2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)
Beijing, China, November 6-9, 2018

rxKinFu: Moving Volume KinectFusion for 3D Perception and Robotics

Dimitrios Kanoulas[1], Nikos G. Tsagarakis[1], and Marsette Vona[2]

***Abstract*— *KinectFusion* is an impressive algorithm that was introduced in 2011 to simultaneously track the movement of a depth camera in the 3D space and densely reconstruct the environment as a Truncated Signed Distance Formula (TSDF) volume, in real-time. In 2012, we introduced the *Moving Volume KinectFusion* method that allows the volume/camera move freely in the space. In this work, we further develop the Moving Volume KinectFusion method (as *rxKinFu*) to fit better to robotic and perception applications, especially for locomotion and manipulation tasks. We describe methods to raycast point clouds from the volume using virtual cameras, and use the point clouds for heightmaps generation (e.g., useful for locomotion) or object dense point cloud extraction (e.g., useful for manipulation). Moreover, we present different methods for keeping the camera fixed with respect to the moving volume, fusing also IMU data and the camera heading/velocity estimation. Last, we integrate and show some demonstrations of rxKinFu on the mini-bipedal robot RPBP, our wheeled quadrupedal robot CENTAURO, and the newly developed full-size humanoid robot COMAN+. We release the code as an open-source package, using the Robotic Operating System (ROS) and the Point Cloud Library (PCL).**

I. INTRODUCTION

Three-dimensional (3D) perception is an important tool for several robotic applications. Apart from understanding the environment itself, e.g. for inspection purposes, most of the autonomous locomotion or manipulation tasks require an accurate knowledge of the surrounding surfaces. Several exteroceptive sensors have been used in the past for acquiring 3D data, such as LiDAR scanners or stereo, depth, RGB-D, monocular, and event cameras. There are applications for which a single frame of data is enough for completing a task. These methods that use successive unaligned information have the advantage of time-varying independence, but always luck the ability to work in areas where the sensor does not provide all the required data at a particular time-frame due to the environment conditions, such as lighting or occlusions. For this reason, the fusion of spatiotemporal camera data are usually required for accurate perception that provides useful information. In limbed robotics, this is crucial both for locomotion and manipulation tasks. For instance, a robot needs to know the ground under its feet when stepping on rough terrain. Unless cameras are mounted under its foot soles, there would always be occlusions between its visual or range sensors and the terrain under its feet, because of its body (legs, arms, etc.). In manipulation, a similar scenario could be true if an object (for instance a box or a mug) needs to be seen from multiple views to be grasped.

Aligning and fusing visual or range data has been studied in the past in the context of sparse Simultaneous Localization and Mapping (SLAM) [1], where features are extracted in the environment and the moving camera is tracked based on them. In 2011, an impressive real-time GPU-based algorithm for dense 3D reconstruction and mapping, named KinectFusion [2], was introduced to function on depth cameras, such as the MS Kinect. The original method was limited in a 3-by-3 meters volume space, which is relative small for robotic applications. For this reason, in 2012 we introduced the Moving Volume KinectFusion algorithm [3], in which we allowed the volume move freely in the space, following the depth camera sensor, through a sequence of volume shifts and remaps. In this way, even though slightly more computationally expensive, the introduced method allowed free-roaming use of KinectFusion in fixed memory space.

In this paper, we further extend the original Moving Volume KinectFusion to the *rxKinFu* algorithm, that applies better to robotic applications, such as locomotion or manipulation. In particular, we first develop three moving volume policies, that may apply to different tasks. Moreover, we introduce a raycasting method to extract point clouds, based on virtual cameras placed in the moving volume. This is particularly interesting since from the same reconstructed environment, one can select the viewpoint that applies to the purpose of the ongoing task. Using the raycasted clouds, the generation of heightmaps or dense object clouds becomes easier. In this direction, we also introduce the use of an IMU sensor mounted on the camera or robot to fuse gravity information into the volume. This helps in the moving volume policies. Furthermore, we integrate color in the reconstructed representation (which was not available in the original system), which can help with various methods that require RGB data to work (e.g. deep learning methods). Last but not least, we integrate rxKinFu into three different robots, i.e., our mini-bipedal robot RPBP [4], [5], our full-size wheeled/legged centaur-like robot CENTAURO (`www.centauro-project.eu`), and our full-size humanoid robot COMAN+ (`www.cogimon.eu`), demonstrating some 3D perceptual methods of our introduced system.

The introduced rxKinFu system was implemented in C++, runs on GPU, and was integrated into the Robotic Operating System (ROS), using the Point Cloud Library (PCL). The code is publicly available as an open-source package: `github.com/RoViL-Team/rxkinfu`.

Next, we review the related work (Sec. I-A), followed

[1]Dimitrios Kanoulas and Nikos G. Tsagarakis are with the Humanoids and Human-Centered Mechatronics Department, Istituto Italiano di Tecnologia (IIT), Via Morego 30, 16163, Genova, Italy. {`Dimitrios.Kanoulas, Nikos.Tsagarakis`}`@iit.it`.

[2]Marsette Vona (NASA Jet Propulsion Laboratory) was with the College of Computer and Information Science, Northeastern University, Boston, Massachusetts, at the time of the associated work. `vona@jpl.nasa.gov`

by a review of the original KinectFusion algorithm [2] and our previous moving volume version [3] (Sec. II). Then, we present our rxKinFu adaptations to the system for 3D perception and robotic applications (Sec. III), and some demonstrations on our three legged robots (Sec. IV). Finally, we conclude with some future research directions.

A. Related Work

Visual SLAM and structure-from-motion methods, were extensively used over the past few years, such as PTAM [6], [7], LSD-SLAM [8], ORB-SLAM [9], DSO [10], or RK-SLAM [11]. Most of these methods use sparse features (except some, such as the dense DTAM [12]) to drive either visual odometry or localization/mapping, using visual cameras with low computational complexity (running usually in CPUs). The disadvantage of sparsity is that the methods rely in the existence of features in the environment. A particular interesting algorithm was the Parallel Tracking and Mapping (PTAM) method [6] that showed impressive results using an RGB camera and was used on a walking robot [13].

Visual SLAM based on depth cameras [14]–[19], has also an increasing interest after the release of cheap depth sensors, such as the MS Kinect or Xtion ASUS. Moreover, the release of cheap GPUs have enabled a whole new area in dense real-time SLAM and reconstruction methods, based on structures such as the Truncated Signed Distance Formula (TSDF) [20] or the Octomap [21]. The new technological advances in the hardware and sensing, led to the introduction of an impressive system named KinectFusion [2], which was based on the TSDF surface representation for real-time mapping and environment reconstruction, for a fixed and relatively small volume (3 cubic meters). Since then, various methods were developed for dense mapping, using depth sensors. In 2012, we extended the original KinectFusion method to a moving volume [3] version that allowed free-roaming camera movements (rotations and translations), while the Kintinuous [22] and the large scale KinectFusion implemented in PCL [23] worked towards translating the volume. Since then, several works are using the concept of KinectFusion for different applications. For instance, BundleFusion [24] was developed towards improving the accuracy of the fusion, DynamicFusion [25] towards the reconstruction and tracking of non-rigid scenes, and ScemanticFusion [26] and DA-RNN [27] towards incorporating object semantics into the reconstruction and tracking.

Our intention is to extend our previously introduced method [3] to more robotic/perception applications. Visual SLAM has been already used in the past on legged robot locomotion and mapping, based on sparse visual features [28]–[30]. There are various works that used KinectFusion as black box for legged robot applications. For instance, the original KinectFusion or subsequent methods, such as Kintinuous, have been used for humanoid robot locomotion [31]–[35]. We intent to provide a free-roaming method, that fuses also IMU data, for legged robot applications, such as locomotion or manipulation.

II. KinectFusion Review

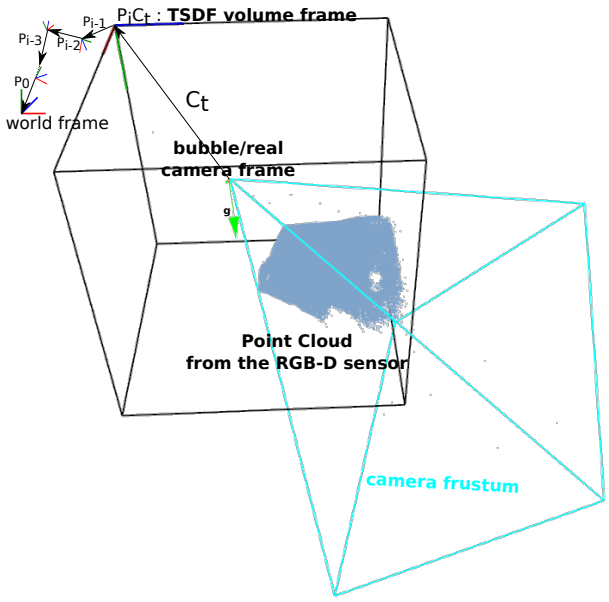

Fig. 1. The voxelized Moving Volume KinectFusion space representation, with the volume transformation frames over time in the initial/world frame.

As mentioned before, the original KinectFusion [2] algorithm works using the Truncated Signed Distance Formula (TSDF) method [20] within a 3D volume structure. The whole volume splits into a fixed amount of voxel grids $\mathbf{v}$, e.g. 512^3 voxels of 3 cubic meters of physical volume space. For each voxel $\mathbf{v}$, two numbers are saved; the signed distance to the closest physical surface d (negative values mean that the cell is behind a surface with respect to the camera) and a weight w that represents the confidence of the distance. For time/space efficiency, only the truncated values are stored, i.e., only cells close to surfaces ($-T < d < T$, for $T = 3$cm) hold values, whereas the rest are either not-initialized ($w = d = 0$) or empty ($T = d$). The input data is a sequence of depth images, from which the camera pose is computed with the Generalized Iterative Closest Point (GICP) [36] method. Through a highly parallelized implementation, every cell is updated (both the distance d and the confidence w) by projecting each new depth image into the volume space. Thus, both camera tracking (camera-to-volume transformation: C_t in frame t) and data fusion (distance/confidence updates) are achieved simultaneously. The original algorithm runs in 30fps, using 512 GPU cores for the MS Kinect range sensor and only depth information. Notice that point clouds could be downloaded from the GPU, using either marching cubes or ray-casting where through zero crossing. The KinectFusion original results were very impressive and worked great for small physical spaces.

A. Moving Volume KinectFusion

In [3], we introduced a tweak in the original method, by allowing free-roaming of the camera, introducing the Moving Volume KinectFusion algorithm (Fig. 1). The main difference with the original method is that the volume moves freely with the camera into the space, through a set of volume

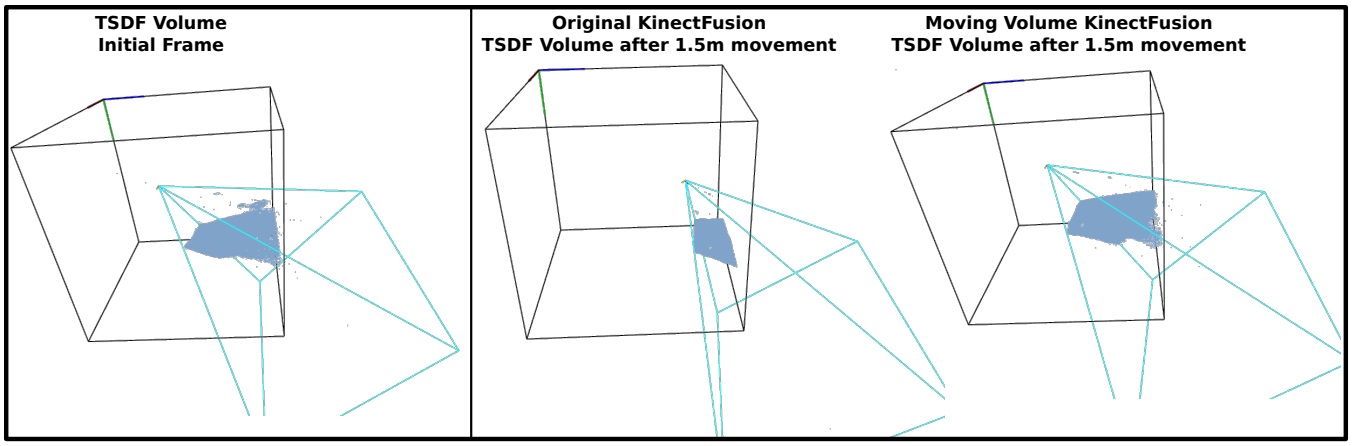

Fig. 2. The original Moving Volume KinectFusion, with the raycasted point cloud from the real camera frustum, compared to the original fixed volume KinectFusion algorithm. The data come from a rocky trail that was crossed with a MS Kinect range sensor, with the instance after 1.5m walking.

shifts/remaps and fast TSDF interpolations, providing also a global camera pose C_g in the world frame. The position of each volume is saved as a rigid transformation P_i (volume i to $i-1$), so one can track the camera movement in time:

$$C_g = P_0 \cdots P_t C_t$$

The interesting part of the moving volume version is that when the camera pose (i.e., the distance l_d and angle α_d difference from the starting pose in the current volume) inside a volume exceeds a distance and an orientation threshold, l_{max} and α_{max}, respectively, the volume is shifted/remapped. If only translation is required, then the volume shifts with an easy per-plane memory shift. When both translation and orientation needs to change for the new volume, then a total volume remap/reslicing takes place. When one cell from the previous volume is remapped exactly to an other cell in the new volume, then the mapping is straightforward, whereas when it is mapped to multiple new cells, then an interpolation is required. We introduced a fast trilinear TSDF interpolation, that keeps the performance of the method in good quality. The system works in 30fps, and only during remapping it drops to 15fps on a GPU. A result of the Moving Volume KinectFusion method, compared to the original one is shown in Fig. 2, where the raycasting takes place from the viewpoint of the original real camera. One can see that when the volume is not shifting/rotating and the physical real camera is reaching the volume's end, the raycasted point clouds disappear. This is not the case with the moving volume version.

III. Moving Volume KinectFusion for 3D Perception and Robotics

The intention for the moving volume KinectFusion development is the ability to function on robotic systems for 3D perception when manipulating or walking in the environment. A free-roaming volume could let a moving robot keep dense information only about the local environment around it, while tracking its pose to a world frame. Though, in [3] we left several open directions to achieve this performance that we try to cover in this work with the newly introduced rxKinFu system.

A. Task-based Remapping Strategies

Originally, the initial camera pose was centered behind the volume. When the distance (l_{max}) and orientation (α_{max}) thresholds were met, the volume shifting/remapping was taking place. In this work, we extend this by introducing three different task-relevant strategies for the camera pose inside the volume. First, we define the TSDF volume's frame to be at its upper-left-back corner (Fig. 1), with the x, y, and z axes towards right, down, and forward directions, correspondingly. We have implemented the following moving volume policies, given a down vector $\mathbf{v_d}$ (e.g. the gravity vector) and a forward vector $\mathbf{v_f}$ (e.g. the direction of the camera movement).

The goal is to determine the rotation matrix R_{vol} and the translation vector $\mathbf{t_{vol}}$ from the new volume frame to the old one, i.e. $P_i = [R_{vol,i} \mid \mathbf{t}_{vol,i}]$ from volume i to $i-1$. The identity transformation denotes no change/movement.

- **Fix Volume (fv):** In this case the volume is not moving at all, as in the original KinectFusion algorithm.
- **Fix Camera in Volume (fcv):** Transform the volume (rotate and translate) as needed, to keep the camera at its initial pose relative to the volume frame. To do that, we need to factor the current camera pose $[R_{Ccurr} \mid \mathbf{t}_{Ccurr}]$ into the initial camera transform $[R_{Cinit} \mid \mathbf{t}_{Cinit}]$ followed by the volume transform:

$$R_{vol} = R_{Ccurr} R_{Cinit}^{-1} \quad (1)$$

$$\mathbf{t_{vol}} = \mathbf{t}_{Ccurr} - R_{vol}\mathbf{t}_{Cinit} \quad (2)$$

- **Fix Down and then Forward in the Volume (fdv):** First, rotate the volume to keep its $+y$-axis direction parallel to a specified down vector, and then orient its $+z$-axis as close as possible to a specified forward vector. The volume is also automatically translated to keep the camera at its initial location. To do that, we let the volume rotation matrix be formed from the following column-wise vectors:

$$R_{vol} = [\mathbf{v}_d \mid \mathbf{v}_f \mid \mathbf{v}_f \times \mathbf{v}_d] \quad (3)$$

For the volume translation vector we need to factor the current camera pose $[R_{Ccurr} \mid \mathbf{t}_{Ccurr}]$ into a new

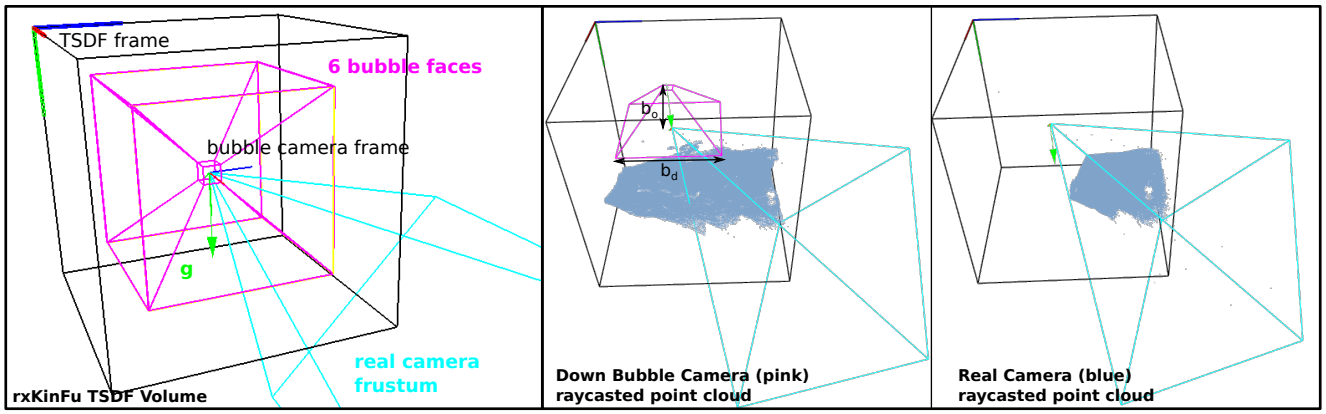

Fig. 3. Left: the proposed rxKinFu TSDF volume, with the 6 virtual bubble camera frustums and the gravity/down vector. Middle: the raycasted point cloud from the down virtual bubble camera after traversing 2m of the trail, using the "Fix Down and then Forward in the Volume (fdv)" strategy and using the gravity as the down vector from an IMU sensor. Right: the raycasted point cloud from the original Moving Volume KinectFusion method.

camera pose $[R'_{Ccurr} \mid \mathbf{t}'_{Ccurr}]$:

$$[R'_{Ccurr} \mid \mathbf{t}'_{Ccurr}] = [R'_{Ccurr} \mid \mathbf{t}_{Cinit}]$$

followed by the volume transformation:

$$[R_{Ccurr} \mid \mathbf{t}_{Ccurr}] = [R_{vol} \mid \mathbf{t_{vol}}] * [R'_{Ccurr} \mid \mathbf{t}_{Cinit}]$$

Now, the two unknowns are the translation part of the volume transformation $\mathbf{t_{vol}}$ and the rotation part of the new camera pose R'_{Ccurr}:

$$R_{Ccurr} = R_{vol}R'_{Ccurr} \Rightarrow R'_{Ccurr} = R_{vol}^{T}R_{Ccurr}$$
$$\mathbf{t}_{Ccurr} = R_{vol}\mathbf{t}_{Cinit} + \mathbf{t_{vol}}$$

The last is resulting to:

$$\mathbf{t_{vol}} = \mathbf{t}_{Ccurr} - R_{vol}\mathbf{t}_{Cinit} \quad (4)$$

- **Fix Forward and then Down in the Volume (ffv):** First, rotate the volume to keep its $+z$-axis direction parallel to a specified forward vector, and then orient its $+y$-axis as close as possible to a specified down vector. The volume is also automatically translated to keep the camera at its initial location. Following the same methodology as *fdv*, we have that:

$$R_{vol} = [\mathbf{v_r} := \mathbf{v}_f \times \mathbf{v}_d \mid \mathbf{v}_d \mid \mathbf{v}_d \times \mathbf{v_r}] \quad (5)$$
$$\mathbf{t_{vol}} = \mathbf{t}_{Ccurr} - R_{vol}\mathbf{t}_{Cinit} \quad (6)$$

These strategies provide several options for the moving volume configuration over time, depending on the robotic task to be completed. For instance, a simple option is to have the moving volume down and forward vectors fixed to the default $+y$ (**mvfd**) and $+z$ (**mvff**) axes of the starting volume configuration.

More interestingly, we have implemented three more options. The first one (**downgrav**) sets the moving volume down vector ($\mathbf{v_d}$) equal to the gravity vector, acquired from an IMU mounted on the camera sensor or robot. This is particularly interesting during locomotion in rough terrain, where heightmaps of the environment need to be generated and the horizontal plane is required (see later Fig. 6 in Sec. IV, where the RPBP robot is using this to acquire point clouds for stepping). The second option (**headvel**) estimates the forward vector ($\mathbf{v_f}$) as the recent camera velocity, while the third option as the camera's $+z$-axis vector (see later Fig. 8 in Sec. IV, where the CENTAURO robot is using this to acquire point clouds for manipulation). One can set a threshold and a weight to control a running average filter for these estimations. These options are particularly interesting during navigation, when the camera-view information is important. An example of the "Fix Down and then Forward in the Volume (fdv)" and "downgrav" options is given in Fig. 3.

IMU Integration: Notice, that the integration of the IMU on the range sensor is of high interest, since one can use it as the down vector ($\mathbf{v_d}$) for applications that require the robot in a standing mode, such as locomotion. There are systems such as the CMU Multisense-SL, that visual/range sensors and IMUs come calibrated from the factory. Although, there are cases where these sensors are separated or their relative pose needs to be re-calibrated. To calculate the transformation between the range sensor and a mounted IMU on it, we collect several depth and gravity data from the IMU. Then, for each pair we calculate the dominant plane to extract the normal vector of the point cloud generated from the depth image. For each plane normal and gravity vector we solve the Procrustes problem [37] to get the transformation between the sensors.

B. Raycasting based on Virtual Bubble Cameras

A very important adaptation that we propose in this paper, is about the method that point cloud raycasting from the TSDF volume takes place. Originally, the default option was to raycast from the real camera view-point. This is a reasonable approach for several applications, but in robotics there are cases that different type of raycasting needs to be applied. For instance, when a robot is locomoting on a terrain, it is preferable to have a raycasting view-point from top-down (potentially in the direction of the gravity vector), so that it can easily see "under" its feet and build a heightmap for planning. As long as the robot has moved in the terrain and the voxels of the TSDF volume have already stored some surface information, accumulating data from the previous frames, it is not possible to produce point clouds

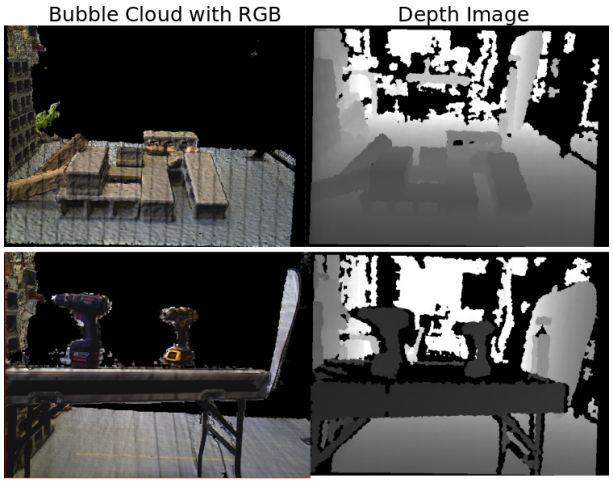

Fig. 4. Left: the raycasted RGB point cloud, from the bubble virtual camera. Right: the depth image of the scene from the real camera.

near the robot feet, raycasting from the real camera view-point if it is not looking towards that direction. On the other side, if the purpose of the robot is to manipulate objects, it may need to change strategy and apply raycasting in a way that a dense set of points on the object is extracted from different view points. For this reason, we introduce a birds-eye view raycasting, based on a set of six virtual bubble cameras. These bubble cameras cover in total the full space around their center of projection (see Fig.3-left).

To define a virtual bubble camera, we let its reference frame be axis-aligned with the frame of the TSDF volume, with the only difference that its z-axis is pointing downwards. We also let the center of projection be at a fixed offset distance b_o above the location of the real camera sensor (see Fig.3-middle). The selection of this offset depends on the details of the environment that are required from the robot. Moreover, we allow a variation in the virtual camera width/height dimensions b_d. The resolution depends on the robotic task to be completed. For instance, to build a heightmap for obstacle collision avoidance, a sparse equally distributed set of points are enough, whereas an object representation may need more dense points. To control further the graphical rectangular prism "bubble" that moves with the real camera, the resolution of the six bubble faces (bottom, front, left, right, back, top) is defined by two values. Each face size (**bs**) and resolution (**br**) defines the frustum of the virtual camera with center of projection at the bubble reference point if and only if the corresponding bubble resolution is positive. In this case, the width and height of the the camera in pixels is given by multiplying the face dimensions in meters times the corresponding face resolution.

In terms of visualization, we have implemented the option to display the bubble prism and the bubble camera frustum. Moreover, one can extract and display the raycast points/meshes for the enabled bubble frustum. Last but not least, we have integrated color inside the TSDF volume, by fusing the color values in each iteration in the GPU, which is an option that was not implemented in the previous version of the system (see Fig. 4).

IV. ROBOTIC APPLICATIONS

We have implemented the rxKinFu system as an open-source ROS package, to allow robotic and perception researchers use it on their robots for various tasks. To demonstrate this capability, we show an integration of the package on three different robotic systems, using RGB-D range sensors (Primesense Carmine 1.09, Kinect v1, and ASUS Xtion PRO): 1) a mini-biped (RPBP), 2) a centaur-like legged/wheeled quadruped (CENTAURO), and 3) a full-size humanoid (COMAN+), visualized in Fig. 5. To show the visual rxKinFu capabilities on the robots, we first run a stepping experiment on the RPBP robot, then three monitoring experiments on the CENTAURO, and a final monitoring experiment on the COMAN+ robot (Fig. 6–Fig. 9).

Fig. 5. The three robots, used in the experiments: 1) the mini-biped RPBP, 2) the quadruped CENTAURO, and 3) the humanoid COMAN+.

On the RPBP mini-biped robot, we tested the capability of rxKinFu to generate raycasted point clouds close to the feet of the robot for the purpose of rock stepping. Notice that we used the method of detecting contact surfaces from point clouds, appropriate for stepping, that we introduced in [38]. In Fig. 6 the sequence of moves for the robot are visualized. Initially, the robot is standing and bends over to look down the ground. When a surface appropriate for stepping is recognized in the point cloud the robot moves statically to place its foot on the contact surface (i.e. a rock). For the rxKinFu options, we let the down vector be the gravity one, coming from the IMU which was mounted on the RGB-D sensor of the robot, while we let the b_o distance between the real camera and the bubble frustum be two times the robot's height. We also used the "Fix Down and then Forward in the Volume (fdv)" option with the down face of the virtual bubble camera acquiring 200px^2. The result of using the virtual camera above the robot instead of the real camera view-point is visualized in Fig. 6, allowing the robot to see around its feet and perform the rock stepping task.

On the CENTAURO robot, we firstly tested the capability to generate raycasted point clouds using different faces of the virtual bubble cameras in different resolutions. In this way, we were able to generate dense point clouds of objects or sparser heightmaps of the ground, from different view-points. Moreover, we tested the capabilities of the raycasting, by tweaking the position of the bubble frustum. Last, we tested the use of the gravity vector (*downgrav*) and the camera velocity (*headvel*) for the *fdv* and *ffv* moving volume strategies. In particular, for experiment 1 (Fig. 8-left) we set a table with two drills and we manually drive the robot towards the objects, moving forward/backward/sideways using its

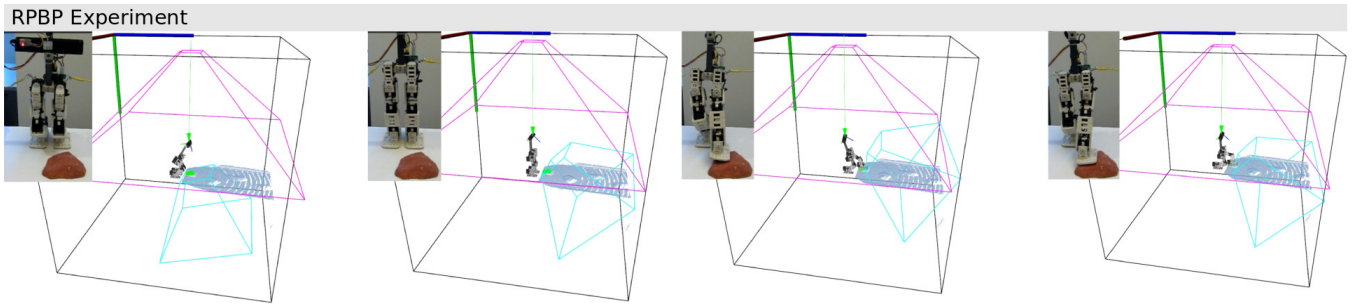

Fig. 6. The sequence of RBPB moves for acquiring and raycating point clouds around its feet using the rxKinFu method. The down face of the virtual bubble camera is activated with 200px^2 resolution and used with the *fdv* moving volume strategy.

wheels. With a simple whole-body Cartesian interface to control the orientation of its Center-of-Mass (CoM), we also tried some small CoM movements. The real camera was horizontal to the ground, facing almost forward. We used the "Fix Forward and then Down in the Volume (ffv)" option with the front face of the virtual bubble camera acquiring a dense 400px^2 cloud. In Fig. 8-left, one could see the nicely raycasted dense point clouds of the drills, using the virtual front bubble camera. Similarly, for experiment 2 (Fig. 8-right) we set some bricks on the ground and we tried similar robot moves as before. Since the camera was heading forward, we had to place the bubble frustum higher and further behind the real camera frame, so that the raycasted point cloud can be extracted, using the down face of the virtual bubble camera. We used also the down (i.e. gravity) vector to align the bubble frustum to the ground in a way that it was straightforward, with a sparse 200px^2 raycasted point cloud to extract directly the corresponding heightmap in real-time. The latter is visualized in Fig. 8-right. Last, for experiment 3 (Fig. 9) we set both the drills on the table and the bricks on the ground and we let CENTAURO roll for a slightly bigger distance, with its real camera facing 45deg downwards. The "Fix Down and then Forward in the Volume (fdv)" strategy is used during volume movements, while the face down virtual bubble camera is raycasting 400px^2 point clouds. The captured instance in Fig. 9 is notable, since the robot has traversed some distance and stands in front of the bricks, but the raycasted cloud appears also under its feet (something that is not visible from the real camera view-point), giving visual capabilities to the robot for locomotion.

Last but not least, for the COMAN+ experiment (Fig. 7) we let the robot make a predefined bending move (similar to those for picking up boxes from the ground) and we tested the ability to raycast dense (400px^2) or sparse (100px^2) point clouds of the box/debris in front of it in real-time, by ignoring also some small dynamic movements of the arms in the scene.

V. CONCLUSIONS AND FUTURE WORKS

In this paper, we presented the development of rxKinFu, which is the a modified Moving Volume KinectFusion algorithm for robotics and 3D perception. We integrated different strategies for remapping the moving volume over time and new ways on raycasting point clouds from the TSDF volume based on virtual cameras. We released the system as an

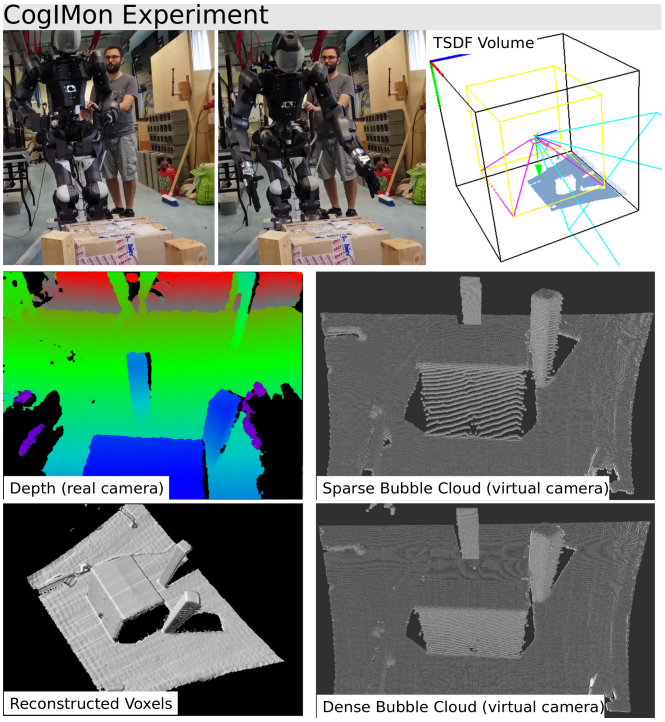

Fig. 7. The experimental results on the COMAN+ humanoid robot, including the TSDF volume, the depth image from the real camera view-point, the reconstructed information from the TSDF voxels, and two raycasted point clouds from the virtual down face of the bubble camera, with different resolutions (100px^2 sparse and 400px^2 dense).

open-source package in ROS, while we applied the method on three different robotic systems: a mini-biped for rock stepping, a legged/wheeled quadruped and a humanoid for scene inspection. We plan to extend this method further and integrate it to other EKF-based IMU/Kinematics state estimators on our robots, while more complicated locomotion/manipulation tasks, such as rock hiking or tool manipulation is our next goal for this system. Last but not least, we plan in extending the method to sensors other than the standard RGB-D ones (Kinect or ASUS) that were used in this work.

ACKNOWLEDGEMENT

This work is supported by the CogIMon (no 644727) EU project. Marsette Vona was at Northeastern University in Boston, MA when part of this work was performed with support from the National Science Foundation (no 1149235). The Titan Xp used for this research was donated by the

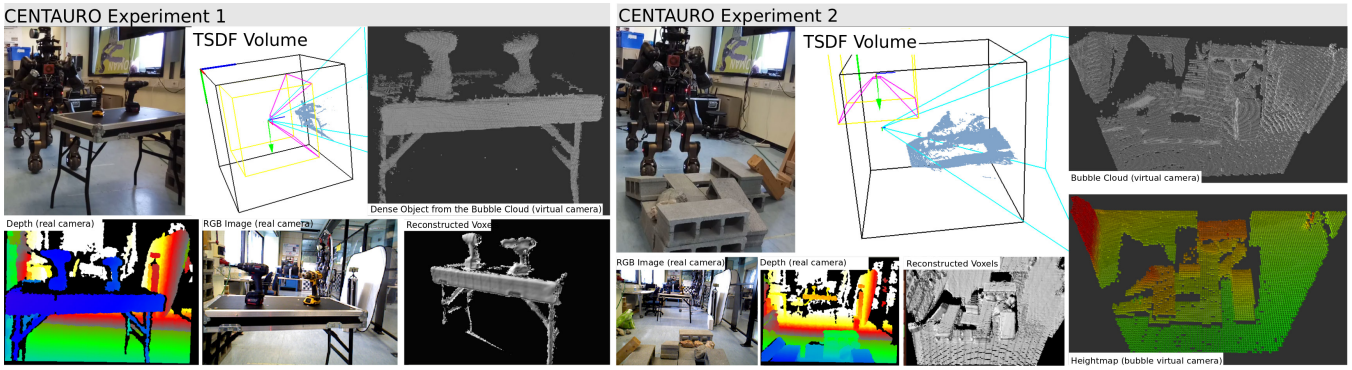

Fig. 8. The experimental results on the CENTAURO robot, including the TSDF volume, the depth/RGB images from the real camera view-point, and the reconstructed information from the TSDF voxels. Left: a dense (400px^2) raycasted point cloud from the virtual front face of the bubble camera. Right: a heightmap generated from the sparse (200px^2) raycasted point cloud from the virtual down face of the bubble camera.

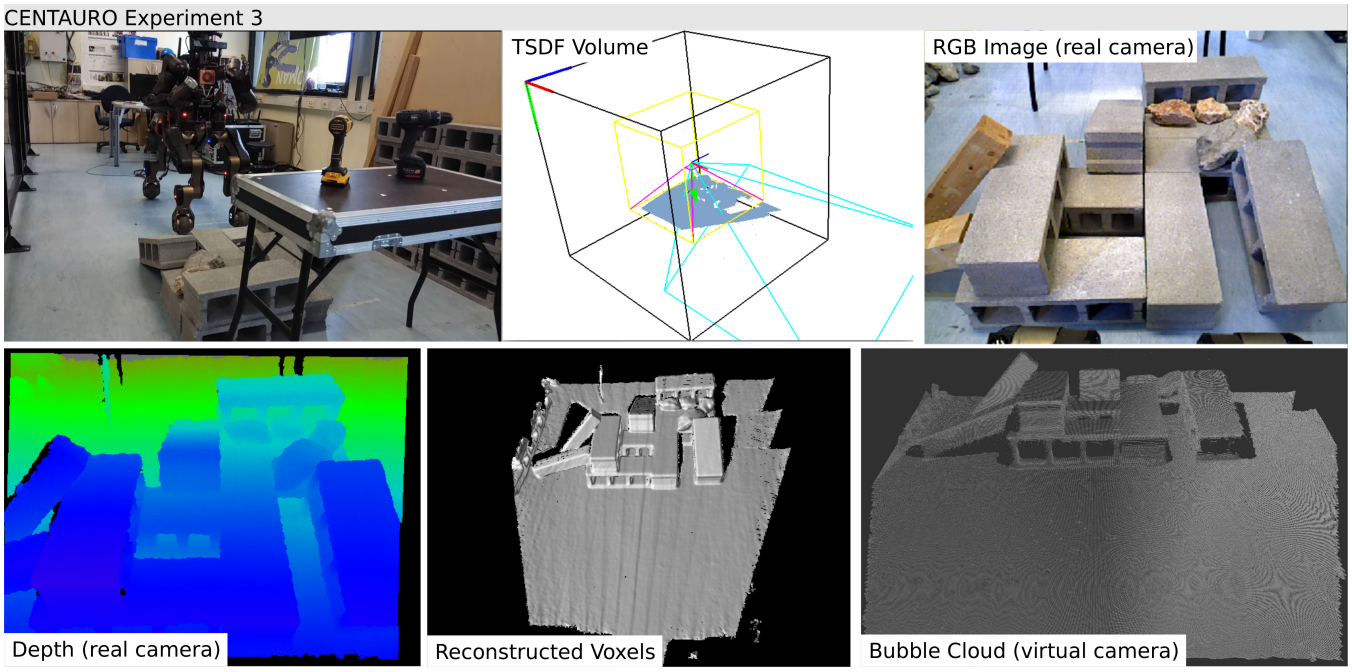

Fig. 9. The experimental results on the CENTAURO robot, including the TSDF volume, the depth/RGB images from the real camera view-point, the reconstructed information from the TSDF voxels, and a dense (400px^2) raycasted point cloud from the virtual down face of the bubble camera. It is notable that even though the robot stands in front of the rock the raycasted cloud is under its feet.

NVIDIA Corporation. The authors would like to also thank Arturo Laurenzi and Enrico Mingo Hoffman for their help with the control system of the CENTAURO and COMAN+.

REFERENCES

[1] M. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba, "A Solution to the Simultaneous Localization and Map Building (SLAM) Problem," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229–241, Jun 2001.

[2] 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 *10th IEEE International Symposium on Mixed and Augmented Reality*, Oct 2011, pp. 127–136.

[3] H. Roth and M. Vona, "Moving Volume KinectFusion," in *British Machine Vision Conference*, 2012.

[4] D. Kanoulas, "Curved Surface Patches for Rough Terrain Perception," Ph.D. dissertation, CCIS, Northeastern University, August 2014.

[5] S. C. Gomez, M. Vona, and D. Kanoulas, "A Three-Toe Biped Foot with Hall-Effect Sensing," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 360–365.

[6] G. Klein and D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," in *6th IEEE/ACM ISMAR*, Nov 2007, pp. 225–234.

[7] ——, "Parallel Tracking and Mapping on a Camera Phone," in *8th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, Oct 2009, pp. 83–86.

[8] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-Scale Direct Monocular SLAM," in *European Conference on Computer Vision (ECCV)*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Springer International Publishing, 2014, pp. 834–849.

[9] R. Mur-Artal, J. M. M. Montiel, and J. D. Tards, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, Oct 2015.

[10] J. Engel, V. Koltun, and D. Cremers, "Direct Sparse Odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 611–625, March 2018.

[11] H. Liu, G. Zhang, and H. Bao, "Robust Keyframe-Based Monocular SLAM for Augmented Reality," in *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2016, pp. 340–341.

[12] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense Tracking and Mapping in Real-Time," in *International Conference on Computer Vision (ICCV)*, 2011, pp. 2320–2327.

[13] D. Belter and P. Skrzypczyski, *Precise Self-Localization of a Walking Robot on Rough Terrain using PTAM*. World Scientific, 2012, pp. 89–96.

[14] J. Stuckler and S. Behnke, "Integrating Depth and Color Cues for Dense Multi-Resolution Scene Mapping Using RGB-D Cameras," in *IEEE International Conference on Multisensor Fusion and Information Integration (MFI)*, 2012, pp. 162–167.

[15] F. Steinbrucker, J. Sturm, and D. Cremers, "Real-time Visual Odometry from Dense RGB-D Images," in *IEEE International Conference on Computer Vision (ICCV) Workshops*, 2011, pp. 719–722.

[16] C. Kerl, J. Sturm, and D. Cremers, "Robust Odometry Estimation for RGB-D Cameras," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2013, pp. 3748–3754.

[17] ——, "Dense Visual SLAM for RGB-D Cameras," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov 2013, pp. 2100–2106.

[18] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-D Mapping With an RGB-D Camera," *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 177–187, Feb 2014.

[19] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D Mapping: Using Depth Cameras for Dense 3D Modeling of Indoor Environments," *Khatib O., Kumar V., Sukhatme G. (eds) Experimental Robotics. Springer Tracts in Advanced Robotics*, vol. 79, 2014.

[20] B. Curless and M. Levoy, "A Volumetric Method for Building Complex Models from Range Images," in *23rd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH. New York, NY, USA: ACM, 1996, pp. 303–312.

[21] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[22] T. Whelan, M. Kaess, M. F. Fallon, H. Johannsson, J. J. Leonard, and J. McDonald, "Kintinuous: Spatially Extended KinectFusion," in *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, Sydney, Australia, July 2012.

[23] F. Heredia and R. Favier, "Kinfu Large Scale in PCL," 2012. [Online]. Available: http://pointclouds.org/documentation/tutorials/using_kinfu_large_scale.php

[24] A. Dai, M. Nießner, M. Zollöfer, S. Izadi, and C. Theobalt, "BundleFusion: Real-time Globally Consistent 3D Reconstruction using On-the-fly Surface Re-integration," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, 2017.

[25] R. A. Newcombe, D. Fox, and S. M. Seitz, "DynamicFusion: Reconstruction and Tracking of Non-rigid Scenes in Real-time," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 343–352.

[26] J. McCormac, A. Handa, A. Davison, and S. Leutenegger, "SemanticFusion: Dense 3D Semantic Mapping with Convolutional Neural Networks," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 4628–4635.

[27] Y. Xiang and D. Fox, "DA-RNN: Semantic Mapping with Data Associated Recurrent Neural Networks," in *Robotics: Science and Systems (RSS)*, 2017.

[28] A. Hornung, S. Bttcher, J. Schlagenhauf, C. Dornhege, A. Hertle, and M. Bennewitz, "Mobile Manipulation in Cluttered Environments with Humanoids: Integrated Perception, Task Planning, and Action Execution," in *IEEE-RAS International Conference on Humanoid Robots*, 2014, pp. 773–778.

[29] M. Fallon, P. Marion, R. Deits, T. Whelan, M. Antone, J. McDonald, and R. Tedrake, "Continuous Humanoid Locomotion over Uneven Terrain using Stereo Fusion," in *15th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2015, pp. 881–888.

[30] T. Laidlow, M. Bloesch, W. Li, and S. Leutenegger, "Dense RGB-D-inertial SLAM with Map Deformations," in *IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS)*, Sept 2017, pp. 6741–6748.

[31] O. E. Ramos, M. Garca, N. Mansard, O. Stasse, J.-B. Hayet, and P. Soures, "Toward Reactive Vision-Guided Walking on Rough Terrain: An Inverse-Dynamics Based Approach," *International Journal of Humanoid Robotics*, vol. 11, no. 02, p. 1441004, 2014.

[32] P. Marion, "Perception Methods for Continuous Humanoid Locomotion Over Uneven Terrain," Ph.D. dissertation, 2016.

[33] A. Tanguy, P. Gergondet, A. I. Comport, and A. Kheddar, "Closed-loop RGB-D SLAM Multi-contact Control for Humanoid Robots," in *IEEE/SICE International Symposium on System Integration (SII)*, Dec 2016, pp. 51–57.

[34] R. Scona, S. Nobili, Y. R. Petillot, and M. Fallon, "Direct Visual SLAM Fusing Proprioception for a Humanoid Robot," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2017, pp. 1419–1426.

[35] A. Rioux and W. Suleiman, "Autonomous SLAM Based Humanoid Navigation in a Cluttered Environment while Transporting a Heavy Load," *Robotics and Autonomous Systems*, vol. 99, pp. 50 – 62, 2018.

[36] A. Segal, D. Hhnel, and S. Thrun, "Generalized-ICP," in *Robotics: Science and Systems*, J. Trinkle, Y. Matsuoka, and J. A. Castellanos, Eds. The MIT Press, 2009.

[37] D. Eggert, A. Lorusso, and R. Fisher, "Estimating 3-D Rigid Body Transformations: a Comparison of Four Major Algorithms," *Machine Vision and Applications*, vol. 9, no. 5, pp. 272–290, 1997.

[38] D. Kanoulas, C. Zhou, A. Nguyen, G. Kanoulas, D. G. Caldwell, and N. G. Tsagarakis, "Vision-based Foothold Contact Reasoning using Curved Surface Patches," in *IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, 2017, pp. 121–128.