A real time visual SLAM for RGB-D cameras based on chamfer distance and occupancy grid

Abdallah Dib, Nicolas Beaufort, François Charpillet

2014 IEEE/ASME International Conference on
Advanced Intelligent Mechatronics (AIM)
Besançon, France, July 8-11, 2014

A Real Time Visual SLAM For RGB-D Cameras Based on Chamfer Distance and Occupancy Grid

Abdallah Dib[1 2 3] Nicolas Beaufort[1] and François Charpillet[1 2 3]

Abstract— **We present a feature based visual SLAM method that uses chamfer distance to estimate the camera motion from RGB-D images. The proposed method does not require any matching which is an expensive operation and always generates false matching that affects the estimated camera motion. Our approach registers the input image iteratively by minimizing the distance between the feature points and the occupancy grid using a distance map. We demonstrate with real experiments the capability of the method to build accurate 3D map of the environment with a hand-held camera.**
While the system was mainly developed to work with RGB-D camera, occupancy grid representation gives the method the ability to work with various types of sensors, we show the capacity of the system to construct accurate 2D maps using telemeter data. We also discuss the similarities between the proposed approach and the traditional ICP algorithm.

I. Introduction

Our objective is to develop a home assistant robot for elderly people at home. Localization and mapping are a key points for us. We are interested in visual SLAM from RGB-D cameras only.
Visual SLAM has been widely investigated in the literature. Recently after the release of the low cost RGB-D cameras (e.g. Kinect, Asus Xtion), indoor visual SLAM has become an active field in the research area of robotics and computer vision.
In this paper, we present a robust visual SLAM method with 6 degrees of freedom using chamfer distance with occupancy grids. Features are first extracted from point cloud and multiple hypotheses are generated from the features by transforming the original set on different axis (for each degree of freedom). Each hypothesis is evaluated with the distance map. The best hypothesis is selected and the process is repeated iteratively, the final transformation matrix of the best hypothesis represents the estimated camera motion.
We show with experiments in real conditions the accuracy of the method and we provide a real time implementation of the system using GPGPU technology.
The main contribution of this paper is the use of chamfer distance extended to 3D with occupancy grid to minimize the distance between an RGB-D image and the occupancy grid that represents the model of the world.
The paper is divided as follow: we start with a background on visual SLAM, occupancy grids and Chamfer distance. Next we present our approach and we discuss the similarities between the proposed method and the ICP algorithm. Experiments are presented in section V followed by a conclusion.

[1]Inria, Villers-lès-Nancy, 54600, France.
[2]Université de Lorraine, LORIA, UMR 7503, Vandoeuvre-lès-Nancy, 54506, France.
[3]CNRS, LORIA, UMR 7503, Vandoeuvre-lès-Nancy, 54506, France.
(first.second)@inria.fr

II. Background

A. Visual SLAM

Visual SLAM can be split into two major parts: sparse (feature based) and dense. Sparse SLAM uses few points in the image for pose estimation, while dense SLAM uses the whole image for registration.
Dense methods use all pixels in the image for registration. The first dense odometry methods were introduced by [1].

A dense method based on RGB-D cameras is described in [2]. These methods use the Lucas-Kanade framework [3] for image alignment by minimizing the photometric error between two consecutive images. A detailed discussion and various optimizations of the Lucas-Kanade framework is provided in [4].

Alternatively, ICP based methods, as introduced by [5], minimize a geometrical error distance. ICP methods require to perform at each iteration of the algorithm an expensive nearest neighbour search. A KD-tree is used by [6] to accelerate the nearest neighbour search. A cache for accelerating KD-tree based ICP is introduced by [7].

Recently, [8] introduced KinectFusion, a system developed by Microsoft for the Kinect SDK, that uses a variant of the ICP algorithm to align the whole image to the scene model. Microsoft introduced a real-time implementation of the ICP algorithm using GPGPU technology.
While Dense SLAM uses the whole image for registration, sparse methods use visual features extraction such as Harris [9], FAST [10], SIFT [11] or SURF [12]. These features are tracked and used to estimate the camera motion. RANSAC [13] is used to remove inconsistent featurs matches [14] [15] [16].

B. Occupancy Grid

Occupancy grids, as introduced by [17], consist of segmentation of the 3D world into voxels. This grid can be seen as a representation of the world, where each voxel within the map holds a probability of its occupancy state. A high probability signifies that the cell is occupied, while a lower probability signifies that the cell is free (not occupied by objects of the environment). The accuracy of the grid depends on its resolution, which is the size of each cell within the grid. Smaller cell size gives better approximation of the environment, but requires more processing time and memory. Using occupancy grids has many advantages, for example, when working with various sensor data it becomes easy to

978-1-4799-5736-1/14/$31.00 ©2014 IEEE

combine different sensor in the same occupancy grid (e.g. multiple cameras, LIDAR). Another advantage is that, when working with RGB-D cameras, a huge amount of points in the point cloud can be reduced using occupancy grids. Occupancy grids can be seen as a tool for compression.

C. Distance Map

Chamfer distance was originally used in 2D image processing, in order to exploit the information contained in the features present in the input image, notably the edges. [18] derived a corrective term from the edge response, using a distance map or chamfer distance.
In image processing, a distance map is created from the edge image, where each pixel in the distance map represents the distance to the closest data pixels in the edge image. This distance map can be used for object matching, Matching is done by translating the object (also known as template) at various locations of the distance image. The matching function is determined by the pixel values of the distance image, which lies under the data pixels of the transformed template. The lower these values are, the better the match between image and template at this location is. The distance map can be determined in two passes thought the image feature array, by a process known as chamfering [19].

In this paper we use a similar process described above for matching a feature set extracted from a point cloud with the occupancy grid. A distance map is created from the occupancy grid, next the set (template) is moved at various locations on the distance map in order to find the transformation (rotation and translation) that minimizes the distance between the feature points and the distance map.

III. Approach

In this section we describe our SLAM method for RGB-D cameras.
Figure 1 shows the pipeline of the system. First the point cloud is constructed from the depth image received at instant *t*. Feature points are extracted from the point cloud and a distance map is created from the occupancy grid. Our method does not need features matching, instead it uses the distance map created from the occupancy grid to estimate the camera motion by minimizing the distance between the feature points and the map.

A. System overview

The point cloud is created from the the depth image by back-projecting each pixel into a 3D point using the following equations:

$$\begin{aligned} X_d &= \frac{(u - c_x)}{f_x} \times Z(p), \\ Y_d &= \frac{(v - c_y)}{f_y} \times Z(p), \\ Z_d &= Z(p), \end{aligned}$$

where (X_d, Y_d, Z_d) are the coordinates of the 3D point in the camera coordinate system for each pixel p, (u, v) are the pixel coordinates in the image, (f_x, f_y, c_x, c_y) are respectively the focal and optical center of the depth camera called *intrinsic parameters*. These parameters can be obtained by a camera calibration process [20], and $Z(p)$ is the depth value for each pixel in the depth image.

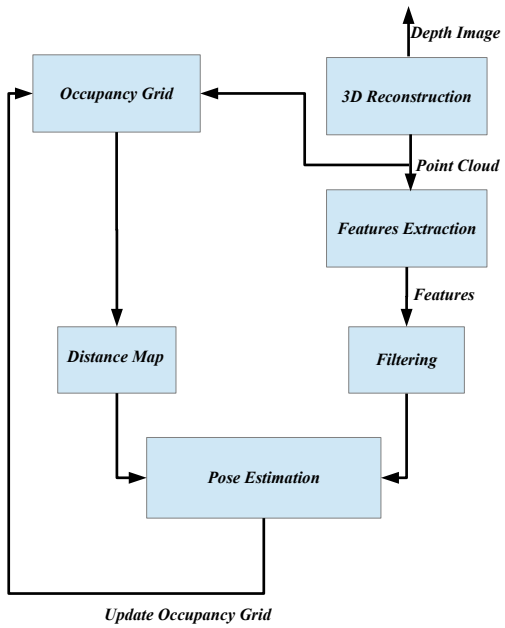

Fig. 1. The pipeline of SLAM system.

Normal vectors are calculated for the point cloud and features are extracted. Section III-B describes the feature extraction method. These feature points are filtered by projecting them to the occupancy to remove points that project to the same voxel, in order to obtain a correct chamfer distance. This part is described in detail in the section III-D.

Next, a distance map is created from the occupancy grid, the feature points are moved (translated and rotated) on the distance map to estimate the camera movement and register the image. This is done by generating multiple hypotheses from the features set. Each hypothesis is obtained by translating and rotating the original set on the different axes X, Y, Z respectively. A score is calculated for each hypothesis by projecting it to the distance map. The hypothesis with lower distance is selected and the process is repeated until the distance reaches a certain threshold. The final transformation matrix of the best hypothesis corresponds to the estimated camera movement. Finally, after registering the new image, the occupancy grid is updated with the point cloud.

Let F_i be the set of 3D voxels positions of the occupancy grid, and let F_j be the set of feature points positions extracted at instant *t*. Our objective is to find the transformation that minimizes the distance between these two sets of points. This can be solved using a traditional ICP pipeline, first by

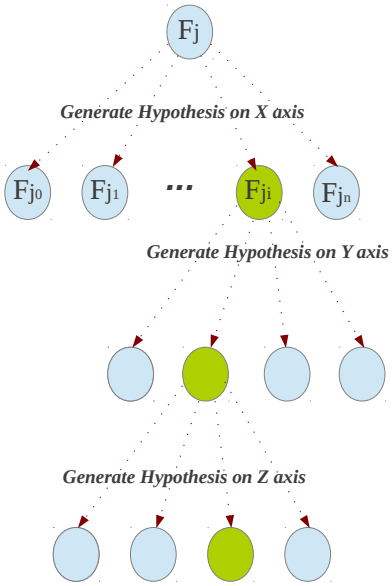

Fig. 2. Hypothesis generation strategy. The algorithm starts on X axis and the hypotheses are obtained by a translation of F_j within a defined interval and evaluated using the distance map. The one that has the closest distance (in green) is selected for the next step. This step is repeated for the 6 DOF (translation and rotation on X, Y and Z axis).

associating points using the nearest neighbour search, then by estimating the rigid transformation between the two data sets using a least square method. In contrast to the ICP algorithm, our method does not require point matching.

To find the transformation (rotation and translation) between the two sets, a distance map DM is created from the occupancy grid. Next, hypotheses $F_{j0}, F_{j1}, ..., F_{jn}$ are generated from the features set F_j for each axis respectively. Each hypothesis is obtained by transforming F_j on each axis within an interval I_k. The algorithm starts on X axis and the generated hypotheses obtained by a translation on X axis are evaluated using the distance map DM. The best hypothesis that has the closest distance to F_i is selected and used for the Y axis and so on. The process is repeated for the 6 DOF (translation and rotation).

For instance, we start on X axis, for an interval of $I_k = [-2cm..2cm]$ and a step of 1 cm, five hypothesis are generated, the first one is obtained by translating the original set by -2 cm, the second by translating the set by -1 cm etc... The five hypotheses are evaluated on the distance map, and the one that has the closest distance to F_i is selected, the selected hypothesis on X axis is used to generate hypotheses on the Y axis and so on.

Figure 2 illustrates the hypothesis generation strategy.

B. Feature Extraction

Our feature extraction method is based on normal vectors. Normal vectors are invariant when observed from different angles. We used a simple normal vector estimation method.

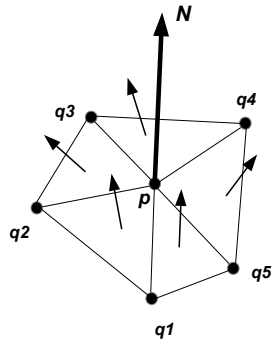

Fig. 3. Normal vectors estimation for point cloud

We refer the reader to [21], where a robust method for normal estimations is proposed by fitting a least square plane to each vertex neighbours. Our implementation of vertex normals estimation is obtained by using an average of the normal vectors of the triangles formed by P and its neighbours as shown in Figure 3. For each Point p in the point cloud, The average of normal vectors composed of the two adjacent vectors is taken within a window. for each adjacent points q_j and q_{j+1} if they are close enough to p we calculate the normal of the triangle composed by the two vectors as follow: $p\vec{q}_j$ and $p\vec{q_{j+1}}$.

$$\vec{N_i} = p\vec{q}_j \wedge p\vec{q_{j+1}},$$

The final normal at point P is the average of the all the normals.

$$\vec{N} = \frac{\sum \vec{N_i}}{n},$$

where n is equal to the number of triangles. Points that are at a distance greater than a certain threshold are excluded from the normal vector computation.

Next, edge features from normal vectors are extracted using a technique similar to the one described in [22] by applying an edge detector filter on the normal vectors. Figure 4 shows the results of the features extraction method.

C. Distance Map Update

This section describes the steps for creating the distance map used to minimize the distance between the feature points and the occupancy grid.

First, Feature points F_j are filtered by projecting them to the occupancy grid and removing points that lie within the same voxel (this is described in details in the section III-D). The distance map is initialized by setting the occupied voxels of the grid to 0 and to infinity otherwise. The algorithm 1 describes the steps for creating the distance which is the most expensive process in the algorithm. In order to accelerate it, unlike typical chamfering methods, in our implementation we only shift the 3 x 3 x 3 window up to a certain distance. We consider that after that distance, there is no need to continue

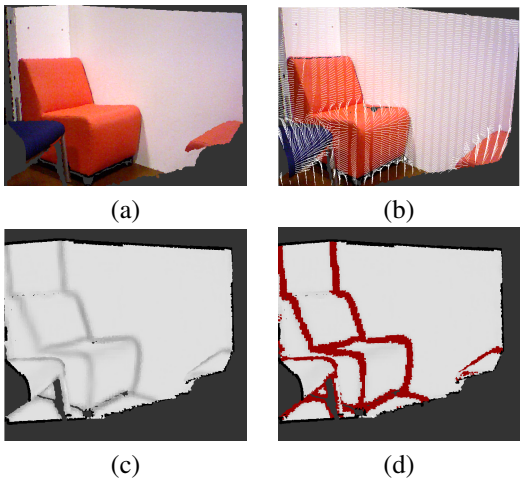

Fig. 4. a) Point cloud. b) Normal vectors. c) Derivation of normal vectors. d) Features points displayed in red.

updating the map. This is illustrated with the number N in the algorithm 1 that tells the number of iterations to perform the map update.

Algorithm 1: Distance Map update

input : F_i, F_j
input : Window(Wx, Wy, Wz)
input : Iterations number N
output: Distance Map DM
/* initialize distance map */
IntializeDistanceMap(DM, Fi)
/* update distance map */
for $iter < N$ **do**
  foreach *cell* c_i *in* DM **do**
    (i_x, i_y, i_z) = IndexInMap(c_i)
    $dist \leftarrow DM(i_x, i_y, i_z)$
    for $i = 1$ *to* W_x **do**
      for $j = 1$ *to* W_y **do**
        for $k = 1$ *to* W_z **do**
          $tmp \leftarrow \sqrt{i + j + k}$
          $dist2 \leftarrow DM(i, j, k)$
          if $(dist + tmp) < dist2$ **then**
            $DM(i, j, k) \leftarrow (dist - tmp)$

D. Score

Each cell in the distance map represents the distance to the closest point in F_i. F_j is transformed by each interval I_k and projected to the distance map.

The distance for each scan F_j to F_i is calculated as follow:

$$distance(F_i, F_j) = \frac{1}{n} \times \sum_{k}^{n} d_k^2, \quad (1)$$

where d_k is the value read from the distance map that corresponds to where the feature point projects (figure 5).

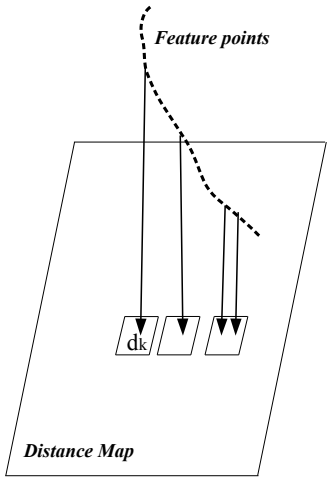

Fig. 5. Projection of a features set to the distance map. Multiple points can project to the same cell position, that is why a filtering step is needed.

As shown in figure 5, depending on the resolution of the map, multiple points can project to the same cell in the map, which affects the score because the same distance is counted multiple times. A filtering step is required, in order to keep only one point projects to the cell and to remove other 3D points from the features set. This is done by projecting the feature points set F_j to the occupancy grid and eliminating points that occupy the same voxel in the gird and keeping only one point.

IV. Similarity with the ICP algorithm

In this section, we discuss the similarity between the method described above and the ICP algorithm.
In ICP, the nearest neighbour search step consists of finding for each point in the first set the nearest neighbour in the second set of points. The next step in ICP is to find the transformation matrix that minimizes the distance between the associated points, this is done by minimizing the following non-linear least square function:

$$f(T) = \frac{1}{n} \times \sum_{k} (A_k - T \times B_k)^2, \quad (2)$$

where A_k and B_k are the associated points and T is the homogeneous representation of the transformation matrix.
In our method, the value stored in each cell of the distance map corresponds to the distance to the closest point in F_i. By comparing equation 1 and 2 it can be seen that the same function is being minimized. In both methods, the sum of squared distance of closest neighbour is being minimized with the only difference that ICP uses a geometrical distance and in the chamfer distance a discrete distance is used.
In contrast to the ICP algorithm, our method does not need to perform a nearest neighbour search because the distance map stores in each cell the distance to the closest point in F_i.

Fig. 6. A 3D model of the apartment with different rooms.

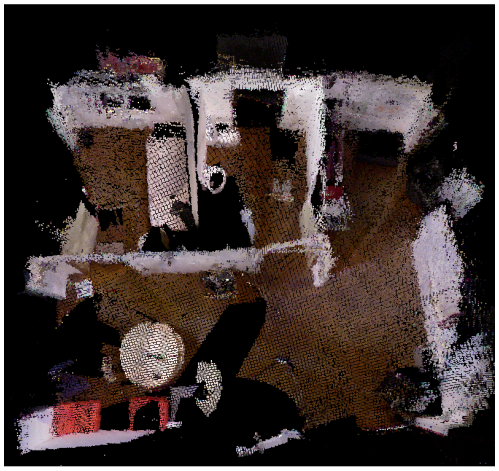

Fig. 7. The 3D map obtained by the SLAM.

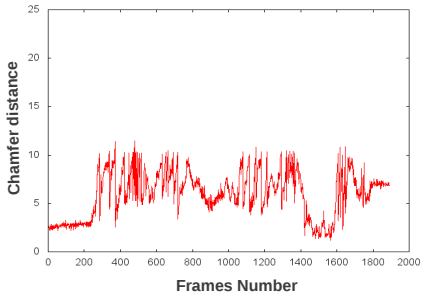

Fig. 8. The distance between the final transformed feature points obtained by the algorithm and the occupancy grid.

V. Experimental results

Experiments were made in real conditions and we used a hand-held Kinect to build the map of the environment. The system runs on an intel i7 processor quad core equipped with a Nvidia Quadro 6000 GPU. The algorithm is implemented on GPU using CUDA. The size of the Occupancy grid is equal to 20 x 20 x 3 meters with a resolution of 5 cm. Figure 6 shows the reference model of the environment. Figure 7 shows the 3D map obtained by the SLAM. The black holes in the map correspond to zones that were not observed by the camera when moving. The furnitures of the 3D model in the image 6 are not placed at the same position as in the real world. Figure 9 shows the superposition of the 3D reference model of the apartment, with the 3D map generated by the SLAM. The image shows that the walls of the reference model (red) match visually the map generated by the SLAM.

A. Robustness and Convergence Speed

Figure 8 shows the evolution in time of the distance between the final transformed feature points obtained by the pose estimation algorithm and the occupancy grid. The figure shows that the chamfer distance is close to zero and bounded by a maximum value. Obtaining a distance close to zero does not mean that the algorithm has found the correct transformation matrix of the camera, the algorithm can converge to a local minimum while keeping a distance close to zero. Our SLAM method like all the existing methods depends on scene and on the features extracted. One advantage of our features extraction method that it is based on scene geometry and does not uses the RGB image and works in dark scenes.

In terms of convergence speed, the algorithm converges in few iterations. In all the tests we did, the algorithm converges in less then 20 iterations. The number of iterations required for convergence depends on the camera movement between two frames, for small camera movement, the algorithm converges in less then 10 iterations.

B. 2D SLAM using telemeter

An important feature of using the occupancy grid, is that it can be adapted to various type of sensors. In this section we tested the SLAM with a Hokuyo telemeter with 30 meters and 270 degrees detection range with 0.25 degrees of angular resolution. The only modification to the method is that no feature extraction is performed. The whole scan is used for matching with the occupancy grid. Figure 10 shows the 2D map generated by the system.

VI. Conclusion

In this paper, a new SLAM method is presented that uses chamfer distance with occupancy grid for RGB-D cameras. The approach does not require feature matching. Instead, the distance between feature points and the occupancy grid is minimized using a distance map. We showed with experiments in real conditions the capacity of the method to construct 3D map of an environment using a hand-held RGB-D camera. We also showed the method works with other type of sensors (the telemeter). We provided a real time implementation using GPGPU technology. We also discussed the similarities between our SLAM and the traditional ICP method.

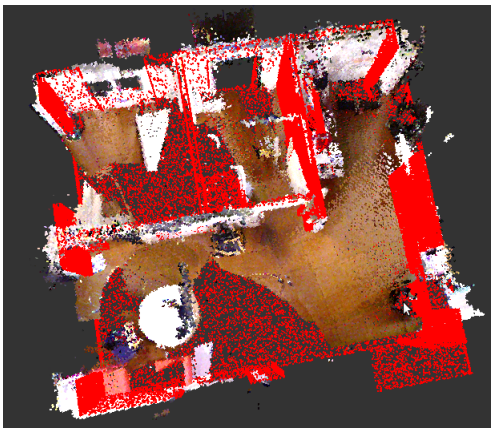

Fig. 9. The superposition of a 3D model (only ground and walls displayed in red) with the map obtained by the SLAM.

Fig. 10. The 2D map obtained by the SLAM using the telemeter..

REFERENCES

[1] A. Comport, E. Malis, and P. Rives, "Real-time quadrifocal visual odometry," *Int. J. Rob. Res.*, vol. 29, no. 2-3, pp. 245–266, Feb. 2010. [Online]. Available: http://dx.doi.org/10.1177/0278364909356601

[2] T. Tykkala, C. Audras, and A. I. Comport, "Direct iterative closest point for real-time visual odometry," in *ICCV Workshops*. IEEE, 2011, pp. 2050–2056.

[3] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)*, April 1981, pp. 674–679.

[4] S. Baker and I. Matthews, "Lucas-kanade 20 years on: A unifying framework," *International Journal of Computer Vision*, vol. 56, no. 3, pp. 221 – 255, March 2004.

[5] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, Feb. 1992. [Online]. Available: http://dx.doi.org/10.1109/34.121791

[6] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Trans. Math. Softw.*, vol. 3, no. 3, pp. 209–226, Sept. 1977. [Online]. Available: http://doi.acm.org/10.1145/355744.355745

[7] A. Nuchter, K. Lingemann, and J. Hertzberg, "Cached k-d tree search for ICP algorithms," *3D Digital Imaging and Modeling, International Conference on*, vol. 0, pp. 419–426, 2007.

[8] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, ser. ISMAR '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 127–136. [Online]. Available: http://dx.doi.org/10.1109/ISMAR.2011.6092378

[9] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. of Fourth Alvey Vision Conference*, 1988, pp. 147–151.

[10] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European Conference on Computer Vision*, 2006, pp. 430–443.

[11] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the International Conference on Computer Vision -Volume 2-*, ser. ICCV '99. Washington, DC, USA: IEEE Computer Society, 1999. [Online]. Available: http://dl.acm.org/citation.cfm?id=850924.851523

[12] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features SURF," *Comput. Vis. Image Underst.*, vol. 110, no. 3, pp. 346–359, June 2008. [Online]. Available: http://dx.doi.org/10.1016/j.cviu.2007.09.014

[13] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, June 1981. [Online]. Available: http://doi.acm.org/10.1145/358669.358692

[14] P. H. S. Torr and A. Zisserman, "MLESAC: A new robust estimator with application to estimating image geometry," *Computer Vision and Image Understanding*, vol. 78, 2000.

[15] K. Konolige, M. Agrawal, R. C. Bolles, C. Cowan, M. Fischler, and B. Gerkey, "Outdoor mapping and navigation using stereo vision," in *Proceedings of the International Symposium on Experimental Robotics*, 2006.

[16] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry for ground vehicle applications," *Journal of Field Robotics*, vol. 23, p. 2006, 2006.

[17] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, June 1989. [Online]. Available: http://dx.doi.org/10.1109/2.30720

[18] D. M. Gavrila and L. S. Davis, "3-d model-based tracking of humans in action: a multi-view approach," 1996, pp. 73–80.

[19] A. Rosenfeld and J. L. Pfaltz, "Distance functions on digital pictures," *Pattern Recognition*, pp. 33–61, 1968.

[20] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[21] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," in *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '92. New York, NY, USA: ACM, 1992, pp. 71–78. [Online]. Available: http://doi.acm.org/10.1145/133994.134011

[22] I. Dryanovski, W. Morris, R. Kaushik, and J. Xiao, "Real-time pose estimation with rgb-d camera," in *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2012 IEEE Conference on*, 2012, pp. 13–20.