Autonomous Robots manuscript No.
(will be inserted by the editor)

Comparing ICP Variants on Real-World Data Sets

Open-source library and experimental protocol.

François Pomerleau · Francis Colas · Roland Siegwart · Stéphane Magnenat

Received: date / Accepted: date

Abstract Many modern sensors used for mapping produce 3D point clouds, which are typically registered together using the iterative closest point (ICP) algorithm. Because ICP has many variants whose performances depend on the environment and the sensor, hundreds of variations have been published. However, no comparison frameworks are available, leading to an arduous selection of an appropriate variant for particular experimental conditions. The first contribution of this paper consists of a protocol that allows for a comparison between ICP variants, taking into account a broad range of inputs. The second contribution is an open-source ICP library, which is fast enough to be usable in multiple real-world applications, while being modular enough to ease comparison of multiple solutions. This paper presents two examples of these field applications. The last contribution is the comparison of two baseline ICP variants using data sets that cover a rich variety of environments. Besides demonstrating the need for improved ICP methods for natural, unstructured and information-deprived environments, these baseline variants also provide a solid basis to which novel solutions could be compared. The combination of our protocol, software, and baseline results demonstrate convincingly how open-source software can push forward the research in mapping and navigation.

Keywords experimental protocol, iterative closest point, registration, open-source, SLAM, mapping

This work was supported by the EU FP7 IP projects *Natural Human-Robot Cooperation in Dynamic Environments (ICT-247870)* and *myCopter (FP7-AAT-2010-RTD-1)*. F. Pomerleau was supported by a fellowship from the Fonds québécois de recherche sur la nature et les technologies (FQRNT).

François Pomerleau
Autonomous System Lab, ETH Zentrum CLA E26, Tannenstrasse 3, 8092 Zurich, Switzerland
E-mail: francois.pomerleau@mavt.ethz.ch

Francis Colas
Autonomous System Lab, ETH Zentrum CLA E26, Tannenstrasse 3, 8092 Zurich, Switzerland
E-mail: francis.colas@mavt.ethz.ch

Roland Siegwart
Autonomous System Lab, ETH Zentrum CLA E14.2, Tannenstrasse 3, 8092 Zurich, Switzerland
E-mail: rsiegwart@ethz.ch

Stéphane Magnenat
Autonomous System Lab, ETH Zentrum CLA E24, Tannenstrasse 3, 8092 Zurich, Switzerland
E-mail: stephane at magnenat dot net

1 Introduction

Laser-range sensors were a cornerstone to the development of mapping and navigation in the past two decades. Nowadays, rotating laser scanners, stereo cameras or depth cameras (RGB-D) can provide dense 3D point clouds at a high frequency. Using the iterative closest point (ICP) registration algorithm [6, 7], these point clouds can be matched to deduce the transformation between them and, consequently, the 6 degrees of freedom motion of the sensor. Albeit originally proposed for object reconstruction, the robotics field has extensively applied registration for global scene reconstruction. ICP is a popular algorithm due to its simplicity: its general idea is easy to understand and to implement. However, the basic algorithm works well only in ideal cases. This led to hundreds of variations (around 400 papers published in the past 20 years, see Figure 1) around the original algorithm that were demonstrated on different and incommensurable experimental scenarios. This highlights both the usefulness of ICP and the difficulty of finding a versatile version. Because there exists no comparison framework, the selection of an appropriate variant for particular experimental conditions is difficult. This is a major problem because registration is at the front-end of the mapping pipeline, and its selection affects arbitrarily the results of all subsequent steps. There is therefore a need for streamlining the selection of a registration algorithm given a type of environment.

The first contribution of this paper is a protocol to allow comparison between ICP variants. This protocol encompasses an experimental methodology and evaluation metrics, as already proposed in other fields such as stereo correspondence detection [22], multi-view stereo reconstruction [24], optical-flow computation [4, 10] and visual odometry [10]. The performance of ICP algorithms is affected by the type of environment, the trajectory

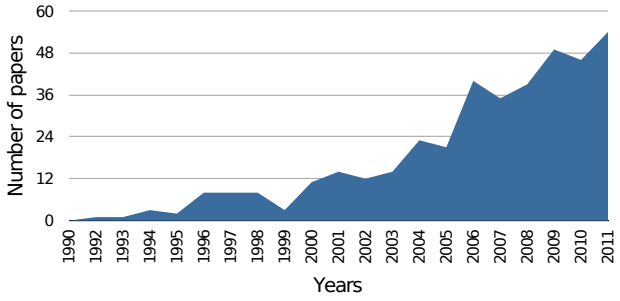

Figure 1 Evolution of the number of publications over the years based on IEEE Xplore. Results were obtained for *Iterative Closest Point* appearing in the abstract or the title of publications.

realized in that environment and the uncertainties of the initial poses. Our protocol provides a consistent way to compare ICP variants in all these conditions.

The second contribution of this paper is an open-source modular ICP library and related helper programs, which allow comparison of several ICP variants within the same framework. This library is based on our optimized implementation of nearest-neighbor search with kd-tree, called libnabo[1]. It is one of the fastest kd-tree libraries for ICP thanks to more compact data structures than rival implementations [9]. Being both modular and fast, our ICP library provides an ideal solution for comparing registration algorithms.

The last contribution of this paper is a revisit of well-established ICP variants using our library and our protocol, using recently published data sets [18] that cover a variety of environments with ground-truth poses. We show that even if the point-to-plane distance metric is in general superior to the point-to-point distance metric, it can be less precise for large disturbances of the initial alignments and loses its advantages in unstructured environments.

2 Related Work

2.1 Overview of ICP

As introduced previously, the body of work related to ICP is very large, and reviewing it is beyond the scope of this paper. We rather focus on the main components of the algorithm as presented in [20]. First, point clouds can be filtered, for example, to remove redundant points or compute descriptors like normals. The Point Cloud Library (PCL) is a good example of state-of-the-art implementations of point cloud filters [21]. Then, a matching function needs to be applied to associate elements from a *reading* point cloud to a *reference* point cloud. This association is usually done in the Euclidean space using kd-tree to accelerate the search [9]. When ICP is applied to robotics, special care needs to be taken to properly handle mismatches or outliers. Different statistics can be used to identify outliers, like removing the higher-distance quantile of all paired points [8]. Finally, the remaining points can be used to minimize the alignment error. The most common distance metrics are point-to-point [6] and point-to-plane [7].

Recently, promising solutions appeared to deal with uncertainty specific to mobile platforms. To name a few, the metric-ICP targets robustness against rotation error [2] while normal distributions transform (NDT) [14] tackles structural uncertainty.

2.2 Registration Benchmarking

The seminal work of Rusinkiewicz and Levoy [20] on the comparison of variants of the ICP algorithm led to significant progress in the field of scan registration. The experiments employ simulated objects, highlighting different spatial constraints and sensor noises. Wulf et al. [26] present an evaluation method for simultaneous localisation and mapping (SLAM) heavily linked to ICP. They compare ICP using pairwise scans and ICP using metascans (i.e., concatenation of past scans) along with full SLAM solutions. They observe that, compared to pairwise match, metascans lead to slower error accumulation but also slow down computational time to a point compromising real-time execution. The authors conclude with the statement that research in robotics benchmarking techniques requires more consideration. The demand for a stronger experimental methodology in robotics is also stressed by Amigoni et al. [1]. The authors survey different SLAM publications in order to highlight proper evaluation metrics that are applied to SLAM algorithms. Three principles of an experimental methodology in science (i.e, comparison, reproducibility/repeatability and justification/explanation) are translated in requirements for stronger SLAM results. As stated in their publication, a sound methodology should allow researchers to gain an insight about *intrinsic* (ex., computational time, parameters used, parameter behaviors) and *extrinsic* (ex., accuracy, precision) quantities. The authors reported that, even though comparisons between algorithms are present in SLAM publications, very few researchers can reuse the same protocol and directly compare their results without having to re-implement other solutions.

Registration quality depends on many external factors. Typically, a single type of environment is selected for evaluation. The latter is mostly urban [17, 26] or well-structured environment, like tunnels [15]. The robustness of registration against initial misalignment is

[1] `http://github.com/ethz-asl/libnabo`, version 1.0.1

explored in [11]. This type of exploration is continued with an evaluation of ICP against NDT in order to compare the valley of convergence of both methods [15]. In the work of Pathak et al. [17], the sensitivity of their registration algorithm to low spatial overlap is identified and used to predict scan-matching failures.

When presenting registration results, authors face the problem of reducing the dimensionality of their results to low-dimension and meaningful performance metrics. Early work mainly focuses on the rapidity of convergence and the final accuracy of different solutions [20]. Typical parameters of interest concern translation and rotation for a total of six dimensions. While summarizing the translation components using the Euclidean distance is commonly accepted, different methods are used for the rotation. The work of Wulf et al. [26] mixes scans in 3D (928 scans over 1 km) with ground-truth poses in 2D. Consequently, the evaluation is done in 2D using Euclidean distance for translation errors and absolute value of the orientation differences. To produce statistics about the overall experiment, the authors propose to use the standard deviation of all errors and the maximum error as evaluation metrics. Doing their evaluation directly in 3D, Tong et al. [25] define two separate root-mean-squared (RMS) errors (i.e, one on translation and another on rotation components). For both errors, they employ the Euclidean distance between the computed poses and the ground-truth poses, using a rotation vector parametrization for the orientation. Addressing the problem of multiple rotation metrics, Huynh [12] proposes an evaluation of six different types of distance for $SO(3)$ used in the scientific litterature. She concludes that the norm of the difference of Euler Angles is not a distance and that the use of geodesic distance on a unit sphere is preferable. Instead of using continuous metrics, Hugli and Schutz [11] propose to use Successful Initial Configuration map, or SIC-map, to display results on a 2D plot. The authors used fixed thresholds on the error to identify failure, weak success and success of the registration. The SIC-maps help to visualize the convergence region but limit the number of samples that can be tested. This type of result representation also makes comparison between different variants difficult to display.

In this paper, we applied the principles proposed by Amigoni et al. [1] to a subset of the SLAM problem: scan registration. In light of the recent work on registration, we aimed to bring those different evaluation types into the same protocol. This protocol should enhance deeper investigation of registration algorithms by considering (1) a set of external factors and (2) a set of performance metrics.

3 Method

In this section, we highlight the different elements that influence the outcome of ICP variants and that can be controlled in order to evaluate those variants. We also introduce robust metrics that we consider for a quantitative assessment of the algorithm.

3.1 Sensitivity to Input

ICP takes two scans as input with an initial alignment of one with respect to the other. As ICP is an approximate algorithm essentially doing local convergence, its result depends on the initial pose. This initial guess is typically provided by inertial-measurement accumulation, odometry or heuristic motion models, which all have limited precision and increasing uncertainty with time between observations. It is therefore important to assess how well an ICP solution converges close to the correct pose based on various initial hypotheses. To this aim, we propose to sample the space of initial alignment by adding perturbations to a ground-truth value. While the error distribution of odometry models is usually not Gaussian for non-linear kinematic models, the deviation from a Gaussian depends on the actual model and command history, which goes beyond the scope of our data sets. As a reasonable approximation, we sampled the perturbations from zero-mean 6D multivariate Gaussian distribution.

Another factor driving the difficulty of scan matching is the amount of outliers. If there are a lot of points that do not correspond to the same features in both scans, ICP runs the risk of converging to a local optimum driven by false matches. We quantified this phenomenon by assessing the overlap ratio of a scan with respect to another (outlier ratio is the complement of the overlap ratio). More formally, the overlap is defined by the ratio of points of a scan A for which there is a matching point in a second scan B. Points are considered as matching in this case if they lie within a distance limit that decreases with the local density of points.

In robotics, this overlap is primarily governed by the field of view and the motion of the sensor. Indeed, without dynamic elements in the scene, the overlap corresponds mainly to the ratio between the intersection of sensor fields of view on the one hand, and the field of view of the reference point cloud on the other hand. If the motion, especially for rotation, is large when compared to the field of view, then the overlap can be too low for ICP to converge properly. For slow sensors, like 2D laser scanners generating 3D point clouds by rotating around an axis, it is therefore preferable to do scan matching for each consecutive pair of scans. However, on faster

sensors like RGB-D cameras running up to 30 Hz, it is often possible and even desirable to skip several scans, as long as the overlap does not fall too low.

Finally, the content of the scans themselves can have a huge influence on the registration quality. Indoor environments typically exhibit a lot of planar surfaces (e.g. ground, walls, ceiling, tables) that are therefore locally regular. In that case, if the matching step is slightly wrong, a wrongly associated point still has a good chance of behaving like the correct point. On the other hand, natural environments with trees, bushes and herbs will have false matches detrimental to the error minimization. Moreover, environments without a reasonable ratio of horizontal and vertical objects might lack information for proper registration. This typically happens in long and straight hallway or outside on open space where the ground is the major surface present.

3.2 Evaluation Metrics

For each ICP solution, initial alignments (i.e. being the ground truth plus perturbation) is applied to all selected pairs of scans. At the end, the evaluation produces samples from the distribution of resulting alignments for each pair of scans. Then, cumulating error distributions over all pairs of scans eases the analysis of samples from that particular ICP solution for a given environment and a given perturbation level. We can also accumulate over the different environments for the marginal distribution of error of a given ICP solution.

However, this distribution lies in $SE(3)$, the special Euclidean group in dimension 3, whereas we are mainly interested in both the translation and rotation. Therefore, we projected the 6D distribution into the translation and rotation errors. Given the ground-truth transformation expressed by a 4×4 homogeneous matrix T_g and its corresponding transformation found by the registration solution T_r, we can define the remaining error ΔT as follows:

$$\Delta T = \begin{bmatrix} \Delta R & \Delta t \\ \mathbf{0} & 1 \end{bmatrix} = T_r T_g^{-1} \tag{1}$$

with its translation error e_t, defined as the Euclidean norm of translation vector Δt:

$$e_t = \|\Delta t\| = \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2} \tag{2}$$

and its rotation error e_r, defined as the Geodesic distance directly from the rotation matrix ΔR:

$$e_r = \arccos\left(\frac{trace(\Delta R) - 1}{2}\right) \tag{3}$$

In order to compare these distributions, we used robust statistics like the median and the quantiles instead of mean and covariance. Indeed, as the error distributions are far from Gaussians, the empirical mean and covariance are not really indicative values for interpreting precision and accuracy. This choice is similar to May et al. [16] where the authors defined A50, A75, A95 as the respective quantiles for probabilities 0.5 (i.e. the median), 0.75 and 0.95 of the error distributions. Another advantage of these statistics is that they allow interpretation in terms of accuracy and precision. The solution under evaluation is accurate if the values of A50, A75 and A95 are close to zero. The solution is precise if the difference between those quantiles are small.

Throughout this paper, we present the cumulative function of the distribution of outcomes against the distance of the outcome with respect to ground truth. Those graphs thus present the proportion of outcomes that lie beneath a given error. Moreover, it is easy to see the value of this error for each quantile. This type of representation was called *Recall-Accuracy threshold* in a previous work [13]. An alternative presentation of those results is to show the histogram of the number of outcomes for each error bin, which corresponds to the derivative of the cumulative that we propose. However, that presentation renders difficult the comparison of many distributions and the depiction of the A50, A75 and A95 statistics.

Finally, the computing time can be an important factor, especially for online applications with real-time constraints and embedded systems with limited processing power. It is however challenging to get an absolute evaluation of the computing time that is relevant for different hardware and different use cases. The choice of programming language, the technical level of the programmers, the amount of parallelism, etc., are all elements that could affect time performance. In general, time evaluation should be considered as qualitative measurement unless all those elements are controlled and known to be as uniform as possible.

3.3 Protocol

With those metrics, we can now propose a protocol for the evaluation of ICP variants that goes beyond parameter identifications.

First, variants should always be compared to a commonly accepted ICP baseline. This contrasts with papers that compare novel variants between themselves in order to highlight a specific hypothesis. While we recognize the interest of these works, the amount of ICP variants presented in the literature calls for more effort to relate them. In Section 5.2, we analyze two classical variants that we considered reasonable choices for ICP baselines.

Second, ICP variants need to be compared on enough data in order to reduce the risk of overfitting and to ensure statistically signifiant interpretations. Specific fields of application may require specialized data sets, but efforts should be made to also compare on generic data sets. To obtain a comparison as unbiased as possible, the data should cover different kinds of environments at different overlap levels. In this paper, we propose to employ a group of 3D robotics data sets covering a variety of environments. Moreover, algorithms should be compared with different perturbation distributions in order to assess their robustness. We propose three different perturbation levels (easy, medium and hard) according to the characteristics of the data set (mainly the scale of the elements in the environment and the noise of the sensor).

Finally, the actual comparison should be made with respect to the distribution of errors rather than being made just on a single result. We propose to use quantiles as robust statistics to quantitatively describe and compare the different results.

4 Modular ICP

ICP is an iterative algorithm performing several sequential processing steps, both inside and outside its main loop. For each step, there exist several strategies, and each strategy demands specific parameters.

To our knowledge, there is currently no software tool to compare these strategies. The PCL has a partial support for filters in its registration pipeline, but not a completely reconfigurable ICP chain[2]. To enable such a comparison, we have developed a modular ICP chain, as illustrated in Figure 2, and made it available as open source in the form of the `libpointmatcher` library[3]. This library is written in C++11, restricted to the subset supported by GCC 4.4 and more recent versions. In the ICP chain, every module is a class that can describe its own possible parameters, therefore enabling the whole chain to be configured at run time using YAML [5]. This text-based configuration aids to explicit parameters used and eases the sharing of working setups with others, which ultimately allows for reproducibility and reusability of the solutions. Table 1 lists the available modules.

Our ICP chain takes as input two point clouds, in 2D or 3D, and estimates the translation and the rotation parameters that minimize the alignment error. We called the first point cloud the *reference* and the second the *reading*. The ICP algorithm tries to align the reading onto the reference. To do so, it first applies filtering to the point clouds, and then it iterates through a sequence of processing blocks. For each iteration, it associates points in reading to points in reference and finds a transformation of reading that minimizes the alignment error.

4.1 Processing Blocks

More specifically, the ICP chain consists of several steps, implemented by modules. The steps and the corresponding types of modules are:

- *Data filtering*: This step applies to both the reference and the reading point clouds. At this step, zero or more `DataPointsFilter` modules take a point cloud as input, transform it and produce another cloud as output. The transformation might add information, for instance surface normals, or might change the number of points, for instance by randomly removing some of them.
- *Transformation*: The reading point cloud is rotated and translated. Additional data, such as surface normals, are transformed as well.
- *Data association*: A `Matcher` module links points in the reading to points in the reference. Currently, we provide a fast k–nearest-neighbor matcher based on a kd-tree, using `libnabo`.
- *Outlier filtering*: Zero or more `OutlierFilter` modules remove (hard rejection) and/or weight (soft rejection) links between points in the reading and their matched points in the reference. Criteria can be a fixed maximum authorized distance, a factor of the median distance, etc. Points with zero weights are ignored in the subsequent minimization step.
- *Error minimization*: An `ErrorMinimizer` module computes a transformation matrix to minimize the error between the reading and the reference. Different error functions are available, such as point-to-point and point-to-plane.
- *Transformation checking*: Zero or more `TransformationChecker` modules can stop the iteration depending on some conditions. For example, a condition can be the number of times the loop was executed, or it can be related to the matching error. Because the modules can be chained, we defined that the relation between modules must agree through an OR-condition, while all AND-conditions are defined within a single module.

[2] We are in contact with PCL developers to integrate parts of our work into it.

[3] `http://github.com/ethz-asl/libpointmatcher`, version 1.0.0 at time of submission of this paper.

	Current module implementations
Data filtering	`FixStepSampling`, `MaxDensity`, `MaxPointCount`, `MaxQuantileOnAxis`, `MinDist`, `ObservationDirection`, `OrientNormals`, `RandomSampling`, `RemoveNaN`, `SamplingSurfaceNormal`, `Shadow`, `SimpleSensorNoise`, `SurfaceNormal`
Data association	`KDTree`, `KDTreeVarDist`
Outlier filtering	`MaxDist`, `MedianDist`, `MinDist`, `SurfaceNormal`, `TrimmedDist`, `VarTrimmedDist`
Error minimization	`PointToPlane`, `PointToPoint`
Transformation checking	`Bound`, `Counter`, `Differential`
Inspection	`Performance`, `VTKFile`
Log	`File`

Table 1 List of processing blocks available in `libpointmatcher`. This list displays the status of the library as of version 1.0.0 and is intended to evolve over time.

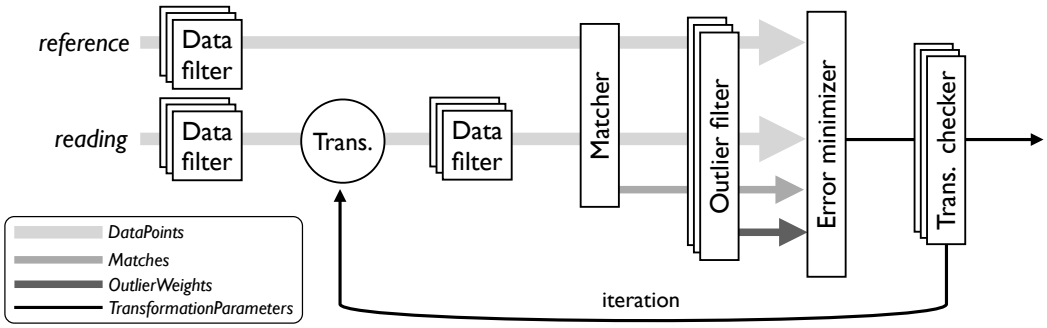

Figure 2 The modular ICP chain as implemented in `libpointmatcher`. Note that some data filters are applied to the reading once and some are applied at each iteration step.

4.2 Data Types

The ICP chain provides standardized interfaces between each step. This allows for the addition of novel algorithms to some steps to evaluate their effect on the global ICP behavior. These interfaces are:

- The `DataPoints` class represents a point cloud. For every point, it has features and, optionally, descriptors. Features are typically the coordinates of the point in the space. Descriptors contain information attached to the point, such as its color, its normal vector, etc. In both features and descriptors, every point can have multiple channels. Every channel has a dimension and a name. For instance, a typical 3D cloud might have the channels "x", "y", "z", "w" of dimension 1 as features (using homogeneous coordinates), and the channel "normal" of size 3 as descriptor. There are no sub-channels, such as "normal.x", for the sake of simplicity. Moreover, the position of the points is in homogeneous coordinates because they need both translation and rotation, while the normals need only rotation. All channels contain scalar values of the scalar type from the template parameter. Although this might be sub-optimal in memory, it eases a lot the interaction between the different modules.
- The `Matches` class is the result of the data-association step, before outlier rejection. It corresponds to a list of associated reference identifiers, along with the corresponding *squared* distance, for all points in the reading. A single point in the reading can have one or multiple matches.
- The `OutlierWeights` class contains the weights of the associations between the points in `Matches` and the points in the reference. A weight of 0 means no association, while a weight of 1 means a complete trust in association.
- The `TransformationParameters` is a transformation in the special Euclidean group of dimension n, $SE(n)$, implemented as a matrix of size $n+1 \times n+1$.

4.3 Implementation

All modules are children of parent classes defined within the `PointMatcher` class. This class is templatized on the scalar type for the point coordinates, typically `float` or `double`. Additionally, the `PointMatcherSupport` namespace hosts classes that do not depend on the template parameter. Every kind of module has its own pair of *.h* and *.cpp* files. Because modules can enumerate their parameters at run time, only the parent classes lie in the

publicly accessible headers. This maintains a lean and easy-to-learn application programming interface (API).

To use `libpointmatcher` from a third-party program, the two classes `ICP` and `ICPSequence` can be instantiated. The first provides a basic registration between a reading and a reference, given an initial transformation. The second provides a tracker-style interface: an instance of this class receives several point clouds in sequence and continuously updates the transformation with respect to a user-provided point cloud. This is useful to limit drift due to noise in the case of high-frequency sensors [19]. A common base class, `ICPChainBase`, holds the instances of the modules and provides the loading mechanism.

When doing research, it is crucial to understand what is going on, in particular in complex processing pipelines like the ICP chain. Therefore, `libpointmatcher` provides two inspection mechanisms: the logger and the inspector. The logger is responsible for writing information during execution to a file or to the console. It will typically display light statistics and warnings. The inspector provides deeper scrutiny than the logger. There are several instances of inspectors in `libpointmatcher`. For instance, one dumps ICP operations as VTK files [23], allowing to visualize the inner loop of the algorithm frame by frame. Another inspector collects statistics for performance evaluation.

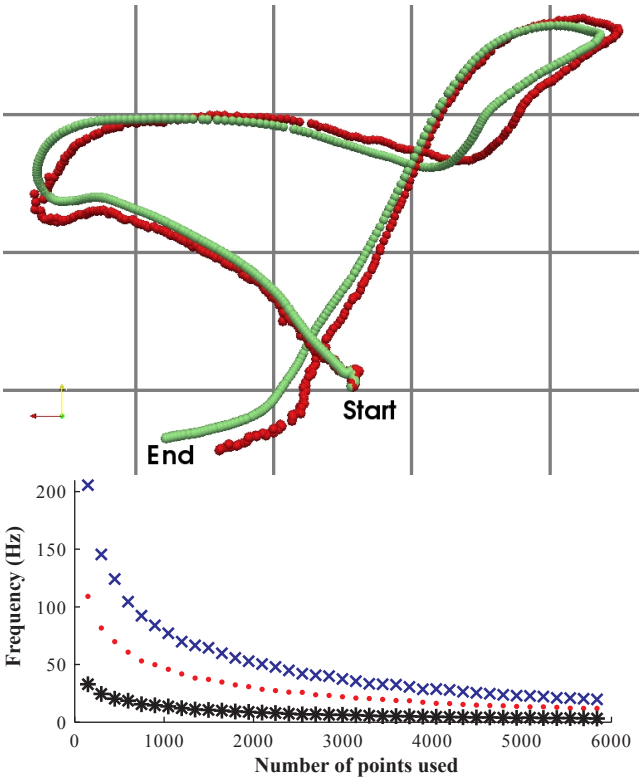

Figure 3 Tracking the pose of a Kinect RGB-D sensor in a home-like environment. *Top*: projection on the xy-plane of a tracked position (dark-red) versus the measured ground truth (light green). Each grid square is half a meter. *Bottom*: performance for different processors: Intel Core i7 Q 820 (blue "×"), Intel Xeon L5335 (red "•"), and Intel Atom Z530 (black "∗")

5 Evaluation

In this section, we show how we applied `libpointmatcher` to two relatively different cases of scan matching: a fast RGB-D camera and a rolling 2D lidar, demonstrating the genericity of our modular ICP chain. In a second part, we give new insights on well-accepted ICP variants using our comparison protocol.

5.1 Applications based on the modular ICP chain

The first application consists of estimating the pose of a Kinect RGB-D sensor in a home-like environment in real-time (30 Hz). Using the `ICPSequence` class of our modular ICP library, this tracker integrates with ROS and publishes the 3D pose as `tf`, the standard way to describe transformations between reference frames in ROS. We explored different parameters related to point-cloud filtering for sensor-noise rejection, the selection of sub-sampling methods and the approximation for the nearest-neighbor search. We first left out points beyond 7 m because these are very noisy with the Kinect. We then sub-sampled the reading randomly, typically keeping 20 % of the 3D points generated from of a 160×120 depth image. For the reference, we used the `SamplingSurfaceNormal` module that efficiently combines sub-sampling and normal generation. This module decomposes the point-cloud space in boxes, by recursively splitting the cloud through axis-aligned hyperplanes in such a way as to maximize the evenness of the aspect ratio of the boxes. When the number of points in a box reaches a threshold value, the filter computes the center of mass of these points and its normal by taking the eigenvector corresponding to the smallest eigenvalue of all points in the box. The reference and the reading points are associated up to a distance of 0.1 m using a kd-tree. As the Kinect works indoors, we performed point-to-plane error minimization. The upper part of Table 2 summarizes the configuration of the ICP chain for this application. The top of Figure 3 shows one of the 27 paths executed while being tracked in parallel with a Vicon system. The Vicon was used to determine the ground truth poses during this evaluation. The bottom of Figure 3 shows the main factor influencing the registration speed: the number of points randomly subsampled for the reading, with real time achieved with

	Step	Module	Description
Kinect tracker	Data filtering of reference	`MaxDist`	keep points closer than 7 m
		`SamplingSurfaceNormal`	random sub-sampling, typically keep 20 %
	Data filtering of reading	`MaxDist`	keep points closer than 7 m
		`RandomSampling`	sub-sampling 17× and normal extraction
	Data association	`KDTree`	kd-tree matching with 0.1 m max. distance
	Outlier filtering	`TrimmedDist`	keep 85 % closest points
	Error minimization	`PointToPlane`	point-to-plane
	Transformation checking	`Differential`	min. error below 1 cm and 0.001 rad
		`Counter`	iteration count reached 30
		`Bound`	transformation beyond bounds
7-floor mapping	Data filtering of reference	`SurfaceNormal`	extraction of surface normal vectors
		`RandomSampling`	random sub-sampling, keep 50 %
	Data filtering of reading	`SurfaceNormal`	extraction of surface normal vectors
		`UniformizeDensity`	keep uniform density
	Data association	`KDTree`	kd-tree matching with 0.5 m max. distance
	Outlier filtering	`TrimmedDist`	keep 95 % closest points
		`SurfaceNormal`	remove when normals are more than 45 degrees off
	Error minimization	`PointToPlane`	point-to-plane
	Transformation checking	`Differential`	min. error below 1 cm and 0.001 rad
		`Counter`	iteration count reached 30
		`Bound`	transformation beyond bounds

Table 2 Configurations of ICP chains for the Kinect tracker and the 7-floor mapping applications.

4,000 points using a single core of a laptop Core i7 Q 820 processor. About 1,700 points are sufficient for high-quality tracking, which is achievable in real time on an old Intel Xeon L5335. An Atom can run at about 10 Hz, with enough points for approximate tracking. The complete results are available in a previous paper [19]. This experiment shows that our library can scale on a large range of computational power and provide high-quality, real-time tracking on current average hardware.

The second application is the mapping of a seven-floor staircase with a search-and-rescue robot (Figure 4). This robot is equipped with tracks and flippers to increase the motion capabilities. However, this implies that the motion estimated from the tracks encoder is highly unreliable, even on flat ground. The robot has a 2D laser scanner mounted on a horizontal axis, allowing it to roll back and forth to acquire 3D scans in front of the robot. In this application, the robot acquires scans with a stop-and-go strategy. The robot maintains an onboard map of the environment (600 k points) that was processed online. When a new scan was available, the robot performed ICP with this map as reference and the scan as reading, like metascan used in [26]. As this is an office environment, we used a point-to-plane variant, which implies that we extracted the normals of the points prior to each registration. The points were associated up to a distance of 0.5 m using a kd-tree. As there was a low expectation of encountering dynamic elements, the 95 % closest points were kept. However, matched points with surface normal vectors differing by more than 45 degrees are discarded. This prevented the points from the ceiling from being matched with the points from the floor above, which would distort the whole map by having floors without thickness. The bottom part of Table 2 summarizes the configuration of the ICP chain. Note that there is no global relaxation or loop closure; the parallel floors visible in Figure 4 are due solely to good registration quality.

Both examples demonstrate the added value of modular ICP chains as they have different requirements that can still be fulfilled with the same open-source ICP library.

5.2 Revisiting well-established ICP variants

In this section, we demonstrate our evaluation protocol on two well-established ICP variants. We have implemented both of them using our library before applying them to different environments. They can provide a fair baseline to which new algorithms can be compared. Furthermore, this shows the relation between environment type, ICP distance metric and convergence performances.

5.2.1 Data sets

We selected six different environments from the "Challenging Laser Registration" data sets [18]. These data sets[4] include ground-truth poses and cover a broad range

[4] `http://projects.asl.ethz.ch/datasets/doku.php?id=laserregistration:laserregistration`

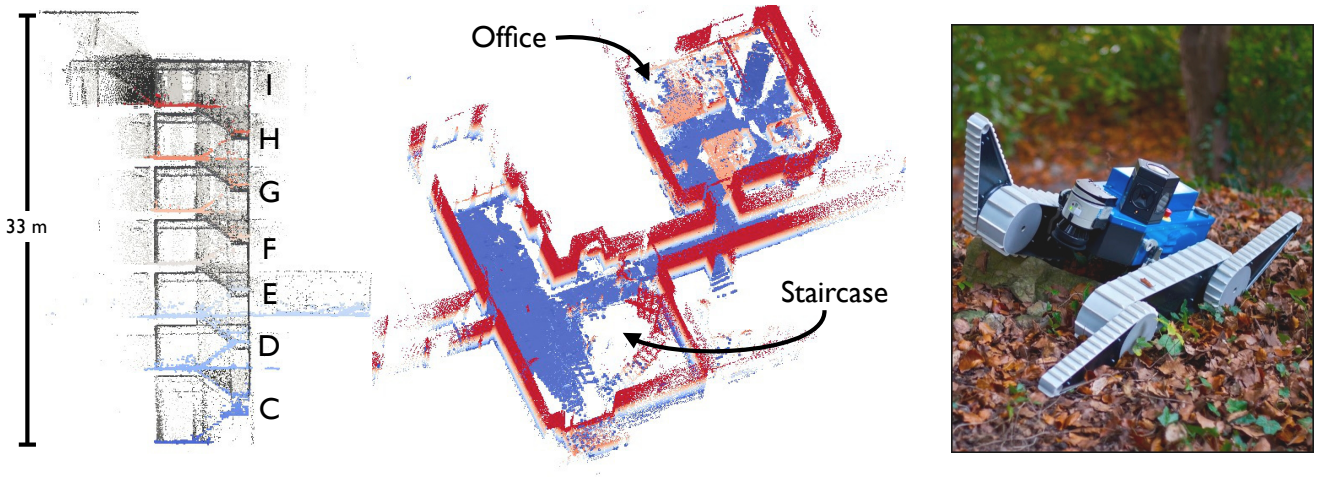

Figure 4 Mapping of a seven-floor staircase using a search-and-rescue robot. *Left*: side view of the resulting map with the floor colored based on elevation. *Middle*: top view of the E floor with the ceiling removed and the points colored based on elevation. *Right*: photograph of the robot with climbing capability.

of applications and conditions, including dynamic outliers such as people walking in the range of the laser while it is scanning. Each data set consists of around 30 full 3D scans. The scans were taken with an Hokuyo UTM-30LX 2D laser range sensor mounted on a tilting platform. The ground-truth poses of the platform were tracked with millimetric precision using a theodolite. Table 3 summarizes the features of the selected data sets: *Apartment* (Figure 5) *ETH*, *Stairs*, *Wood* (in summer), *Gazebo* (in winter, see Figure 6), and Mountain *Plain*. These six data sets cover various types of environments: artificial and natural, cluttered and open, homogeneous and highly variable.

Figure 7 shows the overlap between each pair of scans in all data sets. First, one can see that the overlap is not exactly symmetric. Indeed, if a scan is smaller than the other, all its points will find a match in the second, but not the other way around. Second, *Apartment* and *Stairs* show clusters of scans with high overlap within themselves but low overlap with others. This is due to the segmentation of the volumes in the environment; typically, scans inside a room will all have a relatively high overlap while between rooms the overlap will quickly drop. In comparison, *ETH*, *Wood* and *Plain* share a pattern showing a high overlap that decreases as the index difference grows, as expected. Finally, *Gazebo* shows relatively high values of overlap for each of its scans because the environment is rather open, with few occlusions. We would expect *Plain* to also show high overlap, but it is not the case due to the ground configuration, which is quite uneven, and the lack of points upwards and sideways, which can be confirmed by the number of points per scan as shown in Table 3.

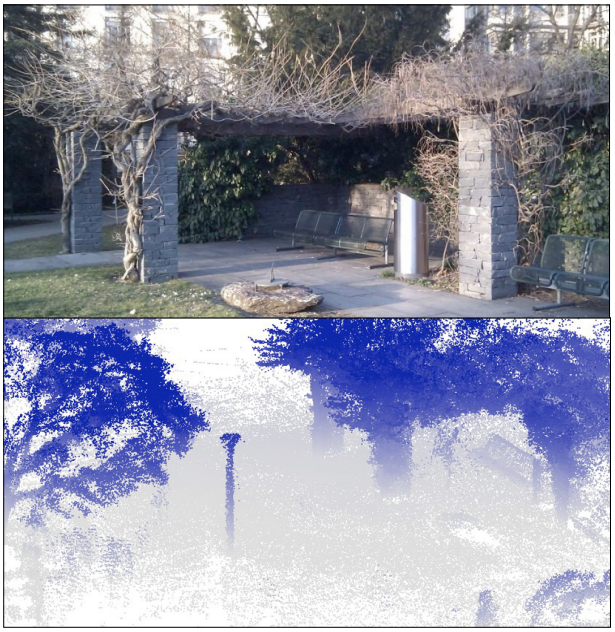

Figure 6 Overview of the *Gazebo* data set. *Top*: photograph of benches under the gazebo covered with wine trees. *Bottom*: aerial view of the gazebo using the acquired scans. The color of the points shows their elevation: high points are in dark blue, low points are in light gray.

The quality of registration is very sensitive to overlap [17]. However, overlap is not homogeneous in a given data set path. For example, Figure 8 shows the evolution of the error in the *Apartment* data set for the point-to-plane distance metric. Scans were registered following the path, which means that every scan was paired with the scan recorded just before. In most cases, the registration is satisfying. However, there are a few

Name	Description	Nbr. scans	Pt. per scan	Poses bounding box	Scene bounding box
Apartment	Single floor with 5 rooms	45	365 k	5 × 5 × 0.06 m	17 × 10 × 3 m
Stairs	Small staircase transitioning from indoor to outdoor	31	191 k	10 × 3 × 2.50 m	21 × 111 × 27 m
ETH	Large hallway with pillars and arches	36	191 k	24 × 2 × 0.50 m	62 × 65 × 18 m
Gazebo (winter)	Wine trees covering a gazebo in a public park	32	153 k	4 × 5 × 0.09 m	72 × 70 × 19 m
Wood (summer)	Dense vegetation around a small paved way	37	182 k	10 × 15 × 0.50 m	30 × 53 × 20 m
Plain	Small concave basin with alpine vegetations	31	102 k	18 × 6 × 2.70 m	36 × 40 × 8 m

Table 3 Characteristics of the six data sets used to revisit well-established ICP variants.

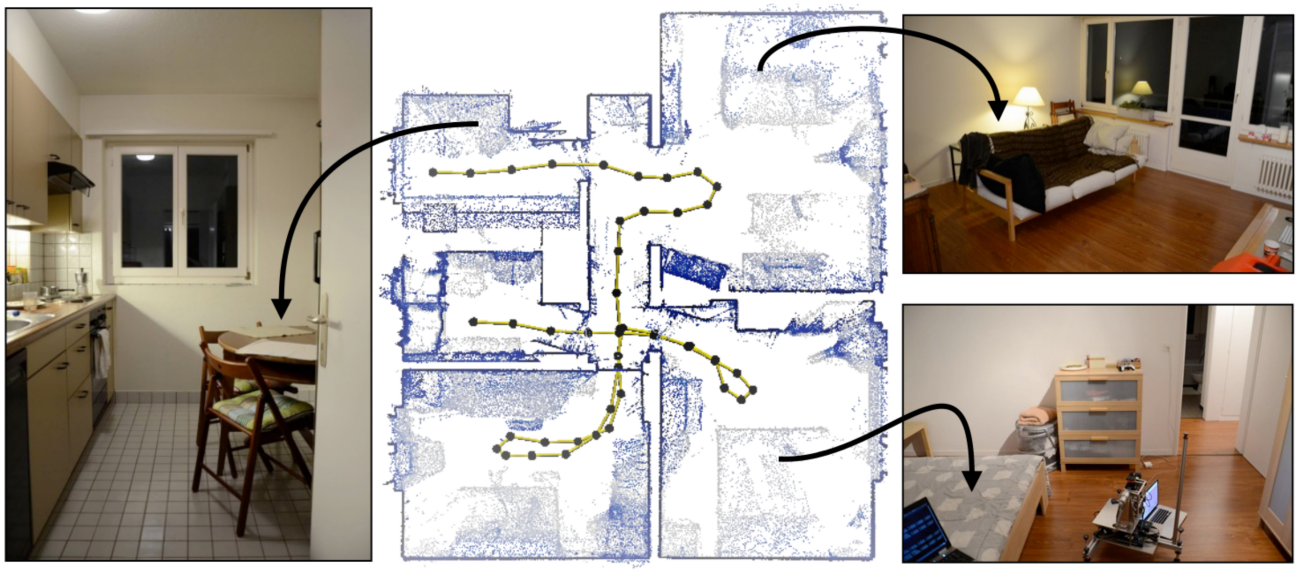

Figure 5 Overview of the *Apartment* data set. *Left*: photograph of the kitchen. *Middle*: top view of the point clouds with the ceiling removed. The color of the points shows their elevation: high points are in dark blue, low points are in light gray. The yellow lines with black dots represent the path of the scanner through the apartment. *Top right*: photograph of the living room. *Bottom right*: photograph of the bedroom.

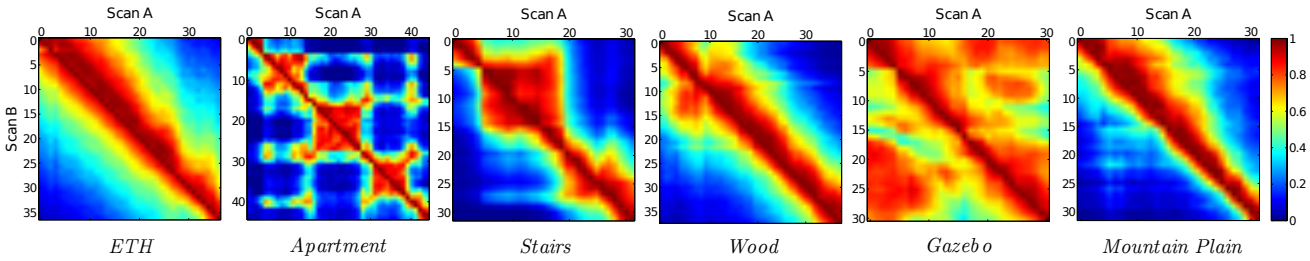

Figure 7 Estimated overlap for all data sets. Tables can be read as the percentage of points in Scan A that are also in Scan B. Dark red is high overlap and dark blue is low overlap. Diagonal elements have a ratio of 1.

places, around openings, where the performance degrades. Those places correspond to opening of the field of view which corresponds to a sudden decrease in the overlap. Change in overlap doesn't appear uniformly in all paths executed while recording data sets. Thus, it is possible that the difference in overlap between two paths shade the impact on the type of environment. To overcome this limitation, we randomly selected 35 pairs of scans, ensuring a uniform coverage of the overlap between 0.30 and 0.99 for all data sets. Those pairs were selected using the values of Figure 7 with the lower bound of 0.30 forced by the lowest overlap value in *Gazebo*.

5.2.2 Perturbations

For the sampling of the initial poses, we designed three different sets of initial perturbations sampled from Gaussian distributions with three different variance magnitudes (see Table 4). Figure 9 shows the cumulative

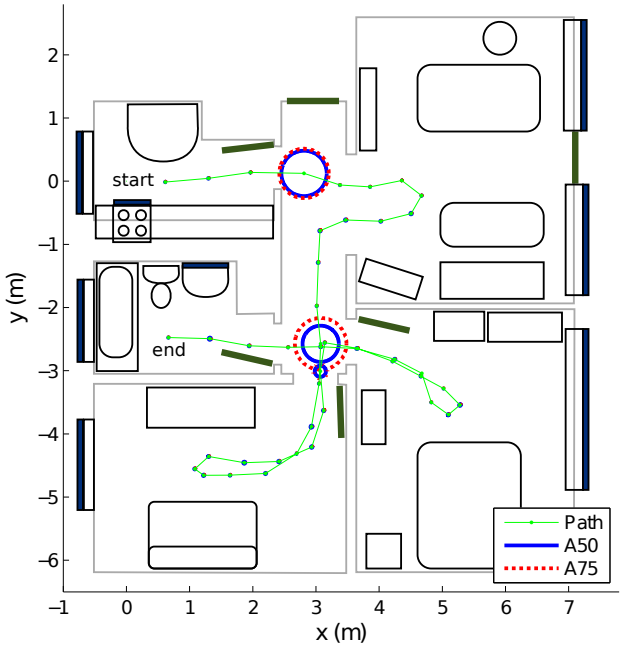

Figure 8 Point-to-plane solution in the *Apartment* data set: separate statistics for every pose. The path of the scanner (green) with the A50 and A75 statistics overlaid on a sketch of the environment.

	Translation	Rotation	Nb. Samples
Easy	0.1 m	10°	64
Medium	0.5 m	20°	64
Hard	1.0 m	45°	64

Table 4 Standard deviations on each component and number of samples for each perturbation level.

probability as a function of translation error for the three perturbation sets: easy, medium and hard. The filled backgrounds show the respective theoretical distributions. It is worth noting that the norm of multivariate-Gaussian–distributed variables is an χ-distribution. The difference and the jaggedness of the sampled distribution compared to the theoretical distribution is due to the relatively low number of samples, 64, compared to the six dimensions of the sampling space. As we aim at proposing those perturbation samples to the community to allow everyone to compare their solution in the same conditions as ours, we felt that significantly increasing the number of perturbations would deter people from trying due to the computation time it would take. The sub-sampling we used required 2,240 tests per perturbation type per environment, which we consider to be a reasonable compromise between the number of samples and the evaluation time.

A list of the selection of scans combined with the precomputed perturbation for all data sets is available by direct communication with the authors and will be

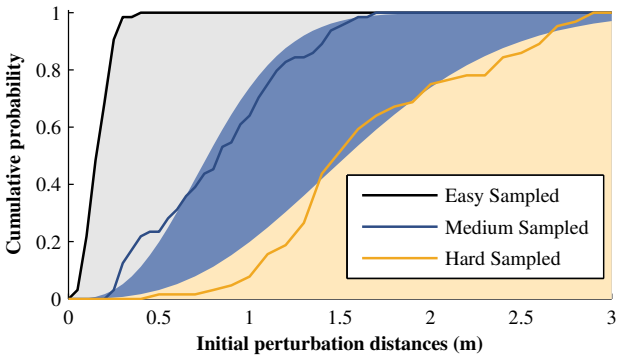

Figure 9 Cumulative probability as function of translation error for each of the perturbation sets. The lines are based on the actual 64 samples; the filled backgrounds correspond to the theoretical curves. The easy sampled and theoretical curves overlay due to scaling.

accessible on a web site for convenience in the near future.

5.2.3 Selection and Optimization of ICP *Parameters*

We wish to revisit two of the textbook ICP variants, using point-to-point [6] and point-to-plane [7] distance metrics, both combined with the trimmed-ICP outlier rejection [8]. We have chosen these because they are the most compared and researchers need to re-implement them every time. We hope to accelerate the comparison process for more modern solutions by providing those two baseline solutions in an open-source library.

Albeit simple, they depend on a certain number of parameters. We have fixed some and optimized others to allow for an efficient convergence of the algorithm. Table 5 shows the final values after optimization. We aimed at both minimizing the error and maximizing the performance, following the method described in a previous work [19].

Our ICP chain starts by sub-sampling both the reference and the reading point clouds. In the case of point-to-point, both point clouds are sub-sampled with uniform probability using the `RandomSampling` module. We explored the space of sub-sampling ratios using probabilities of keeping points in the range of {0.001, 0.01, 0.05, 0.1, 0.5, 1.0} for the reading and {0.001, 0.01, 0.05, 0.1, 1.0} for the reference. In the case of point-to-plane, because we wanted to extract the normals, we used the `SamplingSurfaceNormal` module. We explore thresholds of sizes {5, 7, 10, 20, 100, 200}. For the reading, we used the same sub-sampling method as for point-to-point, looking for ratios of {0.001, 0.01, 0.05, 0.1, 0.5, 1}. After an exhaustive search, this optimization returns ratios of 0.05 for both the reference and the reading for

	Step	Module	Description
Point-to-point	Data filtering of reference	`MinDist`	keep points beyond 1 m
		`RandomSampling`	random sub-sampling, keep 5 %
	Data filtering of reading	`MinDist`	keep points beyond 1 m
		`RandomSampling`	random sub-sampling, keep 5 %
	Data association	`KDTree`	kd-tree matching with approx. constant ϵ of 3.16
	Outlier filtering	`TrimmedDist`	keep 75 % closest points
	Error minimization	`PointToPoint`	point-to-point
	Transformation checking	`Counter`	iteration count reached 150
		`Differential`	min. error below 1 cm and 0.001 rad
	Step	Module	Description
Point-to-plane	Data filtering of reference	`MinDist`	keep points beyond 1 m
		`SamplingSurfaceNormal`	sub-sampling 7× and normal extraction
	Data filtering of reading	`MinDist`	keep points beyond 1 m
		`RandomSampling`	random sub-sampling, keep 5 %
	Data association	`KDTree`	kd-tree matching with approx. constant ϵ of 3.16
	Outlier filtering	`TrimmedDist`	keep 70 % closest points
	Error minimization	`PointToPlane`	point-to-plane
	Transformation checking	`Counter`	iteration count reached 150
		`Differential`	min. error below 1 cm and 0.001 rad

Table 5 Configurations of ICP chains for revisiting well-established ICP variants. *Top*: point-to-point. *Bottom*: point-to-plane.

point-to-point, and a ratio of 0.05 for the reading and a threshold of 7 points for the reference for point-to-plane.

The matching step looks for the nearest neighbors of every point using a kd-tree. We use the `KDTree` module, which has three parameters: the number of nearest neighbors in the reference to associate to each point in the reading, an approximation factor ϵ allowing a maximum error of $1 + \epsilon$ between the returned nearest neighbor and the true nearest neighbor [3] and a maximal distance beyond which neighbors are not considered any more. We use only one neighbor for the sake of simplicity. We choose a value of 3.16 for ϵ because as shown in a previous work [19], this value leads to the fastest registration.[5] Indeed, with a smaller ϵ, nearest-neighbor queries take longer, and with a larger ϵ, more iterations are required until convergence because of the matching errors.

Following the original implementation, we do not set any distance limit to the association. Our nearest-neighbor library, `libnabo`, has been shown to be one of the fastest kd-tree for ICP [9].

We then rejected outliers whose distance is larger than a certain quantile. Using the `TrimmedDist` module, we explored keeping a ratio of {0.2 0.5 0.7 0.75 0.8 0.85 0.90 0.95 0.9999}. Based on this search, we decided to keep the 75 % closest points for point-to-point and 70 % for point-to-plane. For further details on parameter behaviors, we refer to a previous work [19].

[5] The semantics of ϵ has been changed since [19] to be compatible with other open-source implementations.

5.2.4 Results

We executed our protocol for both solutions leading to a total of 80,640 registrations (i.e. 2 solutions × 6 data sets × 35 paired scans × 3 types of perturbation × 64 perturbations). The overall translation results propose that point-to-plane (A50 = 0.76 m) is more accurate by 20 % than point-to-point (A50 = 0.97 m) solution. The advantage is reversed when looking at the difference between A95 and A50, which shows that point-to-point is more precise by 30 %. The same trend is observed for the rotation with the accuracy gain cranking to 40 % for point-to-plane while the precision advantage stays at 30 % for point-to-plane. For a deeper investigation, all results in Table 6 are subdivided into three categories: (1) data sets, (2) perturbation levels and (3) distance metrics. We can observe once more that most of the times the results of point-to-plane are better than point-to-point. Point-to-point error can however out-perform point-to-plane error for hard perturbations.

To explore the influence of the environment, Figure 10 compares the translation error combining all perturbations for each solution. Note that the A95 values for *ETH* exceed the graph, being 12.16 m for point-to-point and 16.87 m for point-to-plane. Focusing on A50 and A75, we see that the gain of point-to-plane over point-to-point is overcome in the data sets *Wood* and *Plain*. This observation proposes that the accuracy of each solution follows the level of structure found in each data set. When looking at the A95 statistics, point-to-plane is in all cases higher than point-to-point, meaning that point-to-plane does not guarantee better worst-

			Apartment			*Stairs*			*ETH*			*Gazebo*			*Wood*			*Plain*		
			A50	A75	A95	A50	A75	A95	A50	A75	A95	A50	A75	A95	A50	A75	A95	A50	A75	A95
Translation	EP	Plane	0.06	0.47	2.11	0.09	1.17	3.49	0.10	0.44	6.06	0.11	0.38	2.08	0.25	1.55	4.75	0.42	1.54	4.15
		Point	0.13	0.54	1.54	0.35	1.29	2.57	0.47	2.23	6.86	0.28	0.60	1.71	0.39	1.48	4.21	0.51	1.46	3.09
	MP	Plane	0.20	1.04	2.98	0.61	2.08	4.64	0.60	4.06	16.3	0.28	0.96	3.51	1.25	2.92	6.62	1.30	2.58	5.58
		Point	0.46	1.03	2.32	0.94	1.86	3.38	1.92	4.29	11.2	0.49	1.13	3.18	1.19	2.52	5.15	1.21	2.17	3.76
	HP	Plane	1.35	2.18	3.66	2.05	3.28	5.50	4.18	8.55	19.6	1.87	3.33	6.95	2.79	4.52	7.86	2.35	4.13	8.85
		Point	1.29	1.99	3.24	1.81	2.78	4.75	3.84	7.06	14.8	1.58	2.79	4.57	2.32	3.73	6.82	2.02	3.14	6.33
Rotation	EP	Plane	0.02	0.20	1.14	0.02	0.31	1.58	0.01	0.02	0.61	0.02	0.08	0.48	0.05	0.34	0.95	0.07	0.20	0.60
		Point	0.07	0.25	0.97	0.12	0.39	1.22	0.05	0.22	0.83	0.04	0.17	0.41	0.09	0.29	0.77	0.09	0.20	0.44
	MP	Plane	0.08	0.47	1.80	0.16	1.08	2.09	0.01	0.25	2.91	0.04	0.35	0.97	0.31	0.78	1.53	0.19	0.38	0.99
		Point	0.20	0.61	1.49	0.33	0.78	1.63	0.14	0.59	1.82	0.15	0.35	0.80	0.32	0.69	1.22	0.20	0.37	0.77
	HP	Plane	1.01	1.72	2.95	1.48	1.91	2.94	1.31	2.09	3.11	0.58	1.31	2.88	1.05	1.56	2.53	0.50	1.09	3.05
		Point	1.04	1.60	2.53	1.10	1.64	2.53	0.97	1.73	3.05	0.58	1.20	2.59	0.97	1.44	2.35	0.46	0.99	2.09

Table 6 Overall view of the precision obtained with our two proposed baselines for different perturbations (easy (EP), medium (MP), hard (HP)). *Top*: Translation error [m]. *Bottom*: rotation error [rad]. Darker tones correspond to high error.

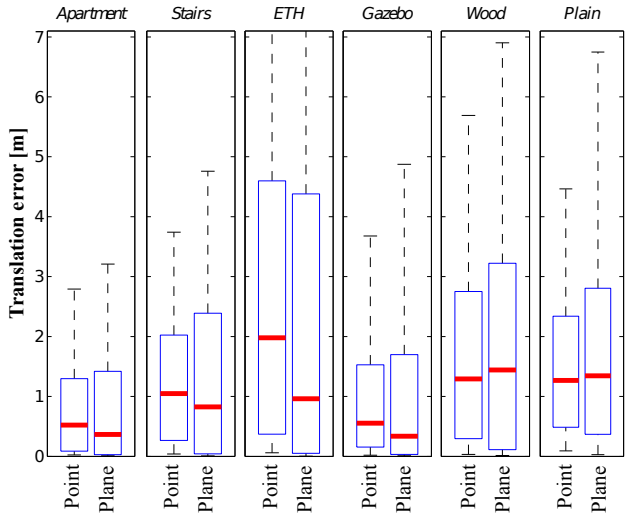

Figure 10 Comparison of point-to-plane and point-to-point performances for all perturbations and clustered environments. Thick red bars correspond to A50 (i.e. the median); the higher end of blue rectangles are A75 and the top end of dashed lines are A95.

case errors than point-to-point. It is worth noting that *ETH* consists of a long hallway with repetitive elements, which seems to drag down the A95 performance in translation while keeping reasonably low rotation errors (see Table 6). The data set *Plain* has an even higher deficiency in term of constraints than *ETH*, with only one major plane representing the ground. Even with this level of constraint, the registrations applied in *Plain* seem to diverge less than in *ETH* for hard conditions represented by A95 statistics.

Given that point-to-plane has a better overall performance, Figure 11 focuses exclusively on that solution and shows the cumulative probabilities of its translation error. Those curves are similar to precision-recall graphs in that the more top-left the curve the better the algorithm performs. The top plot emphasizes the influence of the environments given easy perturbations. This type of situation would happen for a mobile robot able to maintain low uncertainty on its localization between registrations. All of the environments keep their median error under 10 cm except *Wood* and *Plain*. Although considered a semi-structured environment, *Gazebo* keeps lower error, with *Apartment*, than the other environments. The bottom plot goes a bit deeper in the analysis by expending the results for *Apartment* to assess the influence of the perturbation levels. Each curve is associated with its initial perturbation level represented as a filled area. Ideally, all pairs of scans would have fewer residual errors after the registration leading to curves closer to zero than their associate perturbation level. One can observe that, for all perturbation types, roughly 25% of the registrations still present worse translation than their initial perturbations. We believe the cause to be mainly the weak robustness of the solution against a range of different overlap ratios.

To demonstrate this low performance, Figure 12 shows the relation between the pre-computed overlap between scans and the translation errors for both solutions over all environments and all perturbation types. The statistics A50, A75 and A95 were extracted for each bin of paired scan sharing the same overlap, with the bin size being 0.08. Both solutions share the same *Outlier Filtering Module* tuned to handle 70 % and 75 % of outliers. This results in both solutions following the same trend leading to poor performance at low overlap values. The error reaches a median error larger than 2 m for a range of overlap from 0.30 to 0.38.

Finally, Figure 13 shows the cumulative probabilities of the time needed to converge for point-to-plane. The figure opposes structured environments (solid lines) to

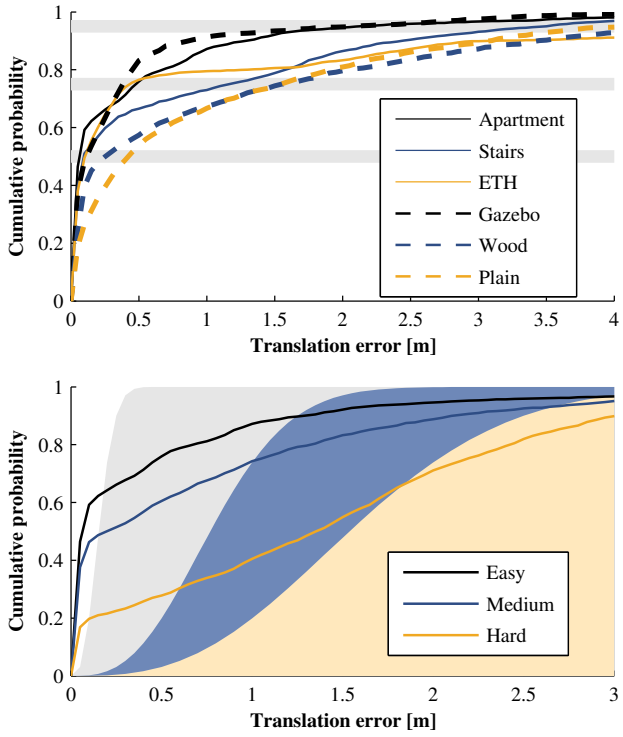

Figure 11 Cumulative probabilities of errors for point-to-plane ICP variant. *Top:* influence of environments given an easy perturbation level. The gray stripes correspond to the quantiles of interest, namely A50, A75 and A95. *Bottom:* influence of the three perturbation levels on the *Apartment* data set with the filled backgrounds correspond to the theoretical curves of initial perturbations.

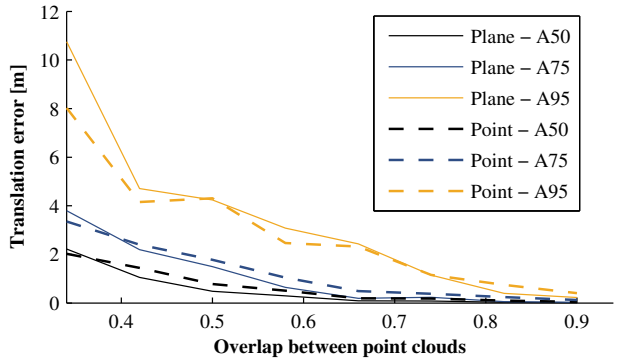

Figure 12 Correlation between the overlap of two scans and the translation error for point-to-plane over all environments and all perturbation types.

unstructured and semi-structured environments (dashed lines). It is interesting to note that in *Plain* the solutions converge rapidly but, based on Table 6, to a large translation error. This means that the observed errors were estimated to be below 1 cm and 0.001 rad (see the line Transformation checking in Table 5) leading to an early exit out of the iteration loop. For the overall performance between the two solutions, point-to-point is

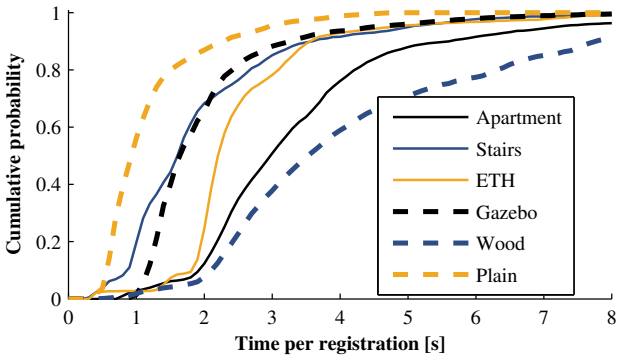

Figure 13 Cumulative probabilities of the time needed to converge for point-to-plane with easy perturbations. The solid lines represent structured environments while dashed lines represent unstructured and semi-structured environments.

80 % faster than point-to-plane with a median time of 1.45 s compared to 2.58 s respectively. This suggests that for point-to-plane, the extra time required to extract surface normal vectors is not compensated for by the saving on the number of iterations required to converge. All the results were obtained on a 2.2 GHz Intel Core i7, using `libpointmatcher` (C++) with separate registrations running on a single core without GPU acceleration. The solutions are not multi-threaded but we executed four tests in parallel on a single machine to reduce the total testing time.

6 Discussion

We have sub-sampled the point clouds using a fixed reduction percentage leading to the use of approximatly 10,000 points per scan. However, the different data sets have a different number of points per scan in average, for instance *Apartment* has twice as much as *Stairs*. It would be better to reduce the point clouds to a fixed number of points instead of a ratio to ensure more constant processing time given that the precision gain is very low for a larger number of points [18]. As demonstrated in Figure 8, overlap between scans can largely vary depending on the motion of the robot and the environment configuration. One of the limitation of trimming outliers based on quartile is that this assumes a constant overlap of scans, which is hard to control with a mobile platform. In order to work around this limitation, it would be important to detect those places and react appropriately. For example, the robot could acquire scans more frequently or reduce its velocity at those places. Also, more flexible outlier-rejection algorithms need to be investigated to cope with the variability of the overlap.

The use of the A95 statistic might seem excessive, but it is important to note that it implies that one registration over 20 is beyond this value. In the robotics context, this is very significant and can be the difference between a stable system and a system that breaks its map every so often.

The point-to-plane solution can be stable for applications where: first, the environment type can be controlled to be highly structured; second, the overlap is kept high while the robot is moving and third, the state estimation used as initial pose for the registration remains within 10 cm and 10 °. These types of conditions are usual for laboratory experiments but are unlikely to happen in real applications.

The procedure we propose relies on some specific data sets in order to have a common ground of comparison in the scientific community. However, as the sensor is the same across all data sets, we cannot measure its effect on the ICP performances. The sensor has nevertheless two important features, noise and field of view, that can have an influence on ICP. Indeed, sensors may have different noise levels and even noise profiles, and different ICP variants might cope better with some than others. Furthermore, the field of view and the point-density profile of the sensor inside its field of view can have a huge influence on the ICP performance as those characteristics govern the overlap and the possibility of multiple pairings between scans.

Finally, as explained previously, some applications require online matching of sensor data. In these cases, the time spent in ICP is a relevant criterion to compare variants. However, processing time is difficult to measure given that internal memory management, processor load and processor types are all relevant factors that cannot easily be compensated for and that can drastically change time measurements. On the other hand, theoretical complexity is not sufficient as different ICP variants will mostly have a comparable complexity but different constant factors. Having a single computer dedicated to running all the different ICP variants in the same condition would yield a general idea of the relative efficiency. However, different ICP variants would scale differently for different practical cases. A comparison of the variants in the specific case of application is thus always pertinent. Our library can facilitate this comparison by highlighting only the relevant changes. Indeed, the efficiency of an implementation is an important factor of time performance that can bias the comparison of algorithms. Having a library in which only the modules to be compared change already significantly reduces this effect by maintaining a homogeneous environment for most data processing.

In a nutshell, researchers using our protocol should maintain a certain uniformity by:

1. Characterizing the main parameters of their novel solution.
2. Evaluating their solutions using the predefined data sets and pairs of scans and perturbations.
3. Recording translation and rotation errors following Equations 2 and 3.
4. Recording computational time excluding data acquisition but including preprocessing steps.
5. Reporting strength and weakness against environment type, perturbation level and overlap ratio.
6. Comparing their results with formal solution in terms of precision and accuracy using A50, A75 and A95 statistics.
7. Making their results publicly available, when possible, so that other researchers can accelerate the comparison process.

7 Conclusion

In this paper, we proposed a protocol to compare ICP variants. We lay the emphasis on the repeatability of the results by selecting publicly available data sets. We also presented an open-source modular ICP library that can further improve on the repeatability by allowing easy tests and comparisons with baseline variants. Thus, this modular library is the companion of choice of our protocol. Finally, we demonstrated our evaluation framework by comparing well-established ICP variants in a rich variety of environments. This refreshes the observations from Rusinkiewicz and Levoy [20] by using data sets closer to robotic applications. The performances of these baseline variants show a high variability and strongly display the need for improved ICP methods for natural, unstructured and information-deprived environments. This need opens the door for other researchers to challenge their novel solutions against our baselines.

We would welcome additional data sets with different sensors and other ICP implementations, but our comparison is already a stepping stone in ICP comparison that can be built upon. We believe that this combination of protocol, software and baseline results shows nicely how open-source software can drive research forward.

References

1. Amigoni, F., Reggiani, M., Schiaffonati, V.: An insightful comparison between experiments in mobile robotics and in science. Autonomous Robots **27**(4), 313–325 (2009)

2. Armesto, L., Minguez, J., Montesano, L.: A generalization of the metric-based Iterative Closest Point technique for 3D scan matching. In: Robotics and Automation, 2010. Proceedings of the IEEE International Conference on, pp. 1367–1372 (2010)
3. Arya, S., Mount, D.: Approximate nearest neighbor queries in fixed dimensions. In: Discrete Algorithms, 1993. Proceedings of the 4th Annual ACM-SIAM Symposium on, pp. 271–280 (1993)
4. Baker, S., Scharstein, D., Lewis, J., Roth, S., Black, M., Szeliski, R.: A Database and Evaluation Methodology for Optical Flow. International Journal of Computer Vision pp. 1–8 (2007)
5. Ben-Kiki, O., Evans, C., döt Net, I.: YAML Ain't Markup Language (YAML^{TM}) version 1.2. `http://www.yaml.org/spec/1.2/spec.html` (2009)
6. Besl, P., McKay, H.: A method for registration of 3-D shapes. Pattern Analysis and Machine Intelligence, IEEE Transactions on **14**(2), 239–256 (1992)
7. Chen, Y., Medioni, G.: Object modeling by registration of multiple range images. In: Robotics and Automation, 1991. Proceedings of the IEEE International Conference on, pp. 2724–2729. IEEE Comput. Soc. Press (1991)
8. Chetverikov, D., Svirko, D., Stepanov, D., Krsek, P.: The Trimmed Iterative Closest Point algorithm. In: Pattern Recognition, 2002. Proceedings of the 16th International Conference on, pp. 545–548 (2002)
9. Elseberg, J., Magnenat, S., Siegwart, R., Nüchter, A.: Comparison of nearest-neighbor-search strategies and implementations for efficient shape registration. Journal of Software Engineering for Robotics **3**(1), 2–12 (2012)
10. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In: Computer Vision and Pattern Recognition (CVPR) (2012)
11. Hugli, H., Schutz, C.: Geometric matching of 3D objects: assessing the range of successful initial configurations. In: 3-D Digital Imaging and Modeling, 1997. Proceedings of the International Conference on Recent Advances in, pp. 101–106 (1997)
12. Huynh, D.Q.: Metrics for 3D Rotations: Comparison and Analysis. Journal of Mathematical Imaging and Vision **35**(2), 155–164 (2009)
13. Jian, B., Vemuri, B.C.: Robust Point Set Registration Using Gaussian Mixture Models. Pattern Analysis and Machine Intelligence, IEEE Transactions on **33**(8), 1633–1645 (2011)
14. Magnusson, M., Lilienthal, A., Duckett, T.: Scan registration for autonomous mining vehicles using 3D-NDT. Journal of Field Robotics **24**(10), 803–827 (2007)
15. Magnusson, M., Nüchter, A., Lorken, C., Lilienthal, A., Hertzberg, J.: Evaluation of 3D registration reliability and speed - A comparison of ICP and NDT. In: Robotics and Automation, 2009. Proceedings of the IEEE International Conference on, pp. 3907–3912 (2009)
16. May, S., Droeschel, D., Holz, D., Fuchs, S., Malis, E., Nüchter, A., Hertzberg, J.: Three-dimensional mapping with time-of-flight cameras. Journal of Field Robotics **26**(11-12), 934–965 (2009)
17. Pathak, K., Borrmann, D., Elseberg, J., Vaskevicius, N., Birk, A., Nüchter, A.: Evaluation of the robustness of planar-patches based 3D-registration using marker-based ground-truth in an outdoor urban scenario. In: Intelligent Robots and Systems, 2010. Proceedings of the IEEE/RSJ International Conference on, pp. 5725–5730 (2010)
18. Pomerleau, F., Liu, M., Colas, F., Siegwart, R.: Challenging Data Sets for Point Cloud Registration Algorithms. The International Journal of Robotics Research (2012)
19. Pomerleau, F., Magnenat, S., Colas, F., Liu, M., Siegwart, R.: Tracking a depth camera: Parameter exploration for fast ICP. In: Intelligent Robots and Systems, 2011. Proceedings of the IEEE/RSJ International Conference on, pp. 3824–3829 (2011)
20. Rusinkiewicz, S., Levoy, M.: Efficient variants of the ICP algorithm. In: 3-D Digital Imaging and Modeling, 2001. Proceedings of the Third International Conference on, pp. 145–152 (2001)
21. Rusu, R., Cousins, S.: 3D is here: Point Cloud Library (PCL). In: Robotics and Automation, 2011. Proceedings of the IEEE International Conference on, pp. 1–4 (2011)
22. Scharstein, D., Szeliski, R.: A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. International Journal of Computer Vision **47**(1), 7–42 (2002)
23. Schroeder, W., Martin, K., Lorensen, B.: Visualization Toolkit: An Object-Oriented Approach to 3D Graphics. 4th edn. Kitware (2006)
24. Seitz, S.M., Curless, B., Diebel, J., Scharstein, D., Szeliski, R.: A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms. In: Computer Vision and Pattern Recognition, 2006. Proceedings of the IEEE Computer Society Conference on, pp. 519–528 (2006)
25. Tong, C.H., Barfoot, T.D., Dupuis, É.: Three-dimensional SLAM for mapping planetary work site environments. Journal of Field Robotics pp. 381–412 (2012)
26. Wulf, O., Nüchter, A., Hertzberg, J., Wagner, B.: Benchmarking urban six-degree-of-freedom simulta-

neous localization and mapping. Journal of Field Robotics **25**(3), 148–163 (2008)