2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

An ICP variant using a point-to-line metric

Andrea Censi

***Abstract*— This paper describes PLICP, an ICP (Iterative Closest/Corresponding Point) variant that uses a point-to-line metric, and an exact closed-form for minimizing such metric. The resulting algorithm has some interesting properties: it converges quadratically, and in a finite number of steps. The method is validated against vanilla ICP, IDC (Iterative Dual Correspondences), and MBICP (Metric-Based ICP) by reproducing the experiments performed in Minguez *et al.* (2006). The experiments suggest that PLICP is more precise, and requires less iterations. However, it is less robust to very large initial displacement errors. The last part of the paper is devoted to purely algorithmic optimization of the correspondence search; this allows for a significant speed-up of the computation. The source code is available for download.**

***Index Terms*— Scan matching, localization, ICP, IDC, MbICP, metric, point-to-line**

I. INTRODUCTION

ICP solves a surface matching problem which can be expressed as follows: given a reference surface $\mathcal{S}^{\text{ref}}$ and a set of points $\{\boldsymbol{p}_i\}$, find the roto-translation $\boldsymbol{q} = (\boldsymbol{t}, \theta)$ that minimizes the distance of the points $\boldsymbol{p}_i$, roto-translated by $\boldsymbol{q}$, to their projection on surface $\mathcal{S}^{\text{ref}}$. That is, in formulae:

$$\min_{\boldsymbol{q}} \sum_i \left\| \boldsymbol{p}_i \oplus \boldsymbol{q} - \Pi\left\{\mathcal{S}^{\text{ref}}, \boldsymbol{p}_i \oplus \boldsymbol{q}\right\} \right\|^2 \qquad (1)$$

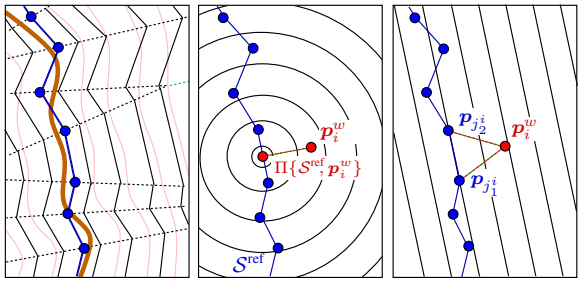
Here the symbol '$\oplus$' denotes the roto-translation operator ($\boldsymbol{p} \oplus (\boldsymbol{t}, \theta) \triangleq \mathbf{R}(\theta)\boldsymbol{p} + \boldsymbol{t}$), and $\Pi\left\{\mathcal{S}^{\text{ref}}, \cdot\right\}$ denotes the Euclidean projector on $\mathcal{S}^{\text{ref}}$. A closed form for the solution is not found in general, due to the arbitrary nature of $\mathcal{S}^{\text{ref}}$ and the non-linearity of $\oplus$.

ICP minimizes (1) iteratively, starting from a first guess $\boldsymbol{q}_0$. The point projections over $\mathcal{S}^{\text{ref}}$ are computed according to the old guess $\boldsymbol{q}_k$, and a solution is found for the new guess $\boldsymbol{q}_{k+1}$. In formulae, at each step k, ICP solves this problem, for which a closed form is available:

$$\min_{\boldsymbol{q}_{k+1}} \sum_i \left\| \boldsymbol{p}_i \oplus \boldsymbol{q}_{k+1} - \Pi\left\{\mathcal{S}^{\text{ref}}, \boldsymbol{p}_i \oplus \boldsymbol{q}_k\right\} \right\|^2 \qquad (2)$$

After its introduction, many ICP variants have been investigated, because the core algorithm can be slightly modified in many ways: which subset of points to use, how to define the error metric, how to discard outliers, etc — see [1] for a short survey of the popular variants used in the vision community. Specialization of the ICP in the robotics community have flourished [2], [3], [4], [5], [6]; the research focused on solving some of the problems that become evident when using ICP for scan-matching: it converges to the wrong solution for large initial errors; correspondence search is expensive; convergence is slow; occlusions and outliers are frequent; as-is, it does not fit well in a probabilistic framework.

A. Censi is with the Control & Dynamical Systems department, California Institute of Technology, 1200 E. California Blvd., 91125, Pasadena, CA. andrea@cds.caltech.edu



(a) Distance to curve and to polyline (b) Point-to-point metric (c) Point-to-line metric

Fig. 1. Near convergence, the point-to-line metric approximates the distance to the surface better than the point-to-point metric used in vanilla ICP.

The main contribution of this paper is the use of a point-to-line metric instead of the point-to-point metric used by vanilla ICP. Call $\boldsymbol{n}_i$ the normal to the surface at the projected point. Then, the point-to-line metric is written as:

$$\min_{\boldsymbol{q}_{k+1}} \sum_i \left(\boldsymbol{n}_i^{\mathsf{T}} \left[\boldsymbol{p}_i \oplus \boldsymbol{q}_{k+1} - \Pi\left\{\mathcal{S}^{\text{ref}}, \boldsymbol{p}_i \oplus \boldsymbol{q}_k\right\} \right] \right)^2 \qquad (3)$$

The idea of using a point-to-line metric is not new at all: it was proposed in [7] — which, together with [8], is credited to be the originators of ICP— however, the non-linear equation (3) was linearized and solved using linear least-squares. This paper introduces (in Appendix I) an exact closed-form solution to (3) in the planar case.

Using the point-to-line metric, *and* an exact solution, drastically improves the convergence properties of the algorithm: PLICP has quadratic convergence instead of the linear convergence of vanilla ICP. Moreover, if the reference surface is a polyline, as it is usually assumed in robotics scan-matching, PLICP provably converges in a finite number of steps. An intuitive explanation is that the point-to-line metric (Fig. 1c) more closely approximates the real surface distance (Fig. 1a) than the point-to-point metric (Fig. 1b).

As an experimental validation, this paper reproduces the experiments performed in [6], so that a direct comparison with ICP, IDC, and MBICP will be possible.

The second contribution of this paper is purely algorithmic: Appendix II presents an optimized algorithm for computing correspondences. On the test data, the use of the point-to-line metric and the algorithmic optimizations allow PLICP to run at more than 500 matchings per second on a Pentium IV 1.8GhZ for typical data (360-ray scans).

978-1-4244-1647-9/08/$25.00 ©2008 IEEE.

II. The PLICP algorithm

The input data are: a reference scan $\boldsymbol{y}_{t-1}$, a second scan $\boldsymbol{y}_t$, and $\boldsymbol{q}_0$, a first guess for the roto-translation to be found. The reference surface $\mathcal{S}^{\text{ref}}$ is created from the first scan $\boldsymbol{y}_{t-1}$: $\mathcal{S}^{\text{ref}}$ is a polyline obtained by connecting sufficiently close points (using a threshold).

In the following, the index i refers to the points in the scan $\boldsymbol{y}_t$, while the index j refers to the points in $\boldsymbol{y}_{t-1}$. The index k refers to the iterations of the algorithm.

Repeat for $k \geq 0$ until convergence or loop detected:

1 – Compute the coordinates of the second scan's points in the first scan's frame of reference, according to the current guess $\boldsymbol{q}_k = (\boldsymbol{t}_k, \theta_k)$. Point $\boldsymbol{p}_i$ is transformed into $\boldsymbol{p}_i^w$ as follows:

$$\boldsymbol{p}_i^w \triangleq \boldsymbol{p}_i \oplus \boldsymbol{q}_k = \mathbf{R}(\theta_k)\,\boldsymbol{p}_i + \boldsymbol{t}_k \tag{4}$$

2 – For each point $\boldsymbol{p}_i^w$, find the two closest points in the first scan; call their indexes j_1^i e j_2^i. Call $\boldsymbol{C}_k$ all the point-to-segment correspondences at step k. $\boldsymbol{C}_k$ can be written as a set of tuples $\langle i, j_1^i, j_2^i \rangle$, meaning: point i is matched to segment j_1^i–j_2^i.

3 – Use a trimming procedure [9] to eliminate outliers.

4 – Rewrite the error function (3) as:

$$J(\boldsymbol{q}_{k+1}, \boldsymbol{C}_k) = \sum_i \left(\boldsymbol{n}_i^{\mathrm{T}} \left[\mathbf{R}(\theta_{k+1})\boldsymbol{p}_i + \boldsymbol{t}_{k+1} - \boldsymbol{p}_{j_1^i} \right] \right)^2 \tag{5}$$

This is the sum of the squares of the distances from point i to the line containing the segment j_1^i–j_2^i.

5 – To obtain $\boldsymbol{q}_{k+1}$, minimize the error function (5) using the algorithm in Appendix *I*.

III. PLICP has quadratic convergence

The experiments discussed in Section V show that PLICP needs much less iterations than ICP or MBICP: the theoretical justification is that using a point-to-line metric gives quadratic convergence instead of linear convergence.

The following results have been proved in Pottman *et al.* [10]:

Proposition 1: (Pottman *et al*) *Properties of the* ICP *algorithm with* ***point-to-point*** *metric:*

- *The error metric is always decreasing; therefore, a local minimum is always reached.*
- *The* ICP *algorithm, in general, exhibits linear convergence:*

$$\|\boldsymbol{q}_k - \boldsymbol{q}_\infty\| < c \left\|\boldsymbol{q}_{k-1} - \boldsymbol{q}_\infty\right\| \tag{6}$$

for some constant $c \in (0, 1)$. The constant c depends on the direction from which the local minimum is reached. The constant can be computed (if one knows the solution), but can also be estimated from the data.

Proposition 2: (Pottman *et al*) *Properties of the* ICP *algorithm with* ***point-to-line*** *metric:*

- *Minimizing the point-to-line metric is equivalent to a Gauss-Newton iteration.*

TABLE I

SYMBOLS USED IN THIS PAPER

	in the source code	*meaning*
$\boldsymbol{y}_{t-1}$	laser_ref	first scan
$\boldsymbol{y}_t$	laser_sens	second scan
$\boldsymbol{q}$	x	pose (to estimate) $= (\boldsymbol{t}, \theta)$
$\boldsymbol{q}_0$	params–>first_guess	first guess for the pose
$\oplus$	oplus_d()	roto-translation: $\boldsymbol{p} \oplus (\boldsymbol{t}, \theta) \triangleq \mathbf{R}(\theta)\boldsymbol{p} + \boldsymbol{t}$
i	i	index over $\boldsymbol{y}_t$'s points
j	j	index over $\boldsymbol{y}_{t-1}$'s points
$\boldsymbol{p}_i$	laser_sens–>points[i].p	i-th point of $\boldsymbol{y}_t$ (cartesian)
$\boldsymbol{p}_i^w$	laser_sens–>points_w[i].p	i-th point of $\boldsymbol{y}_t$, in $\boldsymbol{y}_{t-1}$'s frame
$\boldsymbol{p}_j$	laser_ref–>points[j].p	j-th point of $\boldsymbol{y}_{t-1}$ (cartesian)
φ_j	laser_ref–>theta[j]	direction of j-th ray
ρ_j	laser_ref–>readings[j]	reading for j-th ray
$\boldsymbol{C}$	laser_sens–>corr	point-to-segment correspondences
j_1^i	laser_sens–>corr[i].j1	best match for $\boldsymbol{p}_i^w$
j_2^i	laser_sens–>corr[i].j2	second best match
$\boldsymbol{n}_i$		normal to the segment $(j_1^i - j_2^i)$
$\mathrm{M}(\boldsymbol{q})$	find_correspondences()	computes $\boldsymbol{C}_k$
$\mathrm{S}(\boldsymbol{C})$	compute_next_estimate()	computes $\boldsymbol{q}_{k+1}$

- *The algorithm converges quadratically in the case of a zero-residual problem, and a good first guess:*

$$\|\boldsymbol{q}_k - \boldsymbol{q}_\infty\|^2 < c \left\|\boldsymbol{q}_{k-1} - \boldsymbol{q}_\infty\right\|^2 \tag{7}$$

Some caveats are necessary. These theoretical results are obtained by considering idealized versions of the algorithms, which do not contain the necessary smart little tweaks ('hacks') that make them work in practice. In reality, for example, the error function might not always be decreasing if some correspondences are discarded as outliers; moreover, the proof of Proposition 1 assumes that the reference surface has bounded derivatives, which is not true in the case of the polyline. Nevertheless, these results show, at least, that the point-to-line metric is superior in the best case (zero-residual, good first guess) to the point-to-point metric.

IV. PLICP converges in a finite number of steps

PLICP converges in a finite number of steps, if the reference surface is a polyline. To prove this, it is convenient to take a somewhat different view on the iteration process.

One can model the scan-matching process as the iterative application of two maps:

1) a "match" function M, that maps a pose $\boldsymbol{q}_k$ to a set of point-to-segment correspondences $\boldsymbol{C}_k$:

$$\boldsymbol{q}_k \xrightarrow{\mathrm{M}} \boldsymbol{C}_k \tag{8}$$

2) a "solve" function S, that minimizes the error metric created from the correspondences:

$$\boldsymbol{C}_k \xrightarrow{\mathrm{S}} \boldsymbol{q}_{k+1} \tag{9}$$

The property allowing such a decomposition is that the error function J in (5) depends on the current estimate $\boldsymbol{q}_k$ *only through* the correspondences set $\boldsymbol{C}_k$: all that matters for defining the point-to-line error function is the segment to which each point $\boldsymbol{p}_i$ is matched (Fig. 1c). For vanilla ICP, instead, the point-to-point metric depends explicitly on $\Pi\{\mathcal{S}^{\text{ref}}; \boldsymbol{p}_i \oplus \boldsymbol{q}_k\}$, the closest point to $\boldsymbol{p}_i$ on $\mathcal{S}^{\text{ref}}$ (Fig. 1b).

With this notation, the algorithm is the consecutive application of "match" and "solve":

$$\boldsymbol{q}_0 \overset{\text{M}}{\to} \boldsymbol{C}_0 \overset{\text{S}}{\to} \boldsymbol{q}_1 \overset{\text{M}}{\to} \boldsymbol{C}_1 \overset{\text{S}}{\to} \boldsymbol{q}_2 \to \dots \tag{10}$$

The common way to think of this process is as an iteration between poses, by grouping together M and S in a composed operation S∘M (first M, then S):

$$\boldsymbol{q}_0 \overset{\text{S}\circ\text{M}}{\to} \boldsymbol{q}_1 \overset{\text{S}\circ\text{M}}{\to} \boldsymbol{q}_2 \to \dots \tag{11}$$

However, in this occasion, it is more convenient to group these maps in the other way: first S, then M; in this way, the computation appears as an iteration between correspondences, through the application of M∘S:

$$\boldsymbol{q}_0 \overset{\text{M}}{\to} \boldsymbol{C}_0 \overset{\text{M}\circ\text{S}}{\to} \boldsymbol{C}_1 \overset{\text{M}\circ\text{S}}{\to} \boldsymbol{C}_2 \to \dots \tag{12}$$

The two descriptions of the matching process are equivalent, but the second one has an advantage: the number of possible point-to-segment correspondences $\boldsymbol{C}_k$ might be very large, but finite nonetheless, while the poses $\boldsymbol{q}_k$ live in a continuous space. From an application of a trivial result of dynamical systems theory, it follows that the iterations will eventually stabilize on a finite orbit.

Proposition 3: The PLICP *algorithm converges in a finite number of steps to either a fixed point or a loop.*

More formally, consider the iterations written as

$$\boldsymbol{C}_{k+1} = [\text{M}\circ\text{S}]\,(\boldsymbol{C}_k) \tag{13}$$

Then there exists a finite n, and a $\delta \geq 1$ ($\delta = 1$ for a fixed point, $\delta > 1$ for a loop) such that

$$\boldsymbol{C}_n = [\text{M}\circ\text{S}]^{\delta}(\boldsymbol{C}_n) \tag{14}$$

This stable set of correspondences $\boldsymbol{C}_n$ corresponds to an equally stable point $\boldsymbol{q}_{n+1} = \text{S}(\boldsymbol{C}_n)$ such that

$$\boldsymbol{q}_{n+1} = [\text{S}\circ\text{M}]^{\delta}\big(\boldsymbol{q}_{n+1}\big) \tag{15}$$

V. EXPERIMENTS

A comparative study of the IDC, ICP, and MBICP has been presented in [6]. Thanks to the authors, it was possible to obtain the data file used and therefore to perform exactly the same experiments. In the following, the proposed algorithms will be compared specifically to the MBICP, as [6] shows that IDC and ICP have worse performance.

The vehicle is a robotic wheel-chair; the sensor is a Sick range-sensor giving 360 rays over a 180° field of view. The scans are taken at an interval of around 0.3m; considerable odometry slip is present.

A. Experiment with artificial errors

The first experiment performed in [6] is used to quantitatively compare the methods' accuracy and realignment interval. The following is the simulation algorithm:

For each scan $\mathcal{S}$:

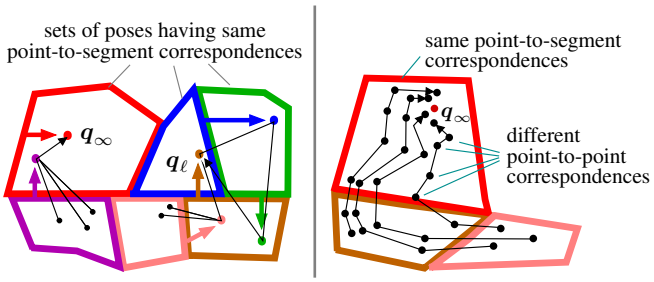
1) *Set $\mathcal{S}^{ref} = \mathcal{S}$.*
2) *Sample a displacement $\boldsymbol{\delta}$ from an error distribution.*
3) *Set $\mathcal{S}^{sens} = \mathcal{S}$ roto-translated by $\boldsymbol{\delta}$.*
4) *Run the scan matching algorithm with input $\mathcal{S}^{ref}, \mathcal{S}^{sens}$; let $\hat{\boldsymbol{q}}$ be the final estimate.*
5) *Because the true $\boldsymbol{q}$ is $(0, 0, 0°)$, assume $\hat{\boldsymbol{q}}$ to be the error.*

Six experiments are presented with increasing initial displacement error (uniform distribution; from $[\pm 0.05\text{m}, \pm 0.05\text{m}, \pm 2°]$ in Experiment 1 to $[\pm 0.2\text{m}, \pm 0.2\text{m}, \pm 45°]$ in Experiment 6). For every experiment, the above procedure is repeated 100 times for each of the 778 scans.

There are pros and cons to this experimental setting. This experiment is ingenious because it allows for using a real-world data-set without the need of a ground-truth. Artificial data can be readily created with the aid of simulators (for example, Stage [11]), however common experience teaches us that the ray-traced bitmap-maps cannot recreate a realistic scenario. One problem with this approach is that it is unrealistic that a scan be matched against itself, as the common situation is that the scans overlap only partially.

Results of PLICP *:* The table in Fig. 3 shows the results. Error samples are sorted into buckets according to the maximum absolute value of their components, expressed in meters and radians. The first three columns are taken straight from [6] and show the results for MBICP, ICP, and IDC.

The fourth column shows the results for PLICP. It is very precise: when the error is in the < 0.001 bucket, the error is actually $(0, 0, 0)$ to machine precision. In fact, because $\mathcal{S}^{\text{ref}} = \mathcal{S}^{\text{sens}}$, PLICP finds the unique global minimum. However, PLICP is less robust to big rotational displacements, also less robust than IDC and ICP. See, for example, Experiment 6 where for 25% of the times PLICP obtains a large error. On the other hand, a 45° odometry error between successive scans could be considered unusual for indoor robotics – as a matter of fact, in this particular log, with scans taken every



(a) PLICP behavior (b) ICP behavior

Fig. 2. This figure illustrates the difference in convergence behavior of PLICP and ICP. Establishing point-to-segments correspondences induces a partition of the state space in discrete regions $\mathbf{Q}_i = \{\, \boldsymbol{q} \mid \text{M}(\boldsymbol{q}) = \boldsymbol{C}_i \,\}$. The points in each $\mathbf{Q}_i$ produce the same correspondences $\boldsymbol{C}_i$: since the next PLICP iteration depends only on the matching information, the successor is unique for each region (bold arrows). PLICP can converge either to a stable point ($\boldsymbol{q}_\infty$) or be captured in a loop (from $\boldsymbol{q}_\ell$ in the figure). Vanilla ICP has a very different behavior. The next step in ICP depends on the particular points matched inside the segments, so ICP allows different trajectories in every $\boldsymbol{C}_i$ area**.** Moreover, because the computation is stopped when the motion is smaller than a threshold, in general the "stable" solution $\boldsymbol{q}_\infty$ is never reached.

	Method	MbICP	IDC	ICP	PLICP	GPM	GPM ∘ PLICP
	Precision (m,rad)	(%)	(%)	(%)	(%)	(%)	(%)
Experiment 1 (0.05m, 0.05m, 2°)	< 0.001	81.27	83.31	57.78	99.85	1.86	99.98
	$(0.001, 0.005)$	18.72	16, 68	42.22	0.01	36.34	0
	$(0.005, 0.01)$	0.00	0.00	0.00	0.01	38.42	0.01
	$(0.01, 0.05)$	0.00	0.01	0.00	0.13	23.37	0.01
	> 0.05	0.00	0.00	0.00	0.00	0.01	0
Experiment 2 (0.10m, 0.10m, 4°)	< 0.001	80.97	83.12	56.62	99.71	1.3	99.98
	$(0.001, 0.005)$	19.02	16.84	42.48	0.02	24.56	0.00
	$(0.005, 0.01)$	0.00	0.00	0.00	0.03	29.12	0.01
	$(0.01, 0.05)$	0.00	0.03	0.00	0.22	42.85	0.01
	> 0.05	0.00	0.00	0.00	0.02	2.17	0.00
Experiment 3 (0.15m, 0.15m, 8.6°)	< 0.001	80.84	82.95	56.62	99.51	0.91	99.95
	$(0.001, 0.005)$	19.15	16.96	43.37	0.03	18.64	0.01
	$(0.005, 0.01)$	0.00	0.00	0.00	0.05	24.58	0.02
	$(0.01, 0.05)$	0.00	0.05	0.00	0.33	50.16	0.02
	> 0.05	0.00	0.03	0.002	0.08	5.71	0
Experiment 4 (0.20m, 0.20m, 17.2°)	< 0.001	81.28	81.96	56.30	98.43	0.61	99.79
	$(0.001, 0.005)$	18.71	16.79	43.58	0.088	14.05	0.01
	$(0.005, 0.01)$	0.00	0.00	0.00	0.13	21.79	0.02
	$(0.01, 0.05)$	0.00	0.80	0.00	0.44	54.25	0.07
	> 0.05	0.00	0.44	0.10	0.92	9.3	0.11
Experiment 5 (0.20m, 0.20m, 32°)	< 0.001	80.92	79.54	54.00	84.48	0.61	99.79
	$(0.001, 0.005)$	18.79	16.36	43.13	0.20	14.05	0.01
	$(0.005, 0.01)$	0.0	0.04	0.00	0.28	21.79	0.02
	$(0.01, 0.05)$	0.0	0.81	0.00	0.93	54.25	0.07
	> 0.05	0.28	3.05	2.85	14.11	9.3	0.11
Experiment 6 (0.20m, 0.20m, 45°)	< 0.001	80.38	74.94	52.18	73.46	0.61	99.79
	$(0.001, 0.005)$	18.86	16.53	42.01	0.23	14.05	0.01
	$(0.005, 0.01)$	0.00	0.37	0.00	0.35	21.79	0.02
	$(0.01, 0.05)$	0.00	0.81	0.01	1.14	54.25	0.07
	> 0.05	0.75	7.32	5.78	24.81	9.3	0.11

PLICP is the most precise.

Some annoying not-so-constrained scans.

GPM's results are constant as they are not influenced much by the initial guess.

...therefore also the results of PLICP∘GPM are constant.

Good results thanks to the realigment of GPM and the precision of PLICP.

Fig. 3. The results for MbICP, IDC, ICP, are taken from [6].

0.3m, the maximum odometry error was 25°.

Using GPM *as a first guess, then* PLICP *:* To overcome the realignment problems of PLICP, one possibility is to use a global algorithm for a quick first coarse realignment, and then use PLICP for the final convergence. Here, a stripped-down version of GPM [12] was used (without weighting, and only one iteration). This seems a winning match, because GPM is not sensible to big displacements, and then PLICP is able to converge very quickly to the solution. Consider Experiment 6: the combination of the two algorithms attains the best accuracy for 99.79% of the trials.

B. One anecdote

The second experiment performed in [6] involves the visual inspection of the reconstructed scan-matched map for the given sensor log. The map reconstructed by PLICP is virtually indistinguishable from the one shown in [6]; it is not shown here for reasons of space.

The following table shows the average number of iterations and the average execution time on a Pentium IV 1.8GhZ. Results for MbICP, ICP, and IDC are copied from [6].

	avg. iterations	*avg. execution time*
MbICP	31.2	0.076 s (13.1 Hz)
ICP	34.7	0.083 s (12.0 Hz)
IDC	30.4	0.240 s (4.1 Hz)
PLICP	7.2	0.0018 s (539 Hz)

The iterations needed by PLICP are much less than those needed by MbICP, ICP and IDC. As for the absolute timing, take it with a grain of salt, as it heavily depends on the implementation of the methods.

VI. Conclusions and Future Work

This paper presented PLICP: a simple ICP variant which uses a point-to-line metric. Thanks to a closed-form minimization, PLICP outperforms MbICP, vanilla ICP, and IDC for accuracy, number of iterations, and raw speed. Moreover, two interesting properties can be proved: 1) PLICP converges quadratically, and 2) PLICP converges in a finite number of steps. To the author's knowledge, PLICP is the only ICP variant used in robotics localization for which such properties have been shown. It is likely that future work will lead to more stringent properties on the number of iterations needed, and, hopefully, to an upper bound as a function of the initial error.

PLICP needs less parameters than the vanilla ICP: it does not need the thresholds $\epsilon_{\boldsymbol{t}}, \epsilon_\theta$ for stopping the convergence, nor it needs an estimate $\max |\boldsymbol{t}|$ and $\max |\theta|$ for efficiently searching the correspondences. The only magic numbers still needed are those governing the rejection/trimming of correspondences.

The experiments show that PLICP is definitely the best option to quickly converge from a good first guess. However, it does not handle well severe odometry errors; in particular,

it is less robust than MBICP to large rotations: this is not surprising, as MBICP has been explicitly designed to handle such cases. One possible solution investigated is to use a quick fast global algorithm like GPM for a first realignment, and then finish the job with PLICP.

There are still many degrees of freedom that could be explored. As explained in Appendix I, one can easily mix the point-to-point and point-to-line metric, and different weights can be attributed to the correspondences. Experimenting with such possibilities is likely to improve the realignment properties.

The data-set and C source-code for the full scan matcher are available at `http://purl.org/censi/2007/csm`. Source code for the closed-form minimization is available also separately in the C, Ruby, and Matlab/Octave languages at `http://purl.org/censi/2007/gpc` .

APPENDIX I
CLOSED-FORM SOLUTION FOR THE 2-D GENERALIZED METRIC

Consider the following non-linear minimization problem:

$$\min_{\boldsymbol{t},\theta} \sum_i \|(\mathbf{R}(\theta)\boldsymbol{p}_i + \boldsymbol{t}) - \boldsymbol{\pi}_i\|^2_{\mathbf{C}_i} \tag{16}$$

where $\boldsymbol{p}_i, \boldsymbol{\pi}_i, \boldsymbol{t} \in \mathbb{R}^2$; $\mathbf{R}(\theta)$ is the 2×2 rotation matrix, and the norm $\|\boldsymbol{a}\|^2_{\mathbf{C}}$ is defined as $\boldsymbol{a}^{\mathrm{T}}\mathbf{C}\boldsymbol{a}$. This is a general form that encompasses both the point–to–point metric ($\mathbf{C}_i = w_i\mathbf{I}_{2\times 2}$, w_i being a weight), and point-to-line metric ($\mathbf{C}_i = w_i\boldsymbol{n}_i\boldsymbol{n}_i^{\mathrm{T}}$, where $\boldsymbol{n}_i \in \mathbb{R}^2$ is the versor normal to the line).

A. Reduction to quadratic form

A closed-form for the solution can be found. The three-dimensional solution (t_x, t_y, θ) will be found in the four-dimensional space $\boldsymbol{x} = [x_1, x_2, x_3, x_4] \triangleq [t_x, t_y, \cos\theta, \sin\theta]$ by imposing the constraint $x_3^2 + x_4^2 = 1$.

Given the vector $\boldsymbol{p}_i = (p_{i0}, p_{i1})$, define the following matrix $\mathbf{M}_i$:

$$\mathbf{M}_i = \begin{bmatrix} 1 & 0 & p_{i0} & -p_{i1} \\ 0 & 1 & p_{i1} & p_{i0} \end{bmatrix} \tag{17}$$

Then (16) can be written in matrix form as

$$\min_{\boldsymbol{x}} \sum_i (\mathbf{M}_i\boldsymbol{x} - \boldsymbol{\pi}_i)^{\mathrm{T}} \mathbf{C}_i (\mathbf{M}_i\boldsymbol{x} - \boldsymbol{\pi}_i) \tag{18}$$

The following step simplifies the expression to obtain a simple quadratic form. Expand (18) to obtain

$$\sum_i (\mathbf{M}_i\boldsymbol{x} - \boldsymbol{\pi}_i)^{\mathrm{T}} \mathbf{C}_i (\mathbf{M}_i\boldsymbol{x} - \boldsymbol{\pi}_i) =$$
$$\sum_i (\boldsymbol{x}^{\mathrm{T}}\mathbf{M}_i^{\mathrm{T}}\mathbf{C}_i\mathbf{M}_i\boldsymbol{x} + \boldsymbol{\pi}_i^{\mathrm{T}}\mathbf{C}_i\boldsymbol{\pi}_i - 2\boldsymbol{\pi}_i^{\mathrm{T}}\mathbf{C}_i\mathbf{M}_i\boldsymbol{x})$$

Ignoring the constant terms, the new function to minimize is

$$\boldsymbol{x}^{\mathrm{T}} \underbrace{\left(\sum_i \mathbf{M}_i^{\mathrm{T}}\mathbf{C}_i\mathbf{M}_i\right)}_{\mathbf{M}} \boldsymbol{x} + \underbrace{\left(\sum_i -2\boldsymbol{\pi}_i^{\mathrm{T}}\mathbf{C}_i\mathbf{M}_i\right)}_{\boldsymbol{g}} \boldsymbol{x} \tag{19}$$

By defining the matrix $\mathbf{W} = \begin{bmatrix} \mathbf{0}_{2\times 2} & \mathbf{0}_{2\times 2} \\ \mathbf{0}_{2\times 2} & \mathbf{I}_{2\times 2} \end{bmatrix}$, the constraint $x_3^2 + x_4^2 = 1$ can be written as $\boldsymbol{x}^{\mathrm{T}}\mathbf{W}\boldsymbol{x} = 1$, and the problem becomes:

$$\min_{\boldsymbol{x}} \boldsymbol{x}^{\mathrm{T}}\mathbf{M}\boldsymbol{x} + \boldsymbol{g}^{\mathrm{T}}\boldsymbol{x} \tag{20}$$

$$\text{subject to} \quad \boldsymbol{x}^{\mathrm{T}}\mathbf{W}\boldsymbol{x} = 1 \tag{21}$$

B. Solution using Lagrange's multipliers

The solution will be derived using Lagrange's multipliers. Define the function $\mathcal{L}(\boldsymbol{x})$:

$$\mathcal{L}(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}\mathbf{M}\boldsymbol{x} + \boldsymbol{g}^{\mathrm{T}}\boldsymbol{x} + \lambda(\boldsymbol{x}^{\mathrm{T}}\mathbf{W}\boldsymbol{x} - 1) \tag{22}$$

Necessary condition for optimality is that ($\partial\mathcal{L}/\partial\boldsymbol{x} = \mathbf{0}^{\mathrm{T}}$):

$$2\boldsymbol{x}^{\mathrm{T}}\mathbf{M} + \boldsymbol{g}^{\mathrm{T}} + 2\lambda\boldsymbol{x}^{\mathrm{T}}\mathbf{W} = \mathbf{0}^{\mathrm{T}} \tag{23}$$

This is equivalent to:

$$\boldsymbol{x} = -\left(2\mathbf{M} + 2\lambda\mathbf{W}\right)^{-\mathrm{T}} \boldsymbol{g} \tag{24}$$

If one puts relation (24) into the constraint (21), one obtains the following expression, in which the only unknown is λ:

$$\boldsymbol{g}^{\mathrm{T}} \left(2\mathbf{M} + 2\lambda\mathbf{W}\right)^{-1} \mathbf{W} \left(2\mathbf{M} + 2\lambda\mathbf{W}\right)^{-\mathrm{T}} \boldsymbol{g} = 1 \tag{25}$$

Even if not obvious at first sight, (25) is a fourth-order polynomial in λ. The following computations will prove this assertion. Partition the matrix $(2\mathbf{M} + 2\lambda\mathbf{W})$ into four sub-matrices:

$$2\mathbf{M} + 2\lambda\mathbf{W} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^{\mathrm{T}} & \mathbf{D} + 2\lambda\mathbf{I} \end{bmatrix} \tag{26}$$

In the middle of the quadratic form (25) there is the matrix $\mathbf{W}$: that matrix has a sparse form, therefore one needs to compute only the last column of $(2\mathbf{M} + 2\lambda\mathbf{W})^{-1}$. Using the matrix inversion lemma, one obtains that $(2\mathbf{M} + 2\lambda\mathbf{W})^{-1} =$

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^{\mathrm{T}} & (\mathbf{D} + 2\lambda\mathbf{I}) \end{bmatrix}^{-1} = \begin{bmatrix} * & -\mathbf{A}^{-1}\mathbf{B}\mathbf{Q}^{-1} \\ * & \mathbf{Q}^{-1} \end{bmatrix} \tag{27}$$

where $\mathbf{Q} = \left(\mathbf{D} - \mathbf{B}^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{B} + 2\lambda\mathbf{I}\right) \triangleq (\mathbf{S} + 2\lambda\mathbf{I})$. The constraint (21) now appears as

$$\boldsymbol{g}^{\mathrm{T}} \begin{bmatrix} \mathbf{A}^{-1}\mathbf{B}\mathbf{Q}^{-1}\mathbf{Q}^{-\mathrm{T}}\mathbf{B}^{\mathrm{T}}\mathbf{A}^{-\mathrm{T}} & -\mathbf{A}^{-1}\mathbf{B}\mathbf{Q}^{-1}\mathbf{Q}^{-\mathrm{T}} \\ (\text{symm}) & \mathbf{Q}^{-1}\mathbf{Q}^{-\mathrm{T}} \end{bmatrix} \boldsymbol{g} = 1 \tag{28}$$

Now write $\mathbf{Q}$ in this way:

$$\mathbf{Q} = (\mathbf{S} + 2\lambda\mathbf{I})^{-1} = \frac{\mathbf{S}^A + 2\lambda\mathbf{I}}{p(\lambda)} \tag{29}$$

where $\mathbf{S}^A = \det(\mathbf{S}) \cdot \mathbf{S}^{-1}$ and $p(\lambda) = \det(\mathbf{S} + 2\lambda\mathbf{I})$. Since $\mathbf{Q}^{-1}\mathbf{Q}^{-\mathrm{T}} =$

$$\frac{\left(\mathbf{S}^A + 2\lambda\mathbf{I}\right)\left(\mathbf{S}^A + 2\lambda\mathbf{I}\right)^{\mathrm{T}}}{p(\lambda)^2} = \frac{\mathbf{S}^A\mathbf{S}^{A^{\mathrm{T}}} + 4\lambda^2\mathbf{I} + 4\lambda\mathbf{S}^A}{p(\lambda)^2} \tag{30}$$

One finally obtains the following polynomials:

$$\lambda^2 \cdot 4\boldsymbol{g}^{\mathrm{T}} \begin{bmatrix} \mathbf{A}^{-1}\mathbf{B}\mathbf{B}^{\mathrm{T}}\mathbf{A}^{-\mathrm{T}} & -\mathbf{A}^{-1}\mathbf{B} \\ (\mathrm{symm}) & \mathbf{I} \end{bmatrix} \boldsymbol{g} \quad + \tag{31}$$

$$\lambda \cdot 4\boldsymbol{g}^{\mathrm{T}} \begin{bmatrix} \mathbf{A}^{-1}\mathbf{B}\mathbf{S}^{A}\mathbf{B}^{\mathrm{T}}\mathbf{A}^{-\mathrm{T}} & -\mathbf{A}^{-1}\mathbf{B}\mathbf{S}^{A} \\ (\mathrm{symm}) & \mathbf{S}^{A} \end{bmatrix} \boldsymbol{g} \quad +$$

$$\boldsymbol{g}^{\mathrm{T}} \begin{bmatrix} \mathbf{A}^{-1}\mathbf{B}\mathbf{S}^{A^{\mathrm{T}}}\mathbf{S}^{A}\mathbf{B}^{\mathrm{T}}\mathbf{A}^{-\mathrm{T}} & -\mathbf{A}^{-1}\mathbf{B}\mathbf{S}^{A^{\mathrm{T}}}\mathbf{S}^{A} \\ (\mathrm{symm}) & \mathbf{S}^{A^{\mathrm{T}}}\mathbf{S}^{A} \end{bmatrix} \boldsymbol{g} \quad = \quad [p(\lambda)]^2$$

Because $p(\lambda)$ is a second-order polynomial, the order of (31) is 4, therefore the solution can be found in closed form. After one has found λ, one can obtain $\boldsymbol{x}$ using (24).

APPENDIX II
FAST CORRESPONDENCE SEARCH

Much of the speed of PLICP comes from a smart implementation of the correspondence-search procedure. This appendix describes it in detail, because it is not specific to PLICP, but can be re-utilized in other ICP-like algorithms as well. As a notable exception, it cannot be utilized in MBICP, because that algorithm uses a different metric for establishing the point correspondences.

Once again, the goal of the correspondence-search algorithm is, for each point $\boldsymbol{p}_i^w$ in the scan $\boldsymbol{y}_t$ to find its closest point $\boldsymbol{p}_j$ in the scan $\boldsymbol{y}_{t-1}$ (recall that the points $\boldsymbol{p}_i^w$ are the points of $\boldsymbol{y}_t$ expressed in $\boldsymbol{y}_{t-1}$'s frame). Call j_1^i the index of such point:

$$j_1^i = \arg\min_j \left\| \boldsymbol{p}_j - \boldsymbol{p}_i^w \right\|^2 \tag{32}$$

The naive algorithm for solving this problem is considering all possible points in sequence. One optimization is possible if the points in $\boldsymbol{y}_{t-1}$ have a radial ordering. In that case, the search can be restricted to only the points in the φ interval $(\angle \boldsymbol{p}_i^w - \Delta\varphi, \angle \boldsymbol{p}_i^w + \Delta\varphi)$; if $\max|\boldsymbol{t}|$ and $\max|\theta|$ are known a priori, an upper bound for $\Delta\varphi$ is

$$|\Delta\varphi| \leq \tan^{-1} \frac{\max|\boldsymbol{t}|}{||\boldsymbol{p}_i^w||} + \max|\theta| \tag{33}$$

A. Many little tricks make a smart algorithm

The smart algorithm is the union of several heuristics. The only assumption needed is that of the radial ordering for the points in $\boldsymbol{y}_{t-1}$. There are no magic numbers for this algorithm; one does not need to know $\max|\boldsymbol{t}|$ and $\max|\theta|$. The source code is shown in Fig. 5 — well, a reasonably simplified version: see file icp/icp_corr_tricks.c in the software package for the actual source code.

(line 11 in Fig. 5) The search for point i starts from point j_1^{i-1}, that is the correspondence found for the previous point $\boldsymbol{p}_{i-1}^w$. This helps because, being point $\boldsymbol{p}_i^w$ and $\boldsymbol{p}_{i-1}^w$ close, also their corresponding points will be close to each other.

(Fig. 4(b); line 22 in Fig. 5) The search proceeds in two directions: for increasing and decreasing values of φ, indicated, respectively, with the indices up and down. The search alternates between the two, always choosing the direction which looks more promising, by comparing the distance to $\boldsymbol{p}_i^w$ in the up direction (last_dist_up) and in the down direction (last_dist_down).

(Fig. 4(c); line 34–39 in Fig. 5) This is an early-stopping criterion. Consider the search in the up direction, when $\varphi_{\text{up}} > \angle \boldsymbol{p}_i^w$, that is, the index is "getting away" from $\boldsymbol{p}_i^w$. In this case (line 35), one can compute min_dist_up, the minimum distance to $\boldsymbol{p}_i^w$ for the points $\boldsymbol{p}_j$, with $j >$ up. If min_dist_up is bigger than the current best_dist, then the search can be stopped in the up direction (line 38).

The distance min_dist_up can be computed by geometric inspection of Fig. 4(c). For $j >$ up, one has that $\delta\varphi \geq \varphi_{\text{up}} - \angle \boldsymbol{p}_i^w$, and therefore $||\boldsymbol{p}_j - \boldsymbol{p}_i^w|| \geq \sin(\delta\varphi)||\boldsymbol{p}_i^w||$. One does not need to compute the exact sine of the angle; any approximation $f(x)$ will do, as long as it is 'optimistic' ($f(x) \leq \sin(x)$).

(Fig. 4(d); line 41–44 in Fig. 5) Finally, the following optimization allows to jump over "uninteresting" points.

For this to work, one need to precompute some "jump tables" for the scan $\boldsymbol{y}_{t-1}$ (only once, they can be re-used over ICP iterations); these tables, in practice, are very cheap to compute (less than 1% of the execution time).

For each point j in $\boldsymbol{y}_{t-1}$, compute up_smaller[j] (up_bigger[j]) as the first point with index $j' > j$ for which the reading $\rho_{j'}$ is smaller (bigger) than ρ_j:

$$\begin{aligned} \text{up\_smaller[j]} &= \min \{\, j' > j \mid \rho_{j'} < \rho_j \,\} \\ \text{up\_bigger[j]} &= \min \{\, j' > j \mid \rho_{j'} > \rho_j \,\} \end{aligned}$$

Consider again the search in the up direction, when $\varphi_{\text{up}} > \angle \boldsymbol{p}_i^w$. If the current point $\boldsymbol{p}_{\text{up}}$ is farther than the best point so far $\boldsymbol{p}_{\text{best}}$, and $\rho_{\text{up}} \geq ||\boldsymbol{p}_i^w||$, then one can conclude that no point $\boldsymbol{p}_j$, with $j >$ up and $\rho_j \geq \rho_{\text{up}}$ can do better, and then the search can directly jump to the point up_smaller[up], the first point with $\rho < \rho_{\text{up}}$. This allows to ignore the points between up and up_smaller[up]. In the case of $\rho_{\text{up}} \leq ||\boldsymbol{p}_i^w||$, the search jumps to up_bigger[up].

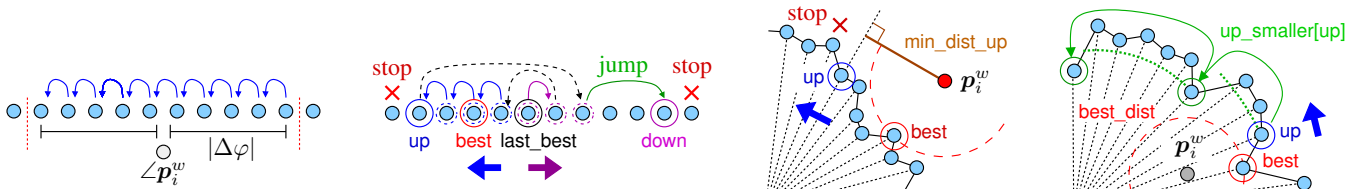
All this procedure is repeated, mutatis mutandis, for the search in direction down (line 50).

B. Qualitative and quantitative comparison

A measure of the efficiency of the algorithm is the number of times it needs to compute the distance between two points. The test is performed on the same data log used for the other experiments. For the naive algorithm, the parameters used are $\max|\boldsymbol{t}| = 0.5$m e $\max|\theta| = 25°$, which are the actual maximum translation and rotation found in the log (they are the 'optimal' parameters).

number of comparisons	smart	naive
total number	$\sim 12 \cdot 10^6$	$\sim 262 \cdot 10^6$
avg. per ray per iteration	6.0	124.9
iterations per second	539	93

One can see that there is a dramatic reduction of the number of comparisons: on average, the smart algorithm does only 6 comparisons per ray per iteration, and this leads to a considerable overall speed-up.



(a) Naive algorithm (b) Smart algorithm (c) Early stopping (d) Using jump tables

Fig. 4. Fig. 4(a) shows the naive algorithm that serially visits the points in $\boldsymbol{y}_{t-1}$; a bound for $\max|\Delta\varphi|$ can be found by equation (33). The smart algorithm does not visit the points serially. To find j_1^i, it starts at point j_1^{i-1} (last_best in the source code), and searches in both directions at the same time (indices up and down): at each step, it chooses the direction which looks more promising. Then, there are various heuristics for early stopping (Fig. 4(c)), and for ignoring some of the points (Fig. 4(d)).

```
1  // Out of the main loop, we remember the last match found.
2  int last_best = invalid;
3
4  for(each point p_i^w in scan y_t ) {
5
6  // Current best match, and its distance
7      int best = invalid; double best_dist = ∞;
8  // Approximated index in scan y_{t-1} corresponding to point p_i^w
9      int start_index = (∠p_i^w − φ_0) · (nrays/2π);
10 // If last match was succesful, then start at that index + 1
11     int we_start_at = (last_best != invalid) ? (last_best + 1) : start_index;
12 // Search is conducted in two directions: up and down
13     int up = we_start_at+1, down = we_start_at;
14 // Distance of last point examined in the up (down) direction.
15     double last_dist_up = ∞, last_dist_down = ∞;
16 // True if search is finished in the up (down) direction.
17     bool up_stopped = false, down_stopped = false;
18
19 // Until the search is stopped in both directions...
20     while ( ! (up_stopped && down_stopped) ) {
21 // Should we try to explore up or down?
22         bool now_up = !up_stopped & (last_dist_up < last_dist_down);
23 // Now two symmetric chunks of code, the now_up and the !now_up
24         if(now_up) {
25 // If we have finished the points to search, we stop.
26             if(up >= nrays) { up_stopped = true; continue; }
27 // This is the distance from p_i^w to the up point.
28             last_dist_up = ||p_i^w − p_up||^2;
29 // If it is less than the best point, up is our best guess so far.
30             if( correspondence is acceptable && last_dist_up < best_dist)
31                 best = up, best_dist = last_dist_up;
32             if (up > start_index) {
33 // If we are moving away from start_cell we can compute a
   bound for early stopping. Currently our best point has distance
   best_dist; we can compute the minimum distance to p_i^w for
   points j > up (see figure 4(c)).
34                 double Δφ = φ_up − ∠p_i^w;
35                 double min_dist_up = sin(Δφ) ||p_i^w||;
36                 if( [ min_dist_up ]^2 > best_dist) {
37 // If going up we can't make better than best_dist,
   then we stop searching in the "up" direction
38                     up_stopped = true; continue;
39                 }
40 // If we are moving away, then we can implement the jump tables
   optimization.
41                 up = // Next point to examine is...
42                 (ρ_up < ||p_i^w||) ? // is p_i^w longer?
43                   up_bigger[up] // then jump to a further point
44                 : up_smaller[up]; // else, to a closer one.
45             } else
46 // If we are moving towards "start_cell", we can't do any ot the
   previous optimizations and we just move to the next point.
47                 up++;
48         } // if(now_up)
49 // This is the specular part of the previous chunk of code.
50         if(!now_up) { ... }
51     }
52     // Set null correspondence if no point matched.
53     ...
54     // For the next point, we will start at best
55     last_best = best;
56 }
```

Fig. 5. Pseudo-C source code for the smart algorithm. This is basically a prettied-up version of the code in file icp/icp_corr_tricks.c.

References

[1] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Third International Conference on 3D Digital Imaging and Modeling (3DIM)*, June 2001.

[2] F. Lu and E. Milios, "Robot pose estimation in unknown environments by matching 2D range scans," *Journal of Intelligent Robotics Systems*, vol. 18, no. 3, pp. 249–275, 1997.

[3] S. Pfister, K. Kriechbaum, S. Roumeliotis, and J. Burdick, "Weighted range sensor matching algorithms for mobile robot displacement estimation," in *Proceedings of the IEEE International conference on Robotics and Automation (ICRA)*, 2002.

[4] B. Jensen and R. Siegwart, "Scan alignment with probabilistic distance metric," in *Proceedings of the IEEE/RSJ International conference on Intelligent Robots and Systems (IROS)*, (Sendai, Japan), 2004.

[5] L. Montesano, J. Minguez, and L. Montano, "Probabilistic scan matching for motion estimation in unstructured environments," in *Proceedings of the IEEE/RSJ International conference on Intelligent Robots and Systems (IROS)*, (Edmonton, Canada), 2005.

[6] J. Minguez, F. Lamiraux, and L. Montesano, "Metric-based scan matching algorithms for mobile robot displacement estimation," *IEEE Transactions on Robotics*, 2006.

[7] Y. Chen and G. Medioni, "Object modeling by registration of multiple range images," in *Proceedings of the IEEE International conference on Robotics and Automation (ICRA)*, (Sacramento, CA, USA), pp. 2724–2729, Apr. 1991.

[8] 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.

[9] D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek, "The trimmed iterative closest point algorithm," in *Proceedings of the International Conference on Pattern Recognition (ICPR)*, (Quebec, Canada), 2002.

[10] H. Pottmann, Q.-X. Huang, Y.-L. Yang, and S.-M. Hu, "Geometry and convergence analysis of algorithms for registration of 3d shapes," *Internation Journal of Computer Vision*, vol. 67(3), no. 3, pp. 277–296, 2006.

[11] B. P. Gerkey, R. T. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *In Proc. of the Int. Conf. on Advanced Robotics (ICAR 2003)*, (Coimbra, Portugal), pp. pp. 317–323, June 30 - July 3 2003.

[12] A. Censi, "Scan matching in a probabilistic framework," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, (Orlando, Florida), pp. 2291–2296, 2006.
Available from: `http://purl.org/censi/2006/gpm`.