Analysis of 3D Dynamic Urban Scenes Based on LiDAR Point Cloud Sequences

Oszkár Józsa[1] and Csaba Benedek[1]

MTA SZTAKI, Budapest
jozsa.oszkar@gmail.com, bcsaba@sztaki.mta.hu

Abstract. In recent years, the spread of domestic, industrial and research robots and the robotic vehicles made 3D sensory information (such as LiDAR sensors) more and more widespread opposed to the conventional 2D image sensory information. A modern LiDAR sensor on board of a vehicle provides enough data to make us able to interpret the space surrounding the vehicle even at large scenes containing a whole street segment.

In this paper, we present a method to reconstruct. The input of this procedure is a LiDAR sensor mounted on a car, which outputs a data stream covering more than 100 meters radius of space, collecting data at 15 Hz. The recording is in real environment on the streets of Budapest in real time, while the processing is offline, implemented on CPU keeping in mind the future implementation on GPUs to reach real time data processing. The data is provided by a Velodyne HDL-64E high performance LiDAR device.

The aim is to segment several region classes (such as roads, building walls, vegetation) and to identify specified objects (such as people, vehicles, traffic signs) in the point clouds through a presegmentation step. To achieve this classification, we need several features such as the color and geometrical properties of the specified objects and their possible geometrical and physical interactions. Also, we need to take into account the time domain features calculated based on the LiDAR data stream. After this presegmentation step we are able to reconstruct building facades in 3D and to track the detected objects in the 3D space. Also, in the future, this processed data set can be registered against 2D images provided by conventional cameras to reproduce realistic, colored 3D virtual spaces.

1 Introduction

Analysis of 3D spaces comes from the need to understand the environment surrounding us and to build more and more precise virtual representations of that space. Remote sensing is a widely researched field for many decades and so is scene interpretation. But in the last decade, as the three dimensional (3D) sensors begun to spread and the commercially available computing capacity has grown big enough to be sufficient for large scale 3D data processing, new methods and applications were born. In many real world problems, 3D sensing and scene interpretation started to take the place of the conventional 2D image processing based on two dimensional imagery data.

In recent years, more and more technologies started to appear that heavily rely on these new 3D methods. Robotic cars, ships and airplanes are successfully tested and in some cases are already used on everyday bases. As Luettel et al.[1] state in their summatory article, automation of our vehicles are the "path to the future".

But not only the automation industry depends heavily on 3D mapping and scene interpretation. There are applications of this technology in industrial manufacturing, geology and mapping, architecture, disaster and crime prevention and control, space exploration, military intelligence and virtual gaming. All these fields have their own special tasks, where 3D data can help both the professionals and the users.

Light Detection and Ranging (LiDAR) devices are common tools for three dimensional mapping as they provide accurate 3D data directly from the device. These devices provide us the so called *point clouds.* A point cloud is a data set with small units of data, each representing a 3D point in the space. A point essentially has at least three information: its 3 coordinates, x, y and z (which can be represented in polar coordinate system or Euler angles or *lattitude*, *longitude*, *altitude* in geographic data sets, etc). Additionally color, intensity information could also be provided by some devices.

These instruments can quickly produce huge amount of data which makes data processing a hard task. Our aim is to develop efficient, fast methods that are able to filter the significant data out of these streams (to reduce the need of computing power to post-process all this large amount of data). Once we are able to select the information useful to us, the next step is to merge this data from our whole scan sequence to even larger and more useful 3D data sets. Matching subsequent scans to each other is called *point cloud registration*, which can provide us even more useful data sets; for example registered urban point clouds can be processed and analyzed similarly as aerial LiDAR scans but at much higher resolution since the sensor is able to scan the scene from a much smaller distance.

The analysis of 3D dynamic urban scenes can be the first step for either real time scene interpretation tasks (essential for traffic monitoring and robotic cars), or for post processing three dimensional data (building realistic virtual models of real cities). In this paper, we present a novel method to preprocess this real-life 3D data to help both registration and recognition tasks.

2 The Velodyne Sensor

The data processed in this paper is provided by the *Velodyne HDL-64E* high performance LiDAR sensor.[1] The typical usage of this sensor is to mount it on a moving platform, practically onto a car. The LiDAR has fast enough scanning

[1] For full product description, visit the official product website http://www.velodyneLiDAR.com/LiDAR/hdlproducts/hdl64e.aspx

speed to scan the whole 3D space at about every 30-60 centimeters of distance as the car travels with typical urban traffic speed. With each revolution, a full 3D scan is made, and overall the device outputs more than 1.3 million 3D points per second.

Alternatively, it can be used as a stationary sensor at outdoor or relatively large indoor scenes (e.g.: warehouses, manufacturing halls). Applications of this static configuration will not be discussed in this this paper, instead, we will be focusing on the more challenging moving platform.

Along the 3D (x-y-z) data, the sensor also gives a reflection intensity value for each point. This intensity value mostly depends on the objects' surface quality: shiny, flat surfaces reflect light much better than matte, scattered surfaces such as roads or vegetation. The angle of attack can also affect this intensity value i.e. if the beam hits the surface in a high angle, the light can scatter and thus the returning intensity will be lower. Although the algorithms presented in this paper do not exploit the intensity information, we will mention examples in Section 7 where it might highly support some of the proposed detection tasks.

3 Related work

In most cases, for point cloud registration in robotic environments the ICP [2] algorithm is used for matching which performs well in homogeneous 3D scans but often fails in noisy, fast-changing real world environments. In robotic applications, the main task is to detect and track moving objects[3, 4], while our current task is to understand a scene with both moving and static objects and build realistic virtual representation of that scene.

Douillard et al. have developed a method to extract "segments" (interest regions) from the point cloud [5, 6]. This approach is similar to ours: with our preprocessing step, we also try to find sets of points that will later help the registration step. Makadia et al. constructed an automatic registration method using large histograms calculated from the surface normal fields[7] but is computationally expensive and works well on homogeneous scans, not on LiDAR data.

Others have taken a different approach. Behley et al. [8] constructed a multiscale 3D histogram descriptor that can be efficiently used for point matching. Quadros et al. presented a feature called *the line image* to support point matching that outperforms the widely used NARF descriptor but requires a computationally expensive principal component analysis (PCA) calculation[9].

After the registration is done, next step is virtual reconstruction. Since the scans are noisy by nature and possibly have holes in them (e.g.: occlusion caused by trees and parking cars standing in front of buildings, etc.), reconstruction is not trivial. Recently, Shen et al. showed an efficient method for facade reconstruction which uses an iterative adaptive partitioning and is robust against the aforementioned issues [10]. Also, Wang et al. showed a method to upsample sparse LiDAR data and to clean the points corresponding to building interiors (since the laser beam travels through windows also, the LiDAR also scans the interiors of buildings)[11].

Also, for already registered, large data sets, there are effective detection, segmentation and recognition algorithms available. Velizhev et al. developed a highly effective algorithm to interpret large outdoor urban scenes[12]. Their method is proposed for mobile robot mapping; though, once we register LiDAR data with the method presented in this paper, our data sets will be similar to theirs.

Han et al. presented a method to detect road boundaries and road segments based on LiDAR data and vehicle movement information [13]. Robotic cars such as Stanley and Junior built for the DARPA Grand Challenges[14] [15], the AnnieWAY self driving car [16] and Carnegie Mellon's Car, Boss[17] for example. use a huge amount of sensors to map their environment and interpret the scene surrounding them.

In many of the cases above, the registration is done by the help of external sensors such as Inertial Measurement Units (IMU), Global Positioning System (GPS) or other sensors on the car (e.g. speedometers, wheel rotation sensors, radars). The novelty of our method is that it does not need any supplementary sensory information, nor any learning algorithms or huge sample database.

4 The proposed method

As mentioned before, point cloud segmentation is done by a grid based approach. First, we fit a regular 2D grid S with W_S rectangle width onto the $P_{z=0}$ plane, where s denotes a single cell. We used a W_S value between 50cm and 80cm. Smaller grid size is not viable due to the resolution; smaller cells would not have enough points in them to calculate good enough statistical information. On the other hand, larger cell size can result in larger number of falsely classified points, since within a large cell, multiple objects can occur. Near-the-center grid cells within the above specified range have hundreds of points in them (grid cells further away obviously have less and less points).

Then, we assign to each $p \in \mathcal{P}$ point of the point cloud to the corresponding cell s_p, which contains the projection of p to $P_{z=0}$. Let us denote by $\mathcal{P}_s = \{p \in \mathcal{P} : s = s_p\}$ the point set projected to cell s. $y_{\max}(s)$, $y_{\min}(s)$ and $\hat{y}(s)$ are the maximum, minimum and average of the elevation values within $\mathcal{P}_s$.

The first step of the process is the segmentation of the cell-map, i.e. we assign to each cell (and to each point in that cell) $s \in S$ an ω_s class label from the finite label set: $\mathcal{L} = \{l_{\text{to}}, l_{\text{so}}, l_{\text{gr}}, l_{\text{sp}}\}$, corresponding to the classes (i) *tall structure object*, (ii) *short street object*, (iii) *ground* and (iv) *sparse region*.

4.1 Classes

At first, sparse regions (l_{sp}) are detected. These encapsulate only a few or no points: we can obtain only very limited, and probably misleading information from these cells regarding the scene structure and objects, therefore we will neglect these regions in the later model phases. The threshold of a cell s being considered as a sparse cell is typically 4-8 points - any cell containing less points than this threshold is considered sparse (the exact value of this threshold also depends on the sensor's revolving speed, slower speeds allow to have larger threshold). Sparse data is classified first, so in later processing we will know not to check these grid cells, thus save some processing time.

A cell should belong to the l_{to} class if it contains a tall object of the street structure, such us building walls, lamp post or tree trunk. These object can be considered static in the scene, since these 3D points obviously belong to objects that are not moving and are large enough to be present in several consecutive scans. These points will be used later for point cloud registration of the consecutive time frames. The two criteria for a grid cell to belong in this class are:

$$y_{\max}(s) < 140 \ \vee \ y_{\max}(s) - y_{\min}(s) > 310 \tag{1}$$

So either the maximal height is larger than 140 centimeters, or the height difference is larger than 310 centimeters. The two criteria are similar. The first one says that if there are points in the grid cell which are higher than the sensor height plus 120 centimeters, then the cell should be considered as a tall object (the sensor has a height about 2 meters). The second criteria says that if there is a height difference of more than 310 centimeters within the cell, it should be considered as a wall. The second criteria is needed for dealing objects standing on a lower point of the ground compared where the car travels (e.g.: elevation differences, walls seen from an overpass road, etc).

Next, the ground cells (l_{gr}) are identified. These cells contain only a surface which is not covered by any objects, but contains a considerable number of ground points. The criteria for the grid cell to belong into this class is:

$$y_{\max}(s) - y_{\min}(s) < 25 \ \wedge y_{\max}(s) < -50 \tag{2}$$

In words: if the points within a cell are "flat" enough, and the maximal height is below -100 centimeters, the grid cell is considered as road surface. The first criteria ensures the flatness or homogeneity of the points. Given a cell with 60 centimeters if width, this allows 22.6 ° of elevation within a cell; higher elevations are highly unrealistic in an urban scene. The second criteria ensures that this patch of flat surface is under the car; again, the sensor is at about 200 centimeters of height so the road surface directly beneath the car has a height of about -200 centimeters. Less than -50 criteria is used to deal with elevation differences that can occur within the field of view of the sensor.

The rest of the point cloud is assigned to class l_{so}. These points belong to short objects like vehicles, pedestrians, short road signs, line posts etc. These entities can be either dynamic or static, which attribute can only be determined later, after further, more complex investigation of the point cloud sequence.

4.2 Terrain model

For point cloud classification, we first derive a terrain model, which approximates the ground elevation level at each cell of S. Planar ground models are frequently applied in urban scenarios, relying on robust plane estimation methods such as RANSAC. However, we experienced that in the considered urban scenes the ground cannot be considered planar, as there are significant elevation differences (often up to 1-2m) between the opposite sides of the observed roads and squares. In these cases, planar ground estimation yields significant errors in the extracted object shapes, e.g. bottom parts can be cut off, or the objects may drift over the ground.

As a first evidence, we can notice that in the ground cells the differences of the observed elevation values are low. Therefore we can perform an initial classification, where each cell s is classified either as ground candidate ($\mathbf{1}_G(s) = 1$) or as undefined region ($\mathbf{1}_G(s) = 0$) by a straightforward thresholding:

$$\mathbf{1}_G(s) = 1 \text{ iff } \big(y_{\max}(s) - y_{\min}(s) < \tau_{\text{gr}}\big),$$

where we used τ_{gr} =30cm. This preliminary map can only be considered as a coarse estimation of the ground, since, among others, some cells of flat car roof or engine hood regions may be erroneously classified as ground. However, these outlier cells can be efficiently eliminated by spatial filtering. With denoting by N_s^ν the $\nu \times \nu$ neighborhood of s, and $\gamma_G(s) = \sum_{r \in N_s^\nu} \mathbf{1}_G(s)$, we can obtain a terrain model of scene:

$$y_{\text{gr}}(s) = \begin{cases} \frac{1}{\gamma_G(s)} \cdot \sum_{r \in N_s^\nu} \hat{y}(s) \cdot \mathbf{1}_G(s) & \text{if } \ \gamma_G(s) > 0 \\ \text{undefined} & \text{otherwise.} \end{cases}$$

where $y_{\text{gr}}(s)$ is the estimated ground-elevation value at cell s. The $y_{\text{gr}}(s)$ feature can be efficiently calculated by deriving the integral images of the $\hat{y}(.)$ and $\mathbf{1}_G(.)$ maps. We used here a large neighborhood ($\nu = 17$ for cell maps with a size around 400×300).

5 Point cloud registration

A single scan has a large amount of points but since it covers a large area, the resolution is sufficiently good only within few meters of distance. Though the device has a sensing distance of more than 100 meters, the measurements at more than 15 meters of distance are too sparse for many detection or reconstruction algorithms. With our proposed method, registration of the subsequent point clouds will be more effective and successful. The registration will help us to fill in the holes due to occlusion and we can achieve very high point density after registering several point clouds.

Although various established techniques do exist for point cloud registration, such as Iterative Closest Point (ICP) [2] and Normal Distribution Transform (NDT) [18], these methods fail, if we try to apply them for the raw Velodyne

LIDAR point clouds for two reasons: 1) All moving points appear as outliers for the matching process, and since in a crowded street scene we expect several moving objects, many frames are erroneously aligned. 2) Due to the strongly inhomogeneous density of the LIDAR clouds, even the static ground points mislead the registration process. The above algorithms often match the concentric circles of the ground (See Fig. 5.1), which yields that the best match erroneously corresponds to a near zero displacement between two consecutive frames. However, we have also observed that the point density is quite uniform in local wall regions which are perpendicular to the ground.

Our key idea is to utilize the point classification result from the previous section to support the registration process. We only use as input of the registration algorithm the points segmented as high objects, since we expect that in majority, these points correspond to stationary objects (such as buildings), thus they provide stable features for registration. Details and test results are in the following sections.

6 Normal Distributions Transform

This method was introduced in Martin Magnusson's doctoral thesis, *The Three-Dimensional Normal-Distributions Transform an Efficient Representation for Registration, Surface Analysis, and Loop Detection* [18] which is a 3D implementation of the 2D Normal Distributions Transform developed by Biber and Straer.[19]

The Normal Distributions Transform itself uses a grid based approach also. First, it divides the space into cubes. For each cube, calculates its so-called local probability density function (PDF) to describe that cube: "Each PDF can be seen as an approximation of the local surface, describing the position of the surface as well as its orientation and smoothness."[18]

For the registration step, it uses Newton's optimization method to find the rotation and translation between the two point clouds, searching for the best match between the PDFs of the two scans. This method is robust to outliers.

For a given scan, we register it to the previous scan with NDT which gives us the transformation matrix that moves the given scan to the previous scan's frame. Once this is done to all of the scans, we are able to transform all the scans into a mutual reference frame. For the running time of this process, please refer to Table 1 in Chapter 8.

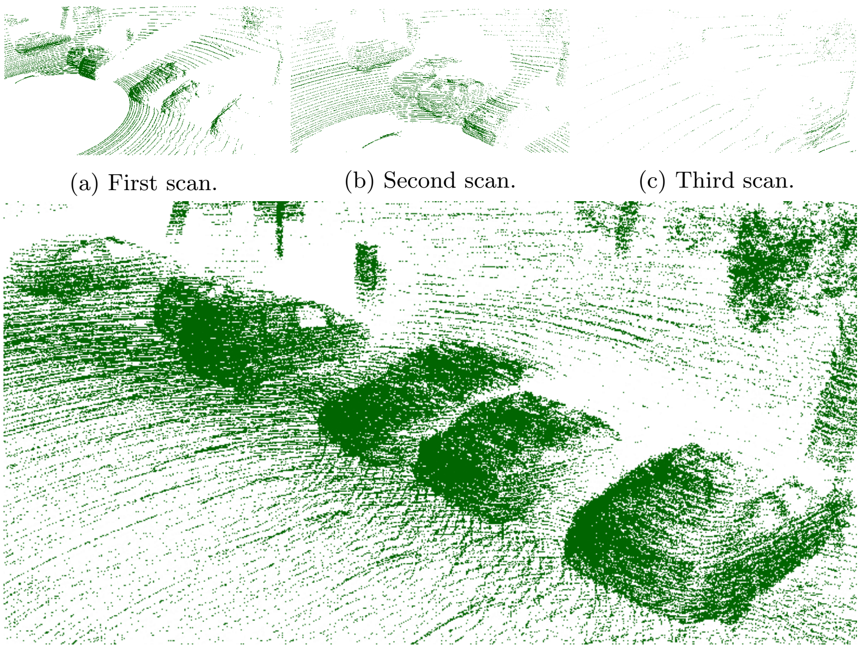
7 Results on the registered data

7.1 Street reconstruction

As expected, the presegmentation helps the point cloud registration. The registration is accurate and can be easily performed on the static points. See Fig 1. for the visualization of the registration: as the car passes by the scene becomes more occluded and sparse, on the final, registered point cloud these affects are taken out and the resolution is significantly higher, too (more results can be seen at the end of the paper).

Also, the big advantage of the road surface detection method over a RANSAC-based plane matching is that it can deal with curvatures, elevation changes in the scene. RANSAC can find the best fitting plane for the road surface but in real world, the road surfaces are not planars, especially in cities built on or next to a hill. Using the proposed method, such difficulties can be overcome and once the surface grid cells are detected, even an elevation map can be built.

One other benefit of the grid based segmentation is that it also removes noise. Grids containing too few points are not useful, they make computation harder. The segmentation filters these points out also.

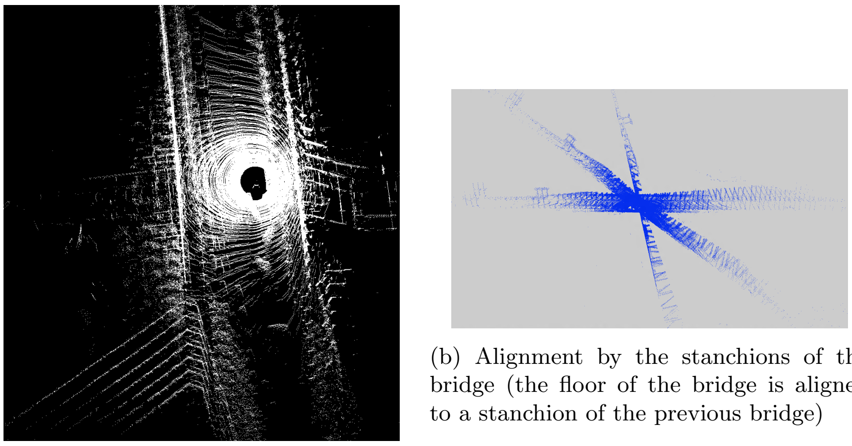

(a) First scan. (b) Second scan. (c) Third scan.

(d) The resulting registered scan (made out of 20 scans, not only the 3 above).

Fig. 1: An image sequence showing the registration process.

7.2 Facade approximation with point clouds and triangle meshes

After registering several point clouds against each other, the resolution can be sufficiently high and uniform to create realistic building facade reconstruction. As the car passes by a building, it collects data from several point of view so most of the holes on the walls due to occlusion can be filled in. Also, after



(a) Alignment by the whole in the middle of the scans

(b) Alignment by the stanchions of the bridge (the floor of the bridge is aligned to a stanchion of the previous bridge)

Fig. 2: Examples of false registration

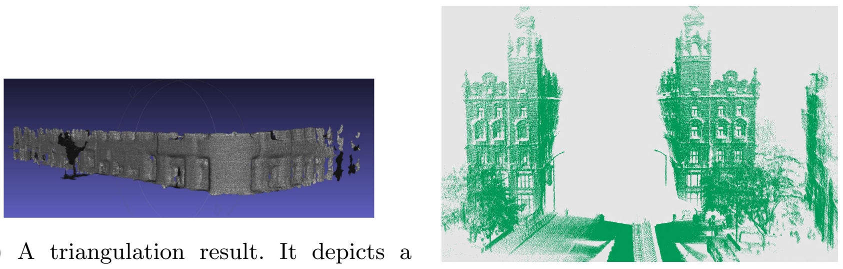
concatenating a few dozen scans, the resolution of the data will be significantly higher which results in a precise 3D reconstruction of wall surfaces and more efficient noise reduction also.

The current workflow we developed is as follows. After the transformation matrices are calculated via the NDT algorithm, they are applied only to the point clouds that contain the static tall points. After transforming these clouds into a mutual reference frame, we will get a large point cloud, which only contains walls, lamp posts, trees ans similar tall objects. On this larger, registered pointcloud a clustering is done which separates the aforementioned objects into separate point clouds. For results, see Fig.3.

7.3 Vegetation detection

Another well defined task is vegetation (typically: trees and bushes) detection. The reason for this is two-fold: on one hand, a popular task is ”vegetation mapping” along with calculating the green area in a city. On the other hand, the removal of the detected vegetation data from the point cloud can help detection algorithms, for example in the case of trees hanging over parking cars.

We have developed a vegetation removal algorithm which calculates a statistical feature for each point based on the distance and irregularity of its neighbors, for results of this algorithm, see Fig.3. As mentioned before, the intensity channel also can serve as a strong indicator of vegetation. Once the intensity and the statistical information will be used as a multifeature descriptor, it will be a strong feature for vegetation detection.


(a) A triangulation result. It depicts a building on Moricz square.

(b) An example of the tilted configuration. It depicts the Klotild Palace, near the Elizabeth bridge in Budapest

Fig. 3: Facade approximation

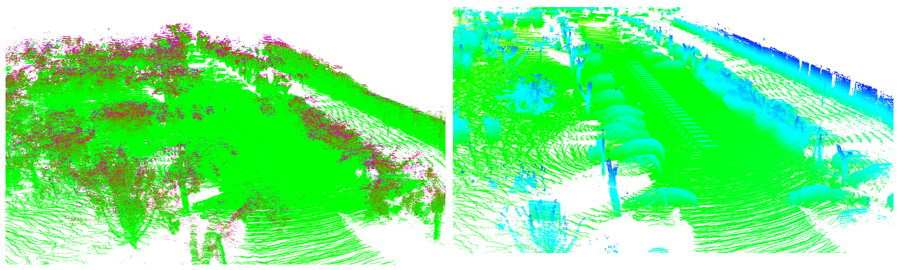
8 Conclusion and further work

The results above show the success of the simple, yet useful presegmentation step that has a great positive effect on the point cloud registration. Dividing the initial point cloud into multiple semantic classes and using only the relevant points results in a faster, more stable registration.

8.1 Processing speed

For the segmentation step, the main aim was to implement a fast algorithm that performs a presegmentation step. This aim has been met; once the data is in the memory (reading from the disk is the slowest step), the algorithm runs in real time on a modern CPU.[2]

[2] The test environment has an Intel Core i7 processor and 8 gigabytes of DDR3 RAM


(a) The detected leaf points are marked with red and purple.

(b) Point cloud after the leaf points are removed. The points are colored by their height for better visibility.

Fig. 4: An example of vegetation removal

Besides the segmentation being fast, it fastens up the registration step also. Once all the noise and interfering moving points are removed, the registration is easier, thus faster. As the registration algorithm tries to match up points, the overall error will be smaller thanks to the ”clean” point cloud and the algorithm will converge faster towards the termination criteria. Registering point clouds containing only stable, stationary points takes only 1-4 seconds (this value depends on the scanning rotation speed and on the amount of points left in the cloud after segmentation). Registering a full cloud takes 5-10 second - if it converges, since in some cases the registration fails.

Dataset name	Number of scans	Speed with presegmenatation (s)	Without presegmenatation (s)
Tas Vezr street	10	0.3209	5.3061
Tas Vezr street	10	0.2709	6.8782
Bartk Bla street	10	0.5888	6.3983
Bartk Bla street	15	0.5056	5.8474
Bartk Bla street	15	0.0868	4.0099*

Table 1: Speed comparison. * denotes failed registration

8.2 Robustness

The proposed workflow is robust enough to perform extremely well in real life environments (some experiments on this topic use only more sterile indoor or even more sterile artificial data sets).

The algorithm has been tested in lots of data sets:

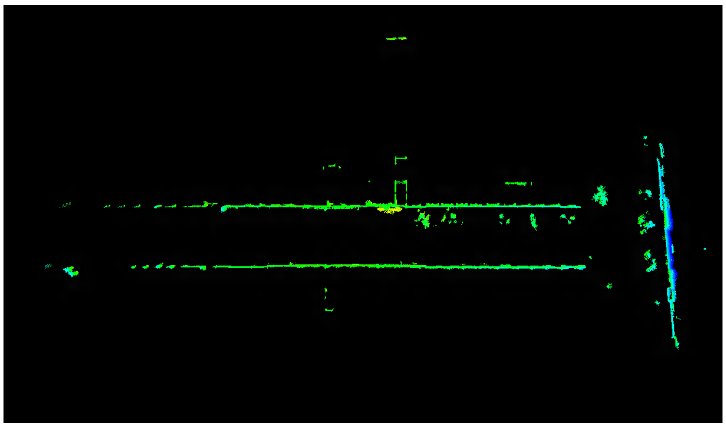
Street name	Number of scans
Kende street	496
Bertalan Lajos street	154
Bartk Bla street	581
Villnyi street	646
Tas Vezr street	345
Edmr street	24
Diszegi street	824
Dvid Ferenc street	972
Szret street	770
Somli street	945
Liberty Bridge	160
Andrssy street	125

Table 2: The data sets used to test the algorithm. Most of the data recordings had been done in the 11th District of Budapest

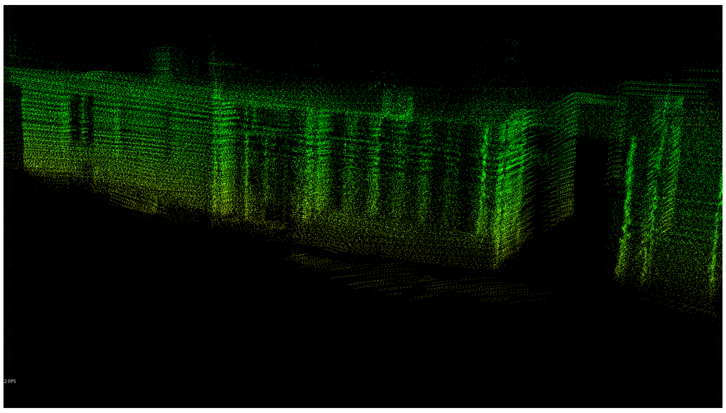
8.3 Further work

For higher level traffic monitoring, a better road interpretation is necessary. Detailed information of the road surface is required for this task; such as dividing the road class into smaller semantic units e.g.: sidewalks, lanes, parking lots, road intersections. Also detection of traffic signs, traffic lamps, crosswalks, etc.

Also, for more detailed traffic interpretation, vehicle detection is necessary. It is a highly researched area, and there are existing effective methods for aerial LiDAR data. After registering a whole street sequence, the data can be used similarly as of an aerial LiDAR data, with much higher resolution.

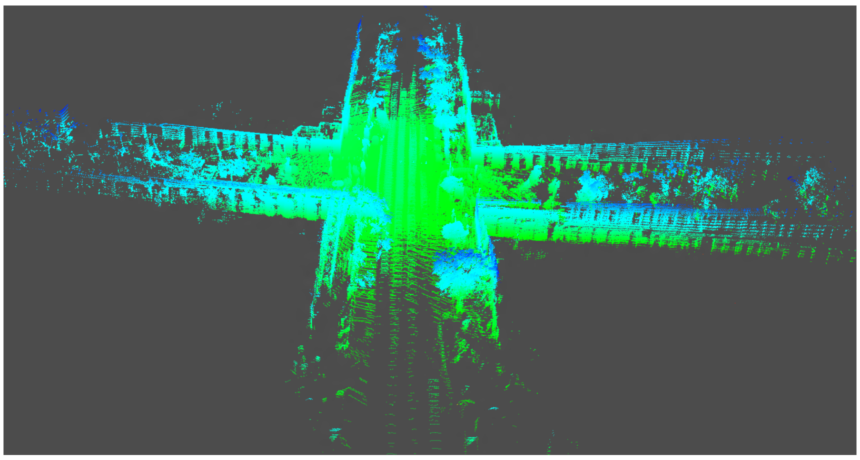


(a) Upper view of the street. Notice, how the interiors of the building gets visible as the scanner passes the door.



(b) Zoomed in picture.

Fig. 5: A large registration of Kende street showing only the building class. Scan sequence starts at the middle of the street and runs to its end. The point cloud is made out of 230 point clouds, the full cloud contains more than 20 million points.


(a) The whole registered street segment

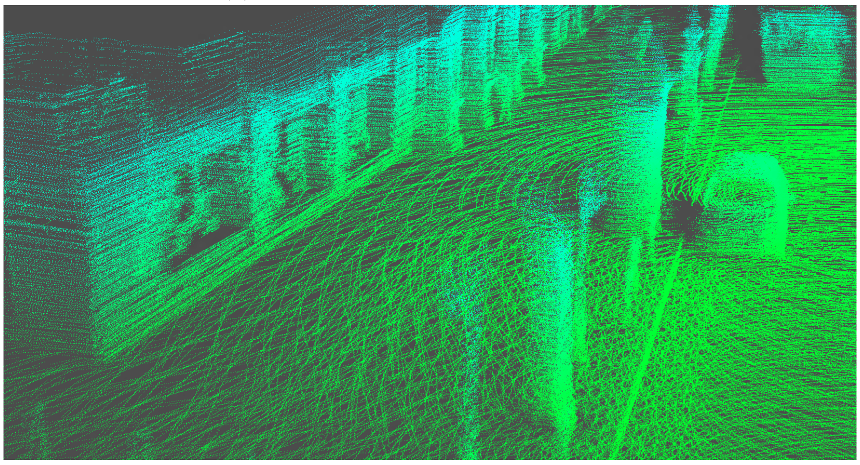

(b) Part of the street, zoomed in. Note the level of detail, even objects in the store window are visible.

Fig. 6: Registered street segment. For registration, 30 point clouds were used, final cloud contains almost 10 million points.

References

1. T. Luettel, M. Himmelsbach, and H.-J. Wuensche. Autonomous ground vehicles: Concepts and a path to the future. *Proceedings of the IEEE*, 100(Special Centennial Issue):1831 –1839, 13 2012.
2. Z. Zhang. Iterative point matching for registration of free-form curves and surfaces. *International journal of computer vision*, 13(2):119–152, 1994.
3. N. Wojke and M. Haselich. Moving vehicle detection and tracking in unstructured environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3082 –3087, may 2012.
4. A. Azim and O. Aycard. Detection, classification and tracking of moving objects in a 3d environment. In *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pages 802 –807, june 2012.
5. B. Douillard, J.P. Underwood, N. Kuntz, V. Vlaskine, A. Quadros, P. Morton, and A. Frenkel. On the segmentation of 3D LIDAR point clouds. In *International Conference on Robotics and Automation (ICRA)*, Shanghai, China, 2011.
6. B. Douillard, A. Quadros, P. Morton, J.P. Underwood, M. De Deuge, S. Hugosson, M. Hallstrom, and T. Bailey. Scan segments matching for pairwise 3d alignment. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3033 –3040, may 2012.
7. A. Makadia, A.I. Patterson, and K. Daniilidis. Fully automatic registration of 3d point clouds. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 1297 – 1304, june 2006.
8. J. Behley, V. Steinhage, and A.B. Cremers. Performance of histogram descriptors for the classification of 3d laser range data in urban environments. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 4391 –4398, may 2012.
9. A. J. Quadros, J. P. Underwood, and B. Douillard. An occlusion-aware feature for range images. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4428–4435, St. Paul, MN, USA, 2012.
10. C. H. Shen, S. S. Huang, H. Fu, and S. M. Hu. Adaptive partitioning of urban facades. In *SIGGRAPH Asia Conference*, pages 184:1–184:10, New York, NY, USA, 2011. ACM.
11. R. Wang, J. Bach, J. Macfarlane, and F.P. Ferrie. A new upsampling method for mobile lidar data. In *Applications of Computer Vision (WACV), 2012 IEEE Workshop on*, pages 17–24. IEEE, 2012.
12. A. Velizhev, R. Shapovalov, and K. Schindler. An implicit shape model for object detection in 3D point clouds. In *ISPRS Congress*, Melbourne, Australia, 2012.
13. J. Han, D. Kim, M. Lee, and M. Sunwoo. Enhanced road boundary and obstacle detection using a downward-looking LIDAR sensor. *IEEE Transactions on Vehicular Technology*, 61(3):971 –985, march 2012.
14. S. Thrun, M. Montemerlo, and A. Aron. Probabilistic terrain analysis for high-speed desert driving. *Proc. Robotics Science and Systems, Philadelphia, PA, USA*, 2006.
15. A. Petrovskaya and S. Thrun. Model based vehicle tracking for autonomous driving in urban environments. *Proceedings of Robotics: Science and Systems IV, Zurich, Switzerland*, 34, 2008.
16. C. Stiller and J. Ziegler. 3d perception and planning for self-driving and cooperative automobiles. In *Systems, Signals and Devices (SSD), 2012 9th International Multi-Conference on*, pages 1–7. IEEE, 2012.

17. W. Zhang. LIDAR-based road and road-edge detection. In *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pages 845 –848, June 2010.
18. M. Magnusson. *The Three-Dimensional Normal-Distributions Transform — an Efficient Representation for Registration, Surface Analysis, and Loop Detection.* PhD thesis, Örebro University, December 2009. Örebro Studies in Technology.
19. P. Biber and W. Strasser. The normal distributions transform: A new approach to laser scan matching. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 2743–2748, Las Vegas, USA, October 2003.