Ročník 2011 electroscope Číslo II

SCENE RECONSTRUCTION FROM KINECT MOTION

M. Šolony [1], P. Zemčík [2]
[1] Department of Computer Graphics and Multimedia, Faculty of Information Technology, VUT v Brně, Božetěchová 2, Brno
[2] Faculty of Information Technology, VUT v Brně, Božetěchová 2, Brno

E-mail: isolony@fit.vutbr.cz, zemcik@fit.vutbr.cz

Abstract:
This paper demonstrates the capabilities of the Kinect [1] device for the purpose of building dense 3D map of the small indoor environments. The range scans from this device provide the information about the 3D structure of scene in the form of 3D point clouds. The alignment problem of these 3D points is solved by tracking the camera movement using the computer vision algorithms, so the exact camera position and rotation known in every time frame can be used to reconstruct a consistent map from multiple Kinect depth images. The purpose of this method is to effectively produce dense 3D maps of small workspaces.

INTRODUCTION

The sensors capable of capturing the depth information from the scene, such as RGB-D cameras, have been available for years but because of their high price they haven't been part of research outside the specialized groups. Recently, the Kinect devices became available for the wide user-base, and because of its relatively low price they became the center of research of many universities and computer vision groups.

The Kinect is a peripheral attachment primarily manufactured for XBox360 that combines standard RGB camera, depth camera consisting of IR projector and IR camera and microphones. This device can be used also outside of the Xbox360 system using open source drivers. This paper presents a simple scene reconstruction algorithm for small workspaces using the Kinect camera as main input device.

The Kinect RGB camera is able to capture the image at the resolution 640x480 pixels at steady 30 frames per second. The per-pixel depth information is acquired by projecting highly unstructured IR pattern from the IR projector located on the device and triangulating against known pattern. The depth map computation is performed internally by the device and the results can be accessed through camera drivers. The depth map is a 2D image, which holds the information about the 3D structure of the scene and can be simply transformed into 3D point cloud. The RGB camera can be used to provide information for the estimation of the relative camera movement between time frames and also color information of 3D points. The main advantage of using Kinect camera is the presence of IR camera which allows the dense reconstruction of dark or plain textured surfaces. However, this device works only within short range (maximum effective range is 6m) and in the comparison with laser scanners, it has much narrower field of view.

3D RECONSTRUCTION OVERVIEW

The scene structure reconstruction plays an important role in the field of 3D reconstruction, mobile robot navigation or augmented reality. The main idea of the reconstruction from multiple scans of the scene is to find the spatial information between consecutive frames, and align the 3D data from each frame to create consistent structure. The reconstructed scene consists of large number of 3D points, which can be further processed to obtain detailed surface. Many approaches have been developed to address the problem of scene reconstruction and map building. Most approaches involve various sensors such as range scanners [2][3],

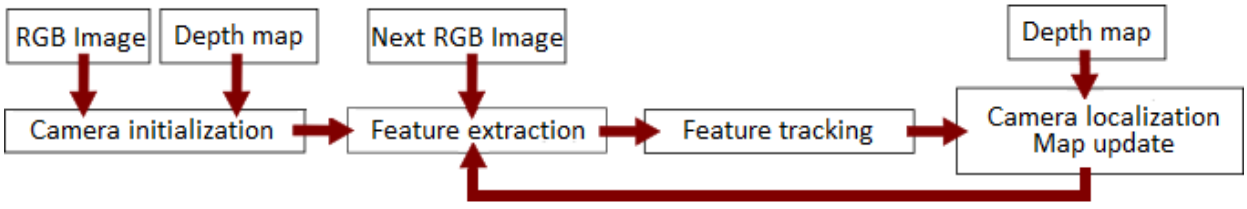


Fig. 1: Scheme of the 3D reconstruction algorithm.

stereo cameras [4] or monocular cameras [5].

The scheme in Fig. 1 shows the components and the overview of the proposed algorithm. The 3D structure of the parts of the scene will be computed from depth maps provided by Kinect depth camera and the global map will be composed by aligning those individual parts. For the alignment, the change in the camera position has to be tracked, and used to transform each reconstructed part of the scene to world coordinate system.

Comparing to the modern map-building algorithms [3], our solution supposes no uncertainty in camera position so it is not as robust as Simultaneous Localization and Mapping (SLAM) [5].

CAMERA LOCALIZATION

Our mapping method is based on the tracking of moving camera in the static environment and aligning and merging the range scans into one map. In each time frame, the position and rotation of the camera with the respect to world coordinate system has to be known to ensure consistent insertion of 3D points. For this task, we decided to track sparse set of feature points with known 3D position. The 2D positions of these points have to be tracked in each consecutive image frame and from the change in their 2D positions the actual position and rotation of camera is updated.

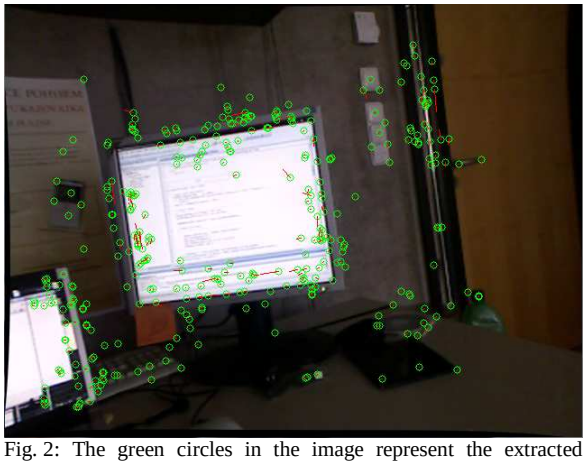


Fig. 2: The green circles in the image represent the extracted feature points and the red lines show their movement with the respect to the previous frame.

To extract set of visual feature points, SURF [9] algorithm has been applied, and for the tracking of these features, KLT tracker [10] has been employed (Fig. 2). The accuracy of the computation of the 3D position of camera depends on the accuracy of feature tracking, so the correspondence of the point pairs has to be verified to ensure better results.

To prevent the false matches to be used for the computation of camera, we can exploit the RANSAC [11] algorithm and epipolar constraints [6] to check the validity of point matches. The RANSAC algorithm uses the random subset of the point matches to compute the parameters of the model. In this case, the model consists of fundamental matrix *F*, which is a 3x3 matrix describing the relations between every point *p* from first image and the corresponding point *p'* in second image. In equation (4), *Fp* describes a line on which the corresponding point *p'* must lie. The fundamental matrix can be estimated given at least seven point correspondences. The model that satisfies most of the point matches is used to determine the inliers (good matches) and outliers (false matches).

$$p'^{T} Fp = 0 \qquad (4)$$

According to the point matches, sets of 2D and their corresponding 3D positions (determined from Kinect depth map) can be built. To estimate the pose of the camera, these correspondences are used to create the system of equations which relate the 3D coordinates of the points with their 2D image coordinates. This algorithm can be formulated as a non-linear least squares problem, which minimizes the reprojection error *d*, i.e. the sum of squared distances between the observed points and the points projected to 2D camera plane using estimated camera pose and known intrinsic parameters:

$$\hat{\theta} = \arg\min_{\theta} \sum_{i=1}^{m} (r_i(\theta))^2 \qquad (5)$$

m is number of correspondences, function $r_i(\theta)$ represents the reprojection error $d = r_i(\theta)^2 = r_x^2 + r_y^2$ and θ are six camera pose parameters, three for translation and three for rotation around world coordinate system axes [12].

SCENE CONSTRUCTION

The representation of scene consists of 3D point clouds, which can be computed from the depth maps provided by Kinect. The depth information is stored in a 2D image (Fig 3.), in which each pixel value represents the distance between camera center and the distance plane (perpendicular to the camera optical axis) containing the 3D point.

The maximal range of the Kinect raw depth is divided to 2^{11} units, and it is possible to convert the raw depth to metric depth [7]. Each pixel with valid depth can be interpreted as 3D point on a ray from center of IR camera, passing through corresponding pixel in the distance defined by depth map. The intrinsic parameters of both RGB and IR cameras and extrinsic mapping between them have to be known, so these cameras have to be calibrated beforehand with one of calibration methods [6]. The process of transforming the pixel depth value into 3D point can be expressed by following equations:

$$X_{ir} = \frac{f_{xir}}{(x - c_{xir})d_m}$$
$$Y_{ir} = \frac{f_{yir}}{(y - c_{yir})d_m} \quad (1)$$
$$Z_{ir} = d_m$$

where X_{ir}, Y_{ir}, Z_{ir} are the 3D point coordinates, x, y is position of the depth pixel in image, f_{xir}, f_{yir} is focal length, c_{xir}, c_{yir} is position of principal point of IR camera, and d_m is depth in meters computed from the depth map value at position (x, y). Both focal length and position of the principal point are estimated by calibration. Note that the 3D point position is computed with the respect to the IR camera coordinate frame.

Fig. 3: The image represents the Kinect depth image, color represents the distance between camera center and distance plane. The black pixels have unknown depth value, mostly because of range constraint, occlusion or reflective surface material.

Because the RGB and IR cameras have different intrinsic parameters and camera centers, the coordinates of color pixel doesn't correspond directly to the corresponding depth pixel. Knowing the extrinsic rotation R and translation T between the RGB and IR camera, the mapping between color image and depth image can be computed using equations (2) and (3). X_{rgb}, Y_{rgb}, Z_{rgb} is the position of 3D point in the coordinate system of RGB camera, and x_{xrgb}, y_{rgb} is the 2D position of pixel in RGB image corresponding to the point x, y in the depth image.

$$\begin{pmatrix} X_{rgb} \\ Y_{rgb} \\ Z_{rgb} \end{pmatrix} = \begin{pmatrix} X_{ir} \\ Y_{ir} \\ Z_{ir} \end{pmatrix} R + T \quad (2)$$

$$x_{rgb} = \frac{X_{rgb} f_{xrgb}}{Z_{rgb}} + c_{xrgb}$$
$$y_{rgb} = \frac{Y_{rgb} f_{yrgb}}{Z_{rgb}} + c_{yrgb} \quad (3)$$

ALGORITHM

The scheme of the mapping algorithm is shown in (Fig. 3). In the initialization step, the initial camera position is set to the center of world coordinate frame and rotation is aligned with the negative z-axis. Using the SURF algorithm, the sparse set of feature points is extracted. The Kinect depth map is used to determine the 3D positions of the extracted points.

In each successive frame, point correspondences are found and checked for epipolar constraints (4) and the outliers are excluded from features set. The 2D positions of points in the new frame and the known 3D positions are used to update the camera pose.

After defined number of frames, the information from the depth camera is processed to add new point cloud to the map. The raw depth measurements are used to compute the 3D positions of points using equations (1). These 3D positions can't be added to the map yet, because they need to be transformed to the world coordinate system first. The transformation can be expressed by the following equation:

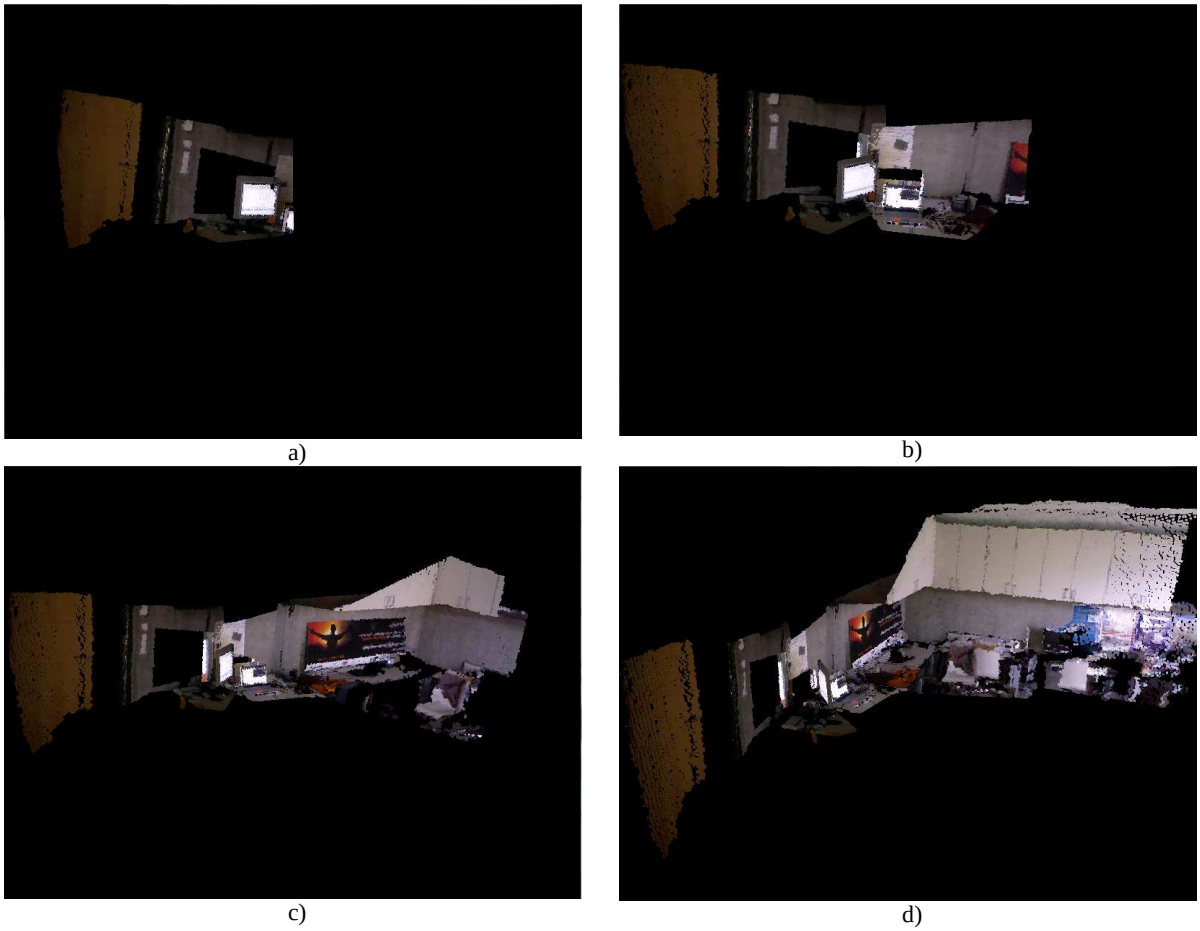
$$\begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} = R^{-1} \begin{pmatrix} X_{rgb} \\ Y_{rgb} \\ Z_{rgb} \end{pmatrix} - R^{-1}T \quad (4)$$

where the matrices R and T are the result of the camera pose estimation, and describe the transformation of 3D point from the coordinate system of the camera at actual position to the world coordinate system. Adding of a new point cloud to map is processed only once in defined number of frames but it is more time consuming than the rest of the algorithm. To ensure real time capabilities of this algorithm, this task is run in separate thread.

The map is checked for the overlapping points, which are merged, and the consecutive frames are processed by the same algorithm.

EXPERIMENTS AND FUTUREWORK

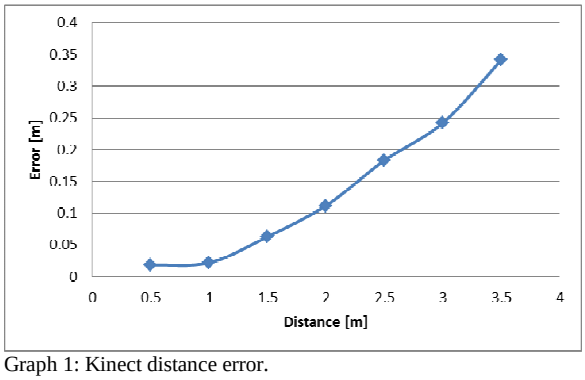
We carried out several experiments to determine the accuracy of this map building algorithm and Kinect itself. The algorithm has been tested in small indoor environments, creating dense 3D maps from pure rotational, pure transitional movement, and the



a) b) c) d)

Fig. 4: a) – d) The reconstruction of small workspace.

combination of both. The metric reconstruction allows to compare the dimensions of the beforehand measured object and its reconstructed image. To measure the error in the 3D position depending on the distance from the Kinect, planar surface was observed from multiple distances, and the variance of the points from the plane was measured. The results are demonstrated in Graph 1.



Graph 1: Kinect distance error.

Performance test were run on a notebook with dual-core processor with frequency of 2.00 GHz and 3GB of RAM. The speed of algorithm depends mostly on the number of features that we are tracking. The proposed algorithm is able to operate in real time (tracking 100 – 150 features) ranging from 18 to 25 frames per second. The durations of the phases of algorithm can be seen in the Table 1.

Comparing this algorithm with state-of-the-art mapping solutions [5], the algorithm suffers from the cumulative error which is caused by small errors in the estimation of the camera pose between the consecutive frames. This problem can be solved by implementing loop closing algorithm [13], which improves the results of maps when the camera returns to the previously visited position. The future work will involve evaluating the camera movement with the respect to ground truth, implementing loop closing algorithm, and also will focus on the optimization of the performance.

	Image query	***Feature tracking and validation***	***3D position update***	***Whole processing***
Average [ms]	8	26	4	38
Percentage	21. 05%	68.42%	10.53%	100%

Table 1: Performance of the algorithm.

ACKNOWLEDGEMENT

This work has been supported by the project of the EU FP7-Artemis project R3COP: Robust Safe Mobile Co-operative Autonomous Systems grant no. 100233.

REFERENCES

[1] Latta, S., Tsunoda, K., Geisner, K., Markovic, R., Bennett, D. A., Perez, K. S.: Gesture Keyboarding. Patent 20100199228, August 5, 2010.

[2] Thrun, S., Burgard, W., Fox, D.: A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In Proc. of the IEEE International Conference on Robotics Automation (ICRA), 2000.

[3] Triebel R., Burgard, W.: Improving simultaneous mapping and localization in 3d using global constraints. In Proc. of the National Conference on Artificial Intelligence (AAAI), 2005

[4] Konolige, K., Agrawal, M.: FrameSLAM: From bundle adjustment to real-time visual mapping. IEEE Transactions on Robotics, 25(5), 2008

[5] Lemaire, T., Berger, C., Jung, I.-K., Lacroix, S.: Vision-Based SLAM: Stereo and Monocular Approaches. International Journal of Computer Vision, 74:343364, 2007

[6] Hartley, R. I., Zisserman, A.: MultipleView Geometry in Computer Vision. Cambridge University Press, second edition, p. 239-259, 2004, ISBN: 0521540518

[7] ROS Kinect Node, http://www.ros.org/wiki/kinect_node, 2011

[8] Bradski, G., Kaehler, A.: Learning OpenCV: Computer Vision with the OpenCV Library. O'Reilly, Cambridge, MA, 2008

[9] Bay, H., Tuytelaars, T., Gool, L. V.: SURF: Speeded up robust features. In 9th European Conference on Computer Vision, Graz Austria, May 2006

[10] Tomasi, C., Kanade T.: Detection and Tracking of Point Features. Carnegie Mellon University Technical Report CMU-CS-91-132, April 1991.

[11] Forsyth, D. A., Ponce, J.: Computer Vision: A Modern Approach. Prentice Hall, us edition, August 2002

[12] Grest, D., Petersen, T., Krüger, V.: A Comparison of Iterative 2D-3D Pose Estimation Methods for Real-Time Applications. Image Analysis, Springer Berlin / Heidelberg, p. 706-715, 2009

[13] Williams, B., Cummins, M., Neira, J., Newman, P., Reid, I., Tardos, J.: An image-to-map loop closing method for monocular SLAM, Proc. International Conference on Intelligent Robots and and Systems, 2008