In Proceedings of 9th International Symposium on Visual Computing (ISVC), Rethymnon, Crete, Greece, 2013.

Efficient Transmission and Rendering of RGB-D Views

Zahid Riaz, Thorsten Linder, Sven Behnke, Rainer Worst, Hartmut Surmann

Fraunhofer IAIS, Schloss Birlinghoven, Sankt Augustin 53754, Germany.
{zahid.riaz, thorsten.linder, sven.behnke, rainer.worst, hartmut.surmann}@iais.fraunhofer.de

Abstract. For the autonomous navigation of the robots in unknown environments, generation of environmental maps and 3D scene reconstruction play a significant role. Simultaneous localization and mapping (SLAM) helps the robots to perceive, plan and navigate autonomously whereas scene reconstruction helps the human supervisors to understand the scene and act accordingly during joint activities with the robots. For successful completion of these joint activities, a detailed understanding of the environment is required for human and robots to interact with each other. Generally, the robots are equipped with multiple sensors and acquire a large amount of data which is challenging to handle. In this paper we propose an efficient 3D scene reconstruction approach for such scenarios using vision and graphics based techniques. This approach can be applied to indoor, outdoor, small and large scale environments. The ultimate goal of this paper is to apply this system to joint rescue operations executed by human and robot teams by reducing a large amount of point cloud data to a smaller amount without compromising on the visual quality of the scene. From thorough experimentation, we show that the proposed system is memory and time efficient and capable to run on the processing unit mounted on the autonomous vehicle. For experimentation purposes, we use standard RGB-D benchmark dataset.

1 Introduction

The availability of affordable sensor systems and state-of-the-art tools and techniques for current robotic systems have made it useful for the researchers to develop autonomous robots which can easily be integrated in real world. The applicability of such robotic systems range from indoor to outdoor robots, surgical robotic arms, assistive robots, industrial robots and rescue robots, where the robots are capable to perform rescue tasks together with humans [1]. For indoor and the outdoor robotics one of the challenging tasks is to generate the 3D maps of the environment. Such maps help the robots to quickly acclimatize to their surroundings and to localize themselves in these maps. This is generally achieved by using *Simultaneous Localization and Mapping (SLAM)* algorithm where a robot localizes itself in a self-generated map of the surrounding [2][3][4][5]. By using these semantic maps the robots can plan, perceive and navigate autonomously in unknown environments.

For a robot-centric rescue operation, a comprehensive situational awareness is a crucial step for the success of robotic aided search and rescue missions. The human operators need to make decisions which are based on the information directly derived from the situation [7][8][9]. The research has shown that the quality and reliability of these decisions depend on the impression of situational awareness rather than the amount of

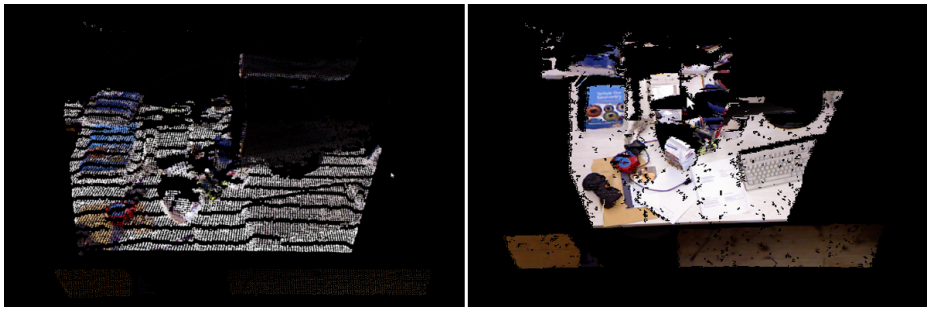

Fig. 1. Example scene generated from a single view from RGB-D benchmark dataset [6]. (Left) Map generated by rendering each point of a downsampled point cloud. The cloud is downsampled from 307200 points to ∼20K. (Right) 3D view from same sequence by rendering the triangular mesh using improved texture quality of an undistorted texture map. The final size of this cloud is ∼680KB as compared to original cloud of size ∼10MB. By utilizing memory and time efficient characteristics of our approach we achieve a data reduction up to 5-12% with better visual quality on the given sequence. For details, refer to section 4.

data provided to the human [10][11][12], i.e. stakeholders do not need all available data but enough data to assess a situation. Moreover in rescue scenarios, some of the areas are not accessible to humans and are suitable for the robots to access and convey useful information to the person remotely connected with them [12][13][14]. In such situations robots can be helpful for the rescue team members if they can access these areas (like tunnels, fire-caught areas) and can efficiently transmit every instance of the sensory data in realtime. In USAR scenarios, a combination of unmanned ground vehicle (UGV) and unmanned aerial vehicle (UAV) acquires a large set of laser scan data and videos of the operational area. Since these robots are connected with each other and to their human team members through a limited transmission bandwidth which may degrade during a field operation. The robot needs to adjust to the communication capacity at each point in time [7][12][15]. In order to make this communication possible between human-robot team member, it is however required to reduce the amount of the data without compromising on the quality of the useful information. For such situations, we propose a memory and time efficient approach to process this data on the onboard processing unit of the vehicle. The goal is to carefully calculate the amount of data which is sufficient to perceive such scenarios and adequate to transmit over the limited bandwidth in real time. We benefit from efficient algorithms which can run in real time on the PC mounted on the robot and reconstruction of the scene can remotely be performed on ordinary CPU instead of using high quality GPU. The final visual quality, memory and time efficiency are not compromised by introducing sparseness in the point cloud data.

The solution is devised by using an intermediate vision and graphics based approach to generate real time 3D model of the environment. Generally, a laser scanner mounted on the robot acquires scans at 30Hz and needs to transmit it to the remote station. For example, a Kinect sensor acquires a point cloud of size more than ∼300K points at 30Hz. This data is reduced to a few thousands of points describing a large area visible to the robot. Since this reduced point cloud contains a small number of points which are

not sufficient to visualize the scene by rendering each point (as can be seen in the left of Figure 1). We render its triangular surface mesh. For a few thousands of points 3D reconstruction can be performed on remote computer by acquiring downsampled point clouds and corresponding RGB images directly transmitted by the robot. A dense point cloud with RGB values provides a detailed representation of the scene but amount of data is large and requires more memory to store and more time to transmit and process this data. On the other hand, with a sparse point cloud representation to a few thousands meaningful points, it is challenging to attain a high visual quality. We achieve a better texture quality by generating an undistorted texture map of the environment and efficiently rendering the triangular mesh. In this way, we achieve a better final visual representation of the scene. The experimentation section 4 shows the results in detail.

2 Related Work

Transmitting the useful data from a robot to its stakeholders is limited by physical constraints. In particular wireless communication channels can be disturbed by several environmental factors like electromagnetic interference, reinforced concrete walls or just by distance [16][12]. Experimentation and field studies showed that such problems can be addressed by different methodology [15][17][18][19], however the most gainful approach seems to be data parsimony. In other words, in USAR domain only those data transmission methods are preferable which transmit needed data as compared to the methods which passes as much data as possible [20]. Birk et al. partially address this issue by presenting an approach which takes three dimensional point data and compresses those surface points to corresponding plane patches, i.e. instead of representing the data as costly point clouds they calculate plane patches of the underling surfaces [21]. This approach takes in consideration the need of weak communication skills by minimizing the data volume, nevertheless it does not utilize color information and the plane patches optimized for human rescue workers perception. A similar approach is utilized by Poppinga et al. which extract convex surface polygons from the point cloud data [22]. This method reduces the data volume up to 50% of the original size. However this approach does not consider special needs of human operators and do not enhance the result by color information. Wiemann et al. reach a compression ratio of up to 65% by using an even more deep polygonal search [23]. But color information is not used in this approach. A slightly different methodology is evaluated by Schnabel et al. [24]. A tree like data structure is utilized to optimize the storage of point in such a way that redundant points are not considered. The authors also make use of color information by storing for each point in the tree the corresponding color data. They achieve a compression ratio up to 30% however this approach does not consider the visual presentation requirements for human operators and bandwidth in USAR application. On contrary, our approach uses an improved data reduction in time efficient manner and utilizes the color information for human perception of the unseen environments.

3 Our Approach

The goal of this paper is to generate a 3D model of a remote scene where 30 frames per seconds are acquired by the robot and required to be transmitted in real time over a limited communication channel capacity. During the design phase, we pay a special attention to the quality of the model, memory efficiency and real time applicability of the system. The input to our system is point cloud data and synchronized RGB images. For benchmarking purpose, we use standard database consisting of RGB and depth images. Point clouds acquired from each depth images are filtered and downsampled. We discard points with no associated depth values. This cloud is further filtered by removing the outliers. Finally the cloud is downsampled by using voxel grid filter. This filtration step removes the noisy points along with outliers and generates a sparse point cloud with a few thousand points in about 20-30 milliseconds approximately. This relatively small set of points can easily be triangulated and efficiently rendered. However, the quality of the texture is important at this stage. We show from our approach that after reduction of points to a very low number, the visualization quality of the cloud is not compromised. Instead of using costly Delaunay triangulation, we use fast surface triangles of surface mesh [25]. Since we triangulated the filtered cloud, it is possible to get triangles with larger sizes which are not smooth with respect to the real surface of the scene and deteriorate final rendered views. To smooth the surface of the final map, we render only those triangles which lie below a certain threshold. This threshold is applied to the perimeter of the triangle. During the fast triangulation calculation if maximum length of the edge is set to d then this threshold is set to $3 \times d$. Figure 2 shows the detail of different modules from our approach.

3.1 Point Matching and Keyframes

In general, multiple sensors are mounted on the autonomous robots to collect a large set of RGB-D data. For example, a Kinect sensor acquires 30 frames of 640×480 RGB and 16-bit depth images per second. In order to reduce this large amount of data, we find keyframes and generate the 3D views from each of these frames. The keyframes can be found by using general approaches [26][27][28]. For each RGB image *Speeded Up Robust Features (SURF)* [29] are calculated after smoothing the image with Gaussian filter with $\sigma = 0.85$. Since the test database contains mostly the translatory motions, so we assume a very small rotation of the camera. However for the purpose of generalization, RANSAC approach can be used for outliers removal [27].

3.2 Filtration and Downsampling

For outdoor field robotics laser sensor are capable to acquire multiple point clouds per second with several hundred thousand points per cloud. The acquired data may contain large sensor and acquisition noise which is not required be transmitted. For RGB images different compression methods and standard codecs are available which facilitate their efficient transmission in real time. However, a challenging task is to transmit point cloud data while not compromising on the quality of the data. In order to efficiently filter the point cloud data we use standard filtering approaches and voxelization to downsample

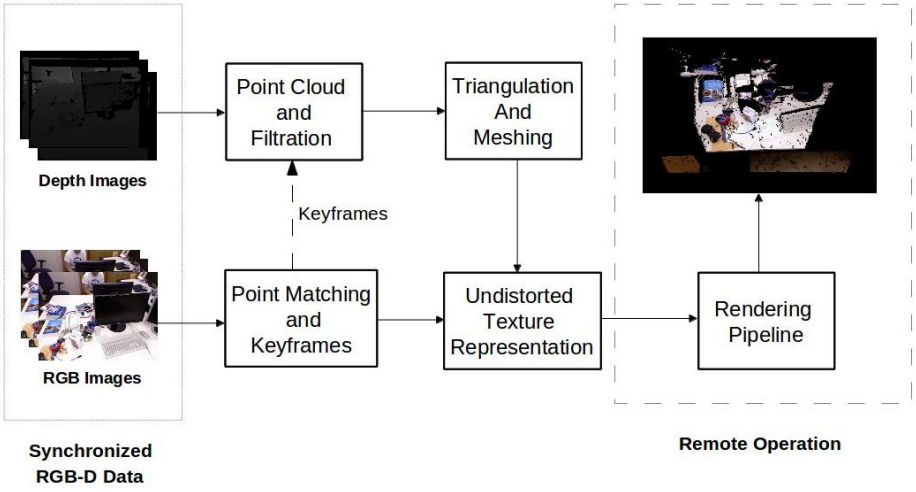

Fig. 2. Overview of our approach. The input to the system is depth image sequence (or point clouds) and synchronized RGB images. The output is 3D reconstruction of the environment which is efficient in computing, storage and capable to apply in real time.

a large point cloud data to reduced set of points which are suitable for triangulation and rendering.

Firstly, We remove those points from the cloud which have no associated depth value. This filtration simply removes all those points where the depth value is either zero or undefined. Secondly, the remaining set of points is filtered by removing the outliers. By setting a filter with $K = 50$ and $T = 1.0$, where K = number of neighboring points to use for mean distance estimation and T = standard deviation multiplier, outliers are removed. We use point cloud library (PCL) implementation for these filters [30]. During these two filtration processes, we use a point cloud of size 307200 which is approximately reduced up to (80%-88%) of the original point cloud size. Note that this result depends upon sensor noise and quality of the cloud however produces significant points reduction. By downsampling this reduced cloud using voxel grid with voxel size of 1.0cm, we finally obtain a downsampled cloud to $\sim$20K points. It is important to note that the computational time for downsampling a cloud is maximum $\sim$30ms and it can be executed in real time on ordinary CPU. This final cloud is useful to generate a sparse mesh by triangulation. We triangulate this mesh using fast surface triangulation method by Marton et. al. [25].

3.3 Texture Mapping and Rendering

In general, the effects of perspective distortions can be ignored during rendering pipeline if the size of the object is smaller then the distance between the object and camera. However, after filtration process we obtain a sparse point could of arbitrary size. This may produce triangles of larger size which leads to texture distortion in final visualization.

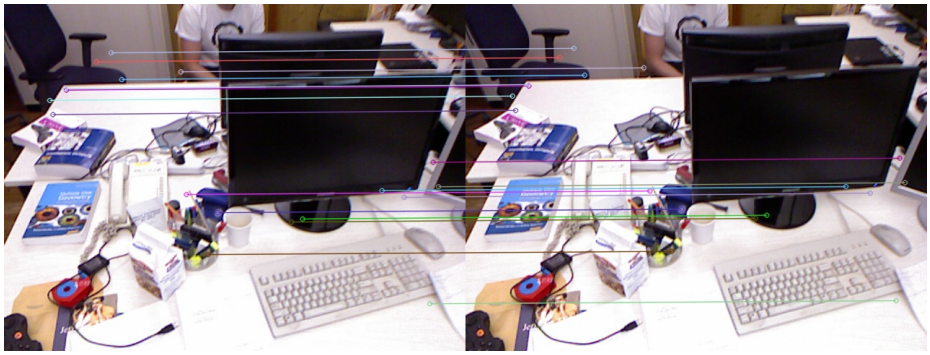

Fig. 3. SURF point matches between consecutive frames. Outliers are removed by applying a distance constraint on the vertical and horizontal shift between key points in consecutive frames. If the number of matched points $N > 4$, we consider these frames and calculate the transformation between two frames. Otherwise this frame is discarded and next frame in the sequence is incrementally considered for point matching.

The affine warping is not directly used because it is not invariant to 3D rigid transformations the triangles. In order to avoid visual distortion, we introduce a texture map which is simply an image with undistorted texture corresponding to each triangle. Later, rendering is performed using this texture map instead of using the corresponding RGB image.

Since 3D position of each triangle vertex as well as the camera parameters are known, we can determine the homogeneous mapping between the image plane and the texture coordinates. This mapping which is usually known as *Homography* H is given by following formula:

$$H = K[R \quad -Rt] \tag{1}$$

where K denotes the camera matrix, R denotes the rotation and t denotes the translation vector. It can easily be shown, that the formula above maps a 2D point of the texture image to the corresponding 2D point of the rendered image of the triangle. The 2D projection p of a general 3D point q in homogeneous coordinates can be written as follows:

$$q = K[R \quad -Rt]p \tag{2}$$

It can be seen that each homogeneous 3D point lying on a plane with $z = 0$, i.e. $p = (x\ y\ 0\ 1)$ leads to above equation 2.

$$q = K\begin{bmatrix} r_1 & r_2 & r_3 & -Rt \end{bmatrix}\begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix} \tag{3}$$

$$q = K.\begin{bmatrix} r_1 \; r_2 \; -Rt \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = Hp \tag{4}$$

with p being the homogeneous 2D point in texture coordinates. Since the camera parameters are known beforehand, the only values to be obtained are the rotation matrix *R* and the translation vector *t*. We use a fixed size shape to store the texture of each triangle. This shape is the upper triangle of a rectangular image for the texture values. In order to fit the texture from any triangle to our fixed shape, we use an additional affine transformation *A*. The final homogeneous transformation *M* is the given by:

$$M = AK\begin{bmatrix} R \; -Rt \end{bmatrix} = AH \tag{5}$$

This transformation is determined in two steps. Firstly, we find the homography H, by obtaining the rotation and transformation of the triangle, by supposing that the initial triangle lies on the texture plane, the first vertex lies on the origin $(0, 0, 1)$ and the first edge lies on the x-axis. Secondly, the affine transformation A is then calculated, so that the mapped triangle on the texture plane fits the upper triangle of the texture map.

4 Experimentation

For experimentation purpose, we test our results on RGB-D benchmark dataset [6]. The first sequence *freiburg1* contains 798 synchronized depth and RGB images of an office environment.

For each RGB image, keypoints are calculated after smoothing each image with Gaussian filter. We set values of $\sigma = 0.85$ [27]. We observe that using image smoothing point matching is closer to accurate. These keypoints and descriptors are calculated using SURF method. However other keypoints like FAST and BREIF may also be used here (for details, refer to section 2). Each SURF descriptor is a vector of size 1×128. These descriptor are matched using Fast Approximate Nearest Neighbor Search (FLANN) [31] implementation of OpenCV library. The correspondences between two frames usually contain outliers. In order to remove these outliers, we restrict the feature point search to a small window size in the vicinity of a given keypoint in next image. The size of this window is chosen by calculating the shift in horizontal and vertical direction of the camera. Since there is slight camera motion in consecutive frames so a point in current frame lies in a small region in the same location in next frame. Any false detection of keypoint either in current or next frame is removed using this window search. The approximate size of this window is set to 20×20 during all experiments. For each keypoint, depth value is extracted from corresponding depth image. We further ignore all those keypoints where the depth value is zero.

The results produced in this paper are tested on the machine with Intel i7-3720QM 2.60GHz processor with 8 GB RAM. A detailed information of the results obtained at each step of our approach is given in Table 1. Using this approach we reduce the point cloud data up to 5-12%, however this loss of data is compensated by efficient rendering of the triangles using improved texture quality of texture map. The texture map in section 3.3 is computed in $\sim$0.24 seconds by removing the perspective distortions in the

	No. of points	**No. of triangles**	**Down-sample time (sec)**	**Size of final map (KB)**	**Rendering time (sec)**
Original Cloud	307200	406533	NA	9395.5	0.27
Our Approach	22474	40168	0.03	8.5	0.03

Table 1. Average values over eight random point clouds from freiburg1 dataset. The figures shows different values of the key steps used in this paper. A comparison to full point cloud to reduced point cloud is given.

given scene and therefore it makes it feasible to render the triangular mesh generated from downsampled point cloud. In order to verify the memory efficiency of the final representation, We also calculate octomap [32]. The final map is stored up to $\sim$35 KB in octomap format. The visual quality of the final point cloud is not compromised which can be seen in Figure 4. The holes in the final visualization of the views arise due to presence of the some of the outliers which are discarded by applying a threshold on the perimeter of the triangles.

5 Conclusions

In this paper, we have proposed an efficient approach to reconstruct an environment while simultaneously reducing the amount of data while preserving the visual quality of the scene. The purpose of this approach is to apply it in robotic aided USAR scenarios where human and robot teams are performing different activities together. Since the amount of data acquired by the robots is large, we focus our attention to reduce the size of the data while not compromising on the quality of data. In this way, we provide a solution for situational awareness by reducing the amount of RGB-D Data required to transmit from robot to user. We evaluated and compared the approach using a standard RGB-D benchmark dataset and achieved a downsampled representation up to (5-12)% of the original cloud while keeping the visual quality still understandable for human users. For future work we plan to make usage of this approach for real world robotic aided search and rescue operations.

Acknowledgment

This research is funded by the EU FP7 ICT program, Cognitive Systems & Robotics unit, under contract 247870, ”NIFTi” (http://www.nifti.eu). and European Consortium for Informatics and Mathematics (ERCIM).

References

1. Kruijff, C.D.G.J.M.: EU FP7 NIFTi ”Natural human-robot cooperation in dynamic environments”. (2010) Funded by the EU FP7 as part of its ICT program, contract #247870.

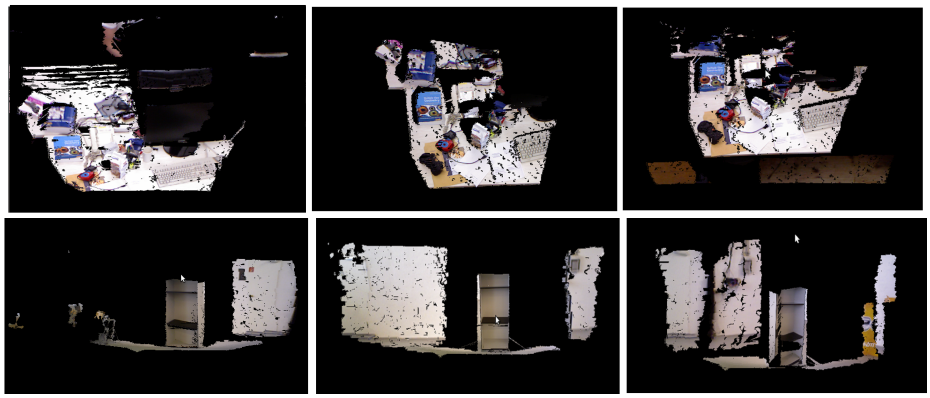

Fig. 4. Sample views generated from the approach used in this paper. (Top row) 3D reconstruction of three different views of the first sequence of the dataset. (Bottom row) 3D reconstruction of three different views of large cupboard of the dataset.

2. Engelhard, N., Endres, F., Hess, J., Sturm, J., Burgard, W.: Real-time 3d visual slam with a hand-held rgb-d camera. In: Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum, Västerås, Sweden (2011)
3. Nüchter, A., Lingemann, K., Hertzberg, J., Surmann, H.: 6d slam - 3d mapping outdoor environments. J. Field Robotics **24** (2007) 699–722
4. Kohlbrecher, S., Meyer, J., von Stryk, O., Klingauf, U.: A flexible and scalable slam system with full 3d motion estimation. In: Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR), IEEE (2011)
5. Stückler, J., Behnke, S.: Multi-resolution surfel maps for efficient dense 3d modeling and tracking. In: Journal of Visual Communication and Image Representation. (2013)
6. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of rgb-d slam systems. In: Proc. of the International Conference on Intelligent Robot Systems (IROS). (2012)
7. Murphy, R., Burke, J.L.: Up from the rubble: Lessons learned about hri from search and rescue. In: Proceedings of the 49th Annual Meetings of the Human Factors and Ergonomics Society. (2005) 437–441
8. Kruijff, G., Colas, F., Svoboda, T., van Diggelen, J., Balmer, P., Pirri, F., Worst, R.: Designing intelligent robots for human-robot teaming in urban search & rescue. In: Proceedings of the AAAI 2012 Spring Symposium on Designing Intelligent Robots. (2012)
9. Larochelle, B., Kruijff, G.J.M.: Multi-view operator control unit to improve situation awareness in usar missions. In: RO-MAN, 2012 IEEE, IEEE (2012) 1103–1108
10. Burke, J., Murphy, R., Coovert, M., Riddle, D.: Moonlight in Miami: An ethnographic study of human-robot interaction in USAR. Human Computer Interaction **19** (2004) 85–116
11. Burke, J., Murphy, R., Rogers, E., Lumelsky, V., Scholtz, J.: Final report for the DARPA/NSF interdisciplinary study on human-robot interaction. IEEE Systems, Man and Cybernetics Part C: Applications and Reviews, special issue on Human-Robot Interaction **34** (2004) 103–112
12. Casper, J., Murphy, R.: Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center. Systems, Man, and Cybernetics, Part B, IEEE Transactions on **33** (2003) 367–385

13. Murphy, R.R., Tadokoro, S., Nardi, D., Jacoff, A., Fiorini, P., Choset, H., Erkmen, A.M.: Search and Rescue Robotics. In: Handbook of Robotics. Springer (2008) 1151 – 1173 ISBN 978-3-540-30301-5.
14. Murphy, R.: Tutorial – introduction to rescue robotics. Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on (2011)
15. Le, V.T., Moraru, V., Bouraqadi, N., Stinckwich, S., Bourdon, F., Nguyen, H.Q.: Issues and challenges in building a robust communication platform for usar robots. (2007)
16. Carlson, J., Murphy, R.: How UGVs physically fail in the field. IEEE Transactions on Robotics **21** (2005) 423–437
17. Sugiyama, H., Tsujioka, T., Murata, M.: Autonomous chain network formation by multi-robot rescue system with ad hoc networking. In: Safety Security and Rescue Robotics (SSRR), 2010 IEEE International Workshop on. (2010) 1–6
18. Ribeiro, C., Ferworn, A., Tran, J.: Wireless mesh network performance for urban search and rescue missions. arXiv (2010)
19. Couceiro, M.S., Rocha, R.P., Ferreira, N.M.: Ensuring ad hoc connectivity in distributed search with robotic darwinian particle swarms. In: Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on, IEEE (2011) 284–289
20. Murphy, R.: Trial by fire [rescue robots]. Robotics & Automation Magazine, IEEE **11** (2004) 50–61
21. Birk, A., Schwertfeger, S., Pathak, K., Vaskevicius, N.: 3d data collection at disaster city at the 2008 nist response robot evaluation exercise (rree). In: Safety, Security & Rescue Robotics (SSRR), 2009 IEEE International Workshop on, IEEE (2009) 1–6
22. Poppinga, J., Vaskevicius, N., Birk, A., Pathak, K.: Fast plane detection and polygonalization in noisy 3d range images. In: Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on, IEEE (2008) 3378–3383
23. Wiemann, T., Nuchter, A., Lingemann, K., Stiene, S., Hertzberg, J.: Automatic construction of polygonal maps from point cloud data. In: Safety Security and Rescue Robotics (SSRR), 2010 IEEE International Workshop on, IEEE (2010) 1–6
24. Schnabel, R., Klein, R.: Octree-based point-cloud compression. In: Symposium on point-based graphics, The Eurographics Association (2006) 111–120
25. Marton, Z.C., Rusu, R.B., Beetz, M.: On Fast Surface Reconstruction Methods for Large and Noisy Datasets. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Kobe, Japan (2009)
26. Henry, P., Krainin, M., Herbst, E., Ren, X., Fox, D.: RGB-d mapping: Using depth cameras for dense 3D modeling of indoor environments (2011)
27. Huang, A.S., Bachrach, A., Henry, P., Krainin, M., Maturana, D., Fox, D., Roy, N.: Visual odometry and mapping for autonomous flight using an RGB-d camera. In: Int. Symposium on Robotics Research (ISRR), Flagstaff, Arizona, USA (2011)
28. Bachrach, A., Prentice, S., He, R., Henry, P., Huang, A.S., Krainin, M., Maturana, D., Fox, D., Roy, N.: Estimation, planning, and mapping for autonomous flight using an rgb-d camera in gps-denied environments. I. J. Robotic Res. **31** (2012) 1320–1343
29. Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-up robust features (surf). Comput. Vis. Image Underst. **110** (2008) 346–359
30. Rusu, R.B., Cousins, S.: 3D is here: Point cloud library (PCL). In: 2011 IEEE International Conference on Robotics and Automation (ICRA), IEEE (2011) 1–4
31. Muja, M., Lowe, D.G.: Fast approximate nearest neighbors with automatic algorithm configuration. In: In VISAPP International Conference on Computer Vision Theory and Applications. (2009) 331–340
32. Hornung, A., Wurm, K.M., Bennewitz, M., Stachniss, C., Burgard, W.: OctoMap: An efficient probabilistic 3D mapping framework based on octrees. Autonomous Robots (2013) Software available at http://octomap.github.com.