Innovation, Sciences et Développement économique Canada
Office de la Propriété Intellectuelle du Canada

Innovation, Science and Economic Development Canada
Canadian Intellectual Property Office

CA 2982044 C 2021/09/07

(11)(21) 2 982 044

(12) BREVET CANADIEN
CANADIAN PATENT

(13) C

(86) Date de dépôt PCT/PCT Filing Date: 2016/04/11
(87) Date publication PCT/PCT Publication Date: 2016/10/13
(45) Date de délivrance/Issue Date: 2021/09/07
(85) Entrée phase nationale/National Entry: 2017/10/06
(86) N° demande PCT/PCT Application No.: EP 2016/057935
(87) N° publication PCT/PCT Publication No.: 2016/162568
(30) Priorité/Priority: 2015/04/10 (EP15163231.2)

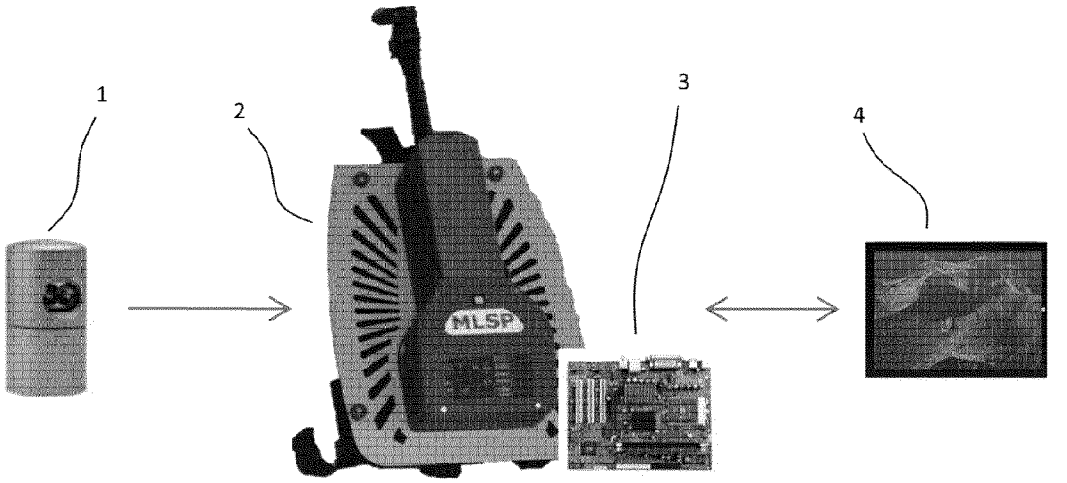
(51) Cl.Int./Int.Cl. *G01S 17/89* (2020.01), *G05D 1/02* (2020.01), *G09B 29/00* (2006.01)

(72) Inventeurs/Inventors:
SEQUEIRA, VITOR, PT;
WOLFART, ERIK, IT;
TADDEI, PIERLUIGI, IT;
CERIANI, SIMONE, IT;
SANCHEZ-BELENGUER, CARLOS, ES;
PUIG ALCORIZA, DAVID, IT

(73) Propriétaire/Owner:
THE EUROPEAN ATOMIC ENERGY COMMUNITY (EURATOM), REPRESENTED BY THE EUROPEAN COMMISSION, BE

(74) Agent: ROBIC

(54) Titre : PROCEDE ET DISPOSITIF DE CARTOGRAPHIE ET DE LOCALISATION EN TEMPS REEL
(54) Title: METHOD AND DEVICE FOR REAL-TIME MAPPING AND LOCALIZATION




(57) Abrégé/Abstract:
A method for constructing a 3D reference map useable in real-time mapping, localization and/or change analysis, wherein the 3D reference map is built using a 3D SLAM (Simultaneous Localization And Mapping) framework based on a mobile laser range scanner A method for real-time mapping, localization and change analysis, in particular in GPS-denied environments, as well as a mobile laser scanning device for implementing said methods.

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
13 October 2016 (13.10.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/162568 A1

(51) International Patent Classification:
*G01C 21/20* (2006.01) *G01S 17/89* (2006.01)
*G05D 1/02* (2006.01)

(21) International Application Number:
PCT/EP2016/057935

(22) International Filing Date:
11 April 2016 (11.04.2016)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
15163231.2 10 April 2015 (10.04.2015) EP

(71) Applicant: **THE EUROPEAN ATOMIC ENERGY COMMUNITY (EURATOM), REPRESENTED BY THE EUROPEAN COMMISSION** [BE/BE]; 1049 Brussels (BE).

(72) Inventors: **SEQUEIRA, Vitor**; Rua de Fontelos, 145, 4430-410 Vila Nova de Gaia (PT). **WOLFART, Erik**; Via Monte Grapp 19, 21020 Bardello (IT). **TADDEI, Pierluigi**; Via Volta 15A, 20064 Gorgonzola (MI) (IT). **CERIANI, Simone**; Via Giovanni XXIII, 3, 21049 Tradate (VA) (IT). **SANCHEZ-BELENGUER, Carlos**; Calle Santos Justo y Pastor 73-3, 46022 Valencia (ES). **PUIG ALCORIZA, David**; Via Bellini 40A, 20020 Lainate (MI) (IT).

(74) Agents: **KIHN, Henri** et al.; Office Freylinger S.A., 234, Route d'Arlon, BP 48, 8001 Strassen (LU).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

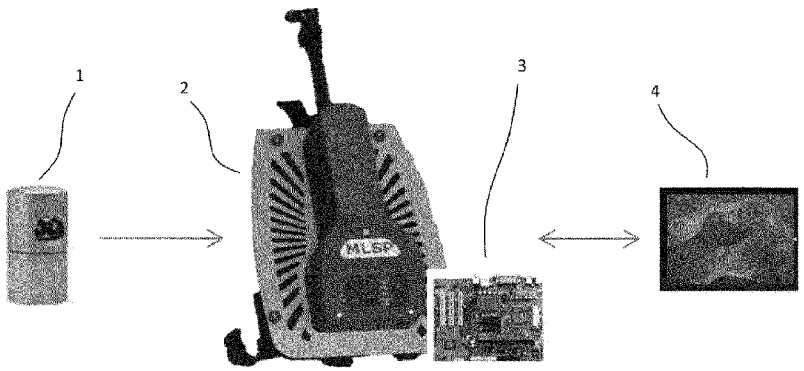
**Published**:

— *with international search report (Art. 21(3))*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

(54) Title: METHOD AND DEVICE FOR REAL-TIME MAPPING AND LOCALIZATION

Fig. 1




(57) Abstract: A method for constructing a 3D reference map useable in real-time mapping, localization and/or change analysis, wherein the 3D reference map is built using a 3D SLAM (Simultaneous Localization And Mapping) framework based on a mobile laser range scanner A method for real-time mapping, localization and change analysis, in particular in GPS-denied environments, as well as a mobile laser scanning device for implementing said methods.

WO 2016/162568 A1

# METHOD AND DEVICE FOR REAL-TIME MAPPING AND LOCALIZATION

## Technical field

[0001] The present invention generally relates to localization and mapping, especially in GPS-denied environments, such as indoors.

## Background Art

[0002] Different solutions have been described or are commercially available to allow for acquire environments for purposes such as localization or mapping. Different approaches have given rise to different solutions.

[0003] Among these, a number of commercial and prototype indoor navigation systems are based on inertial sensors (e.g. DLR's FootSLAM, Chirange Geospatial Indoor Tracking). They are small and inexpensive, however the position accuracy is low and drifts significantly over time. Furthermore, inertial systems do not generate map information. Therefore, they are only suitable for positioning and navigation purposes, not for map generation.

[0004] Other indoor positioning systems are based on the transmission of radio signals – similarly to GPS signals in outdoor environments. Some system use existing infrastructure (e.g. WiFi networks in airports, Navizon), others require the installation of dedicated infrastructure (e.g. NextNav, SenionLab).The systems have virtually no sensor costs (the client application uses a smart phone with dedicated software application), but they require network infrastructure emitting the radio signal. Furthermore, they do not generate map information. Therefore, they are only suitable for positioning and navigation purposes, not for map generation.

[0005] A further interesting product uses 3D scanning. ZEB1 is a commercial product that uses 3D laser scanning for fast (indoor) mapping. The laser is mounted on a spring and an oscillating movement needs to be created by hand. It generates an accurate 3D model of the indoor environment. However, the system does not provide immediate feedback to the user, as data processing is carried out off-line. Hence, the system is suitable for mapping but not for real-time localization.

[0006] A still further solution is a laser backpack developed at UC Berkley. It is a R&D project which proposes a backpack equipped with several 2D line scanners used to

generate a 3D model of indoor environments. Again, it does not provide for on-line visualization.

[0007] A last solution is called LOAM (Lidar Odometry and Mapping) and consists of a portable sensor with associated algorithms that combine laser scanning and video imagery for real-time localization and mapping.

[0008] Almost all these solutions lack real-time/on-line visualization and more importantly they do not allow for any direct user interaction on the acquiring and processing steps.

[0008a] US2014/005933A1 discloses a system and method for mapping parameter data acquired by a robot mapping system. Parameter data characterizing the environment is collected while the robot localizes itself within the environment using landmarks. Parameter data is recorded in a plurality of local grids, i.e., sub-maps associated with the robot position and orientation when the data was collected. The robot is configured to generate new grids or reuse existing grids depending on the robot's current pose, the pose associated with other grids, and the uncertainty of these relative pose estimates. The pose estimates associated with the grids are updated over time as the robot refines its estimates of the locations of landmarks from which it determines its pose in the environment. Occupancy maps or other global parameter maps may be generated by rendering local grids into a comprehensive map indicating the parameter data in a global reference frame extending the dimensions of the environment.

[0008b] TIMOTHY LIU ET AL: "Indoor localization and visualization using a human-operated backpack system", INDOOR POSITIONING AND INDOOR NAVIGATION (IPIN), 2010 INTERNATIONAL CONFERENCE ON, IEEE, PISCATAWAY, NJ, USA, 15 September 2010 (2010-09-15), pages 1-10, XP031809367, ISBN: 978-1-4244-5862-2 discloses techniques for indoor localization and visualization using a human-operated backpack system equipped with 2D laser scanners and inertial measurement units (IMU), in which scan matching based algorithms are used to localize the backpack in complex indoor environments. To address misalignment between successive images used for texturing when building 3D textured models, the authors propose an image based pose estimation algorithm to refine the results from the scan matching based localization.

[0008c] WO2015/017941A1 discloses systems and methods for generating data indicative of a three-dimensional representation of a scene. Current depth data

indicative of a scene is generated using a sensor. Salient features are detected within a depth frame associated with the depth data, and these salient features are matched with a saliency likelihoods distribution. The saliency likelihoods distribution represents the scene, and is generated from previously-detected salient features. The pose of the sensor is estimated based upon the matching of detected salient features, and this estimated pose is refined based upon a volumetric representation of the scene. The volumetric representation of the scene is updated based upon the current depth data and estimated pose. A saliency likelihoods distribution representation is updated based on the salient features. Image data indicative of the scene may also be generated and used along with depth data.

**Technical problem**

[0009] It is an object of the present invention to provide a system, device and method which do not only allow for real-time acquisition, mapping and localization particularly in GPS-denied environments, but which will also provide for real-time visualization and the possibility for user interaction. Moreover, the present invention should allow for also providing real-time comparison of the current acquired data with previously acquired maps. This would allow identifying differences or changes that occurred since the last mapping. Such on-line identification of changes or differences may be of great benefit in applications such as security inspections, civil construction, as well as emergency or disaster management.

**General Description of the Invention**

[0010] To achieve this object, the present invention proposes, in a first aspect, a method for constructing a 3D reference map of an environment useable in (a method for) real-time mapping, localization and/or change analysis, comprising the following steps:

(a) acquiring (3D) scanner data of the environment with a mobile real-time laser range scanner at a rate of at least 5 frames (i.e. 5 point clouds), preferably at least 10 frames per second,

(b) constructing, using the (3D) scanner data for each of a plurality of poses of the laser range scanner, each pose having an associated point cloud defined by the scanner data, a map presentation, the map presentation having a data structure configured for random sample access thereto in constant time, fast nearest neighbor search and scalability over large areas, and

(c) building, using the map presentation, the 3D reference map for the environment using a 3D Simultaneous Localization And Mapping (3D SLAM) framework, said building comprising

(i) using an odometer module estimating a current pose of the laser range scanner for each point cloud based on the registration of the (last) point cloud to the local map presentation,

(ii) using a local trajectory optimization module refining the trajectory of a (sub)set of point clouds in order to minimize the drift in the local map presentation, and

(iii) performing offline a global trajectory optimization by reconstructing an entire map of the environment (preferably by using the entire set of point clouds) taking advantage of (or taking into account) loop closures, thereby forming said 3D reference map.

[0011] The invention further relates to such a method, wherein the local trajectory optimization module comprises a local window mechanism optimizing a trajectory fragment composed by a set of poses and their associated point clouds with respect to a map built up to the last registered set, wherein points are preferably converted in world coordinates using pose interpolation in $\mathbb{SE}3$ group and wherein a generalization of Iterative Closest Point method is preferably used to find the trajectory that better aligns all the points to the map; wherein the local window mechanism operates such that, when the distance between the first and the last pose in the list is larger than a threshold, cloud poses are optimized and a new list is produced with the refined pose and the input clouds.

[0012] In a particularly preferred embodiment, the data structure is set to natively handle 3D points and is based on a hybrid structure composed by a sparse voxelized structure used to index a (compact dense) list of features in the map presentation, allowing constant time random access in voxel coordinates independently from the map size and efficient storage of the data with scalability over the explored space.

[0013] In a still further preferred embodiment, the data structure may maintain five different representations of the data stored, thereby granting consistency between internal data representations after each map update, the five representations being

(i) a (compact and dense) list of features, $L$ and an index to the last element, $L_{last}$ where each element, $l_i \in L$, contains all the information associated to a

feature in the map, such as position and normal unit vector in world coordinates, and preferably additional information,

(ii) a (compact and dense) validity mask, $M$, where each element, $m_i \in M$, is a boolean value indicating if its corresponding sample, $l_i \in L$, is valid or not, ensuring that $m_i = 0, i > L_{last}$,

(iii) a list of holes, $H$, where each element, $h_i \in H < L_{\text{last}}$, indicates that $l_{h_i}$ is not valid so, $m_{h_i} = 0$,

(iv) a sparse voxel representation $V$, built with a parametrizable cell size, that stores in each cell, $v_i \in V$, the index of its corresponding feature in $L$, wherein features in $L$ and cells in $V$ are related in a one-to-one manner, based on the position of $l_i$ and the cell size of $V$, and

(v) a kd-tree, $K$, which is used to perform nearest neighbor searches on the map and which only stores references to the dense list $L$ to keep its memory footprint low.

[0014] The present method may further comprise the step, wherein, given an area of interest expressed by a central position and a radius, inner features are selected by looping over the elements stored in $L$ and the kd-tree $K$ is rebuilt as a fast mechanism for nearest neighbor searches.

[0015] In a second aspect, the invention relates to a method for real-time mapping, localization and change analysis of an environment, i.e. relative to the 3D reference map of the environment which is available from a method according to the first aspect of the invention as described above or from a such a 3D reference map already updated or modified through a previous run of the present method, in particular in a GPS-denied environment, preferably comprising the following steps:

(a) acquiring (3D) scanner data of the environment with a real-time laser range scanner at a rate of at least 5 frames (point clouds) per second,

(b) during place recognition, identifying a current location of the laser range scanner inside a known environment (i.e. within the 3D reference map) with no prior knowledge of the scanner pose during place recognition and pre-computing of simple and compact descriptors of the scene acquired by the laser range scanner using a reduced search space within the scene; in order to self-localize the scanner

in real-time, or identifying a current location of the laser range scanner within the 3D reference map making use of the pre-computed descriptor space in order to self-localize the scanner in real-time using pre-computed compact descriptors of the 3D reference map at potential scanner poses,

(c) after determination of the localization of the scanner in the known environment (i.e. within the 3D reference map), tracking the scanner pose by registering current scanner data inside a 3D reference map using standard Iterative Closest Point method employing data comprising nearest-neighbor information,

(d) calculating the distance between each scan point in the current scanner data and nearest point in the 3D reference map, wherein change analysis is performed by applying a threshold to this distance, (whereby each point in the current scanner data which has a corresponding neighbor in the reference model that is further than the threshold is considered a change), and

(e) displaying information about the 3D reference map and the current (3D) scanner data on a real-time user interface, said information being preferably color-coded according to a change/no-change classification of said information.

[0016] Preferably, step (b) comprises the identification of a set of possible locations of the scanner based on the scanner data of step (a), said step (b) further the following substeps:

(b1) based on the last scanner data, computing an associated descriptor $\mathbf{q}$ and recovering a set of candidate locations $\Gamma$. The candidate locations have a descriptor similar to $\mathbf{q}$, i.e. the distance in descriptor space is smaller than a threshold radius $r$. The set of candidate locations $\Gamma$ can be recovered by performing a radial search on $T$ given a threshold radius $r$ in the descriptor space, preferably for 360° horizontal view scanner data, increasing the candidate locations by computing additional input descriptors by horizontally shifting range values, each descriptor corresponding to the readings that the scanner would produce if rotated on its local axis and then rotating according to $i$ each resulting set of candidate locations,

(b2) associating a weight $\mathbf{w}_{\Gamma_p}$ to each potential location $\mathbf{\Gamma}_p \in \Gamma$:

$$\mathbf{w}_{\Gamma_p} = 1 - \frac{\|\mathbf{d}_p - \mathbf{q}\|}{r},$$

where $\mathbf{d}_p$ is the descriptor associated to the location $\mathbf{\Gamma}_p$ retrieved from $T$, $\mathbf{w}_{\mathbf{\Gamma}_p}$ is 1 for perfectly matching descriptors and 0 for descriptors on the search sphere boundary, and

(b3) collecting weights in $\mathbf{w}$ and normalizing these weights to have $\max \mathbf{w} = 1$.

[0017] Advantageously, step (b) further comprises the substeps

(b4) updating the set of candidate locations while the sensor moves by estimating the movement (using the odometer module as described in step (c)(i) of the method of the first aspect above) and re-evaluating the weight for each initial candidate pose based on the query results at the new pose, and

(b5) iterating the update substep until the candidate poses converge to a single location (i.e. until the method is able to disambiguate the current pose).

[0018] Particularly for ground motion, whereby the laser range scanner is mounted on a person (e.g. with a backpack) or on a vehicle traversing a floor, the method may comprise the following steps

(i) identifying in the 3D reference map the extents of a floor, wherein floor extraction is performed over a sparse voxel representation of the environment (3D reference map), $V$, where each full cell, $v^{(i)}$, of the sparse voxel representation contains a normal vector to the surface locally defined by the points around its centroid, $\bar{\mathbf{n}}^{(i)}$, by extracting a subset of voxels that represent candidate floor cells, $F \subseteq V$, by checking that the vertical component in their associated normals is dominant, i.e. $\bar{\mathbf{n}}^{(i)} \cdot (0,0,1)^T \geq \varepsilon$, where $\varepsilon$ is typically a value between 0.5 and 1

(ii) determining reachability of cells, wherein given a reachable cell $f \in F$, all surrounding cells $(g^{(1)}, g^{(2)}, \ldots, g^{(m)}) \in F$ are considered as reachable if the following conditions are satisfied:

$$\left\| f - g^{(i)} \right\| \leq \theta_0 \tag{6}$$

$$\left\| f_Z - g_Z^{(i)} \right\| \leq \theta_1 \tag{7}$$

$$C_{g^{(i)}} \cap V = \varnothing \tag{8}$$

where $\theta_0 \geq V_{\text{cellSize}}$ in (6) stands for the maximum step distance (e.g. 0.5 meters for a walking motion, or $V_{\text{cellSize}}$ for a car motion), $\theta_1$ in (7) stands for the maximum vertical step size and $C_{g^{(i)}}$ in (8) stands for the simplified volume of the observer, centered over the floor cell $g_i$.

[0019] The map structure useable in the context of the present invention preferably comprises two different lists of elements that are stored and synchronized: a (compact) list of planes, $L$, and a (dense) grid of voxels, $V$, built with a specific voxel size, each plane $l_i \in L$ storing a position in world coordinates, $\mathbf{p_i}$, and a unit normal, $\overline{\mathbf{n_i}}$; wherein each voxel, $v_i \in V$ stores a current state that can be either full, empty or near, full voxels storing an index to the plane $l_{v_i} \in L$, whose associated position falls into, empty cells storing a null reference and near cells storing an index to the plane $l_{v_j} \in L$ whose associated position distance $d_v$ to the voxel centre is the smallest; preferably a near voxel is considered only if the distance $d_v$ is under a given threshold $d_{max}$, otherwise the voxel is considered empty.

[0020] To improve overall system robustness, it is considered to combine the scanner tracking with an odometer (e.g. using the odometer module as described in step (c)(i) of the method of the first aspect above), such that after a pose has been estimated, its associated points in world coordinates are stored into a kd-tree (thus creating an odometer map), given a new acquisition (point cloud acquired) by the laser range scanner, (i.e. when a registration algorithm creates the sets of points) ($\mathbf{p}_i^W$), it looks for nearest neighbors in both the 3D reference map ($\mathbf{q}_i^M, \overline{\mathbf{n}}_i^M$) and in the previously fixed point cloud (odometer map) ($\mathbf{q}_i^O, \overline{\mathbf{n}}_i^O$), wherein correspondences are defined as:

$$c_i^{(j)} =$$

$$\begin{cases} \{\mathbf{p}_i^W, \mathbf{q}_i^M, \overline{\mathbf{n}}_i^M\} & \left\|\mathbf{p}_i^W - \mathbf{q}_i^M\right\| - s \leq \left\|\mathbf{p}_i^O - \mathbf{q}_i^O\right\| \\ \{\mathbf{p}_i^W, \mathbf{q}_i^O, \overline{\mathbf{n}}_i^O\} & \left\|\mathbf{p}_i^W - \mathbf{q}_i^M\right\| - s > \left\|\mathbf{p}_i^O - \mathbf{q}_i^O\right\| \end{cases},$$

where $s$ corresponds to the voxel cell size and compensates the different resolution between the voxelized ground truth map and the non-discretized kd-tree of the previously fixed cloud.

[0021] In a third aspect, the invention proposes a mobile laser scanning device for real-time mapping, localization and change analysis, in particular in GPS-denied environments, implementing one or more of the methods described herein. In particular, the invention relates to a mobile laser scanning device for real-time mapping, localization and change analysis, in particular in GPS-denied environments, comprising a real-time laser range scanner, a processing unit, a power supply unit and a hand-held visualization and control unit, wherein the real-time laser range scanner is capable of acquiring the environment with a rate of at least 5 frames, preferably at least 10 frames per second to provide scanner data, the processing unit is arranged to analyze said scanner data and to provide processing results comprising 3D map/model, localization and change information to the hand-held visualization and control unit, which is arranged to display said processing results and to allow a user to control the mobile laser scanning device.

[0022] A device according to the invention is thus capable of on-line, real-time processing providing 3D mapping/modelling of the environment, precise localization of the user (with respect to generated map or existing map/model), change detection with respect to previously acquired model and relies fully on laser signal which makes it independent of ambient illumination and GPS signal. Moreover, it does not require additional sensors such as GPS or inertial sensors. Nonetheless, the present invention does not exclude adding further sensors if deemed useful. Thus, optional sensors may be added to enrich the generated model (e.g. color cameras). Furthermore, although the device is capable of providing on-line and real-time results to the user, it is further foreseen to use the acquired data and to further process it off-line, e.g. for refinement of acquired 3D model for future localization and change analysis.

[0023] The device according to the present invention may be used and is useful in numerous applications such as e.g. 3D (indoor) mapping/modelling, facility management, accurate, real-time indoor localization and navigation, design information verification, change analysis (e.g. for safeguards inspections), progress monitoring (e.g. for civil construction), disaster management and response, etc.

[0024] In the mobile laser scanning device, the visualization and control unit is preferably a touch screen computer, more preferably a tablet computer.

[0025] The mobile laser scanning device is most preferably a backpack or vehicle mounted device.

[0026] In a fourth aspect, the invention proposes the use of methods or of mobile laser scanning devices as described herein for 3D outdoor and indoor, preferably indoor mapping/modelling; facility management; accurate and real-time indoor localization and navigation; assistance to disabled or elderly people; design information verification; change analysis, such as for safeguards inspections; progress monitoring, such as for civil construction; or disaster management and response.

[0027] A fifth aspect concerns a computer program product having computer executable instructions for causing a programmable device, preferably a mobile laser scanning device or its processing unit as described herein to execute one or more of the methods of the present invention.

[0028] In a sixth aspect, the invention also relates to a computer-readable medium, having stored therein data representing instructions executable by a programmed processor, the computer-readable medium comprising instructions for causing a programmable device, preferably a mobile laser scanning device of the invention or its processing unit, to execute one of the present methods.

[0028a] In yet another aspect, a method for real-time mapping, localization and change analysis of an environment is provided. The method comprises the following steps:

(A) if no 3D reference map of the environment exists, constructing a 3D reference map of said environment by

(a) acquiring the environment with a mobile real-time laser range scanner (1) at a rate of at least 5 frames per second to provide 3D scanner data,

(b) constructing a map presentation using the 3D scanner data for each of a plurality of poses of the laser range scanner (1), each pose having an associated point cloud defined by the 3D scanner data, the map presentation having a data structure set to natively handle 3D points which is based on a hybrid structure composed by a sparse voxelized structure used to index a compact dense list of features in the map presentation allowing constant time random access in voxel

Date reçue/Date Received 2020-08-28

coordinates independently from the map size and efficient storage of the data with scalability over the environment, and

(c) building, using the map presentation, the 3D reference map for the environment using a 3D Simultaneous Localization And Mapping (3D SLAM) framework, said building comprising

(i) using an odometer module, estimating a current pose of the laser range scanner (1) based on the registration of a last point cloud to a local map presentation,

(ii) using a local trajectory optimization module, refining the trajectory of a set of point clouds in order to minimize the drift in the local map presentation, and

(iii) performing offline a global trajectory optimization by reconstructing an entire map of the environment taking into account loop closures of trajectories, thereby forming said 3D reference map;

and

(B) based on an existing 3D reference map of the environment, performing real-time mapping, localization and change analysis of said environment by

(d) acquiring the environment with the real-time laser range scanner (1) at a rate of at least 5 frames per second to provide 3D scanner data,

(e) during place recognition, identifying a current location of the laser range scanner (1) inside the environment with no prior knowledge of the laser range scanner pose during place recognition, and pre-computing of simple and compact descriptors of a scene acquired by the laser range scanner (1) using a reduced search space within the scene in order to self-localize the scanner in real-time, each descriptor of the scene comprising a range image of regular bins where each bin has an estimated median range value,

(f) after determination of the localization of the scanner in the environment, tracking the scanner pose by registering current scanner data inside said existing 3D reference map of the environment using standard Iterative Closest Point method employing data comprising nearest-neighbor information stored in the 3D reference map,

Date Reçue/Date Received 2021-06-03

(g) calculating the distance between each scan point in the current scanner data and nearest point in the 3D reference map, wherein change analysis is performed by applying a threshold to this distance, whereby each point in the current scanner data which does not have a corresponding neighbor in the reference model is considered a change,

(h) displaying information about the 3D reference map and the current 3D scanner data on a real-time user interface.

[0029] The above aspects, further even more particulars of variants, alternatives and combination of features, as well as their advantages will be described more in detail below.

**Brief Description of the Drawings**

[0030] Preferred aspects and embodiments of the invention will now be described, by way of example, with reference to the accompanying drawings.

Fig. 1: Hardware components of a preferred embodiment of a mobile laser scanning device according to the present invention, called Mobile Laser Scanning Platform (MLSP system, or simply MLSP) comprising a 3D laser scanner 1, a backpack 2, a processing unit 3 (contained within the backpack, shown separately for illustration only) and a tablet 4.

Fig. 2: Snapshot (black-and-white of an originally colored snapshot) of the user interface as it is provided to the user in real-time. It shows a tunnel environment which has been

Date Reçue/Date Received 2021-06-03

scanned at two points of time. In the actual color display, green indicates no change between the acquisitions and red indicates new constructions between the two acquisitions.

Fig. 3: Effect of the loop closure on a sample track of the Kitti datasets. The trajectory is shown as estimated on-line and as globally optimized trajectory. The (actual) map is colored according to the normal vectors of the points with a different scheme for the two maps (such as violet area being the local map).

Fig. 4: Point selection time for both local map (typically containing less than 1M features) and global map (typically containing more than 1M features), Fig. 4(a) Local map point selection time for different search radius and Fig. 4(b) Global map point selection time for different search radius.

Fig. 5: Preferred embodiment of proposed map representations. Full cells are displayed as dark gray boxes. Near cells are represented as light gray boxes with a line connecting their centroid with the associated nearest neighbor. Empty cells are displayed as white boxes.

Fig. 6: Rotation histograms for a symmetric environment and for a non-symmetric one.

Fig. 7: Inlier selection. Axes represent the main dominant dimensions of the detected transformations. Each point represents a candidate transformation grayed according to the iteration in which they have been marked as outliers (some outlier transformations too far from the center have been omitted). Dark gray points in the central ellipses represent transformations marked as inliers. The ellipses represent the normal estimations at specific subsequent iterations.

Fig. 8: Results of the floor extraction algorithm. Black points represent the scanner positions during acquisition. These locations have been used to automatically select the set of initial active cells.

Fig. 9: Empirical parameter selection for the search space reduction. (left) deviation of the sensor with respect to the mean height to the floor observed during several walkthroughs. (right) deviation of the sensor with respect to the vertical axis (Z) observed during several walkthroughs.

Fig. 10: Drift analysis using only place recognition (tracking by classification) where the classifier contains data related to multiple environments. The ground truth for such

experiment is considered the final trajectory generated by the tracking module initialized in the correct building for a description of the adopted error metric.

Fig. 11(a) and (b): Two sample paths used to generate the drift analysis shown in Figure 10. Dashed line the ground truth path estimated using the complete system. Solid line the path estimated using tracking by classification. The black circle shows the frame after which the user has been uniquely identified in a specific building.

Fig. 12: Results of the proposed inlier selection algorithm.

Fig. 13: Results of the odometer integration during a sample walk-through inside a building where the sensor moves to a non-mapped room (A, illustrated in (right)) without losing track of its position and, then, it performs two loops outside the building (C and D).

Fig. 14: Tracking accuracy comparison between the standard ICP (dashed line) and the proposed robust implementation (solid line) in an environment without outliers (upper) and in an environment with outliers (lower).

Fig. 15: System overall performance during tracking for a backpack mounted setup: solid gray lines are the time spent in processing each frame (in seconds).The dashed horizontal line indicates the maximum execution time for real-time results using a Velodyne HDL-32E sensor (12Hz).

[0031] Further details and advantages of the present invention will be apparent from the following detailed description of several non-limiting aspects and embodiments with reference to the attached drawings. Indeed, the detailed description below is not to be construed to limit the scope of the invention, but rather to illustrate particular aspects presented in the general description, claims and drawings.

**Description of Preferred Embodiments**

[0032] As already mentioned previously, one of the main advantages of preferred embodiments of the present invention as herein described lies in the concept of providing real-time change analysis and monitoring in GPS-denied (e.g. indoor) environments. The user is able to inspect a facility and view the changes on a handheld device as he walks through the facility. The preferred underlying methodologies and algorithms are summarized below and further detailed thereafter.

[0033] A basic workflow for a previously unknown (unscanned) location requires in principle two steps: (A) the construction of a 3D reference model at T0 and (B) the

localization, tracking and change analysis based on 3D reference model at T1. When revisiting such a location or in cases where an appropriate map already exists, step (B) will be sufficient.

[0034] (A) Construction of 3D reference map

[0035] The 3D reference map is built using a 3D SLAM (Simultaneous Localization And Mapping) implementation based on a mobile laser range scanner as described below. The main features preferably are:

1) An efficient map presentation that allows random sample access in constant time, fast nearest neighbor search and scalability over large areas (see section A.2. below).
2) The SLAM framework (see section A.3. below) contains:
    a) An odometer to estimates the current pose based on the registration of the last cloud to the local map representation.
    b) A local trajectory optimization that refines the trajectory of a set of clouds in order to minimize the drift in the generated map.
    c) A global trajectory optimization that allows reconstructing an entire map of the environment taking advantage of loop closures.

[0036] The odometer is typically performed in real-time. The map optimization can be carried out in a post-processing step.

[0037] (B) Localization, tracking and change analysis based on 3D reference model

[0038] The real-time localization, tracking and change analysis generally requires an existing 3D reference map of the environment which has been previously been generated as described above. The main components preferably are

1) During place recognition the system identifies the current location inside the known environment with no prior knowledge of the sensor pose. It pre-computes simple and compact descriptors of the scene and uses an efficient strategy to reduce the search space in order to self-localize the sensor in real-time (see section B.2. below).
2) Once the sensor is localized in the known environment, the system starts tracking the sensor pose by registering the current observation (3D scan) inside the 3D reference map using the standard Iterative Closest Point (ICP) method. In

order to accurately track the sensor pose in real-time, the system implements a number of improvements, e.g. it employs a data structure specially designed for fast nearest neighbor searches (see section B.3. below).

3) Given the nearest-neighbor information in the data structure, MLSP can efficiently calculate the distance between each scan point in the current observation and nearest point in the 3D reference model. The change analysis is performed by applying a simple threshold to this distance, e.g. each point in the current scan which does not have a corresponding neighbor in the reference model (or which has a corresponding neighbor in the reference model but which that is farther than the threshold) is considered a change. The real-time user interface shows the 3D reference model and the current observations which are color-coded according to a change/no-change classification.

[0039] A. Construction of 3D reference map

[0040] Precise 3D mapping and 6DOF trajectory estimation using exteroceptive sensors are key problems in many fields. Real-time moving laser sensors gained popularity due to their precise depth measurements, high frame rate and large field of view.

[0041] In one preferred aspect, the present invention proposes an optimization method or framework for Simultaneous Localization And Mapping (SLAM) that properly models the acquisition process in a scanning-while-moving scenario. Each measurement is correctly reprojected in the map reference frame by considering a continuous time trajectory which is defined as the linear interpolation of a discrete set of control poses in $\mathbb{SE}3$. The invention also proposes a particularly efficient data structure that makes use of a hybrid sparse voxelized representation, allowing large map management. Thanks to this the inventors were also able to perform global optimization over trajectories, resetting the accumulated drift when loops are performed.

[0042] The inventors experimentally showed that such framework improves localization and mapping w.r.t. solutions that compensate the distortion effects without including them in the optimization step. Moreover, the inventors show that the proposed data structure provides linear or constant operations time w.r.t. the map size in order to perform real time SLAM and handles very large maps.

[0043] A.1. Introduction

[0044] Generation of 3D maps and estimation of trajectories are fundamental building blocks for a wide variety of applications in robotics, autonomous guidance and surveillance. Simultaneous Localization And Mapping (SLAM) techniques jointly build the map of an unknown environment and localize the sensor in the same environment. SLAM formulations have been proposed for standard cameras, depth cameras and laser scanners. Most SLAM systems based on laser scanners use variations of the Iterative Closest Point (ICP) algorithm to perform scans alignments. A review of ICP algorithms focused on real time applications can be found in S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in 3DIM, 2001. Real time moving 3D LIDAR sensors, such as Velodyne scanners, recently gained popularity: these devices have a high data rate, often provide a complete 360° horizontal field and have a good accuracy on distance measurements.

[0045] Such sensors (scanners) acquire measurements while moving and thus represent non-central projection systems that warp acquired frames along the trajectory path. Alignment of such produced point clouds requires a proper treatment of the warping effect on the 3D points. The SLAM framework proposed in F. Moosmann and C. Stiller, "Velodyne SLAM," in IVS, 2011, unwarps each cloud given the current speed of the sensor, performs ICP and unwarps again the points with the new estimated speed. LOAM algorithm (J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time," in RSS, 2014) performs a continuous estimation of the motion by focusing on edges and planar features to remove the warping effect in each cloud. When a complete frame is generated it unwarps the final point cloud using the predicted final pose. The work of C. H. Tong, S. Anderson, H. Dong, and T. D. Barfoot, "Pose interpolation for laser-based visual odometry," Journal of Field Robotics, vol. 31, pp. 731-757, 2014, performs interpolation employing a continuous-time Gaussian Process Model (GPGN) that relies on matched features in the acquisition reflectance images.

[0046] In a preferred aspect of the present invention it is proposed to use a local window mechanism that optimizes a trajectory fragment composed by a set of poses and their associated point clouds with respect to the map built up to the last registered set. Points are converted in world coordinates using pose interpolation in $\mathbb{SE}3$ group and a generalization of ICP is used to find the trajectory that better aligns all the points to the map. In this formulation the unwarp operation is part of the optimization strategy.

[0047] An important aspect for SLAM systems is their scalability to large environments and a real time management of the map to support the optimization routine. Generally scalability is achieved using sparse structures such as general octrees, dense voxel maps that use volume cyclical indexing, or sparse representations based on voxel hazing. In one aspect, the invention focuses on a data structure that natively handles 3D points and that is based on a hybrid structure composed by a sparse voxelized structure, which is used to index a compact dense list of features. This allows constant time random access in voxel coordinates independently from the map size and efficient storage of the data with scalability over the explored space. The presently proposed structure is capable of maintaining in memory the entire global map and to update local sections in case graph optimization is employed (e.g. to perform loop closures).

[0048] Main contributions of some embodiments of the invention are (i) the use of a generalized ICP algorithm incorporating the unwarping in the estimation process, (ii) the use of an efficient structure for the map management that allows both fast spatial queries and big environment management. The inventors have validated their approach using publicly available datasets and additional acquired indoor/outdoor environments.

[0049] Section A.2. below presents the data structure for map management and its available operations; Section A.3. presents the optimization framework; Section A.4. shows experimental results obtained with this method and, Section A.5. draws some conclusions.

[0050] A.2. Map representation

[0051] A data structure suited for real-time SLAM applications should provide (i) random sample access in constant time (on average) to stored features, (ii) exhaustive feature iteration in linear time w.r.t. the number of elements stored and (iii) fast nearest neighborhood searches given a query feature. Moreover, it should provide (iv) scalability over the explored space and (v) it should efficiently support feature addition and removal.

[0052] Property (i) is generally associated to dense voxel representations, where memory requirements for scalability (iv) are the major drawback and exhaustive explorations (ii) are slow. Property (ii), conversely, is associated to sparse structures, where memory requirements (iv) are very low, but random access times (i) are slow (logarithmic in case of kd-trees). To exploit the intrinsic benefits of both dense and

sparse structures while retaining all the required properties, the proposed preferred map structure maintains five different representations of the data stored. Consistency between internal data representations should be granted after each map update.

(i) A compact and dense list of features, $L$ and an index to the last element, $L_{last}$, where each element, $l_i \in L$, contains all the information associated to a feature in the map (position, normal and additional information).

(ii) A compact and dense validity mask, $M$, where each element, $m_i \in M$, is a boolean value indicating if its corresponding sample, $l_i \in L$, is valid or not, ensuring that $m_i = 0, i > L_{last}$.

(iii) A list of holes, $H$, where each element, $h_i \in H < L_{\text{last}}$, indicates that $l_{h_i}$ is not valid so, $m_{h_i} = 0$.

(iv) A sparse voxel representation $V$, built with a parametrizable cell size, that stores in each cell, $v_i \in V$, the index of its corresponding feature in $L$. Features in $L$ and cells in $V$ are related in a one-to-one manner, based on the position of $l_i$ and the cell size of $V$. The present sparse voxel representation is based on an OpenVDB structure (K. Museth, "Vdb: High-resolution sparse volumes with dynamic topology," *ACM Transaction on Graphics,* vol. 32, no. 3, 2013).

(v) A kd-tree, $K$, that is used to perform nearest neighborhood searches on the map. $K$ only stores references to the dense list $L$ to keep its memory footprint low. The kd-tree can be built on a local region of the map if required (e.g. following an area around the last observation location).

[0053] By having a dense list of features, time for exhaustively exploring the entire map is linear in the number of elements contained. On the other hand, arbitrary queries are solved at constant random access time (on average) by exploiting the OpenVDB sparse voxel structure and caching system.

[0054] Given a new feature $p$ to be added to the map, the proposed data structure is modified as follows: consider the feature's world position, $\mathbf{p}^W$ and compute its corresponding voxel cell, $v_i$. If the cell is already filled ($v_i \geq 0$), its associated information is retrieved from $l_{v_i}$ and the value is updated if required. Otherwise ($v_i < 0$) a new

feature is added to the structure. To do so, the insertion position, $j$, in $L$ is computed as follows:

$$j = \begin{cases} h_0 \in H & if\ H \neq \varnothing \\ L_{\text{last}} + 1 & if\ H = \varnothing \end{cases}$$

then, internal values are updated as follows:

$$v_i = j, \qquad l_j = p, \quad m_j = 1$$

and

$$L_{\text{last}} = L_{\text{last}} + 1 \quad if\ H = \varnothing$$

$$H = H - \{h_0\} \quad if\ H \neq \varnothing$$

[0055] This way, while the set of holes contains elements, feature addition fills the gaps in the dense representation. When no holes remain, features are added at the end of the list.

[0056] In case a feature of the map has to be deleted, its corresponding voxel cell, $v_i$, is computed in the same way as before. The value stored in $v_i$ indicates the feature position in the dense list, $l_{v_i}$, and values are updated as follows:

$$m_{v_i} = 0, \quad H = H + \{v_i\}, \quad v_i = -1$$

[0057] This way, deleting features generates new holes in the dense list, without updating the value of $L_{\text{last}}$. Since $M$ and $H$ are correctly updated during the operation, internal data representation is still consistent, but the presence of too many holes may lead to decreasing performance.

[0058] To face this problem, the inventors propose in a particularly preferred embodiment to introduce a compact operation that populates the holes with the last elements in the lists by performing a swap in both $L$ and $M$ vectors. Affected values in $V$ are then updated according to the new positions and $L_{\text{last}}$ is moved to the new last element of the compacted list. The cost of this operation is linear with respect to the number of holes so, in case $H = \varnothing$, it does nothing.

[0059] Finally, in order to provide a fast mechanism for nearest neighbor searches, given an area of interest expressed by a central position and a radius, inner features

may be selected by looping over the elements stored in $L$ (linear cost to the number of samples in the map) and the kd-tree $K$ is rebuilt. Elements in $K$ only store a reference to the associated features in $L$, thus $K$ memory space is kept small (linear in the number of features present in the area of interest) and constant on average. The same operation can be performed without iterating over the entire list by visiting the voxel structure. The inventors investigate in the experimental section the differences between these two mechanisms.

[0060] Once the tree has been created, it will remain valid even if new features are added (already existing elements in $L$ are not changed) or existing features are deleted (elements in $L$ are marked as holes, but their value is not replaced), but not if both operations are performed (removed elements in $L$ can be overwritten).

[0061] To perform the proposed operations efficiently, cloud additions are preferably postponed until a new kd-tree is required. When this happens, already existing features in the map outside the area of interest are deleted, creating new holes. Then, postponed clouds are added, by only adding the features that are inside the interest area. This way, previously created holes are filled with the new samples in constant time. If after all the additions there are still holes (more features were deleted than added), a compact operation may be performed, with a linear cost with respect to the remaining number of holes. Finally, $K$ is rebuilt using the elements of $L$ and can be used until a new one is required.

[0062] A.3. SLAM framework

[0063] A preferred optimization framework of the invention is composed by two consecutive modules: an odometer that estimates the pose of each cloud given a map and a local trajectory optimizer that refines the trajectory of a set of clouds. Both modules employ the map data structure as described herein to handle the growing map.

[0064] Each feature stored in the map $M$ is composed by a point world position $\mathbf{p}^W$, its normal unit vector $\mathbf{n}^W$ and additional information (e.g., reflectance). The latter are not used in the registration steps. This framework can also be extended to perform a global trajectory optimization that allows reconstructing an entire map of the environment taking advantage of loop closures.

[0065] The input of such a framework is a set of 3D point clouds $\{C_i\}$ produced with the data streamed by the sensor (in case of a Velodyne scanner, the point cloud is

generated after a complete revolution of the sensor). Each point cloud $C_i$ is composed by a set of points $P = \{\mathbf{p}_j\}, j = 1 \cdots N_p$, a set of relative timestamps $T = \{t_j\}$ and a set of normal unit vectors $N = \{\mathbf{n}_j\}$. Relative timestamps are assigned such that the first point produced has timestamp 0 and the last one has 1. Normal unit vectors may be estimated with the unconstrained least square formulation proposed in H. Badino, D. Huber, Y. Park, and T. Kanade, "Fast and accurate computation of surface normals from range images," in ICRA, 2011, taking advantage of box filtering on the point cloud grid structure.

[0066] Odometer

[0067] Initially, one needs to produce a first estimate of the sensor's trajectory by recovering the pose of each point cloud. Since the sensor is moving, one considers as representative pose of the cloud the sensor pose when the last point is received.

[0068] One performs a point-plane ICP between a subset of points of the last received cloud and the map. Like in F. Moosmann and C. Stiller, "Velodyne SLAM," in IVS, 2011, the selected points of the cloud are unwarped by considering the last estimated motion before performing the registration.

[0069] Given the cloud to be registered $C_i$ one considers the last relative motion estimated using the pose of the previous two registered clouds $\mathbf{\Gamma}_{i-1}$, $\mathbf{\Gamma}_{i-2} \in \mathbb{SE}3$:

$$\gamma = \log\left(\mathbf{\Gamma}_{i-2}^{-1} \cdot \mathbf{\Gamma}_{i-1}\right) \in \mathbb{SE}3$$

[0070] where $\gamma$ is expressed in $\mathfrak{se}3$ algebra with the inverse mapping function $\log(\cdot)$ (H. Strasdat, "Local accuracy and global consistency for efficient slam." Ph.D. dissertation, Imperial College London, 2012).

[0071] One then considers the subset of selected points $P_s = \{\mathbf{p}_{s_j}\}, j = 1 : N_S$, with associated normals $N_s = \{\mathbf{n}_{s_j}\}$ and relative timestamps $T_s = \{t_{s_j}\}$. The unwarp is performed on the selected points by computing:

$$\hat{\mathbf{\Gamma}}_i = \mathbf{\Gamma}_{i-1} \cdot \exp(\gamma)$$

$$\mathbf{p}_{s_j} = \hat{\mathbf{\Gamma}}_i^{-1} \cdot \mathbf{\Gamma}_{i-1} \cdot \exp(\gamma * t_{s_j}) \otimes \mathbf{p}_{s_j}$$

$$\mathbf{n}_{s_j} = \mathbf{R}\left(\hat{\mathbf{\Gamma}}_i^{-1} \cdot \mathbf{\Gamma}_{i-1} \cdot \exp(\gamma * t_{s_j})\right) \otimes \mathbf{p}_{s_j}$$

[0072] where $\hat{\mathbf{\Gamma}}_i$ is the predicted pose of the cloud $C_i$ and $\tilde{\mathrm{p}}_{s_j}$ and $\tilde{\mathrm{n}}_{s_j}$ are the selected points in the local coordinate frame of the predicted cloud pose $\hat{\mathbf{\Gamma}}_i . \exp(\cdot)$ maps group $\mathbb{SE}3$ to the algebra $\mathfrak{se}3$.

[0073] Given these elements one performs the registration by estimating the pose $\mathbf{\Gamma}^*_{ODO_i}$ with a point-plane ICP between the unwarped points and normals $\tilde{\mathrm{p}}_{s_j}$ and $\tilde{\mathrm{n}}_{s_j}$ and the map $M$, providing $\hat{\Gamma}_i$ as initial guess.

[0074] Each registered cloud $C_i$ with its associated pose $\mathbf{\Gamma}^*_{ODO_i}$ is added to a list of registered clouds $RC_{ODO}$:

$$RC_{ODO} \leftarrow \left\{RC_{ODO}, \left[C_i, \Gamma^*_{ODO_i}\right]\right\}$$

[0075] Local trajectory optimizer

[0076] This module takes as input the list of clouds with their associated poses $RC_{ODO}$ and performs a trajectory refinement by employing a local window approach. When the distance between the first and the last pose in the list is larger than a threshold, cloud poses are optimized and a new list $RC_{REF} = \left\{\left[C_i, \mathbf{\Gamma}^*_{REF_i}\right]\right\}, i = 1 : N_c$ is produced with the refined pose and the input clouds. Notice that this step properly integrates the unwarping in the optimization.

[0077] The objective function $e(\cdot)$ minimized in this step is the sum of the individual alignment errors of each cloud $e_i(\cdot)$:

$$e\left(RC_{ODO}, \mathbf{\Gamma}_{ODO_0}\right) = \sum_{i=1}^{N_c} e_i\left(C_i, \mathbf{\Gamma}_{ODO_{i-1}}, \mathbf{\Gamma}_{ODO_i}\right) \quad (1)$$

[0078] which, in turn, depends on the pose associated with the first and the last point of the cloud. The initial pose of the first cloud in the sequence, $\mathbf{\Gamma}_{ODO_0}$ is assumed to be the ending pose of the last cloud of the previous optimized set. $e_i(\cdot)$ is computed as the total error of a point-plane ICP generalized on a trajectory defined by the linear interpolation in $\mathbb{SE}3$ between two poses:

$$e_i\left(C, \mathbf{\Gamma}_1, \mathbf{\Gamma}_2\right) = \sum_{j=1}^{N_s} \left[\left(\mathbf{p}_j^W - \mathbf{p}_{NN}\right) \cdot \mathbf{n}_{NN}\right]^2 \quad (2)$$

$$\mathbf{p}_j^W = \Gamma_{12t} \otimes \mathbf{p}_{s_j} \quad (3)$$

$$\Gamma_{12t} = \mathbf{\Gamma}_1 \cdot \exp(t_{s_j} \log(\mathbf{\Gamma}_1^{-1} \cdot \mathbf{\Gamma}_2)) \quad (4)$$

[0079] where $\Gamma_{12t}$ represents the world pose interpolated at time $t_j$ associated with the point $\mathbf{p}_{s_j}$ selected for the registration. Given $\mathbf{p}_{w_j}$, the estimated world coordinates of the current point, $\mathbf{p}_{NN}$ and $\mathbf{n}_{NN}$ are respectively its closest point retrieved from the map and its associated normal.

[0080] The entire objective function is minimized by alternating a Gauss-Newton step and the search for new correspondences in the map, until a convergence criterion is satisfied or a maximum number of iterations is reached.

[0081] The inventors suggest to use the manifold formulation proposed in R. Kuemmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in ICRA, 2011: the optimization is performed over a perturbation vector $\Delta\mathbf{\Gamma}$, composed by element of the $\mathfrak{se}3$ algebra over a pose $\Delta\mathbf{\Gamma}$ in $\mathbb{SE}3$. The composition operation is defined as $\widetilde{\mathbf{\Gamma}} = \exp(\Delta\mathbf{\Gamma}) \otimes \mathbf{\Gamma}$. The Jacobians of the terms in the objective function are evaluated by applying the composition rule as

$$\left. \frac{\partial e_i\left(C, \widetilde{\mathbf{\Gamma}}_1, \widetilde{\mathbf{\Gamma}}_2\right)}{\partial \Delta\mathbf{\Gamma}_1} \right|_{\substack{\Delta\mathbf{\Gamma}_1 = 0 \\ \Delta\mathbf{\Gamma}_2 = 0}},$$

[0082] and similarly for $\frac{\partial e(\cdot)}{\partial \Delta\mathbf{\Gamma}_2}$. Each term $e_i(\cdot)$ in Equation 1 involves a pair of consecutive poses, thus the approximated Hessian results in a block tridiagonal matrix easily tractable by standard algorithms for Cholesky factorization on sparse matrices.

[0083] Once the optimization is terminated, the list $RC_{REF}$ can be updated with the optimized poses. Then, the entire set of points and normals of the clouds are converted into world coordinates according to Equation 3 and then added to the map $M$. At this stage one takes advantage of the efficient strategy to update the local map described in section A.2.: before adding points, one firstly deletes from the map all points that are further than a given radius from the last trajectory pose and then one adds the transformed clouds from $RC_{REF}$. Once the map is updated a new kd-tree is created on

the resulting points to allow subsequent nearest neighbor searches. The list $RC_{ODO}$ is cleared and the odometer guess for the next cloud registration is updated according to the last two poses of $RC_{REF}$. The proposed formulation represents an adherent description of the real sensor model, which acquires points while moving: point transformations in world coordinates involve both initial and final poses of each cloud. Moreover, the estimation of each pose (apart the first and the last) is directly influenced by two clouds.

[0084] Global trajectory optimizer

[0085] The proposed framework can be extended to perform an off-line global optimization of the trajectory. Indeed, a limit of the proposed local trajectory optimizer consists in the inability to refine points (and consequently poses) that have already been added to the map. This limitation is generally acceptable when exploring environments at local scale but, when moving in very large environments, drift can be accumulated. For these cases, global optimization techniques that exploit loop closures or external absolute measurements have to be taken into account.

[0086] The inventors propose a global trajectory optimization that makes use of an enriched map description: for each feature in the map one adds to its position and normal in world coordinates ($\mathbf{p}^W$ and $\mathbf{n}^W$), the original coordinates of the point $\mathbf{p}^L$ and the normal unit vector $\mathbf{n}^L$ in the local sensor reference frame, the relative timestamp $t$ and the index $ID$ of the cloud that originates it. It can be noticed that, given the cloud index and a trajectory, local coordinates of points and normal are redundant information, but the inventors prefer to store them to avoid recomputations.

[0087] The inventors also propose to employ two maps $M_l$ and $M_g$, respectively a local and a global map. $M_l$ is used by the odometer and the local trajectory optimizer modules. When one needs to remove points from $M_l$ one moves them to the global map instead. Moreover, at each step of the local optimizer, the selected correspondences used in the generalized ICP are added to a list $L_c = \{[\mathbf{p}_{q_i}, \mathbf{n}_{q_i}, ID_{q_i}, t_{q_i}], [\mathbf{p}_{NN_i}, \mathbf{n}_{NN_i}, ID_{NN_i}, t_{NN_i}]\}, i = 1 : N_L$, where for each query point $\mathbf{p}_{q_i}$ taken from cloud $ID_{q_i}$ with its associated normal $\mathbf{n}_{q_i}$ and timestamp $t_{q_i}$ one retrieves from $M_l$ data associated to the nearest neighbor used at the last step of the

optimization: its position $\mathbf{p}_{NN_i}$, normal vector $\mathbf{n}_{NN_i}$, cloud index $ID_{NN_i}$ and timestamp $t_{NN_i}$. Note that all the information is in local coordinates of the sensor.

[0088] Having local information in the map is fundamental at this step and memory requirements remain low given that one does not need to store entire clouds, but only the points that are added to the map at each step. It has to be noticed that the list $L_c$ has to be populated after each step of the local optimizer, since addition of new clouds may overwrite old points in the map.

[0089] Similarly, one creates a list of all the poses $L_\Gamma = \{\mathbf{\Gamma}_i\}$ associated to the clouds by stacking the poses refined by the local optimization step. Notice that given $N_c$ clouds, the pose list contains $N_c + 1$ elements. The global trajectory optimization is performed by minimizing

$$e(L_c, L_\Gamma) = \sum_{i=1}^{N_L} \left[ \left( \mathbf{p}_{q_i}^W - \mathbf{p}_{NN_i}^W \right) \mathbf{n}_{NN_i}^W \right]^2 \tag{5}$$

where

$$\mathbf{p}_{q_i}^W = \mathbf{\Gamma}_{q_i}^W \otimes \mathbf{p}_{q_i}$$
$$\mathbf{\Gamma}_{q_i}^W = \mathbf{\Gamma}_{ID_{q_i}} \cdot \exp(t_{q_i} * \log(\mathbf{\Gamma}_{ID_{q_i}-1}^{-1} \cdot \mathbf{\Gamma}_{ID_{q_i}}))$$
$$\mathbf{p}_{NN_i}^W = \mathbf{\Gamma}_{NN_i}^W \otimes \mathbf{p}_{NN_i}$$
$$\mathbf{n}_{NN_i}^W = \mathbf{\Gamma}_{NN_i}^W \otimes \mathbf{n}_{NN_i}$$
$$\mathbf{\Gamma}_{NN_i}^W = \mathbf{\Gamma}_{ID_{NN_i}} \cdot \exp(t_{NN_i} * \log(\mathbf{\Gamma}_{ID_{NN_i}-1}^{-1} \cdot \mathbf{\Gamma}_{ID_{NN_i}}))$$

[0090] The objective function in Equation 5 still represents a generalized point-plane ICP, where both the query and the model point are expressed in local coordinates and transformed into world coordinates with the poses associated to their clouds and the interpolation timestamps.

[0091] Optimizing Equation 5 with Gauss-Newton still results in a sparse approximated Hessian matrix, since each term of the summation involves only three (when $ID_{NN_i} = ID_{q_i} - 1$) or four poses of the entire trajectory, but the matrix is not tridiagonal block, since two points from the same cloud can be associated to points of different clouds. For this reason the inventors employ a graph optimization approach, as

proposed in R. Kuemmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *ICRA*, 2011.

[0092] To reduce computation time it is proposed to never recompute feature associations, assuming that features are properly matched by the local trajectory optimizer. Once the optimization is terminated both the global and the local map are updated by computing the world coordinates of all features.

[0093] This optimization can be applied to a complete sequence of clouds to refine an entire trajectory. Moreover, in presence of loop detections, the correspondences representing the loop allow estimating a trajectory that refines the entire poses, constraining the loop to close correctly.

[0094] Notice however that such global optimization is not suitable for real-time computation, since it involves all the poses and all the associations performed along the entire trajectory.

[0095] Nevertheless it shows that, by retaining the proper information, the present data structure can be employed for global optimization and loop closures. Global trajectory refinement could be performed more efficiently with pose graph optimization solutions, like the one presented in M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3d reconstruction at scale using voxel hashing," *ACM Transactions on Graphics*, 2013, but the ability of maintaining big maps in memory is a key factor to recreate the maps after loops are closed.

[0096] A.4. Experimental results

[0097] The inventors tested the system on real datasets acquired using a Velodyne HDL-32E. A first dataset was acquired by an operator carrying the sensor while exploring an indoor environment of about 10 x 35 x 3 meters. Similarly, a second dataset was acquired in an indoor industrial building of about 16 x 65 x 10 meters. A third dataset was acquired with the sensor mounted on the roof of a car while driving in normal traffic conditions performing four loops in a town district, each one about 500 meters long. Moreover, the inventors evaluated their framework against the publicly available Kitti datasets (H. Strasdat, "Local accuracy and global consistency for efficient slam." Ph.D. dissertation, Imperial College London, 2012) that provides car mounted Velodyne HDL-64E acquisitions taken in various urban environments and at various speeds. The Kitti training datasets also makes available a GPS measured ground truth of each single track. The provided 3D point clouds, though, have been already

unwarped using the estimated motion of the on-board odometry system. For this reason the inventors made use of only those training tracks for which the native raw data was available.

[0098] The local trajectory optimization can be employed to generate high definition local 3D models of the acquired environments. To verify the quality of the generated models, the inventors have processed the two indoor datasets using a voxel resolution of 1 cm with a threshold to trigger the local optimization of 2 m. This results in approximately 8 million of points for the first dataset and approximately 24 million for the second. Then, a reference model has been created by pairwise registering scans of the environment taken with the high resolution ZF 5010C scanner using the method of J. Yao, M. R. Ruggeri, P. Taddei, and V. Sequeira, "Automatic scan registration using 3d linear and planar features," *3D Research*, vol. 1, no. 3, pp. 1–18, 2010. The inventors have accurately registered the two models and computed the point-point distances between them. No visible distortions are present in the models and the histograms of the distances between the two clouds have peaks lower than 0.02m, which is within the nominal accuracy of the Velodyne HDL-32E sensor used.

[0099] To estimate the tracking quality and accumulated drift, the inventors have run the present framework on all Kitti training datasets using as input data the raw readings of the sensor (10 tracks in total). Moreover, to demonstrate the benefit of incorporating the sensor motion in the optimization framework, they have also run the present system on the same tracks but employing the official preprocessed clouds of the datasets (unwarped using the estimated motion of the on-board odometry system). In this case the inventors did not perform any unwarp during the optimization (i.e., they used only the odometry module). For these experiments they used a voxel size of 15 cm in the maps and they did not perform loop closures. Figure 2 shows both experiment results in terms of average relative translation and rotation error generated using trajectory segments of 100m, 200 m, ..., 800 m length (refer to H. Strasdat, "Local accuracy and global consistency for efficient slam." Ph.D. dissertation, Imperial College London, 2012 for a description of the adopted error metric). It is clear that by incorporating the cloud unwarping into the optimization framework yields better results and reduces both translational and rotational drift (in particular translation error improved by 0.3 point percentage on average). Notice that the current state of the art algorithm for the Kitti benchmark that only employs LIDAR data (LOAM) performs better. It must be noted

though that it has been validated directly on the original unwarped point clouds and that it processes clouds only at 1Hz.

[00100] To evaluate the improvements introduced by the proposed global optimization strategy after integrating loop closures, the inventors enabled this feature over a sample track of the Kitti dataset containing a single loop. Their loop detection mechanism is very simple and not suitable for a real application: one detects a loop when the distance between the current pose and a previous pose far in time is lower than a threshold. Then, one registers the last cloud on the global map and if this succeeds, one adds the found correspondences to the global optimization. Figure 3 shows the effect of the loop closure in the considered track. Experimental results also showed the improvement of the global optimization over both the odometer and the local optimized trajectory.

[00101] The inventors compared their system with the publicly available Velodyne SLAM [F. Moosmann and C. Stiller, "Velodyne SLAM," in IVS, 2011] that also performs a motion compensation on the acquired point clouds. To compare the two systems the inventors measured drift accumulated using the outdoor car dataset. Since the same location is revisited multiple times, they estimated drift by registering the generated initial local map with the one generated at each subsequent passage. The translation and orientation components of the registration transform aligning the current local map to the initial one indicate how much drift has been accumulated. One of the salient characteristics of [F. Moosmann and C. Stiller, "Velodyne SLAM," in *IVS*, 2011] is the presence of a map refinement strategy (called *adaption)* based on a set of heuristic tests that positively influence the trajectory estimation. Since the present system is focused on the optimization strategy by proper modeling the problem, the inventors disabled this feature in the original work to focus the analysis on the trajectory estimation. Results after each loop are shown in Table I. It can be noticed that one accumulates less drift than the original work. Moreover the present system is a natural formulation of the problem that requires less configuration parameters than the heuristic strategies of the Velodyne SLAM. Performance of the present system is superior to the Velodyne SLAM system both in terms of execution time and in the ability of maintaining a global map of the environment, while in the original work only a local map is maintained. The ability of using the global map has been confirmed, in case of the use of loop closure and the global optimization technique to correct the drift accumulated in the first loop and the use of the global map for the next loops.

[00102] In order to evaluate the performance of the proposed map representation, the inventors have measured the execution time of each operation while running the outdoor car dataset on a PC equipped with an Intel Xeon E5-2650 CPU.

[00103] As expected, addition operations are performed in a linear time w.r.t. the number of features added to the map, being the average time 36.4ns per feature, which gives an average cloud insertion time of 1.81ms for the HDL-32E sensor.

[00104] Delete operations in the present SLAM framework are only performed over the local map, just before updating the kd-tree.

[00105] TABLE I: Drift error reports

| loop | yaw | pitch | roll | dx | dy | dz |
|---|---|---|---|---|---|---|
| Local trajectory optimizer | | | | | | |
| 1st | -1.4° | -0.7° | -0.0° | -0.62m | -0.26m | 0.39m |
| 2nd | -2.7° | -0.3° | -0.1° | -1.16m | -0.86m | 0.89m |
| 3rd | -4.2° | -0.4° | -0.7° | -1.17m | -1.16m | 1.80m |
| 4th | -5.5° | -0.8° | -1.0° | -2.37m | -1.45m | 2.33m |
| Velodyne SLAM | | | | | | |
| 1st | 3.33° | 0.05° | -0.9° | 1.53m | 0.80m | 3.60m |
| 2nd | 6.54° | 0.3° | -1.7° | 2.97m | 1.82m | 7.29m |
| 3nd | 9.96° | 0.5° | -2.5° | 4.54m | 2.87m | 11.04m |
| 4nd | 13.2° | 0.9° | -3.0° | 5.93m | 4.16m | 14.54m |

[00106] Features to be deleted are selected by performing a radius search around the point of interest (e.g. the last estimated sensor pose) and added to the global map. Results show a constant deletion time per feature that takes on average 30.84ns.

[00107] Selection of features to be deleted from the local map can be performed in two manners: by using the voxel structure or by iterating over the dense list. Figure 4(a) shows the average search times based on the number of features stored in the map and the search radius. As it can be noticed, using the dense list always provides the same performance (linear to the number of features stored, independently of the search radius). On the other hand, voxel search times increase as the radius does and, in all the cases, present worst results.

[00108] Since no points are deleted from the global map, *compact* operations only happen in the local one. Thanks to the proposed strategy of postponing the addition of new clouds until a new kd-tree is requested, only 7.79% of the times the number of holes created is greater than the number of features added, being necessary to perform a *compact* operation. In these cases, execution times show a linear behavior w.r.t. the number of holes remaining, being the average time of each operation 1.81ms.

[00109] Finally, for loop closure operations, the global map has to be queried around an interest area. As happened with the local map, this selection can be performed in two manners. Figure 4(b) shows the results of using the voxel structure and the dense list. As it can be noticed, for search radius under 90 meters, the voxel over-performs the dense list. However, as the radius grows, caching fails in the internal sparse voxel structure lead to a great performance loss.

[00110] The system is able to process clouds at 12.93Hz (i.e., in real time w.r.t. the Velodyne acquisition rate) when the local trajectory optimization is not active, while the frequency decreases to 7.5Hz using the local trajectory optimization, which is close to real time. It has to be noticed that the registration and the local optimization are not coded to run in multi-thread, thus the inventors expect that performance can be increased both in the odometer and in the local optimization.

[00111] In the odometer mode the time spent in registering clouds is the 54% of the total, while in the local optimization mode 30% of the time is spent for the odometer registration and 35% for the local trajectory optimization. The registration includes the nearest neighbor search time, while the impact of each operation performed over the local and global maps is summarized in Table II, when working on odometer mode (first row) and when performing the local trajectory optimization (second row). Addition, deletion and compact operations on the local map are shown in columns *add, delete* and *compact*, respectively, where deletion times also include the point selection over the local map and the addition to the global map. The impact of building the kd-tree over the entire local map is shown in the column *kd-tree* and, finally, the impact of adding the deleted points of the local map into the global map is shown in the column *add g.*

[00112] TABLE II: System performance

| | freq. | add | delete | compact | kd-tree | add g. |
|---|---|---|---|---|---|---|
| Odom. | 12.93Hz | 2.1% | 0.9% | 0.2% | 25.0% | 0.3% |
| Local | 7.54Hz | 1.2% | 0.5% | 0.1% | 13.7% | 0.2% |

[00113] A.5. Conclusion

[00114] The present document presents a framework for local optimization of point clouds acquired using moving lasers. In particular the inventors incorporated the acquisition motion into the optimization by interpolating each acquired point cloud between its starting and ending position. The inventors experimentally showed, using publicly available datasets, that by correctly modelling the sensor movement it is possible to reduce odometry estimation errors.

[00115] Moreover, they present an efficient data structure to manage large voxelized 3D maps constituted by sparse features. The map data structure is suited for both local map optimization and for offline global optimization. Their experiments show that, for the former problem, such a structure provides real-time odometry and nearly real time local refinement. These performances may even be enhanced by taking advantage of multi-thread operations when local trajectory optimization is performed (e.g., nearest neighbor search, cloud unwarping).

[00116] B. Localization, tracking and change analysis based on 3D reference model

[00117] Approaches based on octrees or kd-trees provide reasonable searching times for nearest neighbors (typically logarithmic w.r.t. the map size) and good scalability. In their approach the inventors introduce an alternative voxel representation that combines the fast random accesses provided by dense voxel representations and the scalability provided by sparse data structures.

[00118] In order to ensure a correct pose tracking, a preferred system performs an efficient selection of points to be used in the registration process that ensures good geometric stability for the ICP algorithm. Then, a strategy to efficiently discard outliers ensures that registration is performed only using correspondences that are globally consistent (inliers).

[00119] The present preferred framework fuses in the registration process w.r.t. the ground truth model a robust odometer that is capable of real time tracking even when the user leaves the map or if the observed environment differs too much from the initially acquired model (e.g. furniture were changed). By re-entering the known map the system automatically recovers the correct position and thus avoids drift accumulation.

[00120] B.1. Main benefits of the preferred embodiments described below

1) A scalable place recognition strategy to localize the sensor in very large environments using a set of pre-computed descriptors and avoiding accesses to the ground truth map.
2) An efficient data structure to represent the map that provides constant time nearest neighbor searches and a low memory footprint.
3) A fast point selection strategy that ensures geometrically stable results.
4) An inlier selection technique that efficiently removes the interference of outliers during the registration process.
5) A fusion between a local odometer and the registration against the ground truth map that exploits static outliers and allows the user to navigate through non-mapped areas.
6) A complete system that provides real-time results with high accuracy.

[00121] The description below is structured as follows: Section B.2. presents a preferred online place recognition and relocalization strategy, Section B.3. shows how to perform online tracking once the user pose has been identified in a known environment. Then Section B.4. presents experimental results and finally Section B.5. draws the conclusions.

[00122] B. 2. Place Recognition

[00123] The place recognition component deals with recovering an initial estimate of the user location and orientation without a priori information. It is able to run online at frame rate to provide candidate locations given the current sensor observation. Moreover, for scalability purposes, it should not make use of the map model during execution since it might provide candidate poses related to distant locations (and thus not loaded in memory), or even different maps. In order to satisfy these two requirements, a pre-processing stage is preferably introduced in order to (1) reduce the search space of available poses and (2) train a robust and compact classifier that, given an observation, efficiently estimates the possibility of being in a specific location.

[00124] Search Space Reduction

[00125] One initially preferably detects navigable areas amongst the entire map. These areas are defined as the volume where the sensor can be placed during the exploration

of the environment. Moreover one may generally assume without loss of generality that the map model $Z$ axis is roughly aligned with the gravity vector.

[00126] Since the inventors focused on ground motion (backpack or vehicle mounted sensor), navigable areas are expected to be in a relatively narrow space over the navigable floor. For this reason, one firstly identifies the extents of the floor. Floor extraction is performed over a sparse voxel representation of the environment, $V$, where each full cell, $v^{(i)}$, contains a normal vector to the surface locally defined by the points around its centroid, $\bar{\mathbf{n}}^{(i)}$. One extracts a subset of voxels that represent candidate floor cells, $F \subseteq V$, by checking that the vertical component in their associated normals is dominant, i.e. $\bar{\mathbf{n}}^{(i)} \cdot (0,0,1)^T \geq \varepsilon$, where $\varepsilon$ is typically a value between 0.5 and 1. However, this constraint alone may lead to classifying too many cells as floor (e.g. tables or empty shelves).

[00127] To address this problem, the inventors propose to introduce the concept of reachability. Given a reachable cell $f \in F$, all surrounding cells $(g^{(1)}, g^{(2)}, \ldots, g^{(m)}) \in F$ are considered as reachable if the following conditions are satisfied:

$$\left\| f - g^{(i)} \right\| \leq \theta_0 \tag{6}$$

$$\left\| f_Z - g_Z^{(i)} \right\| \leq \theta_1 \tag{7}$$

$$C_{g^{(i)}} \cap V = \varnothing \tag{8}$$

[00128] where $\theta_0 \geq V_{\text{cellSize}}$ in (6) stands for the maximum step distance (e.g. 0.5 meters for a walking motion, or $V_{\text{cellSize}}$ for a car motion), $\theta_1$ in (7) stands for the maximum vertical step size and $C_{g^{(i)}}$ in (8) stands for the simplified volume of the observer, centered over the floor cell $g_i$ (a bounding cylinder in the present implementation).

[00129] Initial reachable cells can be provided manually but, since the generation of the map is preferably performed by placing the scanner over reachable cells, this initialization can be automatically performed assuming floor cells below the acquisition positions as reachable.

[00130] According to these conditions, detecting all floor cells $F^* \subseteq F$ is performed in a flooding-algorithm style, as illustrated in Table III showing algorithm where, initially, $A$ stores the first set of reachable cells.

**Table III:** Flooding floor extraction.

**Require:** $A \neq \emptyset$, $F \neq \emptyset$, $F \cap A = \emptyset$
$F^* \leftarrow \emptyset$
**While** $A \neq \emptyset$ **do**
  $B \leftarrow \emptyset$
  **While** $A \neq \emptyset$ **do**
    $a \leftarrow A$.pop()
    $F^*$.push($a$)
    **For all** $f \in F$ **do**
      $t_1 \leftarrow \|f - a\| \leq \theta_0$
      $t_2 \leftarrow \|f_Z - a_Z\| \leq \theta_1$
      $t_3 \leftarrow C_f \cap V = \emptyset$
      **If** $t_1$ **and** $t_2$ **and** $t_3$
        $F$.remove($f$)
        $B$.push($f$)
      **End if**
    **End for**
  **End while**
  $A \leftarrow B$
**End while**
**Return** $F^*$

[00131] Once the floor has been identified, navigable space, $N$, is defined as the set of cells, $n^{(i)} \in N$, above floor cells where $n^{(i)} \cap V = \emptyset$.

[00132] In order to further reduce the navigable space without loss of precision, the inventors also propose to introduce physical constraints related to particular operability of the system (e.g. vertical and angular limits on the possible sensor pose for a specific sensor mounting) that provides an effective navigable space $N^* \subseteq N$. Such constraints are empirically selected by running a set of experiments on sample datasets (see Section B.4.).

[00133] Pose classifier

[00134] In order to build a pose classifier one initially needs to define a compact representation of each single observation. In particular the inventors adopt the simple and fast-to-compute compact descriptor defined by Taddei, P., Sánchez, C., Rodríguez, A. L., Ceriani, S., Sequeira, V., 2014. Detecting ambiguity in localization problems using depth sensors. In: 3DV: one splits the range image in $W_b \times H_b$ regular bins and, for each one, one estimates a median range value. All these values are stacked in a descriptor of the observed frame $\mathbf{d}$.

[00135] One then randomly generates a set of training poses $\Gamma_T = \{\mathbf{\Gamma}_{T_0},\ldots,\mathbf{\Gamma}_{T_i},\ldots,\mathbf{\Gamma}_{N_T}\}$ in the known effective navigable space $N^* \subseteq N$. For each pose $\mathbf{\Gamma}_{T_i}$ one synthesizes a depth image by ray-casting the 3D map to a sensor image plane aligned to the provided pose and the inventors extract its descriptor $\mathbf{d}_{T_i}$ from the generated depth image. One builds a kd-tree $T = \{\mathbf{d}_{T_i} \rightarrow \mathbf{\Gamma}_{T_i}\}$ that maps all generated descriptors $\mathbf{d}_{T_i}$ to their corresponding pose $\mathbf{\Gamma}_{T_i}$. Given a descriptor $\mathbf{q}$, the set of location/descriptor pairs that are close in the descriptor space can be retrieved by performing efficient searches on $T$, with logarithmic complexity in the descriptor space. Notice that, given the set of training samples $\{\mathbf{d}_{T_i} \rightarrow \mathbf{\Gamma}_{T_i}\}$, it is also possible to build more compact classifiers, e.g. as described in Glocker, B., Izadi, S., Shotton, J., Criminisi, A., 2013. Real-time rgb-d camera relocalization. In: ISMAR. Nevertheless the inventors experimentally observed that $N^*$ was small enough to retain the full training set in memory and to perform classification by radial nearest neighbor searches in the descriptor space of the kd-tree.

[00136] During execution the pose classifier is used to recover the most possible locations given the current observation. In particular, the inventors split the process in two different stages: the initialization, which deals to the estimation of possible locations when no a priori information is available and the update, which deals with the evaluation of candidate locations and the resampling of them.

[00137] In the initialization step one needs to draw a set of possible location of the sensor given a single sensor observation. The inventors propose to proceed as follows:

[00138] 1. Given the last sensor observation one computes its associated descriptor $\mathbf{q}$ and recovers a set of candidate locations $\Gamma$ performing a radial search on $T$ given a threshold $r$ in the descriptor space. In case of sensors providing 360 horizontal field of view, one may increase the candidate locations by computing additional input descriptors by horizontally shifting the range values. Each descriptor corresponds to the readings that the sensor would produce if rotated by on its local axis. Each resulting set of candidate locations are then rotated according to $i$.

[00139] 2. One associates a weight $\mathbf{w}_{\Gamma_p}$ to each potential location $\mathbf{\Gamma}_p \in \Gamma$:

$$\mathbf{w}_{\Gamma_p} = 1 - \frac{\|\mathbf{d}_p - \mathbf{q}\|}{r},$$

[00140] where $\mathbf{d}_p$ is the descriptor associated to the location $\mathbf{\Gamma}_p$ retrieved from $T$. $\mathbf{w}_{\mathbf{\Gamma}_p}$ is 1 for perfectly matching descriptors and 0 for descriptors on the search sphere boundary.

[00141] 3. Finally, weights are collected in $\mathbf{w}$ and normalized to have $\max \mathbf{w} = 1$.

[00142] The update stage deals with the update of the possible locations $\Gamma = \mathbf{\Gamma}_0, \ldots, \mathbf{\Gamma}_N$ while the sensor moves given their associated weights $\mathbf{w} = \mathbf{w}_0, \ldots, \mathbf{w}_N$. Notice that this step makes use of an odometer that registers one cloud to its predecessor according to the technique explained in the next section. In particular the inventors proceed as follows:

[00143] 1. One uses the odometer and the current observation to update all locations in $\Gamma$.

[00144] 2. When a given distance is travelled since last potential locations were created, a new descriptor $\mathbf{q}$ is computed from the last observation. This is is used to retrieve from $T$ a set of possible locations $\hat{\Gamma}$, similarly to step 1 in the initialization stage.

[00145] 3. The weight associated to each possible location $\hat{\mathbf{\Gamma}}_j \in \hat{\Gamma}$ is computed as:

$$\mathbf{w}_{\hat{\mathbf{\Gamma}}_j} = \frac{\Pi(\mathbf{q} \mid \hat{\mathbf{\Gamma}}_j)\, \Pi(\hat{\mathbf{\Gamma}}_j)}{\Pi(\mathbf{q})}. \qquad (9)$$

[00146] and once all weights have been computed, they are normalized to have a maximum value of 1.

[00147] 4. One updates $\Gamma = \hat{\Gamma}$ and $\mathbf{w} = \mathbf{w}$ and repeats the iteration of the update stage.

[00148] Equation (9) computes the weight associated to each possible location using the Bayes theorem expressed in possibility theory alike in Dubois, D., 2006. Possibility theory and statistical reasoning. Computational Statistics and Data Analysis 51 (1), 47 – 69, the Fuzzy Approach to Statistical Analysis. Individual terms of (9) are:

$$\Pi(\mathbf{q} \mid \hat{\mathbf{\Gamma}}_j) = 1 - \frac{\left\| \mathbf{d}_j - \mathbf{q} \right\|}{r} \qquad (10)$$

$$\Pi(\hat{\mathbf{\Gamma}}_j) = \max_{\substack{k=1,..N \\ d(\mathbf{\Gamma}_k, \hat{\mathbf{\Gamma}}_j) \leq 1}} \mathbf{w}_k * \left(1 - d(\mathbf{\Gamma}_k, \hat{\mathbf{\Gamma}}_j)\right) \tag{11}$$

$$d(\mathbf{\Gamma}_k, \hat{\mathbf{\Gamma}}_j) = \frac{\left\| \mathbf{\Gamma}_k - \hat{\mathbf{\Gamma}}_j \right\|}{d_{\max}} \tag{12}$$

$$\Pi(\mathbf{q}) = \frac{\hat{N}}{N_T} \tag{13}$$

[00149] Equation (10) estimates the possibility of the descriptor $\mathbf{q}$, given the pose $\hat{\mathbf{\Gamma}}_j$ in the same way as in step 2 of the initialization stage (In case of multiple input descriptors , each must be taken into account individually). Equation (11) evaluates the likelihood of being at pose $\hat{\mathbf{\Gamma}}_j$ by finding the most compatible location in the set of potential locations $\Gamma$. This compatibility is defined as the weighted relative distance (Equation (12)) between the previous potential pose $\mathbf{\Gamma}_k$ and pose $\hat{\mathbf{\Gamma}}_j$. Equation (13) estimates the distinctiveness of the current observation by comparing the number of neighbors retrieved w.r.t. the size of the training set, e.g. extremely ambiguous poses like in corridors will produce lots of results, resulting in high ambiguity.

[00150] The update stage is iterated until potential poses converge to a single location, i.e. when the covariance of the centroid of $\Gamma$ computed according to weights $\mathbf{w}$ is small. At this point one considers the problem solved and the pose tracking component is started.

[00151] The preferred system outlined above is based on an iterative re-weighting of possible locations with fast bootstrapping that uses a single sensor observation. A key factor for scalability to large maps is the pre-computation of lightweight descriptors from the reference maps and their organization in a kd-tree structure with associated poses. This way, queries in the descriptor space are used to efficiently populate the system with candidate locations given the first observation. Then, in subsequent update steps the estimated motion and queries in the descriptor space are used to draw a new set of possible locations and their associated weights.

[00152] This approach is comparable with the general Monte Carlo Localization techniques presented in [Thrun, S., Fox, D., Burgard, W., Dellaert, F., 2001. Robust monte carlo localization for mobile robots. Artificial intelligence 128 (1), 99–141] and [Thrun et al. (2005) Thrun, Burgard, and Fox] that make use of particle filters. However

their techniques aim at precisely estimating the sensor probability distribution by approximating it with a set of weighted particles in order to solve all stages of the localization problem [Thrun, S., Burgard, W., Fox, D., 2005. Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). The MIT Press].

[00153] The present place recognition component, instead, only needs a fast and rough pose estimation, since precise pose tracking is performed by the subsequent tracking component (Section B.3.) once a unique location has been identified. Moreover, the present system only has to ensure that possible locations are not discarded and thus does not require a precise sensor pose probability density estimation. For this reason, one does not require a dense sampling of the navigable space, as Section B.4. shows. However a low sampling density may lead to tracking loss in certain cases due to wrong particle initialization. This problem is overcome by drawing a new set of particles each time the update stage is performed.

[00154] B.3. Pose Tracking

[00155] The pose tracking component deals with computing the local motion of the sensor as it moves around the environment. Knowing the previous estimated pose, when a new acquisition is received, the inventors perform a local registration between the map and the observed points. From the resulting transformation, the implicit motion is inferred and applied to the previously estimated pose.

[00156] To accurately track the sensor pose in real-time, it is important (1) to employ a data structure specifically designed for nearest neighbour searches and (2) to correctly select a stable and representative subset of the input points to perform the registration. Nevertheless, in order to ensure correct estimations, (3) outliers have to be properly detected. This is particularly important in degenerate environments which contains few large dominant directions, e.g. long corridors, tunnels or symmetrical environment where there are few proper points to hinder erroneous registrations.

[00157] Map Representation

[00158] In the presently proposed map structure, two different lists of elements are stored and synchronized: a compact list of planes, $L$, and a dense grid of voxels, $V$, built with a specific voxel size. Each plane $l_i \in L$ stores a position in world coordinates, $\mathbf{p_i}$, and a unit normal, $\overline{\mathbf{n_i}}$. Each voxel, $v_i \in V$ stores a current state that can be either full, empty or near. Full voxels store an index to the plane $l_{v_i} \in L$, whose associated

position falls into. In particular reference map points belonging to the voxel are used to estimate the plane parameters. Empty cells store a null reference and near cells store an index to the plane $l_{v_j} \in L$ whose associated position distance $d_v$ to the voxel centre is the smallest. Notice that the inventors preferably consider near voxel only if the distance $d_v$ is under a given threshold $d_{max}$, otherwise the voxel is considered empty. Figure 5 illustrates the proposed representation.

[00159] With this map representation, all nearest neighbour searches are pre-computed offline and stored inside the dense grid. At run time, given a query point in world coordinates, the inventors approximate the computation of its nearest neighbour in the map by calculating the voxel that contains it. Then, if the cell state is *full* or *near*, one returns the associated plane. Otherwise, one notifies that there are no neighbors.

[00160] Notice that, for the proposed approach (1) all operations performed during a single search present a constant execution time, regardless of the size of the map. In comparison kd-tree structures provide, on average, logarithmic times w.r.t. the size of the map. Moreover, by properly setting $d_{max}$ one (2) implicitly performs an initial outlier rejection of correspondences that are too separated when looking for nearest neighbours in ICP.

[00161] The main disadvantage of using dense voxel structures for representing large environments consists in their memory footprint. The inventors solve this problem using a three-level hierarchical structure where intermediate nodes are blocks of $32 \times 32 \times 32$ nodes. This way, when a node is completely empty, it does not have to be stored and, given the proposed leaf size, $2^5 \times 2^5 \times 2^5$, one can address each single internal cells using only two bytes, plus an extra bit to mark empty ones ($15+1$ bits). Additionally, the present implementations allow streaming so that only the part inside the range of the sensor has to be in memory. Since the sensor moving speed is orders of magnitude below the associated load operations, on-line execution memory requirements are always bounded and the map is always updated around the sensor pose.

[00162] Point Selection Strategy

[00163] One should ensure that the selected subset of points of the current acquisition is representative enough to correctly lock the less defined degrees of freedom during the registration. Similarly to the work described in [Gelfand et al. (2003) Gelfand, Ikemoto, Rusinkiewicz, and Levoy], the inventors consider the contribution of moving a

point, $\mathbf{p}_i$, and its associated normal, $\bar{\mathbf{n}}_i$, by a transformation vector $[\Delta\mathbf{r}^T \Delta\mathbf{t}^T]$ on the point-plane distance. This can be expressed as:

$$\Delta d_i = \left[\Delta\mathbf{r}^T \Delta\mathbf{t}^T\right] \cdot \begin{bmatrix} \mathbf{p}_i \times \bar{\mathbf{n}}_i \\ \bar{\mathbf{n}}_i \end{bmatrix} \quad (14)$$

by linearising rotations using the small angles approximation.

[00164] Considering only rotations in Equation (14), the error introduced in the point-plane distance is proportional to the point distance w.r.t. the sensor and to the angle between its normal and the viewing ray. This leads to selecting far points and points whose normal is as perpendicular as possible w.r.t. the viewing ray. Unfortunately moving laser produces non uniformly distributed points and, in particular, distant areas are acquired with a lower point density and thus provide poorly estimated normals. Also, for circular environments when the sensor approaches the symmetry axis, angles between viewing rays and normals vanishes.

[00165] The inventors preferably solve these problems by explicitly distinguish between translations and rotations. In order to properly constrain translations, they consider only point normals. They compute the covariance matrix for translations $C_t$ as:

$$C_t = \left[\bar{\mathbf{n}}_1 \quad \cdots \quad \bar{\mathbf{n}}_k\right] \cdot \begin{bmatrix} \bar{\mathbf{n}}_1^T \\ \cdots \\ \bar{\mathbf{n}}_k^T \end{bmatrix}$$

[00166] and extract its associated eigenvectors $\bar{\mathbf{x}}_1$, $\bar{\mathbf{x}}_2$, $\bar{\mathbf{x}}_3$, and eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_3$. Acquisition points are then classified into three bins, $\{b_1, b_2, b_3\}$ as follows:

$$\mathbf{p}_i \in b_j \leftrightarrow \left|\mathbf{p}_i \cdot \bar{\mathbf{x}}_j\right| > \left|\mathbf{p}_i \cdot \bar{\mathbf{x}}_k\right|, \quad \forall k \neq j$$

[00167] When the three bins are balanced, the translation degrees of freedom are equally constrained. On the other hand, in degenerate cases, e.g. long corridors, one bin will be considerably less populated than the others, e.g. the one containing the points whose associated normals are parallel to the longitudinal axis.

[00168] W.r.t. orientations, one computes the principal rotation axes using cross products between positions and normals. The resulting covariance matrix is defined as follows:

$$C_R = \begin{bmatrix} \mathbf{p}_1 \times \bar{\mathbf{n}}_1 & \cdots & \mathbf{p}_k \times \bar{\mathbf{n}}_k \end{bmatrix} \cdot \begin{bmatrix} \mathbf{p}_1 \times \bar{\mathbf{n}}_1^T \\ \cdots \\ \mathbf{p}_k \times \bar{\mathbf{n}}_k^T \end{bmatrix}$$

[00169] Similarly to translations, one calculates the associated eigenvectors $\bar{\mathbf{x}}_1$, $\bar{\mathbf{x}}_2$, $\bar{\mathbf{x}}_3$, and eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_3$. Then, points from the input cloud are classified into three bins, $\{b_1, b_2, b_3\}$ as follows:

$$\mathbf{p}_i \in b_j \leftrightarrow \left| \bar{\mathbf{n}}_i \cdot \bar{\mathbf{x}}_j \right| < \left| \bar{\mathbf{n}}_i \cdot \bar{\mathbf{x}}_k \right|, \quad \forall k \neq j$$

[00170] For each bin, one approximates the rotation centre as the weighted mean of the contained positions, according to their distance to the sensor (This approximation is valid for sensors. For other fields of view an alternative approximation may be required):

$$\mathbf{c} = \frac{\sum_{i=1}^{n} \|\mathbf{p}_i\| \cdot \mathbf{p}_i}{\sum_{i=1}^{n} \|\mathbf{p}_i\|}$$

[00171] and, then, for each point in the bin, one estimates how much it contributes on locking rotations over its corresponding eigenvector, $\mathbf{x}$, as:

$$d_i = \left(1 - \left| \mathbf{x} \cdot \bar{\mathbf{n}}_i \right|\right) \cdot \frac{(\mathbf{c} - \mathbf{p}_i) \cdot \left(\mathbf{x} \times \bar{\mathbf{n}}_i\right)}{\|\mathbf{p}_i - \mathbf{c}\|} \qquad (15)$$

[00172] First term in Equation (15) weights the influence of a given point normal according to its perpendicularity to the rotation axis (the more perpendicular the higher the weight). The second term numerator estimates the quality on locking rotations over $\mathbf{x}$ by computing the angle between the vector connecting the centre of rotation to the point, and the vector perpendicular to the plane defined by the normal and the rotation axis. Finally, the denominator normalizes the result in the range $[0..1]$, so point selection is independent from the distance to the sensor.

[00173] When bins associated with small $d$ values contain too many points, rotations around the axis considered are poorly constrained: one needs to select only the points with the highest values. Figure 6 illustrates this concept showing sample histograms of $d$ values recovered from an environment with high symmetry and from another one where rotations are properly defined.

[00174] Registration and Inlier Selection

[00175] For registration purposes, the inventors consider as wrong correspondences those between sensor points (in world coordinates), $\mathbf{p}_i^W$, and map points, $\mathbf{q}_i$, that are inconsistent with the rest of the correspondences. This occurs when: (a) the point seen by the sensor corresponds to an object that is not present in the map (i.e. something that was added or removed after the original acquisition) or (b) the estimated $\mathbf{p}_i^W$ is far from its corresponding point in the map. In both cases, the nearest neighbour does not make geometrical sense w.r.t. the other correspondences. Classical ways to identify these outliers employ heuristics based on relative positions and normals between corresponding points: neighbours whose distance is larger than a given threshold or with very different normal orientations are considered outliers. Examples can be found in [Rusinkiewicz and Levoy (2001)]. These approximations are useful when using high tolerance values (e.g. corresponding points further than 5 may be wrong in most cases) but, in these cases, their discriminative power is low.

[00176] The inventors initially consider the bins related to translations described above. Then they evaluate if rotations are properly defined over all the axes. If this is not the case for a given axis, they add a bin containing the points that better constrain such rotation, i.e. points with largest $d_i$ values.

[00177] Then, they consider the last estimated motion (using the two previously registered poses) to perform an initial guess on the new sensor pose:

$$\hat{\mathbf{\Gamma}}_t = \mathbf{\Gamma}_{t-1} \cdot \left( \mathbf{\Gamma}_{t-2}^{-1} \cdot \mathbf{\Gamma}_{t-1} \right) \in \mathbb{SE}3$$

[00178] Starting from this guess, each iteration of the ICP algorithm creates $n$ random sets of points, $S$, where each set $s^{(j)} \in S$ contains $k$ randomly selected points from each bin (typically $k=1$). For each one of these points, one computes the associated position in world coordinates, $\mathbf{p}_i^W$, using $\hat{\mathbf{\Gamma}}_t$ and its corresponding nearest plane in the map, $\{\mathbf{q}_i, \bar{\mathbf{n}}_i\}$, is searched, creating the correspondence $c_i^{(j)} = \{\mathbf{p}_i^W, \mathbf{q}_i, \bar{\mathbf{n}}_i\} \in s^{(j)}$ Once all correspondences in each set are solved, the rigid transformation $\mathbf{T}^{(j)} = [\mathbf{R}^{(j)} \mathbf{t}^{(j)}]$ that minimizes the expression

$$E^{(j)} = \sum_{i=i}^{n} \left( \left( \mathbf{R}^{(j)} \mathbf{p}_i + \mathbf{t}^{(j)} - \mathbf{q}_i \right) \cdot \bar{\mathbf{n}}_i \right)^2$$

is computed for each of them independently.

[00179] Considering that correspondences of each set are defined over observed points that properly lock on all six degrees of freedom, their associated rigid transformations are expected to be similar. However, in the presence of outliers and considering the reduced number of points for each set, resulting transformations will be randomly different. One may approximate the estimation error with a gaussian distribution and identify outlier correspondences by removing the sets that diverge from such distribution. One proceeds iteratively by initally considering all transformations and computing the associated normal distribution $N(\mu,\Sigma)$ where:

$$\mu = \frac{1}{n}\sum_{j=1}^{n}\gamma^{(j)}$$

$$\Sigma = \frac{1}{n}\left[\gamma^{(1)}-\mu \quad \cdots \quad \gamma^{(n)}-\mu\right]\cdot\begin{bmatrix}(\gamma^{(1)}-\mu)^T\\ \cdots \\ (\gamma^{(n)}-\mu)^T\end{bmatrix}$$

[00180] being $\gamma^{(n)}$ the rigid transformations associated with each set expressed as a vector, where rotations are in yaw, pitch, roll angles. Then, according to $N(\mu,\Sigma)$ mahalanobis distances for each set are computed as

$$d^{(j)} = (\gamma^{(j)}-\mu)^T\Sigma^{-1}(\gamma^{(j)}-\mu) : \chi_6^2$$

[00181] and transformations with an associated probability smaller than 1% are discarded. This process is iteratively repeated (updating $N(\mu,\Sigma)$ with the remaining transformations at each step) until no transformations are discarded, or a minimum number of inlier transformations is reached. The final registration is estimated considering only the correspondences present in the sets associated with the remaining transformations.

[00182] Figure 7 shows the results after the proposed inlier selection strategy. Notice how all independently computed transformations are distributed around a well defined central position. Also notice that, after each iteration of outlier removal, the distributions quickly converge to the final estimated transformation, when considering all the correspondences marked as inliers.

[00183] Odometer integration

[00184] To improve the overall system robustness, the inventors preferably combine their proposed sensor tracking component with an odometer.

[00185] After a pose has been estimated, its associated points in world coordinates are stored into a kd-tree. Given a new acquisition, when the registration algorithm creates the sets of points ($\mathbf{p}_i^W$), it looks for nearest neighbours in both the reference map ($\mathbf{q}_i^M, \bar{\mathbf{n}}_i^M$) and in the previously fixed cloud ($\mathbf{q}_i^O, \bar{\mathbf{n}}_i^O$). Correspondences are, then, defined as:

$$c_i^{(j)} =$$

$$\begin{cases} \{\mathbf{p}_i^W, \mathbf{q}_i^M, \bar{\mathbf{n}}_i^M\} & \left\|\mathbf{p}_i^W - \mathbf{q}_i^M\right\| - s \leq \left\|\mathbf{p}_i^O - \mathbf{q}_i^O\right\| \\ \{\mathbf{p}_i^W, \mathbf{q}_i^O, \bar{\mathbf{n}}_i^O\} & \left\|\mathbf{p}_i^W - \mathbf{q}_i^M\right\| - s > \left\|\mathbf{p}_i^O - \mathbf{q}_i^O\right\| \end{cases}$$

[00186] where $s$ corresponds to the voxel cell size and compensates the different resolution between the voxelized ground truth map and the non-discretized kd-tree of the previously fixed cloud.

[00187] Main benefits are that (a) surfaces missing in the reference map can be exploited during the registration process and that (b) the system allows exploring non-mapped areas by continuously tracking the user.

[00188] B.4. Results

[00189] In order to evaluate the proposed localization and tracking system, the inventors ran several tests using four different datasets acquired with a LIDAR scanner: (a) a two floor building with a big lab downstairs and several offices on the first floor, with an approximated surface of 1400; (b) a conference building with a single floor and an approximated surface of 1800; (c) an industrial workshop with very high ceilings and with an approximated surface of 3000; (d) a large underground tunnel that can be explored by a car, and with a total length of 2.2. All models are obtained by registering the acquisitions to a common reference frame using the method of Yao, J., Ruggeri, M. R., Taddei, P., Sequeira, V., 2010. Automatic scan registration using 3d linear and planar features. 3D Research 1 (3), 1–18. The final map is generated by storing points and associated normals (and, if present, colours) after a voxel subsampling step of size 1 or 10.

[00190] For these datasets, the inventors evaluated the system using a Velodyne HDL-32E sensor mounted in three different configurations: on a backpack for walkthroughs, on a Segway and on the top of a car. Results were generated using a computer with an Intel Xeon CPU @ 2.80 GHz with 8 GB of RAM and a 64 bits operating system.

[00191] Place Recognition

[00192] In order to reduce the search space for the place recognizer, floors for all the maps were computed using the proposed flooding algorithm. At this stage, the inventors used big voxel cells (20) to perform the computations, since there is no need for a highly detailed representation of the floor limits. Average floor computation time for the three buildings was only 0.14 whilst the tunnel dataset took 3.91. Figure 8 shows the results for the office building.

[00193] Once floors were computed, the inventors estimated the effective navigable space, $N^* \subseteq N$. In particular, for the backpack mounted application, the inventors ran several tests including normal walking over flat surfaces, running, walking on stairs and performing fast rotations. During these tests, the position of the observer was continuously tracked and logged. Some of the results achieved are presented in Figure 8.

[00194] Figure 9 (left) shows the histogram of deviations with respect to the mean height above the floor. Results show a distribution with a standard deviation of $\sigma = 2.97$. This way, considering that the backpack-mounted sensor stands $10$ above the carrier's head, and that the human height between the $5\%$ and $95\%$ percentiles is in the range $[150.7 \ldots 188.7]$, according to [McDowell MA and CL(2008)], the effective height range above the floor was estimated as $[154.7 \ldots 204.6]$.

[00195] W.r.t. orientations, one considers a free motion over the $Z$ axis. The other two degrees of freedom are constrained since persons typically only bend some degrees while walking. Figure 9 (right) shows the histogram of deviations with respect to the absolute $Z$ axis observed during the evaluation tests (values are centered in $\mu = 3.66^\circ$ with standard deviation $\sigma = 2.37^\circ$). According to this, the present training process only considers deviations of $\pm 8.41^\circ$ to the vertical axis ($\mu + 2\sigma$).

[00196] Given these parameters, the total volume reduction on the search space (considering only positions) is shown in Table IV. Notice how, for regular buildings (office (a) and conference (b) building), the resulting search space is around $2\% - 3\%$ of

the total volume of the map whilst, in the workshop (c) and the tunnel (d) this ratio is considerably lower due to the high ceilings of the first, and the low density of navigable areas in the second.

[00197] Table IV: Navigable space reduction.

| map | volume ($m^3$) | navigable ($m^3$) | ratio |
|---|---|---|---|
| (a) | 26214.4 | 677.6 | 2.58% |
| (b) | 19660.8 | 564.3 | 2.87% |
| (c) | 1101000.5 | 669.9 | 0.06% |
| (d) | 72170864.6 | 11329.6 | 0.02% |

[00198] To measure place recognition performances alone, the inventors used five acquisition sequences and estimated the ground truth tracks by employing their tracking component with a manually initialized sensor position. They then trained a place recognition classifier jointly using the three different buildings. Each single track was then processed using the place recognition component alone (tracking based on classification). Since they did not provide information about the specific building in which the user was moving, the first candidate solutions were spread uniformly over all the three environments. During the experiments, each time the sensor moved more than 2 the place recognizer was queried. The total number of bins in the descriptor used was $12 \times 1$, and queries were performed with a radius of 6 in the descriptor space. A-priori possibilities for potential poses were computed considering $d_{\max} = 1$ and that locations were only comparable if their relative orientation was smaller than 45. The total size of the training set used for the three buildings was $1473$ KB.

[00199] They observed that after two or three iterations, roughly within $10$ from the starting position, candidate solutions clustered in the unique correct environment and then closely followed the correct sensor position. Figure 10 shows the drift analysis results of all the sequences employed. Notice that for each track the inventors removed the initial pose estimations related to the frames where the system still does not have enough information to uniquely identify the environment. As described by Geiger, A., Lenz, P., Stiller, C., Urtasun, R., 2013. Vision meets robotics: The kitti dataset. International Journal of Robotics Research the graphs shows the translation and rotation average drift errors given any point on the tracks after a specific track length.

[00200] Pose tracking

[00201] The pose tracking component has been evaluated by isolating each one of its components and generating individual results (map representation, point selection, inlier

selection and odometer integration) and by measuring the overall accuracy and performance of the complete system.

[00202] Map representation

[00203] To evaluate the scalability of the proposed map representation and to compare how it performs w.r.t. standard kd-trees, the inventors measured the space requirements of loading the entire voxel structure of each dataset in memory and isolated the nearest neighbour searches in the registration process to estimate the average computation time per query.

[00204] Table V shows the memory footprint of each dataset (fourth column), considering the voxel size (second column) and the dimensions of the complete voxel structure (third column). Notice that for the industrial building (c), two cases are considered: one that extends the original map by adding information about the exterior, (c)-o, and the original map where only the interior is stored (c)-i.

[00205] It is also important to notice that, in all the cases, the voxel data structure memory size is smaller than the point cloud that generated them.

[00206] Table V: Map sizes for the different datasets.

| map | voxel (m) | dimensions (m) | size (MB) |
|---|---|---|---|
| (a) | 0.1 | $25.6\times64\times16$ | 23.89 |
| (a) | 0.05 | $22.4\times59.2\times11.2$ | 124.72 |
| (b) | 0.1 | $64\times32\times9.6$ | 15.57 |
| (c)-o | 0.1 | $134.4\times64\times19.2$ | 69.11 |
| (c)-o | 0.05 | $129.6\times64\times19.2$ | 404.28 |
| (c)-i | 0.05 | $89.6\times51.2\times24$ | 304.71 |
| (d) | 0.1 | $442\times425.6\times284$ | 860.37 |

[00207] Table VI compares nearest neighbour searching times of the proposed map representation w.r.t. a standard kd-tree. For this test, both structures contained the same number of points and queries were performed using the same data. Results in columns 2 and 3 are expressed in nanoseconds per point and represent the average time considering all queries. Column 4 shows the average nearest neighbour error of the proposed map representation, due to the discretization of the space. Column 5 shows the total number of points in the map.

[00208] Table VI: Map average nearest neighbour computation times and errors for the proposed map representation compared with a standard kd-tree implementation.

| map | voxel (ns) | kd-tree (ns) | error (mm) | size (# points) |
|---|---|---|---|---|
| (a) | 53.7 | 573.3 | 0.220 | 184570 |
| (b) | 54.99 | 687.61 | 0.244 | 149030 |
| (c) | 77.32 | 744.46 | 0.083 | 1308380 |
| (d) | 69.23 | 876.26 | 0.185 | 9049443 |

[00209] Notice how, average searching times are always around 10 times faster than using kd-trees. Also notice how, the overall error in cases (a), (b), and (d), where a voxel cell size of 10 was used, is around 0.2. If this is reduced to 5, as shown in case (c), the error falls to 0.08.

[00210] Point selection

[00211] In the experiments, the inventors observed that their point selection technique to ensure geometric stability always provided robust results. They also observed that, if this feature was not enabled, tracking was lost when navigating on corridors. However, no significant differences were detected when comparing the stability of the results w.r.t. the technique proposed by [Gelfand et al. (2003) Gelfand, Ikemoto, Rusinkiewicz, and Levoy]. On the other hand, execution times were always smaller with the present technique, since the binning strategy used avoids sorting points according to their locking capabilities.

[00212] An additional test to evaluate the point selection strategy for symmetric environments was performed. In this case, the present technique properly locked orientations by selecting correct points, but the one proposed on [Gelfand et al. (2003) Gelfand, Ikemoto, Rusinkiewicz, and Levoy] failed. In this case, the present point selection strategy is not affected by the distance between points and the sensor. This way, critical points like the ones shown in cases A and C can be selected. This fact is evident when comparing results for case B. Since the present selection is normalized according to distances, the effect of the furthest points does not compromise the selection of the closest ones.

[00213] Inlier selection

[00214] In order to evaluate the proposed inlier selection strategy, the inventors proceeded as follows: the inventors mounted a Velodyne HDL-32E sensor on a tripod without moving it. The first frame was used as reference model and, during the rest of the experiment, outliers were progressively added (e.g. , people were moving around

and objects moved). This way, they could classify inliers correspondences by evaluating the distance between each point and its nearest neighbour in the reference frame.

[00215] Figure 4 shows the final precision of the present inlier selection strategy w.r.t. the number of outliers in the input cloud. Notice how, when the total number of outliers is below 20%, the present precision is almost always 100% (no wrong correspondences are selected for registration). As the overall number of outliers increases, precision decreases. On average, the proposed experiment had 27.83% wrong correspondences, that lead to a precision of 98.97%.

[00216] Odometer integration

[00217] To illustrate the benefits of the proposed odometer integration in the pose update component, the inventors recorded a track where, starting from the inside of building (a), they moved into a non scanned room and performed some loops by going out of the building and entering from a different door. Figure 13 shows the results achieved.

[00218] Notice how, when the sensor leaves the known environment (cases A, C and D), the tracking relies on the odometer only. Also, during the transitions between the known map and the non-mapped areas, the point selection strategy proposed gradually takes more information from the most convenient map without any specific logic to deal with these situations (take for example the transition shown in case C, right). As it can be observed, the accuracy of the proposed registration algorithm ensures that, when the user reenters the map after exploring the non-mapped areas, the odometer drift is low enough so that the tracking using the reference map can continue. Finally, when the sensor is moving inside the known space, it can be noticed how some of the points used for registration are taken from the odometer. This is generally due to the presence of points that have no valid correspondences in the reference map, but they do in the local map of the odometer. For instance, the environment in case B has big windows that allow the sensor to acquire points from the outside, which are not present in the original map.

[00219] Overall accuracy and performance

[00220] To measure the overall accuracy of the proposed pose tracking technique, the inventors performed an analysis of the ICP residuals after each cloud registration. This is imposed by the lack of a ground truth trajectory for free motion over a large indoor

scenario, since the area to cover is too big for using accurate external reference systems.

[00221] Figure 14 (upper) shows the average point-plane distances when moving inside an outlier-free scenario for both, the classical point-plane ICP algorithm and for the present robust ICP. The absence of outliers was ensured by performing the acquisitions immediately after scanning the ground truth model, represented using a voxel cell size of 10. Residuals for both approaches are almost identical, peaking on 2, which is within the nominal accuracy of the Velodyne HDL-32E sensor.

[00222] On the other hand, Figure 14 (lower) shows significant differences when changes are introduced into the environment. In this case, the track was recorded after refurbishing the environment. The present robust ICP implementation provides much better results than using the classical point-plane ICP, due to the efficient selection of inlier correspondences. In this case, residuals peak in 3 due to the interference of the outliers in the point-plane distance estimation for computing the shown histogram.

[00223] Given that the system must provide results in real-time, the inventors measured the overall performance during the pose tracking for different kinds of motion in all the datasets. Figure 15 shows the execution time spent in registering each cloud and computing the new pose of the sensor for a walking setup scenario. This process takes normally between 20 and 30ms but, at some frames, a peak around 100ms is observed. These peaks are related to the kd-tree generation for the odometer, which is triggered when a fixed distance is travelled since the time of the last update of the tree. The faster the sensor moves, the more this event will affect the overall performance. To avoid frame dropping, the kd-tree generation and the odometer runs in different threads and the latter uses the last available kd-tree until the new one is ready.

[00224] In Table VII average performance of the system is shown for three different setups (walking, segway mounted and car mounted). Notice how, the faster the sensor moves, the lowest the performance due to the odometer kd-tree updates. Since the Velodyne HDL-32E sensor provides readings at 12 Hz, all cases ensure real-time results, leaving processor time for performing additional operations. Finally, notice that in the current implementation all tracking computations were performed using a single CPU core.

[00225] Table VII: System execution performance for different setups.

| setup | performance |
|---|---|
| backpack mounted | 39.98 |
| segway mounted | 27.36 |
| car mounted | 23.34 |

[00226] 5 Conclusion

[00227] The present invention presents a complete system with preferred embodiments to assist in indoor localization applications that provides real-time results and scales well to the map size, allowing the exploration of very large environments.

[00228] By adding a pre-processing stage, an efficient place recognizer has been proposed that exploits the local motion of the sensor, measured using an odometer, and a compact and fast-to-compute descriptor. During the training of the place recognizer, a search space reduction strategy has been proposed that considers the physical constraints related to a particular operation mode of the system.

[00229] Pose tracking is performed using an efficent map representation, that provides constant nearest neighbour searching times, and that keeps memory requirements low. The present registration algorithm provides robust results by (1) selecting points that ensure geometric stability, (2) efficiently discarding outliers and (3) being fused with a local odometer whic allows using points not present in the reference map for registration and navigating through non-mapped areas.

[00230] Experimental results have proven the system to be highly scalable, perform tracking at frame rate leaving plenty of CPU time to run additional operations and to produce very accurate results (within the nominal accuracy of the sensor used), even when plenty of outliers are introduced.

## Claims

1. A method for real-time mapping, localization and change analysis of an environment comprising the following steps:

(A) if no 3D reference map of the environment exists, constructing a 3D reference map of said environment by

(a) acquiring the environment with a mobile real-time laser range scanner (1) at a rate of at least 5 frames per second to provide 3D scanner data,

(b) constructing a map presentation using the 3D scanner data for each of a plurality of poses of the laser range scanner (1), each pose having an associated point cloud defined by the 3D scanner data, the map presentation having a data structure set to natively handle 3D points which is based on a hybrid structure composed by a sparse voxelized structure used to index a compact dense list of features in the map presentation allowing constant time random access in voxel coordinates independently from the map size and efficient storage of the data with scalability over the environment, and

(c) building, using the map presentation, the 3D reference map for the environment using a 3D Simultaneous Localization And Mapping (3D SLAM) framework, said building comprising

(i) using an odometer module, estimating a current pose of the laser range scanner (1) based on the registration of a last point cloud to a local map presentation,

(ii) using a local trajectory optimization module, refining the trajectory of a set of point clouds in order to minimize the drift in the local map presentation, and

(iii) performing offline a global trajectory optimization by reconstructing an entire map of the environment taking into account loop closures of trajectories, thereby forming said 3D reference map;

and

(B) based on an existing 3D reference map of the environment, performing real-time mapping, localization and change analysis of said environment by

Date reçue/Date Received 2020-08-28

(d) acquiring the environment with the real-time laser range scanner (1) at a rate of at least 5 frames per second to provide 3D scanner data,

(e) during place recognition, identifying a current location of the laser range scanner (1) inside the environment with no prior knowledge of the laser range scanner pose during place recognition, and pre-computing of simple and compact descriptors of a scene acquired by the laser range scanner (1) using a reduced search space within the scene in order to self-localize the scanner in real-time, each descriptor of the scene comprising a range image of regular bins where each bin has an estimated median range value,

(f) after determination of the localization of the scanner in the environment, tracking the scanner pose by registering current scanner data inside said existing 3D reference map of the environment using standard Iterative Closest Point method employing data comprising nearest-neighbor information stored in the 3D reference map,

(g) calculating the distance between each scan point in the current scanner data and nearest point in the 3D reference map, wherein change analysis is performed by applying a threshold to this distance, whereby each point in the current scanner data which does not have a corresponding neighbor in the reference model is considered a change,

(h) displaying information about the 3D reference map and the current 3D scanner data on a real-time user interface.

2. The method as claimed in claim 1, wherein said information is color-coded according to a change/no-change classification of said information.

3. The method as claimed in claim 1 or 2, wherein the local trajectory optimization module comprises a local window mechanism optimizing a trajectory fragment composed by a set of poses and their associated point clouds with respect to a map built up to the last registered set; wherein the local window mechanism operates such that, when the distance between the first and the last pose in the list is larger than a threshold, cloud poses are optimized and a new list is produced with the refined pose and the input clouds.

4. The method as claimed in claim 3, wherein points are converted in world coordinates using pose interpolation in $\mathbb{SE}3$ group.

Date reçue/Date Received 2020-08-28

5. The method as claimed in claim 3 or 4, wherein a generalization of Iterative Closest Point method is used to find the trajectory that better aligns all the points to the map.

6. The method as claimed in any one of claims 1 to 5, wherein the data structure maintains five different representations of the data stored, thereby granting consistency between internal data representations after each map update, the five representations being

   (i) a compact and dense list of features, $L$ and an index to the last element, $L_{last}$ , where each element, $l_i \in L$, contains all the information associated to a feature in the map,

   (ii) a compact and dense validity mask, $M$, where each element, $m_i \in M$, is a boolean value indicating if its corresponding sample, $l_i \in L$, is valid or not, ensuring that $m_i = 0, i > L_{last}$,

   (iii) a list of holes, $H$, where each element, $h_i \in H < L_{\text{last}}$, indicates that $l_{h_i}$ is not valid so, $m_{h_i} = 0$,

   (iv) a sparse voxel representation $V$, built with a parametrizable cell size, that stores in each cell, $v_i \in V$, the index of its corresponding feature in $L$, wherein features in $L$ and cells in $V$ are related in a one-to-one manner, based on the position of $l_i$ and the cell size of $V$, and

   (v) a kd-tree, $K$, which is used to perform nearest neighbor searches on the map and which only stores references to the dense list $L$ to keep its memory footprint low.

7. The method as claimed in claim 6, wherein said information associated to a feature in the map comprises position and normal unit vector in world coordinates.

8. The method as claimed in claim 6 or 7, wherein, given an area of interest expressed by a central position and a radius, inner features are selected by looping over the elements stored in $L$ and the kd-tree $K$ is rebuilt as a fast mechanism for nearest neighbor searches.

Date reçue/Date Received 2020-08-28

9. The method as claimed in claim 1, wherein step (e) comprises the identification of a set of possible locations of the scanner based on the scanner data of step (d), said step (e) further comprising the following substeps:

(e1) based on the last 3D scanner data, computing an associated descriptor $\mathbf{q}$ and recovering a set of candidate locations $\Gamma$ by performing a radial search on $T$ given a threshold radius $r$ in the descriptor space increasing the candidate locations by computing additional input descriptors by horizontally shifting range values, each descriptor corresponding to the readings that the scanner would produce if rotated on its local axis and then rotating according to $i$ each resulting set of candidate locations,

(e2) associating a weight $\mathbf{w}_{\Gamma_p}$ to each potential location $\mathbf{\Gamma}_p \in \Gamma$:

$$\mathbf{w}_{\Gamma_p} = 1 - \frac{\left\| \mathbf{d}_p - \mathbf{q} \right\|}{r},$$

where $\mathbf{d}_p$ is the descriptor associated to the location $\mathbf{\Gamma}_p$ retrieved from $T$, $\mathbf{w}_{\Gamma_p}$ is 1 for perfectly matching descriptors and 0 for descriptors on the search sphere boundary, and

(e3) collecting weights in $\mathbf{w}$ and normalizing these weights to have $\max \mathbf{w} = 1$.

10. The method as claimed in claim 9, wherein in substep e1), computing the associated descriptor $\mathbf{q}$ and recovering the set of candidate locations $\Gamma$ is done for 360° horizontal view scanner data.

11. The method as claimed in claim 1, wherein step (e) comprises the identification of a set of possible locations of the scanner based on the scanner data of step (d), said step (e) further comprising the following substeps:

(e1) based on the last 3D scanner data, computing an associated descriptor $\mathbf{q}$ and recovering a set of candidate locations $\Gamma$, the candidate locations having a descriptor similar to $\mathbf{q}$,

(e2) associating a weight $\mathbf{w}_{\Gamma_p}$ to each potential location $\mathbf{\Gamma}_p \in \Gamma$:

Date Reçue/Date Received 2021-02-16

$$\mathbf{w}_{\Gamma_p} = 1 - \frac{\left\|\mathbf{d}_p - \mathbf{q}\right\|}{r},$$

where $\mathbf{d}_p$ is the descriptor associated to the location $\Gamma_p$ retrieved from $T$, $\mathbf{w}_{\Gamma_p}$ is 1 for perfectly matching descriptors and 0 for descriptors on the search sphere boundary,

(e3) collecting weights in $\mathbf{w}$ and normalizing these weights to have $\max \mathbf{w} = 1$,

(e4) updating the set of candidate locations while the scanner moves by estimating the movement and re-evaluating the weight for each initial candidate pose based on the query results at the new pose, and

(e5) iterating the update substep until the candidate poses converge to a single location.

12. The method as claimed in any one of claims 1 to 11, whereby the laser range scanner (1) is mounted on a person or on a vehicle traversing a floor, comprising the following steps

(i) identifying in the 3D reference map the extents of the floor, wherein floor extraction is performed over a sparse voxel representation of the environment, $V$, where each full cell, $v^{(i)}$, of the sparse voxel representation contains a normal vector to the surface locally defined by the points around its centroid, $\bar{\mathbf{n}}^{(i)}$, by extracting a subset of voxels that represent candidate floor cells, $F \subseteq V$, by checking that the vertical component in their associated normals is dominant, i.e. $\bar{\mathbf{n}}^{(i)} \cdot (0,0,1)^T \geq \varepsilon$, where $\varepsilon$ is typically a value between 0.5 and 1

(ii) determining reachability of cells, wherein given a reachable cell $f \in F$, all surrounding cells $(g^{(1)}, g^{(2)}, \ldots, g^{(m)}) \in F$ are considered as reachable if the following conditions are satisfied:

$$\left\|f - g^{(i)}\right\| \leq \theta_0 \qquad (6)$$

$$\left\|f_Z - g_Z^{(i)}\right\| \leq \theta_1 \qquad (7)$$

Date Reçue/Date Received 2021-02-16

$$C_{g^{(i)}} \cap V = \varnothing \tag{8}$$

where $\theta_0 \geq V_{\text{cellSize}}$ in (6) stands for the maximum step distance, $\theta_1$ in (7) stands for the maximum vertical step size and $C_{g^{(i)}}$ in (8) stands for the simplified volume of the observer, centered over the floor cell $g_i$.

13. The method as claimed in claim 12, wherein the method is for ground motion.

14. The method as claimed in any one of claims 1 to 13, wherein the map structure comprises two different lists of elements that are stored and synchronized: a compact list of planes, $L$, and a dense grid of voxels, $V$, built with a specific voxel size, each plane $l_i \in L$ storing a position in world coordinates, $\mathbf{p_i}$, and a unit normal, $\overline{\mathbf{n_i}}$; wherein each voxel, $v_i \in V$ stores a current state that can be either full, empty or near, full voxels storing an index to the plane $l_{v_i} \in L$, whose associated position falls into, empty cells storing a null reference and near cells storing an index to the plane $l_{v_j} \in L$ whose associated position distance $d_v$ to the voxel centre is the smallest.

15. The method as claimed in claim 14, wherein a near voxel is considered only if the distance $d_v$ is under a given threshold $d_{max}$, otherwise the voxel is considered empty.

16. The method as claimed in any one of claims 1 to 15, wherein, to improve overall system robustness, the scanner tracking is combined with an odometer, wherein

after a pose has been estimated, its associated points in world coordinates are stored into a kd-tree,

given a new acquisition by the laser range scanner (1), when a registration algorithm creates the sets of points ($\mathbf{p}_i^W$), it looks for nearest neighbors in both the reference map ($\mathbf{q}_i^M, \overline{\mathbf{n}_i}^M$) and in the previously fixed point cloud ($\mathbf{q}_i^O, \overline{\mathbf{n}_i}^O$), wherein correspondences are defined as:

$c_i^{(j)} =$

Date reçue/Date Received 2020-08-28

$$\begin{cases} \{\mathbf{p}_i^W, \mathbf{q}_i^M, \bar{\mathbf{n}}_i^M\} & \left\|\mathbf{p}_i^W - \mathbf{q}_i^M\right\| - s \le \left\|\mathbf{p}_i^O - \mathbf{q}_i^O\right\| \\ \{\mathbf{p}_i^W, \mathbf{q}_i^O, \bar{\mathbf{n}}_i^O\} & \left\|\mathbf{p}_i^W - \mathbf{q}_i^M\right\| - s > \left\|\mathbf{p}_i^O - \mathbf{q}_i^O\right\| \end{cases},$$

where $s$ corresponds to the voxel cell size and compensates the different resolution between the voxelized ground truth map and the non-discretized kd-tree of the previously fixed cloud.

17. The method as claimed in any one of claims 1 to 16, wherein the environment is a GPS-denied environment.

18. A mobile laser scanning device for real-time mapping, localization and change analysis arranged for implementing the method as claimed in any one of claims 1 to 17.

19. The mobile laser scanning device as claimed in claim 18, comprising a real-time laser range scanner (1), a processing unit (3), a power supply unit and a hand-held visualization and control unit (4), wherein the real-time laser range scanner (1) is capable of acquiring the environment with a rate of at least 5 frames per second to provide scanner data, the processing unit (3) is arranged to analyze said scanner data and to provide processing results comprising 3D map/model, localization and change information to the hand-held visualization and control unit (4), which is arranged to display said processing results and to allow a user to control the mobile laser scanning device.

20. The mobile laser scanning device as claimed in claim 18 or 19, wherein the visualization and control unit (4) is a tablet computer.

21. The mobile laser scanning device as claimed in any one of claims 18 to 20, wherein said mobile laser scanning device is a backpack (2) or vehicle mounted device.

22. The mobile laser scanning device as claimed in any one of claims 18 to 21, wherein the environment is a GPS-denied environment.

23. Use of a method as claimed in any one of claims 1 to 17 or of a mobile laser scanning device as claimed in any one of claims 18 to 22 for 3D indoor mapping/modelling; facility management; accurate and real-time indoor localization and navigation; assistance to disabled or elderly people; design information verification; change analysis; progress monitoring; or disaster management and response.

Date reçue/Date Received 2020-08-28

24. The use as claimed in claim 23, wherein change analysis comprises safeguards inspections or progress monitoring comprises civil construction.

Date reçue/Date Received 2020-08-28

Fig. 1

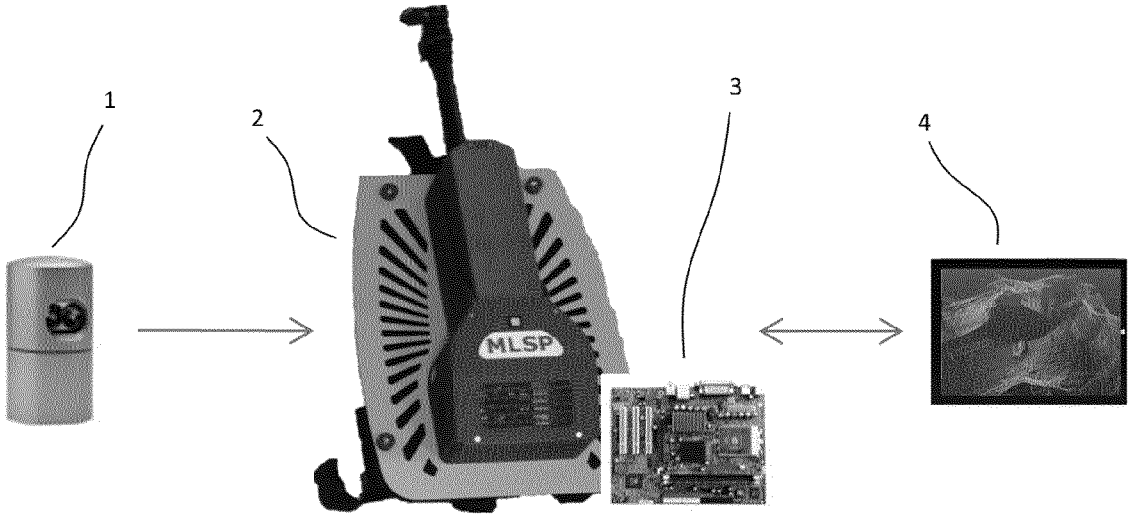



Fig. 2

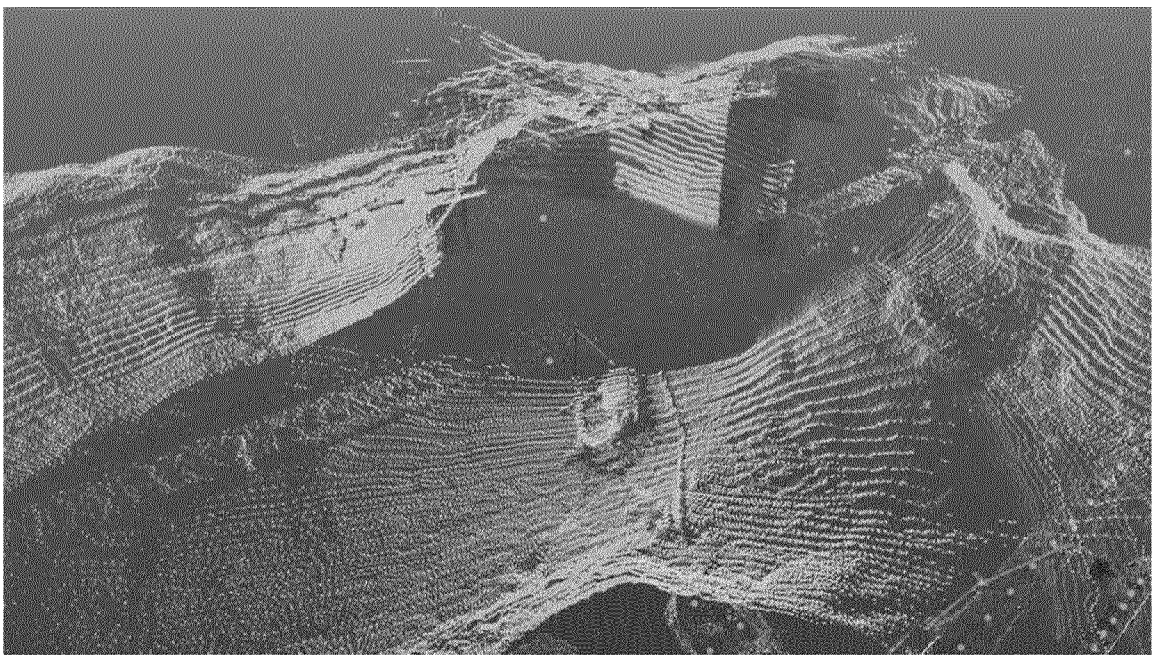


Fig. 3

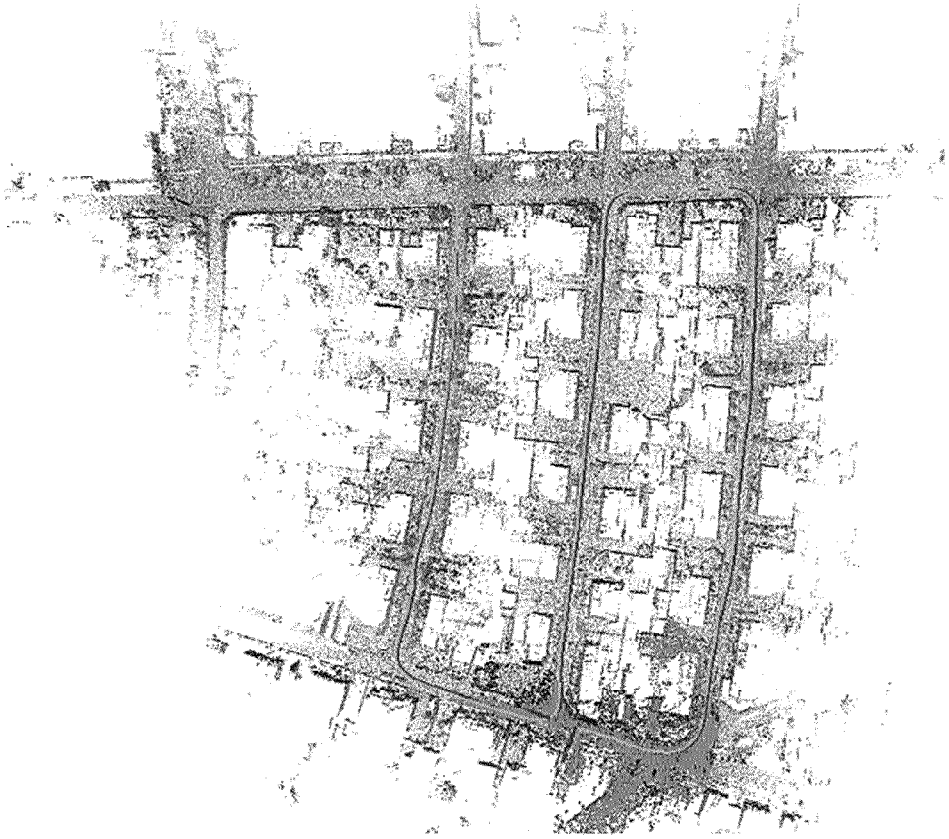


Fig. 4(a)

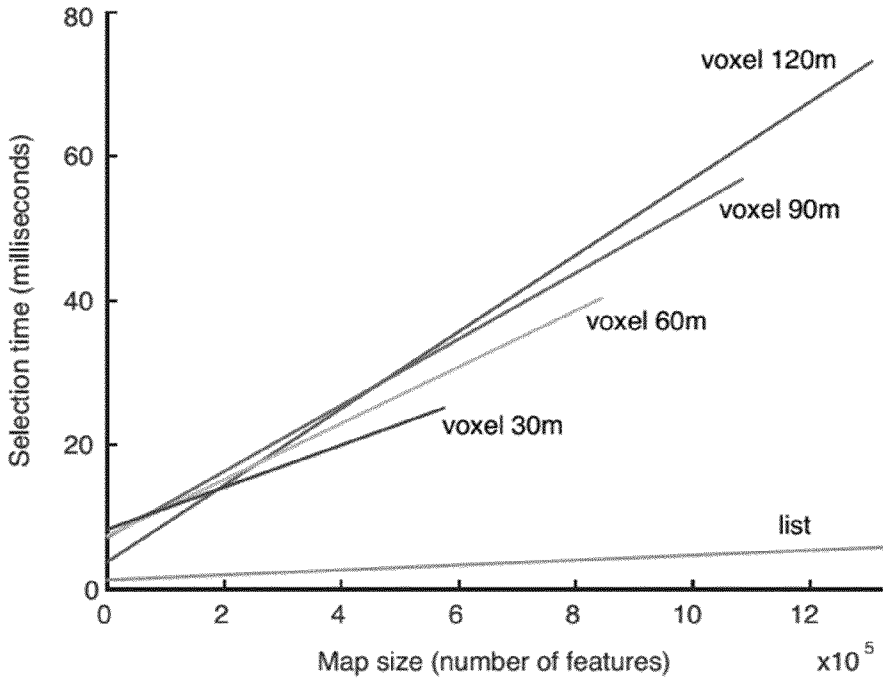

80
60
40
20
0
Selection time (milliseconds)
voxel 120m
voxel 90m
voxel 60m
voxel 30m
list
0
2
4
6
8
10
12
Map size (number of features)
x10 5

Fig. 4(b)

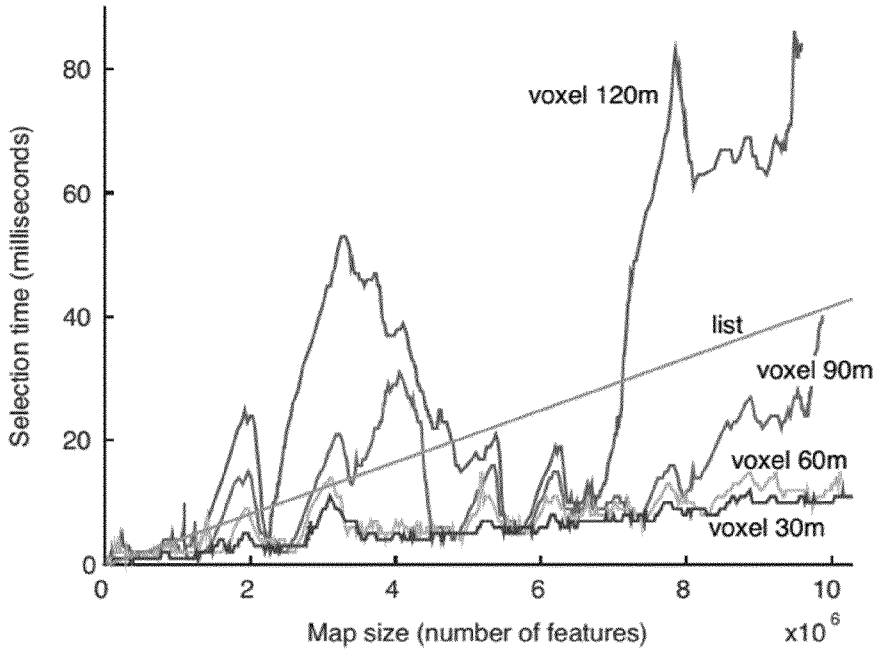

Selection time (milliseconds)
80
60
40
20
0
0
2
4
6
8
10
Map size (number of features)
x10 6
voxel 120m
list
voxel 90m
voxel 60m
voxel 30m

Fig. 5

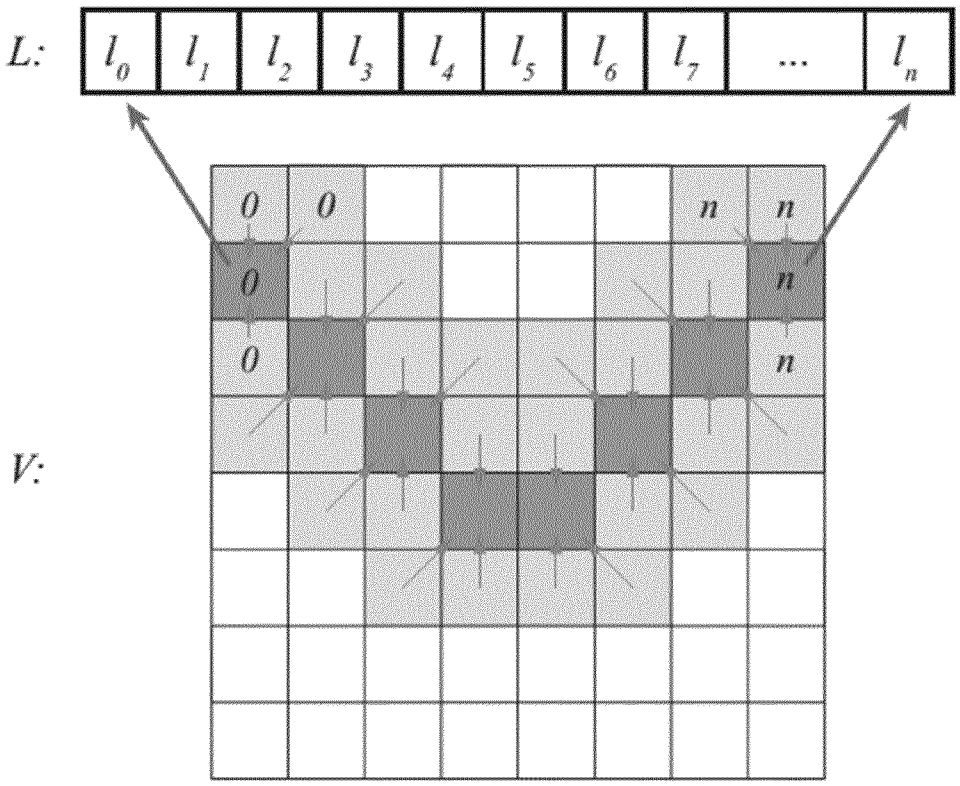

L:
l0
l1
l2
l3
l4
l5
l6
l7
...
ln
V:
0
0
n
n
0
n
0
n

**Fig. 6**

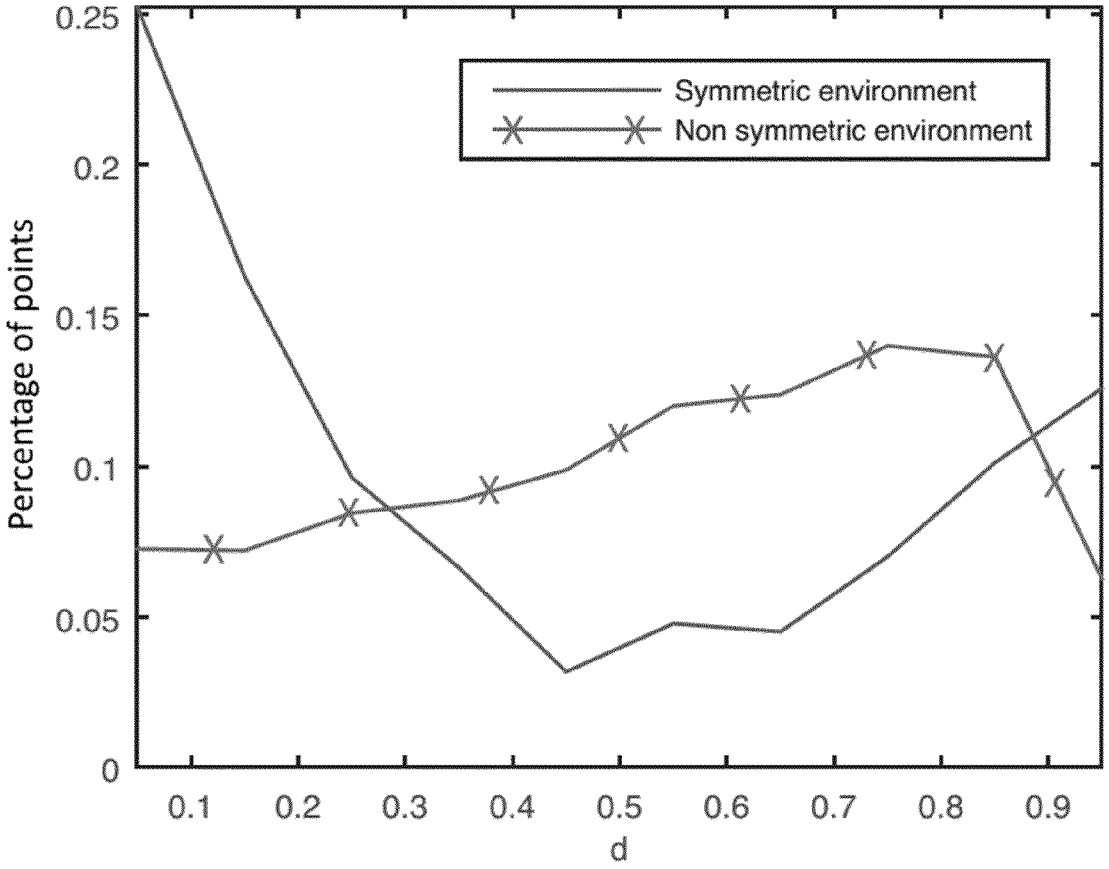

Symmetric environment
Non symmetric environment
Percentage of points
d
0.25
0.2
0.15
0.1
0.05
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

**Fig. 7**

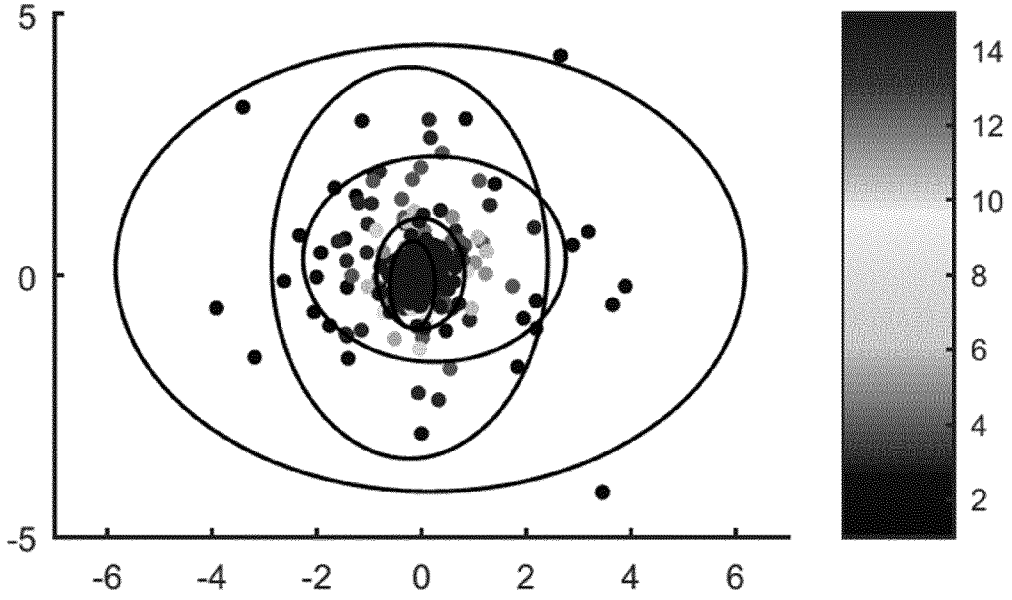

5
0
-5
-6
-4
-2
0
2
4
6
14
12
10
8
6
4
2

Fig. 8

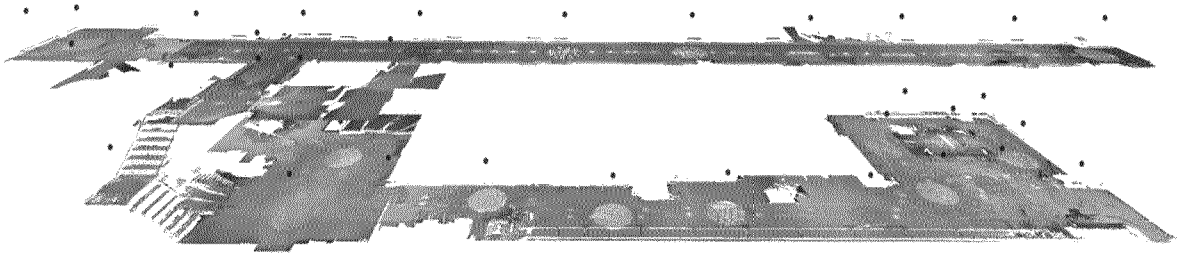


Fig. 9

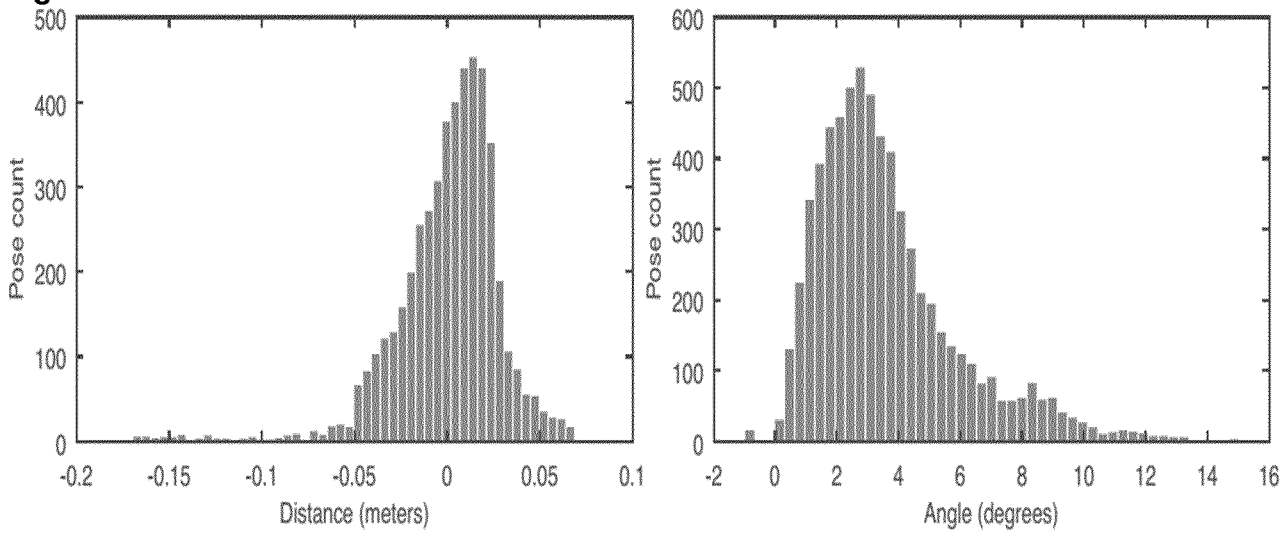



Fig. 10

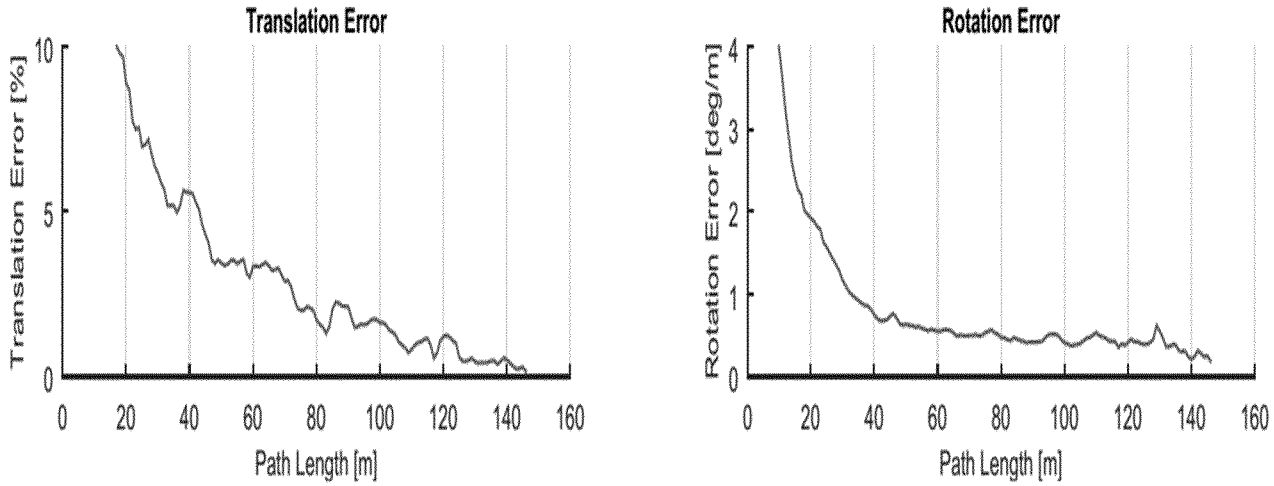



Fig. 11(a)

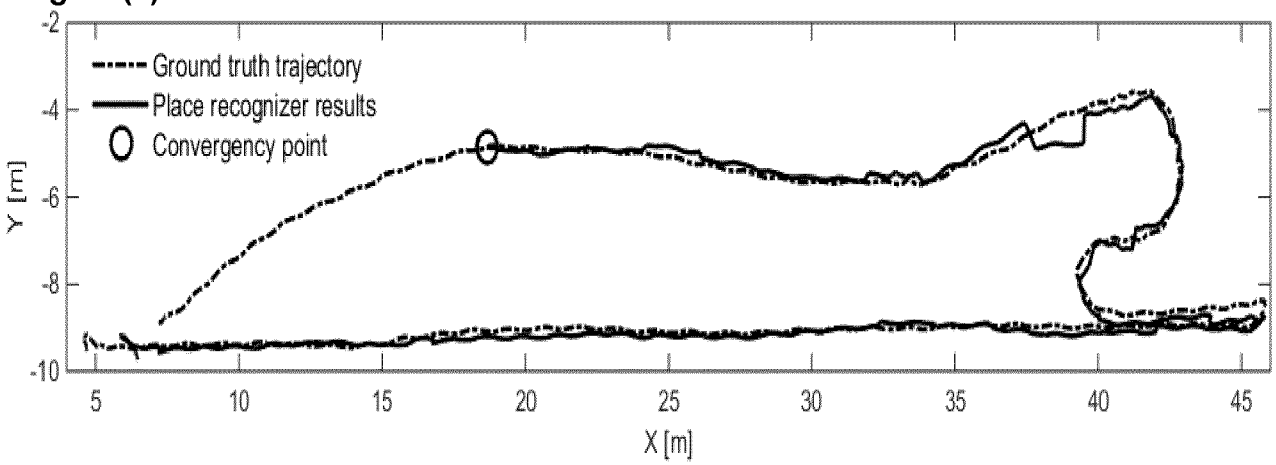


**Fig. 11(b)**

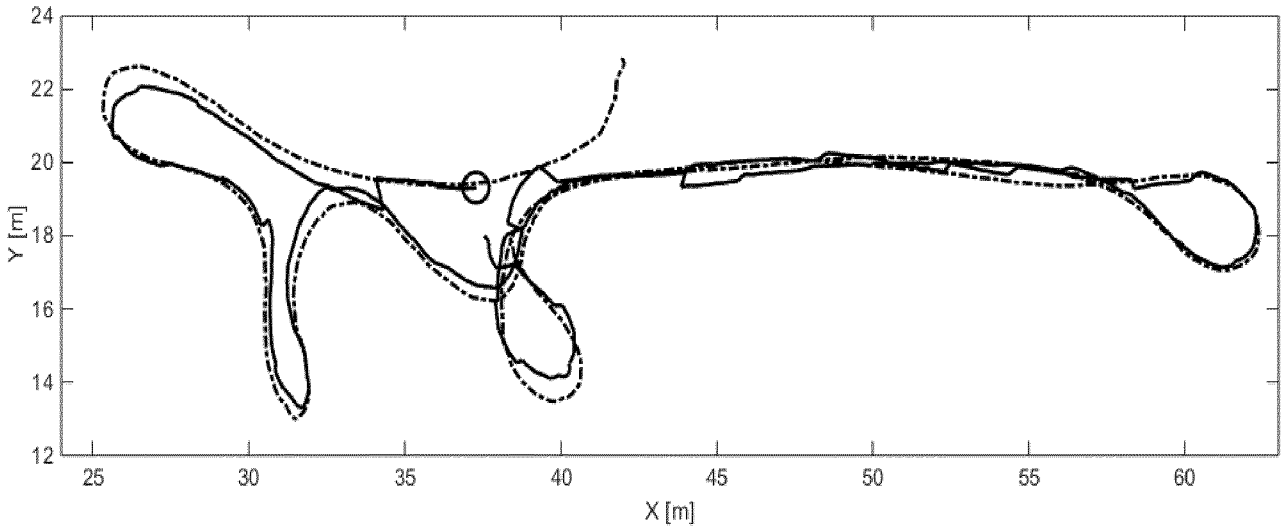



**Fig. 12**

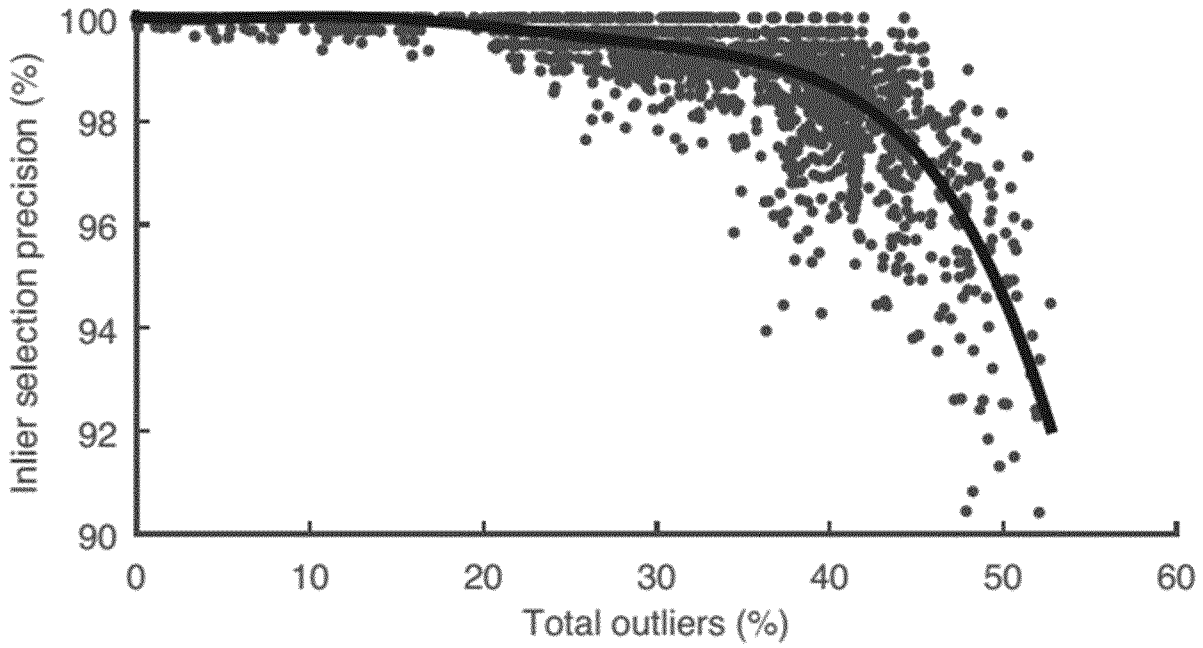



**Fig. 13**

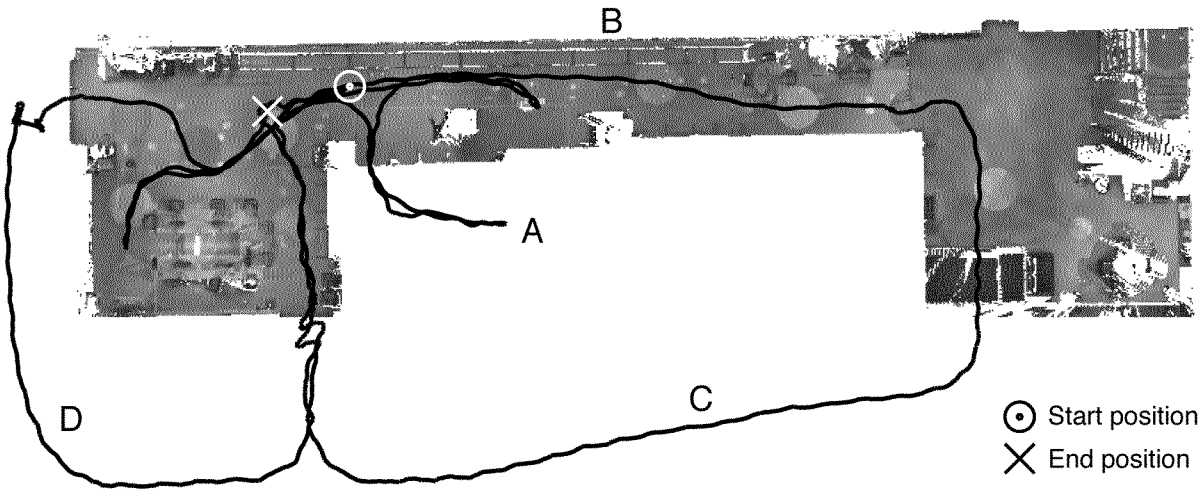


**Fig. 14**

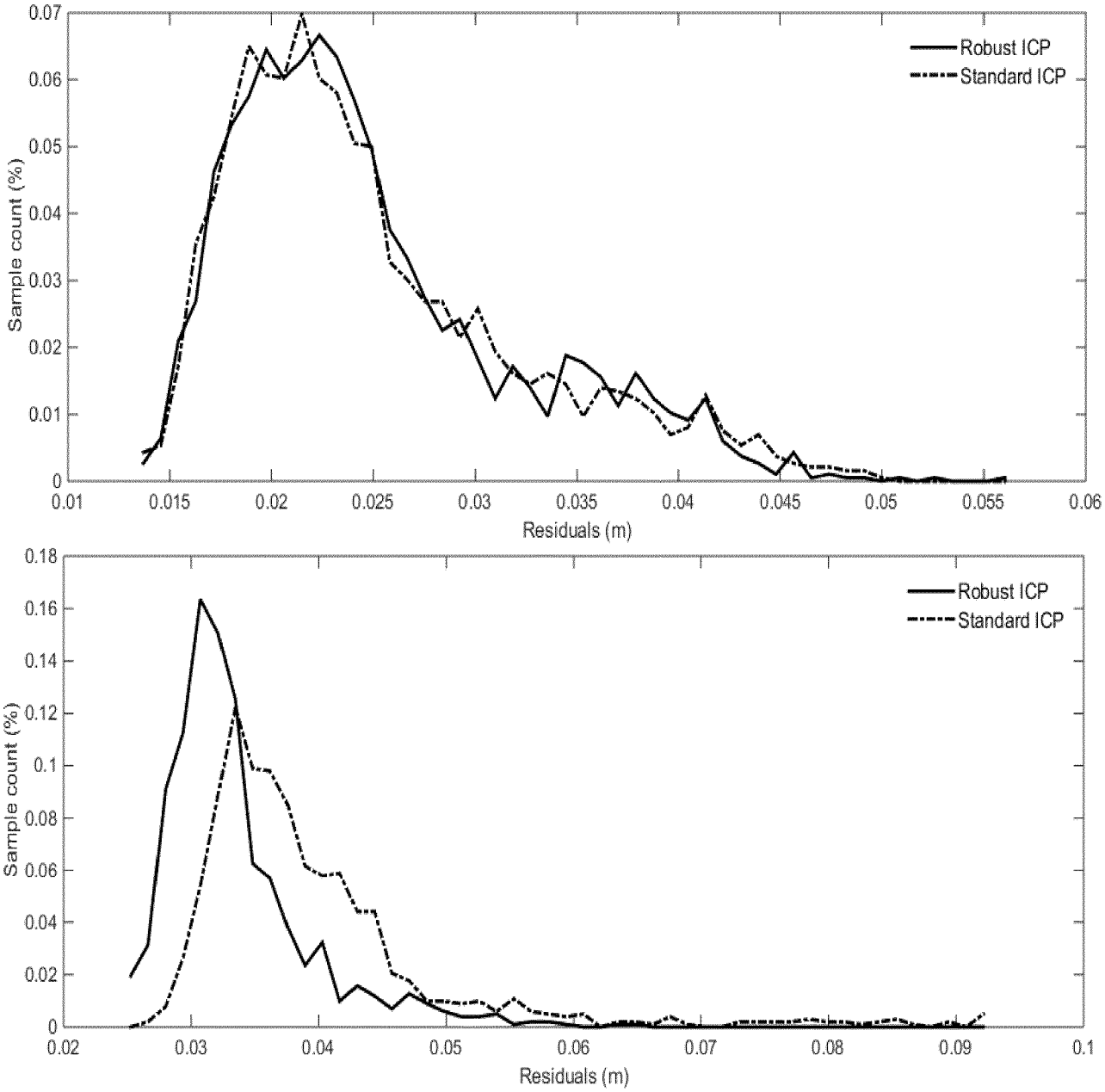
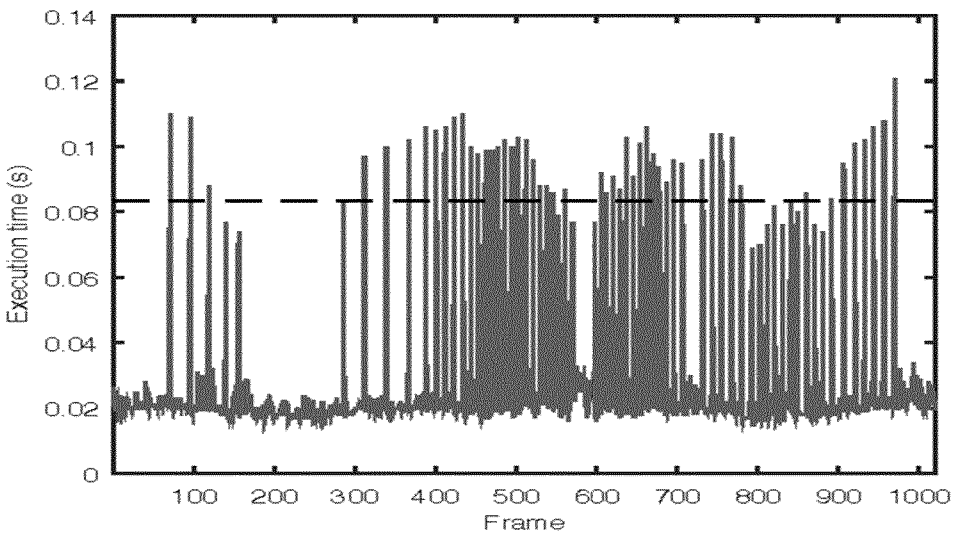
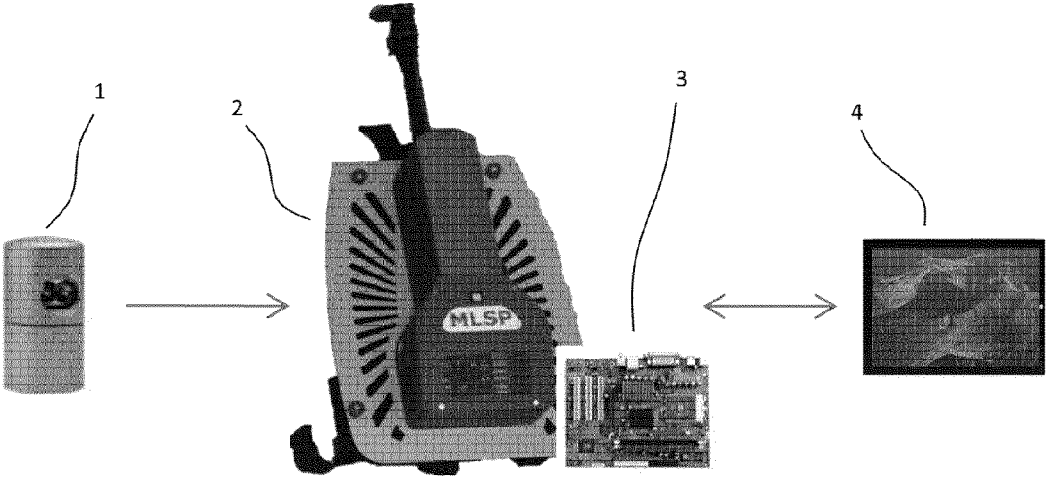

Robust ICP
Standard ICP
Sample count (%)
Residuals (m)
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0.01
0.015
0.02
0.025
0.03
0.035
0.04
0.045
0.05
0.055
0.06
Robust ICP
Standard ICP
Sample count (%)
Residuals (m)
0.18
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1

**Fig. 15**


Execution time (s)
Frame
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
100
200
300
400
500
600
700
800
900
1000


1
2
3
4
MLSP