Analyzing LiDAR Scan Skewing and its Impact on Scan Matching

Anas Al-Nuaimi*, Wilder Lopes†, Paul Zeller‡, Adrian Garcea*, Cassio Lopes†, Eckehard Steinbach*
*Chair of Media Technology, Technische Universität München (TUM), Munich, Germany. {first.last@tum.de}
†Signal Processing Lab, Universidade de São Paulo (USP), São Paulo, Brazil. {wilder@usp.br,cassio@lps.usp.br}
‡NavVis GmbH, Munich, Germany. {paul.zeller@navvis.com}

***Abstract*—This paper presents a systematic study of the scan skewing problem. Scan skewing is the non-rigid deformation of point clouds acquired by LiDAR's and is the result of their sequential scanning nature. We theoretically analyze the impact of skewing on scan matching and subsequently quantify the impact using synthetic LiDAR data with controlled skew distortions. We also show how the Geometric-Algebra LMS, an iterative point set registration filter, can be tuned to incorporate skewing aware weights to reduce the impact of skewing on scan matching. Results with real 3D LiDAR data are also presented.**

I. Introduction

LiDAR (Light Detection and Ranging) can be used to capture point clouds of indoor environments. A LiDAR sensor infers the 3D locations of points on the sensed surface by measuring the time required for laser beams emitted in known directions to be reflected back to the sensor.

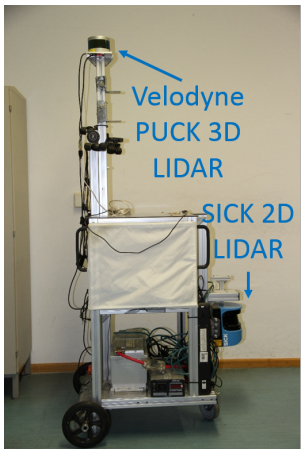
LiDAR's typically provide a relatively sparse scan of the environment as seen in the scans shown in Figure 1b. Individual scans of a LiDAR sensor mounted on a moving platform, as that shown in Figure 1a, can be matched to estimate the scanner movement and "stitch" the scans (see Figure 1c) to provide extensive and densely sampled point clouds.

Accurate *Scan matching* is challenging as tiny errors especially in the rotation lead to propagating errors that manifest themselves in trajectory drift. A fundamental problem that can lead to rotation estimation errors in scan matching is that of scan *skewing*. It's a phenomenon that is a direct consequence of platform mobility and the fact that one scan is comprised of *sequentially* scanned points within a certain time frame.

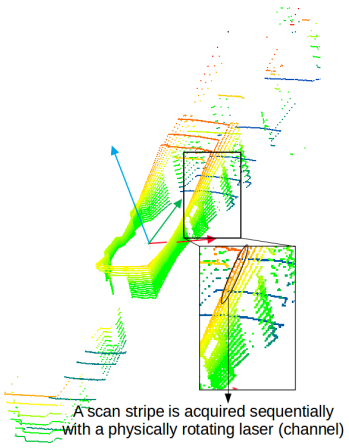
Few works address skewing explicitly. In [2] the alignment of the previous scan is used to de-skew the current scan before matching it. In [3] the scan skew is part of the Iterative Closest Point (ICP) core. We found no systematic study on scan skewing. Hence, the objectives of this work are two-fold:

1) Systematically investigate and understand the impact of scan skewing (Sections II & IV).
2) Reduce the impact of skewing without the aid of additional sensors. For that we derive a variant the adaptive GA-LMS filter [4] that incorporates the point to plane error metric (Section V-A).

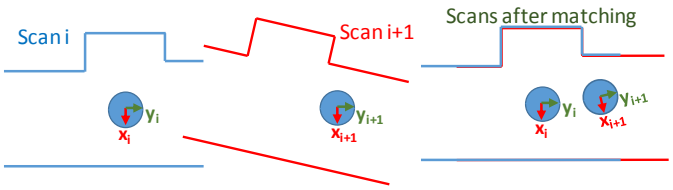
Part of this work was done during a research visit of Anas Al-Nuaimi to USP's Signal Processing Lab upon invitiation from Prof. Cassio Lopes. It was funded by USP (grant: USP PRPG 02/2015) and the TUM Graduate School.
978-1-5090-2425-4/16/$31.00 ©2016 IEEE



(a) Mapping trolley with LiDAR



(b) A LiDAR scan



(c) Scan matching (scans depicted from above)

Fig. 1: Our mapping trolley [1] with the Velodyne PUCK LiDAR mounted on top is shown in (a). The LiDAR acquires scans made of 3D points as the one shown in (b) where the height is color-coded. Each tilted stripe of points (point plane) is scanned by one mechanically rotating laser sequentially. One scan represents a relatively sparse sampling of the environment. Matching multiple scans as depicted in (c) is necessary to reconstruct an indoor environment in high quality.

II. The Skewing Problem

A. Scan Distortion

LiDAR's typically have a physically rotating laser emitter and hence points are captured *sequentially* while completing a full rotation. Hence, a full rotation generates a scan. The PUCK, which is used in this work, has 16 such lasers (*channels*) rotating around the vertical axis each tilted by an angle from the horizontal plane resulting in the distinct point arrangement seen in Figure 1b.

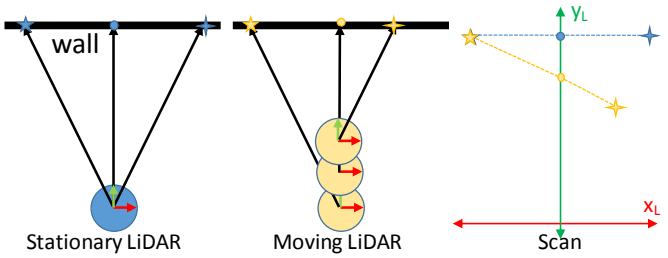
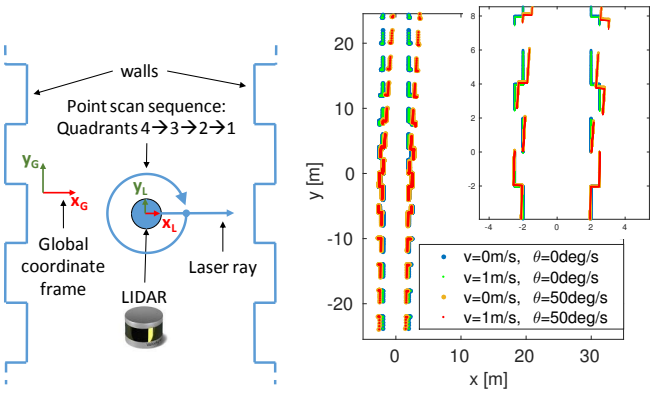


Fig. 2: Illustration of the scan skewing problem. The LiDAR is depicted from above by the circular shape with two vectors identifying its local coordinate system. Points on a wall are scanned sequentially in the following order: star, circle and diamond. In blue, a stationary sensor is shown as opposed to the realistic case of a moving sensor shown in yellow. The movement of the local coordinate system while scanning the points translates into a non-rigid deformation of the geometry.



(a) Simulation setup (b) Resulting scans

Fig. 3: A 2D scan generated for a synthetic indoor environment made up of planar walls. The setup is shown in (a) from a top view. The LiDAR's local coordinate system is shown as x_L, y_L as opposed to the global coordinate system shown as x_G, y_G. The simulated LiDAR laser rotates around z clock-wise. Different movement scenarios are simulated involving a translation along y_G by a constant speed v and an angular rotational speed θ around the (invisible) z-axis. The outcome is shown in (b). Especially LiDAR rotation during scanning is seen to incur strong distortions. Also, the farther away the points are from the sensor origin, the larger the distortion. Comparing quadrant 1 (scanned last) to quadrant 4 (scanned first) the distortion is seen, as expected, to increase as the scan proceeds towards later points since the local reference frame will have moved the most.

Assuming the LiDAR is stationary during one full rotation of the laser and disregarding sensor noise and calibration errors, a geometrically correct sampling of the environment can be achieved. However, as already mentioned, such sensors are typically mounted on mobile platforms. Hence, the assumption of stationarity is mostly violated. Accordingly, the sensor will have a slightly different pose when scanning each point.

The LiDAR captures scans in its local coordinate frame. Since one scan comprises all points scanned in one complete rotation of the laser, the movement of the local coordinate frame when iterating through the points should ideally be accounted for. Unfortunately, the sensor is not capable of doing that as it does not know how it moved. Retrieving the motion, after all, is the ultimate target of scan matching.

The inadvertent outcome of this phenomenon is a distortion in the scan as illustrated in Figure 2. Depending on the type of motion, different types of distortions with varying severity can occur as seen in Figure 3. Especially rotational movement is seen to incur noticeable non-rigid distortion affecting mainly points that are far away from the sensor origin. Obviously, the stronger the motion, the stronger the distortions.

B. Theoretical Effect on Mapping

Let's start with any point $\mathbf{p_i}$ expressed in the global coordinate frame as $\mathbf{p_i} = [\mathbf{p_i}(x), \mathbf{p_i}(y), \mathbf{p_i}(z)]^T$. In scan j's local coordinate frame it can be expressed as $\mathbf{p_i^j} = \left[\mathbf{p_i^j}(x), \mathbf{p_i^j}(y), \mathbf{p_i^j}(z)\right]^T$ where the relationship between $\mathbf{p_i}$ and $\mathbf{p_i^j}$ is given by

$$\mathbf{p_i^j} = \mathbf{R_{i(j)}} \cdot (\mathbf{R_0^j} \cdot \mathbf{p_i} + \mathbf{t_0^j}) + \mathbf{t_{i(j)}}, \tag{1}$$

where $\mathbf{R_{i(j)}}$ and $\mathbf{t_{i(j)}}$ are the individual rotation matrix and translation vector due to skewing of point i while capturing scan j. $\mathbf{R_0^j}$ and $\mathbf{t_0^j}$, on the other hand, is the coordinate transformation that expresses points described in the global coordinate frame (0) in the local coordinate frame (j).

Assuming a constant velocity and rotation speed motion model, the following approximations can be used: $\mathbf{R_{i(j)}} \approx \mathbf{R_{i(j+1)}}$, $\mathbf{t_{i(j)}} \approx \mathbf{t_{i(j+1)}}$. This means that for consecutive scans the individual deformation for *corresponding* points is almost the same. Indeed, Figure 4 shows real scan results that agree with this assumption. Intuitively, a like-wise non-rigid transformation of two scans should not impact the matching. This can be proven by rewriting the point set alignment cost function between two consecutive scans (see [5]) as follows:

$$\begin{aligned}
\mathcal{F} &= \sum_i ||\mathbf{R_{i(j+1)}}(\mathbf{R_j^{j+1}}\mathbf{p_i^j} + \mathbf{t_j^{j+1}}) + \mathbf{t_{i(j+1)}} - \\
&\quad (\mathbf{R_{i(j)}}\mathbf{p_i^j} + \mathbf{t_{i(j)}})||_2^2 && (2)\\
&\approx \sum_i ||\mathbf{R_{i(j+1)}}(\mathbf{R_j^{j+1}}\mathbf{p_i^j} + \mathbf{t_j^{j+1}}) - \mathbf{R_{i(j)}}\mathbf{p_i^j}||_2^2 && (3)\\
&\approx \sum_i ||\mathbf{R_{i(j+1)}}(\mathbf{R_j^{j+1}}\mathbf{p_i^j} + \mathbf{t_j^{j+1}} - \mathbf{p_i^j})||_2^2 && (4)\\
&= \sum_i ||\mathbf{R_j^{j+1}}\mathbf{p_i^j} + \mathbf{t_j^{j+1}} - \mathbf{p_i^j}||_2^2 && (5)
\end{aligned}$$

The terms $\mathbf{t_{i(j)}}$ and $\mathbf{t_{i(j+1)}}$ canceled by virtue of the subtraction combined with the above assumption $\mathbf{t_{i(j)}} \approx \mathbf{t_{i(j+1)}}$. Similarly, the assumption $\mathbf{R_{i(j)}} \approx \mathbf{R_{i(j+1)}}$ allows factoring out the deformation rotations in (4). Since a rigid rotation does not change the norm, $\mathbf{R_{i(j+1)}}$ drops out of the equation. What remains is a cost function dependent only on the scan-to-scan transformation expressed by $\mathbf{R_j^{j+1}}$ and $\mathbf{t_j^{j+1}}$ and is hence skewing independent.

The preceding derivation might tempt the reader to conclude that the scan skewing problem does not affect the scan matching. However, the derivation is based on the assumption

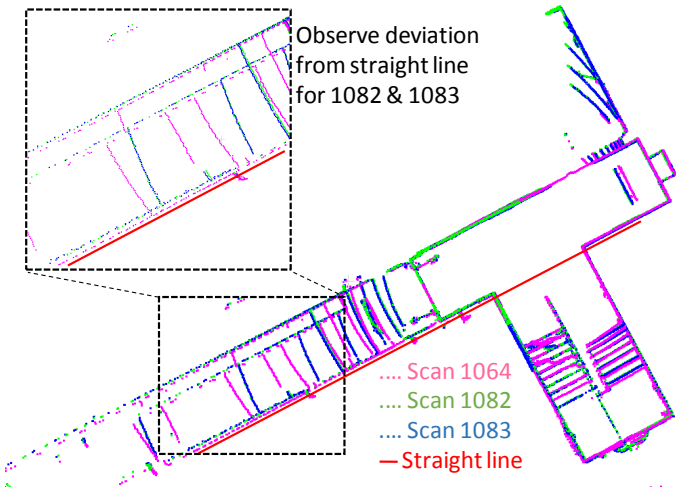


Fig. 4: Real scans produced by the PUCK seen from above after matching. Three scans are shown: two consecutive scans (1082,1083) captured as the mapping platform was turning, and one scan (1064 i.e. around 2 seconds earlier) captured before the platform started to turn. Noticeable scan skewing is observed for scans 1082 and 1083 seen by the deviation of their points from the ideal straight line. The skewing is seen to be almost the same for corresponding points (notice their near optimal alignment along all parts).

that the point-wise deformation is similar for geometrically corresponding points. As will be shown in Section IV this is not necessarily true as motions get more complex. Especially changes in the rotation speed mean the distortion similarity assumption is violated.

The preceding argumentation should convince the reader that accounting for the skewing is important for accurate scan matching. For mapping platforms equipped with inertial measurement units (IMU) the data can be *de-skewed* by estimating the local coordinate frame transformation between individual point scans and using the estimated transformation to correct the distortion. This is possible because IMUs typically run at a substantially higher sampling rate and their data can be considered reliable for short time frames as those spanned between two consecutive scans. Good IMUs, however, are quite expensive. Moreover, to be usable for de-skewing they require precise calibration w.r.t. the LiDAR. The latter can be quite challenging [6].

Based on the preceding discussions, the objective of this work is to systematically investigate scan skewing and develop a method to reduce its negative impact. For that we first built an ICP-based scan matching framework which is explained in greater detail in Section III.

III. SCAN MATCHING FRAMEWORK

We built a scan matcher using C++ and the Point Cloud Library (PCL) [7]. The pseudo-code is listed in Algorithm 1.

The function `matchScan` implements the ICP algorithm. As its name implies, it currently implements a scan-to-scan matching, hence, a map is not built. It uses the previously estimated transformation to already predict the motion of

Algorithm 1: The scan matching pseudo-code. Estimate the individual scan-to-scan transformations T[i].

```
oldScan ← loadPointCloud (pointCloudFile[1]);
T[1] ← Identity(4,4);
for i ← 2 to numScanFiles do
    newScan ← loadPointCloud (pointCloudFile[i]);
    T[i] ← matchScan (newScan,oldScan,T[i-1]);
    oldScan ← newScan;
```

the current scan. This simple trick leads to a substantial improvement in the scan matching performance. The function's pseudo-code is shown in Algorithm 2. It can be seen that iterations of correspondence estimation and subsequent transformation estimation are performed until convergence.

Algorithm 2: `matchScan(scanSrc,scanTar,Tprev)`.

```
hasConverged ← false;
T ← Tprev;
while !hasConverged do
    scanSrcPreAligned ← transform(scanSrc,T);
    corrs ← computeCorrespondences(scanSrcPreAligned,scanTar);
    Trefine ← estimateTfromCorrs(corrs,scanSrcPreAligned,scanTar) ;
    T ← Trefine ·T ;
    hasConverged ← checkConvergence(Trefine);
```

`estimateTfromCorrs` implements three algorithms to estimate the alignment from the correspondences:

1) Standard ICP minimization with the point-to-point (pt2pt) metric [8]. Henceforth termed "ICPpt2pt".
2) Standard ICP minimization with the LLS point-to-plane (pt2pl) metric [9]. Henceforth termed "ICPpt2pl".
3) Minimization using a new variant of the Geometric Algebra Adaptive filter we developed in our previous work [4], [5]. Here, we derive and use the filter which uses the point-to-plane metric explained in Section V-A.

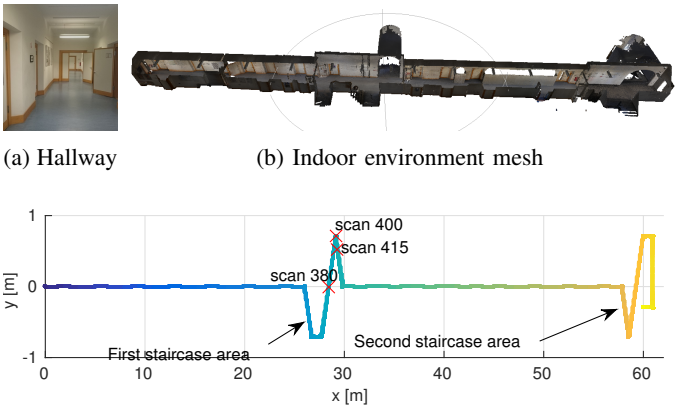
IV. EXPERIMENTAL VERIFICATION OF SKEWING IMPACT

To systematically investigate the problem of scan matching from the perspective of skewing we use a synthetic scan dataset. The dataset is generated using our virtual LiDAR that simulates a 3D PUCK LiDAR. The LiDAR scans points sequentially while rotating anti-clockwise around the z-axis from 0 to 360°(this is important information when designing the weighting function (23)).

To make the data as realistic as possible the LiDAR translations and rotations are smooth curves generated by fitting spline functions on a series of user-defined way points with associated directions.

We generated synthetic scans by moving a virtual LiDAR along the trajectory shown in Figure 5c inside the 3D mesh model shown in Figure 5b which does not contain skewing and which was captured in the hallway shown in Figure 5a. We created four different dataset variants by adding controlled distortions with the characteristics shown in Table I.

Figure 6 shows the estimated trajectories obtained by running standard ICPpt2pt and ICPpt2pl (Section III) on the



(a) Hallway (b) Indoor environment mesh

(c) Trajectory

Fig. 5: Synthetic and real LiDAR data are acquired for the hallway shown in (a). The textured 3D mesh of the hallway used to generate the synthetic data is shown with parts of the ceiling and walls removed in (b). (c) shows the virtual trajectory followed by the simulated LiDAR. The gradual color change modulates the time index starting with blue and ending with yellow. Three specific scans are marked by their numbers.

TABLE I: The four synthetic dataset variants with different types of distortions.

variant	1	2	3	4
Symmetric Gaussian noise	no	yes	no	yes
Skewing	no	no	yes	yes

synthetic scan datasets. Skewing noticeably affects the scan matching leading to errors that manifest themselves in the shape of trajectory drift.

Having shown that skewing does indeed incur scan matching errors it is time to investigate them in more detail. We use our ground-truth data to analyze the scan-to-scan errors in terms of rotation (around the +ve z-axis) and translation. These errors are shown in Figure 7. As expected, the results show that in regions involving LiDAR rotation the scan matching exhibits visible rotation errors. What is however unexpected is the observation that the type of error is not a function of rotation direction only but also of the change in *absolute* rotation speed. Increases in the absolute rotation speed lead to a +ve error (equivalent to clock-wise drift) for +ve rotation angles and -ve error (equivalent to counter-clock-wise drift) for -ve rotation angles. Conversely, decreasing absolute rotation speed switches the type of error. Another interesting observation is that when the rotation speed stops changing (local peaks and minima in the ground-truth scan-to-scan rotation) the rotation error goes to zero. The latter observation verifies the arguments of Section II-B which suggested that a constant rotation speed incurs similar skewing deformation for corresponding points which was shown to have no impact on the alignment of consecutive scans.

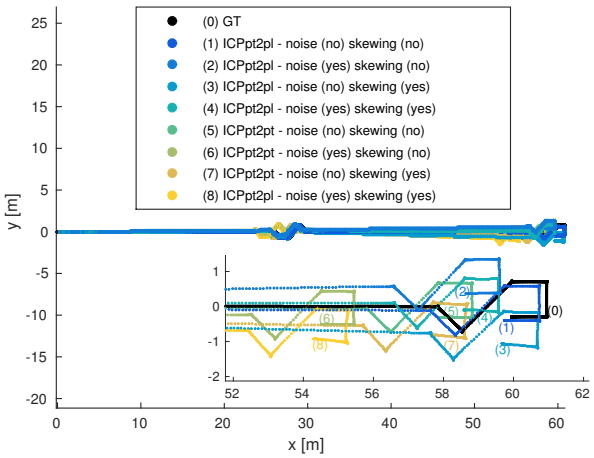


Fig. 6: Estimated trajectories after scan matching of the synthetic dataset explained in Section IV using standard scan-to-scan ICP (see Section III). The trajectories are seen to coincide with the ground truth (GT) trajectory to a large degree. However, the zoom-in reveals that pt2pl scan matching (curves 1-4) is substantially more accurate than pt2pt scan matching (curves 5-8) in accordance with wide-spread belief throughout the community (See e.g. [10]). As expected, the trajectories corresponding to no added distortions (1 & 5) are the closest to the ground truth among their peers that use the same metric. Also, the trajectories with skewed data (3 & 7) have a visible drift.

V. Skewing-Aware Matching with GA-LMSpt2pl

Having theoretically derived and experimentally verified the impact of skewing we seek to minimize its impact. Since we now know which points are affected the most by skewing it is conceivable to tune the contributions of points towards the final estimation from ICP by suitable weighting. For that we want to deploy the Geometric Algebra Alignment filter we developed (GA-LMS) [4]. It is a filter that computes the alignment of two point sets connected by correspondences iteratively which allows to control the contribution of different correspondences by suitable weighting. The GA-LMS is comparable to the ICPpt2pt. However, we have already shown (Section IV) that a pt2pl metric results in a more robust trajectory estimation. Hence, we first derive here the GA-LMS with point-to-plane metric, henceforth termed "GA-LMSpt2pl".

A. Deriving GA-LMSpt2pl

1) Rotation Estimation: To derive the GA-LMSpt2pl we rewrite the Linear-Algebra pt2pl cost function

$$\mathcal{F}(\mathbf{R}) = \frac{1}{K}\sum_{i=1}^{K} \|(\mathbf{y_i} - \mathbf{R}\mathbf{x_i}) \cdot \mathbf{n_i}\|^2 \tag{6}$$

(with $\cdot$ representing the dot product) using a GA formulation similar to the one we present in [4]. Replacing the rotation matrix $\mathbf{R}$ in (6) by the *rotor operator* $r(\cdot)\widetilde{r}$ of GA yields

$$J(r) = \frac{1}{K}\sum_{i=1}^{K} \left|(y_i - r x_i \widetilde{r}) \cdot n_i\right|^2. \tag{7}$$

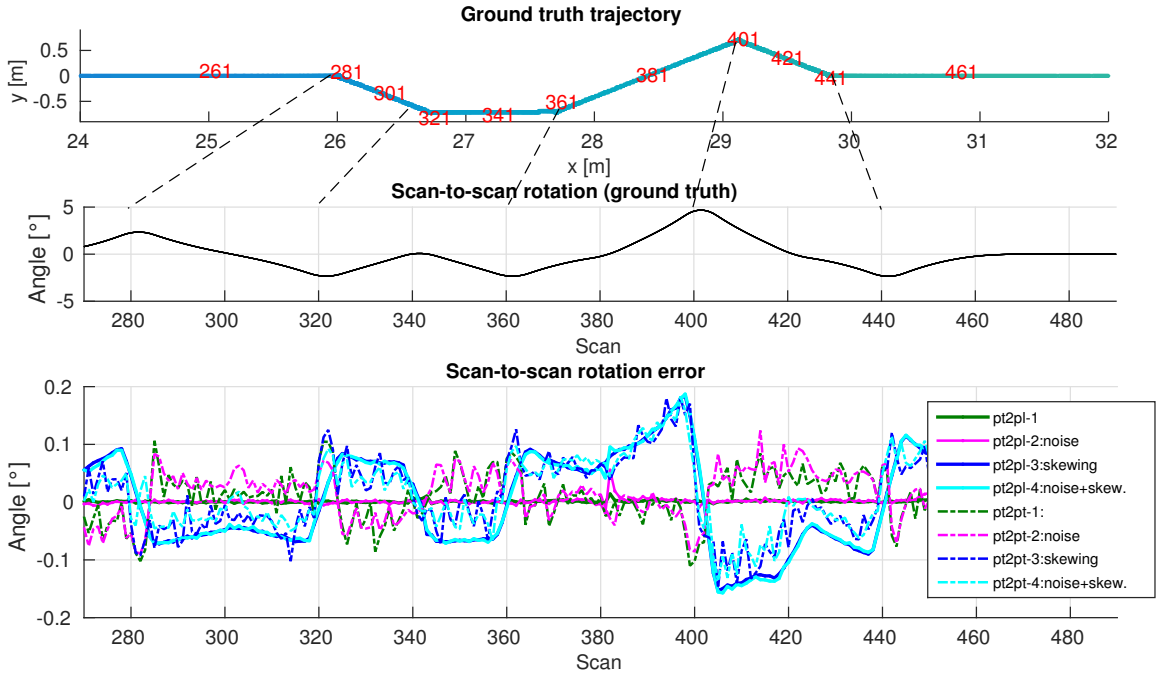


Fig. 7: Scan-to-scan errors for a selected part of the trajectory. The upper two figures show the ground-truth (GT) trajectory and the GT scan-to-scan rotation around the vertical axis, respectively. Notice, the LiDAR is not simulated to be rigidly attached to the platform (no kinematic model) hence a straight trajectory does not necessarily imply no LiDAR rotation. Focusing on the rotation errors (bottom) the following observation can be made: pt2pt matching results in fluctuating and less stable rotation error (compare solid to dashed curves); When adding just noise, pt2pl scan matching, as expected, still results in very accurate rotation estimation (compare solid green and pink curves) owing to the robustness introduced by relying on surface normals instead of individual points; Scans with skewing lead to two types of rotation angle error: +ve error or -ve error (see oscillating pattern of dark blue curves); The type of rotation error, surprisingly, does not depend on the rotation direction only but also on the change in *absolute* rotation speed. For example, in the scan range 320-360 the rotation is always negative, however, we observe -ve rotation error when the absolute rotation speed is increasing and +ve error when the absolute rotation speed is decreasing. This pattern reverses for +ve rotation angles. The rotation error does not necessarily go to zero when the rotation speed goes to zero (300, 380, 420). The rotation error goes to zero only when the rotation speed *change* goes to zero (scans around 280, 320, 340, 360, 400, 440). The observation about when the error goes to zero is in accordance with the derivation of Section II-B which showed that when the rotation speed is constant, the skewing does not affect the registration.

Notice, from now on vectors are no longer shown in boldface: in the context of GA, vectors are multivectors for which we adopt the notation introduced in [4]. Hence, multivectors are shown in small non-bold font. Multiplications generally imply GA multiplications. The grade operator is denoted by $\langle\rangle$. The $\cdot$ operator represents inner product as opposed to $\wedge$ which denotes GA exterior product. $\widetilde{r}$ denotes the reversion of r.

The AF provides an estimate for the bivector r via a recursive rule of the form,

$$r_i = r_{i-1} + \mu G, \tag{8}$$

where i is the (time) iteration, μ is the AF step size, and G is a multivector-valued update quantity related to the estimation error (analogous to the standard formulation in [11, p. 143].

A proper selection of G is required to enforce $J(r_i) < J(r_{i-1})$ at each iteration. This work adopts the steepest-descent rule [11], [12] to devise an Adaptive Filter (AF) that is designed to follow the opposite direction of the reversed gradient of the cost function, namely $\widetilde{\nabla} J(r_{i-1})$ (note the analogy between the reversed $\widetilde{\nabla}$ and the hermitian conjugate ∇^* from the standard formulation). This way, G is proportional to $\widetilde{\nabla} J(r_{i-1})$,

$$G \triangleq -B\widetilde{\nabla} J(r_{i-1}), \tag{9}$$

in which B is a general multivector, in contrast with the standard case in which B would be a matrix [11]. Setting B equal to the identity matrix results in the steepest-descent update rule ([11], Eq. 8-19). In GA though, the multiplicative identity is the multivector (scalar) 1 ([13], p.3), thus $B = 1$.

Define $\mathcal{C}(r) \triangleq \left|(y_i - r x_i \widetilde{r}) \cdot n_i\right|^2$. Using the GA scalar product and the definition of magnitude of a multivector (see [4]), $\mathcal{C}$ can be expanded as

$$\begin{aligned} \mathcal{C}(r) &= \left|(y_i - r x_i \widetilde{r}) \cdot n_i\right|^2 \\ &= [(y_i - r x_i \widetilde{r}) \cdot n_i] * \widetilde{[(y_i - r x_i \widetilde{r}) \cdot n_i]} \\ &= \left\langle [(y_i - r x_i \widetilde{r}) \cdot n_i][(y_i - r x_i \widetilde{r}) \cdot n_i] \right\rangle \\ &= [(y_i - r x_i \widetilde{r}) \cdot n_i]^2, \end{aligned} \tag{10}$$

in which $\widetilde{[(y_i - r x_i \widetilde{r}) \cdot n_i]} = [(y_i - r x_i \widetilde{r}) \cdot n_i]$ due to fact that vectors (grade-1 multivectors) are not affected by the reversion

operator. Plugging (10) back into (7), the gradient of $J(r)$ can be obtained as follows

$$\nabla_r J(r) = \frac{1}{K}\sum_{i=1}^{K}\nabla_r \mathcal{C}(r), \tag{11}$$

where $\nabla_r \mathcal{C}(r) = \nabla_r[(y_i - rx_i\widetilde{r})\cdot n_i]^2$. This way,

$$\begin{aligned}\nabla_r \mathcal{C}(r) &= 2[(y_i - rx_i\widetilde{r})\cdot n_i]\nabla_r[(y_i - rx_i\widetilde{r})\cdot n_i] \\ &= -2[(y_i - rx_i\widetilde{r})\cdot n_i]\nabla_r[(rx_i\widetilde{r})\cdot n_i] \\ &= -2[(y_i - rx_i\widetilde{r})\cdot n_i]\nabla_r\langle rx_i\widetilde{r}n_i\rangle.\end{aligned} \tag{12}$$

The last term in (12) is expanded as

$$\nabla_r\langle rx_i\widetilde{r}n_i\rangle = \partial_r\langle \dot{r}M_i\rangle + \partial_r\langle T_i\dot{\widetilde{r}}\rangle. \tag{13}$$

The terms $M_i = x_i\widetilde{r}n_i$ and $T_i = n_i r x_i$ are obtained applying the *cyclic reordering property* $\langle AD\cdots C\rangle = \langle D\cdots CA\rangle = \langle CAD\cdots\rangle$ [14]. The first term on the right-hand side of (13) is $\partial_r\langle \dot{r}M_i\rangle{=}M_i$ ([15], Eq. 7.10), and the second term is $\partial_r\langle T_i\dot{\widetilde{r}}\rangle{=}{-}\widetilde{r}T_i\widetilde{r}{=}{-}\widetilde{r}(n_i r x_i)\widetilde{r}$ (see Appendix of [4])). Plugging back into (13),

$$\begin{aligned}\nabla_r\langle rx_i\widetilde{r}n_i\rangle &= x_i\widetilde{r}n_i - \widetilde{r}(n_i r x_i)\widetilde{r} \\ &= \widetilde{r}[(rx_i\widetilde{r})n_i - n_i(rx_i\widetilde{r})] \\ &= 2\widetilde{r}(rx_i\widetilde{r}\wedge n_i),\end{aligned} \tag{14}$$

where the relation $ab - ba = 2(a\wedge b)$ was used ([16], p.39). Substituting (14) in (12) results in

$$\begin{aligned}\nabla_r \mathcal{C}(r) &= -4[(y_i - rx_i\widetilde{r})\cdot n_i]\widetilde{r}(rx_i\widetilde{r}\wedge n_i) \\ &= 4[e_i\cdot n_i]\widetilde{r}(n_i\wedge rx_i\widetilde{r}),\end{aligned} \tag{15}$$

which allows to finally obtain the gradient of $J(r)$

$$\nabla_r J(r) = \frac{4}{K}\sum_{i=1}^{K}(e_i\cdot n_i)\widetilde{r}(n_i\wedge rx_i\widetilde{r}). \tag{16}$$

Substituting (16) into (9) (with $B = 1$, as aforementioned) results in

$$G = \frac{4}{K}\sum_{i=1}^{K}(e_i\cdot n_i)(n_i\wedge rx_i\widetilde{r})r, \tag{17}$$

which upon plugging into (8) yields

$$\boxed{r_i = r_{i-1} + \mu\frac{4}{m}\sum_{i=1}^{m}(e_i\cdot n_i)(n_i\wedge rx_i\widetilde{r})r}, \tag{18}$$

where a substitution of variables was performed to enable writing the algorithm in terms of a rank captured by m, i.e., one can select $m\in[1,K]$ to choose how many correspondence pairs are used at each iteration. This allows for balancing computational cost and performance, similar to the Affine Projection Algorithm (APA) rank [12], [11]. If $m = K$, (18) uses all the available points, originating the *geometric-algebra steepest-descent* algorithm. This paper focuses on the case $m = 1$ (one pair per iteration) which is equivalent to approximating $\widetilde{\nabla}J(r)$ by its *current value* in (17) [11],

$$\frac{4}{K}\sum_{i=1}^{K}(e_i\cdot n_i)(n_i\wedge rx_i\widetilde{r})r \approx 4(e_i\cdot n_i)(n_i\wedge rx_i\widetilde{r})r, \tag{19}$$

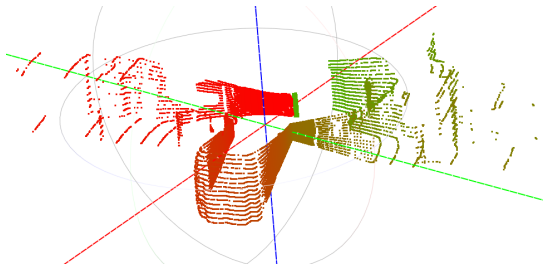


Fig. 8: The skewing iteration weights computed according to (23) for scan #405. The weights are color-coded ($[0,1]->$[Green,Red]).

resulting in the GA-LMS update rule,

$$\boxed{r_i = r_{i-1} + \mu(i)(n_i\wedge rx_i\widetilde{r})r}, \tag{20}$$

in which $\mu(i) = \mu_c\cdot(e_i\cdot n_i)$ and the factor 4 was absorbed by μ_c. Note that the step size μ is weighted by the inner product between the error vector e_i and the normal n_i. Thus, the net step size $\mu(i)$ becomes *time-dependent*, having its value down-weighted as long as e_i has its norm decreased at each AF iteration.

2) Estimating translation: The standard Linear-Algebra cost function for the pt2pl metric [9], [17] incorporates an operator that accounts for rotation and translation instead of just the rotation matrix R of (6). Replacing this operator using GA is not straight-forward and we opt to use a similar strategy as the one we presented in [5] to estimate the translation. Hence, first the centroids of the source and target point clouds which are connected by correspondence are computed and subtracted from the points of their respective clouds (see [5, eq. (2)]) leaving only the rotation inside the cost function. After rotation estimation, the translation is computed as the subtraction of the rotated source centroid from the target centroid. This decoupled approach, although standard for pt2pt metrics [18], is inferior to pt2pl which estimates rotation and translation jointly. This is especially problematic in our case as there is a very high density of points near the sensor origin (see Figure 1b) making the centroids an unreliable measure of translation when these points happen to lie on walls. To compensate for this issue we use the curvature weights, explained in Section V-B, to compute weighted centroids instead, increasing the contribution of geometrically salient points in the centroid computation.

B. Skewing-Aware GA-LMSpt2pl

We use the GA-LMSpt2pl, derived in Section V-A, to reduce the impact of scan skewing. More specifically, (20) is modified to incorporate a skewing weight $w(i)$:

$$r_i = r_{i-1} + w(i)\mu(i)(n_i\wedge rx_i\widetilde{r})r \tag{21}$$

where

$$w(i) = f_w(w_c(i), w_s(i)) \tag{22}$$

with $f_w()$ being a function that combines the two weights. The skewing weight of iteration (correspondence) i, $w_s(i)$, is calculated as follows:

$$w_s(i) = cos(\theta_z(i)/4) \tag{23}$$

where $\theta_z(i)$ is the in-plane angle of the vector connecting the source scan origin to point i and is limited to $[0, 360)°$(i.e. $\theta_z(i) = atan(\mathbf{p_i^j}(y)/\mathbf{p_i^j}(x))$. The weight for the first scanned points ($\theta_z \approx 0$) is almost 1 whereas points scanned towards the end ($\theta_z \approx 360$) have a weight that is almost zero as seen in Figure 8. Without loss of generality this function is designed to incorporate the property of the simulated scanner (see Section IV). A scanner rotating in a different direction or starting from a different angle (as the one used to get the real data used in Section VI) requires a simple and obvious adjustment of the weighting function.

The curvature weight w_c, on the other hand, accounts for the importance of a point geometrically as points with a higher curvature typically lie on edges and thus are more distinctive for the alignment process. It is calculated using:

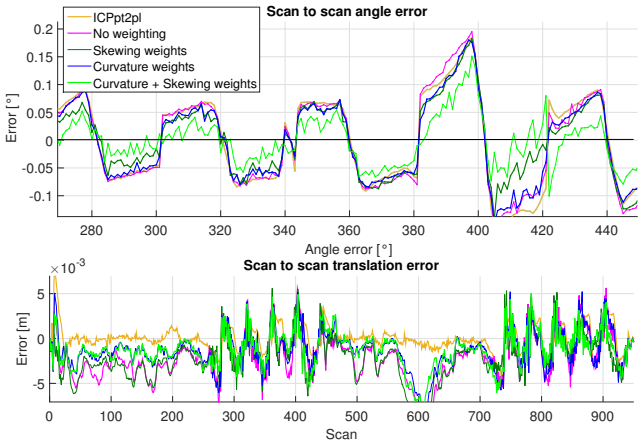
$$w_c(i) = max(0.25, min(c_i/c_r, 1.0)) \tag{24}$$

where c_i is the curvature value of source point i and c_r is a reference curvature value indicating points lying on a highly curved surface. These weights are also used for the centroid computation which is critical for the translation estimation (see SectionV-A2).

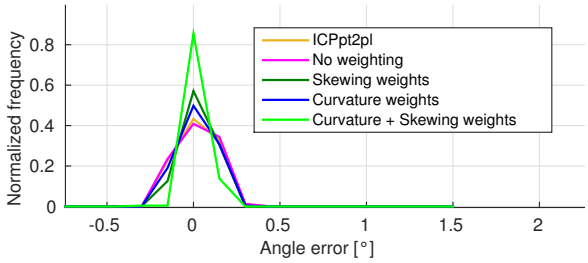
Figure 9 compares scan matching between different variants of GA-LMSpt2pl that are obtained when turning on and off skewing weighting and curvature weighting. The GA-LMSpt2pl (Section V-A) delivers a similar rotation estimation accuracy as the standard ICP pt2pl proving it to be a credible substitute. More importantly, as seen in Figure 9a, combining skewing and curvature weighting substantially reduces the rotation estimation errors in the scan regions involving rotations. Indeed, Figure 9b shows a significantly improved rotation error distribution with errors mostly being around $0°$. Also, while the standard GA-LMSpt2pl consistently understimates the translation due to the unreliability of the centroid estimation (see Section V-A2), the curvature weight effectively reduces this problem leading to a more accurate trajectory that is less "compressed" as seen in Figure 9c.

VI. Evaluation with Real Data

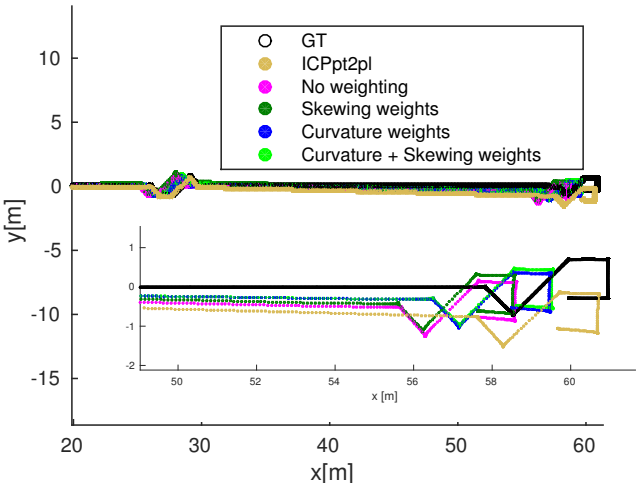
We obtained real data corresponding to the synthetic scan data (see Section IV) using the trolley-mounted Velodyne PUCK shown in Figure 1a. We computed GT data using Hector SLAM [19] running on SICK 2D LiDAR data. We performed scan matching using ICPpt2pl and GA-LMSpt2pl with skewing and curvature weighting. The GA-LMSpt2pl histogram of scan-to-scan angle estimation error is only moderately better compared to ICPpt2pl as opposed to the results with synthetic data shown in Figure 7. Also here we observe the trajectory "compression" when using GA-LMSpt2pl. These results call upon the combination of GA-LMSpt2pl with ICPpt2pl. The results in Figure 10 show that the combination



(a) Scan-to-scan errors

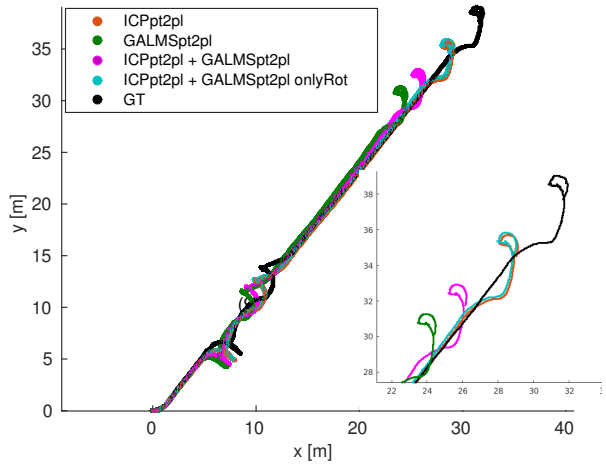


(b) Scan-to-scan rotation error histogram

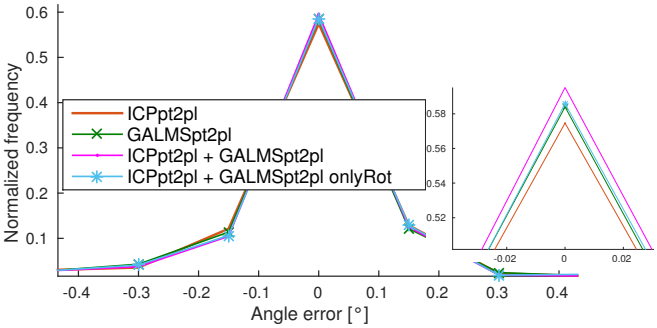


(c) Estimated trajectories (with zoom-in)

Fig. 9: Scan matching is run on the third dataset variant involving only skewing (Table I). The scan matching is conducted using ICPpt2pl and different variants of the GA-LMSpt2pl (Section V-A) involving skewing weights (23) and/or curvature weights (24). The angle error curves in (a) show that a combination of skewing and curvature weights according to (22) ($f_w(w_s, w_c) = w_s \cdot w_c$) reduces orientation estimation errors (compare green and orange curves) in the regions with strong rotations which is furthermore confirmed by the rotation error histogram in (b). The translation error curves show that the curvature weight also helps in reducing the underestimation of the translation in the GA-LMSpt2pl (compare blue and green curves to pink curve). This leads to a more accurate trajectory that is less "compressed" as seen in (c).



(a) Estimated trajectories (with zoom-in)



(b) Scan-to-scan rotation error histogram (with zoom-in)

Fig. 10: Evaluation with real data. ICPpt2pl + GA-LMSpt2pl implies running ICPpt2pl and then refining the outcome with GA-LMSpt2pl. GA-LMSpt2pl includes curvature and skewing weights with $f_w(w_s, w_c) = max(w_s, w_c)$. onlyRot means the GA-LMSpt2pl performs only rotation refinement.

indeed results in the smallest scan-to-scan errors. Also the trajectory compression is significantly reduced. However, the centroid estimation seems to be undoing part of the translation estimation obtained from ICPpt2pl. Hence, in a final variant we combined ICPpt2pl with GALMSpt2pl that only refines the rotation. This combination has a better scan-to-scan angle error distribution compared to ICPpt2pl while having a trajectory that does not have the same compression problems as those of pure GALMSpt2pl.

The reason why the improvements here are not as significant as those observed in Figure 9 may relate to measurement noise. The results of Figure 9 were produced with the synthetic dataset with only skewing (no noise). Here, however, we are dealing with a real dataset that also includes real noise. While we verified that noticeable improvements are achieved in the synthetic dataset with noise and skewing, we note that the real sensor noise is not strictly Gaussian and it is stronger for far away points. The latter suffer the most from skewing and hence accounting for skewing may deliver the expected benefit as the noise component may be more dominant.

VII. CONCLUSIONS

In this paper we systematically investigate the problem of LiDAR scan skewing. We first analyze the source of this phenomenon and derive its impact on scan matching. Using synthetic scan data with controlled distortions we quantify the impact of scan skewing on scan matching and verify the validity of the derivations and explanations. Finally, we derive a new variant of the Geometric Algebra LMS [4] to incorporate the point-to-plane metric. We use it in conjunction with a skewing-aware iteration update scheme to reduce scan matching estimation errors due to skewing. We show significant gains in terms of rotation estimation accuracy on synthetic data. Results with real data show smaller improvements.

Future developments could focus on developing the GA-LMSpt2pl to jointly estimate rotation and translation. Also, the sequential nature of the filter could be exploited to perform matching using subsets of points employing convergence criteria and thus save unnecessary computations.

REFERENCES

[1] R. Huitl, G. Schroth, S. Hilsenbeck, F. Schweiger, and E. Steinbach, "TUMindoor: an extensive image and point cloud dataset for visual indoor localization and mapping," in *IEEE International Conf. on Image Processing*, Orlando, FL, USA, Sep 2012.

[2] F. Moosmann and C. Stiller, "Velodyne slam," in *2011 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2011, pp. 393–398.

[3] M. Bosse and R. Zlot, "Continuous 3d scan-matching with a spinning 2d laser," in *IEEE ICRA 2009*, May 2009, pp. 4312–4319.

[4] W. B. Lopes, A. Al-Nuaimi, and C. G. Lopes, "Geometric-algebra lms adaptive filter and its application to rotation estimation," *IEEE Signal Processing Letters*, vol. 23, no. 6, pp. 858–862, June 2016.

[5] A. Al-Nuaimi, W. Lopes, E. Steinbach, and C. G. Lopes, "6dof point cloud alignment using geometric algebra-based adaptive filtering," in *IEEE Winter Conference on Applications of Computer Vision*, Lake Placid, NY, USA, Mar 2016.

[6] Z. Taylor and J. Nieto, "Motion-based calibration of multimodal sensor arrays," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 4843–4850.

[7] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *IEEE Intern. Conf. on Robotics and Automation*, May 2011, pp. 1–4.

[8] Zhengyou Zhang, "Iterative point matching for registration of free-form curves and surfaces," *Int. J. Comput. Vision*, vol. 13, no. 2, Oct. 1994.

[9] Kok-Lim Low, "Linear least-squares optimization for point-to-plane icp surface registration," *Chapel Hill, Univ. of North Carolina*, vol. 4, 2004.

[10] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *Intern. Conf. on 3-D Digital Imaging and Modeling*. IEEE, 2001, pp. 145–152.

[11] A.H. Sayed, *Adaptive filters*, Wiley-IEEE Press, 2008.

[12] Paulo S. R. Diniz, *Adaptive Filtering: Algorithms and Practical Implementation*, Springer US, 4 edition, 2013.

[13] D. Hestenes and G. Sobczyk, *Clifford Algebra to Geometric Calculus: A Unified Language for Mathematics and Physics*, Fundamental Theories of Physics. Springer Netherlands, 1987.

[14] E. Hitzer, "Introduction to Clifford's Geometric Algebra," *Journal of the Society of Instrument and Control Engineers*, vol. 51, no. 4.

[15] E. Hitzer, "Multivector differential calculus," *Advances in Applied Clifford Algebras*, vol. 12, no. 2, pp. 135–182, 2002.

[16] D. Hestenes, *New Foundations for Classical Mechanics*, Fundamental Theories of Physics. Springer, 1999.

[17] Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image and vision computing*, vol. 10, no. 3.

[18] D. W. Eggert, A. Lorusso, and R. B. Fisher, "Estimating 3-d rigid body transformations: a comparison of four major algorithms," *Machine Vision and Applications*, vol. 9, no. 5-6, pp. 272–290, 1997.

[19] S. Kohlbrecher, O. Von Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *IEEE Intern. Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2011, pp. 155–160.