as the number of points in each scan increases (see Fig 2). Intuitively, this slow growth in computational complexity is due to the fact that the vast majority of tasks in the heap are for lower-resolution searches. Only very promising nodes need to be evaluated at full resolution.

D. Many-to-One

The dominant approach in graph-based mapping systems is to identify pairs of robot poses that *may* have overlapping views using the SLAM system's current position estimates and uncertainties. A scan matcher is then run on each of these pairs, one at a time, producing an alignment for each pair. We call this the "one-to-one" operating mode, because one laser scan is being matched to one other laser scan.

One-to-one matching is computationally expensive: it scales linearly with the number of scan matching attempts.

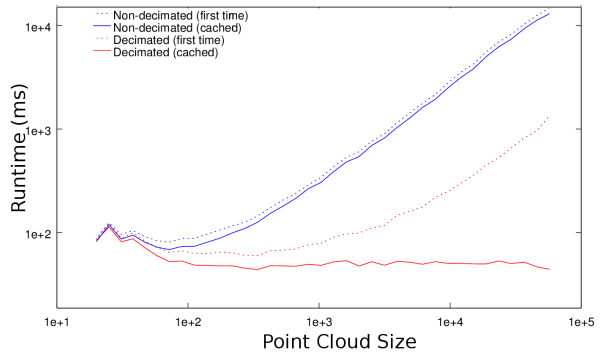

Fig. 2. Performance benefit of point decimation. Our method allows query points to be decimated with no impact on quality, resulting in substantial speedups. The first time a node is processed, there is an additional cost in rotating the query points (dashed red line). However, once these rotated points are cached, runtime becomes virtually independent of the number of points (solid red). In both cases, however, decimation provides a significant improvement. Interestingly, computational costs can actually increase for very small point clouds. We attribute this to greater matching ambiguity, causing more nodes to be expanded. Experimental settings: poses were sampled from co-visible "red" regions in Intel dataset, initial pose errors of up to $\pm$ 10 m and $\pm$ 180 degrees, and a search range of $\pm$ 15 m, $\pm$ 180 degrees @ 1 degree step size. Note log/log scale.

One-to-one matching also produces *too much* data. Most of the matches are wrong, and must be rejected. While one-to-one matching might identify multiple correct loop closures, it is the *first* loop closure that is most useful to a SLAM system. Once the first loop closure is found, the reduction in posterior uncertainty generally makes it easy to find additional matches.

One of the main contributions of this paper is a rethinking of the conventional one-to-one matching approach. Consider a robot that has just moved to a new position and acquired a new scan. We might now want to ask: what previous laser scans match the current scan? Based on our SLAM estimate, we might generate a list of S potential poses. Suppose that, instead of posing S separate scan matching queries, we ask a single one: "what is the best possible alignment between the current pose and *any* of the S other scans?" This is a "many-to-one" query, in that S scans are being matched to the robot's current scan.

Our heap-based scan matcher can be easily modified to accommodate "many-to-one" queries: we simply allow each node in the heap to belong to a different pair of scans. As the nodes in the heap are extracted and expanded, the search will automatically switch between all S candidate poses.

Critically, we will demonstrate that this "many-to-one" query requires *far* less computation time than the alternative set of one-to-one queries. This is due to the fact that non-competitive match hypotheses never get explored, whereas in one-to-one matching, each hypothesis is forced to generate at least one solution.

E. Many-to-Many Matching

The "many-to-many" approach is an additional improvement over the many-to-one approach. Because every match in the many-to-one approach references a common laser scan, the success of the operation hinges on the properties of that scan. That scan could be largely featureless due to the robot facing a wall, or it could simply be a rare view that has no good matches. In either case, when there are no good matches, a large number of heap nodes tend to be expanded.

The essential idea in many-to-many matching is to eliminate the dependence on one "special" node. This can be done in a variety of ways, though a simple approach is to randomly sample pairs of laser scans where additional loop closures are desired. (One could also imagine a system that computes the same set of S candidates as in the many-to-one case, but considers matches to the most recent k laser scans, leading to kS candidates.)

F. Practical Considerations

1) Cost table generation: Our approach is formulated in terms of a lookup table L. In our implementation, we generate this lookup table using raw LIDAR data. (An alternative might be to used an a priori map.) The resolution of this lookup table has an impact on the overall accuracy of the scan matcher. In this paper, we use a resolution of 1/32 m. See Fig. 1 for an example of one of the cost surfaces.

The raw points from two LIDAR scans of the same environment often do not align well. First, noise in individual observations makes it unlikely that two points will align exactly. A straight-forward way of dealing with this is generate low costs for "near misses", i.e., by rendering a "fuzzy ball" into the lookup table, rather than a single low-cost point. In this paper, we render quadratic surfaces that saturate at a cost of 255 at a distance of 0.1 m from the observed LIDAR point.

A second problem is that LIDAR scans of the same environment, taken from different locations, typically sample the position of walls at different points along the wall. In general, these sample points will not line up. Our approach for dealing with this is to extrapolate lines between LIDAR returns in cases where it is reasonably likely that a solid surface exists. In this paper, we do this using a simple distance heuristic: if two points are within 1 m of each other, we extrapolate a line segment between them.

2) Image pyramid generation: The previous section describes, in general, how a low-resolution version of a cost-lookup table can be constructed. Scan matching alignments queried against this low-resolution cost function always score at least as well as any higher-resolution query. In this section, we describe how this basic idea is exploited in a complete scan matching system.

Our implementation pre-computes low-resolution versions of every lookup table. We begin with the highest-resolution lookup table and successively compute versions with half the resolution. This process repeats, computing quarter-resolution, eighth resolution, etc., until the entire original lookup table is represented with a single pixel.

Equation 5 gives the precise recipe for computing the value at every cell of a low-resolution lookup table given a higher-resolution lookup table. Specifically, every pixel in the lower-resolution table is the max over a grid of pixels in the higher-

resolution table of dimension $(2D-2) \times (2D-2)$. Note that these max operations overlap between adjacent values.

Consider the 1D example below, in which the numbers in each column represent the *indices* of the highest-resolution lookup table over which the max operator is applied in order to compute the value at each cell in a lower-resolution lookup table:

D = 1:	0	1	2	3	4	5	6	7
D = 2:	0,1,2		2,3,4		4,5,6		6,7,8	
D = 4:	0,1,2,3,4,5,6				4,5,6,7,8,9,10			

The row labeled $D = 1$ represents the highest-resolution version, and the next two decimated versions are also shown. The "overlap" between pixels is necessary in order to compute conservative bounds.

The full image pyramid can be efficiently constructed recursively. The procedure is to apply a $\max$ convolution of kernel width 3, followed by a decimation of factor 2. I.e., the first pixel of $D = 4$ is the maximum value over $L(i)$ for $i \in \{0, 1, 2, 3, 4, 5, 6\}$. This can be computed by computing the max over the first *three* pixels in the $D = 2$ lookup table.

Because of this recursive construction, computing the image pyramids is a relatively fast operation. A megapixel lookup table (1024×1024) can be converted into an image pyramid in about 5 ms. The asymptotic complexity of building an image pyramid is $O(N)$ for an input image with N pixels, and our empirical evaluation runtimes also exhibit linear scaling. Our practical systems cache these image pyramids.

3) Point cloud decimation: Recall that the core of our method searches for a translation that best aligns a point cloud with a lookup table. While we pre-compute the various pyramid levels of the cost function, we compute decimated point clouds *on demand*.

The primary reason for this stems from our approach for dealing with rotations. Because the core method considers only translation, we explicitly rotate the point cloud at fixed intervals over the search range. In other words, we initiate searches for rotated copies of the original point cloud. However, it is often the case that some rotations can be quickly ruled out— even at very low resolution, they may score very poorly. Thus, it would be wasteful both in terms of computational costs and memory costs to pre-compute all possible rotated and decimated versions of the point clouds. When the matching algorithm needs a particular rotated and decimated version of the original point cloud, it is computed on demand. That result is cached, since it is likely that there will be multiple translation queries on that point cloud.

4) Overall search strategy: The essential data structure in our algorithm is a heap. Entries in the heap correspond to a rotation and a contiguous and square-shaped range of translations. Each heap entry independently specifies which point cloud and lookup tables should be used, allowing the search to simultaneously consider many-to-one and many-to-many type queries.

Searches are initialized by creating a small number of heap entries at very low resolution. The search proceeds by extracting the heap entry with the best score. If that heap entry corresponds to an alignment attempt using the highest-resolution lookup table, then we return that alignment. It is optimal, due to the fact that we never under-estimate the cost of a low-resolution alignment attempt: any more promising alignments must have already been ruled out.

If the heap entry does not reference the highest-resolution lookup table, we create four new heap entries, each with one quarter of the translational range.

Fig. 3. Generating ground-truth laser data. Our evaluation makes extensive use of this pre-processed map of the Intel Research Center. Derived from real robot data, we can use ray-casting to simulate realistic laser data with ground truth. The red regions were manually annotated to mark places where the robot could plausibly visit. Pairs of "co-visible" poses, which are likely to have significant overlap in their laser scans, are found by sampling two "red" poses and rejecting those whose connecting line does not lie within the red region. In all cases, robot orientation is randomly sampled from $[-\pi, \pi]$. Scans cover 360 degrees and, unless specified otherwise, are 1440 points with additive range noise of $\sigma = 0.01$. Cost functions are rendered with a cell size of 1/32 m.

IV. Evaluation

A. Performance (One-to-One)

The computational time required by our method is strongly affected by the search window. In turn, the search window is affected by how much positional uncertainty the robots accumulate: large uncertainties lead to large search windows.

We have evaluated our method in three different operating regimes: low uncertainty, moderate uncertainty, and large uncertainty. These regimes differ in terms of the translational and rotational search ranges. (Note that the search window is twice the size of the range, so that a 180 degree search range covers [-180 deg, 180 deg], or a full revolution.)

In each of these regimes, we simulate 1080 point LIDAR scans, consistent with a Hokuyo laser range finder. Timing data excludes the cost of decimating images and point clouds, since this time is both small *and* it is amortized over all the scan matching operations using that scan. The rotation resolution is 2 deg— fine enough for an incremental "hill climbing" stage to easily improve upon if needed.

To evaluate the performance of our system, we use the same Intel dataset used by earlier work [12] (see Fig 3). Starting with the standard raw Intel data from the Radish [9] data repository, a SLAM solution was computed, and a posterior gridmap was rendered. From this data, we are

able to generate synthetic ground-truthed laser measurements using ray-casting. While this data is simulated, it is derived from real-world data, and thus has realistic levels of clutter.

For this experiment, we randomly select two poses from the red-colored region (which was human labeled to represent places that the robot could physically go) and perform a scan-matching operation. We obtain the following results in our three regimes:

	Translation	Rotation	One-to-one time
Low	$\pm$ 2 m	$\pm$ 5 degrees	0.9 ms
Moderate	$\pm$ 30 m	$\pm$ 10 degrees	20 ms
Large	$\pm$ 50 m	$\pm$ 180 degrees	310 ms

These three regimes can be characterized in terms of typical applications. The *low* uncertainty regime is intended to be representative of "laser odometry" applications, where the robot has traveled only a short distance since the previous scan, and thus has accumulated little uncertainty. The *moderate* uncertainty regime is intended to model a typical loop-closing problem in a relatively benign environment (where wheel slippage is not a major problem, for example). The *large* uncertainty regime models a kidnapped-robot, or a very challenging large-scale SLAM system where robots seldom have the chance to close loops.

B. Performance of Many-to-One Matching

To evaluate the "many-to-one" approach, we again used the Intel dataset. In this case, we sample $S + 1$ random locations; one represents the current position of the robot, while the other S represent candidate scans. Formulated this way, we can compare the one-to-one matching approach to the many-to-one matching approach (see Fig. 6). Note that both approaches agree *exactly* on the best match amongst the S candidates.

The many-to-one approach finds the best loop closure within the set of S candidates dramatically faster than the one-to-one matcher. When $S = 50$, the many-to-many approach is 24x faster than one-to-one; for $S = 200$, this advantage increases to 45x. The particular speedups are data dependent, but we have consistently obtained similar results.

C. Performance of Many-to-Many Matching

We evaluated our many-to-many approach by randomly sampling S pairs of laser scans. We compared this to a many-to-one approach with S candidate pairs generated as before. Both approaches are thus considering the same number of candidate matches. For this experiment, we used our Intel evaluation framework and moderate uncertainty parameters.

The many-to-many approach increased performance by an additional 2.8x in our large uncertainty experiment (see Fig. 6). The impact in the moderate uncertainty experiment was much less– only about 22% for $S = 50$.

We also measured a noticeable improvement in the scores of the matches produced, which we attribute to the increase in diversity amongst the LIDAR scans (see Fig. 5).

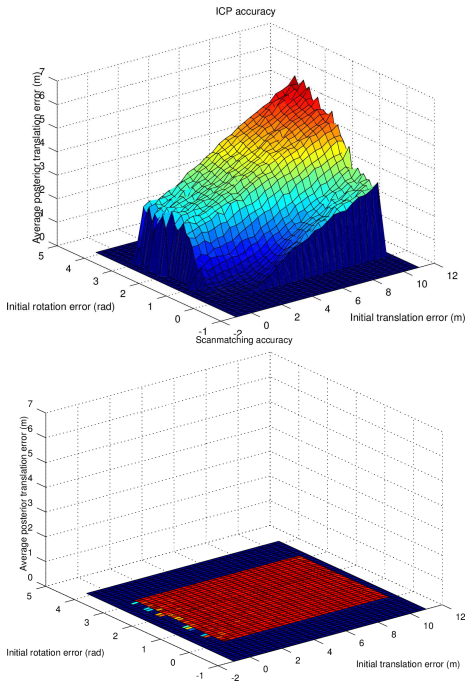

Fig. 4. Accuracy comparison. Using 950,000 co-visible pairs generated from the Intel Research Center dataset, we evaluated the posterior accuracy of ICP versus the proposed scan matching method. Initial alignment errors were sampled over a range of 10 m and 2π radians. The scan matcher's search window was set to 15 m and 2π radians, and no hill-climbing refinement was used. The proposed method has dramatically lower error, consistently finding alignments close to the ground-truth alignments. ICP's performance is poor for all but very small initial errors.

V. Performance Discussion

The performance of our approach is strongly affected by the quality of the best match: the search must continue expanding nodes on the heap until there are no nodes that *might* have a better score. When a match exists with near-zero error, it is likely that few other search nodes will need to be expanded.

When multiple queries are posed simultaneously, the critical factor affecting performance is the quality of the best match *amongst* those queries. Additional queries can be added to the search efficiently because the performance is dominated by the quality of the *best* query in the set. This is the fundamental reason why posing the problem as a "many-to-many" matching problem is advantageous: it avoids spending time computing alignments for queries that have high error (and are unlikely to be correct anyway).

VI. Conclusion

We have described an improved multi-resolution scan matcher that produces exactly the same results as a brute-force scan matcher. In particular, we have described how to construct the levels of an image pyramid such that conservatism is guaranteed. We evaluated the performance of this algorithm over several different operating regimes. In a low-noise laser-odometry scenario with a 40 Hz Hokuyo, our approach would require only about 3.6% of the computational power of a single core of a contemporary computer.

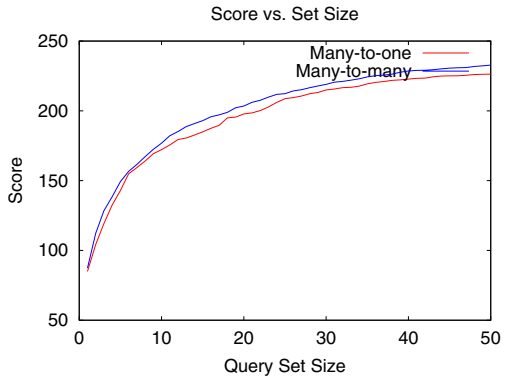

Fig. 5. Many-to-many impact on scan-matching quality. The scan quality (the sum of the scores for each point as computed by a lookup table) is plotted for both many-to-one and many-to-many approaches. Perfect matches have a score of 255. In addition to a modest improvement in runtime, many-to-many matching noticeably improved scan quality.

Our higher-uncertainty experiments indicate the plausibility of scan matching even for kidnapped robots.

But even larger and more important gains to mapping are possible when the traditional "one-to-one" operating model is reconsidered. We showed how a "many-to-one" and "many-to-many" approaches to scan matching can dramatically reduce the computational costs of finding loop closures. These approaches achieve this speed at the cost of reporting only the best match amongst a set of match candidate. However, this is also the information most valuable to a SLAM system.

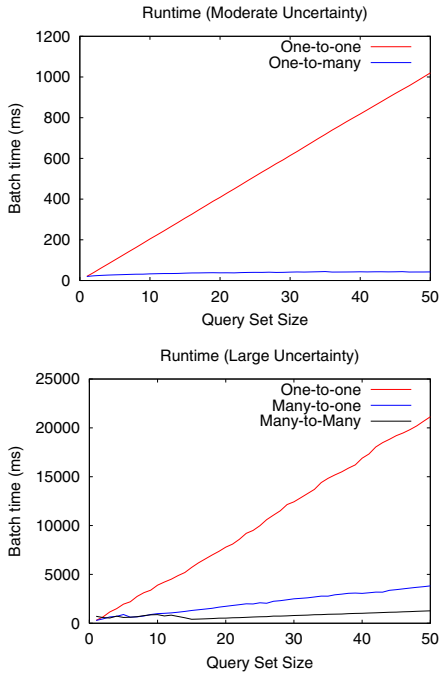

Fig. 6. One-to-one vs many-to-one vs many-to-many on Intel benchmark dataset. In this experiment, we examine the effects of query set size on runtime under two different noise regimes. In both cases, the many-to-one and many-to-many approaches are dramatically faster than the one-to-one approach. The many-to-many approach is only marginally faster than many-to-one on the moderate uncertainty dataset (not plotted), but 2.8x faster on large data. At $S = 50$, moderate uncertainty, the runtime of many-to-one was 42.3 ms. At $S = 50$ (large uncertainty) the runtime of many-to-one was 3310 ms, and many-to-many was 1167 ms. All timings are averaged over multiple runs. While not visible at this scale, all three methods empirically have linear growth characteristics.

REFERENCES

[1] P.J. Besl and N.D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[2] Michael Carsten Bosse. *ATLAS: A Framework for Large Scale Automated Mapping and Localization*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, February 2004.

[3] Andrea Censi. An ICP variant using a point-to-line metric. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2008.

[4] Andrea Censi, Luca Iocchi, and Giorgio Grisetti. Scan matching in the hough domain. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2005.

[5] I. J. Cox. Blanche- An experiment in guidance and navigation of an autonomous robot vehicle. *Robotics and Automation, IEEE Transactions on*, 7(2):193–204, 1991.

[6] Mark Cummins and Paul Newman. Probabilistic appearance based navigation and loop closing. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Rome, April 2007.

[7] Albert Diosi and Lindsay Kleeman. Fast laser scan matching using polar coordinates. *International Journal of Robotics Research*, 26(10):1125–1153, 2007.

[8] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America. A*, 4(4):629–642, Apr 1987.

[9] Andrew Howard and Nicholas Roy. The robotics data set repository (radish), 2003.

[10] Kurt Konolige and Ken Chou. Markov localization using correlation. In *IJCAI '99: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1154–1159, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[11] F. Lu and E. Milios. Robot pose estimation in unknown environments by matching 2d range scans. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 935–938, 1994.

[12] Edwin Olson. Real-time correlative scan matching. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4387–4393, Kobe, Japan, June 2009. IEEE.

[13] Edwin Olson and Pratik Agarwal. Inference on networks of mixtures for robust robot mapping. *International Journal of Robotics Research*, 32(7):826–840, July 2013.

[14] Edwin Olson, Johannes Strom, Ryan Morton, Andrew Richardson, Pradeep Ranganathan, Robert Goeddel, Mihai Bulic, Jacob Crossman, and Bob Marinier. Progress towards multi-robot reconnaissance and the MAGIC 2010 competition. *Journal of Field Robotics*, 29(5):762–792, September 2012.

[15] Andrew Richardson and Edwin Olson. PAS: Visual odometry with perspective alignment search. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, September 2014.

[16] Thomas Röfer. Using histogram correlation to create consistent laser scan maps. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 625–630, 2002.

[17] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3rd edition, 2009.

[18] Niko Sünderhauf and Peter Protzel. Switchable constraints for robust pose graph slam. In *IROS*, pages 1879–1884, 2012.

[19] S. Thrun, W. Burgard, and D. Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, San Francisco, CA, 2000. IEEE.

[20] S. Thrun, M. Diel, and D. ahnel. Scan alignment and 3d surface modeling with a helicopter platform, 2003.