Spatially Stratified Correspondence Sampling for Real-Time Point Cloud Tracking

Jeremie Papon, Markus Schoeler, and Florentin Wörgötter
Bernstein Center for Computational Neuroscience (BCCN)
III. Physikalisches Institut - Biophysik, Georg-August University of Göttingen
jpapon@gwdg.de, mschoeler@gwdg.de, worgott@gwdg.de

Abstract

In this paper we propose a novel spatially stratified sampling technique for evaluating the likelihood function in particle filters. In particular, we show that in the case where the measurement function uses spatial correspondence, we can greatly reduce computational cost by exploiting spatial structure to avoid redundant computations. We present results which quantitatively show that the technique permits equivalent, and in some cases, greater accuracy, as a reference point cloud particle filter at significantly faster run-times. We also compare to a GPU implementation, and show that we can exceed their performance on the CPU. In addition, we present results on a multi-target tracking application, demonstrating that the increases in efficiency permit online 6DoF multi-target tracking on standard hardware.

1. Introduction

Visual tracking is a crucial challenge for many computer vision applications such as visual surveillance, action recognition, and robotic demonstration learning. In these, visual tracking serves as a precursor to higher-level inference, making robust tracking fundamental to their success. Multi-target visual tracking (MTVT) is a well-established field which goes back over thirty years [6]. One popular tracking methodology is Sequential Bayesian Filtering (SBF), which recursively determines the time-changing posterior distribution of target states conditioned on previous observations. Particle Filtering has received considerable attention as a method of approximating this posterior. It was first introduced to the vision community by Isard and Blake [8] for single targets, and was subsequently extended to multiple targets [7, 19, 20].

There are two standard approaches that have been used to extend the Particle Filter to multiple targets. The first represents all targets jointly in a single particle filter by assigning individual particles to particular labels [18]. This means that, for a given total number of particles, there will be fewer for each individual target - resulting in reduced accuracy. The second approach is to add additional dimensions to the state space for each additional target [17]. Unfortunately, this approach quickly increases the dimensionality of the state space, which also results in a need for a very high number of particles for the filter to remain accurate.

In both of the above approaches, the computational complexity increases exponentially as targets are added (for constant level of accuracy). As a consequence of this, it is beneficial to use a separate particle filter for each target. One way of doing this is to add factors to the observation and/or process models of the filters which explicitly model occlusions and interactions between targets [9, 13]. Alternatively, one can use a discrete processing step to resolve the association of target detections [10, 1].

While these approaches have attempted to address the problem of multiple targets, in general they suffer from one fundamental problem - they all significantly increase the computational resources required. This increase can be seen in the need for more particles - due to assignment of particles to individual targets, a larger state space, or independent filters for each target. While there has been work addressing this problem by offloading processing to a GPU [3], in this work we take a different approach, and search for fundamental changes to the point cloud correspondence particle filter which can reduce computational complexity without affecting accuracy.

The primary contribution of this work is the use of a supervoxel-based stratified sampling approach to greatly reduce the computational complexity of point cloud correspondence particle filtering. We show that the approach allows performance (on a standard CPU) exceeding that which can be obtained on a recent GPU implementation [3]. Additionally, we present extensive experiments demonstrating the benefits of this approach, as well as show qualitative results from a real-world application. We have released

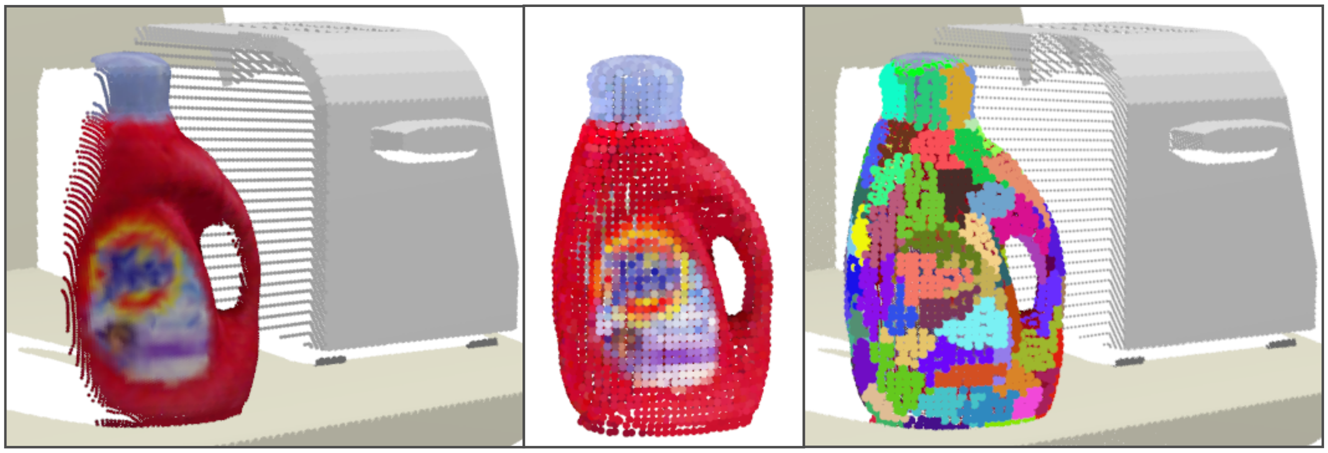


Figure 1. Example of data from "Tide" sequence. The left frame shows an example of the raw input cloud. Sampling effects from the synthetic RGB-D camera are visible in the quantization of points, especially on the edges of objects. The middle frame shows the voxelized model representation we use, while the right frame shows an example of supervoxel strata used for sampling with $R_{seed} = 0.07m$.

code for our tracker as part of the Point Cloud Library [1] so that the community may take advantage of its quick performance in their applications.

This paper is organized as follows: In Section 2 we present the point cloud correspondence-based particle filter, and then present the sampling approach in Section 3. We then give results from experiments on both synthetic and real-world data in Section 4 and discuss the impact of sampling on computational complexity and tracker error.

2. Particle Filters in 3D

The underlying mechanics of 3D point cloud correspondence particle filtering remain the same as in other particle filters, and so we shall not discuss them extensively here; for a detailed introduction to the topic, we refer the reader to [5] or [20]. Rather, we shall only discuss the aspects that differentiate it - the models and how they are scored and propagated. The models here consist of point clouds, and the measurement function relies on point to point correspondence for scoring, rather than a global per-detection metric (such as a histogram distance, commonly used in 2D trackers). The dynamic model uses real-world 3D coordinates which also include orientation, rather than 2D pixel coordinates in the image plane. The primary novelty of the approach we present here lies in how we score individual particle predictions using the measurement model.

2.1. Model Representation

In this work we use voxelized 3D models of tracked objects, allowing tracking through any change in pose, and additionally allowing accurate tracking of pose itself. We represent objects as clouds of voxels corresponding to the surface of the object. A visual representation of such a model is given in Figure 1.

Points for objects are stored in a model-centered reference frame (which we shall denote with superscript ^{m}), with each containing an XYZ position, an HSV color, as well as a surface normal vector. That is, each point p of the model k consists of a nine-dimensional vector:

$$p_k^m = [x^m, y^m, z^m, H, S, V, n_x, n_y, n_z], \tag{1}$$

and a model for an object O_k consists of a vector of n_k such points p^m:

$$O_k^m = [p_0^m ... p_{n_k}^m]. \tag{2}$$

Points of an object model given above are model-relative - they must be transformed into the world coordinates in order to evaluate their fit to observations.

2.2. Dynamic Model

Our trackers use a time-dependent state vector consisting of translation and rotations around the object reference frame x-axis (roll - γ), y-axis (pitch - β), and z-axis (yaw - α). This yields a position state vector for particle j at time t of

$$\mathbf{x}_t^j = [d_x, d_y, d_z, \gamma, \beta, \alpha]. \tag{3}$$

Each object model is tracked using a set of N such particles. Additionally, we have velocity state vector

$$\mathbf{v}_t = [v_x, v_y, v_z, v_\gamma, v_\beta, v_\alpha], \tag{4}$$

which is not tracked individually per particle, but rather as a whole for the model. While the use of independent per-particle velocity states potentially helps in complicated tracking scenarios, in our experiments we were unable to observe any tangible benefit. Moreover, in order to avoid

[1]http://www.pointclouds.org/

instability in the tracking results we needed to significantly increase the number of particles for a given noise level. As such, we have chosen to use the "group-velocity", and leave it to future work to investigate the possibility of independent velocity states.

Motion is modeled using a constant velocity model in discrete time with a variable sampling period T, giving the dynamic model

$$\mathbf{x}_t = \mathbf{x}_{t-1} + T\mathbf{v}_{t-1} + \omega, \tag{5}$$

with noise vector ω assumed to be zero mean Gaussian with fixed covariance. Particle velocities are updated after weighting of individual particles using the measurement model, and are a weighted average of the change in position

$$\mathbf{v}_t = \frac{1}{TN}\sum_{j=1}^{N} w_j(\mathbf{x}_t^j - \mathbf{x}_{t-1}^j), \tag{6}$$

where w_j is the normalized weight for particle j.

2.3. Measurement Model

As points for the model are given in a model-centered frame of reference, $p^m = [x^m, y^m, z^m, 1]$, we must transform them to the world frame using a 3D affine transformation quaternion. This yields positions in the world frame for each of our η model points for a particular particle j:

$$\begin{bmatrix}\mathbf{p}_1^j\\ \mathbf{p}_2^j\\ \vdots\\ \mathbf{p}_\eta^j\end{bmatrix}\begin{bmatrix}[x_1, y_1, z_1, 1]^\mathsf{T}\\ [x_2, y_2, z_2, 1]^\mathsf{T}\\ \vdots\\ [x_\eta, y_\eta, z_\eta, 1]^\mathsf{T}\end{bmatrix} = diag(\mathbf{B}^j)\begin{bmatrix}[x_1^m, y_1^m, z_1^m, 1]^\mathsf{T}\\ [x_2^m, y_2^m, z_2^m, 1]^\mathsf{T}\\ \vdots\\ [x_\eta^m, y_\eta^m, z_\eta^m, 1]^\mathsf{T}\end{bmatrix}. \tag{7}$$

Once we have our transformed points, we then must establish correspondences between each particle's model points and a world point so that we can score how well a state matches the current world model observation. That is, for each transformed point $\mathbf{p}_{1\ldots\eta}^j$, we select corresponding point $\mathbf{p}^*$ in the observation which has minimal spatial distance.

To find these correspondences, we first compute a KD-tree of the spatial dimensions of the world model points. This allows us to efficiently search for the nearest correspondence for each transformed point. We create this tree for the world model rather than the transformed model (even though the former has more points) as there is only one world, but many particles and models. Computing it for the models would require a KD-tree for each particle in each model. Additionally, computing it for the world allows us to take advantage of sampling strategies which significantly reduce run-time complexity. Finally, using the world-model allows us to take advantage of the sequential octree first presented in [15], greatly improving performance in the case of full and partial occlusions.

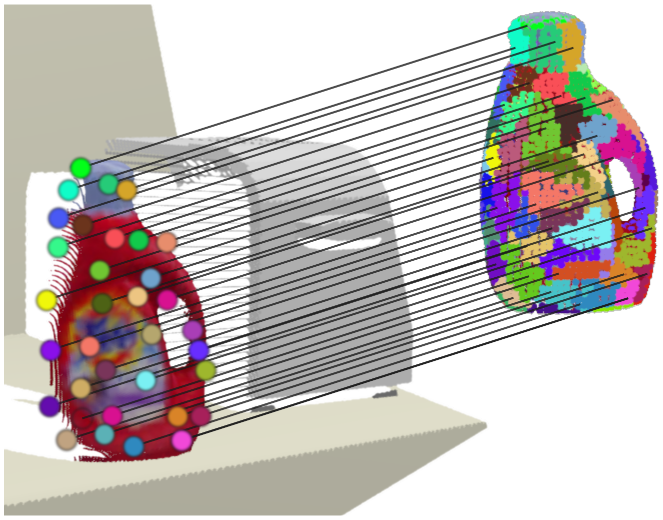


Figure 2. The model is divided into strata (shown as separate colors) by the supervoxel algorithm. Each particle independently selects a random sample (or samples) from each stratum for correspondence matching, and then searches for a correspondence for it in the observation.

Once we have selected (with replacement) an observed point correspondence for each model point, we must calculate a weight $\tilde{w}^j$ corresponding to the global similarity of the transformed points to the world observation. This is accomplished by summing the individual correspondence scores computed using weighted Euclidean distance in normalized world-spatial-, color-, and normal-space. In our experiments we set the weighting factors to have a 1:1:2 ratio (spatial:normal:color), as this balances the scoring between color and geometric shape, and found experimentally that it produced consistently good tracking results. The calculated particle weights $\tilde{w}^j$ are then normalized, and a final state estimate can be computed by taking the weighted average of all particles

$$\mathbf{x}_t = \sum_{j=1}^{N} w_j \mathbf{x}_t^j, \tag{8}$$

and the group-velocity can be computed using Equation 6.

3. Supervoxel Stratified Correspondence Sampling

While the tracking methodology discussed above works, in practice its run time performance is very poor, even for single objects. Moreover, speed of tracking is highly dependent on the size of object models as well as voxel resolution used. To address this, we propose a sampling scheme which selects a limited number of points from the model to transform and test. By doing this, we achieve linear asymptotic time complexity for the particle filter with respect to the number of particles - there is no dependence on the number

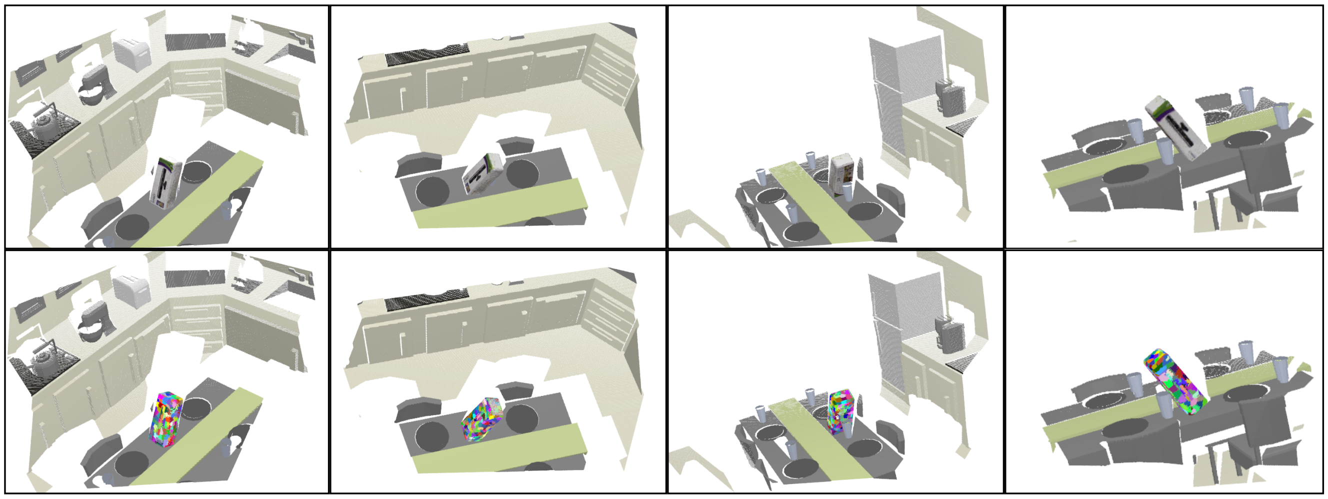


Figure 3. Tracking on the artificial "Kinect Box" sequence. The top row shows tracked output overlaid on input data, while the bottom row shows the supervoxl strata that are used for sampling.

of points in the models or the voxel resolution used. The only step which is dependent on the number of input points is the KD-tree construction, but this is only done once for the world model independent of the number of trackers, and is done as a pre-processing step regardless (for normal computation).

The proposed sampling scheme is as follows. We select a spatial sampling resolution R_{seed} based on the number of desired sample points per particle N_s. We then divide the model into strata, where each stratum is a supervoxel using the method of Papon et al.[14]. Supervoxels are a voxel-based surface patch representation that use connectivity, colors, and normals so that their edges conform well to object part boundaries. The strata are evenly divided over the spatial structure of the model, as seen in Figure 1. Additionally, using supervoxels as the strata ensures that we sample the important features of the models - for example in the model of Figure 1, we have a stratum for the brand logo, as well as ones for the concavities of the handle.

For each particle, we randomly select a point from each stratum using uniform sampling, and then transform and score it as described in the previous Section. As an additional step, we also select $\frac{N_s}{4}$ points uniformly from the entire model. Using strata reduces the noise which occurs when sampling from the whole model exclusively, while sampling randomly from the entire distribution improves occlusion performance.

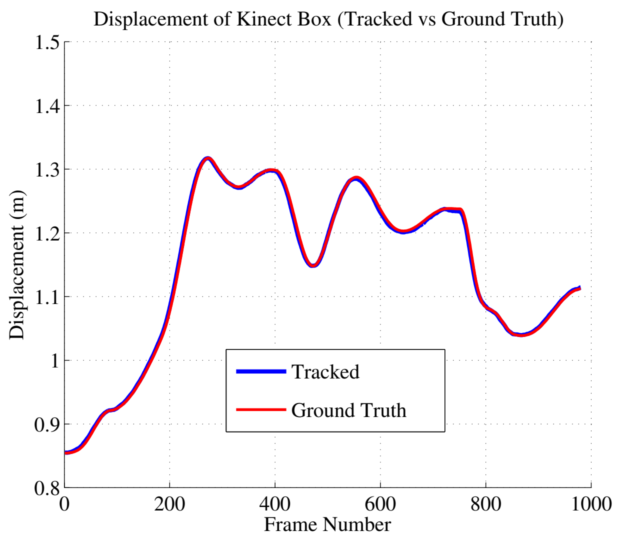
While sampling will tend to produce noisier tracking results for low N_s, it also greatly reduces the computational complexity, as we only need to transform and test a small subset of the model points. This allows one to greatly increase the number of particles for a given frame-rate. Importantly, each particle is testing a separate random subset of model points. This results in the product of N_s, the number of sample points per particle, and N, the number of particles, reaching a critical level where coverage becomes sufficient that error is equivalent to sampling all points. In the results presented below, we shall demonstrate that this critical level can be used to significantly decrease run time for a given level of error. That is, we shall show that the number of points that must be tested overall, for a given level of error, is lower when stratified sampling is used. This means that we can significantly increase accuracy for a given frame-rate, reducing run-time complexity to the point that we can track 6 DoF pose for multiple objects in real-time.

4. Results

In this Section we first present results on a set of synthetic videos to quantify the effect of the stratified sampling, and compare results to a state of the art GPU particle filter [3]. We then present qualitative results on real videos in a robotic learning application, where we track multiple interacting targets with significant occlusions. In both synthetic and real cases, input consists of RGB-D sequences. Trackers were initialized using an external pose - in the synthetic case, from ground truth, and in the real case, using a pose estimation algorithm [2]. Object models were generated by registering multiple views of the objects using the same RGB-D sensor employed for tracking. All experiments were performed on a standard desktop computer (Intel i7 3.2Ghz), using four cores, and real data was obtained using a Kinect RGB-D camera.

4.1. Synthetic Data

In our first experiment, we demonstrate the effectiveness of our stratified sampling strategy using four synthetic tracking videos from [3]. These RGB-D sequences are set



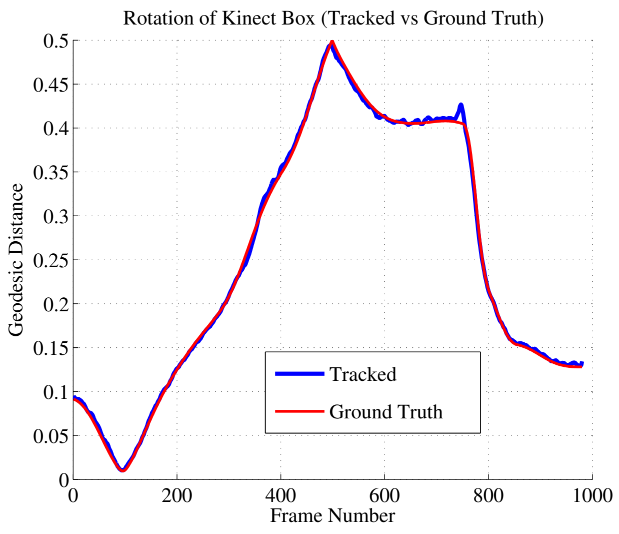


Figure 4. Displacement and rotation ground truth, with an example tracked result from a single run at $N_{samples} = 100$ and $N_{particles} = 1000$ (a frame rate of 20 fps).

in a virtual kitchen (see Figure 3) and each contain a single item to track as the camera moves. Ground truth trajectories of the cameras can be found in [3], but for our purposes it is sufficient to note that the trajectories are complex, consisting of large variations in position, orientation, and velocity.

To evaluate our approach, we compute root mean square (RMS) error in both translation and orientation, averaged over 25 test runs for each sequence. Computation times are measured in ms per frame, and are also averaged across all frames of the 25 test runs. In order to compare with [3], we have combined their RMS error results for each dimension (x, y, z, roll, pitch, yaw) into two measurements - displacement and rotation. Rotation is calculated using the unit quaternion distance metric [11], which is equivalent to the geodesic distance on the unit sphere. This combination reduces the amount of data to compare without loss, as the choice of orientation of the dimensions is arbitrary and without import. Example displacement and geodesic ground truths for the "Kinect Box" sequence can be found in Figure 4.

Timing results are given in Figure 5, showing results for the "Kinect Box" sequence (the most challenging of the four) scanning across number of particles and number of sample points. Plots for the other sequences can be found in the supplementary material and on the author's website. One can observe that, for a given level of sampling, the RMS error decreases for both displacement and geodesic as the number of particles increases. More importantly, it is also apparent that, for a given level of error, run-time per frame can be minimized by reducing the number of samples used and increasing the number of particles. Additionally, one can observe that RMS error appears to be asymptotic, with lower sampling levels approaching the asymptote at lower run-times.

We should also note that the minimum error asymptote observed is likely a consequence of the sampling resolution of the synthetic Kinect camera. For example, in the "Kinect Box" sequence, average distance to neighboring points (8-neighborhood) on the tracked box surface is 3.3 mm. This corresponds almost exactly to our observed error asymptote. This can be observed in all four sequences - our minimal error corresponds closely to the average point to point resolution of the observations on the model.

Our performance compares favorably to the results of Choi and Christensen [3] - for a given level of error, we achieve per-frame run times that are between half and a tenth of their published results. Additionally, we consistently reach the error asymptote, at considerably lower run times. We should also note that the highest sampling level shown corresponds to a complete sampling of the model, and is equivalent to the baseline PCL implementation, although we have made some slight modifications to the resampling and dynamic model which improve results. As can be seen, we are at least an order of magnitude faster than this base implementation.

4.2. Real Sequences

One application of our tracker is to provide semantic understanding and imitation of assembly tasks. This can be accomplished by tracking all interacting parts of an assembly as a human demonstrates, and then using the trajectories and poses in order to train a robot to replicate the construction. Additionally, the tracked output can be used as an input for the robot during construction in order to verify that

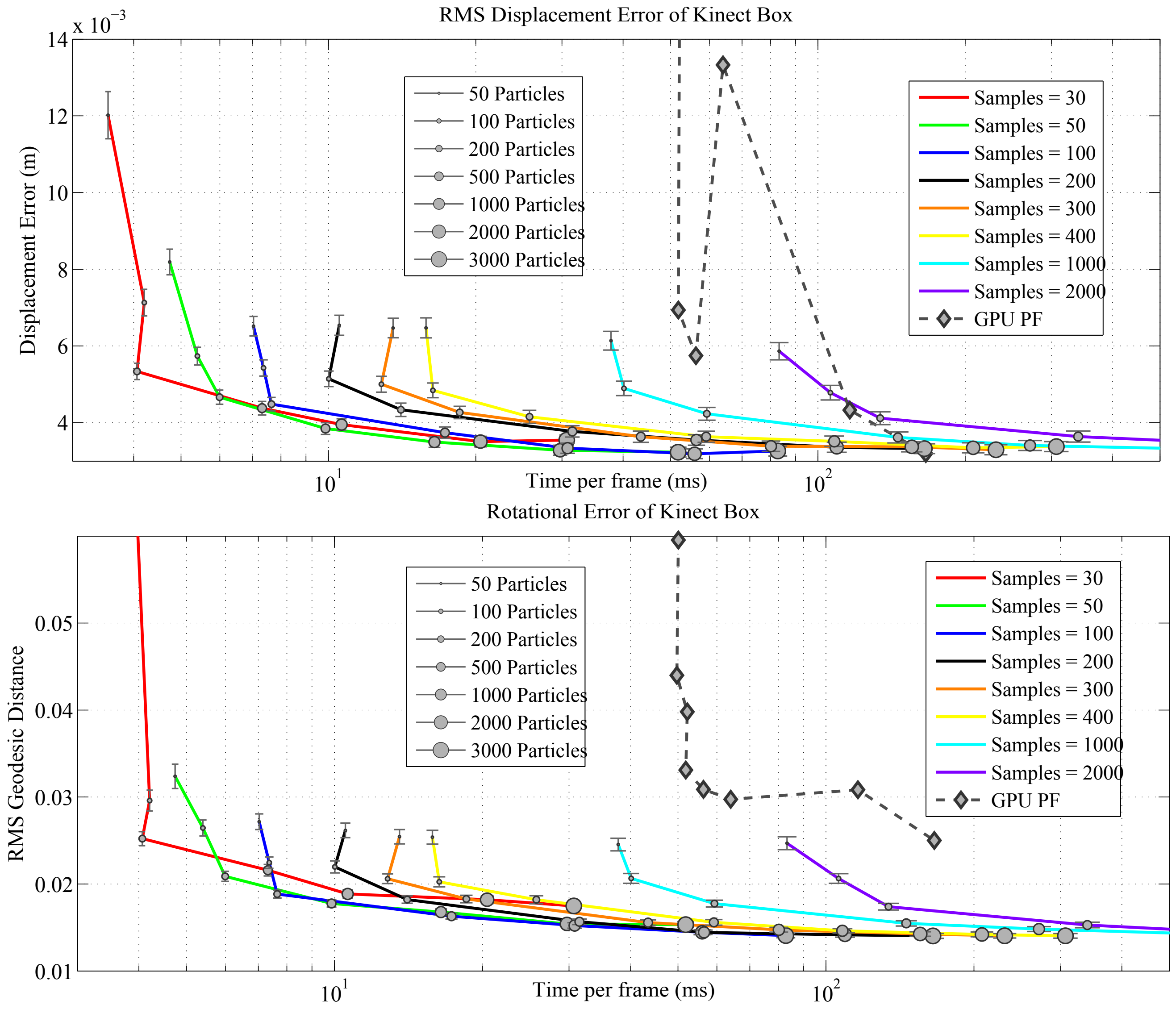


Figure 5. Results on the Kinect Box artificial sequence. Each colored curve represents a certain number of samples, and gives mean RMS error averaged over 25 trial runs for increasing numbers of particles (shown by increasing circle sizes). It is clear that using more samples and using more particles tends to decrease error. The black dotted lines give the results of the GPU method of Choi and Christensen [3].

it has successfully completed each step of the task.

As a demonstration of this, we use the well established "Cranfield" benchmark set [4]. This set consists of eight pieces which can be assembled in a number of different orders. In our experiments, models consist of voxelized point clouds derived from high-resolution models of the pieces, and initial poses for tracking are found using a combined object recognition and pose estimation algorithm [2]. Each object is tracked using an independent particle filter, with $N_{samples}$ set to 50, and $N_{particles}$ set to 1000.

Figure 6 shows a montage of screenshots captured as a human demonstrates assembly of the benchmark. As can be seen, all pieces are successfully tracked from start to finish, with each tracker outputting smooth trajectories that can be used for training a robot using Dynamic Motion Primitives (DMP) [12]. In Figure 7 we show tracks from multiple different human demonstrations - one can observe the different strategies that people employ in assembling the benchmark. The tracks in the lower right corner of the Figure are from a robot reproducing the assembly after being trained on the human demonstrations [16].

5. Conclusion

In this paper we have presented a novel spatially stratified sampling approach which greatly reduces the computational complexity of 3D Point Cloud particle filters. We evaluated the tracker using synthetic sequences for which precise ground truth exists, as well as real sequences of

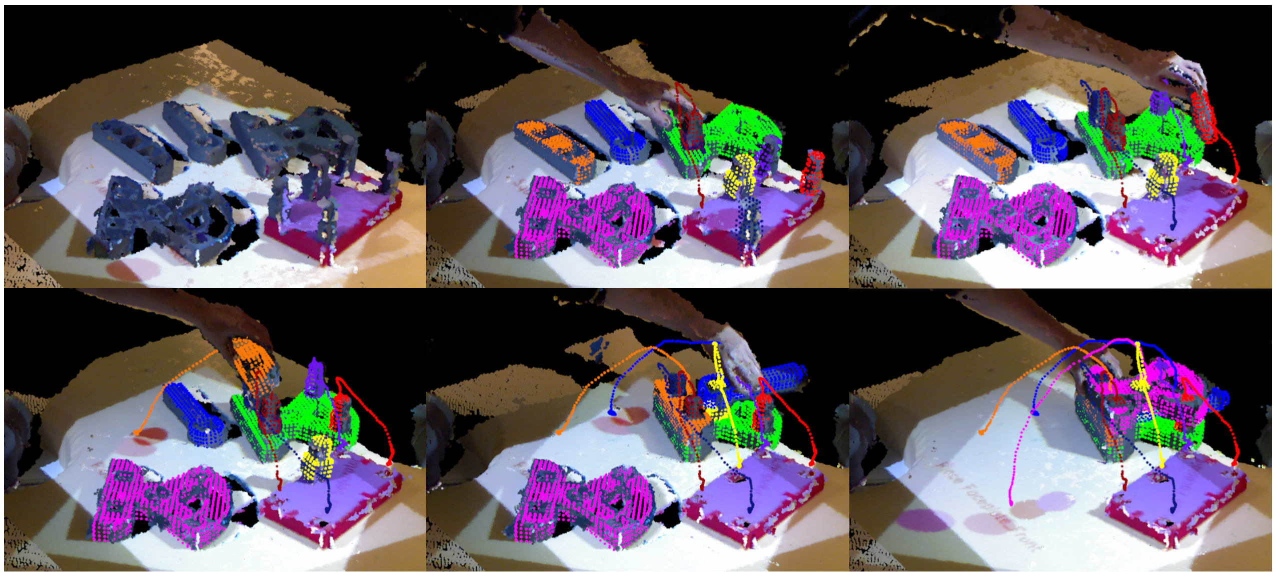


Figure 6. Human demonstration of assembly of the Cranfield Scenario on data from two fused RGB-D cameras. Tracking runs live for all objects at once at sufficient frame rates to track the whole task. Extracted trajectories are shown as traces.

a robot-teaching application. To demonstrate the effect of stratified sampling on performance, we conducted a sweep over the parameter space of number of particles and samples. This sweep showed the clear effectiveness of the proposed method in matching and even out-performing a GPU implementation on the CPU.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 270273, Xperience.

References

[1] M. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool. Robust tracking-by-detection using a detector confidence particle filter. In *Computer Vision, 2009 IEEE 12th International Conference on*, Sept 2009. 1

[2] A. Buch, Y. Yang, N. Krger, and H. Petersen. In search of inliers: 3d correspondence by local and global voting. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, June 2014. 4, 6

[3] C. Choi and H. Christensen. Rgb-d object tracking: A particle filter approach on gpu. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, Nov 2013. 1, 4, 5, 6

[4] K. Collins, A. J. Palmer, and K. Rathmill. The development of a benchmark for the comparison of assembly robot programming systems. In *Robot technology and applications*, 1985. 6

[5] A. Doucet, N. De Freitas, and N. Gordon, editors. *Sequential Monte Carlo methods in practice*. 2001. 2

[6] T. Fortmann, Y. Bar-Shalom, and M. Scheffe. Multi-target tracking using joint probabilistic data association. In *Decision and Control including the Symposium on Adaptive Processes, 1980 19th IEEE Conference on*, Dec 1980. 1

[7] C. Hue, J.-P. Le Cadre, and P. Perez. Tracking multiple objects with particle filtering. *IEEE Transactions on Aerospace and Electronic Systems*, 38(3):791 – 812, jul 2002. 1

[8] M. Isard and A. Blake. Condensation conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998. 1

[9] Z. Khan, T. Balch, and F. Dellaert. Mcmc-based particle filtering for tracking a variable number of interacting targets. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(11):1805–1819, Nov 2005. 1

[10] S. Koo, D. Lee, and D.-S. Kwon. Multiple object tracking using an rgb-d camera by hierarchical spatiotemporal data association. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, Nov 2013. 1

[11] J. Kuffner. Effective sampling and distance metrics for 3d rigid body path planning. In *Robotics and Automation (ICRA) 2004. IEEE International Conference on*, April 2004. 5

[12] T. Kulvicius, K. J. Ning, M. Tamosiunaite, and F. Wörgötter. Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting. *IEEE Trans. Robot.*, 28(1):145–157, 2012. 6

[13] O. Lanz. Approximate bayesian multibody tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(9):1436–1449, Sept 2006. 1

[14] J. Papon, A. Abramov, M. Schoeler, and F. Wörgötter. Voxel cloud connectivity segmentation - supervoxels for point clouds. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, June 2013. 4

[15] J. Papon, T. Kulvicius, E. E. Aksoy, and F. Wörgötter. Point cloud video object segmentation using a persistent su-

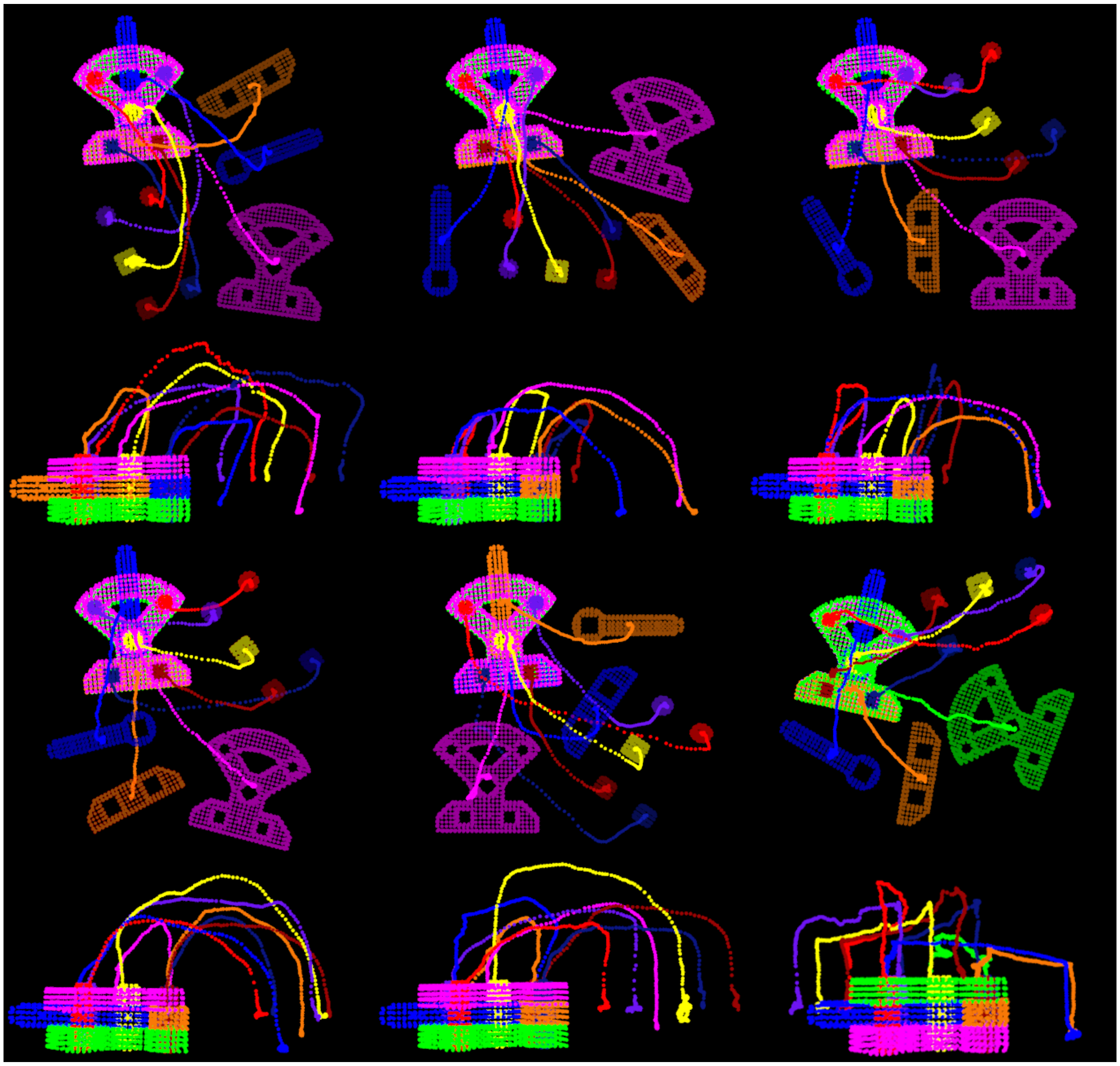


Figure 7. Tracking results from six different recordings of the Cranfield Scenario. The tracks in the bottom right corner are from the robot constructing the object, while the other five are from five different human demonstrators. In the overhead views, starting poses are shown (in slightly darker colors) for the objects.

pervoxel world-model. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, Nov 2013. 3

[16] J. Rossmann, N. Wantia, E. E. Aksoy, S. Haller, and J. Piater. Active learning of manipulation sequences. In *Robotics and Automation (ICRA) 2014. IEEE International Conference on*, 2014. 6

[17] D. Schulz, W. Burgard, D. Fox, and A. Cremers. Tracking multiple moving targets with a mobile robot using particle filters and statistical data association. In *Robotics and Automation (ICRA) 2001. IEEE International Conference on*, 2001. 1

[18] J. Vermaak, A. Doucet, and P. Perez. Maintaining multimodality through mixture tracking. In *Computer Vision 2003. IEEE International Conference on*, Oct 2003. 1

[19] J. Vermaak, S. Godsill, and P. Perez. Monte carlo filtering for multi target tracking and data association. *IEEE Transactions on Aerospace and Electronic Systems*, 41(1):309 – 332, jan. 2005. 1

[20] B.-N. Vo, S. Singh, and A. Doucet. Sequential monte carlo methods for multitarget filtering with random finite sets. *IEEE Transactions on Aerospace and Electronic Systems,*, 41(4):1224 – 1245, oct. 2005. 1, 2