Point Cloud Registration Refinement in an Urban Environment using 2D Edge-Maps

David Avidar, David Malah, and Meir Barzohar
Andrew and Erna Viterbi Faculty of Electrical Engineering, Technion, Haifa 32000, Israel
davidar@campus.technion.ac.il, malah@ee.technion.ac.il, barzoharmeir@gmail.com

***Abstract*—As 3D point cloud acquisition sensors become increasingly prevalent in urban environments (e.g., LiDAR sensors for autonomous vehicles), the need arises to find efficient ways to align large amounts of such 3D data, often in real-time. In this work, we propose a novel method for 3D point cloud registration refinement in an urban environment (e.g., between Terrestrial LiDAR Scans - TLS - and Airborne LiDAR Scans - ALS), assuming an initial coarse registration is available. The proposed method is based on estimation of the direction of gravity, wall detection, projection of the point clouds on a perpendicular horizontal plane, and conversion into 2D edge-maps. Then, two methods are considered for alignment between the 2D edge-maps: a 2D variant of the well-known ICP (Iterative Closest Point) algorithm, and Edge-Map Phase-Correlation (EMPC). We demonstrate the usefulness of the proposed methods for registration in this challenging task, where the 2D variant of ICP achieves a meaningful advantage over 3D ICP in terms of runtime, while maintaining comparable registration accuracy.**

***Index Terms*—ICP, LiDAR, Phase-Correlation, 3D point cloud, Registration, Urban environment**

I. Introduction

In recent years, sensors for acquisition of 3D data, such as LiDAR scanners or stereo cameras, have been used in various applications, such as 3D urban modeling, topographic surveying, cultural heritage sites preservation, precision agriculture, robotic navigation, and especially, in autonomous driving. Since LiDAR scanners, such as those mounted on autonomous vehicles, often acquire very large volumes of 3D data (e.g., hundreds of thousands or even millions of points per second), the need arises to efficiently align such 3D data, often in real-time.

In this work, we consider the challenge of rigid registration refinement, or alignment in the same coordinate system, of 3D point clouds acquired in an urban environment from terrestrial and airborne viewpoints. Given a large-scale "global" point cloud (e.g., a result of Airborne LiDAR Scanning of a city), a "local" point cloud (e.g., a result of Terrestrial LiDAR Scanning, using a sensor mounted on a ground vehicle), and an initial "coarse" alignment between them, our goal is finding a transformation (with 6 DoF - Degrees of Freedom - 3 for rotation and 3 for translation), that will refine the initial registration as accurately as possible in a computationally efficient way.

An initial coarse registration between the TLS and ALS clouds is obtained by the viewpoint-dictionary-based method, proposed in [1], in which the global point cloud is converted into a dictionary of viewpoints, and each viewpoint is characterized by its *geometric view* of the environment - a panoramic range image. Then, given a local cloud, an efficient dictionary search is performed, based on Phase-Correlation between panoramic range images, quickly finding likely initial registrations between the local and global clouds.

This is followed in [1] by a registration refinement step, using the well-known Iterative Closest Point (ICP) algorithm [2], iteratively minimizing a cost function defined over nearest-neighbor pairs of points between two 3D point cloud. However, it was found in [3] that the registration refinement step in this scenario consists, on average, of 80% of the total registration time per local cloud. Thus, we set out in this work to improve the computational efficiency of the registration refinement step, without compromising on registration accuracy.

II. Proposed method

In this section, it is assumed that an initial coarse registration between the point clouds is available - e.g., using [1].

The proposed method is based on taking advantage of the typical structure of urban scenes in order to efficiently refine the registration between two 3D point clouds (a local cloud and a global cloud) by first converting them into 2D edge-maps. Fig. 1 shows a block diagram of the process used to convert both clouds into edge-maps. The following subsections (II-A - II-D) will expand on each of the algorithmic steps shown in the diagram. We note that both the global and local clouds may be downsampled, using a voxel-grid (as done in [1]), before their conversion to edge-maps. In the diagram, gravity direction estimation is applied only to the local point cloud, as it is assumed that the global cloud is already given in a reference frame in which the z-axis is aligned with the negative gravity direction. If this is not so, the gravity estimation step can be used on the global cloud as well.

After converting the clouds into edge-maps, the edge-maps are aligned in order to find the translation $(\Delta x, \Delta y)$ and rotation angle in the xy-plane $(\Delta\phi)$ necessary for refining the registration between the original 3D point clouds. When aligning the edge-maps, only the relevant part of the global edge-map is used, according to the previously found initial (coarse) registration (see green square in Fig. 2). As described in subsection II-E, we consider two methods for edge-map alignment: Edge-Map Phase-Correlation (EMPC) and Edge-based 2D ICP.

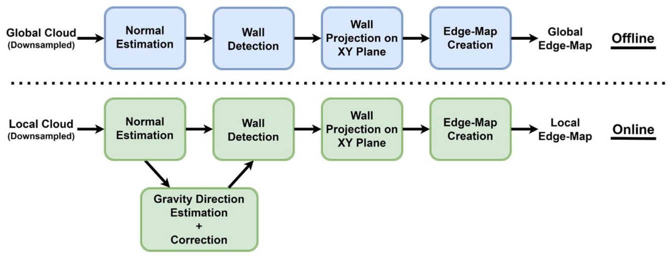

Global Cloud (Downsampled)
Normal Estimation
Wall Detection
Wall Projection on XY Plane
Edge-Map Creation
Global Edge-Map
Offline
Local Cloud (Downsampled)
Normal Estimation
Wall Detection
Wall Projection on XY Plane
Edge-Map Creation
Local Edge-Map
Online
Gravity Direction Estimation + Correction

Fig. 1. Block diagram of the edge-map creation process employed on both the global cloud and local cloud.

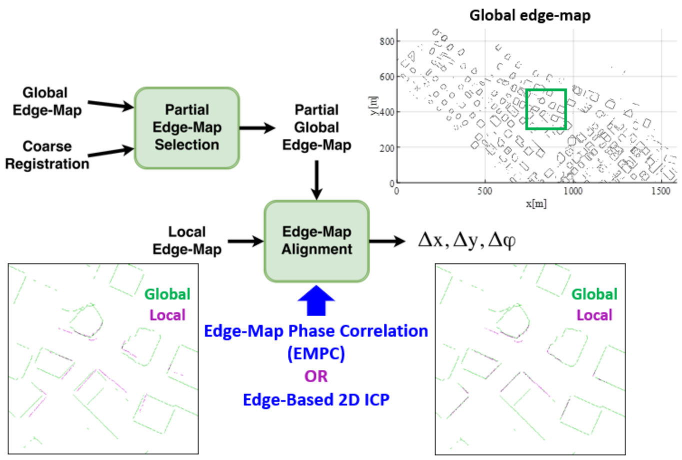


Fig. 2. Block diagram describing the framework used for local-to-global edge-map alignment.

We note that in general, rigid registration between two 3D point clouds requires finding a transformation with 6 DoF. By aligning 2D edge-maps we may find only 3 DoF (two for translation - Δx and Δy, and one for rotation - $\Delta\phi$). The 3 missing DoF are Δz, $\Delta\theta$ (Pitch, rotation around the y-axis), and $\Delta\psi$ (Roll, rotation around the x-axis). The two missing rotation angles are accounted for by rotating the clouds (both local and global, if necessary), such that the $z-axis$ is aligned with the negative direction of the gravity vector, whose estimation is described in subsection II-B. Since in this section we use an initial registration found by the viewpoint-dictionary-based method presented in [1], Δz is assumed to be known because the grid viewpoints are set at a distance from the ground that corresponds to distance of the local LiDAR sensor from the ground. The assumption that Δz is known typically holds when the ground slope angle is relatively low (e.g., lower than 10^o). In scenes were the ground slope may be larger, Δz may be estimated, after solving all the other DoF, by matching the ground surface of the local cloud to the ground surface of the relevant part of the global cloud.

We note that the algorithmic steps described in the following subsections (II-A - II-D) can be used in the same way either on a local cloud or on a global cloud.

A. Normal vector estimation

This step involves computing a normal vector $\mathbf{n}_i$ for each point $\mathbf{p}_i$ in the point cloud, where $i \in \{1, 2, ..., N_{pts}\}$ and N_{pts} is the number of points in the cloud. Normal vector estimation was done based on the work of Hoppe et al. [4], using PCA on a neighborhood ($Nhood_k(\mathbf{p}_i)$) of the k-nearest-neighbors of each point $\mathbf{p}_i$ (e.g., $k = 30$). For each such neighborhood the covariance matrix C_i is computed and its eigenvector that correspond to the smallest eigenvalue is selected as the normal vector $\mathbf{n}_i$. As discussed in section II-B, the sign of the normal vector is not important in our proposed method, so there is no need to make sure normal vectors on the same "surface" have the same orientation.

B. Gravity vector estimation

The purpose of this algorithmic step is the estimation of the gravity vector from a given 3D point cloud, acquired in an urban scene. The point cloud can then be transformed to a new reference frame (by rotation, changing the pitch and roll angles), such that the new z-axis is aligned with a direction that is negative to the estimated gravity vector. This step can be used in the same way on either a local cloud or a global cloud, but is described here for a local cloud.

The estimation is based on the following assumptions:

- The angle between the positive z-axis unit vector ($\hat{\mathbf{z}}$) of the original reference frame, in which the point cloud is given, and the negative gravity unit vector ($-\hat{\mathbf{g}}$) is smaller than θ_{max} (e.g., 20^o). We assume this holds true in most urban environments, since the local sensor is mounted on a vehicle and the road slope angle at which the vehicle can be found is typically lower than 19^o (see [5]).
- Each local cloud contains points sampled on at least two walls that are non-parallel and their normal vectors are perpendicular to the direction of gravity.

The first step involves the selection of all the points in the cloud $\mathbf{p}_i$ (and their corresponding normal vectors $\mathbf{n}_i$), that may belong to walls, according to the following criteria:

- The angle between the normal vector $\mathbf{n}_i$ and the z-axis is about 90^o, up to a difference of θ_{max}, that is: $|\sphericalangle(\mathbf{n}_i, \hat{\mathbf{z}}) - 90^o| < \theta_{max}$.
- The *surface variation* of the neighborhood of the point $\mathbf{p}_i$, defined in [6] as $\frac{\lambda_{i,3}}{\lambda_{i,1}+\lambda_{i,2}+\lambda_{i,3}}$ (using the eigenvalues of C_i where $\lambda_{i,3} \leq \lambda_{i,2} \leq \lambda_{i,1}$), is lower than a threshold value τ (e.g., 0.05), and hence is likely to belong to a mostly flat or slightly curved surface.

Next, the selected normal vectors are used to create an Extended Gaussian Image (EGI) [7], in order to detect dominant normal vector orientations of walls in the cloud. For that purpose, the normal vectors are mapped to the unit sphere, and their orientations are discretized, using a grid with pixel size $\beta \times \beta$ (e.g., $\beta = 0.5^o$), defined over the azimuth-elevation angular space (azimuth: $\phi \in [0^o, 180^o]$, elevation: $\theta \in [-\theta_{max}, \theta_{max}]$). The azimuth angle is limited to a 180^o domain (instead of 360^o as in [7]), by multiplying by -1 each normal vector whose azimuth angle is outside that domain. The normal vectors are accumulated into the angular grid according to their orientations such that a pixel with a high value in the EGI corresponds to a dominant normal vector orientation in the scene. In order to detect the dominant normal vector orientations in the scene, M largest local maxima in the EGI are selected (e.g., $M = 10$), where a circular neighborhood with radius $r = 2\beta$ is used to detect the maxima. For each of the selected maxima, an equivalent normal vector $\bar{\mathbf{n}}_j$ is computed as its mean normal vector, where $j \in \{1, 2, ..., M\}$.

In order to find the estimated gravity vector ($\hat{\mathbf{g}}_{est}$) in the scene, a RANSAC-like algorithm is used, where in each iteration a pair of local maxima are selected and used to estimate $\hat{\mathbf{g}}_{est}$. In contrast to RANSAC, the pair of maxima is not selected randomly but in a deterministic manner, such that all possible pairs are tested. This is feasible due to the relatively low number of dominant wall orientations in the scene. For example, if $M = 10$ than the total number of unique pairs of maxima is $\binom{M}{2} = \frac{M!}{2!(M-2)!} = 45$.

In each iteration the following operations are performed:

1) A pair of local maxima is selected. The two maxima are associated with two mean normal vectors: $\bar{\mathbf{n}}_{j_1}$ and $\bar{\mathbf{n}}_{j_2}$.
2) An estimated gravity vector is computed, using a cross-product: $\hat{\mathbf{g}}_{est} = \bar{\mathbf{n}}_{j_1} \times \bar{\mathbf{n}}_{j_2}$.
3) The number of inliers is counted, where an inlier is defined as a normal vector $\mathbf{n}_i$ where $|\sphericalangle(\mathbf{n}_i, \hat{\mathbf{g}}_{est}) - 90^o| < 2\beta$.

Finally, the selected estimated gravity vector $\hat{\mathbf{g}}_{est}$ is the one with the most inliers, and the cloud is rotated such that the z-axis of the new reference frame is aligned with $-\hat{\mathbf{g}}_{est}$. An example of a gravity vector estimation result for a TLS local cloud is shown in Fig. 3.

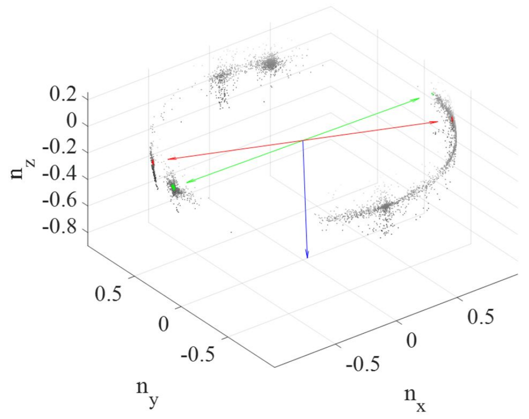


Fig. 3. Example of a gravity estimation result ($\hat{\mathbf{g}}_{est}$ - arrow pointing down, shown in blue) for a TLS local cloud. Normal vectors possibly belonging to walls are mapped to points on the unit sphere. The red and green lines represent two of the most dominant normal vector orientations in the scene.

C. Wall detection

After the local cloud's reference frame has been corrected based on the estimated gravity vector, wall detection takes place, based on the possible-wall-points found in subsection II-B. Out of these, only the points whose normal vector $\mathbf{n}$ is perpendicular to the estimated gravity direction up to an angle of 2β (where $|\sphericalangle(\mathbf{n}, \hat{\mathbf{g}}_{est}) - 90^o| < 2\beta$), are considered to belong to a wall.

D. 2D projection and edge-map creation

In this step, the previously detected wall-points are first rotated (around the z-axis), according to a coarse registration, found in our case using the viewpoint-dictionary-based method from [1]. They are then projected onto the x, y-plane, in order to create a binary *edge-map*, where "1" represents the presence of a wall and "0" represents its absence. A square pixel grid is defined over the x, y-plane where the size of each pixel is $d \times d$ (e.g., $d = 1/3m$). The initial occupancy map is created by setting the value "1" in each pixel that contains at least one projected wall-point. In order to improve edge continuity, a dilation operation is applied to the occupancy map, using the MATLAB function "imdilate" [8], with a disc structuring element with a radius of $3d$. Next, in order to avoid "thick" edges that may later result in decreased registration refinement accuracy, a thinning operation is used, based on MATLAB's function "bwmorph" [9]. Finally, connected components whose "length" is shorter than $2m$ are removed, using 8-adjacency. The "length" of a connected component is defined as the major axis of the ellipse that has the same normalized second central moments as the connected component.

E. Edge-map alignment

As described in Fig. 2, given a local edge-map and the coarse registration of its corresponding local cloud to the global cloud, only the relevant part of the global edge-map is used for alignment. This part of the global edge-map is of the same size as the local edge-map, and its center is set as the x, y coordinates of the local sensor according to the coarse registration. Next, given the local edge-map and the relevant part of the global edge-map we seek to find the translations $\Delta x, \Delta y$ and the rotation angle $\Delta\phi$, that correspond to the alignment of the two edge-maps. Two different approaches for edge-map alignment were considered: Edge-Map Phase-Correlation (EMPC), and Edge-based 2D ICP. Both methods are briefly described here and their registration refinement results are compared in section V to each other, as well as to the widely used 3D ICP refinement method [2].

1) Edge-map alignment using Phase-Correlation: This method for image alignment is based on the work of Reddy and Chatterji [10], regarding image registration using phase-correlation. In general, this method allows finding the translation, rotation and scale factor between two images, but in our case both edge-maps are given in the same scale (in both edge-maps, the "physical" pixel size is $d \times d$), so we aim to find only the translation and rotation between them. Fig. 4 shows a block diagram of the process for aligning two edge-maps based on two phase-correlation steps.

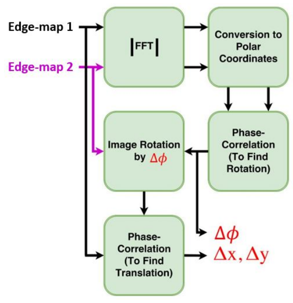

Edge-map 1
Edge-map 2
|FFT|
Conversion to Polar Coordinates
Image Rotation by $\Delta\phi$
Phase-Correlation (To Find Rotation)
Phase-Correlation (To Find Translation)
$\Delta\phi$
$\Delta x, \Delta y$

Fig. 4. Block diagram of the Edge-Map Phase-Correlation method (EMPC), used for aligning two edge-maps.

2) Edge-map alignment using 2D ICP: Another approach we considered for edge-map alignment is based on converting the edge-maps into 2D point clouds (or ”edge-clouds”), and finding a registration between them using a 2D version of the well-known ICP algorithm [2]. Conversion of an edge-map into a 2D point cloud is done by placing a point at the center of each edge-map pixel whose value is ”1”.

Then, a standard 2D version of ICP [2] is used with two modifications whose purpose is avoiding outlier pairs of corresponding points, since outlier detection was not explicitly handled in [2]. Other outlier removal methods, such as described in [11], are possible as well.

Firstly, on the initial iteration, nearest-neighbor point-pairs are considered inliers if the distance between them, D, is smaller than a threshold value that depends on the distance r between the local edge-cloud point and the origin (its distance from the local sensor). For inlier pairs: $D < 2d_{grid} + 2r\sin(\alpha/2)$, where d_{grid} (e.g., $3m$) and α (e.g., 4^o) are the viewpoint grid distance and panoramic range-image angular resolution, used in the viewpoint-dictionary-based registration method from [1]. This upper bound on D is a result of the triangle inequality, under the following assumptions: (a) the distance between the ground truth local sensor location and the selected grid viewpoint is smaller than $2d_{grid}$, (b) the coarse registration azimuth error is smaller than α. The purpose of this bound is to take into consideration that a rotation error in the coarse registration causes larger location errors for points that are far from the local sensor, and thus not filtering out useful point-pairs in the first ICP iteration.

Secondly, on the other iterations, inlier point pairs are those where: $D < D_{max}$ (e.g., $D_{max} = 2.5m$).

III. Experimental Setup

The global cloud used to test the proposed registration refinement methods is an ALS cloud, acquired in an urban environment (Vancouver, Canada), that covers an area of approximately $1km^2$. A set of 94 local clouds (TLS) is also used, such that each cloud fulfills our assumptions from section II-B: the local road slope is lower than $\theta_{max} = 20^o$, and each local cloud contains points sampled on at least two non-parallel walls, whose normal vectors are perpendicular to the gravity vector. Fig. 5 shows the global cloud and the ground truth local sensor locations.

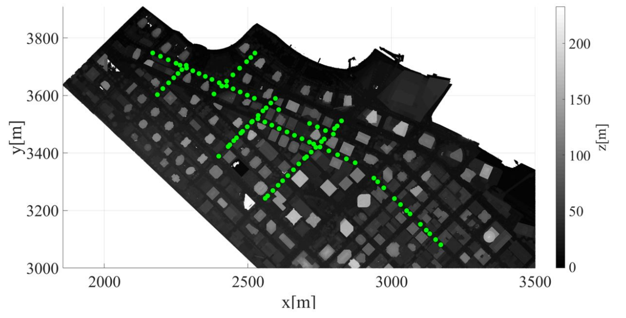


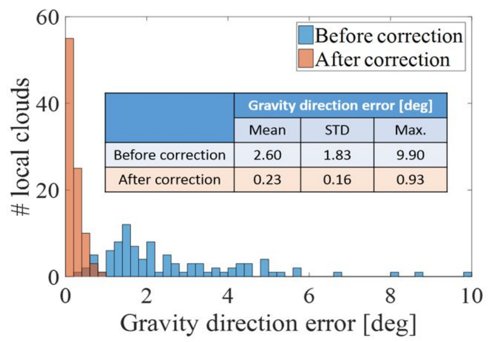
Fig. 5. A top view of the global cloud used for testing the proposed registration refinement methods. The ground truth local sensor locations of the 94 local clouds are shown in green.

The overall registration process uses the viewpoint-dictionary-based method [1] for finding coarse registrations between the local clouds and the global cloud (the same set of parameters is used as in [1]). Then, registration refinement is carried out in three separate ways: 3D ICP (for comparison), Edge-Map Phase-Correlation (EMPC), and Edge-Based 2D ICP.

IV. Gravity Estimation Results

In this subsection, gravity vector estimation results, achieved using the method described in section II-B, are presented. The gravity direction error e_g is defined as the angle between the estimated gravity vector $\hat{\mathbf{g}}_{est}$ and the ground truth gravity vector $\hat{\mathbf{g}}_{gt}$: $e_g = \arccos(\hat{\mathbf{g}}_{est} \cdot \hat{\mathbf{g}}_{gt})$.

Fig. 6 shows a comparison of the gravity direction errors before and after the correction described in section II-B. Before correction, $\hat{\mathbf{g}}_{est}$ is assumed to be the same as the negative direction of the z-axis in each of the 94 local cloud. As can be seen in Fig. 6, the proposed gravity vector estimation and correction reduces the errors by approximately an order of magnitude (mean, STD, and max. error). For example, the mean gravity direction error is reduced from 2.60^o (before correction) to 0.23^o (after correction).



	Gravity direction error [deg]		
	Mean	STD	Max.
Before correction	2.60	1.83	9.90
After correction	0.23	0.16	0.93

Fig. 6. Comparison of gravity direction errors before (blue) and after (orange) gravity vector estimation and correction. The errors were computed over 94 TLS clouds.

V. Registration Refinement Results

A comparison of the registration results achieved by the three methods (3D ICP, EMPC, and Edge-Based 2D ICP),

TABLE I
COMPARISON OF REGISTRATION RESULTS OF 3D ICP WITH THE PROPOSED METHODS: EDGE-MAP PHASE-CORRELATION (EMPC), AND EDGE-BASED 2D ICP.

Registration Refinement Method	**Localization Error [m]**			**RRE [deg]**			**Runtime [sec]**		
	Mean	STD	Max	Mean	STD	Max	Mean	STD	Max
3D ICP [2]	**0.40**	**0.20**	1.13	0.67	**0.30**	1.73	1.62	0.35	2.71
Edge-Map Phase-Correlation (EMPC)	0.57	0.45	3.21	0.98	0.76	5.65	0.46	**0.02**	**0.54**
Edge-Based 2D ICP	**0.40**	0.21	**1.09**	**0.60**	0.38	**1.61**	**0.44**	0.08	0.83

over the 94 TLS clouds, is shown in Table I (error statistics and runtime comparison). Registration accuracy is measured in terms of the local sensor's localization error and RRE (Relative Rotation Error), which is the sum of absolute Euler angles of the error rotation matrix [12]. As can be seen in Table I, 3D ICP and the Edge-Based 2D ICP method achieve comparable localization error (mean error is $0.40m$ for both methods), while in terms of RRE, the Edge-Based 2D ICP method achieves slightly better results (mean RRE of 0.60^o instead of 0.67^o). The EMPC method achieves less accurate registration results, especially due to exceptional cases, where the local edge-maps have a low number of edges (small number of walls in the scanned scene).

The runtime of all the compared registration refinement methods was measured for all of the 94 local clouds (MATLAB implementations, run on PC with i7-5820K CPU @ 3.30GHz). For the proposed edge-based methods (EMPC and 2D ICP), the runtime measurements include the local cloud normal vector estimation, gravity estimation and correction, wall detection, edge-map creation, and finally, edge-map alignment. As can be seen in Table I, the Edge-Based 2D ICP method achieves a meaningful improvement (by a factor of 3.7) in runtime over the widely used 3D ICP method, without compromising registration accuracy. The mean runtime of 3D ICP was $1.62sec$ (STD=$0.35sec$), while the mean runtime of the Edge-Based 2D ICP method was $0.44sec$ (STD=$0.08sec$). This demonstrates the advantage of the proposed Edge-Based 2D ICP method over 3D ICP for registration refinement of 3D point clouds acquired in a typical urban environment, under the assumptions from section II-B.

VI. CONCLUSION

In this work, we proposed a novel approach for 3D point cloud registration refinement in an urban environment, based on 2D edge-map alignment. In addition, a gravity direction estimation algorithm for 3D urban point clouds was presented. It was shown, using a challenging dataset of TLS and ALS clouds, that the proposed registration refinement method achieved a meaningful reduction in runtime in comparison to the widely-used (3D) ICP algorithm, while maintaining comparable registration accuracy (see section V). Other than 3D urban modeling and autonomous vehicle localization, we believe the proposed methods could prove useful also in indoors scenarios, such as robot localization and mapping.

ACKNOWLEDGMENTS

This work was supported by the Omek consortium, which is a program of the Israel Innovation Authority. We thank Viktor Shenkar, Yigal Eilam and Ramon Axelrod, at GeoSim, for helpful discussions and for providing us with the point cloud data used in this paper. We would also like to acknowledge the support of Erez Nur, technical manager of Omek. The support of Nimrod Peleg and Yair Moshe of the Signal and Image Processing Lab (SIPL), at the Technion, is greatly appreciated.

REFERENCES

[1] D. Avidar, D. Malah, and M. Barzohar, "Local-to-global point cloud registration using a dictionary of viewpoint descriptors," The IEEE International Conference on Computer Vision (ICCV), pp. 899-891, IEEE, 2017.
[2] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," IEEE Transactions on pattern analysis and machine intelligence, Vol. 14, No. 2, pp. 239–256, 1992.
[3] D. Avidar, "Local-to-Global 3D Point Cloud Registration using a Viewpoint Dictionary," MSc Thesis, Technion, 2018.
[4] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," SIGGRAPH Conference Proceedings, Vol. 26, No. 2, 1992.
[5] Baldwin Street (Wikipedia webpage), URL: https://en.wikipedia.org/wiki/Baldwin_Street, Accessed: Apr. 10, 2018.
[6] M. Pauly, M. Gross, and L. P. Kobbelt, "Efficient simplification of point-sampled surfaces," Proceedings of the conference on Visualization of the IEEE Computer Society, pp. 163–170, 2002.
[7] B. K. P. Horn, "Extended Gaussian images," Proceedings of the IEEE, Vol. 72, No. 12, pp. 1671–1686, 1984.
[8] R. Van Den Boomgaard and R. Van Balen, "Methods for fast morphological image transforms using bitmapped binary images," CVGIP: Graphical Models and Image Processing, Vol. 54, No. 3, pp. 252–258, Elsevier, 1992.
[9] L. Lam, S. W. Lee, and C. Y. Suen, "Thinning methodologies-a comprehensive survey," IEEE Transactions on pattern analysis and machine intelligence, Vol. 14, No. 9, pp. 869–885, IEEE Computer Society, 1992.
[10] B. S. Reddy and B. N. Chatterji, "An FFT-based technique for translation, rotation, and scale-invariant image registration," IEEE transactions on image processing, Vol. 5, No. 8, pp. 1266–1271, IEEE, 1996.
[11] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," Proceedings of the Third International Conference on 3-D Digital Imaging and Modeling, pp. 145–152, IEEE, 2001.
[12] Y. Ma, Y. Guo, J. Zhao, M. Lu, J. Zhang, and J. Wan, "Fast and Accurate Registration of Structured Point Clouds with Small Overlaps," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 1–9, 2016.