3D Moving Object Reconstruction by Temporal Accumulation

Anas Abuzaina, Mark S. Nixon, John N. Carter
School of Electronics and Computer Science,
Faculty of Physical Sciences and Engineering,
University of Southampton, UK
Email: aa6g08,msn,jnc@ecs.soton.ac.uk

***Abstract*—Much progress has been made recently in the development of 3D acquisition methods and technologies, which increased the availability of low-cost 3D sensors, such as the Microsoft Kinect. This promotes a wide variety of computer vision applications needing object recognition and 3D shape retrieval. We present a novel algorithm for full 3D reconstruction of unknown moving objects in 2.5D point cloud sequences, such as those generated by 3D sensors. Our algorithm incorporates structural and temporal motion information to build 3D models of moving objects and is based on motion compensated temporal accumulation. Unlike other 3D reconstruction methods, the proposed algorithm does not require ICP refinement, key-point detection, feature description, correspondence matching, provided object models or any geometric information about the object. Given only a fixed centre or axis of rotation, the algorithm integrally estimates the best rigid transformation parameters for registration, applies surface resampling, reduces noise and estimates the optimum angular velocity of the moving object.**

I. Introduction

The increasing availability of low-cost 3D sensors such as the Microsoft Kinect has allowed many 3D reconstruction methods to be developed. The reconstruction of 3D models of rigid objects is generally achieved by the following steps: First the data acquisition step where point clouds or range images (depth maps) are generated by the 3D sensor. This data is 2.5D where only the surfaces facing the sensor are captured. Secondly, an optional segmentation and filtering step is applied to separate the observed object from its background. Thirdly scans from different viewpoints are aligned together in one coordinate frame (registration). Then the aligned scans are typically resampled and merged (integrated) by surface reconstruction techniques into a seamless 3D surface and rendered for display.

We present an algorithm for 3D reconstruction of point clouds that is based on motion-compensated temporal accumulation. Given a fixed centre or axis of rotation, the algorithm estimates the best rigid transformation parameters for registration, and reconstructs the full geometry of rotating 3D objects from 2.5D point clouds, such as those generated by 3D sensors. This algorithm accumulates surface information over the sequence after applying a series of rigid transformations on the point clouds of the input sequence, and then resamples the accumulated point clouds based on a computed space partitioning and votes for the best reconstruction based on the number of points in each resampled point cloud.

The novelty of the algorithm relies in the fact that it does not not require key-point detection, feature description, correspondence matching, any subsequent ICP refinement steps, provided object models or any geometric information about the object. Moreover, the algorithm performs surface resampling, noise reduction and estimates the optimum angular velocity of the moving object integrally.

For this research, we assume that the object is already segmented, the motion is purely rotational, and the angular velocity is approximately constant subject to monotonic change. However the algorithm can be generalised to any rigid motion (translational and rotational).

This paper is structured as follows: first we will discuss the related work to our algorithm in terms of registration and temporal accumulation. Second, we give a detailed overview of our algorithm with establishing the foundations for understanding its methodology. Then we validate and verify the theory and give experimental analysis. After that we show experimental results on synthetic and real data and finally we evaluate the performance of the algorithm in terms of noise and processing time and conclude this paper.

II. Related Work

A. Registration

For rigid objects, registration concerns finding the transformation (rotation and translation) that aligns data sets into one global coordinate system. The goal is to find transformations that align the 2.5 point clouds acquired from different views to one consistent 3D point cloud model correctly representing the object.

There are many methods for 3D data registration which can be classified in a number of ways. Generally, registration methods can be divided to rough (global) and fine registration (local), a detailed review of methods for 3D registration is in [1].

The dominant and most widely used method for local registration of three dimensional data is the ICP (Iterative Closest Point) algorithm [2] and its variants [3]. The high speed of the ICP algorithm and its variants resulted in development of many real time, high quality dense 3D model acquisition methods using structured light [4]–[6], and more recently using 3D sensors, such as the Microsoft Kinect [7]. ICP based methods require an initial alignment between 3D point sets to be known, as it depends on local optimisation and

not guaranteed to the global optimal alignment, and require significant overlap between point clouds to be registered.
Global registration methods usually have a non-iterative approach which depends on the detection of local distinguishable points (key-points) and estimating the correspondences between point sets by computing and comparing feature descriptors of such key-points [8]–[10] . A recent comprehensive list and evaluation of 3D key-points detectors is presented in [11]. The complexity of feature-based approaches typically depends on the resolution of the solution. These approaches are often used to compute an initial alignment and are followed by fine registration algorithms such as the ICP.
There have been other methods attempting to achieve *automatic* global registration by transforming the 3D point clouds to different domains, and then the resultant images are compared and correlated in those domains to obtain the transformation aligning the input point clouds, or point correspondences between them, such as in [12], [13]. Such methods are limited to special cases, such as point clouds with smooth or isometric surfaces, or availability of normal components. Moreover in many cases the transformation to other spaces can be non-informative and ambiguous.

B. Temporal Accumulation

A temporal accumulation algorithm was used to determine bulk motion of walking people in 2D image sequences for gait analysis purposes [14]. [15] used temporal accumulation of Fourier descriptors to extract arbitrarily moving arbitrary shapes by tracing a locus of votes in the form of the template shape, adjusted for the estimated motion of the object relative to the time reference of each frame. [16] perform temporal evidence accumulation on stereo image sequences for extraction of specified objects undergoing linear motion. Temporal accumulation was used in [17] within an evidence gathering algorithm which incorporates structural and motion parameters to detect moving spheres in point cloud sequences.
A colour-augmented search algorithm is used in [18] to accumulate coloured point clouds from successive time frames for a moving vehicle, to track the vehicle and build 3D a model of it. Their algorithm requires 3D point clouds and a colour image to align successive frames, the alignment is found by a pre-filtered iterative coarse to fine optimisation by first aligning the centroids of each colour-interpolated point cloud and then projecting them to a 2D occupancy grid and then by voting based on Euclidean as well as colour difference between point pairs.
This approach is similar to ours in terms of using the best alignment of the point clouds of the object to estimate velocity. However it is very different in its methodology for finding the alignment and input data. Their algorithm assumes a small motion between frames, and hence alignment based on centroid locations is most likely to be a "good enough" rough initial alignment. If the motion is large, which leads to no significant overlap between frames, alignment would fail and hence the registration would be false.
Our algorithm does not require any colour information, computation of object centroids, projecting to a 2D plane or significant overlap between frames.

III. Algorithm

```
for every point cloud $P_t$ ($t = 0; t < N; t{+}{+}$) do
    for every angular velocity $w_j$
    ($w_j = w_{min}; w_j = w_{max}; w{+}{+}$) do
        Rotate $P_t$ by $R(w_j, t)$
        Add to velocity cloud $V_j$
    end
    Save $V_j$ in $A$
end
for every $V_j$ in $A$ do
    Build an octree representation $O_j$ for $V_j$
    for every occupied voxel in $O_j$ do
        Find centroid of points within voxel
        Add centroid point to downsampled cloud $D_j$
    end
    Save $D_j$ in $B$
end
Search $B$ for $D_j$ with minimum number of points.
```

Algorithm 1: 3D object reconstruction by temporal accumulation.

The proposed algorithm finds the registration of a series of point clouds based on motion-compensated temporal accumulation. Motion compensation is simply back-projection along the expected line of motion to convert the coordinates to the same temporal frame of reference as the initial frame. The algorithm takes each cloud in the sequence of a rotating 3D object and applies a series of rigid transformations based on velocities given in a predefined range. Transformed point clouds are accumulated together based on their velocity, each velocity in the range will have a corresponding "velocity cloud" that is the accumulation of point cloud transformed by a transformation corresponding to that velocity. An octree-based [19] space partitioning algorithm is applied to downsample each velocity cloud, and the resulting downsampled velocity cloud with the minimum number of points is taken as the truest representation of the object.

A. Rotation in 3D

By definition, a rotation R about an origin is a transformation that preserves the origin; so rotations centre on the coordinate system, as matrix or vector multiplication has no effect on the zero vector (the coordinates of the origin). To rotate an object P around any fixed point $a = (a_x, a_y, a_z)$ in 3D space, we take the fixed point as the origin of a Cartesian coordinate system, then apply the rotation. This is done by applying a translation vector $T = -[a_x, a_y, a_z]^\top$, then rotation, then translating back to the point of rotation, given by:

$$P' = TRT^{-1}P \tag{1}$$

Rotations in 3D can be represented by the axis-angle representation in which a 3D unit vector represents an axis and a scalar angle describes the magnitude of the rotation about that axis. Six degrees of freedom are needed to describe a rotation in 3D; three to determine position of the axis or point of rotation, two for the normalized direction vector of the axis and one for the scalar angle.

B. Angular Velocity

Having a sequence $S = \{P_0, P_1, \ldots, P_{N-1}\}$ of N point clouds, where each point cloud P_t represents a set M of 3D points distributed in $\mathbb{R}^3$, $P = \{p_0, p_1, \ldots, p_{M-1}\}$, $p_i = \{x_i, y_i, z_i\}$ of a rigid object rotating around a fixed axis of a direction given by the unit vector $u = [u_x, u_y, u_z]$, and passing through the point $a = (a_x, a_y, a_z)$,with angular velocity w, given by the equation:

$$w(\theta, u) = \frac{d\theta}{dt} u \tag{2}$$

which is defined as the rate of change of angular displacement θ measured by degrees per unit time t, which with a time reference relative to the arbitrary start point (e.g. point cloud number versus point cloud 0). The angular velocity describes the speed of rotation and the orientation of the instantaneous axis about which the rotation occurs. The first step of the algorithm is to rotate each point cloud in the sequence by a series of rotations; the instantaneous rotation angle θ is given by multiplying the angular velocity w by time t:

$$\theta = wt \tag{3}$$

where $w \in W = [w_{min}, w_{max}]$. At any angular velocity, according to [20], the instantaneous rotation by an angle of θ about an axis in the direction of unit vector u is given by:

$$R(\theta, u)|_{w,t} =$$

$$\begin{pmatrix} \cos\theta + u_x^2 c & u_x u_y c - u_z \sin\theta & u_x u_z c + u_y \sin\theta \\ u_y u_x c + u_z \sin\theta & \cos\theta + u_y^2 c & u_y u_z c - u_x \sin\theta \\ u_z u_x c - u_y \sin\theta & u_z u_y c + u_x \sin\theta & \cos\theta + u_z^2 c \end{pmatrix} \tag{4}$$

where $c = (1 - \cos\theta)$. If the position and direction of the axis of rotation are given, to find the rotation we need to solve for only one degree of freedom, that is the angle of rotation θ.

However if only a point of rotation is known, i.e. the direction of the axis of rotations remains unknown, the dimensionality of the problem increases, as there are three degrees of freedom to be solved. This can be solved by applying equation (4) three consecutive times; by taking three angles around the three main axis. thus the angular velocity will be presented by three angles instead of an angle and an axis, as given by:

$$w = \left(\frac{d\gamma}{dt}\hat{x}, \frac{d\phi}{dt}\hat{y}, \frac{d\psi}{dt}\hat{z} \right) \tag{5}$$

where γ, ϕ, ψ are angles of consecutive rotations about the X, Y, Z axis respectively, and $\hat{x}, \hat{y}, \hat{z}$ are the unit vectors of these axes.

C. Temporal Accumulation

In the second stage of the algorithm, all point clouds rotated by the rotation corresponding to the same angular velocity are accumulated together; each angular velocity $w_j \in W$ will be represented as a point cloud V_j which is the sum of all object points of all point clouds rotated by this velocity.

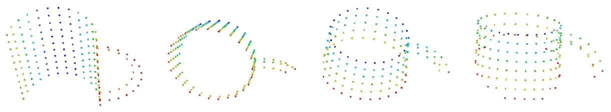
Fig. 1: Four frames from a 2.5D sequence of rotating object.

$$V_j = \sum_{t=0}^{N-1} P'_{j,t} = \sum_{t=0}^{N-1} TR|_{w_j,t} T^{-1} P_t \tag{6}$$

The total temporal accumulation process A of the complete sequence can be summarised by the equation:

$$A = \bigcup_{j=w_{min}}^{w_{max}} V_j \tag{7}$$

For each accumulated "velocity point cloud" V_j, the 3D space is spatially partitioned into fixed-sized volumes (voxels) using an octree data structure [19], such that all points $p_i \in V_j$ are associated with a voxel $v_k \in O_j$, where O_j is the octree representation of V_j.

Then a downsampling process is performed such that all points within each of the voxels in the octree are approximated with only one point which is their centroid. All velocity point clouds will have the same number of points before downsampling. However after downsampling, points from different point clouds transformed by a rotation corresponding to the correct angular velocity will theoretically coincide, or near enough to be in the same voxel, requiring less voxels in the voxel grid, and subsequently less points in the downsampled velocity point cloud D_j, figure 2. Hence the downsampled velocity point cloud with the least number of points (D^*) is the truest alignment of the unknown 3D object:

$$D^* = \underset{M}{\operatorname{argmin}}(B) \tag{8}$$

where B is the vector of the downsampled velocity point clouds and M is the number of points in each cloud.

$$B = \bigcup_{j=w_{min}}^{w_{max}} D_j \tag{9}$$

We can also find the best estimate for the object's angular velocity (w^*) by this equation:

$$w^* = \arg\min_{w \in W}(B) \tag{10}$$

IV. Analysis

The algorithm is verified by implementing it on a 2.5D synthetic sequence of a rotating object. Figure 1 shows four point clouds from a sequence of thirty of an object rotating about the horizontal X axis of the origin $(0, 0, 0)$. Only the front surface of the object is available to resemble real data generated by depth sensors. The object has an angular velocity of 20 degrees per unit time (dt^{-1}). Given a range of velocities $[0, 100]$ with a step of 10 dt^{-1}, we apply the algorithm; the point clouds are transformed and accumulated

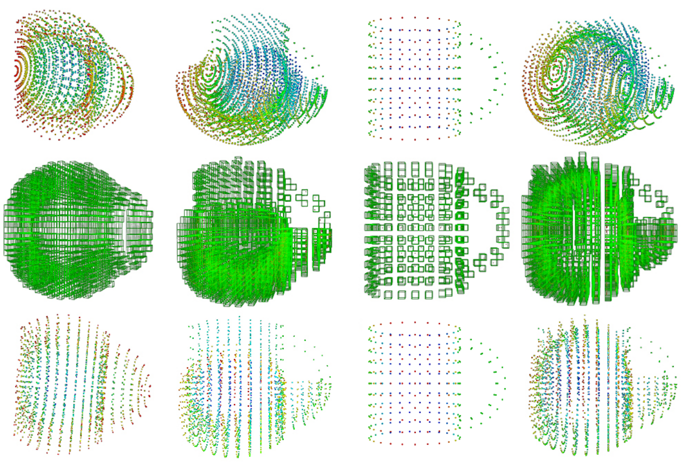

Fig. 2: Steps of proposed algorithm. Top row shows velocity clouds (0-3) accumulated from the 2.5D sequence according to angular velocities (0-30) dt^{-1} respectively. Middle row shows the octree space partitioning of the velocity clouds into voxels. Bottom row shows the downsampled velocity clouds. The correct representation of the object is the downsampled point cloud with minimum number of points.

into the velocity clouds, the 3D space of each velocity point cloud is partitioned into voxels by an octree structure, and downsampled, figure 2. Note that all velocity clouds have the same number of points before downsampling.

If we take the histogram of the number of points in the voxels, the velocity cloud corresponding to the correct velocity should have a histogram with a peak value of the number of point clouds in the sequence, 30 in this case, as each correctly rotated point cloud accumulates one point. However because the sequence is 2.5D, as with real data generated by depth sensors, the histogram should be centred at approximately half the number of point clouds. Histograms of the rest of the accumulated velocity clouds will be centred towards the left at low number of points in the voxel and typically have multiple peaks with higher values for numbers of voxels, figure 3.

This observation is based on the choice of voxel size relative to the density of the point clouds; in this case the voxel size was chosen to be small enough to allow only coincident or very close points to exist within the voxel. Larger voxels will still verify this observation, but will allow more points to be within them resulting for higher number of peaks. The angular velocity also has an effect on the shape of the histogram; the larger it is, the smaller the overlap area between point clouds is and hence more voxels containing less points. Additionally, the cloud's point density and the isometry of the object also affect the shape of the histogram. Nevertheless the significant property of the accumulated velocity clouds is the distribution of the histogram rather than the number of peaks or their amplitude.

V. Results

Figure 4 show 2.5D point clouds from four sequences, one synthetic and three real. And figure 5 shows the reconstructed full 3D point clouds generated using the proposed algorithm.

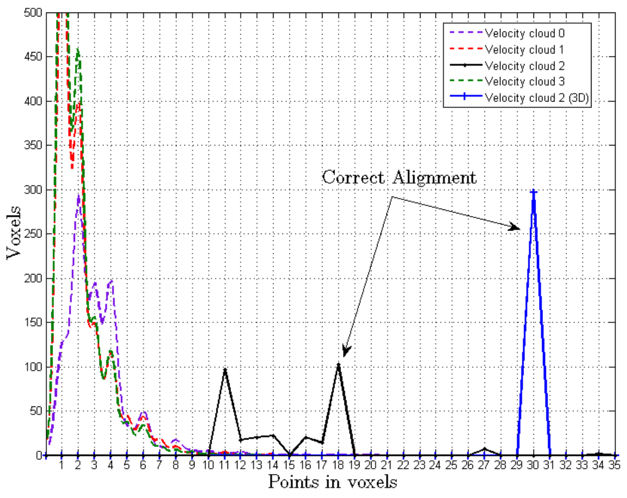

Fig. 3: Histograms of accumulated velocity clouds. It is clear that the histogram of velocity cloud 2 is centered on higher number of points, thus it is considered the most correct alignment. The blue histogram is generated from the same velocity cloud but from a 3D sequence instead of 2.5D. Note that the peak is centered at the number of point clouds in the sequence (30), which verifies the methodology of the algorithm.

A. Surface Resampling and Reconstruction

The final step of every 3D model acquisition method is to reconstruct the surface of the object after estimating the alignment between multiple scans. When point clouds are registered to one model, overlapping areas of different point clouds coincide, resulting for the density of points to be multiplied. Moreover, typical quantisation errors and missing data from 3D sensors can generate empty areas in the point cloud. This variety of point density on the registered model negatively affects any subsequent rendering or recognition processes. This problem is solved by resampling. Whether it is up-sampling or downsampling, resampling methods aim to smooth surfaces to have an approximately constant point density, a robust resampling algorithm is presented in [21].

In our algorithm, resampling is integral; it is achieved in the stage before last of our algorithm when the points in octree voxels are approximated with their centroid. The resultant downsampled point cloud will have an approximately uniform point density; the density is dependent on the voxel size used in the octree.

Detailed surface reconstruction is beyond the scope of this research, here the Poisson surface reconstruction algorithm [22] is applied on the resultant cloud that is the output of our algorithm to reconstruct the 3D surface of the rotating object, figure 6. For more accurate resampling that will produce smoother reconstructed surfaces, an upsampling and hole filling algorithm such as in [21] can be applied after using our algorithm to obtain the correct object velocity, and transform and accumulate the original point clouds of the sequence.

VI. Evaluation

A. Noise

Even with significantly corrupted input data, our algorithm was still able to detect the correct angular velocity

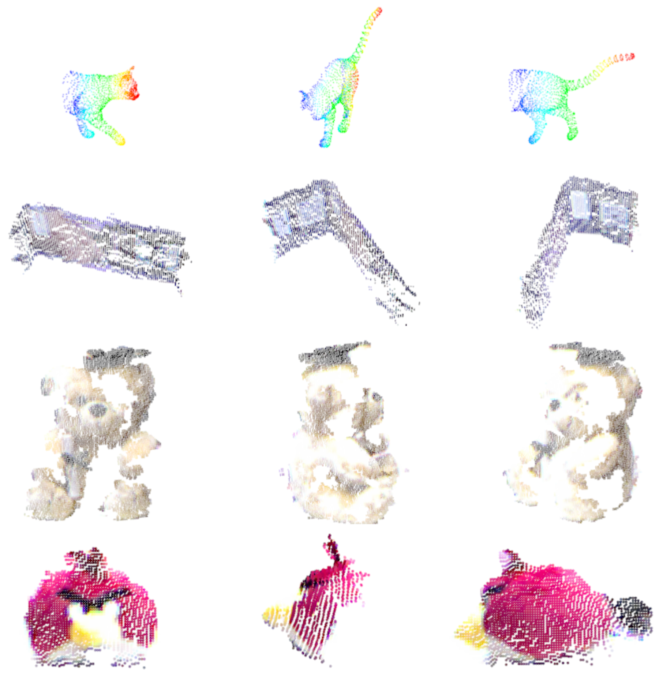

Fig. 4: Three point clouds from four 2.5 sequences. Top row is of a synthetic sequence of 18 frames. Bottom three rows are of real objects placed on a turn-table, each sequences has 8 frames.

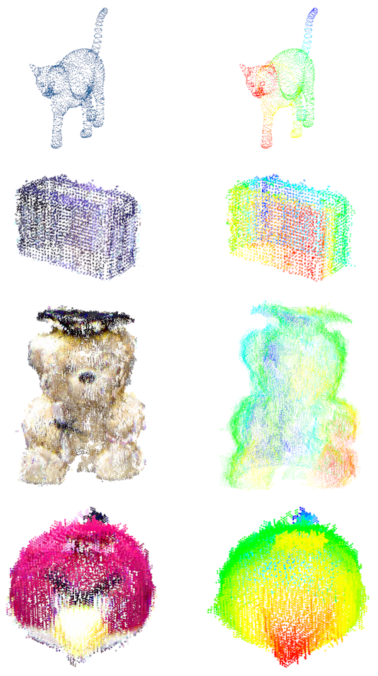

Fig. 5: Reconstructed objects by proposed algorithm.

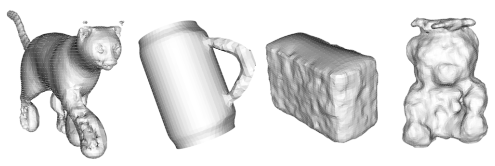

Fig. 6: Poisson surface reconstruction.

and reconstruct the object. Moreover, the noise is integrally reduced in the reconstructed point cloud as a result of space partitioning and resampling stages of our algorithm, figure 5. The performance of the algorithm is quantified against noise by conducting an actual implementation on data corrupted with noise. Figure 8 shows the performance of the algorithm applied on a synthetic object with the presence of noise; to give a useful meaning to the noise readings, we introduce the error metric ρ, given by the ratio of mean cloud-to-cloud distance to the width of the object, equation (12). The mean cloud-to-cloud (C2C) distance is the mean of all distances from each point in the reconstructed point cloud to its closest point in the original ground truth model.

$$\rho = \left(\frac{1}{N} \sum_{c \in C}^{N} \min_{g \in G} \|c - g\| \right) / width \times 100\% \tag{11}$$

where N is the number of points in the reconstructed cloud C, and G is the ground truth model.

The noise is modeled by an added Gaussian distribution model with increasing standard deviation (σ) and zero mean (μ), given by the function:

$$N(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \tag{12}$$

The performance of the algorithm against noise is shown in figure 8.

B. *Time*

The performance of the algorithm in the case of given axis of rotation is $O(N \cdot M \cdot W)$ while in the case of a point of rotation is $O(N \cdot M \cdot W^3)$. Where N is the number of point clouds, M is the average number of points in each point cloud, and W is the number of velocities in the given range. Table 1 shows the algorithm's processing time.

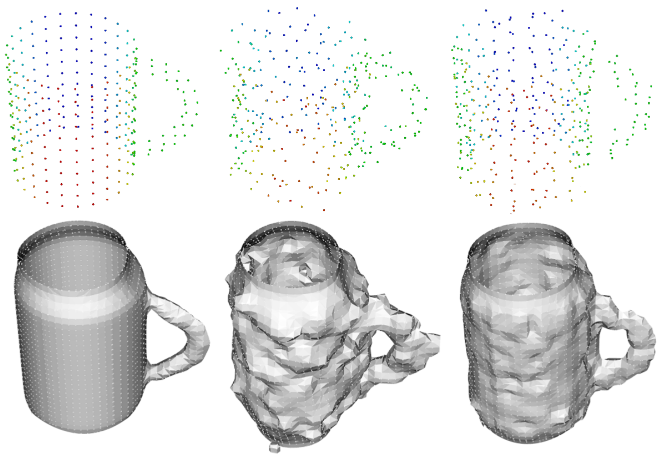

Fig. 7: Noise reduction. Top row: Original point cloud from sequence of 30 clouds (left), added Gaussian noise (σ = 0.004) (middle), and result of implementing the proposed algorithm on noisy sequence (right). Bottom row shows the surface reconstruction of each point cloud.

	Mug	Cat	Box	Teddy	Bird
No. Clouds	30	18	8	8	8
Total Points	4896	30600	33453	58485	18464
No. Velocities	10	150	10	30	45
Time (s)	0.16	13.01	1.46	6.65	3.25

TABLE I: Algorithm performance. (running on 2.4 GHz Intel Core i7 and 4GB RAM).

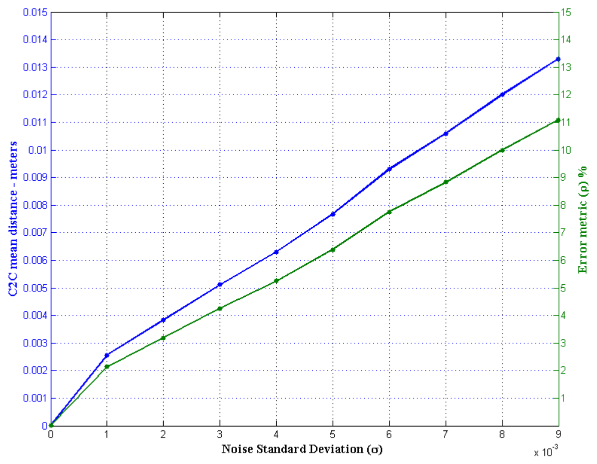

Fig. 8: Performance against noise. The error metric at no added noise is $\rho = 0.0044\%$, which assures the accuracy of the algorithm. The object dimensions are $(0.120 \times 0.095 \times 0.080)$ meters^3.

VII. Conclusions

We presented a novel algorithm for full 3D reconstruction of unknown moving objects in 2.5D point cloud sequences. Our algorithm incorporates structural and temporal motion information to build 3D models of moving objects that is based on motion compensated temporal accumulation. We verified the theory of the algorithm, provided experimental analysis, showed experimental results on synthetic and real data and finally we evaluated the performance of the algorithm. Unlike other 3D reconstruction methods, our algorithm does not require ICP refinement, key-point detection, feature description, correspondence matching, provided object models or any geometric information about the object. We demonstrated how the algorithm integrally estimates the best rigid transformation parameters for registration, applies surface resampling, reduces noise, estimates the optimum angular velocity of the moving object and finally fully reconstructs unknown moving objects.

References

[1] J. Salvi, C. Matabosch, D. Fofi, and J. Forest, “A review of recent range image registration methods with accuracy evaluation,” *Image and Vision Computing*, vol. 25, no. 5, pp. 578–596, 2007.

[2] P. J. Besl and N. D. McKay, “Method for registration of 3-d shapes,” in *Robotics-DL tentative*. International Society for Optics and Photonics, 1992, pp. 586–606.

[3] S. Rusinkiewicz and M. Levoy, “Efficient variants of the icp algorithm,” in *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*. IEEE, 2001, pp. 145–152.

[4] T. Weise, T. Wismer, B. Leibe, and L. Van Gool, “In-hand scanning with online loop closure,” in *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 1630–1637.

[5] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy, “Real-time 3d model acquisition,” in *ACM Transactions on Graphics (TOG)*, vol. 21, no. 3. ACM, 2002, pp. 438–446.

[6] T. Weise, B. Leibe, and L. Van Gool, “Accurate and robust registration for in-hand modeling,” in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[7] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon, “Kinectfusion: Real-time dense surface mapping and tracking,” in *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, 2011, pp. 127–136.

[8] R. B. Rusu, N. Blodow, and M. Beetz, “Fast point feature histograms (fpfh) for 3d registration,” in *Robotics and Automation, 2009. ICRA’09. IEEE International Conference on*. IEEE, 2009, pp. 3212–3217.

[9] N. Gelfand, N. J. Mitra, L. J. Guibas, and H. Pottmann, “Robust global registration,” in *Proceedings of the Third Eurographics Symposium on Geometry Processing*, ser. SGP ’05. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2005.

[10] J. Novatnack and K. Nishino, “Scale-dependent/invariant local 3d shape descriptors for fully automatic registration of multiple sets of range images,” in *Computer Vision–ECCV 2008*. Springer, 2008, pp. 440–453.

[11] F. Tombari, S. Salti, and L. Di Stefano, “Performance evaluation of 3d keypoint detectors,” *International Journal of Computer Vision*, pp. 1–23, 2013.

[12] A. Censi and S. Carpin, “Hsm3d: feature-less global 6dof scan-matching in the hough/radon domain,” in *Robotics and Automation, 2009. ICRA’09. IEEE International Conference on*. IEEE, 2009, pp. 3899–3906.

[13] A. Makadia, A. Patterson, and K. Daniilidis, “Fully automatic registration of 3d point clouds,” in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1. IEEE, 2006, pp. 1297–1304.

[14] D. K. Wagg and M. S. Nixon, “On automated model-based extraction and analysis of gait,” in *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*. IEEE, 2004, pp. 11–16.

[15] M. G. Grant, M. S. Nixon, and P. H. Lewis, “Extracting moving shapes by evidence gathering,” *Pattern Recognition*, vol. 35, no. 5, pp. 1099–1114, 2002.

[16] J. A. Artolazábal and J. Illingworth, “3dsvht: extraction of 3d linear motion via multi-view, temporal evidence accumulation,” in *Advanced Concepts for Intelligent Vision Systems*. Springer, 2005, pp. 563–570.

[17] A. Abuzaina, T. Alathari, M. S. Nixon, and J. N. Carter, “Detecting moving spheres in 3d point clouds via the 3d velocity hough transform,” in *IVMSP Workshop, 2013 IEEE 11th*. IEEE, 2013, pp. 1–4.

[18] D. Held, J. Levinson, and S. Thrun, “Precision tracking with sparse 3d and dense color 2d data,” in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, 2013, pp. 1138–1145.

[19] D. J. Meagher, “High-speed image generation of complex solid objects using octree encoding,” Sep. 15 1987, US Patent 4,694,404.

[20] C. J. Taylor and D. J. Kriegman, “Minimization on the lie group so (3) and related manifolds,” *Yale University*, 1994.

[21] R. B. Rusu, N. Blodow, Z. Marton, A. Soos, and M. Beetz, “Towards 3d object maps for autonomous household robots,” in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*. IEEE, 2007, pp. 3191–3198.

[22] M. Kazhdan, M. Bolitho, and H. Hoppe, “Poisson surface reconstruction,” in *Proceedings of the fourth Eurographics symposium on Geometry processing*, 2006.