NICP: Dense Normal Based Point Cloud Registration

Jacopo Serafin[1] and Giorgio Grisetti[1]

Abstract— **In this paper we present a novel on-line method to recursively align point clouds. By considering each point together with the local features of the surface (normal and curvature), our method takes advantage of the 3D structure around the points for the determination of the data association between two clouds. The algorithm relies on a least squares formulation of the alignment problem, that minimizes an error metric depending on these surface characteristics. We named the approach Normal Iterative Closest Point (NICP in short). Extensive experiments on publicly available benchmark data show that NICP outperforms other state-of-the-art approaches.**

I. INTRODUCTION

Registering two point clouds consists in finding the rotation and the translation that maximize the overlap between the two clouds. This problem is a crucial building block in several applications that solve more complex tasks. These include simultaneous localization and mapping (SLAM), object detection and recognition, augmented reality and many others. Point cloud registration has been addressed by many authors and an excellent overview is given by Pomerleau *et al.* in [1]. Usually, this problem is solved by using some variant of the Iterative Closest Point (ICP) algorithm, originally proposed by Besl and McKey [2]. Among all these variants, Generalized ICP (GICP) [3] showed to be one of the most effective and robust.

The availability of affordable high speed RGB-D sensors, like the Microsoft Kinect or the Asus Xtion, enlarged the basin of potential users and resulted in an increased interest from researchers. Registration techniques that were previously designed for 3D laser scanners, when used in conjunction with these sensors, suffer from the different noise characteristics of the depth measurements. Furthermore, an RGB-D camera can deliver data up to 60Hz and such frame rate is not achievable by common 3D laser scanners, that typically require several seconds for a dense scan. RGB-D cameras led to the development of approaches that benefit from this high data rate like Kinect Fusion (KINFU) [4] or Dense Visual Odometry (DVO) [5]. These methods, however, are not well suited to deal with 3D laser scanners data since they require a good guess of the current transformation that is available only at high frame rates.

In this paper, we present a complete system to recursively align point clouds that combines and extends state-of-the-art approaches in both domains to achieve enhanced speed and robustness. More in particular, our system combines and extends the point-to-plane error metric proposed in GICP, while using a scene representation inspired by the Normal Distribution Transform (NDT) [6]. Similarly to KINFU and DVO, we use an image projection or line of sight criterion to determine the data association.

Fig. 1. Side views of the reconstruction of the ETH Hauptgebaude dataset. From top to bottom: ground-truth, NICP, GICP and NDT reconstruction. NICP was able to reconstruct consistently the scene while both NDT and GICP generated a shortened corridor (*count the arcs*).

In contrast to the above mentioned approaches, our method exploits the structure of the scene not only when searching for correspondences, but also in the solution of the alignment problem. This is done, as described in our previous work [7], by using an extended representation for the measurements, in which the Euclidean coordinates of each point are augmented with the surface normal. In the solution of the alignment, we minimize the Mahalanobis distance of each corresponding point pair *and* their normals. This results in minimizing a distance among 6D vectors instead of 3D points. For this reason, we named the approach Normal Iterative Closest Point (NICP in short). In our previous work [7], where we focused on a synthetic analysis of the error function, we did not provide details on our complete registration pipeline. In this paper we describe a full approach that can directly operate on raw 3D sensor data, either depth images or 3D laser scans. More in detail, we discuss how to extract the statistics from the raw data, and how to represent the point clouds through data structures that support efficient correspondence searches and the least squares optimization described in [7]. In this work we report the results of comparative experiments with other state-of-the-art methods on 29 standard benchmarking datasets developed by Pomerleau *et al.* in [8] and [9]. Our

*This work has partly been supported by the European Commission under FP7-600890-ROVINA.

[1]Both authors are from the Department of Computer, Control, and Management Engineering "Antonio Ruberti" of Sapienza University of Rome. Via Ariosto 25, I00185 Rome, Italy. Email: {serafin,grisetti}@dis.uniroma1.it.

current implementation can be run on-line on a standard laptop, and, in our experimental evaluation, NICP showed to outperform all the other approaches. Figure 1 shows a comparison of our approach with GICP and NDT in the sequential alignment of 3D laser data. An open source implementation of our algorithm as standalone C++ library is available at `http://goo.gl/W3qXbE`.

Additionally, on top of our registration pipeline we implemented a naive mapping algorithm which renders our system a complete depth-tracker. In our extended experiments, this algorithm is reported to outperform the well known KINFU approach, at a fraction of the computation and without requiring a GPU implementation. Yet, our method is very suitable for being ported to such platforms, but it is out of the scope of this paper.

The remainder of this paper is organized as follows: in Sec. II we give an overview of the state-of-the-art methods for point cloud registration; in Sec. III we shortly introduce the ICP algorithm; subsequently in Sec. IV we present our system; Sec. V shows comparative experiments and Sec. VI conclude the paper.

II. RELATED WORK

The problem of registering two point clouds has been addressed since more than two decades and the available methods can be categorized in two main groups: sparse approaches that rely on few meaningful points in the scene, and dense approaches that directly operate on the entire set of points. Sparse approaches perform data association based on the local appearance of points. For this reason they can be used when no prior information about the relative position between the clouds is available. This comes at the cost of a more complex system. On the contrary, dense approaches align two clouds by considering every point and using simple heuristics to determine the data association. Dense methods are usually faster and easier to implement than sparse approaches and therefore, preferred for tracking purposes. As a counterpart however, they are more sensitive to wrong initial guesses. Our approach belongs to the class of dense algorithms.

The Iterative Closest Point (ICP) algorithm [2] is one of the earliest and most used techniques for registering point clouds. It is an iterative algorithm that refines an initial estimate of the relative transformation. At each step the algorithm tries to match pairs of points between the two clouds starting from the current transform estimate. Once these correspondences are determined, an improved transformation is usually computed through the Horn [10] formula that minimizes the Euclidean distance between corresponding points. After its introduction, ICP has been subject to several improvements. Most notably, Chen *et al.* [11], replaced the Euclidean distance error metric used by the original ICP with a more robust point-to-plane criterion. This captures the idea that the points measured by the sensors are sampled from a locally continuous and smooth surface.

Segal *et al.* [3] developed a probabilistic version of ICP called Generalized-ICP (GICP). GICP models the sensor noise and utilizes the local continuity of the surface sampled through the cloud. This algorithm, is a point-to-plane variant of ICP that exploits the surface normals when assigning the weight to each matching correspondence in the objective function. The error metric minimized is still a scaled distance between 3D points. In contrast to GICP, our method utilizes an extended measurement vector composed by both the point coordinates and its surface normal. Therefore our error metric measures a distance in a 6D space. Billings *et al.* [12] extended GICP by modeling measurement noise in each shape, whether isotropic or anisotropic.

Steinbrücker *et al.* [5] proposed Dense Visual Odometry (DVO) that takes advantage of the additional light intensity channel available on RGB-D sensors. The main idea behind this approach is to compute an image containing the neighborhood of the edges in the RGB image. Thanks to the depth channel these edges correspond to 3D points, and thus they can be straightforwardly reprojected in the image plane. The transformation is found by minimizing the reprojection distance of the edges on that plane. The objective function thus minimizes a set of 2D distance and this further reduces the observability of the transform resulting in a narrow convergence basin. On the other end, thanks to the reduced amount of data considered by DVO, it can run at high frame rates. This makes the assumption of a good initial guess almost always verified. Unfortunately, DVO suffers from the noise and the blur affecting moving RGB cameras, and it is sensitive to illumination conditions. In fact, this is more a limitation of the sensor than of the algorithm, however these issues make the approach inadequate to operate in scenarios characterized by poor illumination and robots moving fast. The main advantage coming from the use of the RGB channel is that in case of poor structure in the 3D data (e.g. when looking at a flat wall), the algorithm is still able to track the camera pose without getting lost if some texture is present. Similarly to DVO, our method uses a line of sight criterion to find correspondences. Our approach, however, rejects false matching point pairs based on normals.

Newcomb *et al.* proposed Kinect Fusion (KINFU) [4], which represents a major breakthrough in dense depth mapping. It is a complete system that combines components of mapping and point cloud registration. The implementation takes advantage of the brute force of the GPU to effectively update the environment representation. The camera tracking is a point-to-plane version of ICP that uses image projection to determine the correspondences. This tracker, however, can easily fail if the sensor is moved too fast resulting in ICP to get lost. As in the case of GICP, our approach shows an increased robustness to these kind of situations.

Magnusson *et al.* [6] proposed to represent the points with 3D Normal Distribution Transforms (3D-NDT). The idea behind this method is to model the scene, similarly to [3], with a set of small Gaussian distributions computed from the neighborhood of each point. Using this representation it is possible, like in the other algorithms, to apply standard numerical optimization methods to compute the transformation between two point clouds. Our method uses an NDT-like

representation of the scene, but the statistics are computed on an image projection instead of a voxel-grid or a KD-tree.

III. ICP

The problem of registering two point clouds consists in finding the rotation and the translation that maximize the overlap between the two clouds. More formally, let $\mathcal{P}^{\mathrm{r}} = \{\mathbf{p}^{\mathrm{r}}_{1:N^{\mathrm{r}}}\}$ and $\mathcal{P}^{\mathrm{c}} = \{\mathbf{p}^{\mathrm{c}}_{1:N^{\mathrm{c}}}\}$ be the two set of points, we want to find the transformation $\mathbf{T}^*$ that minimizes the distance between corresponding points in the two scenes:

$$\mathbf{T}^* = \underset{\mathbf{T}}{\operatorname{argmin}} \sum_{\mathcal{C}} \left(\mathbf{p}^{\mathrm{c}}_i - \mathbf{T} \oplus \mathbf{p}^{\mathrm{r}}_j\right)^T \mathbf{\Omega}_{ij} \left(\mathbf{p}^{\mathrm{c}}_i - \mathbf{T} \oplus \mathbf{p}^{\mathrm{r}}_j\right). \tag{1}$$

In Eq. 1 the symbols have the following meaning:

- $\mathbf{T}$ is the current estimate of the transformation that maps $\mathcal{P}^{\mathrm{r}}$ in the reference frame of $\mathcal{P}^{\mathrm{c}}$;
- $\mathbf{\Omega}_{ij}$ is an information matrix that takes into account the noise properties of the sensor or of the surface;
- $\mathcal{C} = \{\langle i, j\rangle_{1:M}\}$, is a set of correspondences between points in the two clouds. If $\langle i, j\rangle \in \mathcal{C}$, means that the point $\mathbf{p}^{\mathrm{r}}_j$ in the cloud $\mathcal{P}^{\mathrm{r}}$ corresponds to the point $\mathbf{p}^{\mathrm{c}}_i$ in the cloud $\mathcal{P}^{\mathrm{c}}$;
- $\oplus$ is an operator that applies the transformation $\mathbf{T}$ to the point $\mathbf{p}$. If we use the homogeneous notation for transformations and points, $\oplus$ reduces to the matrix-vector product.

In practice the correspondences are not known, however, in presence of a good approximation for the initial transform, they can be found through some heuristic like nearest neighbor. In its most general formulation, ICP iteratively refines an initial transform $\mathbf{T}$ by alternating the search for correspondences and the solution of Eq. 1. At each new step the correspondences are recomputed by taking into account the most recent transformation.

Over time ICP has evolved in a large number of variants of increased robustness and performance, which differ by the choice of the information matrix $\mathbf{\Omega}_{ij}$ or by the heuristic chosen to find the correspondences. By setting $\mathbf{\Omega}_{ij}$ to the identity, the problem reduces to the original ICP formulation and minimizes the Euclidean distance between corresponding points. Choosing an $\mathbf{\Omega}_{ij}$ with a null eigenvalue along the normal direction, minimizes the point-to-plane metric. In a similar way, the heuristics to determine the correspondences can be refined by taking into account the appearance of the neighborhood of the points.

IV. DENSE NORMAL BASED POINT CLOUD REGISTRATION

Our algorithm is an instance of ICP that deviates from the general scheme presented in Section III. Rather than just considering the Euclidean distance between the points, it takes into account the local properties of the surface both in the search for correspondences, and in the computation of the alignment.

Given the raw sensor data (3D scan or depth image), we compute the point cloud by applying an unprojection function, as discussed in Section IV-A. Once we have the 3D cloud, our method labels each of its points with the properties of the surface around them, namely normal and curvature. We describe in Section IV-B how this operation is carried on efficiently. Then, our algorithm finds the correspondences based on the computed features, as explained in Section IV-C. Subsequently, we determine a candidate transformation by solving a least squares problem that minimizes the distance between the corresponding features, as described in Section IV-D. This differs from all the ICP variants that just minimize the distance between corresponding points. Compared to the more traditional point-to-plane metric, our least squares formulation allows to solve an additional degree of freedom in the orientation of the surfaces, and it has a larger convergence basin. The procedure is iterated up to a maximum number of times, or until convergence is reached.

We remark that two matching points in two different clouds are unlikely to be the same point in the space. Thus, minimizing the Euclidean distance of points that are not the same, introduces an arbitrary error. We overcome this issue by reformulating the correspondence between points as a partial overlap between small surfaces centered in the points, and introducing an error metric that takes into account this aspect.

A. Projection and Unprojection of a 3D Point Cloud onto a Range Image

At low level, 3D sensors provide an indirect measure of the cloud. For instance, depth cameras provide a depth image where the value of the pixel $(u,\ v)$ has the depth d of the object closest to the observer, and lying on a beam passing through that pixel. To obtain a 3D cloud, one needs to apply an unprojection function that depends on the camera parameters. Similarly, a 3D laser provides for each point an azimuth θ, elevation ϕ and the range d measured at that elevation. Typically, both azimuth and elevation are subject to quantization and one can see a 3D scan as a range image where $(\theta,\ \phi)$ are the coordinates of a pixel on a spherical surface. The value d of the pixel is its depth. Thus, without loss of generality, we can introduce a projection function $\mathbf{s} = \pi(\mathbf{p})$ that maps a point $\mathbf{p}$ from the Cartesian to the measurement space. The point in measurement space $\mathbf{s}$ is either a $(u\ v\ d)$ triplet in case of a depth camera, or a $(\theta\ \phi\ d)$ triplet in case of a 3D scanner. Let then be the function $\mathbf{p} = \pi^{-1}(\mathbf{s})$ the inverse of a projection function, π^{-1} maps a triplet of sensor coordinates into a point in the Cartesian space.

The first step of our algorithm is to apply the unprojection function to the raw measurements to compute a Cartesian representation of the cloud. This is done only once at the beginning of the computation, for both input measurements.

B. Extraction of the Statistics of the Surface

A point measurement gathered by a range sensor is a sample from a piece-wise continuous surface. This intuition is at the base of the point-to-plane metric used by Chen *et al.* in [11] and GICP [3].

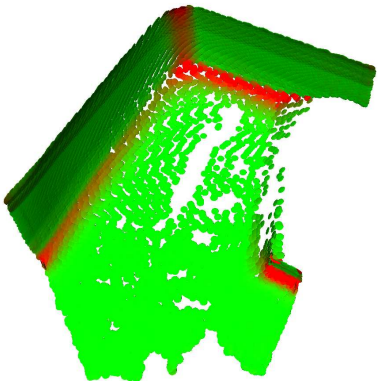

Fig. 2. This figure shows the effect of extracting the surface statistics. According to their curvature σ, green ellipsoids corresponds to points lying on flat regions while red ones to corners.

We locally characterize the surface around a point $\mathbf{p}_i$ with its normal $\mathbf{n}_i$ and curvature σ_i. To compute the normal we extract the covariance matrix of the Gaussian distribution $\mathcal{N}_i^{\mathrm{s}}(\mu_i^{\mathrm{s}},\ \mathbf{\Sigma}_i^{\mathrm{s}})$ of all the points that lie in a sphere of radius R centered in the query point $\mathbf{p}_i$. In our experiments R ranged between 10 cm for depth camera datasets to 25 cm for the 3D laser ones. If the surface is well defined, it can be approximated by a plane and only one eigenvalue of the covariance will be substantially smaller than the other two. The normal is taken as the smallest eigenvector and, if needed, reoriented towards the observer.

More formally, for each point $\mathbf{p}_i$ we compute the mean μ_i^s and the covariance $\mathbf{\Sigma}_i^s$ of the Gaussian distribution as follows:

$$\mu_i^s = \frac{1}{|\mathcal{V}_i|}\sum_{\mathbf{p}_j\in\mathcal{V}_i}\mathbf{p}_i \tag{2}$$

$$\mathbf{\Sigma}_i^s = \frac{1}{|\mathcal{V}_i|}\sum_{\mathbf{p}_j\in\mathcal{V}_i}(\mathbf{p}_i-\mu_i)(\mathbf{p}_i-\mu_i)^T \tag{3}$$

where $\mathcal{V}_i$ is the set of points composing the neighborhood of $\mathbf{p}_i$ and μ_i is the centroid of $\mathcal{V}_i$.

A naive implementation of the above algorithm would require expensive queries on a kd-tree where the cloud is stored to determine the points inside the sphere. Subsequently, to compute the sums in Eq. 2 and 3, we need to iterate on this set twice. To speed up the calculation, we use an approach based on integral images, described in [13], where we evaluate Eq. 2 and 3 in constant time. Once we have the parameters of the Gaussian, we compute its eigenvalue decomposition as follows:

$$\mathbf{\Sigma}_i^{\mathrm{s}} = \mathbf{R}\begin{pmatrix}\lambda_1 & 0 & 0\\ 0 & \lambda_2 & 0\\ 0 & 0 & \lambda_3\end{pmatrix}\mathbf{R}^T. \tag{4}$$

Here $\lambda_{1:3}$ are the eigenvalues of $\mathbf{\Sigma}_i^{\mathrm{s}}$ in ascending order, and $\mathbf{R}$ is the matrix of eigenvectors that represent the axes of the ellipsoid approximating the point distribution. We use the curvature $\sigma_i = \lambda_1 / (\lambda_1+\lambda_2+\lambda_3)$ to discriminate how well the surface is fitted by a plane (see [14] for more details). The smaller the σ, the flatter is the surface around the point.

In practice, due to the sensor noise, even surfaces that are perfectly planar will not have a 0 curvature (or alternatively an eigenvalue that is null). To reduce the effect of this noise, when needed, we modify the covariance matrix $\mathbf{\Sigma}_i^{\mathrm{s}}$ by imposing a "disc" shape. This is done by adjusting the length of the axis of the ellipsoid:

$$\mathbf{\Sigma}_i^{\mathrm{s}} \leftarrow \mathbf{R}\begin{pmatrix}\epsilon & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}\mathbf{R}^T \tag{5}$$

where ϵ is a small coefficient, in our experiments we set ϵ to 0.001. If the surface is not well approximated by a local plane, we leave $\mathbf{\Sigma}_i^{\mathrm{s}}$ untouched. This process has been introduced first by GICP.

At the end of this procedure, each point $\mathbf{p}_i$ is labeled with its surface characteristics $\langle \mu_i^{\mathrm{s}},\ \mathbf{\Sigma}_i^{\mathrm{s}},\ \sigma_i\rangle$. Figure 2 shows its typical outcome.

C. Correspondence Finding Through Projection

Similar to [4] and [5] our approach selects the correspondences by using a line-of-sight criterion. Namely, both clouds are projected on a range image whose viewpoint is the actual estimate of the transformation, and points that fall in the same pixel that have compatible normals and curvature are said to correspond. While the approach by itself is straightforward, an efficient implementation is not, due to the potentially large amount of data the algorithm has to manipulate. In the remainder of this section we describe a procedure that reduces the memory movements and does not manipulate the clouds. We assume the point clouds are stored in arrays and are not necessarily ordered in any way. To describe our procedure we first introduce the concept of *index image*. Given a projection model, $\pi(\Re^3) \rightarrow \Re^2$, an index image $\mathcal{I}$ is an image where the pixel $\mathcal{I}_{uv} = i$ contains the *index* of the point $\mathbf{p}_i$ in the array such that $\pi(\mathbf{p}_i) \rightarrow (u,v,d)^T$. If multiple points fall in the same pixel we select the closest and with a normal oriented towards the center of projection. This operation can be simply implemented by a depth buffer.

Let $\mathcal{I}(\pi,\mathcal{P})$ be an index image computed from the cloud $\mathcal{P}$ through the projection π. Once at the beginning of the iterations, we compute the current index image by projecting all points of the current cloud $\mathcal{P}^{\mathrm{c}}$:

$$\mathcal{I}^{\mathrm{c}} = \mathcal{I}(\pi,\mathcal{P}^{\mathrm{c}}). \tag{6}$$

Since our optimization procedure does not move the current cloud, $\mathcal{I}^{\mathrm{c}}$ stays fixed during the entire alignment, and we do not need to recompute it at every iteration.

Conversely, we recompute the index image $\mathcal{I}^{\mathrm{r}}$ of the reference cloud $\mathcal{P}^{\mathrm{r}}$ at each iteration, after applying the actual estimate $\mathbf{T}$ that maps the reference cloud in the frame of the current cloud:

$$\mathcal{I}^{\mathrm{r}} = \mathcal{I}(\pi,\mathbf{T}\oplus\mathcal{P}^{\mathrm{r}}). \tag{7}$$

Here $\oplus$ applies the transformation $\mathbf{T}$ to the entire cloud $\mathcal{P}^{\mathrm{r}}$.

Finally, from $\mathcal{I}^{\mathrm{r}}$ and $\mathcal{I}^{\mathrm{c}}$ we generate a candidate correspondence for each pixel coordinate as $\langle i,j\rangle_{uv} = \langle \mathcal{I}^{\mathrm{c}}_{uv}, \mathcal{I}^{\mathrm{r}}_{uv}\rangle$.

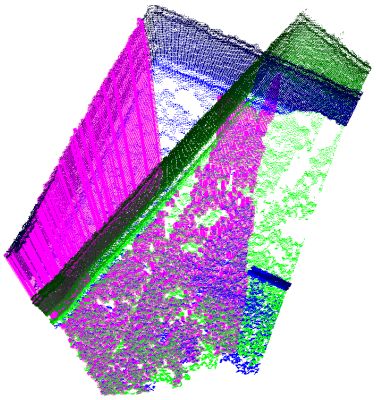

Fig. 3. This picture shows found correspondences as violet lines connecting corresponding points in the blue and green clouds.

A candidate correspondence $\langle i, j \rangle$ between the point $\mathbf{p}_i^{\mathrm{c}}$ and the point $\mathbf{p}_j^{\mathrm{r}}$ is discarded if one of the following holds:

- Either $\mathbf{p}_i^{\mathrm{c}}$ or $\mathbf{p}_j^{\mathrm{r}}$ do not have a well defined normal;
- The distance between the point in the current cloud and the reprojected point in the reference cloud is larger than a threshold:

$$\|\mathbf{p}_i^{\mathrm{c}} - \mathbf{T} \oplus \mathbf{p}_j^{\mathrm{r}}\| > \epsilon_d; \tag{8}$$

- The magnitude of the log ratio of the curvatures of the points is greater than a threshold:

$$|\log \sigma_i^{\mathrm{c}} - \log \sigma_j^{\mathrm{r}}| > \epsilon_\sigma; \tag{9}$$

- The angle between the normal of the current point and the reprojected normal of the reference point is greater than a threshold:

$$\mathbf{n}_i^{\mathrm{c}} \cdot \mathbf{T} \oplus \mathbf{n}_j^{\mathrm{r}} < \epsilon_n. \tag{10}$$

In our experimental setup we set ϵ_d, ϵ_n and ϵ_σ respectively to 0.5 m, 0.95 and 1.3 for depth camera datasets, while we used 1.5 m, 0.9 and 1.3 for 3D laser data. When processing the same class of datasets (depth camera or laser scans) we kept the parameters fixed to these nominal values.

Figure 3 illustrates an example of the correspondence selection. By using index images, we avoid copying points, normals and covariance matrices in auxiliary structures, resulting in an increased speed.

D. Determining the Transformation Given the Correspondences

Once we have a set of candidate correspondences $\mathcal{C} = \langle i, j \rangle_{1:M}$, we compute the transformation between the two frames by using an iterative least squares formulation. We recall that, given the i^{th} point in a cloud, we have the following information from the previous steps: the Cartesian coordinates $\mathbf{p}_i$, the surface curvature σ_i, the surface normal $\mathbf{n}_i$ and the covariance matrix $\mathbf{\Sigma}_i^{\mathrm{s}}$.

Let $\tilde{\mathbf{p}}$ be a point with normal $\tilde{\mathbf{p}} = \begin{pmatrix} \mathbf{p}^T & \mathbf{n}^T \end{pmatrix}^T$ and $\mathbf{T}$ be a transformation matrix parametrized by a rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{t}$. The $\oplus$ operator on points with normals is:

$$\tilde{\mathbf{p}}' = \mathbf{T} \oplus \tilde{\mathbf{p}} = \begin{pmatrix} \mathbf{R}\mathbf{p} + \mathbf{t} \\ \mathbf{R}\mathbf{n} \end{pmatrix} \tag{11}$$

Given a correspondence and a current transform $\mathbf{T}$, the error $\mathbf{e}_{ij}(\mathbf{T})$ is a 6D vector

$$\mathbf{e}_{ij}(\mathbf{T}) = \left(\tilde{\mathbf{p}}_i^{\mathrm{c}} - \mathbf{T} \oplus \tilde{\mathbf{p}}_j^{\mathrm{r}}\right). \tag{12}$$

Substituting Eq. 12 in Eq. 1 leads to the following objective function:

$$\sum_{\mathcal{C}} \mathbf{e}_{ij}(\mathbf{T})^T \tilde{\mathbf{\Omega}}_{ij} \mathbf{e}_{ij}(\mathbf{T}). \tag{13}$$

Here $\tilde{\mathbf{\Omega}}_{ij}$ is a 6×6 information matrix that scales the components of the errors. Ideally, we want to have an $\tilde{\mathbf{\Omega}}_{ij}$ that rotates corresponding points so that their normals align, and that penalizes mostly the distance along the normal direction, while neglecting the distance along the plane tangents. To this end, we impose independence between the translational and the normal components, and we select an $\tilde{\mathbf{\Omega}}_{ij}$ of the following form

$$\tilde{\mathbf{\Omega}}_{ij} = \begin{pmatrix} \mathbf{\Omega}_i^{\mathrm{s}} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Omega}_i^{\mathrm{n}} \end{pmatrix}. \tag{14}$$

Here $\mathbf{\Omega}_i^{\mathrm{s}} = \mathbf{\Sigma}_i^{\mathrm{s}-1}$ is the surface information matrix around the current point $\mathbf{p}_i^{\mathrm{c}}$, and $\mathbf{\Omega}_i^{\mathrm{n}}$ is the information matrix of the normal. If the curvature is small enough we set $\mathbf{\Omega}_i^{\mathrm{n}}$ as follows:

$$\mathbf{\Omega}_{ij}^{\mathrm{n}} \leftarrow \mathbf{R} \begin{pmatrix} \frac{1}{\epsilon} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \mathbf{R}^T, \tag{15}$$

otherwise we impose $\mathbf{\Omega}_i^{\mathrm{n}}$ to the identity. With these information matrices, a correspondence between two points is minimized by allowing the points to slide onto each other along the tangential direction of the surface, and rotating them so that their normals align. Note that setting $\mathbf{\Omega}_i^{\mathrm{n}}$ to zero makes our objective function identical to GICP.

Our algorithm minimizes Eq. 13 by using a local parametrization of the perturbation in the following form: $\mathbf{\Delta T} = \begin{pmatrix} \Delta t_x & \Delta t_y & \Delta t_z & \Delta q_x & \Delta q_y & \Delta q_z \end{pmatrix}^T$. It consists of the translation vector $\mathbf{\Delta t}$ and the imaginary part of the normalized quaternion $\mathbf{\Delta q}$. By using a damped Gauss-Newton algorithm, at each iteration, our method solves the following linear system

$$(\mathbf{H} + \lambda \mathbf{I})\mathbf{\Delta T} = \mathbf{b}. \tag{16}$$

Here, $\mathbf{H} = \sum \mathbf{J}_{ij}^T \tilde{\mathbf{\Omega}}_{ij} \mathbf{J}_{ij}$ is the approximated Hessian and $\mathbf{b} = \sum \mathbf{J}_{ij}^T \tilde{\mathbf{\Omega}}_{ij} \mathbf{e}_{ij}(\mathbf{T})$ is the residual. Once we determined the perturbation $\mathbf{\Delta T}$ from Eq. 16, we refine the current one by applying $\mathbf{\Delta T}$ as:

$$\mathbf{T} \leftarrow \mathbf{\Delta T} \oplus \mathbf{T}. \tag{17}$$

The Jacobian $\mathbf{J}_{ij}$ is obtained by computing the derivative of Eq. 12 evaluated in $\mathbf{\Delta T} = \mathbf{0}$:

$$\mathbf{J}_{ij} = \left.\frac{\partial \mathbf{e}_{ij}(\mathbf{\Delta T} \oplus \mathbf{T})}{\partial \mathbf{\Delta T}}\right|_{\mathbf{\Delta T}=\mathbf{0}} = \begin{pmatrix} -\mathbf{I} & 2[\mathbf{T} \oplus \mathbf{p}_j^{\mathrm{r}}]_\times \\ \mathbf{0} & 2[\mathbf{T} \oplus \mathbf{n}_j^{\mathrm{r}}]_\times \end{pmatrix} \tag{18}$$

where $[\mathbf{p}]_\times$ is the cross product matrix of the vector $\mathbf{p}$. In practice, by exploiting the block structure of the Jacobian and its substantial sparsity, it is possible to construct efficiently the linear system in Eq. 16.

TABLE I

MEAN OF RELATIVE TRANSLATIONAL AND ROTATIONAL ERROR FOR THE DEPTH CAMERA DATASETS. GREEN CELLS IN THE TABLE HIGHLIGHT THE BEST RESULT FOR EACH SPECIFIC DATASET AMONG ALL THE COMPARED ALGORITHMS (GICP, DVO, NDT AND NICP).

	Relative Translational Error [m]				**Relative Rotational Error [deg °]**			
Dataset	**GICP**	**DVO**	**NDT**	**NICP**	**GICP**	**DVO**	**NDT**	**NICP**
high-fast-fly	**0.284**	1.150	0.452	0.291	20.203	41.831	**17.552**	24.119
high-fast-rot	0.892	0.824	1.142	**0.357**	35.386	24.718	29.081	**14.260**
high-fast-tr	0.115	0.174	0.132	**0.096**	5.933	5.393	6.438	**5.308**
high-medium-fly	0.076	0.249	0.225	**0.069**	**4.760**	9.372	9.833	5.886
high-medium-rot	0.123	0.276	0.217	**0.103**	8.403	8.636	9.960	**8.369**
high-medium-tr	0.046	0.147	0.112	**0.043**	2.538	4.614	5.602	**2.524**
high-slow-fly	0.055	0.128	0.150	**0.053**	**3.189**	4.713	4.857	3.300
high-slow-rot	**0.090**	0.226	0.213	**0.090**	3.956	6.091	5.854	**3.785**
high-slow-tr	**0.038**	0.150	0.064	0.042	1.706	4.531	2.911	**1.433**
low-fast-fly	**0.282**	1.204	0.704	0.294	**12.783**	36.736	19.885	14.643
low-fast-rot	0.329	1.094	0.614	**0.316**	20.775	29.819	26.936	**15.481**
low-fast-tr	0.073	0.519	0.175	**0.072**	**4.938**	14.499	6.465	4.961
low-medium-fly	**0.069**	0.521	0.280	0.077	**3.750**	13.908	6.937	4.301
low-medium-rot	0.075	0.488	0.185	**0.054**	6.979	12.686	8.703	**6.750**
low-medium-tr	**0.063**	0.346	0.094	**0.063**	3.828	8.872	4.899	**3.741**
low-slow-fly	0.059	0.407	0.162	**0.052**	2.842	14.647	3.968	**2.659**
low-slow-rot	0.100	0.466	0.231	**0.089**	4.321	11.435	5.915	**3.741**
low-slow-tr	0.048	0.418	0.114	**0.039**	1.971	9.059	3.013	**1.632**
medium-fast-fly	0.460	1.282	0.652	**0.343**	33.917	38.484	35.010	**29.471**
medium-fast-rot	0.744	0.614	0.330	**0.285**	27.339	18.163	15.617	**13.920**
medium-fast-tr	0.090	0.217	0.128	**0.086**	5.167	5.716	6.278	**5.012**
medium-medium-fly	0.070	0.422	0.236	**0.058**	4.275	12.702	11.081	**4.132**
medium-medium-rot	0.114	0.280	0.160	**0.076**	8.572	**6.209**	8.479	7.693
medium-medium-tr	**0.036**	0.229	0.091	0.038	1.978	5.998	3.937	**1.953**
medium-slow-fly	0.050	0.204	0.113	**0.040**	2.742	6.106	3.564	**2.560**
medium-slow-rot	0.060	0.212	0.111	**0.055**	2.993	5.245	4.532	**2.433**
medium-slow-tr	**0.032**	0.215	0.074	0.034	1.610	6.240	3.068	**1.432**

V. EXPERIMENTS

We compared our system with several state-of-the-art approaches by using publicly available benchmark data acquired with both depth cameras and laser scanners.

When operating with depth cameras we compared NICP with GICP, DVO, NDT and KINFU. Note that KINFU is not a simple registration algorithm, since it combines a registration and a mapping algorithm. Each time a new cloud is available, KINFU registers it on a local map (which is a point cloud in a global reference frame) and then it adds the new data to the local map. Using local maps helps to reduce drift in the estimate.

To allow for a fair comparison, we built on top of our registration algorithm a mapping component. In the remainder of this paper we will refer to this algorithm as NICP+. In NICP+, the registration occurs always between a new cloud and a local map. Similarly to KINFU, each time new clouds arrive we register them onto the local map and we augment it by adding all the points of the new aligned clouds. To prevent the linear growth in the number of points, we decimate the elements of the local map by voxelizing them at a resolution of 2.5 cm. Points that fall in the same voxel are averaged.

We used the KINFU implementation provided by the Point Cloud Library (PCL) [15], while for DVO and NDT we considered the ROS [16] package suggested by the authors and available on the web. Since GICP is a special case of our algorithm, where the error in the normal part of the optimization stage is neglected, and the correspondences are selected based only on point distance, we used our own GICP implementation. Whereas DVO requires the intensity channel, thus it does not belong to the same class of the other algorithms, we included it in the comparison since it is very fast and stable when the lighting conditions are favorable. Note that our implementation of GICP benefits of all the data structures and of surrounding algorithms that are used in NICP, namely the extraction of the statistics and the calculation of the correspondences. To stress this aspect, we performed additional experiments on 3D laser data where we gradually transformed GICP in NICP, by increasing the weight of the normal component. We did not consider KINFU and DVO for 3D laser data because they require respectively Kinect-like depth images and RGB images. To measure the performance of an algorithm we used the benchmarking tools of Sturm *et al.* [17], and we computed the Relative Pose Error (RPE). The RPE measures the pairwise alignment error between successive poses and it is one of the best metric for the evaluation of visual odometry or camera tracking systems. All datasets have been processed on a i7-3630QM running at 2.4 GHz and with an nVidia GeForce GT 650M graphic card. All the parameters used in the experiments can be found on the website linked before.

As shown in Table II, in terms of processing time, NDT resulted to be the slowest algorithm in computing a single registration. It required in fact about 100 ms. KINFU took

TABLE II

AVERAGE CLOUD REGISTRATION TIME FOR EACH ALGORITHM.

GICP	**DVO**	**NDT**	**KINFU**	**NICP**	**NICP+**
12 ms	3 ms	100 ms	50 ms	12 ms	33 ms

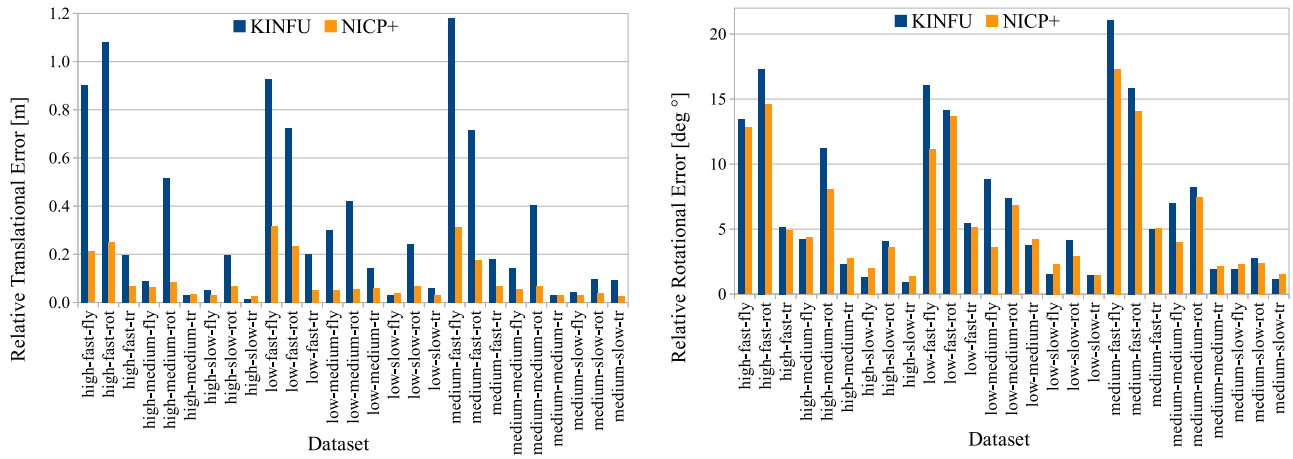

Fig. 4. Mean relative translational (left) and rotational (right) error obtained with KINFU and NICP+ on the depth camera datasets.

about 50 ms to process each new point cloud but this low frame rate is probably mainly due to the low end video card in our system. All other approaches, GICP, DVO, NICP and NICP+, were able to execute in real time computing respectively a registration in about 12 ms, 3 ms, 12 ms and 33 ms. As explained before, NICP+ is slower with respect to NICP because of the mapping component needed to augment the reference cloud. DVO reaches the highest frame rate, since it processes a substantial smaller quantity of data.

A. Depth Camera Datasets

We performed a comparison of the above mentioned approaches on the benchmark data in [8]. Each dataset consists in a sequence of depth and RGB images acquired with a RGB-D camera. For benchmarking purposes, the ground-truth is available. The datasets cover 3 environments of increasing complexity (low, high, medium), with 3 types of motions (rotational, translational, fly) at 3 different speeds (slow, medium, fast). Using datasets where the motion of the camera is high allows to test the robustness of the algorithms to poor initial guesses. Indeed a big camera velocity implies an increasing average distance between two processed frames.

Table I reports the results obtained in processing all the benchmarking datasets with GICP, DVO, NDT and NICP algorithms. For each approach and for each dataset, we processed all the images in sequence and we generated the estimated trajectories. Note that, since these datasets are recorded with a high frame rate, the RPE is computed on poses with a difference in time of 1 second. Green cells in the table highlight the best result for each specific dataset among all the compared algorithms. NICP is almost always more accurate than GICP, DVO and NDT. The effect of using normals in the error function minimization allows to better reduce the rotational error and, as a result, also the translational error benefits from this.

For the reasons explained before, to compare results also with respect to KINFU, we performed an other set of experiments using the NICP variant NICP+. Figure 4 demonstrates how NICP+ performs better with respect to KINFU. The reader might notice that KINFU obtains its worst results when dealing with datasets where the camera does not move slowly. This is due to the fact that KINFU assumes to have a good initial guess, that could not be verified in those cases.

B. 3D Laser Data

Here we show a comparison between GICP, NDT and NICP on 3D laser benchmarking datasets provided by Pomerleau *et al.* in [9]. We present the results on two different datasets:

- **ETH Hauptgebaude** captures the main building of ETH Zurich and the laser moved basically along a straight direction in a long corridor;
- **ETH Stairs** is a record of the *internal* and *external* part of a building, the laser moved along a corridor and it went outside passing through two doorways. Between the doorways the scanner climbed five steep stairs.

Like for the depth camera datasets we evaluated the algorithms computing the RPE error. Instead of considering an interval of time of one second between two poses, however, we computed the error at each frame. This has been done because the frame rate of 3D laser datasets are much lower.

The images in Fig. 1 and Fig. 5 illustrate the ground-truth, and the reconstructions obtained with NICP, NDT and GICP. While NICP was able to consistently reconstruct both scenes from the datasets considered, GICP and NDT were not. Note that, despite the fact that the ETH Hauptgebaude dataset is mostly composed of *curved surfaces*, NICP performed very well. Since NICP is an extended version of GICP that makes use of the normals in the error function, we run another experiment to quantify this effect. To this end we executed NICP with different scaling factors for the information matrices $\mathbf{\Omega}^n$ of the normals: 0 (GICP), 0.33, 0.66 and 1 (NICP). Note that when $\mathbf{\Omega}^n$ is null, the block of the error vector relative to the normals is always null. This, together with computing the correspondences based only on point distance, results in NICP behaving as GICP. The more we increase this scaling factor, the more accurate is the result and the lower the standard deviation (Figure 6). These results show that the normals provide relevant information that contributes to increase the results and the robustness of NICP.

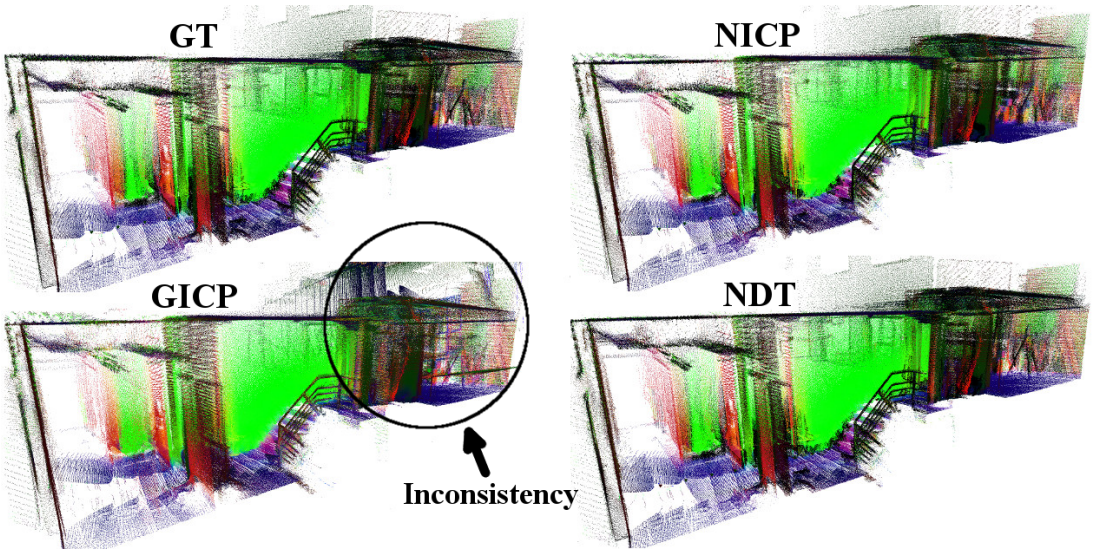

Fig. 5. Side views of the reconstruction of the ETH Stairs dataset. Top-left: ground-truth. Top-right: NICP reconstruction. Bottom-left: GICP reconstruction. Bottom-right: NDT reconstruction. Both NICP and NDT consistenly reconstructed the scene, GICP failed in aligning the part on the back of the stairs.

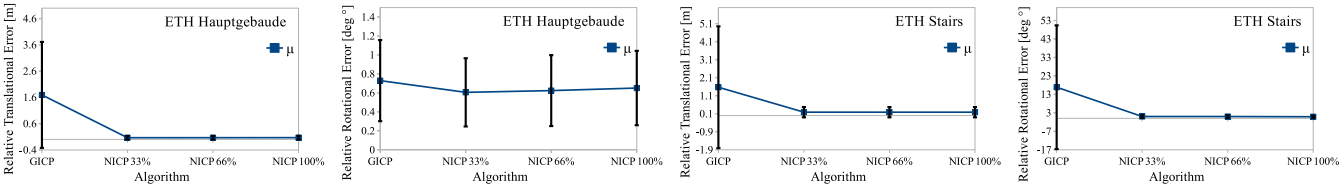

Fig. 6. Evolution of the mean of the RPE, and its standard deviation, when incrementally increasing the weight of the surface normals in the algorithm. The first and second plot illustrate respectively the translational and the rotational part of the error for the ETH Hauptgebaude dataset. The third and fourth plot illustrate respectively the translational and the rotational part of the error for the ETH Stairs dataset.

VI. Conclusions

In this paper we presented a novel system to register point clouds that takes into account an extended measurement vector. We discussed all the relevant steps needed for the implementation of such a system, and we provided open source reference code. We performed statistical experiments comparing our approach to other state-of-the-art methods showing that our algorithm offers better results and higher robustness to poor initial guesses. Moreover, our method demonstrated to be able to run on-line on a mid-range laptop.

In the future we plan to provide a GPU implementation of our system. Since the algorithms and the data structures are highly parallelizable we expect to achieve good performances also on embedded systems.

References

[1] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing icp variants on real-world data sets," *Autonomous Robots*, 2013.

[2] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1992.

[3] A. V. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," in *Proc. of Robotics: Science and Systems (RSS)*, 2009.

[4] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, , P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinect-Fusion: Real-time dense surface mapping and tracking," in *Proc. of the Int. Symposium on Mixed and Augmented Reality (ISMAR)*, 2011.

[5] F. Steinbrücker, J. Sturm, and D. Cremers, "Real-time visual odometry from dense RGB-D images," in *ICCV Workshops*, 2011.

[6] M. Magnusson, T. Duckett, and A. J. Lilienthal, "Scan registration for autonomous mining vehicles using 3d-ndt," *Journal on Field Robotics*, 2007.

[7] J. Serafin and G. Grisetti, "Using Augmented Measurements to Improve the Convergence of ICP," in *4th International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, ser. LNCS 8810. Springer, 2014, pp. 566–577.

[8] F. Pomerleau, S. Magnenat, F. Colas, M. Liu, and R. Siegwart, "Tracking a depth camera: Parameter exploration for fast icp," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 3824–3829.

[9] F. Pomerleau, M. Liu, F. Colas, and R. Siegwart, "Challenging data sets for point cloud registration algorithms," *The International Journal of Robotics Research*, vol. 31, no. 14, pp. 1705–1711, Dec. 2012.

[10] B. K. Horn, H. M. Hilden, and S. Negahdaripour, "Closed-form solution of absolute orientation using orthonormal matrices," *Journal of the Optical Society of America*, 1988.

[11] J. Chen and G. Medioni, "Object modeling by registration of multiple range images," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1991.

[12] S. D. Billings, E. M. Boctor, and R. H. Taylor, "Iterative most-likely point registration (imlp): a robust algorithm for computing optimal shape alignment," *PloS one*, vol. 10, no. 3, 2015.

[13] S. Holzer, R. B. Rusu, M. Dixon, S. Gedikli, and N. Navab, "Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 2684–2689.

[14] M. Pauly, M. Gross, and L. P. Kobbelt, "Efficient simplification of point-sampled surfaces," in *Proceedings of the conference on Visualization'02*. IEEE Computer Society, 2002, pp. 163–170.

[15] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

[16] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[17] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.