Real-Time Correlative Scan Matching

Edwin B. Olson
University of Michigan
Department of Electrical Engineering and Computer Science
Ann Arbor, MI 48109
Email: ebolson@umich.edu
http://april.eecs.umich.edu

***Abstract*— Scan matching, the problem of registering two laser scans in order to determine the relative positions from which the scans were obtained, is one of the most heavily relied-upon tools for mobile robots. Current algorithms, in a trade-off for computational performance, employ heuristics in order to quickly compute an answer. Of course, these heuristics are imperfect: existing methods can produce poor results, particularly when the prior is weak.**

The computational power available to modern robots warrants a re-examination of these quality vs. complexity trade-offs. In this paper, we advocate a probabilistically-motivated scan-matching algorithm that produces higher quality and more robust results at the cost of additional computation time. We describe several novel implementations of this approach that achieve real-time performance on modern hardware, including a multi-resolution approach for conventional CPUs, and a parallel approach for graphics processing units (GPUs). We also provide an empirical evaluation of our methods and several contemporary methods, illustrating the benefits of our approach. The robustness of the methods make them especially useful for global loop-closing.

I. Introduction

Consider a robot sensing an environment from two poses x_0 and x_1; at each position, it obtains a two-dimensional lidar scan (z_0 and z_1). These lidar scans capture a horizontal cross-section of the environment typically sampled at one degree intervals. Provided that some parts of the environment are visible from both x_0 and x_1, it is generally possible to find a rigid-body transformation T that will project the points z_1 so that they align with z_0. This process of matching the scans z_0 and z_1 is known as *scan matching*. The solution to a scan matching problem is the rigid-body transformation T, which is parameterized by three values: two translational components (Δx and Δy) and a rotational component (θ). Aside from being an interesting perceptual problem, scan matching is at the center of most navigation, mapping, and localization systems. This is because the rigid body transformation T exactly corresponds to the motion of the robot as it travelled from x_0 to x_1. Since lidar-derived data is typically of far-higher quality than odometry (which is prone to unpredictable wheel slippage), scan matching plays a central role in estimating the motion of the robot.

The primary challenge in designing a scan matcher is to minimize the runtime complexity while maximizing the quality (and robustness) of the solutions. Most existing methods are designed around computationally-efficient local searches that produce answers quickly, but are not robust to initialization error. The problem is that scan matching, when viewed as an optimization problem, is rarely convex: the cost surface can be very complicated, having many local minima (see Fig. 1). The vehicle's dead-reckoning error can cause the initial estimate to be far from the global maximum; as a result, many approaches fail to identify the global maximum.

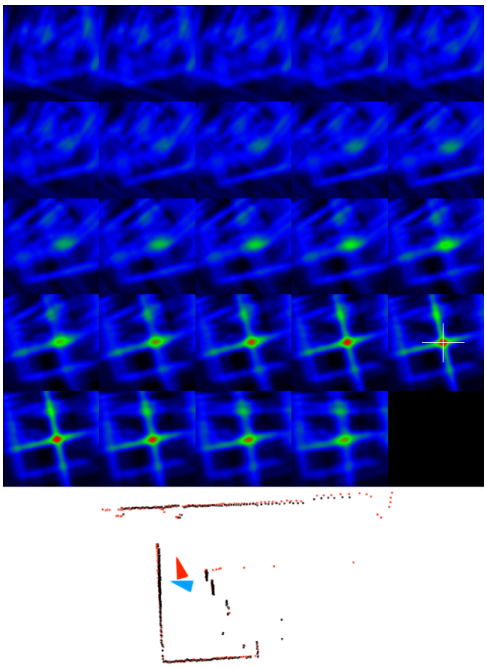

Fig. 1. Correlation (3D) Cost Function. Given two laser scans (bottom), we compute a rigid-body transform that aligns them by computing the cost function in three dimensions (translation in $\hat{x}$ and $\hat{y}$) and θ). Each tile represents a slice of the cost volume for a fixed θ. The numerical maximum is then identified (white cross hairs).

This paper describes a family of scan matching algorithms based upon cross-correlation of two lidar scans. Our approach casts the problem in a probabilistic framework: it finds the rigid-body transformation that maximizes the probability of having observed the data. Rather than trusting a local search algorithm to find the global maximum (an approach that does not work well in the presence of initialization noise, as we will illustrate), we perform a search over the entire space of plausible rigid-body transformations. This plausible region is derived from a prior which, in turn, can be derived from the commanded motion or wheel/visual odometry.

The central contributions of this paper are:

- We describe the theoretical and practical advantages of a correlative scan matching approach.
- We present a multi-resolution implementation capable of real-time operation on a conventional microprocessor;
- We show how the correlation-based method can be mapped onto a Graphics Processing Unit (GPU), freeing the CPU for other tasks.
- We show how covariance estimates can be obtained from the matching operation.
- We present a thorough empirical evaluation of our methods versus three different algorithms in common use.

The quality and robustness of our methods, coupled with their ability to operate in real-time, make them ideal for any robotic platform in which robustness and accuracy are of high importance. Naturally this includes Simultaneous Localization and Mapping (SLAM) applications, but virtually any navigation or localization system would benefit.

We begin the paper with a brief review of previous work (Section II). Our CPU and GPU based methods are described in Section III. Section IV describes our empirical evaluation methods and compares the performance of our method to several methods in wide use.

II. Prior Work

Iterative Closest Point (ICP) [1], [2] and Iterative Closest Line (ICL) [3], [4], [5] are used pervasively in scan matching. In ICP, each point in the query scan is associated with the reference scan according to a distance metric (most commonly Euclidean distance). A rigid-body transformation that best aligns the reference and query points can then be computed. Horn's exact closed-form algorithm [6] is especially well suited to the task.

Lu and Milios [7] describe two scan matching methods based on ICP. The first method considers the rotational and translational components separately: alternately fixing one, then optimizing the other. Given rotation, least-squares is used for optimizing translation. A search over rotations is conducted using the global-section method [8]. Their second method, dubbed IDC, combines two ICP-like algorithms with different point-matching heuristics.

ICL is similar to ICP, except that instead of matching query points to reference points, the query points are matched to lines extracted from the reference points. This approach is motivated by the fact that the sensor samples the environment sparsely, and that different lidar scans may sample different parts of the environment. In other words, even if the environment is the same in the reference and query scans, there may not be reasonable correspondences for all the points. The simplest ICL variants interpolate lines between each pair of adjacent lidar points, but this is undesirable in many cases (e.g. banister railings and depth discontinuities). Alternatively, heuristic methods can be used to determine which pairs of points are likely to be part of a connected surface, or lines/splines can be extracted from larger sets of points [9], [3], [4]. As part of his work on the robot Blanche, Cox described one of the earliest ICL scan matching algorithms [10]. It is interesting to note that the algorithm, running on hardware available at the time, achieved an update rate of 0.125 Hz. It is generally desirable for scan matching methods to produce covariance estimates, and Cox suggested using the uncertainty estimate that falls out from the least-squares formulation. In simple environments (especially when a low-complexity prior map is available) this can work well. In practice, however, uncertainty estimates derived from least-squares are far too confident; this is because the least-squares step is conditioned on the data association. Unfortunately, data association is initially unknown, and iterative methods can fail to compute the correct correspondences. Since the uncertainty estimate derived from least-squares does not reflect uncertainty in data association, the uncertainty estimates tend to be over confident.

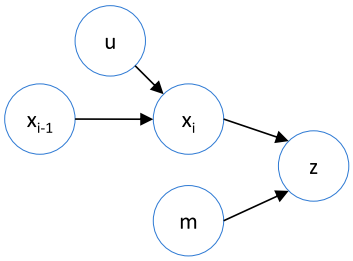

Fig. 2. Graphical model for probabilistic scan matching. x_{i-1} represents the previous position of the robot, u the motion of the robot, x_i the new robot position, m the world model, and z the laser scan observation.

The literature is filled with other scan matching heuristics. These include using polar coordinates [11], the Normal Distribution Transform, feature-based methods [3], [4], Hough transforms [12], and histograms [13], [14].

Our work is quite similar in spirit to Konolige's correlation based localization approach [15]. While the formulations of the problem are almost identical, we describe new methods for computing the answers.

Graphics Processing Units (GPUs) have not yet been widely used for robotics applications. A literature search revealed only one paper on motion planning[16]; surprisingly, it is from 1990. We believe our implementation is the first example of accelerating mapping and localization using a GPU.

III. Approach

A. Probabilistic formulation

We model the scan matching problem according to the graphical model shown in Fig. 2. The robot is moving from x_{i-1} to x_i, according to some motion u. The observation z is dependent on the environment model m and the robot's position.

Our goal is to find the posterior distribution over the robot's position, $p(x_i|x_{i-1}, u, m, z)$. We apply Bayes' rule and remove irrelevant conditionals, giving:

$$p(x_i|x_{i-1}, u, m, z) \quad \propto \quad p(z|x_i, m)p(x_i|x_{i-1}, u) \qquad (1)$$

The first term, $p(z|x_i, m)$ is the observation model: how likely is a particular observation, if the environment and the robot's

position are known? The second term, $p(x_i|x_{i-1}, u)$ is the motion model of the robot, as obtained (for example) from control inputs or odometry.

While the motion model is typically known in terms of a multivariate Gaussian distribution, the observation model is more difficult to compute and more complex in structure. It generally has multiple extrema, for example, as seen in Fig. 1.

The central contribution of this paper is a method for efficiently computing the distribution $p(z|x_i, m)$ so that we can compute the posterior distribution of the robot's position (as in Eqn. 1). While previous authors have suggested hill-climbing in order to find local maxima of $p(z|x_i, m)$ (in hopes of obtaining the maximum likelihood solution), our approach will more thoroughly characterize the distribution. Our approach results in both a more robust maximum likelihood estimate and a principled uncertainty estimate.

Like previous work, we assume that each individual lidar return z_j is independent, allowing us to write:

$$p(z|x_i, m) = \prod_j p(z_j|x_i, m) \tag{2}$$

The probability distribution for a single lidar sample z_j should, in principle, consider which surface of the map m would be visible from position x_i along a particular bearing. This would require an expensive ray-casting type operation. Like others [15], we neglect visibility and occlusion effects, and approximate the probability of z_j in terms of its distance from *any* surface in m.

In a Simultaneous Localization and Mapping (SLAM) context [17], the model m is derived from previous lidar observations. In other cases, the model m may be known in advance [10].

B. Lookup-Table Rasterization

The computation of the probability $p(z|x_i, m)$ can be accelerated by building a 2D lookup table. We follow the approach of previous approaches [18], [19] by pre-computing a lookup table containing log probabilities of lidar observation at each (x, y) position in the world.

Our rasterization process begins with map m. For each observable point m_i in the map, we can the compute the conditional probability that the sensor observes a nearby point p given that m_i was the cause of that observation. We repeat this process for each point in the map, recording the maximum probability for each point in the lookup table. Since our lookup table must be viewpoint independent, we approximate the potentially banana-shaped distribution arising from the sensor model (which has independent range and bearing noise) as a radially-symmetric distribution. With currently available sensors, this seems to be a reasonable approximation. The resulting lookup table can be visualized as an image, as seen in Fig. 3.

A critical question is: "where does the model m come from?" In rare cases, a model is known in advance [10], but more often, the model must be estimated from previous observations of the environment. Estimating this model is a complex issue on its own, and in the limit, requires a full solution to the SLAM problem.

In this paper, we will simply use an earlier laser scan (the *reference scan*) as our model. This approach has the advantages of being easy to implement, robust (in that model cannot diverge due to earlier data association errors), and perhaps most pragmatically, serves as a baseline implementation aiding replication of our results. More sophisticated implementations can build up more detailed models by combining multiple scans (e.g. CARMEN's Vasco), or by extrapolating continuous surfaces from lidar points [4].

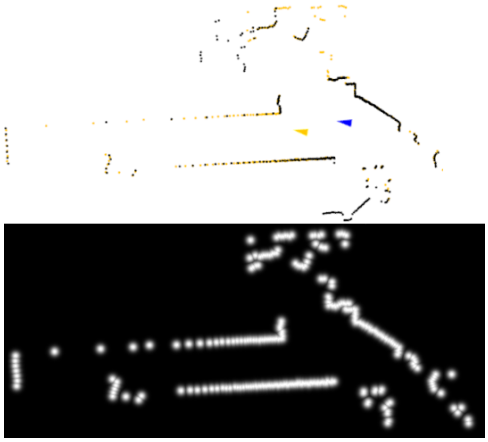

Fig. 3. Rasterized Cost Table. Given a reference scan, the log-probability of observing a new point is encoded in a lookup table. Top: the reference scan, bottom: the resulting lookup table. Bright values indicate large probabilities.

C. Approach Overview

To review, our goal is to estimate the distribution $p(z|x_i, m)$, which via Eqn. 1, can be used to obtain the distribution $p(x_i|x_{i-1}, u, m, z)$. Unlike earlier approaches, we are interested not only in the maximum likelihood value of $p(x_i|...)$, but the distribution itself so that we can obtain a measure of uncertainty. There is no simple expression for this distribution: it must be evaluated numerically. In the following two sections, we will describe two algorithms for rapidly evaluating the distribution $p(z|x_i, m)$ over many values of x_i.

D. Multi-Level Resolution Implementation

This section describes a fast correlation based method intended for use on conventional microprocessors. Conventional CPUs are not well suited to computing vast numbers of samples of $p(z|x_i, m)$. Our approach reflects this, attempting to minimize the number of evaluations required while 1) characterizing the distribution over a large area and 2) precisely locating the maximum likelihood value. We describe our approach in three pieces: building from a naive implementation towards our multi-resolution approach.

1) Brute Force: In principle, we need to evaluate $p(z|x_i, m)$ over a three-dimensional volume of points; the three dimensions correspond to the unknown parameters of T: Δx, Δy, and θ. Our naive implementation consists of three

nested loops; for each voxel, the probability is computed and recorded. The evaluation at a single voxel involves a fourth nested loop, iterating over each point in the query scan, projecting it, and looking up the cost in the lookup table. This method is quite slow, as our results sections illustrates.

2) Computing 2D Slices: One of the reasons the brute-force method is slow is that the points in the query scan are reprojected for each voxel. This is unfortunate, since for a given value of θ, the projected query points are related by pure translation for the $\hat{x}$ and $\hat{y}$ search directions. In other words, a significant amount of computational time can be saved by iterating over θ in the outer-most loop at which time the query points are properly rotated. The inner two loops (for $\hat{x}$ and $\hat{y}$) simply translate the query points. This operation can be made even faster by making the step size of the translational search match the resolution of the lookup table. This method is significantly faster than the brute force method, as illustrated in the results section.

3) Multi-Level Resolution: Our final CPU-based implementation makes use of two raster lookup tables rendered at different resolutions. The first is a high-resolution (3 cm resolution in our implementation) and the second is a low resolution table (30 cm resolution).

At very low resolutions, it is possible for details in the reference scan to disappear. Instead, we compute the low resolution table so that each cell is set to the maximum value of the corresponding cells in the high-resolution map. This ensures that the probabilities computed by the low-resolution map are always *at least as large* as those in the high-resolution map. In other words, it guarantees that we will not miss maxima.

Our strategy is to use the low-resolution map to quickly identify areas that might contain the global maximum and areas that probably do not. The goal is to minimize the volume that is searched at high resolution. The method is:

1) Evaluate the probability $p(z|x_i, m)$ over the entire 3D search window using the low-resolution table.
2) Find the best voxel in the low-resolution 3D space that has not already been considered. Denote this value as L_i. If $L_i < H_{best}$, terminate: H_{best} is the best scan matching alignment.
3) Evaluate the search volume inside voxel i using the high resolution table. Suppose the log-likelihood of this voxel is H_i. Note that $H_i \leq L_i$ since the low-resolution map overestimates the log likelihoods. If $H_i > H_{best}$, set $H_{best} = H_i$.

This multi-level resolution method is extremely fast, making real-time correlative scan matching possible. It is also exceptionally robust, as illustrated by the results section.

E. Graphics Processor Approach

Graphics Processing Units (GPUs) are capable of very large computational throughput and are well suited to evaluating a function like $p(z|x_i, m)$ over a range of values. We believe we are the first to use GPUs in a robot mapping and localization context.

Our implementation is written as an OpenGL GLSL shader, rather than a vendor-specific language like NVidia's CUDA. While this requires our implementation to cast the problem in terms of a 3D rendering operation (i.e., computing functions by drawing textured polygons), the correlative scan matching method is simple enough to make this straight-forward.

Like the CPU based implementations, we compute "slices" of $p(z|x_i, m)$, fixing the orientation of x_i for each individual tile. Our fragment shader takes two textures as inputs: an 1D array consisting of the query points, and a 2D texture corresponding to the lookup table. We used the exact same lookup table in the GPU implementation as in the CPU implementation.

Each fragment computes the value of $p(z|x_i, m)$ for a particular value of x_i. This value is passed to the shader via a texture coordinate. Each fragment in a polygon automatically receives a linearly interpolated texture coordinate based on the texture coordinates specified at the vertices of the polygon. In other words, when we render polygons to the screen, the texture coordinates do not correspond to textures at all: they are interpreted by the fragment shader as the rigid-body transformation it should apply to the query points.

The fragment shader itself most closely resembles the naive CPU method: each fragment iterates over the query points, projecting each point according to the local value of x_i, then fetching the log-likelihood from the lookup table. The shader simply adds together the log-likelihoods from each pixel and outputs the resulting value to the frame buffer. The host CPU can then examine the frame buffer for the maximum likelihood solution.

On the host CPU side, we draw a quadrilateral for each orientation of x_i we wish to evaluate. A single quadrilateral corresponds to a set of translations with a fixed orientation. The resulting data from the GPU is shown in Fig. 1. Note that the color-coding in the figure is a visualization aid: each pixel output by the GPU is a scalar.

F. Computing Covariances

In many applications, the maximum likelihood estimate of x_i is sufficient. However, our methods allow a principled estimation of uncertainty.

Once the value of the cost function has been evaluated over a range of values of x_i, a multivariate Gaussian distribution can be fit to the data. Let $x_i^{(j)}$ be the j^{th} evaluation of x_i:

$$\begin{aligned} K &= \sum_j x_i^{(j)} x_i^{(j)T} p(x_i^{(j)}|x_{i-1}, u, m, z) \\ u &= \sum_j x_i^{(j)} p(x_i^{(j)}|x_{i-1}, u, m, z) \\ s &= \sum_j p(x_i^{(j)}|x_{i-1}, u, m, z) \\ \Sigma_{x_i} &= \frac{1}{s}K - \frac{1}{s^2}uu^T \end{aligned} \tag{3}$$

Estimating the scan matcher's uncertainty from computed values of $p(z|x_i, m)$ takes into account both major sources of

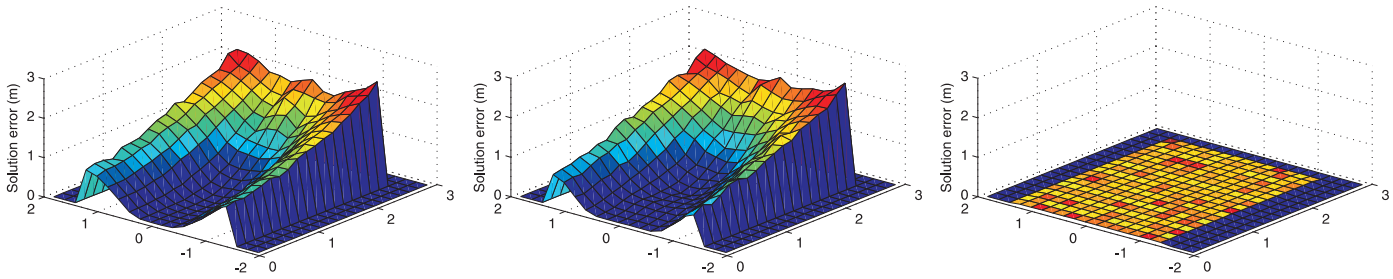

Fig. 4. Posterior Accuracy. The mean posterior translational error (z axis) for three methods was computed on 200,000 iterations with initialization error of up to 74 degrees (x axis) and 3 meters (y axis). Left: ICP, middle: Hill-Climbing, right: Correlation (our method). As the initial estimate of the rigid-body transformation deteriorates, so does the quality of the output for both ICP and Hill-Climbing. In contrast, the posterior error of the proposed method is independent of the initial error. The results for ICL were very similar to ICP and have been omitted.

ambiguity: the noise of the sensor itself, and the uncertainty of which query points should associate to which parts of the model. The shortcoming of this approach is that the resulting

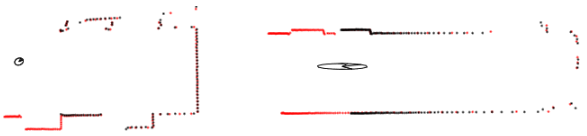

Fig. 5. Sample Covariances. Left: position is well constrained in both $\hat{x}$ and $\hat{y}$ directions, yielding confident covariance estimate. Right: long hallway provides only a few longitudinal constraints, yielding an elongated uncertainty ellipse.

Gaussian is fit only to the samples that have been computed. Any high-probability areas that are not within the sampled volume will not be reflected in the Gaussian, resulting in an over-confident estimate. Thus, it is important to evaluate the probability $p(z|x_i, m)$ over large areas of x_i— something our approaches do well. See Fig. 5 for covariance estimates arising from two cases.

IV. Results

A. Experiment Design

A thorough empirical evaluation of a scan matcher requires a great deal of ground-truthed data. At present, the only reasonable way of obtaining this data is through simulation. The traditional shortcomings of simulation (unrealistically uncluttered environments with minimal variety) can be avoided, however, by simulating new data from a map generated from real data. In this paper, we simulate lidar data from a map built from the Intel Research Center data set (see Fig. 6). The raw data is available from the Radish [20] data repository. This map was constructed using a scan matcher employing the methods in this paper in conjunction with a robust loop-closing method [4].

In order to make the simulated observations more realistic (and to account for the fact that not all parts of the building were observed during the data collection), legal observation positions were hand-annotated; these regions appear in blue. Our experiments randomly selected a legal pose from the blue area, then selected another pose nearby. This second pose must satisfy two conditions: it must also be a legal pose (i.e., in a blue area), and there must be a continuous line of blue between the two scans. This, coupled with our use of 360-degree scans, ensures that the two scans contain at least some overlapping views. These precautions ensure that the simulated scans include at least *some* overlapping features; if there are no overlapping features, then no scan matching method could be expected to compute a reasonable answer.

This sampling procedure produces realistic scans that exhibit real-world clutter, noise, and occlusion effects due to the robot's different positions. We did not model the effects of dynamic objects like pedestrians. Finally, the simulated laser

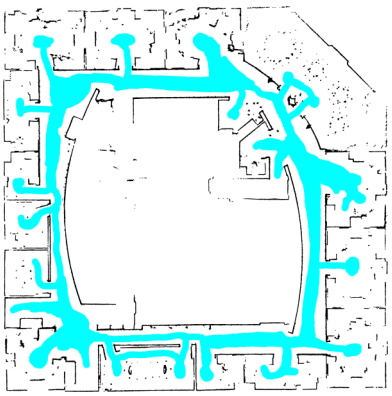

Fig. 6. Intel Research Center. A map derived from real data was used to generate ground-truthed observations. Robot poses are constrained to be in the blue area, which helps to ensure realistic scans.

scans were corrupted by Gaussian noise consistent with that of an LMS-210 lidar.

B. Comparison Methods

In addition to the correlation-based methods described in this paper, we have implemented variants of three popular algorithms in order to provide a competitive comparison. These methods are intended to be representative of algorithms in use.

1) ICP: The Iterative Closest Point (ICP) algorithm used here uses Euclidean distance for its distance metric with a 1.0 m matching limit. The variant here is symmetric: for each point in both scans, we find the closest point from the other scan. For example, suppose that a_i is the point in a closest to b_j, and

that b_k is the point in b closest to a_i. If b_i is more than twice as far from a_j as b_k, then the correspondence (a_i, b_i) is deleted. This refinement helps recognize parts of the environment that are visible from only one scan, and rejects the long-range correspondences that might otherwise result. In the example above, it is likely that b_j has no corresponding reference point. Our symmetric variant additionally creates additional correspondences, helping to reduce error of the match. We use Horn's algorithm [6] to compute the optimal rigid-body transformation given a set of correspondences.

2) ICL: The Iterative Closest Line (ICL) algorithm used in our experiments associated query points with the closest line segment using Euclidean distance. The line segments are generated from consecutive pairs of points that are no more than 1.0 m apart. Suppose that query point q_i has been associated with a line segment s: the point on s closest to q_i are associated and are passed to Horn's algorithm in order to compute a rigid-body transformation.

3) Hill-Climbing: The hill climbing method used here is modelled after the Vasco scan matcher, which is distributed with CARMEN. It is a local search method that repeatedly attempts steps along the axial directions, accepting steps that improve the match. The steps start off relatively large (1 m in x and y and 10 degrees in θ); when no step yields a reduction in error, the step sizes are reduced in half. When the step sizes reach a pre-determined minimum (0.001 m for x and y and 0.05 deg for θ), the search ends. Like our methods, Vasco's optimization is formulated in probabilistic terms and uses a log-likelihood lookup table. In order to enable a direct comparison to our correlation based methods, we use the same rasterized lookup table.

Our primary experiment consisted of about 250,000 iterations; for each iteration, we selected new poses, added noise to the prior estimate, and ran each scan matcher. In Fig. 4, the average translation error in the solution (z axis) is plotted versus the prior's translational noise (x axis) and rotational noise (y axis). (The symmetry of the plots is due to prior estimates having both positive and negative rotational errors.)

In general, errors of more than a few centimeters or a few degrees are unacceptable: they cause error to accumulate in the robot's position estimate too quickly. Intuitively, we expect higher levels of noise in the prior to be reflected in larger errors in the solution. This is indeed the case for ICP, ICL, and Hill Climbing. It is also noteworthy that the Hill Climbing method had slightly *higher* error than ICP even when there was little noise in the initialization.

The central result of this paper is this: the error of the correlation based methods is both very low *and* robust to noise in the prior. The price for this superior performance is greater computational cost. However, as we will see in Section IV-C, this cost is quite manageable.

The resolution of the lookup table influences the accuracy of the results. We characterized the error of our methods as a function of lookup table resolution over a range of 0.001m to 1m. We found no advantage to resolutions finer than 1.5cm, with 3.1cm resolution being only slightly worse. Error increases rapidly at resolutions coarser than 6cm. Based on this data, we recommend resolutions of 3cm.

C. Performance

It should come as little surprise that the correlative methods described here are typically slower than ICL, ICP, and Hill Climbing. The primary motivation for the correlative methods was an improvement in quality. However, the correlation based methods *are* fast enough for real-time use (see Fig. 7). The CPU used for the experiments was an Intel Core2 6600 at 2.4 GHz. While the CPU has multiple cores, our experiments made use of only one thread. We evaluated two different GPUs, spanning two generations of performance. The 7600 denotes an NVidia 7600GS running at 400 MHz, while the GTX260 denotes an NVidia GTX260 (which has 192 stream processors) at 650MHz. The grid resolution was 1/32 meter, and for the correlative methods, the θ step size was set to one degree.

The computational complexity of ICP and ICL, even with our simple implementations, is quite good (see Figure 7). Hill-Climbing is particularly fast. While their computational complexity scales very well with increasing positional uncertainty, their quality is completely unacceptable in these high uncertainty regimes (see Fig. 4).

As expected, the computational complexity of the correlative methods increases rapidly with increasing prior uncertainty: the complexity scales with the volume of the uncertainty. For relatively small uncertainties, such as those that would be encountered in an incremental scan matching context, several correlative algorithms are capable of real-time performance. In these contexts, the search area decreases with the update rate of the scanner: with fast updates, the robot simply doesn't have time to accumulate a significant amount of positional error. At 75Hz, for example, a vehicle would have to have a velocity error of 37 m/s (82 mph) in order to warrant a 0.5 m search window. The size of the search window in θ is equally generous: at 75Hz, the robot would have to spin at 240 rpm in order to warrant a 20 degree window. Yet, even at a 75Hz update rate, the multiple-resolution correlative algorithm is fast enough (8.4 ms) to operate at real time.

For loop closing applications, real-time performance is less critical: the ability to find correct correspondences becomes the critical factor. A robot will only close a large loop on a very intermittent basis (perhaps once every few minutes). These loop closures are where large uncertainty windows tend to appear. We see the performance of the correlative methods (particularly the multiple-resolution implementation, which can handle 4 m translational errors with 90 degree rotational errors at over 10Hz) is entirely adequate for these purposes.

Variability in runtime is also a challenge for real-time systems. The runtime complexity of ICP, ICL, Hill-Climbing, and the multiple-resolution correlative algorithm are data-dependent. With ICP and ICL, it can take about 2.5 times longer (see Fig. 8) to compute the answer for some scans as for others. The single-resolution correlative methods are generally

	ICP	ICL	Hill-Climb	Correlative (Naive)	Correlative (2D Slices)	Correlative (Multi-Res)	Correlative (GPU 7600GS)	Correlative (GPU GTX260)
0.5 m, 20 deg	56 ms	64 ms	1.1 ms	246 ms	27 ms	8.4 ms	98 ms	26 ms
2.0 m, 40 deg	99 ms	105 ms	1.3 ms	7512 ms	692 ms	20.8 ms	1563 ms	289 ms
4.0 m, 90 deg	145 ms	174 ms	1.0 ms	65282 ms	5029 ms	86.1 ms	13166 ms	2012 ms

Fig. 7. Runtime Results. The processing time to register two scans is given for three different search window sizes. The first window size is typical of an incremental scan matching operation (used for correcting odometry, for example), while the third window represents a fairly large "loop-closing" problem. For three different search windows, we collected timing estimates. Note that our implementation of ICP and ICL were simple implementations and could be made to substantially faster by employing a quad-tree or similar data structure.

(0.5 m, 20 deg)	ICP	ICL	Hill-Climb	Correlative (Naive)	Correlative (2D Slices)	Correlative (Multi-Res)	Correlative (GPU 7600GS)	Correlative (GPU GTX260)
mean time	56 ms	64 ms	1.1 ms	246 ms	27 ms	8.4 ms	98 ms	26 ms
10th percentile	34 ms	33 ms	1 ms	228 ms	24 ms	5 ms	94 ms	19 ms
50th percentile	55 ms	64 ms	1 ms	245 ms	26 ms	7 ms	97 ms	23 ms
90th percentile	84 ms	101 ms	1 ms	259 ms	29 ms	12 ms	103 ms	33 ms
P90/P10 ratio	2.47	2.65	1.0	1.13	1.21	2.40	1.10	1.74

Fig. 8. Runtime Variability. For real-time applications, runtime variability can be a significant design consideration. The ICP, ICL, Hill Climbing, and Multi-Resolution algorithms exhibit significant runtime variability due to the data-dependence of their algorithms. The variability of the GPU (especially GTX260) is somewhat surprising, and may be attributable to large overhead costs.

better, though the variability of the GTX260 GPU implementation was higher than expected. This variability only occurs at the small search size: when performing larger searches, the GPU methods become very consistent. We suspect that some of this variability is due to the overhead of setting up the operation.

V. Conclusions

This paper has presented a family of scan matching algorithms based upon correlations. The approach is extremely robust to initialization noise, dramatically out-performing methods currently in use. We demonstrated that it is possible to use these robust algorithms in real-time on both CPUs and GPUs.

Our method is probabilistically motivated, making it easy to incorporate a probabilistic prior and to compute a covariance estimate. Implementations of our routines are available at http://april.eecs.umich.edu.

References

[1] P. Besl and N. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[2] S. Thrun, M. Diel, and D. ahnel, "Scan alignment and 3d surface modeling with a helicopter platform," 2003. [Online]. Available: citeseer.ist.psu.edu/thrun03scan.html

[3] M. C. Bosse, "ATLAS: a framework for large scale automated mapping and localization," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, USA, February 2004.

[4] E. Olson, "Robust and efficient robotic mapping," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, USA, June 2008.

[5] A. Censi, "An icp variant using a point-to-line metric," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2008.

[6] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America. A*, vol. 4, no. 4, pp. 629–642, Apr 1987.

[7] 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)*, 1994, pp. 935–938. [Online]. Available: citeseer.ist.psu.edu/lu94robot.html

[8] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C*, 2nd ed. Cambridge, UK: Cambridge University Press, 1992.

[9] M. A. Fischer and R. C. Bolles, "A paradigm for model-fitting with applications to image analysis and automated cartography," 1981.

[10] I. J. Cox, "Blanche- An experiment in guidance and navigation of an autonomous robot vehicle," *Robotics and Automation, IEEE Transactions on*, vol. 7, no. 2, pp. 193–204, 1991.

[11] A. Diosi and L. Kleeman, "Fast laser scan matching using polar coordinates," *International Journal of Robotics Research*, vol. 26, no. 10, pp. 1125–1153, 2007.

[12] A. Censi, L. Iocchi, and G. Grisetti, "Scan matching in the hough domain," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2005.

[13] T. Rfer, "Using histogram correlation to create consistent laser scan maps," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2002, pp. 625–630.

[14] M. Bosse and J. Roberts, "Histogram matching and global initialization for laser-only slam in large unstructured environments," in *ICRA*, 2007, pp. 4820–4826.

[15] K. Konolige and K. Chou, "Markov localization using correlation," in *IJCAI '99: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 1154–1159.

[16] J. Lengyel, M. Reichert, B. R. Donald, and D. P. Greenberg, "Real-time robot motion planning using rasterizing computer graphics hardware," in *In Proc. SIGGRAPH*, 1990, pp. 327–335.

[17] S. Thrun, "Robotic mapping: A survey," in *Exploring Artificial Intelligence in the New Millenium*, G. Lakemeyer and B. Nebel, Eds. Morgan Kaufmann, 2002, to appear.

[18] 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: IEEE, 2000.

[19] M. Montemerlo, N. Roy, and S. Thrun, "Perspectives on standardization in mobile robot programming: The Carnegie Mellon navigation (CARMEN) toolkit," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, Las Vegas, NV, October 2003, pp. 2436–2441.

[20] A. Howard and N. Roy, "The robotics data set repository (radish)," 2003. [Online]. Available: http://radish.sourceforge.net/