Moving Volume KinectFusion

Henry Roth
roth@ccs.neu.edu
Marsette Vona
http://ccis.neu.edu/research/gpc

College of Computer and Information Science
Northeastern University
Boston, MA

Newcombe and Izadi et al's KinectFusion [5] is an impressive new algorithm for real-time dense 3D mapping using the Kinect. It is geared towards games and augmented reality, but could also be of great use for robot perception. However, the algorithm is currently limited to a relatively small volume fixed in the world at start up (typically a $\sim$ 3m cube). This limits applications for perception.

Here we report *moving volume KinectFusion* with additional algorithms that allow the camera to roam freely. We are interested in perception in rough terrain, but the system would also be useful in other applications including free-roaming games and awareness aids for hazardous environments or the visually impaired.

Our approach allows the algorithm to handle a volume that moves arbitrarily on-line (Figure 1).

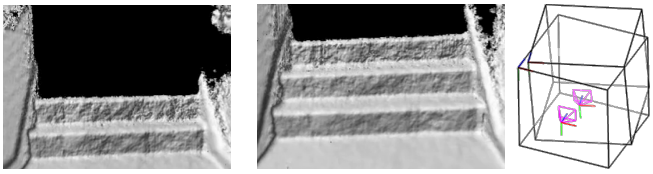

Figure 1: Remapping to hold the sensor pose fixed relative to the volume. Raycast images before and after a remapping show a third step coming into view as the volume moves forward. A reconstruction of the volume and camera poses shows that the volume-to-volume transform is calculated to maintain the camera at the rear center of the volume.

We based our implementation on the open-source *Kinfu* code that has recently been added to the Point Cloud Library (PCL) from Willow Garage [6], and we have submitted our code for inclusion there as well.

With our algorithm in place, the "absolute" camera pose C_g, a 4×4 rigid transform expressing the current camera pose in the very first volume frame, can be calculated at any time t as

$$C_g = P_0 \cdots P_{\mathrm{vf}(t)} C_t \tag{1}$$

where C_t is the current camera tracking transform from KinectFusion taking camera coordinate frame to its *parent volume frame*, each $P_{i>0}$ takes volume frame i to volume frame $i-1$, $P_0 = I_{3\times3}$, and $\mathrm{vf}(t)$ is a bookkeeping function that maps a depth image timestamp to the index of its parent volume. (Volume frames are generally sparser than camera frames.)

Moving volume KinectFusion both tracks *global* camera motion (equation 1) and simultaneously builds a spatial map of the *local* surroundings. However, this is not a true SLAM algorithm as it does not explicitly close large-scale loops and will inevitably incur drift over time. Rather, it can be considered a 6D *visual odometry* approach in that the camera pose ${}^aC_b$ at any time b relative to an earlier time a is

$${}^aC_b = C_a^{-1} P_{\mathrm{vf}(a+1)} \cdots P_{\mathrm{vf}(b)} C_b. \tag{2}$$

Of course the significant additional benefit beyond visual odometry alone is that a map of local environment surfaces is also always available.

After the Kinfu tracking phase gives the current local camera pose C_t we determine if a new volume frame is needed by calculating linear and angular camera offsets l_d, a_d relative to a desired local camera pose C_s.

$$D = \begin{bmatrix} R_d & \mathbf{t}_d \\ 0 & 1 \end{bmatrix} = C_s^{-1} C_t, \quad l_d = \|\mathbf{t}_d\|, \quad a_d = \|\mathrm{rodrigues}^{-1}(R_d)\| \tag{3}$$

A new volume frame is triggered if $l_d > l_{\max}$ or $a_d > a_{\max}$. We typically use $l_{\max} = 0.3$m, $a_{\max} = 0.05$rad, and

$$C_s = \begin{bmatrix} I_{3\times3} & \mathbf{t}_s \\ 0 & 1 \end{bmatrix}, \quad \mathbf{t}_s = \begin{bmatrix} W_m/2 \\ H_m/2 \\ -D_m/10 \end{bmatrix} \tag{4}$$

for volume W_m, H_m, D_m meters, which is the default initial camera pose for Kinfu. This keeps the camera centered just behind the volume (Figure 1, right; note that the origin of each volume frame is the upper left corner of the volume with $\hat{\mathbf{x}}$ right, $\hat{\mathbf{y}}$ down, and $\hat{\mathbf{z}}$ pointing into the page). Other strategies for determining C_s may make sense—for example keeping the camera centered in the volume, orienting the volume to task-relevant directions—but are subject to the constraint that the camera must see scene surfaces within the volume.

To introduce a new volume frame we *remap* the new volume from the old. We maintain a swap buffer in GPU memory the same size as the volume buffer for this; memory requirements for this large data structure are thus doubled but still feasible on current GPUs. After the remap the buffers are swapped and a new relative volume transform P_{n+1} is set as

$$P_{n+1} = C_t C_{t+1}^{-1} \tag{5}$$

where C_{t+1} is the new camera transform. Conceptually $C_{t+1} = C_s$, though we allow an offset in some cases.

Remapping—sometimes called *reslicing* for the 3D case—has been studied for medical images [3], but speed is often sacrificed for accuracy. Efforts have been made to improve the speed [2], but generally reslicing has not been done in real time. Here we require a fast parallel algorithm which is tuned for common-case KinectFusion data.

Our approach is hybridized in two ways. First, if $l_d > l_{\max}$ but $a_d \leq a_{\max}$ we use a fast and exact memory shift algorithm, otherwise we use a more traditional resampling based on trilinear interpolation. Second, during resampling we take advantage of the fact that in the common case much of the volume is either uninitialized or marked "empty": we do a nearest-neighbor lookup first, and only if that is within the truncation band do we continue with a more expensive interpolation.

Using a novel battery-powered Kinect we collected 18 rocky terrain datasets comprising an estimated 662m path length. (Though the Kinect cannot cope with direct sunlight it does work outdoors on a reasonably overcast day.) The richness of 3D depth features makes our approach work well on rocky terrain—no camera tracking failures were incurred, and reconstructed surfaces appear to be high quality (quantitative analysis of the geometry is future work). We present performance and tracking accuracy measurements for our algorithm on 6 datasets, comparing it with the original Kinfu implementation and with ground truth and reference results for RGB-D SLAM [1] where applicable.

While two other groups are also developing approaches to translate the KinectFusion volume [4, 7], a key distinction of our method is the ability to rotate the volume in addition to translation. Since the volume is rectilinear this can be useful to control its orientation, e.g. to maximize overlap of the camera frustum or to align the volume with task-relevant directions, such as the average ground surface normal in locomotion.

[1] N. Engelhard, F. Endres, J. Hess, J. Sturm, and W. Burgard. Real-time 3D visual SLAM with a hand-held RGB-D camera. In *RGB-D Workshop, European Robotics Forum*, 2011.

[2] J. Fischer and A. del Río. A fast method for applying rigid transformations to volume data. In *WSCG*, 2004.

[3] J. Hajnal, N. Saeed, E. Soar, A. Oatridge, I. Young, and G. Bydder. A registration and interpolation procedure for subvoxel matching of serially acquired MR images. *Journal of Computer Assisted Tomography*, 19(2):289–296, 1995.

[4] Francisco Heredia and Raphael Favier. Kinfu Large Scale in PCL. `http://www.pointclouds.org/blog/srcs`, 2012.

[5] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In *UIST*, pages 559–568, 2011.

[6] R. Rusu and S. Cousins. 3D is here: Point cloud library (PCL). In *ICRA*, 2011. (`http://www.pointclouds.org`).

[7] T. Whelan, J. McDonald, M. Kaess, M. Fallon, H. Johannsson, and J. Leonard. Kintinuous: Spatially extended KinectFusion. In *RGB-D Workshop at RSS*, 2012.