Preliminary Environment Mapping for Redirected Walking

Christian Hirt[*] Markus Zank[†] Andreas Kunz[‡]

ETH Zurich
Innovation Center Virtual Reality

Abstract

Redirected walking applications allow a user to explore large virtual environments in a smaller physical space by employing so-called redirection techniques. To further improve the immersion of a virtual experience, path planner algorithms were developed which choose redirection techniques based on the current position and orientation of the user. In order to ensure a reliable performance, planning algorithms depend on accurate position tracking using an external tracking system. However, the disadvantage of such a tracking method is the time-consuming preparation of the physical environment which renders the system immobile. A possible solution to eliminate this dependency is to replace the external tracking system with a state-of-the-art inside-out tracker based on the concept of Simultaneous Localization and Mapping (SLAM).

In this paper, we present an approach in which we attach a commercially available SLAM device to a head-mounted display to track the head motion of a user. From sensor recordings of the device, we construct a map of the surrounding environment for future processing in an existing path planner for redirected walking.

Index Terms: Human-centered computing—Virtual reality; Computing methodologies—Tracking

1 Introduction

Redirected walking (RDW) is used in immersive virtual reality applications and allows a user to explore an extensive virtual environment in a small physical space while walking freely. For RDW systems, it is crucial to have accurate information about the user's position and orientation and they therefore rely on external tracking systems. In order to address the issue of their immobility, we aim to replace the external tracking system with a commercially available inside-out tracker which is based on a simultaneous localization and mapping (SLAM) algorithm. In SLAM tracking, the environment is continuously recorded while points of interest, so-called features, are extracted. These features are stored in a continuously expanding map, but have no geometrical meaning in a spatial sense. Therefore, we create an additional preliminary map, which shall be used in future processing steps to extract geometrical information about the environment.

In this paper, we present a sequence of algorithms which create a map of the surrounding environment by evaluating and fusing raw data captured by a Google Tango Yellowstone tablet.

2 Related Work

RDW was first introduced by Razzaque et al. [4], showing that a virtual room can be larger than the available physical space by employing a redirection technique which manipulates the user's rotation around his own axis (i.e. rotational gain). Besides further improvement on RDW in general, notable milestones were the introduction of path planners evaluating potential future states to apply the most suitable redirection techniques. Zmuda et al. [5] introduced the first path planner predicting the user's state after a redirection technique was used. These states were then ranked based on specific properties and accordingly the best redirection technique was chosen. Nescher et al. [2] further presented a Model Predictive Controller (MPC), which serves as an active path planner solving a finite horizon optimal control problem.

Due to technical limitations, SLAM-based RDW implementations have not been realized so far. However, a potential system setup was shown by Nescher et al. [3] introducing a concept for ad hoc RDW.

3 Methodology

In our setup, we use a Google Tango Yellowstone tablet which allows a 6 degrees of freedom SLAM tracking. It is attached to the front of a head-mounted display to align the user's and the tablet's field of view. Consequently, the head position and orientation of the user can be deduced at any given time using the tablet's tracking data. The SLAM algorithm continuously populates a map with features found in the recorded environment, but this particular map is unsuitable for the existing planning algorithms due to its fundamental difference in structure. Specifically, the existing planning algorithms were originally created to receive a set of 2D coordinates outlining the tracking space. The previously employed external tracking system spanned a rectangular tracking area which was defined by the four respective corners. Because the dedicated tracking area was stationary, the coordinates were manually measured and entered into the system. In that way, the algorithms could plan for the complete tracking space at any given time. However, when using a SLAM tracker, the tracking space is not known beforehand and may even change. Therefore, the map needs to be updated regularly to allow the planning algorithms to account for an expanding or changing tracking environment.

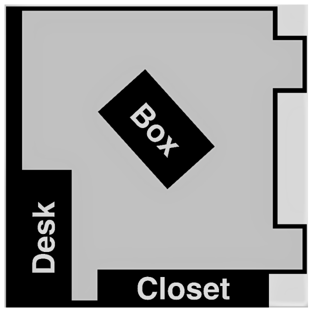
In our approach, we access the data recorded by the tablet's IR depth sensor which scans its surroundings with 5 Hz up to a distance of 3 meters. This recorded point cloud describing the environment is locally preprocessed before it is sent to an external notebook. In this preprocessing step, two different maps are generated continuously. First, an adapted version of a wall map algorithm presented by Hirt et al. [1] is implemented due to its simplicity. The second map is an occupancy map which covers smaller and moving objects while compensating for the slow update rate of the first map. Splitting the preprocessing into two separate maps is necessary to achieve both a fast and accurate representation at the same time.

3.1 Occupancy Map

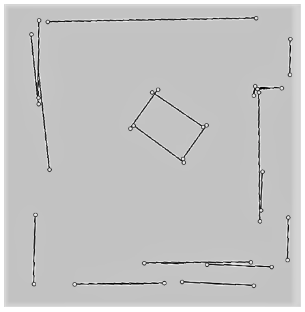
In this part, an occupancy grid is implemented to detect smaller static and moving obstacles. The grid covers an area of 6×6m, which is limited by the depth sensor's range, with the device located in the center. The occupancy grid is oriented parallel to the floor and consists of cells describing the respective real object's shape and position. Each of the grid cells can have one of three states: *FREE*, *OCCUPIED* or *PENDING*. The grid cell resolution is gradually reduced with increasing distance to the device to account for the decreasing accuracy of the sensor and to lower the memory demand. The smallest cell size is defined as $A_0 = 100 \times 100$mm and the

[*]e-mail: hirtc@ethz.ch
[†]e-mail: mzank@ethz.ch
[‡]e-mail: kunz@iwf.mavt.ethz.ch

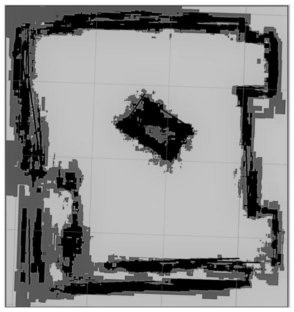
IEEE Virtual Reality 2018
18-22 March 2018, Reutlingen, Germany



(a) Model of the evaluation space with labeled objects



(b) Recorded wall map for the evaluation space using the method by Hirt et al. [1]



(c) Recorded world map for the evaluation space

Figure 1: Testing the world map generation in an evaluation space (6×6m) - black: occupied (saturated), dark grey: occupied, light grey: free

maximum size is set to $A_{max} = 1200 \times 1200$mm. Furthermore, since the occupancy map needs to cover moving objects in the scene, the grid is generated anew with each recorded point cloud and is instantly pushed to the external computer.

For every update of the occupancy map, each cell is initialized in a *PENDING* state and the recorded 3D point cloud is separated into a floor cloud and an obstacle cloud. In order to populate the map, the floor cloud is projected onto the occupancy map first. Each cell holds an integer value n denoting the number of projected points. If a previously determined threshold n_{free} is exceeded, the state of the cell is changed to the state *FREE*. When all floor points are mapped to the grid, n is reset to 0 and the same mapping procedure is repeated for the obstacle cloud, but changing the state to *OCCUPIED* if n exceeds a second threshold n_{occ}. With both projections completed, the occupancy map is pushed to the external notebook.

3.2 World Map

Both maps are transferred to the external notebook where a consistent representation is needed for reliably fusing the information contained in both maps. Due to its compatibility with the previous two maps, a high resolution (10×10mm) occupancy grid is chosen which allows the states *PENDING*, *FREE* and *OCCUPIED*. To take the varying accuracy of the two maps into account, a value Γ_c is introduced which describes the occupancy of each grid cell c. The definition (1) for Γ_c holds for the world map at all time.

$$cell\ state = \begin{cases} PENDING & \text{if } \Gamma_c = 0 \\ FREE & \text{if } \Gamma_c < 0 \\ OCCUPIED & \text{if } \Gamma_c > 0 \end{cases} \tag{1}$$

The initial value of Γ_c is defined as 0 for all cells, which indicates that no information has been recorded in the area so far. During the recording, this value is further addressed iteratively for each cell c as shown in equation (2).

$$\Gamma_c = \Gamma_{c,old} + s_c \cdot w_c \tag{2}$$

with s_c being the state value and w_c denoting a weighting coefficient. The state values s_c are defined as shown in definition (3) and are directly extracted from the transmitted occupancy map.

$$s_c = \begin{cases} 0 & \text{if the cell is } \mathit{PENDING} \\ 1 & \text{if the cell is } \mathit{OCCUPIED} \\ -1 & \text{if the cell is } \mathit{FREE} \end{cases} \tag{3}$$

The weighting coefficient w_c accounts for the depth sensor accuracy drop-off. The size of the smallest cell A_0 is used to calculate w_c for each cell c size as shown in equation (4).

$$w_c = \frac{A_0}{A_c} \tag{4}$$

with A_c being the size of the specific cell c. This allows measurements in high resolution areas of the occupancy map to have a higher impact on the world map.

Because addressing the single cells in the world map is done iteratively, the values for Γ_c need to be limited to ensure that the world map can dynamically adjust to moving objects. To prevent an unbound growth, an upper and a lower saturation value Γ_{max} resp. Γ_{min} are defined symmetrically ($\Gamma_{max} = -\Gamma_{min}$). The upper saturation value Γ_{max} is further directly used to incorporate the high accuracy wall map into the world map by setting Γ_c to Γ_{max} for all corresponding cells.

The algorithms are applied in a larger room (6×6m), adding a single obstacle in the area to simplify the evaluation (see Figure 1(a)). The resulting wall map and world map are shown in Figure 1(b) resp. 1(c). Whereas the geometry of the surroundings is clearly recognizable in the world map, there are some artifacts standing out close to walls and objects.

4 Conclusion

In this paper, we showed a first step to potentially bridge the gap between mobile SLAM and RDW in the future. Depth information about the environment is recorded and used to generate a map in real-time which now needs to be further processed and evaluated in a real RDW scenario. Besides that, different approaches to smooth the maps (e.g. artifacts removal) could be implemented and tested.

References

[1] C. Hirt, M. Zank, and A. Kunz. Real-time wall outline extraction for redirected walking. In *VRST'17 Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology*, p. 72. ACM, 2017.

[2] T. Nescher, Y.-Y. Huang, and A. Kunz. Planning redirection techniques for optimal free walking experience using model predictive control. In *3D User Interfaces (3DUI), 2014 IEEE Symposium on*, pp. 111–118. IEEE, 2014.

[3] T. Nescher, M. Zank, and A. Kunz. Simultaneous mapping and redirected walking for ad hoc free walking in virtual environments. In *IEEE Virtual Reality Conference*, pp. 239–240. IEEE, 2016.

[4] S. Razzaque, Z. Kohn, and M. C. Whitton. Redirected walking. In *Proceedings of EUROGRAPHICS*, vol. 9, pp. 105–106. Manchester, UK, 2001.

[5] M. A. Zmuda, J. L. Wonser, E. R. Bachmann, and E. Hodgson. Optimizing constrained-environment redirected walking instructions using search techniques. *IEEE transactions on visualization and computer graphics*, 19(11):1872–1884, 2013.