Int. J. Appl. Math. Comput. Sci., 2016, Vol. 26, No. 1, 63–79
DOI: 10.1515/amcs-2016-0005

EFFICIENT RGB–D DATA PROCESSING FOR FEATURE–BASED SELF–LOCALIZATION OF MOBILE ROBOTS

MAREK KRAFT [a], MICHAŁ NOWICKI [a], RUDI PENNE [b,c], ADAM SCHMIDT [a], PIOTR SKRZYPCZYŃSKI [a,*]

[a] Institute of Control and Information Engineering
Poznań University of Technology, ul. Piotrowo 3A, 60-965 Poznań, Poland
e-mail: {marek.kraft;michal.nowicki}@put.poznan.pl,
{adam.schmidt;piotr.skrzypczynski}@put.poznan.pl

[b] Faculty of Applied Engineering
University of Antwerp, Salesianenlaan 90, 2660 Hoboken, Belgium

[c] Department of Mathematics
University of Antwerp, Middelheimlaan 1, 2020 Antwerp, Belgium
e-mail: rudi.penne@uantwerpen.be

The problem of position and orientation estimation for an active vision sensor that moves with respect to the full six degrees of freedom is considered. The proposed approach is based on point features extracted from RGB-D data. This work focuses on efficient point feature extraction algorithms and on methods for the management of a set of features in a single RGB-D data frame. While the fast, RGB-D-based visual odometry system described in this paper builds upon our previous results as to the general architecture, the important novel elements introduced here are aimed at improving the precision and robustness of the motion estimate computed from the matching point features of two RGB-D frames. Moreover, we demonstrate that the visual odometry system can serve as the front-end for a pose-based simultaneous localization and mapping solution. The proposed solutions are tested on publicly available data sets to ensure that the results are scientifically verifiable. The experimental results demonstrate gains due to the improved feature extraction and management mechanisms, whereas the performance of the whole navigation system compares favorably to results known from the literature.

Keywords: visual odometry, simultaneous localization and mapping, RGB-D, tracking, point features.

1. Introduction

Determining the pose of a robot or a sensor with respect to the surrounding environment is a common problem in mobile robotics. Certain classes of mobile robots, e.g., walking or flying robots, require self-localization with six degrees of freedom—their pose is described by three coordinates for the position and the pitch, roll and yaw angles for orientation: $\mathbf{x}_R = [x_r y_r z_r \theta_r \phi_r \psi_r]^T$. For such robots the assumption commonly made in classic self-localization algorithms that the robot moves on a plane and the sensory readouts (from range sensors in particular) are parallel to that plane does not hold anymore, because these robots do not maintain their attitude (pitch and roll angles) stable enough to keep a sensor perpendicular to the gravity vector all the time. Moreover, the reliable dead reckoning assumption does not hold for robots moving in 3-D. Although this assumption was also removed in some earlier 2-D self-localization solutions (Skrzypczyński, 2009), in walking and humanoid robots or micro aerial vehicles (Engel *et al.*, 2012) reliable odometry from proprioceptive sensing is not available at all or is extremely poor, making these robots dependent on self-localization with exteroceptive sensors. Passive vision has many practical limitations (Davison *et al.*, 2007; Bączyk and Kasiński, 2010), whereas 3-D laser range finders with mechanical scanning are bulky, heavy, and often slow. Thus, compact, fast-frame-rate RGB-D cameras are the sensors of choice

*Corresponding author

for 3-D indoor navigation (Stoyanov *et al.*, 2011).

The emergence of inexpensive RGB-D sensors such as Kinect or Xtion caused a rapid progress in self-localization methods: simultaneous localization and mapping (SLAM) (Bailey and Durrant-Whyte, 2006; Durrant-Whyte and Bailey, 2006) and visual odometry (VO) (Scaramuzza and Fraundorfer, 2011). VO algorithms make it possible to determine the motion of the robot based solely on a sequence of images, without creating a map of the environment. A VO algorithm can also serve as a basis for a SLAM task formulated as a graph optimization problem (Kuemmerle *et al.*, 2011). The pose-graph is composed of the successive poses of the moving sensor, whereas edges of the pose-graph represent constraints between these poses (Durrant-Whyte and Bailey, 2006). These constraints are yielded by visual odometry and loop closures detected whenever the sensor returns to an already known part of the scene.

This article builds upon the concept of a fast, lightweight, feature-based RGB-D visual odometry system presented by Nowicki and Skrzypczyński (2013a). Here, this concept is further developed by investigating new methods for the extraction and management of features, and by demonstrating the use of our improved VO processing pipeline both as a stand-alone self-localization system (PUT VO hereinafter), and as a front-end for a pose-based SLAM system (PUT SLAM). This constitutes the main novelty with respect to the paper of Kraft *et al.* (2014), which was a basis for this substantially extended version. The key idea of our approach is to extract salient point features from the RGB images at selected RGB-D data frames (keyframes), track them visually over a sequence of RGB images, and then use the depth (D) images associated with the first and the last frame in the sequence to recover the 3-D positions of these keypoints at the beginning and at the end of the tracking process. This alleviates the computational burden as only selected depth images are processed. As the visual features are tracked frame-to-frame, the correspondences are considered to be known, which makes it possible to simplify the feature matching when we want to compute the inter-frame transformation to compute sensor motion between the consecutive keyframes.

This approach stands in contrast to the majority of known RGB-D visual odometry or SLAM systems, which use some form of dense point matching or frame-to-frame matching of sparse keypoints. In a parallel work (Belter *et al.*, 2015), we compare several architectures of a feature-based VO and SLAM system using RGB-D data, finding that the tracking-based VO pipeline is simple, fast and precise if it is fed with good quality RGB-D data at a high frame rate. Therefore, in this article we focus on keypoint extraction methods that are fast and robust to the artifacts specific to RGB-D data. As the technology of RGB-D sensors develops quickly, we prefer to investigate mathematically solid geometric criteria for reliable point features in 3-D that may be applied to the combined RGB and D data regardless of the physical nature of the depth measurement process.

In principle, the method should be applicable to both structured light sensors, like the popular Kinect (Khoskelham and Elberink, 2012), and time-of-flight (ToF) sensors (Hansard *et al.*, 2012), like the new Kinect 2. The new feature detector allows narrowing the detected set of features to the ones that are the best picks from the point of view of further processing using higher level algorithms in the VO pipeline. Moreover, we propose to adopt some efficient statistical learning algorithms and heuristics for the management of the extracted features. The aim of these methods is to control the spatial distribution of the features with respect to the image frame, preventing degraded configurations that can easily lead to singularities or numerical instability in the motion estimation process (Umeyama, 1991). The usefulness of the proposed methods is demonstrated using real-life examples—the PUT VO and PUT SLAM systems applied to publicly available data sets.

The remainder of this article is organized as follows. In Section 2 we review the most directly related work in RGB-D visual odometry and SLAM. Section 3 gives an overview of the architecture of our self-localization system in both configurations used for the experiments: pure frame-to-frame visual odometry and pose-based SLAM including graph optimization. Section 4 provides a detailed description of the methods and algorithms for feature extraction, management and tracking. Results demonstrating the ability of the proposed system to successfully process RGB-D data from data sets of various properties are provided in Section 5. Section 6 concludes the paper and formulates the directions of further research.

2. Related work

Contemporary approaches to the SLAM and VO problem using RGB-D data can be divided into three distinctive groups, depending on the way they make use of image (RGB) and depth map (D) channels of the acquired data. Many researchers adopted the 3-D point cloud matching approach that was previously successfully applied in robot navigation based on 3-D laser scans (Nüchter *et al.*, 2007) for use with RGB-D data. Most of the point-cloud-type motion estimation methods rely on the iterative closest points (ICPs) paradigm, which can be implemented in several variants (Segal *et al.*, 2009). However, due to the high density of depth data represented as a point cloud, using such data in direct motion estimation, e.g., through matching that involves consecutive RGB-D frames, requires significant computational power and is therefore time consuming.

To alleviate this problem, parallel processing using

graphic cards (general purpose graphics processing unit—GPGPU) can be applied, but such an approach is known to be power-consuming. This makes the solutions based on point clouds prohibitive in many mobile robotics applications, as they typically require the processing to be done in real-time, with relatively low power consumption.

The KinectFusion system (Izadi *et al.*, 2011) employs the GPGPU to run an efficient version of the ICP algorithm on the Kinect data stream in real-time. While KinectFusion is limited to small workspaces, some derived works, like the Kintinuous algorithm (Whelan *et al.*, 2012) removed this restriction. A problem with ICP-based approaches is also that they rely solely on the geometry of the environment. If the geometric structure is insufficient (e.g., a long corridor with flat walls) the ICP algorithm often yields a wrong estimate of the sensor motion. To alleviate such problems, caused by certain environment geometries, the recently presented improved Kintinuous system (Whelan *et al.*, 2013) incorporates also a photometry-based procedure for tracking the sensor, which is an implementation of the algorithm of (Steinbrücker *et al.*, 2011). However, this system still requires massively parallel processing on a GPGPU, which makes it unsuitable for small robots with limited resources.

A completely different method for RGB-D data processing is presented by Kerl *et al.* (2013). This is a representative example of dense methods, relying mostly on RGB images. These methods are based on the photo-consistency assumption—the pixels representing the same scene points remain similar over a sequence of images of the scene taken by a moving sensor. The assumption allows matching the characteristic image points across multiple consecutively registered frames. An advantage of such an approach is high accuracy achieved with an acceptable computational workload. A major disadvantage, however, is the lack of robustness to disturbances of photometric consistency caused by the presence of moving objects in the field of view of the camera, sudden changes in illumination, etc. (Scaramuzza and Fraundorfer, 2011).

An alternative to VO algorithms using dense D or RGB data are the algorithms operating on a subset of characteristic points in the image called keypoints or features. The transformation between two poses of the sensor at which the RGB-D frames were taken is determined based on the above-mentioned keypoints detected in either the RGB image or the depth data part of the RGB-D frame. The concept of using discrete point features for navigation is popular both in VO systems (Bachrach *et al.*, 2012), as well as in systems based on the SLAM concept (Endres *et al.*, 2012). A majority of the proposed self-localization solutions detect the distinctive keypoints in RGB images. The depth data are used for the determination of the spatial coordinates of the 3-D scene points represented by the image keypoints. The information on the scene structure is not directly used. The keypoint detection and matching across multiple images acquired in the sequence is performed using automated methods. These are based on the analysis of the photometric characteristics of the image patch constituting the keypoint neighborhood. An up-to-date analysis of the usefulness of popular keypoint detection and description algorithms in the context of robot navigation is given by (Schmidt *et al.*, 2013b).

The detectors for extracting keypoints from the depth data—3-D point clouds acquired by an RGB-D sensor facilitate the use of the knowledge of the geometric structure of the scene. Examples of such feature detectors include the PFH (*persistent feature histogram*) (Rusu *et al.*, 2008) and the newer detector/descriptor NARF (*normal aligned radial feature*) (Steder *et al.*, 2011). However, a recently performed experimental evaluation of a simple VO system using NARF keypoints has shown that, although the accuracy of the system was acceptable, the keypoint extraction procedure is too slow to meet the requirements of real-time, on-board operation in robotics applications (Nowicki and Skrzypczyński, 2013b).

To the best of the authors' knowledge, the literature of the subject does not mention any keypoint detector taking into account both the photometric and geometric aspects of RGB-D data. The development of such a joint detector may be beneficial, as in real-life situations the depth portion of data contains many areas in which the depth values are unknown ("holes") or they are contaminated by artifacts, whereas the RGB portion of data contains a valid scene representation. A few attempts have been made to develop a keypoint descriptor operating on both RGB and D data, but the proposed descriptor works in pair with standard keypoint detectors, operating exclusively on RGB or D images (Nascimento *et al.*, 2012). Properties of the range images that may be very useful for robust detection of depth-related features are investigated by Penne *et al.* (2013), who proposed a method for detecting planarity when using ToF range images is proposed. However, this method is tested in experiments on planarity tests and the segmentation of planar regions, but not used for feature detection. As keypoint-based RGB-D data processing methods are among the most computationally effective and accurate in the context of navigation of mobile robots, new keypoint detection algorithms capable of taking into account the specific characteristics of RGB-D data are in demand.

3. Structure of the self-localization system

3.1. RGB-D visual odometry. In this work, we consider a self-localization system using RGB-D data. The system is divided into two parts: the front-end, which is an implementation of our tracking-based VO

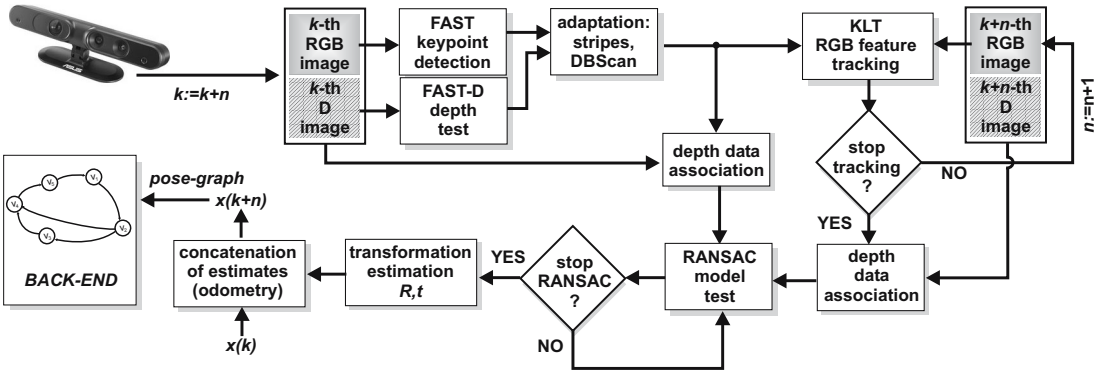


Fig. 1. Block scheme of the PUT VO/SLAM system based on feature tracking.

concept, and the back-end, implementing pose-graph optimization and loop closure detection. These two parts run asynchronously, exchanging only the necessary data structures. The design assumes that the system will be capable of real-time operation on a standard low-end PC, without resorting to the use of the GPGPU (Nowicki and Skrzypczyński, 2013a). At the same time, an important goal was to achieve as small an error of the sensor motion estimation as possible.

With these assumptions and requirements in mind, a decision was made at an early design stage to select an approach based on RGB image keypoint detection. The depth data were used to establish the spatial coordinates of scene points corresponding to the detected image keypoints. As keypoint detection, description and matching is time-consuming (especially for large sets of keypoints), the described system takes advantage of photometric tracking of detected keypoints across short RGB image sequences. This allows reducing the computational overhead as only the depth images are processed. Moreover, it is possible to establish the geometric relation between the first and the last RGB-D frame in the sequence without resorting to keypoint matching, as the tracking process follows the position of the keypoints from frame to frame. A detailed block diagram of the visual odometry front-end is given in Fig. 1, where the index k refers to a point in time (keyframe index) and n is the number of RGB images on which the photometric tracking of keypoints is performed.

The first stage of RGB-D data processing is the detection of distinctive keypoints in the RGB image using the FAST algorithm (Rosten and Drummond, 2006). The detected keypoints are then verified and filtered using depth image data by the original FAST-D detector, described in Section 4.3. The filtered keypoints are then tracked in a sequence of successive RGB images coming from the RGB-D data stream acquired by the sensor. The tracking is performed using the Lucas–Kanade algorithm (Baker and Matthews, 2004). The tracking process lasts until one of the criteria below is met (whichever occurs first):

- a maximum, predefined number of frames for the tracking process is reached,
- the number of successfully tracked features falls below a predefined level (the level is still greater than the minimum number of features necessary to calculate the inter-frame transformation).

The image coordinates of the corresponding keypoints and their depth in the first and the last frame of the RGB-D sequence for which the tracking was performed are used in further processing. Firstly, the 3-D coordinates of the points corresponding to the keypoints found in the first and the last frame are computed. As the tracking is initiated only if the depth of the keypoint was verified using the FAST-D detector, the number of cases in which a keypoint is assigned an invalid depth is significantly reduced, and the number of 3-D points is high enough to allow the computation of transformation. As the image correspondences between keypoints in the first and the last frame in the sequence are known, so are the correspondences between their 3-D counterparts.

With two sets of corresponding points in space ($\mathcal{A}$ and $\mathcal{B}$), one can calculate an unambiguous geometric transformation ${}^B_A\mathbf{T} = [{}^B_A\mathbf{R}, {}^B_A\mathbf{t}]$ between the coordinate systems A and B associated with the first and the last RGB-D frame, respectively. The transformation is composed of two components—the rotation and the translation between the two positions on the path of the sensor. There are several approaches to estimating the 3-D rigid transformation ${}^B_A\mathbf{T}$ (Eggert *et al.*, 1997). In the current implementation of the PUT VO system, the least squares estimation algorithm proposed by Umeyama (1991) is used. The direct availability of 3-D coordinates from the depth data allows the computation of transformation from a minimum of three point correspondences, whereas approaches relying only on the image data require at least five corresponding point pairs (Stewénius *et al.*, 2006).

Keypoint correspondences used for transformation estimation constitute the set $\mathcal{Z}$. The set can contain erroneous matches as a result of occlusions, inaccurate

tracking, distortions and noise, illumination changes, etc. This raises the need for the RANSAC robust estimation algorithm to be integrated as a part of the processing pipeline. In each RANSAC iteration, three random pairs of points are drawn from $\mathcal{Z}$. Based on this subset (the so-called minimum sample), the transformation is computed and its consistency with the remaining elements of $\mathcal{Z}$ is evaluated. The measure of consistency is the inlier ratio, that is, the ratio of the number of points in $\mathcal{Z}$ consistent with the computed transformation to the overall number of elements in $\mathcal{Z}$. A pair of points is considered consistent with the computed transformation if the transformation ${}^{B}_{A}\mathbf{T}$ for a point from $\mathcal{A}$ gives as a result the coordinates of the corresponding point from $\mathcal{B}$ with a certain accuracy.

RANSAC iterations are performed until a maximum predefined iteration number is reached, or upon reaching a specified threshold value of inlier ratio. The number of the required RANSAC iterations is estimated using a simple probabilistic model (Choi *et al.*, 2009) to improve speed. In most cases, robust estimation is completed in a few RANSAC iterations, as the 3-D correspondences set is derived mostly from correctly tracked points. When the RANSAC-based model search is finished, the transformation is re-estimated from all the inlier-pairs. If the number of inliers is high, iterative model correction is applied by rejecting the inliers that are the least probable within the model estimated so far (Raguram *et al.*, 2013).

3.2. Extending PUT VO to pose-based SLAM. Even the most precise visual odometry system is unable to detect and close any loop, and thus it cannot reduce the unavoidable trajectory drift. In order to enable such a correction, the proposed PUT VO system is paired with a back-end for pose-graph optimization. The back-end is based on the open-source g^{2}o software package for least squares optimization (Kuemmerle *et al.*, 2011).

The pose-graph assembled from the consecutive pose estimates produced by the VO front-end is optimized by minimization of a non-linear error function that is represented by the constraints (edges) of this graph. The back-end computes a globally consistent trajectory of the sensor, provided that all the constraints in the pose-graph are correct. The global optimization occurs whenever a loop closure is detected on RGB images by matching the keypoints from frames belonging to poses that are located far enough along the trajectory. If a significant similarity between two distant frames is discovered by comparing descriptors of their salient point features, a transformation is computed between these frames, and it is added as a constraint to the pose-graph.

Our system considers only local, "metric" loop closures (Strasdat, 2012), which are detected comparing features from the current keyframe to features from a finite set of previous keyframes, which are located far enough along the sensor's trajectory (i.e., with respect to the keyframe index), but are close enough in the sense of the Euclidean distance computed in the global reference frame. The implementation requires that a keyframe considered for loop closure detection be located at least 15 keyframes from the current sensor pose, but no farther than 5 m from this pose. This approach can handle small to middle scale loops, while the detection of large scale loop closures requires an efficient appearance-based place recognition method, such as the well-known FAB-MAP or one of its improved variants (Cummins and Newman, 2010). Integration of such a method in PUT SLAM is considered future work.

Because the PUT VO pipeline is based on tracking, there is no possibility to re-use the descriptors of keypoints (they are not computed for each frame). The keypoints with descriptors have to be computed specifically for the loop closure, but for a fraction of all RGB-D frames—only those that are considered loop closure candidates (Belter *et al.*, 2015). Taking into account the results presented by Schmidt *et al.* (2013b), we decided to implement the loop closure mechanism that can be configured to use alternatively one of the two detector/descriptor pairs: the classic yet efficient SURF (Bay *et al.*, 2008) and the more recent, very fast to compute, ORB (Rublee *et al.*, 2011).

4. Robust keypoint detection and tracking

4.1. Motivation and preliminaries. Contrary to typical applications of photometric keypoint detectors in image processing, applications that make use of RGB-D data require the detection of keypoints that are not just photometrically distinctive, but also supplemented with a reliable depth measurement. From the knowledge about the typical artifacts and measurement errors in depth cameras, we make the assumption that robust RGB-D keypoints are located on planar surfaces, because optical ranging senors usually produce erroneous depth measurements when the laser/IR beam or pattern illuminates a sharp edge. Thus, the direct neighborhood of a robust keypoint should be approximately planar. In order to derive a planarity test for a keypoint's neighborhood that is computationally efficient but general enough to be applied to the data from various RGB-D sensors, we formulate some coplanarity and colinearity criteria on the basis of an analysis of the image formation process in range cameras.

We assume a pinhole model for the range camera. For the analytic description of the observed scene (point features), we choose the camera reference frame (Fig. 2). As usual, the intrinsic camera parameters (neglecting lens distortions) are given by the calibration matrix (Hartley and Zisserman, 2004) K. As we assume rectangular pixels (zero skewness), the pinhole model is defined by

four intrinsic parameters: focal length f in pixels, aspect ratio τ, and pixel coordinates of the principal point $C_{u_0 v_0}$. For each pixel p_{uv} on the sensor, the 3-D coordinates in the camera reference frame can be computed by

$$\begin{pmatrix} x_p \\ y_p \\ z_p \end{pmatrix} = f \cdot K^{-1} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}, \tag{1}$$

and we can obtain the world coordinates (X, Y, Z) of the scene point P by simple scaling

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \frac{Z}{f} \begin{pmatrix} x_p \\ y_p \\ z_p \end{pmatrix} = Z \cdot K^{-1} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}. \tag{2}$$

We conclude that if a range camera is calibrated, and if it measures the z-depth (e.g., by means of structured light, like the first-generation Kinect sensor), then we can recover the 3-D position of observed feature points P.

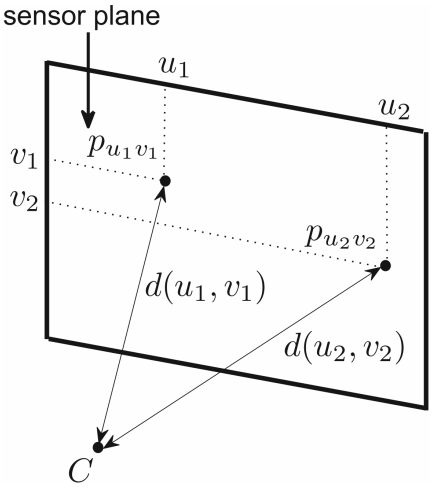


Fig. 2. Image formation, the coordinate system, and calibration parameters for a depth camera.

On the other hand, there also exist range sensors that provide radial depths, rather than z-depths (e.g., ToF cameras (Hansard *et al.*, 2012)). These devices measure the distance between the optical centre C and each point P_{uv} that is imaged in (u, v) (Fig. 2): $D(u, v) = |C - P_{uv}|$. In this case, the scale factor in (2) for transformation of the image p_{uv} to the world point P is not directly available by measurement, but can be computed by means of the *internal radial distance* $d(u, v) = |C - p_{uv}|$. These internal distances do not depend on the observed scene and refer to the intrinsic geometry of the (pinhole) camera:

$$d(u, v) = \sqrt{(u - u_0)^2 + \frac{(v - v_0)^2}{\tau^2} + f^2}. \tag{3}$$

In the work of Penne *et al.* (2015) the calibration of a ToF-camera is directly done by means of d, which is considered a parameter-free, more flexible alternative for the pinhole model (see also Mertens *et al.*, 2013). From the geometric similarity of perspective projection (still assuming zero skewness), we see that

$$P_{uv} = \frac{D(u, v)}{d(u, v)} (u - u_0, \frac{v - v_0}{\tau}, f). \tag{4}$$

We conclude that a spatial feature P_{uv} is obtained from the calibrated image coordinates (u_s, v_s) by simple scaling: $P_{uv} = \rho \cdot (u_s, v_s, 1)^T$, where ρ depends on the nature of the range measurement:

$$\rho_Z = Z \text{ if the z-depth is given,} \tag{5}$$

$$\rho_D = \frac{D(u, v)}{d(u, v)} f \text{ if the radial depth is given.} \tag{6}$$

Notice that ρ_Z only involves the range measurement, whereas ρ_D needs an additional computation involving the calibration parameters.

4.2. General planarity detection method for depth sensors. On the basis of the preliminaries given in the previous section, we formulate here general planarity criteria, which may be applied to both the classes of the range sensors considered: providing either the z-depth or the radial depth.

Let P_1, P_2, P_2, P_4 be four scene points available in the image as p_1, p_2, p_3, p_4, given by pixel coordinates $p_i = (u_i, v_i)$. The world coordinates for each of these points P_i are given by

$$\begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} = \rho \cdot K^{-1} \begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix}, \tag{7}$$

where ρ depends on the kind of range sensor (z-depth or radial depth). The coplanarity condition for the quadruple P_1, P_2, P_3, P_4 is given by

$$\begin{vmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \\ 1 & 1 & 1 & 1 \end{vmatrix} = 0 \tag{8}$$

or, after expanding with respect to the last row,

$$\begin{vmatrix} x_2 & x_3 & x_4 \\ y_2 & y_3 & y_4 \\ z_2 & z_3 & z_4 \end{vmatrix} - \begin{vmatrix} x_1 & x_3 & x_4 \\ y_1 & y_3 & y_4 \\ z_1 & z_3 & z_4 \end{vmatrix} + \begin{vmatrix} x_1 & x_2 & x_4 \\ y_1 & y_2 & y_4 \\ z_1 & z_2 & z_4 \end{vmatrix} - \begin{vmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{vmatrix} = 0.$$

Each term in this equation can be rewritten as

$$\begin{vmatrix} x_i & x_j & x_k \\ y_i & y_j & y_k \\ z_i & z_j & z_k \end{vmatrix} = \rho^3 \cdot \frac{1}{\det(K)} \cdot \begin{vmatrix} u_i & u_j & u_k \\ v_i & v_j & v_k \\ 1 & 1 & 1 \end{vmatrix}. \tag{9}$$

Finally, the coplanarity constraint (8) can be translated in the following equation:

$$[234]\rho_2\rho_3\rho_4 - [134]\rho_1\rho_3\rho_4 + [124]\rho_1\rho_2\rho_4 - [123]\rho_1\rho_2\rho_3 = 0, \quad (10)$$

only involving the range depths ρ_i and the "signed double triangle areas",

$$[ijk] = \begin{vmatrix} u_i & u_j & u_k \\ v_i & v_j & v_k \\ 1 & 1 & 1 \end{vmatrix}, \quad (11)$$

directly measurable in the uncalibrated image (independent of K). In the special case when p_1, p_2 and p_3 are collinear (on line L), the coplanarity condition for the world points P_1, P_2, P_3, P_4 is equivalent to requiring that P_1, P_2, P_3 be collinear. The coplanarity equation now simplifies to

$$[23]\rho_2\rho_3 - [13]\rho_1\rho_3 + [12]\rho_1\rho_2 = 0 \quad (12)$$

because $[123] = 0$ and $[ij4] = [ij]\cdot|p_4 L|$. Here $[ij]$ stands for the "signed distance" between p_i and p_j ($1 \leq i, j \leq 3$), with the sign determined by the orientation of L that makes p_4 lie on the left-hand side of L.

4.3. Practical planarity detection: FAST-D. The previous section provided mathematically sound and general criteria for determining both coplanarity and colinearity in depth images, regardless of the image formation mechanism. However, applying such criteria for practical verification of keypoints requires taking into account the uncertainty of the measured depth values (Khoskelham and Elberink, 2012), which is not straightforward as the general method provides no explicit uncertainty model or a tunable threshold that can account for the depth errors. Therefore, taking into account the general idea underlying the collinearity test presented before, we describe a practical approach, which may be tuned to be less strict and to accept features within an acceptable error margin of planarity. Moreover, this method, named FAST-D, is closely related in the geometry considered to the image keypoint detector FAST (Rosten and Drummond, 2006), which we use to find corner-like features in RGB images. FAST-D has an additional mechanism enabling verification of the depth measurements associated with the detected keypoints.

FAST keypoints are detected in images that were corrected to remove lens distortions and account for the sensor calibration, and converted into grayscale. The x and y image coordinates of keypoints are established with subpixel precision to improve the system's accuracy. As the Kinect sensor (both versions) is calibrated beforehand, it is possible to recover the corresponding depth values

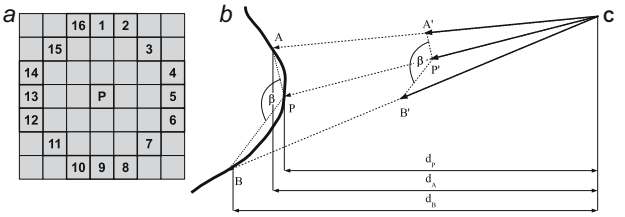


Fig. 3. FAST-D keypoint filtering: schematic depiction of the sampling points in the keypoint neighborhood (a) and geometric relations (b).

for any integer RGB image coordinate. FAST-D performs additional keypoint filtering based on the depth image. The filtering starts with the rejection of the keypoints for which there is no valid depth measurement. These keypoints are located on the areas that correspond to the "holes" in the depth image.

For each one of the remaining keypoints, a 7×7 neighborhood on the depth image is examined. The depth value d_p for the central pixel P and the depth values d_1 to d_{16} for the pixels 1 to 16, constituting a discrete Bresenham circle (the same as used in the original FAST) with a 3 pixel radius around P, are used for further filtering (Fig. 3(a)). The depth of all the points is divided by the depth of the central point:

$$\lambda_i = \frac{d_i}{d_P}, \quad i = 1, \ldots, 16. \quad (13)$$

This normalization makes the computations independent of the actual distance of the feature of the sensor. In order to cope with the exponential error of the depth measurements (Khoskelham and Elberink, 2012), only the parts of image with the corresponding depth smaller than a clipping threshold (set to 5 m) are analysed. Moreover, as the distance to the object increases, the analysed pixels of the Bresenham circle correspond to the more distant 3D points. Therefore, a larger local surface is analysed and the influence of the depth measurement errors decreases. As a result, the proposed detector performed consistently over the selected depth range. The result is a vector of 16 coefficients for the ring pixels – λ_1 to λ_{16} cf. (13). The transformed depth of the central pixel is equal to 1. Let us now consider the central point P and sample opposite points A and B situated on the 16-pixel ring (as depicted in Fig. 3(a)). The points and the sensor form a spatial arrangement (as shown in Fig. 3(b)). The points are projected on the image plane as image points $\mathbf{x}_P$, $\mathbf{x}_A$ and $\mathbf{x}_B$, with the measured depths of d_P, d_A, d_B. The focal point of the system is denoted by C.

Let us denote by A', B' and P' the transformed points. The scaling is proportional, so the triangle pairs CAP and $CA'P'$ as well as CBP and $CB'P'$ are similar. The angle β between the vectors $\overrightarrow{PA}$ and $\overrightarrow{PB}$ and between the vectors $\overrightarrow{P'A'}$ and $\overrightarrow{P'B'}$ is therefore the same. With the depth of the central point equal to 1, we can use the laws

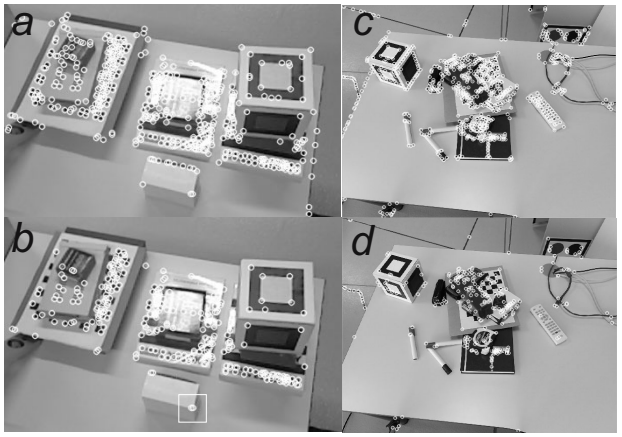


Fig. 4. Detected point features on example Kinect 1 and Kinect 2 frames: Kinect 1, no filtration—OpenCV FAST (a), Kinect 1, FAST-D filtration (b), Kinect 2, no filtration—OpenCV FAST (c), Kinect 2, FAST-D filtration (d).

of projective geometry to compute the 3-D coordinates of the points A' and B' denoted by $\mathbf{X}_{A'}$ and $\mathbf{X}_{B'}$. To do this, we multiply their homogeneous image coordinates by their respective coefficients λ_A and λ_B (Hartley and Zisserman, 2004):

$$\mathbf{X}_{A'} = \lambda_A \mathbf{x}_A, \quad \mathbf{X}_{B'} = \lambda_B \mathbf{x}_B. \tag{14}$$

The value of the angle β between the vectors $\overrightarrow{P'A'}$ and $\overrightarrow{P'B'}$ can be determined through the computation of the dot product of their versors, as the dot product of two unit vectors depends only on the angle between the said vectors. In our specific case, the following equation is satisfied:

$$\beta = \arccos\left(\frac{\overrightarrow{P'A'}}{|\overrightarrow{P'A'}|} \cdot \frac{\overrightarrow{P'B'}}{|\overrightarrow{P'B'}|}\right). \tag{15}$$

By using the value of β for thresholding and performing the test for all eight opposite point pairs lying on the Bresenham circle as shown in Fig. 3(a), one can effectively filter out keypoints that are not situated on stable, locally planar areas. If a feature successfully passes seven out of the eight tests, it is kept and used in the VO front-end pipeline. The experimental evaluation revealed that performing such an additional test for a single feature takes only about 0.015 ms.

Figure 4 demonstrates the results of filtering with the FAST-D algorithm for example frames yielded by the older Kinect 1 sensor (structured light range measurements, left column of Fig. 4) and the recently introduced Kinect 2 sensor, which uses a ToF camera (right column of Fig. 4). In both cases the threshold value of β of 145° was applied. Many of the keypoints detected by the original FAST algorithm are located on edges and corners of the physical objects (Fig. 4(a), (c)). The FAST-D approach eliminates most of these potentially unreliable RGB-D features (Fig. 4(b),(d)), but some isolated keypoints, particularly on corners are not discarded (example denoted by a rectangle in Fig. 4(b)) of the local planarity test due to the imprecise representation of the scene depth in the D image from the Kinect sensor. Since the range uncertainty depends on the distance between the sensor and the observed object (Khoskelham and Elberink, 2012), it may happen for very close or distant objects that while the intensity changes locally, giving rise to a corner-like feature, the depth values provided by Kinect in the same region remain almost constant.

4.4. Management of features. To make the feature detection more robust, we use two techniques: unsupervised clustering of the keypoints and detection of features in subimages. The clustering of features solves the problem that arises when many detected keypoints are located on a small area in the image, as seen in Fig. 5(a). This situation may lead to mismatches in feature tracking. To obtain isolated point features representing highly textured areas, we employ DBScan (Ester *et al.*, 1996), an unsupervised, density-based clustering algorithm working without any prior knowledge about the number of clusters (classes) in the image. Only two parameters need to be set in the algorithm—the minimum number of points to form a cluster and the maximum distance within the class. The algorithm considers one keypoint and looks for other points in its neighborhood, defined by the maximum Euclidean distance (in pixels). Whenever there are no neighbors and the conditions of cluster formation are not met, the keypoint is described as a nonclustered feature. Using an optimized implementation of DBScan, $O(n \log n)$ computational complexity is achieved, where n is the number of keypoints. Then, from each cluster established by DBScan, only two FAST-D features with the highest FAST score and enough distance between them are retained, while other points are discarded (Fig. 5(c)). It prevents the tracking algorithm from mistaking visually similar features that are located in the close neighborhood, and then reduces the number of outliers in the transformation between the two sets of points.

In order to compute a proper transformation between the two RGB-D frames represented by two sets of 3-D points, these points have to be spatially distributed in the field of view of the sensor. If most of the keypoints are located in one particular area of the image, e.g., at the bottom, the least-squares estimation of the transformation between the two point patterns may be ill-conditioned (Umeyama, 1991). Moreover, the system can fail whenever the RGB-D sensor is moving very fast, and the detected patterns of points shift from one area of the image to another leaving no overlapping features. Therefore, to ensure approximately equal

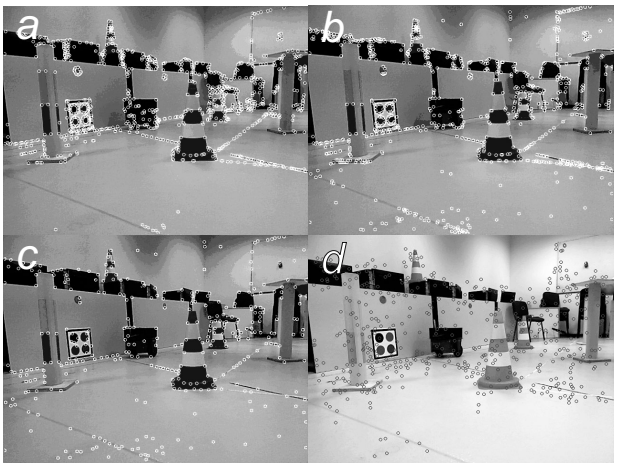


Fig. 5. Results of point feature management (features shown as small circles): all features detected in an RGB image by the FAST-D algorithm (a), FAST-D features detected in stripes imposed on the original image (b), features detected in stripes and clustered by using the DBscan algorithm (c), and the alternative organization of visual features obtained using the equi-distributed traversal quadtree algorithm (d).

distribution of the keypoints over the whole area of the image, this image is divided into six horizontal stripes of equal height. Then FAST-D detection is performed in each of these stripes individually, with the FAST detector threshold chosen adaptively (using `DynamicAdaptedFeatureDetector` from OpenCV) to ensure that a similar number of point features is detected in each subimage.

The concept of horizontal stripes is motivated by the fact that a mobile robot's sensor often sees the ground plane (floor) and some objects above it in one image, and these parts of the frame have quite different characteristics as to the number and location of possible point features (Fig. 5(b)). However, this approach works also for images acquired by a tilted sensor, when the ψ_r angle is non-zero and the line of the horizon (or the floor) is not parallel to the horizontal axis of the image (Kraft *et al.*, 2014). The number of stripes is a trade-off between our intention to divide the whole image into as many horizontal parts as possible, and the requirements of the detector, which cannot find keypoints too close to the boarders of an image (or a subimage). The stripes have to overlap slightly, in order to avoid a situation that the keypoints cannot be detected in some areas. Thus, having more than 6 stripes (for a 640×480 resolution), we would have more areas in which redundant features can be detected, and we would waste too much computation time.

An alternative approach to the management of point features for visual SLAM was presented by Strasdat (2012). He organizes the point features in a quadtree, and presents a traversal method, based on a combination of breadth-first and depth-first search in this tree, which ensures uniform feature distribution on the image. We implemented this algorithm as an alternative feature management method in the PUT VO/SLAM system (Fig. 5(d)), and in Section 5 we compare the performance of the quadtree-based management with our original approach, based on the stripes and DBScan.

4.5. Tracking of features on RGB images. In contrast to feature matching typically used in RGB-D-based motion estimation (Endres *et al.*, 2012), the tracking-based approach does not need to compute and match the descriptors of features, and performs keypoint detection only on a subset of images acquired from the RGB-D sensor. Thus, tracking is computationally less demanding than matching using classic descriptors, like SIFT (Lowe, 2004) or SURF.

The idea of tracking is to detect features at the keyframe, and then look for the position of this feature in the new image by searching locally. In our system, the tracking is performed using a pyramid implementation (Bouguet, 2000) of the Lucas–Kanade algorithm (Baker and Matthews, 2004), which tracks visually salient point features using a model of affine image changes. When the tracking starts on a new sequence of RGB frames, the point features in the initial image are obtained from the FAST-D keypoints and fed to the Lucas–Kanade tracker, which tells where these features should be located in the next image of the sequence. We use the FAST features instead of the Shi–Tomasi algorithm (Shi and Tomasi, 1994), which is more common in this application due to the higher computational efficiency of FAST.

The PUT VO system tracks features over a number of images of the RGB-D sequence between the two keyframes that are processed with depth images. If either the number of features that are well-tracked (do not have a high dissimilarity value computed by the Lucas–Kanade tracker) drops below a threshold or the maximum allowed number of the RGB frames in tracking is reached (max. $k = 5$), the tracker finishes its operation, and a new keyframe is established to extract a new set of features by using FAST-D. At a new keyframe we compute new keypoints, but also re-use some old features that were accepted as inliers when computing the last motion transformation. They may constitute up to 50% of the new set of keypoints. This strategy promotes visually salient "matured" features that are good to track. The 3-D transformation between the two keyframes at the beginning and at the end of the tracking sequence is computed using a robust estimation scheme. To avoid a drift when the sensor/robot is not moving, we do not update its pose and do not change the keyframe if the estimated motion transformation is close to identity.

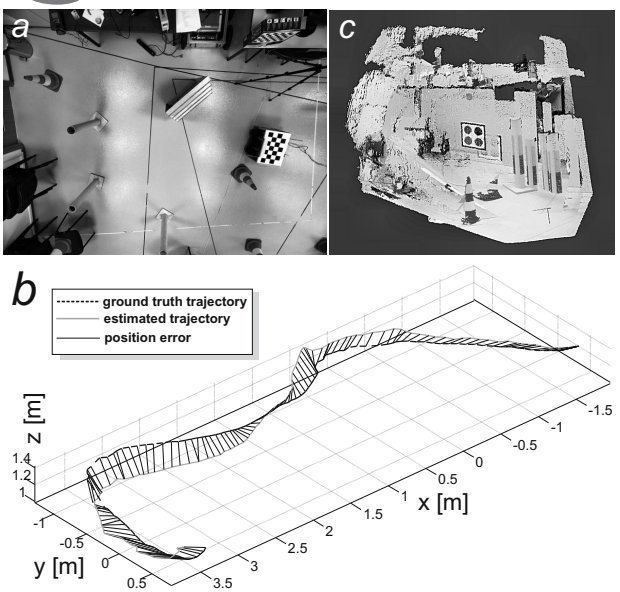


Fig. 6. PUT RGB-D data set—experimental setup and collected data: RGB-D data acquisition procedure (a), 3-D view of the ground truth and estimated trajectories in one of the experiments (b), 3-D rendering of a fragment of the scene resulting from the acquired RGB-D frames and the estimated trajectory (c).

5. Experiments and results

5.1. Experimental methodology. The trajectory estimation quality assessment for the PUT VO and PUT SLAM systems was performed using publicly available RGB-D data sets, to ensure that our results are scientifically verifiable. The data sets used in the experiments are the TUM RGB-D benchmark described by Sturm *et al.* (2012) and our own PUT RGB-D data set (Schmidt *et al.*, 2013a)[1]. These data sets were collected using the Kinect sensor. In both data sets the RGB-D frames are tagged with synchronized information on the ground truth position of the sensor obtained from an external motion capture system. The methods and algorithms used to obtain the ground truth data for the PUT RGB-D data set are described by Schmidt *et al.* (2013a), and an example view from an overhead camera of the motion capture system is shown in Fig. 6(a).

All experiments with the PUT VO and PUT SLAM systems were performed on a laptop PC with a 2.5 GHz processor and 8 GB of RAM. The VO front-end uses only a single core of the processor, however, the optimization back-end with loop closure detection is implemented in a separate thread and runs in parallel to the VO pipeline. In all tests, the maximum number of tracked features was 500. The maximum number of RGB images for feature tracking in-between two keyframes was set to 5. Both PUT VO and PUT SLAM run in real-time. While the speed achieved by particular variants of these systems is shown further on in this section (cf. Table 1), here we highlight some details as to the processing times in the tracking-based front-end: the per-frame feature detection and management (DBScan and stripes) time averaged over the tested RGB-D sequences was 10.8 ms, the averaged tracking time between two consecutive keyframes was measured as 13.7 ms, while the transformation estimation (within the RANSAC framework) took less than 1 ms for most of the tested sequences. Typically, only few iterations of RANSAC were necessary to establish an acceptable transformation model, which is attributed to the correct feature associations maintained by the Lucas–Kanade tracker.

We used the evaluation tools provided with the TUM RGB-D benchmark (Sturm *et al.*, 2012). The error metrics used are the relative pose error (RPE), which shows the local drift of the VO system, and the absolute trajectory error (ATE), which illustrates the difference between the estimated trajectory of the sensor and the ground truth trajectory.

5.2. Evaluation of improved RGB-D data processing. The impact of the improvements to RGB-D data processing on the quality of the sensor pose estimates was investigated in detail on the PUT VO system in order to avoid a situation when the results are altered by trajectory optimization in the pose-based SLAM back-end.

Figure 7 presents quantitative results of the tests of the PUT VO system conducted for the PUT RGB-D data set sequence `trajectory_1`. The sequence containing 500 RGB-D frames was used to investigate the effects of the improved feature extraction method and the new feature management techniques on the accuracy of the sensor's trajectory estimation. The 3-D view of the estimated trajectory (Fig. 6(b)) demonstrates the absolute trajectory errors, whereas Fig. 6(c) depicts a fragment of the scene (laboratory room) reconstructed from 140 consecutive RGB-D frames registered together using the estimated trajectory.

The RPE tool was used to evaluate the accuracy of local position and orientation estimation along the estimated trajectory, because in a visual odometry system the absolute position and orientation errors increase steadily (Scaramuzza and Fraundorfer, 2011). The RPE reveals the relative errors in translation and rotation between the successive RGB-D frames. Figure 7(a) shows the relative translation errors for the PUT VO system using the FAST-D algorithm and feature management mechanisms. To enable a direct comparison, results for the basic version of our VO (Nowicki and Skrzypczyński, 2013a), which is hereinafter referred to as the "tracking VO" approach, are depicted. The normalized histogram of these errors is shown in Fig. 7(c). The effects of the modifications made to the feature extraction procedures

[1]The data set is available at `http://www.vision.put.poznan.pl/?p=70`.

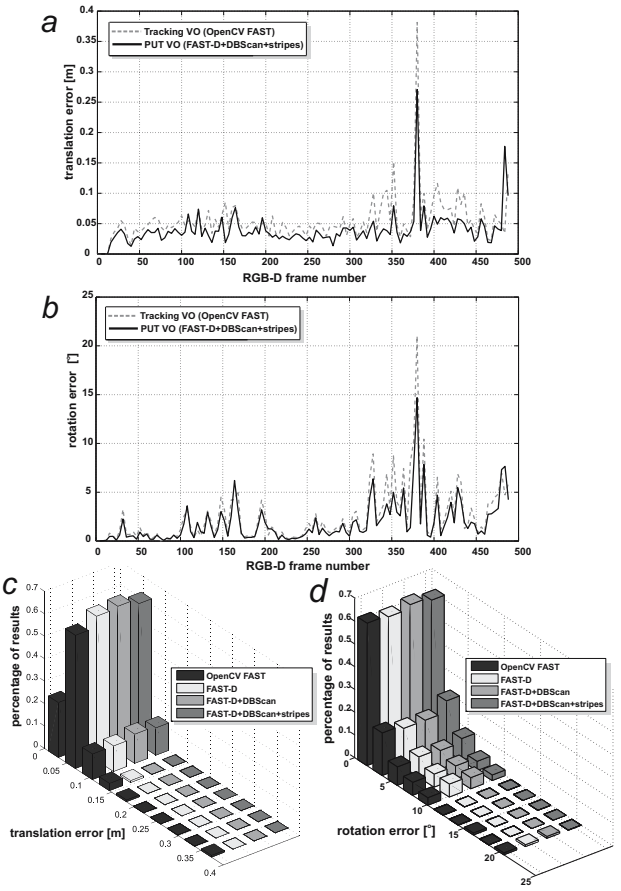


Fig. 7. Influence of the point feature extraction and management methods on the relative translation and rotation error of the PUT VO system: comparison of relative translation errors (a), comparison of relative orientation errors (b), as well as normalized histograms of the translation (c) and orientation (d) errors.

on the relative orientation errors are plotted in Fig. 7(b), whereas Fig. 7(d) shows the normalized histogram of these errors. Quantitative differences between the results achieved applying the particular feature extraction and management techniques are small, but it is clearly visible that the introduction of new feature management methods reduces the number of large errors in the estimated translation and rotation, compared to the basic version of the VO pipeline, which uses the simple FAST detector.

Knowing the impact of the new RGB-D data processing methods on the operation of the PUT VO system, we further investigate the influence of the improvements made to the front-end on the whole self-localization system. To this end, we compare the quantitative results: RPE and ATE values obtained by four configurations of the system under investigation, which represent particular improvements: Tracking is the simplest VO system without any improvements to the RGB-D data processing, VO (FAST-D) uses only the new FAST-D feature detector, VO (FAST-D+Quadtree)

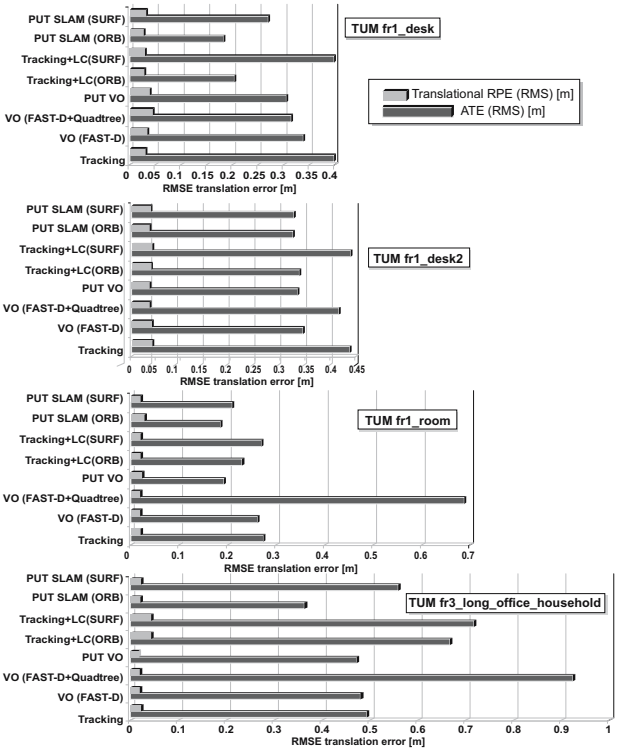


Fig. 8. Comparison of translation errors: the RPE and ATE for selected configurations of the PUT VO/SLAM system on four sequences from the TUM RGB-D benchmark.

represents an alternative version of our VO system that uses the approach of Strasdat (2012) for feature management, and, finally, PUT VO is the full RGB-D visual odometry system with all improvements proposed in this paper. This analysis is completed by comparing the translation errors obtained by another four configurations of the self-localization system, this time employing the pose-graph optimization back-end, i.e., g^2o. This part of the comparison demonstrates how the new front-end (PUT SLAM) improves results over the simple Tracking VO approach with the loop closure mechanism (Tracking+LC). Moreover, we test the loop closure in two versions, which differ in the ORB or SURF detector/descriptor pair being used. All these results are plotted in Fig. 8, using four example RGB-D data sequences from the TUM RGB-D benchmark. The accompanying qualitative results are demonstrated in Fig. 9, which shows the trajectories recovered from the two more challenging sequences than those used in Fig. 8: `fr1_room` (left column) and `fr3_long_office_household` (right column) visualizing the ATEs. Unfortunately, the evaluation tools we have adopted from Sturm *et al.* (2012) do not support visualization of loop closure constraints. The loop closure mechanism is investigated more thoroughly by Belter *et al.* (2015), employing custom visualizations. The trajectories in Fig. 9 are estimated by the three most representative variants of our self-localization system.

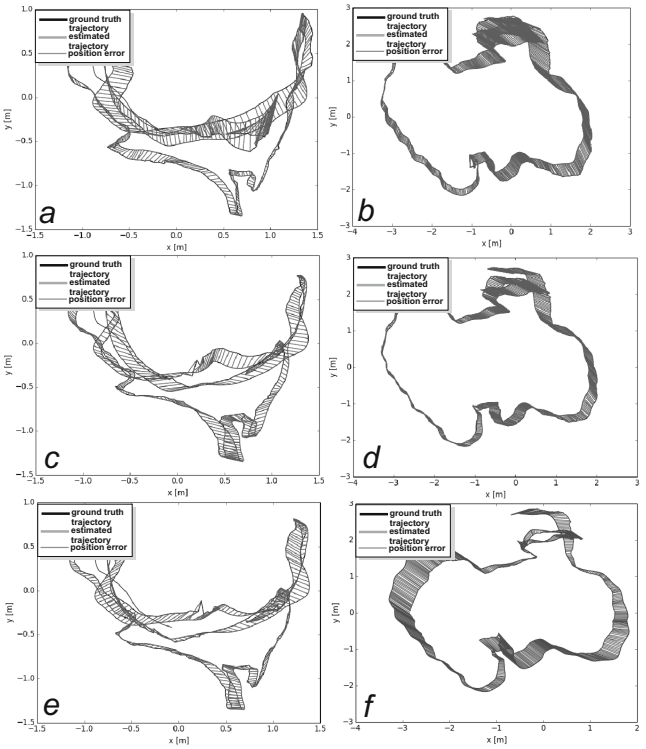


Fig. 9. Estimated trajectories with the ATE for two example sequences from the TUM RGB-D benchmark: simple tracking VO (a), (b), PUT VO with FAST-D and feature management (c), (d), and PUT SLAM with ORB-based loop closure (e), (f).

It is evident that the introduction of both the new detector and the feature management procedures decreases both the RPEs and ATEs. Surprisingly, the alternative quadtree-based data management procedure performs poorly, particularly for the larger scenes (`fr1_room` and `fr3_long_office_household`). This is probably caused by the tendency to force initialization of keypoints in areas which are naturally featureless. Such keypoints result in features that cannot be tracked reliably by the Lucas–Kanade algorithm. Moreover, we observed that the simple and fast ORB detector/descriptor performs better than SURF. This is attributed to the constant budget of time we allocate to the loop closure detection procedure, in order to keep the whole system real-time. As shown in the paper by Schmidt *et al.* (2013b), feature matching using ORB results in a higher matching ratio than when using SURF. Therefore, it is possible to limit the number of the detected ORB features to an arbitrary number of the most prominent ones. On the other hand, the lower matching reliability of SURF means that all features have to be used. When combined with the significantly higher detection and description time of the SURF algorithm, this means that adding a new frame to the system is much more time-consuming for the SURF-based variant, leaving less processing time for the loop detection procedure. Besides matching the binary ORB descriptors is much more efficient than matching the floating-point SURF features. As a result, if the SURF algorithm is used, fewer loop-closure candidates can be evaluated and thus fewer constraints are incorporated into the graph optimization. Considering that there was no noticeable difference in the quality of the matches, ORB proved to be the algorithm of choice.

We also performed a series of experiments that demonstrate the ability of our tracking-based VO pipeline to handle situations when the sensor moves with a relatively high velocity. Obviously, for an RGB-D sensor moving very fast, the main problem with our feature-based approach would be reliable detection of the keypoints, due to the unavoidable motion blur. In fact, this problem, not the processing time, sets the physical limit on the velocity of the sensor/robot. However, as long as the amount of motion blur is acceptable to the FAST detector, the tracking-based VO system operates correctly, with the accuracy of trajectory estimation degrading slowly for higher velocities, as the distances between the consecutive images increase, assuming that the RGB-D sensor works with a constant frame rate. This is demonstrated by the results of a simple experiment: the mobile robot with a Kinect sensor traveled three times roughly the same seven metres long trajectory (consisting of a straight part and an arc), while its ground truth position was recorded by the overhead cameras.

The runs were performed with three different velocities: 0.434 m/s, 0.862 m/s, and 1.175 m/s. For these sequences, the following RPE values were obtained: 0.06 m, 0.11 m, and 0.13 m, respectively. The claim that our front-end can handle rapid motion is also supported by the results obtained for the TUM RGB-D benchmark sequences—some of them, e.g., the `fr1_desk`, contain sudden turns of the handheld sensor. The failure modes of our approach identified during the experiment include situations when there are not enough visual features (e.g., the sensor is approaching a white wall), or there are not enough features with a valid depth (objects are too far from the RGB-D sensor). These situations cannot be handled within the current framework, which requires both the RGB and depth data.

5.3. Comparison with RGB-D SLAM. In order to demonstrate the advantages of our RGB-D data processing methods, we compare the performance of the PUT VO and PUT SLAM systems with the results achieved the RGB-D SLAM system of Endres *et al.* (2012), which is available as open source software and widely considered in the literature as a reference implementation. RGB-D SLAM is feature-based, and may be used in a number of configurations that differ mainly in the detector/descriptor pair being used. For the comparison we use the variants applying SURF or ORB. Although the code of RGB-D SLAM is available,

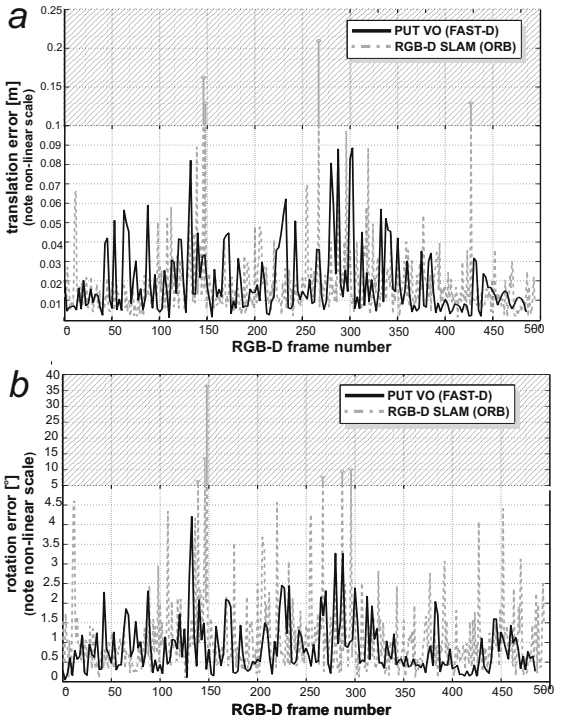


Fig. 10. Comparison of the relative translation (a) and rotation (b) errors along the estimated trajectory for the PUT VO system and the open-source RGB-D SLAM system on the TUM `fr1_desk` sequence.

we found it practically impossible to replace the front-end of this system with our solution, keeping not only the same back-end, but also the original loop closure procedures. Therefore, our comparison concerns not only the front-end, but the whole VO/SLAM solutions.

Figures 10(a) and (b) compare the relative translation and rotation errors of PUT VO and RGB-D SLAM along the estimated trajectory of the TUM RGB-D benchmark `fr1_desk` sequence. One can notice large errors in the relative orientation and translation appearing sporadically in the RGB-D SLAM estimates (hatched areas), whereas PUT VO (i.e., the front-end of PUT SLAM) retains limited relative orientation errors throughout the entire experiment, also significant translation errors occur less frequently.

Figure 11(a) shows the ATE for PUT VO, whereas Fig. 11(b) depicts the ATE of PUT SLAM (using ORB features for loop closure). The trajectory recovered by RGB-D SLAM (also a variant with ORB) and the ATE are demonstrated in Fig. 11(c). From these results we may conclude that for a relatively short sequence the advantage of the RGB-D SLAM system over the much faster PUT VO is not visible. PUT SLAM, using the same g^{2}o back-end as RGB-D SLAM, recovers the trajectory with an even smaller ATE, demonstrating that the improved RGB-D data processing is beneficial also to the self-localization system using pose-graph optimization. However, the fact that the gain in terms

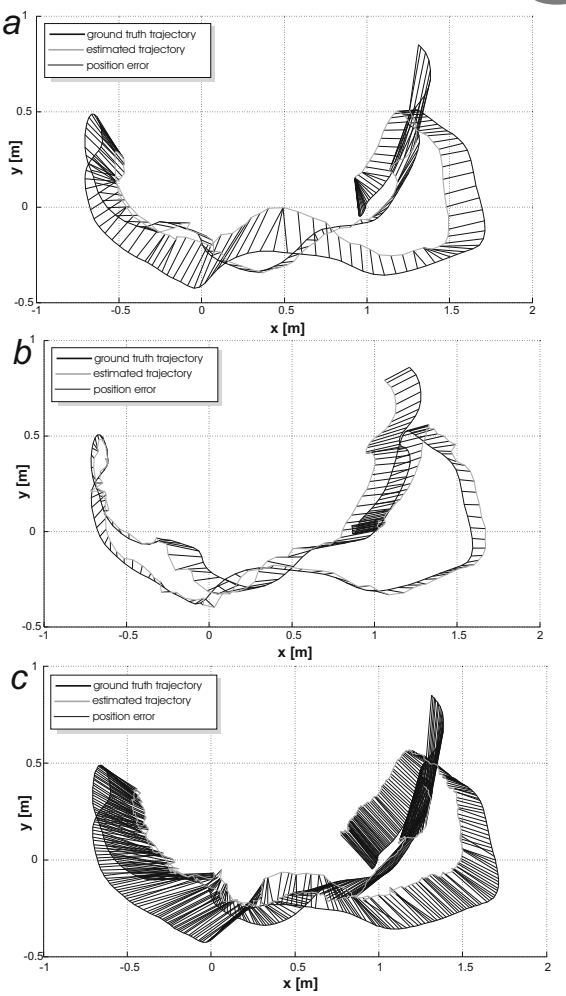


Fig. 11. Comparison of trajectories (with the ATEs visualized) recovered from the TUM `fr1_desk` sequence by the PUT VO system (a), the PUT SLAM system (b), and the open-source RGB-D SLAM system (c).

of the absolute trajectory error due to the loop closures and optimization differs significantly for the particular environments suggests that not all loop closures are correctly detected. We use a simple strategy, which relies completely on the matching of the local descriptors of salient visual features. Perhaps a more advanced strategy, involving active testing of the discovered loop closures against the environment model (Endres *et al.*, 2014), would bring more consistent precision improvements in pose-based SLAM with respect to RGB-D visual odometry alone.

Observations as to the small relative errors incurred by the frame-to-frame motion estimation in PUT VO are confirmed by the quantitative results in Table 1. This table presents the root mean square errors (RMSEs) and the maximum errors of translation and rotation for the same trajectory as Figs. 10 and 11. Table 1 shows also that PUT VO and PUT SLAM are much faster than RGB-D SLAM. The PUT SLAM version achieves a similar speed

Table 1. Comparison of the relative translation errors (RPEs) for the PUT VO/SLAM and RGB-D SLAM systems: the TUM RGB-D benchmark `fr1_desk` sequence.

system and type of feature detector or detector/descriptor	translation RMSE [m]	translation max. error [m]	rotation RMSE [°]	rotation max. error [°]	frames per second [Hz]
PUT VO (FAST-D)	0.038	0.20	1.33	6.50	30.6
PUT SLAM (FAST-D+ORB)	0.027	0.11	1.51	7.78	29.4
PUT SLAM (FAST-D+SURF)	0.032	0.17	1.97	14.37	29.3
RGB-D SLAM (ORB)	0.025	0.21	2.29	36.43	10.5
RGB-D SLAM (SURF)	0.022	0.14	2.15	36.43	10.0

as the VO version, since the pose-graph optimization is implemented in a separate thread and synchronized with the front-end. Thus, a single cycle of loop closure and optimization cannot take more time than motion estimation between two consecutive RGB-D keyframes (including tracking). This constraint ensures real-time operation of the whole PUT SLAM system. Loop closure detection has linear complexity in the number of locations (i.e., keyframes in our case) being considered as the loop closure candidates (Cummins and Newman, 2010); thus, in a general case, the time constraint limits the scale of loops that can be detected. However, in PUT SLAM this limitation is alleviated by the local loop closure formulation: the number of candidate keyframes is already limited by the required maximum Euclidean distance from the current sensor pose. Therefore, in most cases all the candidate keyframes can be evaluated within the tight time period, particularly using the fast ORB computations.

6. Conclusions

The experiments have demonstrated that the extraction of point features from RGB-D data that simultaneously takes into account the images and the depth data increases the robustness and precision of the visual odometry method, used either as a stand-alone self-localization system, or as a front-end in pose-based SLAM. Management of the detected keypoints proved to be important as well, as it allows us to obtain features with the spatial distribution that is well suited for robust estimation of the motion between the RGB-D frames that are located relatively far apart. The proposed FAST-D point feature detector is built upon the solid mathematical analysis of the image formation and general coplanarity tests, but provides a flexible and computation-efficient solution for real-time RGB-D data processing. The robustness and computation speed of the proposed algorithms is of particular importance to the tracking-based visual odometry pipeline, which is very efficient, but only if it is fed with reliable point features.

The proposed PUT VO system is characterized by high precision of the local motion estimation and achieves relative position errors along the trajectory at the same level as for the more complicated RGB-D SLAM system. Our results obtained on the TUM RGB-D benchmark can be also compared to the performance of the new version of RGB-D SLAM (Endres *et al.*, 2014), and results achieved by several visual odometry algorithms, which were compared by Whelan *et al.* (2013). However, the trajectory recovered by PUT VO using the frame-to-frame motion estimation always exhibits some drift. In PUT SLAM a possibility to decrease this drift arises whenever the sensor re-visits already explored areas. It was shown by the experimental results that the pose-based SLAM version of our system also benefits from the improved precision of the front-end, providing that all the pose-graph constraints resulting from the detected loop closures are correct. Any incorrect loop closures have to be rejected by the back-end (Belter *et al.*, 2015), as the front-end is not aware of the whole pose-graph structure and the outcome of the optimization process.

A more advanced strategy for loop closure detection and outlier rejection is one of the main goals of our further research on RGB-D based self-localization. Another promising direction is to directly include some of the point features in the graph optimized by the back-end. The g^2o package makes it possible to include features in the optimization process, but further research is needed on the modelling of the uncertainty of these features.

Acknowledgment

This research was financed through a grant of the National Science Centre, Poland, funded according to the decision DEC-2013/09/B/ST7/01583. An initial version of this work was presented at the special session on *Robotic perception employing RGB-D images* during the 13th National Conference on *Robotics* in Kudowa Zdrój, Poland, 2014.

References

Bachrach, A., Prentice, S., He, R., Henry, P., Huang, A.S., Krainin, M., Maturana, D., Fox, D. and Roy, N. (2012). Estimation, planning, and mapping for autonomous flight using an RGB-D camera in GPS-denied environments, *International Journal of Robotics Research* **31**(11): 1320–1343.

Bączyk, R. and Kasiński, A. (2010). Visual simultaneous localisation and map-building supported by structured landmarks, *International Journal of Applied Mathematics and Computer Science* **20**(2): 281–293, DOI: 10.2478/v10006-010-0021-7.

Bailey, T. and Durrant-Whyte, H. (2006). Simultaneous localization and mapping: Part II, *IEEE Robotics & Automation Magazine* **13**(3): 108–117.

Baker, S. and Matthews, I. (2004). Lucas–Kanade 20 years on: A unifying framework, *International Journal of Computer Vision* **56**(3): 221–255.

Bay, H., Ess, A., Tuytelaars, T. and Van Gool, L. (2008). Speeded-up robust features (SURF), *Computer Vision and Image Understanding* **110**(3): 346–359.

Belter, D., Nowicki, M. and Skrzypczyński, P. (2015). On the performance of pose-based RGB-D visual navigation systems, *in* D. Cremers *et al.* (Eds.), *Computer Vision, ACCV 2014, Part II*, Lecture Notes in Computer Science, Vol. 9004, Springer, Zurich, pp. 1–17.

Bouguet, J.Y. (2000). Pyramidal implementation of the Lucas–Kanade feature tracker, description of the algorithm, *Technical report*, Intel Corp., Microprocessor Research Labs., Pittsburgh, PA.

Choi, S., Kim, T. and Yu, W. (2009). Performance evaluation of RANSAC family, *British Machine Vision Conference, London, UK*.

Cummins, M. and Newman, P. (2010). Accelerating FAB-MAP with concentration inequalities, *IEEE Transactions on Robotics* **26**(6): 1042–1050.

Davison, A.J., Reid, I.D., Molton, N.D. and Stasse, O. (2007). MonoSLAM: Real-time single camera SLAM, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **29**(6): 1052–1067.

Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localization and mapping: Part I, *IEEE Robotics & Automation Magazine* **13**(2): 99–110.

Eggert, D.W., Lorusso, A. and Fisher, R.B. (1997). Estimating 3-D rigid body transformations: A comparison of four major algorithms, *Machine Vision and Applications* **9**(5–6): 272–290.

Endres, F., Hess, J., Engelhard, N., Sturm, J., Cremers, D. and Burgard, W. (2012). An evaluation of the RGB-D SLAM system, *IEEE International Conference on Robotics and Automation, St. Paul MN, USA*, pp. 1691–1696.

Endres, F., Hess, J., Sturm, J., Cremers, D. and Burgard, W. (2014). 3-D mapping with an RGB-D camera, *IEEE Transactions on Robotics* **30**(1): 177–187.

Engel, J., Sturm, J. and Cremers, D. (2012). Camera-based navigation of a low-cost quadrocopter, *IEEE/RSJ International Conference on Intelligent Robots & Systems, Vilamoura, Portugal*, pp. 2815–2821.

Ester, M., Kriegel, H.-P., Sander, J. and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise, *International Conference on Knowledge Discovery and Data Mining, Portland, OR, USA*, pp. 226–231.

Hansard, M., Lee, S., Choi, O. and Horaud, R. (2012). *Time-of-Flight Cameras: Principles, Methods and Applications*, Springer, Berlin.

Hartley, R.I. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*, 2nd Edn., Cambridge University Press, Cambridge.

Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A. and Fitzgibbon, A. (2011). KinectFusion: Real-time 3D reconstruction and interaction using a moving depth camera, *ACM Symposium on User Interface Software and Technology, New York, NY*, pp. 559–568.

Kerl, C., Sturm, J. and Cremers, D. (2013). Robust odometry estimation for RGB-D cameras, *IEEE International Conference on Robotics and Automation, Karlsruhe, Germany*, pp. 3748–3754.

Khoskelham, K. and Elberink, S.O. (2012). Accuracy and resolution of Kinect depth data for indoor mapping applications, *Sensors* **12**(2): 1437–1454.

Kraft, M., Nowicki, M., Schmidt, A. and Skrzypczyński, P. (2014). Efficient RGB-D data processing for point-feature-based self-localization, *in* C. Zieliński and K. Tchoń (Eds.), *Postępy robotyki*, PW, Warsaw, pp. 245–256, (in Polish).

Kuemmerle, R., Grisetti, G., Strasdat, H., Konolige, K. and Burgard, W. (2011). g2o: A general framework for graph optimization, *IEEE International Conference on Robotics and Automation, Shanghai, China*, pp. 3607–3613.

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints, *International Journal of Computer Vision* **60**(2): 91–110.

Mertens, L., Penne, R. and Ribbens, B. (2013). Time of flight cameras (3D vision), *in* J. Buytaert (Ed.), *Recent Advances in Topography*, Engineering Tools, Techniques and Tables, Nova Science, Hauppauge, NY, pp. 353–417.

Nascimento, E., Oliveira, G., Campos, M.F.M., Vieira, A. and Schwartz, W. (2012). BRAND: A robust appearance and depth descriptor for RGB-D images, *IEEE/RSJ International Conference on Intelligent Robots & Systems, Vilamoura, Portugal*, pp. 1720–1726.

Nowicki, M. and Skrzypczyński, P. (2013a). Combining photometric and depth data for lightweight and robust visual odometry, *European Conference on Mobile Robots (ECMR), Barcelona, Spain*, pp. 125–130.

Nowicki, M. and Skrzypczyński, P. (2013b). Experimental verification of a walking robot self-localization system with the Kinect sensor, *Journal of Automation, Mobile Robotics & Intelligent Systems* **7**(4): 42–51.

Nüchter, A., Lingemann, K., Hertzberg, J. and Surmann, H. (2007). 6D SLAM—3D mapping outdoor environments, *Journal of Field Robotics* **24**(8–9): 699–722.

Penne, R., Mertens, L. and Ribbens, B. (2013). Planar segmentation by time-of-flight cameras, *in* J. Blanc-Talon *et al.* (Eds.), *Advanced Concepts for Intelligent Vision Systems*, Lecture Notes in Computer Science, Vol. 8192, Springer, Berlin, pp. 286–297.

Penne, R., Raposo, C., Mertens, L., Ribbens, B. and Araujo, H. (2015). Investigating new calibration methods without feature detection for ToF cameras, *Image and Vision Computing* **43**: 50–62.

Raguram, R., Chum, O., Pollefeys, M., Matas, J. and Frahm, J. (2013). USAC: A universal framework for random sample consenus, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **35**(8): 2022–2038.

Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection, *9th European Conference on Computer Vision (ECCV'06), Graz, Austria*, pp. 430–443.

Rublee, E., Rabaud, V., Konolige, K. and Bradski, G. (2011). ORB: an efficient alternative to SIFT or SURF, *IEEE International Conference on Computer Vision (ICCV), Barcelona, Spain*, pp. 2564–2571.

Rusu, R., Blodow, N., Marton, Z. and Beetz, M. (2008). Aligning point cloud views using persistent feature histograms, *IEEE/RSJ International Conference on Intelligent Robots & Systems, Nice, France*, pp. 3384–3391.

Scaramuzza, D. and Fraundorfer, F. (2011). Visual odometry, Part I: The first 30 years and fundamentals, *IEEE Robotics & Automation Magazine* **18**(4): 80–92.

Schmidt, A., Fularz, M., Kraft, M., Kasiński, A. and Nowicki, M. (2013a). An indoor RGB-D dataset for the evaluation of robot navigation algorithms, *in* J. Blanc-Talon *et al.* (Eds.), *Advanced Concepts for Intelligent Vision Systems*, Lecture Notes in Computer Science, Vol. 8192, Springer, Berlin, pp. 321–329.

Schmidt, A., Kraft, M., Fularz, M. and Domagala, Z. (2013b). The comparison of point feature detectors and descriptors in the context of robot navigation, *Journal of Automation, Mobile Robotics & Intelligent Systems* **7**(1): 11–20.

Segal, A., Haehnel, D. and Thrun, S. (2009). Generalized-ICP, *Robotics: Science and Systems, Seattle, WA, USA*.

Shi, J. and Tomasi, C. (1994). Good features to track, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94), Seattle, WA, USA*, pp. 593–600.

Skrzypczyński, P. (2009). Simultaneous localization and mapping: A feature-based probabilistic approach, *International Journal of Applied Mathematics and Computer Science* **19**(4): 575–588, DOI: 10.2478/v10006-009-0045-z.

Steder, B., Rusu, R.B., Konolige, K. and Burgard, W. (2011). Point feature extraction on 3D range scans taking into account object boundaries, *IEEE International Conference on Robotics and Automation, Shanghai, China*, pp. 2601–2608.

Steinbrücker, F., Sturm, J. and Cremers, D. (2011). Real-time visual odometry from dense RGB-D images, *Workshop on Live Dense Reconstruction with Moving Cameras/ International Conference on Computer Visision, Barcelona, Spain*, pp. 719–722.

Stewénius, H., Engels, C. and Nistér, D. (2006). Recent developments on direct relative orientation, *ISPRS Journal of Photogrammetry and Remote Sensing* **60**(4): 284–294.

Stoyanov, T., Louloudi, A., Andreasson, H. and Lilienthal, A. (2011). Comparative evaluation of range sensor accuracy in indoor environments, *5th European Conference on Mobile Robots, Örebro, Sweden*, pp. 19–24.

Strasdat, H. (2012). *Local Accuracy and Global Consistency for Efficient Visual SLAM*, Ph.D. thesis, Imperial College, London.

Sturm, J., Engelhard, M., Endres, F., Burgard, W. and Cremers, D. (2012). A benchmark for the evaluation of RGB-D SLAM systems, *IEEE/RSJ International Conference on Intelligent Robots & Systems, Vilamoura, Portugal*, pp. 573–580.

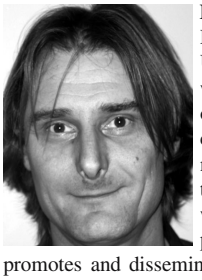
Umeyama, S. (1991). Least-squares estimation of transformation parameters between two point patterns, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **13**(4): 376–380.

Whelan, T., McDonald, J., Kaess, M., Fallon, M., Johannsson, H. and Leonard, J. (2012). Kintinuous: Spatially extended KinectFusion, *Robotics: Science and Systems, Sydney, Australia*.

Whelan, T., Johannsson, H., Kaess, M., Leonard, J. and McDonald, J. (2013). Robust real-time visual odometry for dense RGB-D mapping, *IEEE International Conference on Robotics and Automation, Karlsruhe, Germany*, pp. 5724–5731.

Marek Kraft graduated from the Poznań University of Technology in 2005. He received the Ph.D. degree from the same university in 2013. In the same year he was appointed an assistant professor at the Institute of Control and Information Engineering there. His research interests include computer vision, embedded systems, robotics, parallel processing and high performance computing.

Michał Nowicki graduated from the Poznań University of Technology, receiving a B.Sc. and an M.Sc. in automatic control and robotics in 2013 and 2014, respectively, and a B.Sc. in computer science from the same university in 2014. He is a Ph.D. student and a research assistant at the Institute of Control and Information Engineering. His research interests include computer vision, simultaneous localization and mapping, walking robots, and Android-based navigation.



Rudi Penne graduated (1986) and received his D.Sc. degree (1992) in mathematics from the University of Antwerp (Belgium). There he was appointed a professor (2013) in the Faculty of Applied Engineering. His research interests cover a broad area, including topology, combinatorial and computational geometry, structural topology and rigidity, and projective geometry with applications in robot kinematics and computer vision. Besides fundamental research, he promotes and disseminates science by means of popularizing articles, books, blogs and lectures.

Adam Schmidt received his M.Sc. and Ph.D. degrees in automatic control and robotics from the Poznań University of Technology in 2007 and 2014, respectively. In 2014 he was appointed an assistant professor at the Institute of Control and Information Engineering of the PUT. His research interests include computer vision, simultaneous localization and mapping and machine learning.

Piotr Skrzypczyński graduated from the Poznań University of Technology (1993). He received the Ph.D. and D.Sc. degrees in robotics from the same university in 1997 and 2007, respectively. He is an associate professor at the Institute of Control and Information Engineering of the PUT, and the head of the Mobile Robotics Laboratory. His research interests include autonomous mobile robots, simultaneous localization and mapping, computer vision, multi-sensor systems, and computational intelligence methods in robotics.

Received: 28 September 2014
Revised: 16 April 2015