A MARKERLESS 3D TRACKING APPROACH FOR AUGMENTED REALITY APPLICATIONS

Michiel Vlaminck, Hiep Luong, Wilfried Philips

Image Processing and Interpretation (IPI)
Ghent University, imec, Belgium

ABSTRACT

Nowadays, augmented reality (AR) applications are still dominated by marker-based solutions to perform pose estimation. The main reason for this is that existing marker-less trackers are still not robust enough to serve the purpose of tracking the head of the user at all times. In this paper, we propose an improvement over existing camera trackers by using only depth information acquired by hand-held depth cameras. The main contribution of this paper is the way we combine a dense tracking approach with a model-based one. More specifically, besides pose estimation based on the alignment of consecutive point clouds, a second thread is running for the alignment of the acquired point cloud with a 3D model. To make the tracking both robust, accurate and fast, we propose a surface reconstruction step combined with a multi-resolution resampling of the 3D model. Our system was extensively tested on an online available dataset and the accuracy proved to be competitive with state-of-the-art visual SLAM methods.

Index Terms— AR, point cloud, registration, Kinect

1. INTRODUCTION AND RELATED WORK

More and more augmented reality (AR) applications are using head-mounted devices to give users a fully *immersive* experience. Among the examples are the HoloLens introduced by Microsoft in 2015 or the recently available Meta 2[1]. These devices are tracking the head of the user mainly using accelerometers and gyroscopes. However, often times it is desirable to track the pose of the head related to an existing 3D model of the environment, in order to be able to fully interact with it. Today, many applications are still using artificial markers that are put onto the scene instead of using *natural landmark* tracking or full-blown SLAM solutions - SLAM being the abbreviation for Simultaneous Localization And Mapping. The main reason is that the current SLAM methods often lack robustness, as they are most of the time only visual-based, such as ORB-SLAM [6], LSD-SLAM [2] and the older PTAM [5]. They are hence subject to subtle changes in lighting conditions or shadows and also heavily rely on the presence of textured objects in the scene. Also, they are less suited to provide the pose related to an existing 3D model in the database, which is often desirable for AR applications. Therefore, it is a lot more interesting to use RGB-D cameras to perform tracking as we can then also incorporate depth information. With the advent of cheap RGB-D sensors such as the Kinect, many researchers have been focusing on this [3, 8, 1, 7, 4]. Early RGB-D sensors were using structured light and were thus only suited for indoor environments. However, since the introduction of the Kinect 2, a consumer-level device is available that is based on Time-of-Flight (ToF) and that can even be used for limited outdoor environments.

The main drawback of the aforementioned solutions is that they are all incremental or sequential and as a result they are prone to drift (known as error propagation) leading to potentially large errors after some period of time. Moreover, in AR we often want to know the precise location and orientation of the person related to an existing 3D model that is present in a database. This is especially the case for industrial applications, where tasks have to be performed based on specific components of machinery. In this paper, we therefore propose a hybrid system in the sense that for the fight against drift we adopt a model-based approach while the core of the tracking is done based on the direct registration of two consecutive point clouds. This way we keep the best of two worlds by being both fast and robust and being able to cope with drift. Regarding the model-based tracking, we introduce a surface reconstruction step as well as a re-sampling step to improve the robustness and accuracy. In addition, we apply a multi-resolution registration scheme to speed-up the pose estimation.

2. APPLICATION

The application for which this research was conducted is the one of *engineering coach* in which a layman is guided towards the completion of a particular task without having the knowledge to do so. The precise instructions are provided to the layman by means of visual cues. The task the layman has to deal with in our application is replacing an oil filter of a cooling machine. In Figure 1, an image is depicted of the working environment.

e-mail: michiel.vlaminck@ugent.be

[1] http://www.metavision.com/

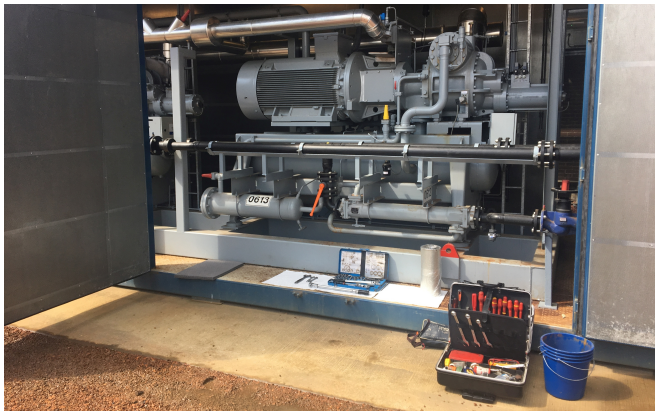

Fig. 1: Image of the cooling container.

Fig. 2: Image depicting the sight of the layman (left) and the (colored) point cloud from the Kinect associated with that (right).

The main prerequisite to solve this use case is of course to find out the current position of the layman and the direction he is looking at. To this end, we used a Kinect 2 sensor that was hand-held by the layman but mimicked his sight. We used the depth data to track his position and orientation. In Figure 2, an image is depicted of the layman and the (colored) point cloud from the Kinect. The final goal would of course be to deploy the system on a head-mounted device that has a ToF camera integrated.

As the tracking of the Kinect only gives us a relative measure on the pose of the user, we incorporate the use of a ground truth 3D point cloud of the machine. Moreover, this point cloud is linked with an existing CAD model that is semantically annotated and contains information on all of the components. Thus, once we figured out the current pose of the layman, we also know the exact position of certain important components in his field of view.

The ground truth point cloud model was created by means of a static laser scan, more specifically using a Leica ScanStation[2]. In Figure 3, an image of the 3D model is depicted. The colour in the image denotes the *height*.

[2]http://leica-geosystems.com/

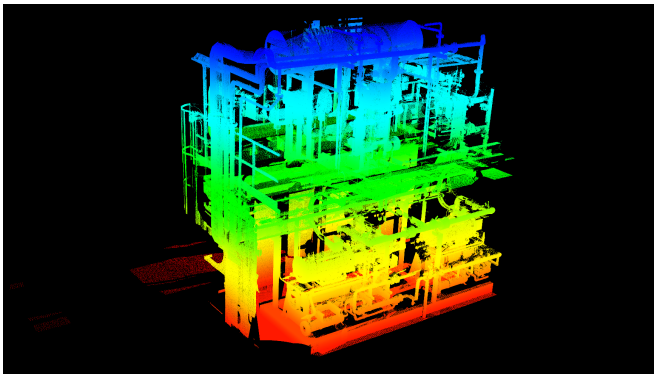

Fig. 3: Image of the ground truth 3D model, obtained by terrestrial lidar scanning using a Leica system. The color denotes the *height*.

3. APPROACH

Our approach consists of four main steps. First, we convert the point cloud to a *surfel* representation by computing several local features, including the normal vector for each point in the point cloud. The second step deals with the alignment of consecutive point clouds, which gives us an initial guess for the current pose. Third, we conduct a surface reconstruction on the 3D model built so far by means of moving least squares. This step will allow us to access the 3D model using different resolutions. Finally, the fourth step deals with the actual alignment of the point cloud with the 3D model to obtain a final accurate estimate on the pose.

3.1. Surfel representation

The word *surfel* is a concatenation of 'surface' and 'element' and hence it can be seen as a building block (element) to represent a surface patch. In essence, it consists of the point $\mathbf{p}_k$ itself, together with its normal vector $\mathbf{n}_k$, its normal confidence c_k and the set of neighbours $\mathbf{N}_k$ that was used to compute the surface normal. We denote the surfel of a point $\mathbf{p}_k$ by $\mathbf{S}_k = \{\mathbf{p}_k, \mathbf{n}_k, \mathbf{N}_k, c_k\}$. Often times, an easy neighbourhood function is chosen, such as the k nearest neighbours or all the points lying within a certain radius. However, this overly simple function causes the estimation of normal vectors to be inaccurate as points can be separated by an object border and can be lying on different surfaces. This can on its turn jeopardize the correct alignment of two point clouds and hence the estimation of the camera pose will also be inaccurate. In the literature, a popular way of estimating normal vectors is by using the principal component analysis of the neighbours of a point. We extended this method by incorporating the principal component analysis in the determination of the *optimal* set of neighbours. Thus, given a set of neighbours of a point $\mathbf{p}_k$ determined by a radius r, we first compute the eigenvectors λ_1, λ_2 and λ_3 corresponding to the eigenvectors $\mathbf{v}_1$, $\mathbf{v}_2$ and

$\mathbf{v}_3$. We then define the standard deviation along an eigenvector as $\sigma_i = \sqrt{\lambda_i}$ for $i \in 1, 2, 3$. Now, we can consider the three values $\psi_1 = \frac{\sigma_1 - \sigma_2}{\sigma_1}$, $\psi_2 = \frac{\sigma_2 - \sigma_3}{\sigma_1}$, $\psi_3 = \frac{\sigma_3}{\sigma_1}$ that represent respectively how linear, planar or scattered the underlying surface is. We then define the *dimensionality* label as $l = \underset{i \in [1,3]}{\operatorname{argmax}}(\psi_i)$. Thus, points lying on the intersection of two planes are assigned the label '0', points lying on a planar surfaces are assigned the label '1' and finally points belonging to volumes or scatter are assigned the label '3'. As the three values ψ_1, ψ_2 and ψ_3 represent a partition of unity Ψ, we can compute its Shannon entropy, which is given by the following formula:

$$E(\Psi) = -\psi_1 \ln(\psi_1) - \psi_2 \ln(\psi_2) - \psi_3 \ln(\psi_3). \quad (1)$$

This Shannon entropy gives a notion of how certain we are that a point is really belonging to one of the three classes. The *ideal* set of neighbors $\mathbf{N}_k$ is then given by the optimal radius r^* given by equation 2:

$$r^* = \underset{r \in [r_{min}, r_{max}]}{\operatorname{argmin}} E(\Psi^r). \quad (2)$$

The eigenvalues, eigenvectors and resulting ψ-values, in combination with the Entropy and dimensionality label are forming a feature vector that will be used in the local alignment step to give a weight to corresponding pairs.

3.2. Moving least squares

To de-noise the point cloud and to smooth surfaces, e.g. along gaps, we apply the *moving least squares* algorithm. This method estimates polynomials through the points and subsequently re-sample the point cloud based on them. For each point $\mathbf{p}_k$ we have its neighbours in $\mathbf{N}_k$ as well as its tangent plane defined as $H_k \triangleq [\mathbf{n}_k, d_k]$ in which d_k represents the distance to the origin. For all points lying in $\mathbf{N}_k$, we can compute the distance to this tangent plane. Subsequently, we fit a polynomial in the set of distances from these points to the surface. To this end, we define a local approximation of degree m by a polynomial $\tilde{p}_k \in \Pi_m$ minimizing, among all $p \in \Pi_m$, the weighted least-squares error:

$$\tilde{p}_k = \underset{p}{\operatorname{argmin}} \sum_{\mathbf{p}_i \in \mathbf{N}_k} (p(\mathbf{x}_i) - f_i)^2 \theta(||\mathbf{p}_i - \mathbf{p}_k||), \quad (3)$$

where $\boldsymbol{I}$ is the vector containing the indices of the points in $\mathbf{N}_k$, $\{\mathbf{x}_i\}_{i \in \boldsymbol{I}}$ are the orthogonal projections of the points $\{\mathbf{p}_i\}_{i \in \boldsymbol{I}}$ onto the tangent plane and $f_i \triangleq \langle \mathbf{p}_i, \mathbf{n} \rangle - d$ is the distance of $\mathbf{p}_i$ to the tangent plane. Finally, $\theta(x) = e^{-(\frac{x}{\sigma_r})^2}$ represents the weighting function that is based on the distances to the tangent plane and the average separation σ_r of the 3D points.

Once the parameters of the polynomials $\tilde{p}_k$ are known, we project the points on the *moving least squares* (MLS) surface. This procedure manipulates the points in such a way that they represent the underlying surface in a better way. In addition, we up-sample the point cloud using *voxel grid dilation* in order to fill small *gaps*. This latter process first dilates a voxel grid representation of the point cloud, built using a predefined voxel size. After that, the newly created points are projected to the MLS surface of the closest point in the point cloud.

3.3. Local ICP

Once the Kinect point cloud has been cleaned up and smoothed, the next step consists in finding the transformation matrix that aligns it with the previously acquired Kinect point cloud. To this end, we adopt the well-known ICP algorithm, an iterative method minimizing an error metric based on closest point correspondences between two point clouds. The error metric we solve in each iteration is the *point-to-plane* metric given in eq. 4.

$$E(\boldsymbol{S}, \boldsymbol{T}; \mathrm{T}) = \sum_{i=1}^{N} \mathbf{w}_i ((\mathrm{T}\mathbf{p}_i^s - \mathbf{p}_{\mathbf{c}(i)}^t) \cdot \mathbf{n}_{\mathbf{c}(i)}^t)^2. \quad (4)$$

In this equation, $\boldsymbol{S}$ and $\boldsymbol{T}$ are respectively the *source* point cloud and the *target* point cloud, in our case respectively the current and previous point cloud acquired by the Kinect. Further, N is the number of corresponding point pairs, $\mathbf{c}$ is the vector containing the indices of the corresponding pairs and $\mathbf{n}^t$ represents the normal vector of the target point $\mathbf{p}^t$.

Usually, the closest point correspondences are determined using the Euclidean norm in 3D space. However, as this approach is often leading to wrong point correspondences when the point density in the point cloud is inhomogeneous, we propose to introduce a weight for every corresponding pair. This weight is determined using the Mahalanobis distance in feature space (cfr. section 3.1) between the two correspondences. The actual weight is computed using the Beaton-Tukey robust M-estimator and is updated in every iteration based on the median weight value in order to filter out outlier correspondences. Thus, the process is denoted as an iteratively re-weighted least squares (IRLS) ICP solution.

3.4. Global ICP

The local ICP provides an initial estimate on the camera pose. However, as this estimation is subject to drift, we perform an additional ICP step using the existing ground truth 3D model. However, as the 3D model contains too many points, it would be too intractable to determine corresponding, *closest* point pairs between the Kinect point cloud and the 3D model. To this end, we convert the 3D ground truth model to an octree-based data structure. This octree comprises a hierarchical space partitioning by subdividing the 3D space recursively

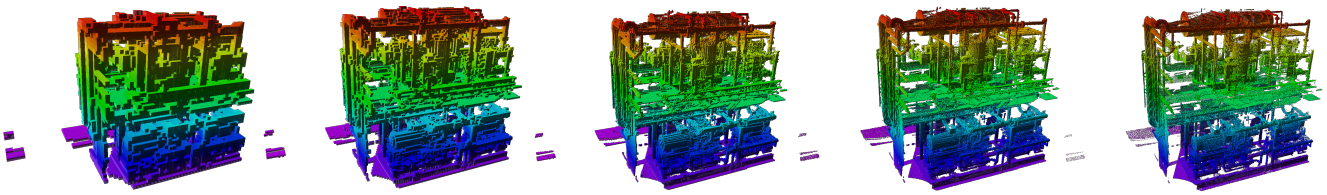

Fig. 4: Several resolution levels of the cooling machine. From left to right, the voxel size is respectively 0.32, 0.16, 0.08, 0.04 and 0.02 meter corresponding with resolution level 11 to 15.

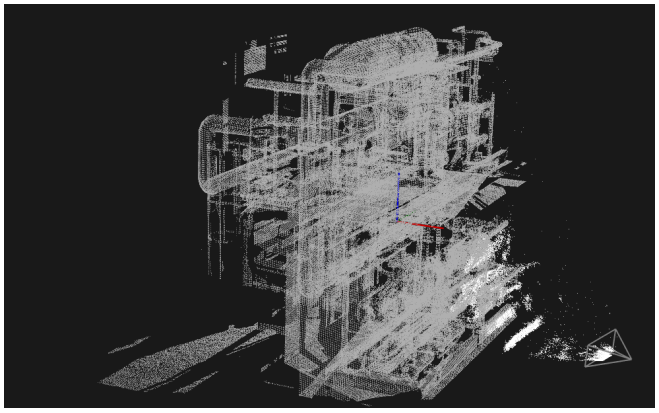

Fig. 5: The point cloud originated from the Kinect (white) overlayed on the *ground truth* point cloud after its pose has been estimated.

into 8 smaller sub cubes. At the leaf level, only one point is kept. This point is computed as follows. First, we select the closest points of the centroid of the (leaf) voxel, which can be very easily determined based on the space partitioning of the octree. Second, using this neighbour set we (re-)estimate the underlying surface using the MLS algorithm described in section 3.2. Finally, using the estimated polynomials, we project the centroid point on this surface. This latter process allows us to extract point clouds with a lower point density while still maintaining points that are lying on the actual surface. Doing so we can select multiple point clouds with different resolutions and perform the ICP algorithm described in section 3.3 on a coarse-to-fine manner, i.e. a multi-resolutional approach, which is leading to a large speed-up. In addition, the octree data structure allows us to find (approximate) nearest neighbours in a very quick manner as we only have to traverse the tree to the leaf voxel, hence corresponding to an $O(\log n)$ time complexity. In our system, the voxel size at the leaf level was set to 1 centimeter. This turned out to be sufficiently detailed to be able to perform an accurate registration. Figure 4 depicts five images showing how the ground truth model looks at the several resolution levels. Figure 5 depicts the point cloud originated from the Kinect aligned with the ground truth point cloud, serving as the basis of our tracking system.

sequence name	length (m)	duration (s)	avg. ang. velocity (deg/s)	avg. transl. velocity (m/s)	frames
fr1_xyz	7.11	30.09	8.92	0.24	788
fr1_floor	12.57	49.87	15.07	0.26	1214
fr1_desk	9.26	23.40	23.33	0.41	575
fr2_xyz	7.029	121.48	1.716	0.058	3666
fr2_desk	18.880	69.15	6.338	0.193	2965

Table 1: Information about the five video sequences that were used in this work to evaluate our system and to compare it against existing SLAM methods. The sequences are part of the RGB-D SLAM benchmark presented in [7].

4. EVALUATION

In order to evaluate our approach quantitatively, we used five different video sequences that are part of the Freiburg RGBD-SLAM dataset, presented in [7]. All of the sequences were recorded with the Kinect and have both a color video stream and a depth video stream. The sequences come with accurate ground truth and as a result, many SLAM systems have been evaluated on this dataset in the past. As the sequences don't come with a ground truth model of the scene, we cannot perform a global alignment with a ground truth model. However, we create a global 3D map on the fly by fusing the points of consecutive Kinect frames together after their pose has been determined. Doing so, we can also cope with drift, even tough the effect is smaller then when a real ground truth model is available.

We carefully selected the five sequences to cover some of the main difficulties in existing SLAM systems, including a high angular velocity or the absence of salient features. The main specifications of the sequences are listed in Table 1. In Figure 6, three images from these sequences are depicted.

As proposed by Sturm *et. al.* [7], we use the *relative pose error* (RPE) as one of the evaluation metrics. In summary, the RPE measures the local accuracy of the trajectory over a fixed time interval (or number of poses Δ). At time i, the RPE is defined as follows:

$$\mathbf{E_i} := (\mathbf{Q}_i^{-1}\mathbf{Q}_{i+\Delta})^{-1}(\mathbf{P}_{i-1}\mathbf{P}_{i+\Delta}), \qquad (5)$$

Fig. 6: Three images depicting sequences *fr1_desk*, *fr1_floor* and *fr2_desk* from the RGBD-SLAM benchmark presented in [7]. The other two sequences that were used, *fr1_xyz* and *fr2_xyz* are the same scenes as *fr1_desk* and *fr2_desk* but with a different camera trajectory.

sequence name	transl. RMSE	rot. RMSE
fr1_xyz	**0.036** m	**2.44°**
fr1_floor	**0.092** m	**3.85°**
fr1_desk	**0.143** m	**8.03°**
fr2_xyz	**0.011** m	**0.65°**

Table 2: The relative pose error (RPE) for the sequences of the RGB-D Slam benchmark presented in [7]. The RPE is deviating a lot for the different sequences, which is mainly due to the variations in angular velocities, cfr. Table 1. Sequence *fr2_xyz* is giving the best results as its average angular velocity is quite low.

where $\mathbf{P}_i$ and $\mathbf{Q}_i$ are respectively the i-th pose of the estimated trajectory $\mathbf{P}_{1:n} \in \mathrm{SE}(3)$ and the ground truth trajectory $\mathbf{Q}_{1:n}$ in which n represents the number of camera poses in the respective RGBD-sequence. We thus obtain a total of $m = n - \Delta$ individual relative pose errors along the sequence. Subsequently, the *root mean squared error* (RMSE) of the translational component is computed over all pose indices, given by:

$$\mathrm{RMSE}(\mathbf{E}_{1:n}, \Delta) := \left(\frac{1}{m} \sum_{i=1}^{m} ||trans(\mathbf{E}_i)||^2 \right)^{\frac{1}{2}}, \quad (6)$$

where $trans(\mathbf{E}_i)$ refers to the translational components of the relative pose error $\mathbf{E}_i$.

The results of our method for this RPE metric are given in Table 2. As can be noticed, the scores for the different sequences differ a lot from each other, which is mainly due to their difference in (angular) velocity. Our method is prone to quick camera motions because of the 'closest point' criterion of the ICP registration process. This imposes a maximum (rotational) speed given a certain frame rate in order for our solution still to work properly. Sequences *fr1_floor* and *fr1_desk* have a much higher average angular velocity compared to the other sequences. Sequence *fr2_xyz* is giving the best results for our system and with an average of 0.01m translational error and $0.65°$ degrees rotational error it is performing quite accurate.

Alternatively, we also consider the *absolute trajectory error* (ATE) which is often used to compare the performance of different SLAM methods. The ATE is computed by comparing the absolute distances between the estimated and the ground truth trajectory. The ATE at time step i can be computed as

$$\mathbf{F}_i := \mathbf{Q}_i^{-1} \mathbf{S} \mathbf{P}_i, \quad (7)$$

where $\mathbf{S}$ is the rigid-body transformation corresponding to the least-squares solution that maps the estimated trajectory $\mathbf{P}_{1:n}$ onto the ground truth trajectory $\mathbf{Q}_{1:n}$. Similar to the RPE, we use the root mean squared error over all pose indices of the translational components, given by:

$$\mathrm{RMSE}(\mathbf{F}_{1:n}) := \left(\frac{1}{n} \sum_{i=1}^{n} ||trans(\mathbf{F}_i)||^2 \right)^{\frac{1}{2}}. \quad (8)$$

The ATE scores of our solution on the five proposed sequences are listed in Table 3. As can be seen, the results measured by the ATE metric are in line with the results measured by the RPE metric. Sequences *fr1_floor* and *fr1_desk* again have a much higher error because of their high angular velocity. Inherently, pure feature-based methods, such as ORB-SLAM can cope better with this. On the other hand, LSD-SLAM, which is - like ours - a dense and direct approach, is also suffering a lot from high (anglular) velocities of the camera. We can still state that our solution performs better than LSD-SLAM in case of abrupt camera movements. For our solution to work properly, the angular velocity should remain rather low. Sequence *fr2_xyz* is giving the best results by our system and with an ATE of 0.69 it is still outperforming LSD-SLAM and RGBD-SLAM. The systems ORB-SLAM

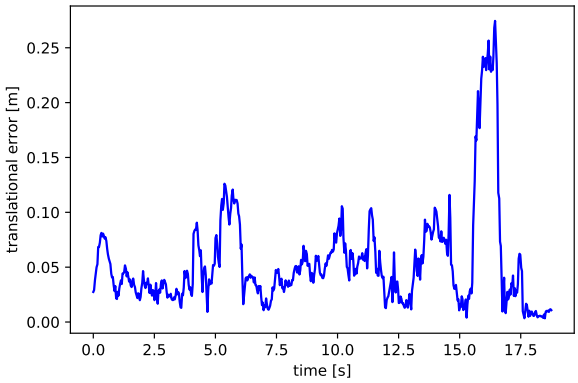

Fig. 7: Graph depicting the drift of sequence *fr1_desk* (with high angular velocity) given by the RPE (RMSE) in function of time. The large ATE of our solution is due to some outlier pose estimates, e.g. the large outlier around 16s. The cause of this is an abrupt movement of the camera at that time.

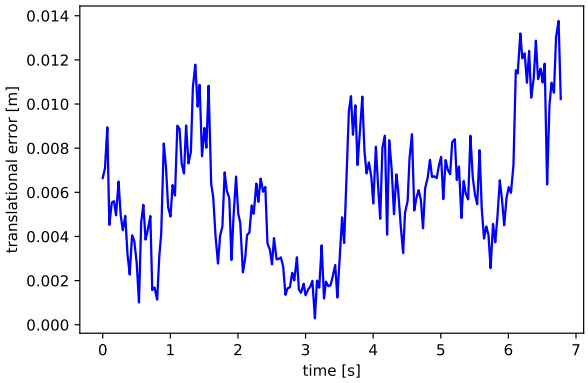

Fig. 8: Graph depicting the drift of sequence *fr2_xyz* (with low angular velocity) given by the RPE (RMSE) in function of time. As in Figure 7, there is still a lot of variation along the sequence but there are no severe outliers of multiple centimeters.

and PTAM are still producing better results. However, they rely on visual features in the scene. The RGBD sequences from [7] are particularly well suited as they are highly textured, but ORB-SLAM and PTAM are a lot more vulnerable when the scene is lacking visual features, as can be seen for the sequence *fr1_floor* leading to an ATE of almost 3.0 instead of an ATE of less than 1.0 in the case of ORB-SLAM.

In order to further analyse the drift of our system, we examined the RPE at different times. For brevity, we only include the graphs for sequences *fr1_desk* (Figure 7) and *fr2_xyz* (Figure 8) as these are the sequences for which our system is respectively worst and best performing. Regarding sequence *fr1_desk*, we can see that the large ATE is due to a some outlier pose estimates, e.g. the large one around 16 s. The cause of this is an abrupt movement of the camera at that time. Figure 8 tells us that for the sequence *fr2_xyz* there is also a large variation for the RPE scores, but there are no sever outlier pose estimates.

5. CONCLUSION

In this paper we proposed a novel camera tracking system incorporating a global 3D map on which a surface reconstruction and re-sampling technique are applied. The experiments demonstrate that our approach has an upper limit regarding the (angular) velocity of the camera. In case this latter is below the limit, we can conclude that our system can compete with state-of-the art visual SLAM methods, while it is only using depth information. This implies that our method can be used in situations where visual methods will, e.g. in case of limited lighting conditions or scenes with very few textured objects. Future work could include the combination of our dense 3D tracking approach with a visual tracker such as ORB-SLAM to further improve robustness and accuracy.

6. ACKNOWLEDGMENTS

This research is part of the ARIA project, an ICON project co-funded by imec, a digital research institute founded by the Flemish Government. The project partner is Evonik, with project support from VLAIO.

7. REFERENCES

[1] Felix Endres, Jürgen Hess, Nikolas Engelhard, Jürgen Sturm, Daniel Cremers, and Wolfram Burgard. An evaluation of the RGB-D SLAM system. In *ICRA*, pages 1691–1696. IEEE, 2012.

[2] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. September 2014.

[3] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. Kinectfusion: Real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 559–568, New York, NY, USA, 2011. ACM.

[4] C. Kerl, J. Sturm, and D. Cremers. Dense visual SLAM for RGB-D cameras. In *Proc. of the Int. Conf. on Intelligent Robot Systems (IROS)*, 2013.

[5] Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth*

Sequence	ORB-SLAM	PTAM	LSD-SLAM	RGBD-SLAM	Kinfu	D-VSLAM	**Ours**
fr1_xyz	0.90	1.15	9.00	1.34	2.6	1.1	**2.45**
fr2_xyz	0.30	0.20	2.15	2.61	-	-	**0.69**
fr1_floor	2.99	-	38.07	3.51	-	-	**6.09**
fr1_desk	1.69	-	10.65	2.58	5.7	2.1	**8.12**
fr2_desk	0.88	-	4.57	9.50	-	1.7	**1.36**

Table 3: The absolute trajectory error (ATE) for the Freiburg sequences of the RGB-D Slam benchmark presented in [7]. Our approach is compared with 6 other state-of-the-art systems. The best result is obtained for sequence *fr2_xyz*, while the sequences *fr1_floor* and *fr1_desk* have the worst outcome. The reason for this is that the first sequence has a very low angular velocity (1.7 deg/s) compared to the latter two sequences (15.1 and 23.3 deg/s respectively).

IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07), Nara, Japan, November 2007.

[6] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. ORB-SLAM: a versatile and accurate monocular SLAM system. *CoRR*, abs/1502.00956, 2015.

[7] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgbd slam systems. In *International Conference on Intelligent Robot Systems (IROS)*, 2012.

[8] 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, Jul 2012.