УДК 004.932.2
DOI: 10.20535/SRIT.2308-8893.2017.4.02

# 3D-MODEL RECONSTRUCTION WITH USE OF MONOCULAR RGB CAMERA

**O.V. VEDMEDENKO, S.S. NIKOLAIEV, Y.A. TYMOSHENKO**

**Abstract.** Every year the edge between the real and digital worlds is becoming more and more blurred. Augmented and virtual reality rapid development creates new opportunities for more productive work and entertainment, revolution in 3D printing technologies begets boost in multiple DIY communities appearance and sharing economy growth. All these factors require new technologies that allow making 3D models from real world objects, but most of these solutions are either very expensive or require complex technical knowledge that most ordinary people do not have. This paper provides a review and comparison of modern methods for 3D models of physical objects real time reconstruction that can be used in present-day mobile solutions.

**Keywords:** 3D model, 3D object, simultaneous localization and mapping problem, SLAM, monocular camera, RGB camera, LSD-SLAM, ORB-SLAM.

## INTRODUCTION

Nowadays a rapid transition from two-dimensional to three-dimensional information space can be observed: flat interfaces are becoming less popular, three-dimensional printing technology has achieved great success in a wide range of different industries, virtual and augmented reality are no longer only ideas of science fiction.

Each of the examples above requires a certain electronic volumetric representation — a 3D-model. 3D interface requires a layout in space, 3D printing technology requires a model of a 3D object, virtual and augmented reality are both, in essence, just a manipulation and display of various 3D models.

However, the creation of three-dimensional content is a very difficult and painstaking work. Designers and 3D illustrators spend a lot of resources to develop even a simple model despite the fact that we see many of them in our daily lives, and those we don't see are often only modifications of objects from the real world. We use the following methods directly to solve such problems. Those methods provide a possibility to turn your smartphone into a monocular 3D scanner — it's very convenient to have a model, which is ready for further modification and use of other software, just a few minutes after you saw it.

## METHODS CLASSIFICATION

Today, having special knowledge, 3D model can be reconstructed with a number of methods that solve the problem described in the literature as the problem of simultaneous localization and mapping (i.e. SLAM). There are many different implementations of this problem and they can be classified by:

1. The way algorithm analyze image information:
   a. Ones which do not use features or key points (feature-less);
   b. Ones which do use features (feature-based).
2. The type of sensor used to capture images: RGB camera, RGB-D camera [1], laser range finders [2].
3. The number of cameras the video stream is received from:
   a. Ones which use one camera (monocular);
   b. Ones which use two or more cameras (stereo).

## ORB-SLAM METHOD

ORB-SLAM [3] is a feature-based monocular SLAM system which can operate in real time, in all kinds of environments. The system is robust to sharp and fast motion, allows fast loop closing algorithm and relocalization algorithm. Also it includes automatic initialization. It uses a survival of the fittest strategy that aimed to select unique points and key frames, which leads to excellent robustness and generates a compact map that grows only when environment varies in time.

Main design idea of ORB-SLAM system is that the same remembered key features are used by the mapping, tracking, and for place recognition to perform relocalization and loop closing. This allows to avoid the need to calculate the depths of recognized features. ORB-SLAM system requires algorithm which can extract features from image in less than 33ms per image, which excludes the popularSIFT (~300 ms) [4], SURF (~300 ms) [5] or the recent A-KAZE (~100 ms) [6]. To obtain general place recognition capabilities, it also requires rotation invariance, which excludes BRIEF [7] and LDB [8].

ORB-SLAM uses ORB [9] method. It is extremely fast, while it has good invariance to viewpoint. This boosts the accuracy of bundle adjustment.

ORB-SLAM system incorporates three components which are running simultaneously: tracking component, local mapping component and loop closing component.

The tracking component is responsible for localizing the camera with every frame and deciding when to create a new key frame. Algorithm performs feature comparison between current and previous frame and optimize the pose using motion-only bundle adjustment. If the tracking for any reason is lost (e.g. because of occlusions or sharp movements), the place recognition module is responsible to perform a global relocalization. When there is an initial estimation of the camera position and feature matchings, a local visible map is reclaimed using the covisibility graph consisting of key frames that is remembered by the system. Then matches with the local map points are searched by reprojection, and camera position is optimized again with all found matches. Finally the tracking thread is inserting a new key frame if needed.

All visual SLAM works in the literature agree that running bundle adjustment with all the points and all the frames is not feasible. The work of Strasdat et al. [10] showed that the most cost effective approach is to keep as much points as possible, while keeping only non-redundant key frames. One approach is to insert key frames very cautiously to avoid an excessive growth of the computational complexity. This restrictive key frame insertion algorithm makes the tracking fail in hard exploration conditions. ORB-SLAM uses survival of the fittest strategy which achieves unprecedented robustness in difficult scenarios by inserting key frames as quickly as possible, and removing later the redundant ones, to avoid the extra cost. This permits a flexible map expansion during exploration, which increases tracking robustness in hard conditions (e.g. camera rotation, fast and sharp movements), while its size is bounded in continual revisits to the same environment, i.e. lifelong operation.

The local mapping is responsible for processing new key frames added by previous components and performs local bundle adjustment, which leads to optimal reconstruction in areas, close to camera position. It is looking for a new features in all new key frames by a comparison to connected key frames in the map, and when it founds new feature it generates a new point. Based on the information gathered during the work of the system, special algorithm is applied to cut all redundant remembered points and save only most informative ones. Also it cuts all redundant key frames.

Each key frame $K_i$ stores:

1. The camera pose $T_{iw}$, which is a rigid body transformation that transforms points from the world to the camera coordinate system.

2. The camera intrinsics, including focal length and principal point.

3. All the ORB features extracted in the frame, associated or not to a map point, whose coordinates are undistorted if a distortion model is provided.

Each map point $p_i$ stores:

1. Its 3D position $X_{w,i}$ in the world coordinate system.

2. The viewing direction $n_i$, which is the mean unit vector of all its viewing directions (the rays that join the point with the optical center of the key frames that observe it).

3. A representative ORB descriptor $D_i$, which is the associated ORB descriptor whose hamming distance is minimum with respect to all other associated descriptors in the key frames in which the point is observed.

4. The maximum $d_{\max}$ and minimum $d_{\min}$ distances at which the point can be observed, according to the scale invariance limits of the ORB features.

The loop closing searches for loops with every new key frame. If a loop is detected, it computes a similarity transformation that informs about the drift accumulated in the loop. Then both sides of the loop are aligned and duplicated points are merged. Finally algorithm performs a pose graph optimization over similarity constraints to achieve global consistency.

Running ORB-SLAM system is demonstrated on fig. 1. At the bottom left corner there is a current frame with features highlighted with green. At the right side there is a built map: remembered features displayed with red dots, green square is a current camera position and blue squares are key frames.

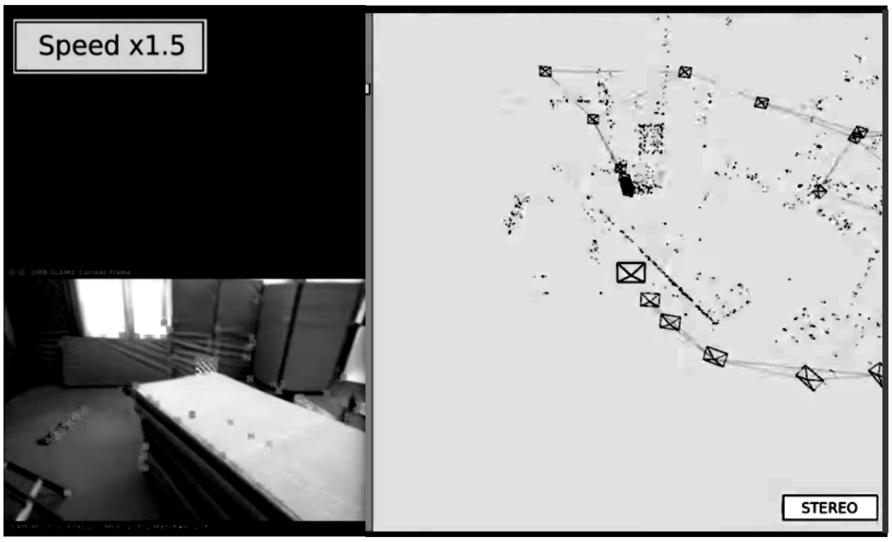

*Fig 1.* Running ORB-SLAM system

## LSD-SLAM METHOD

Large-Scale Direct Monocular SLAM (LSD-SLAM) [11] — a direct (feature-less) monocular SLAM algorithm. Along with highly accurate pose estimation based on direct image alignment, it provides the 3D environment real-time reconstruction as a graph of key frames with semi-dense depth maps. These are obtained by a huge number of pixel-by-pixel comparisons.

The fundamental idea behind feature-based approaches is to split the overall problem — estimating geometric information from images — into two sequential steps: first, a set of feature observations is extracted from the image. Second, the camera position and scene geometry is computed as a function of these feature observations only. While this decoupling simplifies the overall problem, it comes with an important limitation: only information that conforms to the feature type can be used. In particular, when using keypoints, information contained in straight or curved edges — which especially in man-made environments make up a large part of the image — is discarded.

Direct visual odometry methods circumvent this limitation by optimizing the geometry directly on the image intensities, which enables using all information in the image. In addition to higher accuracy and robustness in particular in environments with little keypoints, this provides substantially more information about the geometry of the environment, which can be very valuable for robotics or augmented reality applications.

A condensed summary of the relevant mathematical concepts is given in the next paragraphs.

$\Omega \subset \mathbb{R}^2$ is the set of normalized pixel coordinates, i.e., they include the intrinsic camera calibration.

$d$ is used to denote the inverse of the depth $z$ of a point, i.e., $d = z^{-1}$.

$SO(3)$ is a rotations around a fixed point in three-dimensional Euclidean space group, which consists of orthogonal $3 \times 3$ matrices with a determinant equals to 1.

A 3D rigid body transform $G \in SE(3)$ denotes rotation and translation in 3D, i.e. is defined by

$$G = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \text{ with } R \in SO(3) \text{ and } t \in \mathbb{R}^3 .$$

During optimization, a minimal representation for the camera pose is required, which is given by the corresponding element $\xi \in SE(3)$ of the associated Lie-algebra. Elements are mapped to $SE(3)$ by the exponential map $G = \exp_{se(3)}(\xi)$, its inverse being denoted by $\xi = \log_{SE(3)}(G)$. With a slight abuse of notation, element of $se(3)$ can be used to represent pose and written as vector $\xi \in R^6$.

The transformation moving a point from frame i to frame j is written as $\xi_{ji}$. For convenience, the pose concatenation operator $\circ : se(3) \times se(3) \to se(3)$ should be defined as

$$\xi_{ki} = \xi_{kj} \circ \xi_{ji} = \log_{SE(3)}(\exp_{se(3)}(\xi_{kj}) \exp_{se(3)}(\xi_{ji})) .$$

Further, 3D projective warp function $\omega$ should be defined, which projects an image point p and its inverse depth d into a by $\xi$ transformed camera frame

$$\omega(p, d, \xi) = \begin{pmatrix} \frac{x'}{z'} \\ \frac{y'}{z'} \\ \frac{1}{z'} \end{pmatrix} \text{ with } \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \exp_{se(3)} \begin{pmatrix} \frac{p_x}{d} \\ \frac{p_y}{d} \\ \frac{1}{d} \\ 1 \end{pmatrix} .$$

A 3D similarity transform $S \in \mathrm{Sim}(3)$ denotes rotation, scaling and translation, i.e. is defined by

$$G = \begin{pmatrix} sR & t \\ 0 & 1 \end{pmatrix} \text{ with } R \in SO(3),\ t \in \mathbb{R}^3 \text{ and } s \in \mathbb{R}^+ .$$

As for rigid body transformations, a minimal representation is given by elements of the associated Lie-algebra $\xi \in \mathrm{sim}(3)$, which now have an additional degree of freedom, that is $\xi \in R^7$. The exponential and logarithmic map, pose concatenation and a projective warp function $\omega_s$ can be defined analogously to the $se(3)$ case.

Propagation of uncertainty is a statistical tool to derive the uncertainty of the output of a function $f(X)$, caused by uncertainty on its input $X$. Assuming $X$ to be Gaussian distributed with covariance $\Sigma_X$, the covariance of $f(X)$ can be approximated (using the Jacobian $J_f$ of f ) by

$$\Sigma_f \approx J_f \Sigma_X J_f^{\mathrm{T}} .$$

The algorithm consists of three major components: tracking, depth map estimation and map optimization.

The tracking component continuously tracks new camera images. That is, it estimates their rigid body pose $\xi \in se(3)$ with respect to the current key frame, using the pose of the previous frame as initialization.

The depth map estimation component uses tracked frames to either refine or replace the current key frame. Depth is refined by filtering over many per-pixel, small-baseline stereo comparisons coupled with interleaved spatial regularization. If the camera moves too far away from the existing map, a new key frame is created from the most recent tracked image. Algorithm threshold a weighted combination of relative distance and angle to the current key frame:

$$\mathrm{dist}(\xi_{ji}) = \xi_{ji}^{\mathrm{T}} W \xi_{ji},$$

where $W$ is a diagonal matrix containing the weights.

Each key frame $K_i$ consists of a camera image $I_i : \Omega_i \to \mathbb{R}$, an inverse depth map $D_i : \Omega_{D_i} \to \mathbb{R}^+$, and the variance of the inverse depth $V_i : \Omega_{D_i} \to$ $\to \mathbb{R}^+$. Note that the depth map and variance are only defined for a subset of pixels $\Omega_{D_i} \subset \Omega_i$ , containing all image regions in the vicinity of sufficiently large intensity gradient, hence semi-dense. Edges $E_{ji}$ between key frames contain their relative alignment as similarity transform $\xi_{ji} \in \mathrm{sim}(3)$, as well as the corresponding covariance matrix $\Sigma_{ji}$.

Once a key frame is replaced as tracking reference — and hence its depth map will not be refined further — it is incorporated into the global map by the map optimization component. To detect loop closures and scale-drift, a similarity transform $\xi \in \mathrm{sim}(3)$ to close-by existing key frames (including its direct predecessor) is estimated using scale-aware, direct $\mathrm{sim}(3)$ — image alignment.

To bootstrap the LSD-SLAM system, it is sufficient to initialize a first key frame with a random depth map and large variance. Given sufficient translational camera movement in the first seconds, the algorithm “locks” to a certain configuration, and after a couple of key frame propagations converges to a correct depth configuration.

Starting from an existing key frame $K_i = (I_i, D_i, V_i)$, the relative 3D pose $\xi_{ij} \in se(3)$ of a new image $I_j$ is computed by minimizing the variance-normalized photometric error

$$E_p(\xi_{ji}) = \sum_{p \in \Omega_{D_i}} \left\| \frac{r_p^2(p, \xi_{ji})}{\sigma^2_{r_p(p,\xi_{ji})}} \right\|_\delta ,$$

$$\text{with } \; r_p(p, \xi_{ji}) = I_i(p) - I_j(\omega(p, D_i(p), \xi_{ji})),$$

$$\sigma^2_{r_p(p,\xi_{ji})} = 2\sigma_I^2 + \left( \frac{\partial r_p(p, \xi_{ji})}{\partial D_i(p)} \right)^2 V_i(p),$$

where $\|\cdot\|_\delta$ is the Huber norm applied to the normalized residual

$$r_\delta = \begin{cases} \dfrac{r^2}{2\delta}, & \text{if } |r| \le \delta, \\ |r| - \dfrac{\delta}{2} & \text{otherwise}. \end{cases}$$

The residual's variance $\sigma^2_{r_p(p,\xi_{ji})}$ is computed using covariance propagation as described above, utilizing the inverse depth variance $V_i$ and assuming Gaussian image intensity noise $\sigma_I^2$.

Running LSD-SLAM system is demonstrated on fig. 2. At the top left corner there is a current frame captured by the camera. At the bottom left there is a current key frame with color-coded depth map (from red — close objects, to blue — far objects). At the right side there is a built point cloud with red square as a current camera position and blue ones as camera trajectory.

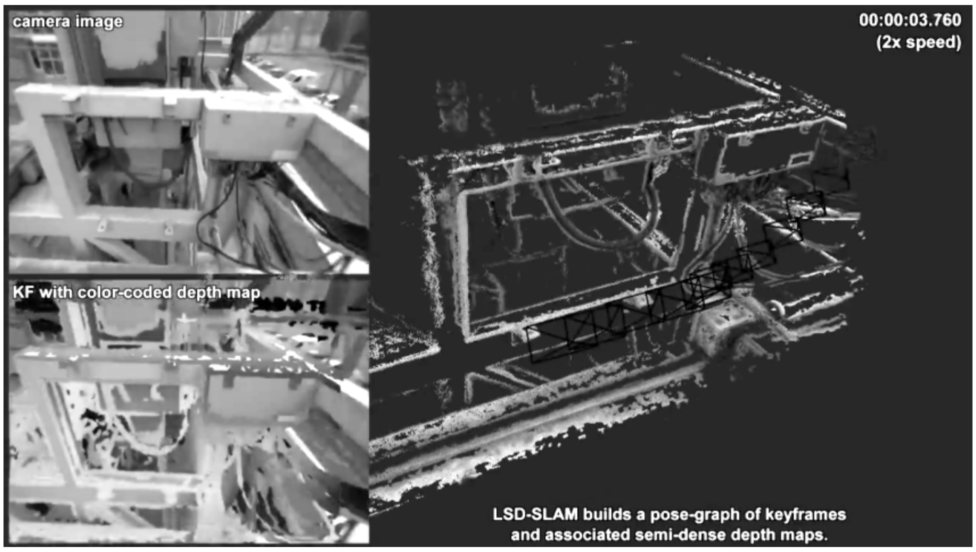
camera image
KF with color-coded depth map
00:00:03.760
(2x speed)
LSD-SLAM builds a pose-graph of keyframes
and associated semi-dense depth maps.

*Fig 2.* Running LSD-SLAM system

The authors of this work were able to obtain promising results using the LSD-SLAM method (see fig. 3–6).

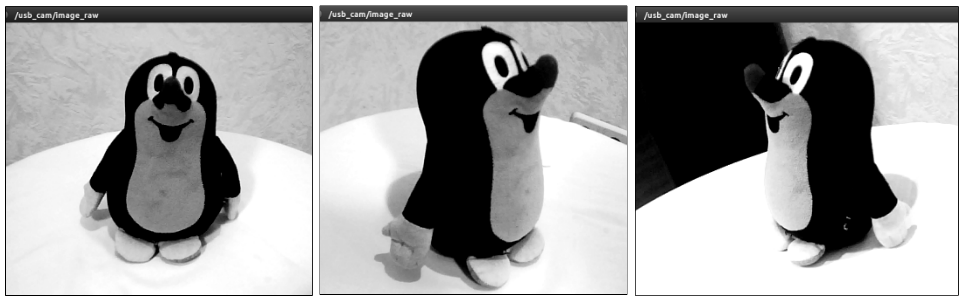

*Fig 3.* Experiment 1. Object

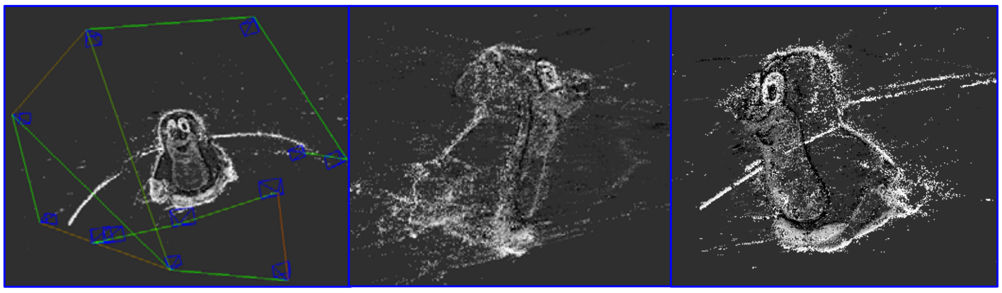

*Fig. 4.* Experiment 1. Result

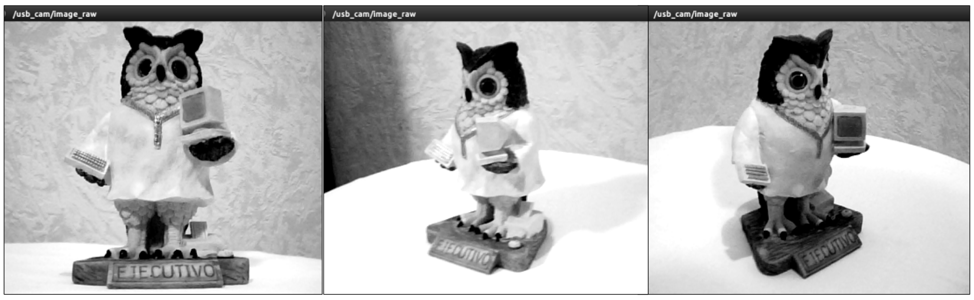

*Fig. 5.* Experiment 2. Object

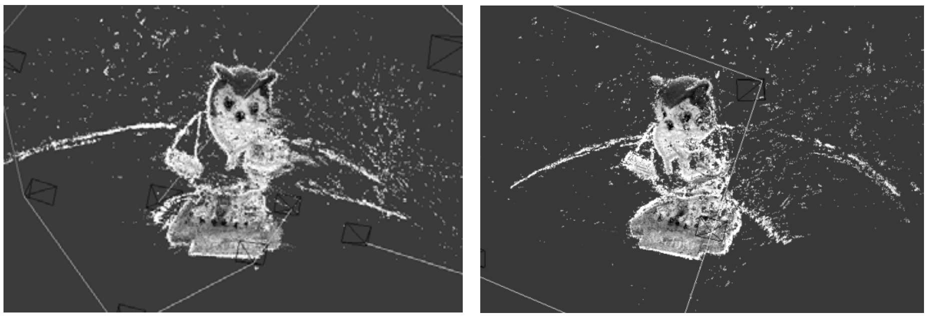

*Fig. 6.* Experiment 2. Result

## METHODS COMPARISON

It's convenient to compare those methods with a following table.

LSD-SLAM compared to ORB-SLAM

| Method | Trajectory loss | Relocalization | Point cloud | Speed of work |
|---|---|---|---|---|
| LSD-SLAM | Method is unstable, trajectory is often lost | Slow and not precise | Monochrome point cloud from RGB camera | Works in real time, but requires fast GPU |
| ORB-SLAM | Method is robust, trajectory is lost only for very sharp movements in the direction of non-scanned areas | Fast and precise | Colored point cloud only from RGB-D camera | Works in real time, fast GPU is not required |

## CONCLUSIONS

The analysis of these methods and their comparison give the following results. If the task is to only localize camera position and build movement trajectory it is clearly necessary to choose ORB-SLAM. In case the user can handle RGB-D camera it is also possible to build colored point cloud representing the three-dimensional model of the environment or object. Instead, if it is crucial to build a 3D model a user needs to choose LSD-SLAM even though the scans may have worse quality.

## REFERENCES

1. *An evaluation* of the RGB-D SLAM system / F. Endres, J. Hess, N. Engelhard etc. // Robotics and Automation (ICRA). — 2012. // IEEE International Conference. — P. 1691–1696.
2. *Robot* pose estimation in unknown environments by matching 2d range scans / Lu F., Milios E. // Journal of Intelligent and Robotic Systems. — 1997. — Vol. 18, № 3. — P. 249–275.
3. *ORB-SLAM*: A Versatile and Accurate Monocular SLAM System / Raúl Mur-Artal, J. M. M. Montiel, Juan D. Tardós // IEEE Transactions on Robotics. — 2015. — Vol. 31, N 5. — P. 1147–1163.
4. *Distinctive* image features from scale-invariant keypoints / D. G. Lowe // International Journal of Computer Vision. — 2004. — Vol. 60, N 2. — P. 91–110.
5. *SURF*: Speeded Up Robust Features / H. Bay, T. Tuytelaars, and L. Van Gool // European Conference on Computer Vision (ECCV). — 2006. — P. 404–417.
6. *Fast* explicit diffusion for accelerated features in nonlinear scale spaces / P. F. Alcantarilla, J. Nuevo, and A. Bartoli // British Machine Vision Conference (BMVC). — 2013. — P. 1–11.
7. *BRIEF*: Binary Robust Independent Elementary Features / M. Calonder, V. Lepetit, C. Strecha, and P. Fua // European Conference on Computer Vision (ECCV). — 2010. — P. 778–792.
8. *LDB*: An ultra-fast feature for scalable augmented reality on mobile devices / X. Yang and K.-T. Cheng // IEEE International Symposium on Mixed and Augmented Reality. — 2012. — P. 49–57.
9. *ORB*: an efficient alternative to SIFT or SURF / E. Rublee, V. Rabaud, K. Konolige, G. Bradski // IEEE International Conference on Computer Vision (ICCV). — 2011. — P. 2564–2571.
10. *Visual* SLAM: Why filter? / H. Strasdat, J. M. M. Montiel, A. J. Davison // Image and Vision Computing. — 2012. — Vol. 30, № 2. — P. 65–77.
11. *LSD*-SLAM: Large-Scale Direct Monocular SLAM / J. Engel, T. Schöps, D. Cremers // European Conference on Computer Vision (ECCV). — 2014. — P. 834–849.

*Received 23.06.2017*

From the Editorial Board: the article corresponds completely to submitted manuscript.