TUM

TECHNISCHE UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK

Adapting the Search Subspace of a Particle Filter using Geometric Constraints

Nikhil Somani, Yaadhav Raaj, Suraj Nair, Alois Knoll

TUM-I1762

Adapting the Search Subspace of a Particle Filter using Geometric Constraints

Nikhil Somani[1,2,4] and Yaadhav Raaj[1,3] and Suraj Nair[1,3] and Alois Knoll[2,4]

***Abstract*— Visual tracking of an object in 3D using its geometrical model is an unsolved classical problem in computer vision. The use of point cloud (RGBD) data for likelihood estimation in the state estimation loop provides improved matching as compared to 2D features. However, point cloud processing is computationally expensive given its big data nature, making the use of mature tracking techniques such as Particle Filters challenging. For practical applications, the filter requires implementation on hardware acceleration platforms such as GPUs or FPGAs.**

In this paper, we introduce a novel approach for object tracking using an adaptive Particle Filter operating on a point cloud based likelihood model. The novelty of the work comes from a geometric constraint detection and solving system which helps reduce the search subspace of the Particle Filter. At every time step, it detects geometric shape constraints and associates it with the object being tracked. Using this information, it defines a new lower-dimensional search subspace for the state that lies in the nullspace of these constraints. It also generates a new set of parameters for the dynamic model of the filter, its particle count and the weights for multi-modal fusion in the likelihood modal. As a consequence, it improves the convergence robustness of the filter while reducing its computational complexity in the form of a reduced particle set.

I. Introduction

Model based visual tracking has been and continues to be an extensively researched area within computer vision. It finds application in several domains within robotics and automation. The problem has been explored both in $2D$ and $3D$ w.r.t. the sensor used and the dimension of the estimated state space. It continues to be a challenging problem given the nonlinearity and noise seen in the available vision sensors. A comprehensive survey of object tracking methods and systems mainly using RGB cameras has been covered in [1]. The recent availability of low cots RGBD sensors have made it possible to explore model based tracking using point cloud data.

In visual tracking, typically the state space approach is adopted to model the position, velocity and acceleration of the object to be tracked. Statistical correspondence methods are used to estimate the object's state by collecting sensor measurements, where model uncertainties and sensor noise is also modeled. The Kalman Filter and Particle Filter are the two most popular methods in model based tracking [2]. The Kalman filter assumes the state space to be distributed by a Gaussian and that the update model has a linear relation with the measurement model. If the state update and measurement models are non linear, the Extended Kalman filter [3] is used where a linear approximation of the same are obtained using the Taylor series expansion. The assumption of a Gaussian state space distribution limits the application of a Kalman filter. The Particle Filter overcomes this limitation by representing the conditional state density using a set of samples called particles, where each particle contains a weight obtained from the measurement (likelihood). Particle Filters are observed to be robust under multi-modal likelihoods due to clutter in the tracking scene and thereby are much better in avoiding local minima as compared to Kalman filters [4], [5]. However, the computation cost associated with Particle Filters increases exponentially with the dimensionality of the modeled state space and the complexity of the measurement model. For tracking using point cloud data, the measurement models are inherently computation and memory intensive. Therefore, known applications of Particle Filter for point cloud based object tracking is rather limited [6]. Nevertheless, particles filter operating on point cloud data and color (RGBD) information serve as a powerful tool for model based object tracking.

Fig. 1: Constrained motion of a manipulated object: the soda can lies on a tray that is being moved by a robot.

For our approach, an important assumption that we make about object models is that some parts of their geometries can be approximated using a set of primitive shapes such as planes, cylinders, spheres, etc. For object recognition and pose estimation, these primitives prove to be very useful since they can be detected more accurately and efficiently than complete CAD models.

Whenever an object interacts with its environment or a manipulator, its motion and dynamics are affected. With information about the primitive shapes in the environment, these restrictions can be modeled as geometric constraints between primitive shapes (see Fig. 1).

Effective exploration and exploitation of primitive shape constraints can lead to considerable reduction in the com-

[1] TUM CREATE, Singapore
[2] Technische Universität München, Germany
[3] raaj.yaadhav, suraj.nair@tum-create.edu.sg
[4] somani, knoll@in.tum.de

putational complexity of Particle Filters used for model based object tracking using RGBD sensors. These constraints originate from the following sources:

- geometric properties intrinsic to the object model
- constraints imposed by the environment on the object on contact and interaction
- constraints imposed by a manipulator, influencing the dynamics of the object
- partial views based on sensor viewpoint

While the object interacts with the environment, real-time detection of the above mentioned primitive shape constraints can enable simplification of the object dynamics and thereby reduce the state space dimensions of the Particle Filter's sample set. Furthermore, the number of particles required to estimate the state can be reduced given the dimension reductions. As a consequence the computational effort required by the Particle Filter can be considerable reduced. This paper demonstrates a novel method for achieving the above mentioned concepts and a systemic implementation with evaluations in simulation and a real-world setup. The methodology demonstrates the feasibility of using Particle Filters with RGBD sensors in real world applications.

The rest of the paper is organized as follows: Section II presents the state of the art in model based object tracking followed by Section III which presents the novel concept of tracking with primitive constraints. In Section IV the adaptive Particle Filter using primitive shape constraints is introduced. Finally, experiments and evaluations are presented in Section V followed by conclusion and future work in Section VI.

II. Related Work

Model based object tracking in general finds several citing in computer vision literature. Several algorithms and systems have been developed over the years. The work presented in [7] provides a unifying framework for model based visual tracking in the form of a library of algorithms for 2D and 3D tracking, where object tracking using a variety of sensors and visual features can be explored. Similarly, [8] developed a framework for automatic modeling, detection, and tracking of 3D objects using RGBD data from sensors such as a Kinect. They employ a multi-modal template based approach presented in [9]. In [10] the authors present a GPU based framework for accelerating Particle Filter based approaches applied to 3D model-based visual tracking.

A special case of 3D visual tracking is tracking of human hands in several degrees of freedom (3-26). Multiple approaches in terms of visual features, tracking methods and data fusion have been explored for handling this high dimensional problem in [11], [12], [13] and [14].

In terms of Particle Filter based tracking methods, several approaches have been documented in literature [15]. [16] presented a Bayesian framework for 3D tracking of deformable objects by exploring multiple-cues. The authors propose a spatio-temporal model of the object using Dynamic Point Distribution Models (DPDMs) in order to improve robustness towards appearance changes. The state estimation is performed using a Particle Filter using three cues (shape, texture and depth) in the likelihood model. In terms of adaptive particle filters, the KLD tracker [17] is an approach for adapting the sample set based on the covariance values. In our approach, we adapt both the sample size and the tracking subspace based on the primitive shape matching constraints.

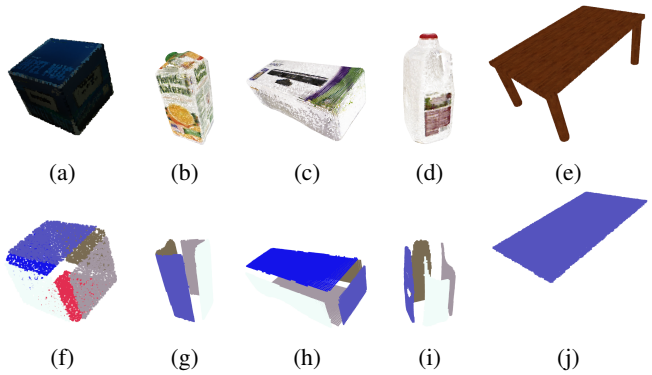


Fig. 2: RGBD Object models (a,b,c,d,i) and corresponding primitive shape decompositions (e,f,g,h,j)

Many of the above tracking papers above were also single camera approaches, allowing nearest neighbour operations to be fast by exploiting the organized nature of 3D data, e.g. in [10]. To handle complete occlusion of objects multiple sensors would be required. Multi-camera tracking approaches were also studied in [18] [19], but these approaches were found to either use individual RGB cameras directly during likelihood estimation, or extracted primitives directly from a fused point cloud scene for comparison.

The core contribution of this paper is a novel approach for object tracking using multiple RGBD sensors through an adaptive Particle Filter. The adaptive nature of the filter is modeled on the use of a primitive shape detection algorithm which exploits the detected geometric constraints connected to the object during the tracking process and thereby exploits these constraints to reduce the search subspace of the Particle Filter increasing its convergence robustness and also reduces its computational complexity. Our approach can also use multiple sensors seamlessly, and works on unorganized point clouds via our fast GPU implementation of the indexed octree library described in [20].

III. Primitive shape constraints for tracking

This section describes our adaptive tracking approach with primitive shape constraints. The overall steps are described in Algorithm 1, and their individual details are described in the following subsections.

A. Object models based on primitive shapes

We use a boundary representation (BREP) [21] for all geometric entities in our system. This representation composes objects from semantically meaningful primitive shapes (e.g., planes, cylinders). Fig. 2 illustrates such Primitive Shape Decompositions for some of the objects used in this work. This object model is motivated from the observation that the estimation of primitive shapes such as planes and

Algorithm 1: Algorithm for tracking with primitive shape constraints (TPSC)

Input: Primitive Shape model $\mathcal{P}_{model}$, last object pose s_{t-1}, point cloud I_t, particle set $\mathcal{S}_{t-1}$, default Guassian noise covariance $\boldsymbol{w}^{def}$

Output: Tracked object pose s_t, particle set $\mathcal{S}_t$

$\mathcal{P}_{scene} \leftarrow$ detectPrimitives(I_t)

$\mathcal{C} \leftarrow$ detectConstraints($\mathcal{P}_{model}, \mathcal{P}_{scene}, s_{-1}$)

$\mathcal{C}_{comb} \leftarrow$ combineConstraints($\mathcal{C}$)

foreach $S^i = (s^i, w^i)_{t-1} \in \mathcal{S}_{t-1}$ **do**

$\hat{s}_t^i \leftarrow \mathrm{f}_{\mathrm{proj}}(\boldsymbol{s}_{t-1}^i)$ (see Table II)

$\hat{w}_t^i \leftarrow [(\boldsymbol{N_t} \cdot \boldsymbol{w}_t^{def})\boldsymbol{N_t} + (\boldsymbol{S} \cdot \boldsymbol{\sigma_t})\boldsymbol{S_t},\ (\boldsymbol{N}_R \cdot \boldsymbol{w}_R^{def})\boldsymbol{N}_R + (\boldsymbol{S}_R \cdot \boldsymbol{\sigma}_R])\boldsymbol{S}_R]$ (Table II)

end

$\mathcal{L} \leftarrow$ computeLikelihood($\hat{\mathcal{S}}_t$)

$\mathcal{S}_t \leftarrow$ resample($\hat{\mathcal{S}}_t, \mathcal{L}$)

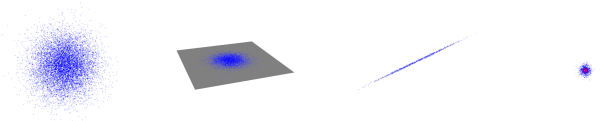
$s_{\mathrm{t}} \leftarrow mean(\mathcal{S}_t)$

cylinders in a point cloud is faster and more robust than detection of the full object models. Based on this simplified model, the object detection and pose estimation problem can be considered as a primitive shape matching problem [22] between the set of model and scene primitive shapes ($\mathcal{P}_m$ and $\mathcal{P}_s$ respectively). In the context of object pose estimation, the model primitive can be considered as a constrained shape, the scene primitive as a fixed shape, and the shape match as a constraint $C_i = \mathrm{Coincident}(\mathcal{P}_s^i, \mathcal{P}_m^i)$. A geometric constraint adds restrictions to the relative pose of the constrained shape w.r.t. the fixed shape.

Each primitive shape $\mathcal{P}_i \in \mathcal{P}$ enforces a set of constraints $C_i = (\boldsymbol{C}_{\boldsymbol{p}_i}, \boldsymbol{C}_{\boldsymbol{n}_i})$ on the position and orientation of the object respectively. Each row of $\boldsymbol{C}_{\boldsymbol{p}_i}$ and $\boldsymbol{C}_{\boldsymbol{n}_i}$ contains a direction along which the constraint has been set. Table II defines these nullspaces for the geometric constraints relevant to this work.

By analyzing the set of primitive shapes that form an object, properties such as symmetry can be determined. Each primitive shape $\mathcal{P}_i \in \mathcal{P}$ in the object is used to create a constraint $C_i = \mathrm{Coincident}(\mathcal{P}_i, \mathcal{P}_i)$. The set of constraints $\mathcal{C} = \bigcup_i C_i$ is then used by a constraint solver to determine a transformation manifold. These transformation manifolds are geometric entities, and provide projection functions. Hence, any pose hypothesis for the object can be projected onto this transformation manifold. The DoFs of this transformation manifold determine the symmetry axes of the object. This information about the symmetry axes and the projection functions from the transformation manifold can be exploited to improve the tracking, as shown in Section IV.

Given a partial view of an object, it might not be possible for a detector to determine its complete pose. This is especially the case for detectors based on primitive shapes, where fine features that form a small fraction of the object are often ignored in simplified models. Given the set of primitive shapes visible from a specific viewpoint, the DoF analysis can be used to determine the axes along which the pose



(a) (b) (c) (d)

Fig. 3: Projection of particles to the nullspace of geometric constraints (a) 3D particle set (b) Particles projected onto a plane (c) Particles projected onto a line (d) Particles projected onto a point

can't be determined. This information can be utilized by the tracker to sample more particles in these DoFs. Since the tracker uses a motion model and maintains a history of the previous pose, it might be able to predict the correct pose of the object in from this partial view.

B. Geometric model of the environment constraints

We exploit knowledge about the environment and its effect on the object's motion, by modeling the environment using the same primitive shapes models as described in Section III-A. Geometric constraints can then be defined between primitive shapes in the environment ($\mathcal{P}_e$) and the object ($\mathcal{P}_m$). As an example, for an object that only moves on a tabletop, a Coincident constraint can be defined between a plane on the object's bottom surface and the plane of the table. This restricts the translation of the object along the normal direction of the tabletop plane, and orientations along the other two orthogonal axes.

C. Constraint solver

We choose the exact geometric solver by [23] for this application. This provides several advantages over iterative approaches. Most importantly, the transformation manifolds and projection functions need to be calculated once for each frame of the scene point cloud and can be re-used for all the particles. Also, the exact solver is much faster than iterative approaches. For our constraints, the exact solver can provide runtimes of approximately 50μs.

IV. Adaptive Particle Filter

The Particle Filter based tracker holds a state-space representation of the 3D object. The pose is modeled in 6D (position and orientation) and represented as

$$\boldsymbol{s}_t = (x_t, y_t, z_t, \alpha_t, \beta_t, \gamma_t) \tag{1}$$

A. Projection sampling

The Particle Filter incorporates a Brownian motion model for the state prediction. As compared to a normal particle filter, in our method the Filter's dynamic model is further influenced by the projections $\mathrm{f}_{\mathrm{proj}}(\boldsymbol{s})$ obtained form the geometric constraints detection system, as discussed in Sec III and Algorith 1. Fig. 3 shows an illustration of the projection operations. In case of one plane match, the translation nullspace is a plane, and all points in the particle set s_t are projected onto the plane (see Fig. 3(b)). For two plane

TABLE I: Projection functions for supported geometric constraints

Fixed	**Constrained**	**Constraint** (C)	**Controlled Space** (S)	**Null Space** (N)
$\mathrm{Pl}_1(\boldsymbol{p}_1, \hat{\boldsymbol{n}}_1)$	$\mathrm{Pl}_2(\boldsymbol{p}_2, \hat{\boldsymbol{n}}_2)$	Dist. (d)	$\boldsymbol{S_R}$: $[\hat{\boldsymbol{n}}_{1_{\perp 1}}; \hat{\boldsymbol{n}}_{1_{\perp 2}}]$ $\boldsymbol{S_t}$: $[\hat{\boldsymbol{n}}_1]$	$\boldsymbol{N_R}$: $[\hat{\boldsymbol{n}}_1]$ $\boldsymbol{N_t}$: $[\hat{\boldsymbol{n}}_{1_{\perp_1}}; \hat{\boldsymbol{n}}_{1_{\perp_2}}]$
$\mathrm{Pl}_1(\boldsymbol{p}_1, \hat{\boldsymbol{n}}_1)$	$\mathrm{Pl}_2(\boldsymbol{p}_2, \hat{\boldsymbol{n}}_2)$	Coinc.	$\boldsymbol{S_R}$: $[\hat{\boldsymbol{n}}_{1_{\perp 1}}; \hat{\boldsymbol{n}}_{1_{\perp 2}}]$ $\boldsymbol{S_t}$: []	$\boldsymbol{N_R}$: $[\hat{\boldsymbol{n}}_1]$ $\boldsymbol{N_t}$: [1]
$\mathrm{L}_1(\boldsymbol{p}_1, \hat{\boldsymbol{a}}_1)$	$\mathrm{L}_2(\boldsymbol{p}_2, \hat{\boldsymbol{a}}_2)$	Coinc.	$\boldsymbol{S_R}$: $[\hat{\boldsymbol{a}}_{1_{\perp 1}}; \hat{\boldsymbol{a}}_{1_{\perp 2}}]$ $\boldsymbol{S_t}$: $[\hat{\boldsymbol{a}}_{1_{\perp 1}}; \hat{\boldsymbol{a}}_{1_{\perp 2}}]$	$\boldsymbol{N_R}$: $[\hat{\boldsymbol{a}}_1]$ $\boldsymbol{N_t}$: $[\hat{\boldsymbol{a}}_1]$
$\mathrm{Pt}_1(\boldsymbol{p}_1)$	$\mathrm{Pt}_2(\boldsymbol{p}_2)$	Coinc.	$\boldsymbol{S_R}$: [] $\boldsymbol{S_t}$: 1	$\boldsymbol{N_R}$: [1] $\boldsymbol{N_t}$: []
$\mathrm{L}_1(\boldsymbol{p}_1, \hat{\boldsymbol{a}}_1)$	$\mathrm{L}_2(\boldsymbol{p}_2, \hat{\boldsymbol{a}}_2)$	Parallel	$\boldsymbol{S_R}$: $[\hat{\boldsymbol{a}}_{1_{\perp 1}}; \hat{\boldsymbol{a}}_{1_{\perp 2}}]$ $\boldsymbol{S_t}$: $[\hat{\boldsymbol{a}}_{1_{\perp 1}}; \hat{\boldsymbol{a}}_{1_{\perp 2}}]$	$\boldsymbol{N_R}$: $[\hat{\boldsymbol{a}}_1]$ $\boldsymbol{N_t}$: $[\hat{\boldsymbol{a}}_1]$

[1] Pl = Plane, L = Line, Pt = Point, Coinc = Coincident, Dist = Distance, $\boldsymbol{p}$ = Point, $\boldsymbol{n}$ = Normal direction, $\boldsymbol{a}$ = Axis

matches, the constraint is a Line-Line Coincident constraint between the lines of intersection of the two sets of matched model and scene planes. The translation nullspace is a line, and all points in the particle set s_t are projected onto it (see Fig. 3(c)). For three plane matches, the constraint is a Point-Point Coincident constraint between the point of intersection of the two sets of matched model and scene planes. The translation nullspace is a point, and all points in the particle set s_t are projected onto it (see Fig. 3(c)).

During the prediction phase several prior state hypothesis s_t^i are generated by the Particle Filter to form the particle (sample) set using the previous particle distribution $(s^i, w^i)_{t-1}$. The motion model used is Brownian in nature: (2) where, w is the white Gaussian noise with defined covariance in the s_t state variables and K is the number of particles.

$$\boldsymbol{s}_t^i = \mathrm{f}_{\mathrm{proj}}(\boldsymbol{s})(\boldsymbol{s}_{t-1}^i) + \boldsymbol{w}_t^i, \quad i = 1, 2, \ldots, K \tag{2}$$

B. Covariance adaptation

In the re-sampling step, the particle filter uses the estimated Gaussian noise covariance value to add random noise to the sampled particles. This Brownian model is an important feature of the particle filter approach and enables it to adapt to measurement noises. This value should closely correspond to the actual random noise in the measurement, e.g, the inherent accuracy and precision of the sensor. This value also depends on the sensor data processing that might introduce delays and therefore additional uncertainty in the actual measurement.

In our adaptive particle filter, we exploit the information about primitive shapes detected in the scene to adapt this noise estimate. The primitive shape detection algorithm provides estimated noise in the detection process (e.g. average fitting error for points on a plane). The default noise covariance vector $\boldsymbol{w}^{def}$ represents the estimated noise in the raw sensor data. For the combined primitive shape constraint used for tracking C_{comb}, the estimated noise in position and orientation is $\boldsymbol{\sigma}_R$ and $\boldsymbol{\sigma}_t$ respectively. A constraint specifies the directions it constrains (Controlled Space Directions) and the directions where it has no effect (Nullspace directions), as listed in Table II. In the directions controlled by the constraint (Controlled Space Directions) the estimated Gaussian noise covariance corresponds to $\boldsymbol{\sigma}_R$ and $\boldsymbol{\sigma}_t$, while in the remaining directions (Nullspace directions) the estimated Gaussian noise covariance is $\boldsymbol{w}^{def}$. (3) combines these to generate the adapted covariance vector $\hat{\boldsymbol{w}}$.

$$\hat{\boldsymbol{w}} \leftarrow [(\boldsymbol{N_t} \cdot \boldsymbol{w}_t^{def})\boldsymbol{N_t} + (\boldsymbol{S} \cdot \boldsymbol{\sigma_t})\boldsymbol{S_t}, (\boldsymbol{N_R} \cdot \boldsymbol{w}_R^{def})\boldsymbol{N_R} + (\boldsymbol{S_R} \cdot \boldsymbol{\sigma}_R])\boldsymbol{S_R}] \tag{3}$$

C. Hypothesis Verification

For every particle hypothesis the object model is projected into the point cloud scene I and a likelihood is computed w.r.t. a distance matching metric. It requires evaluation of a distance measure D_p between the projected point cloud of the object model and the corresponding nearest neighbor points in the measurement P_t. This is followed by the computation of the distance measure D_c on the color separation of the two point sets.

$$D_p = \frac{1}{N} \sum_{n=1}^{N} \frac{dist(\boldsymbol{m}_{xyz}^n, \boldsymbol{I}_{xyz}^n)(1 - \boldsymbol{m}_{normal}^n \cdot \boldsymbol{I}_{normal}^n))}{r} \tag{4}$$

(4) presents the first metric where, N is the number of projected points based on the camera perspective (only points visible to the camera view are considered by analyzing the normals), $dist(\boldsymbol{m}, \boldsymbol{I})$ refers to the distance between a model point $\boldsymbol{m}$ and the approximated nearest neighbor point in the scene within a search radius r. The computed distance is further biased by the angle between the normal directions of the model point and the matched scene point. This is obtained using the dot product between the two normals wherein a better match reduces the overall distance measure. (4) represents the metric.

$$\begin{aligned} \boldsymbol{q}_m &= f_{histo2D}(m_{hsv}) \\ \boldsymbol{q}_I &= f_{histo2D}(I_{hsv}) \\ D_c &= \left[1 - \sum_{n=1}^{N} \sqrt{\boldsymbol{q}_m(n)\, \boldsymbol{q}_I(n)}\right]^{1/2} \end{aligned} \tag{5}$$

TABLE II: Nullspace definitions for supported geometric constraints

Fixed	**Constrained**	**Constraint** (C)	**Projection Functions** ($\mathrm{f}_{\mathrm{proj}}(\boldsymbol{s})$)	
			Translation ($\mathrm{proj}_p(\boldsymbol{u})$)	**Rotation** ($\mathrm{proj}_r(\boldsymbol{R}_i)$)
$\mathrm{Pl}_1(\boldsymbol{p}_1, \hat{\boldsymbol{n}}_1)$	$\mathrm{Pl}_2(\boldsymbol{p}_2, \hat{\boldsymbol{n}}_2)$	Dist. (d)	$\boldsymbol{u} + ((\boldsymbol{p}_1 - \boldsymbol{u}) \cdot \hat{\boldsymbol{n}}_1)\hat{\boldsymbol{n}}_1 + d\hat{\boldsymbol{n}}_1$	$\boldsymbol{R}_{AA}(\hat{\boldsymbol{n}}_1 x \hat{\boldsymbol{n}}_2, \angle(\hat{\boldsymbol{n}}_1, \boldsymbol{R}_i\hat{\boldsymbol{n}}_2))\boldsymbol{R}_i$
$\mathrm{Pl}_1(\boldsymbol{p}_1, \hat{\boldsymbol{n}}_1)$	$\mathrm{Pl}_2(\boldsymbol{p}_2, \hat{\boldsymbol{n}}_2)$	Coinc.	$\boldsymbol{u} + ((\boldsymbol{p}_1 - \boldsymbol{u}) \cdot \hat{\boldsymbol{n}}_1)\hat{\boldsymbol{n}}_1$	$\boldsymbol{R}_{AA}(\hat{\boldsymbol{n}}_1 x \hat{\boldsymbol{n}}_2, \angle(\hat{\boldsymbol{n}}_1, \boldsymbol{R}_i\hat{\boldsymbol{n}}_2))\boldsymbol{R}_i$
$\mathrm{L}_1(\boldsymbol{p}_1, \hat{\boldsymbol{a}}_1)$	$\mathrm{L}_2(\boldsymbol{p}_2, \hat{\boldsymbol{a}}_2)$	Coinc.	$\boldsymbol{p}_1 + ((\boldsymbol{u} - \boldsymbol{q}) \cdot \hat{\hat{\boldsymbol{a}}})\hat{\hat{\boldsymbol{a}}}$	$\boldsymbol{R}_{AA}(\hat{\boldsymbol{a}}_1 x \hat{\boldsymbol{a}}_2, \angle(\hat{\boldsymbol{a}}_1, \boldsymbol{R}_i\hat{\boldsymbol{a}}_2))\boldsymbol{R}_i$
$\mathrm{Pt}_1(\boldsymbol{p}_1)$	$\mathrm{Pt}_2(\boldsymbol{p}_2)$	Coinc.	$\boldsymbol{p}_1$	$\boldsymbol{R}_i$
$\mathrm{L}_1(\boldsymbol{p}_1, \hat{\boldsymbol{a}}_1)$	$\mathrm{L}_2(\boldsymbol{p}_2, \hat{\boldsymbol{a}}_2)$	Parallel	$\boldsymbol{u}$	$\boldsymbol{R}_{AA}(\hat{\boldsymbol{a}}_1 x \hat{\boldsymbol{a}}_2, \angle(\hat{\boldsymbol{a}}_1, \boldsymbol{R}_i\hat{\boldsymbol{a}}_2))\boldsymbol{R}_i$

[1] Pl = Plane, L = Line, Pt = Point, Coinc = Coincident, Dist = Distance, $\boldsymbol{p}$ = Point, $\boldsymbol{n}$ = Normal direction, $\boldsymbol{a}$ = Axis, $\boldsymbol{u}$ = Input Point, $\boldsymbol{R}_i$ = Input rotation matrix, $\boldsymbol{R}_{AA}(\boldsymbol{a}, \theta)$ = Rotation matrix defined by Axis $\boldsymbol{a}$ and angle θ

(5) shows the colour distance measure which is obtained by generating the joint probability histograms [24] of the projected model points and the matched scene points obtained from the nearest neighbor search as a part of the point matching distance measure. The RGB values of the points are first converted to a HSV color space and thereafter the the joint probability histograms q_m and q_I are computed for the projected model and scene points respectively. In the next step a distance between the two histograms is computed using the Bhattacharya distance metric [25]. The two distance measures are fused in a likelihood function under a Gaussian model wherein a weighted sum of the the two distance measures is applied. A parameters μ_p and μ_c define the fusion weight of the point and color matching metrics respectively such that $\mu_p + \mu_c = 1$. (6) presents the likelihood model where D_f is the fused distance measure and λ is the Gaussian likelihood model covariance.

$$P(z|s_t^i) = exp(-\sum_{p=1}^{K}(D_f^2/\lambda)$$
$$D_f = \mu_p D_p + \mu_c D_c \tag{6}$$

Based on the computed likelihood, the weight w_i of each particles is updated. Based on the normalized weights distribution a importance based re-sampling strategy [26] is applied to the particle set.

D. Complexity

Applying the Particle Filter for tracking objects in the scheme mentioned above is challenging in terms of computational complexity arising from a very high number of particles required to track the object in 6D. The point cloud based likelihood models themselves are computationally heavy give their big data nature. Furthermore, sensor noise, scene clutter and free dynamics of the object effects the stability of the tracker resulting in tracking loss. In our work we precisely address this problem of Particle Filters through geometric constraints. We define a finite parameter set which influences the sub search space of the Particle Filter. It is gives as follows:

$$PF_{params} = \begin{Bmatrix} K, & particle\ count\ K > 0 \\ w, & process\ noise\ covariance \\ \mu_c, & colour\ modality\ weight \\ & 0 \leq \mu_c \leq 1 \end{Bmatrix} \tag{7}$$

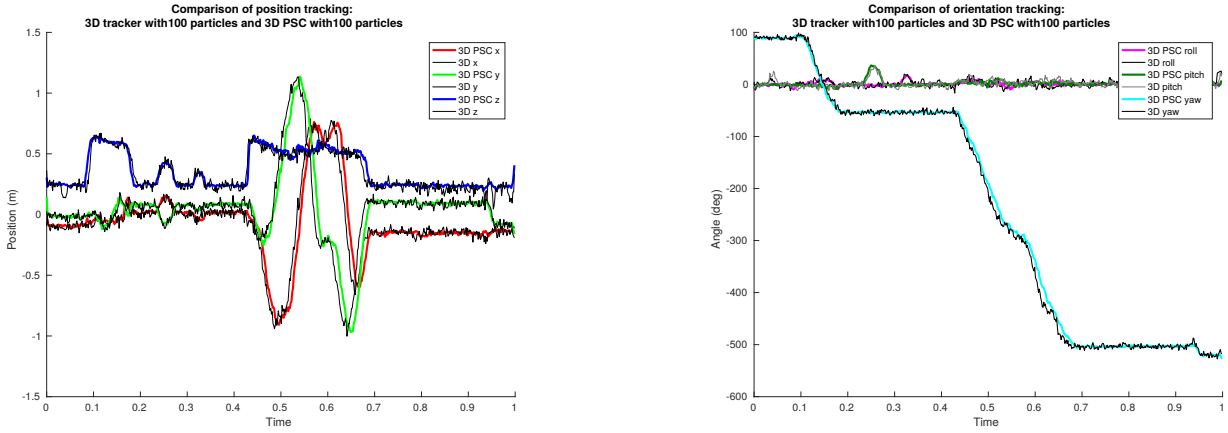
On startup, the Particle Filter is initialized using a rough pose of the object using a standalone detector. This information is sufficient to obtain a initial estimate of the object's state. However, to maintain robust track of the object which exhibits free dynamics, the parameter set represented in (7) requires one-line adaptation in real-time. In order to achieve this the companion geometric constraint detection system operates in close synchronization with the Particle Filter. At each time step, the constraint detection system gets the current estimated pose of the target from the Particle Filter. Based on the association between the target pose and the geometric constraints detected (plane, line or point) a new set of parameters are generated for the Particle Filter. This process essentially locks one or more degrees of freedom of the update model of the Particle Filter which is reflected in the new covariance set for the dynamic update model. In addition the reduction in the state space dimension allows reduction in the particle count resulting in reduced computational complexity of the filter. Furthermore, it is observed that the color modality allows convergence of the Particle Filter when the filter initialization is considerably away from the ground truth or under tracking loss scenarios. In such circumstances the influence of the color modality can be enhanced using the μ_c parameter. Fig. 7 compares the performance of these different tracker modes in cases of tracking loss.

V. Experiments and Evaluations

We conducted several experiments, both in simulated scenes and on real data from a Kinect v2 sensor. Through these experiments, which are described in detail in the following subsections, we show how objects can be accurately tracked by our system with very few particles.

A. Experimental Setup

Our live experimental setup (see Fig. 4) consists of 3 NUC/Kinect units ($C1, C2, C3$) with a tracking space of approximately (3m x 3m x 2.5m). RGB and Registered Depth



(a) Trajectory for translation

(b) Trajectory for orientation

Fig. 5: Comparison of trajectories for object WoodBox using our live experimental setup

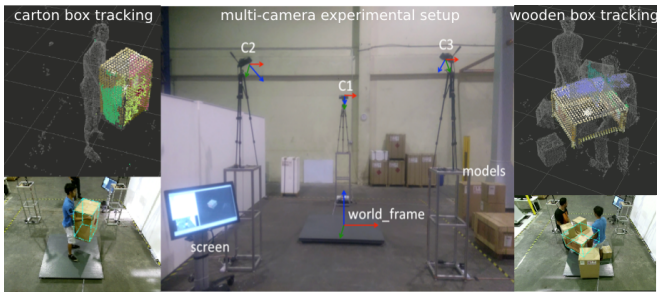


Fig. 4: Experimental setup with illustrations of some live tracking scenarios

images are extracted via the libfreenect2 and iai-kinect2 [27] drivers and sent via the ROS (Robot Operating System) interface to a central computer. The 3 cameras are extrinsically calibrated to a central WorldFrame. We then create a fused streaming point cloud by generating, transforming and fusing the points together into the same frame on the GPU. A voxel grid filter followed by polynomial fitting [28] is used to filter out noise, and finally generate data $(r, g, b, x, y, z, n_x, n_y, n_z)$ at approx. 10Hz. We run our experiments on a Intel 6700K CPU and a Geforce GTX 1070.

B. Evaluation of TPSC

For evaluation, we used the set of models available from [6]. The objects and their simplified models based on primitive shape are shown in Fig 2.

We quantitatively examined several aspects of our tracking approach. Firstly, we evaluate the effect of color information (controlled by parameter μ_c) on the tracking performance. Secondly, we determine the benefits of using primitive shape constraints. For each object in the evaluation, we used 4 different tracking approaches: Only 3D (i.e., using only 3D point cloud information D_p), color+3D (i.e., $\mu_p D_p + \mu_c D_c$), 3D+PSC (D_p with primitive shape constraints), and the full TPSC (i.e. $\mu_p D_p + \mu_c D_c$ with primitive shape constraints).

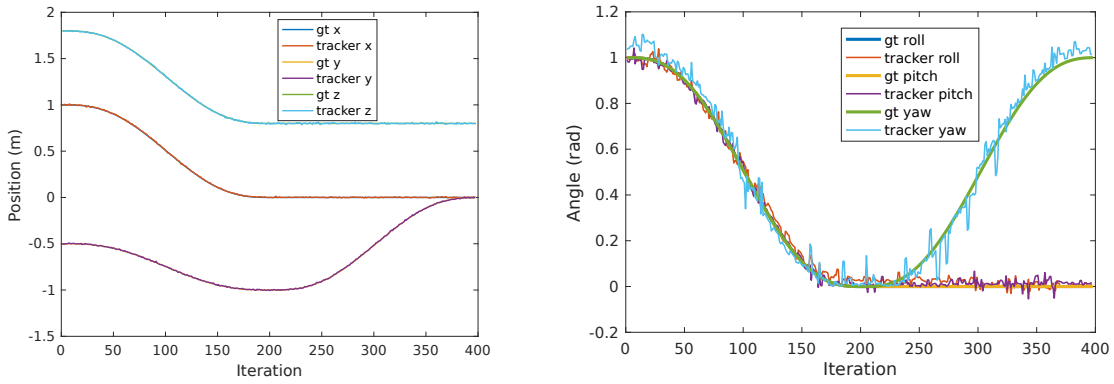
For the tracking scenario, we used a tabletop scene with the object. In the first segment of the ground truth trajectory, the object is moved in all 6DoFs until it reaches the table. In the second segment, the object moves along the table with 3DoFs (translation along x and y axes, and rotation along z-axis). The results showing the errors in each axis for all the object and tracker combinations are summarized in Table III. Tracking trajectories in rotation and translation for the object JuiceBox using the full TPSC tracker over the simulated dataset are shown in Fig. 6. Fig. 7 compares the evolution of tracking errors from an initial position for the 4 different trackers.It is clear that the full TPSC tracker shows the best convergence properties.

From the evaluation, and a comparison with the results in [6], it is clear that our approach can achieve reasonably accurate tracking by use of very few particles. In most cases, use of color information improves the tracking, but there are a few outliers. This is more prominent for the Box, and the probable reason is a low-resolution color model and similar textures on all faces of the box. In particular, it can be observed that tracking with use of primitive shapes always improves the tracking accuracy, and requires lesser particles.

C. Application in logistics domain

We use some examples from the logistics domain to evaluate our tracking approach. This domain is very relevant for our approach since a large fraction of cargo items are comprised of simple shapes such as cuboids and cylinders. We show a tracking scenario with 3 kinect V2 sensors, where some cargo items are tracked (see Fig. 8).

We use simulated object trajectories to generate a synthetic dataset and use this to compare two trackers under different settings. Our baseline tracker ("Only 3D") is a simple particle filter that doesn't use the primitive shape information. The second tracker is our primitive shape based tracking approach ("3D+PSC") described in Section IV. We quantitatively evaluate several properties of the trackers. We study the



(a) Trajectory for translation

(b) Trajectory for orientation

Fig. 6: Tracking trajectories for object JuiceBox using TPSC

TABLE III: Evaluation of TPSC

Object	**Tracker**	**#Particles** (min. – avg. – max.)	**RMS Errors (mm)** X	Y	Z	**RMS Errors (deg)** Roll	Pitch	Yaw
Box	Only 3D	500	10.2	8.7	7.9	7.7087	6.1825	11.6113
	color+3D	500	10.0	9.8	7.6	6.0418	6.0899	10.3905
	3D+PSC	150 – 255.7 – 500	10.6	3.8	**3.3**	**1.7763**	1.8372	**2.5001**
	color+3D+PSC	150 – 255.1 – 500	**8.7**	**3.3**	5.0	1.7898	**1.7909**	3.1615
Juice	Only 3D	500	15.4	11.8	6.8	7.9867	7.1782	9.8036
	color+3D	500	13.4	8.3	6.2	8.0417	8.0357	8.4454
	3D+PSC	150 – 335.2 – 500	12.0	**3.7**	6.1	1.9949	3.0015	4.0498
	color+3D+PSC	150 – 304.3 – 500	**3.9**	3.9	**4.8**	**1.4417**	**1.0215**	**2.6449**
Milk	Only 3D	500	10.7	6.0	8.2	3.9158	4.3284	7.2542
	color+3D	500	8.4	7.2	6.7	6.1082	4.8459	8.4107
	3D+PSC	150 – 379.8 – 500	5.9	4.6	7.1	3.2659	3.1813	4.2037
	color+3D+PSC	150 – 340.3 – 500	**5.0**	**4.0**	**5.8**	**3.1820**	**2.6500**	**3.7363**
KinectBox	Only 3D	500	10.4	23.2	14.1	4.9606	6.1498	5.0923
	color+3D	500	10.4	16.9	12.7	4.7392	7.0245	5.3812
	3D+PSC	150 – 342.5 – 500	9.3	19.2	9.7	3.6295	**3.3586**	3.3702
	color+3D+PSC	150 – 355.2 – 500	**5.9**	**10.4**	**7.4**	**1.9650**	3.4623	**1.7482**

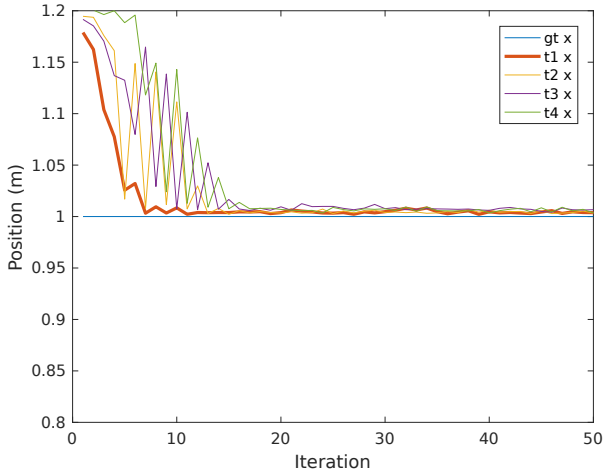
effect of varying the number of particles on the accuracy of tracking, i.e. the RMS tracking errors in rotation and translation w.r.t the ground truth. The results of this experiment are summarized in Table IV. From this evaluation, we can see that the use of primitive shape constraints improves the tracking error. In some cases, the performance of the "3D+PSC" tracker with 50 particles is close to or better than the "Only 3D" tracker with 200 particles. This confirms our expected behavior that the use of primitive shapes reduces the search subspace and hence, the required number of particles to sample the space.

We also plot the tracking errors over time for this dataset for the object WoodBox. Fig. 10 shows the results for this experiment, where tracking errors for the "3D+PSC" tracker in blue are lower than that of the "Only 3D" tracker throughout the tracking sequence.

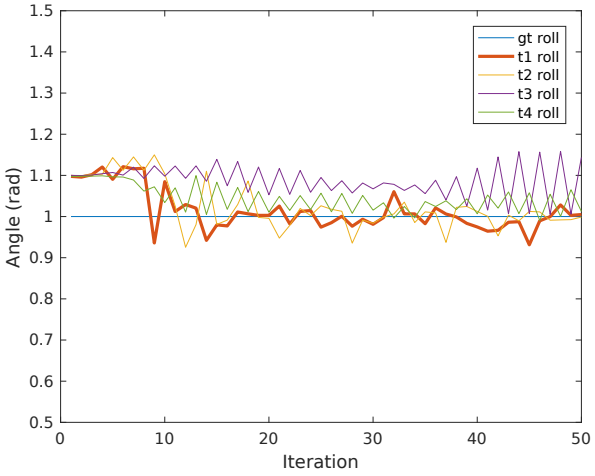
In addition to the synthetic data, we also performed several experiments with real sensor data captured using our experimental setup. Some images of these experiments showing the properties of our tracker are shown in Fig. 8. We compared the tracking trajectories for these two solvers on this database. Fig. 9 shows this comparison, where the trajectories for the "3D+PSC" tracker seem more smooth compared to those for the "Only 3D" tracker.

VI. Discussion

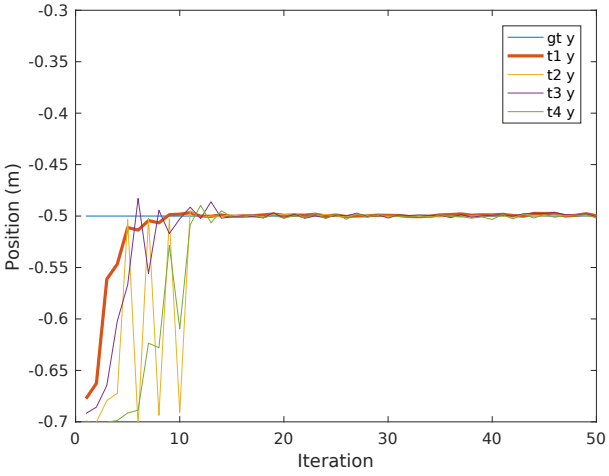
In this paper, we have presented an approach for exploiting geometric information about the object and its environment to improve the performance of a Particle Filter based tracker. This is achieved by combining an intelligent primitive shape detection component that can also estimate the shape fitting error, and an exact geometric constraint modeling framework that can generate the geometric nullspace with bounds based on these errors. An adaptive Particle Filter formulation restricts its search to this nullspace, and also adapts the process covariances according to the estimated fitting error.



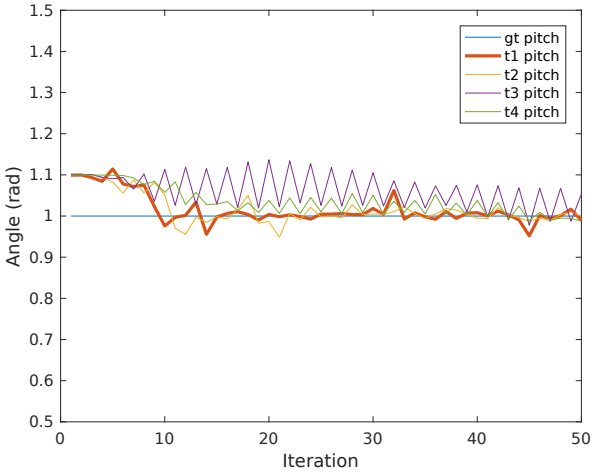
(a) Translation X



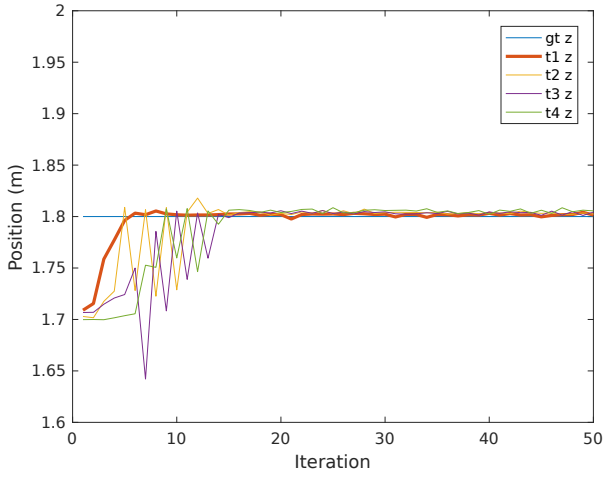
(b) Rotation Roll



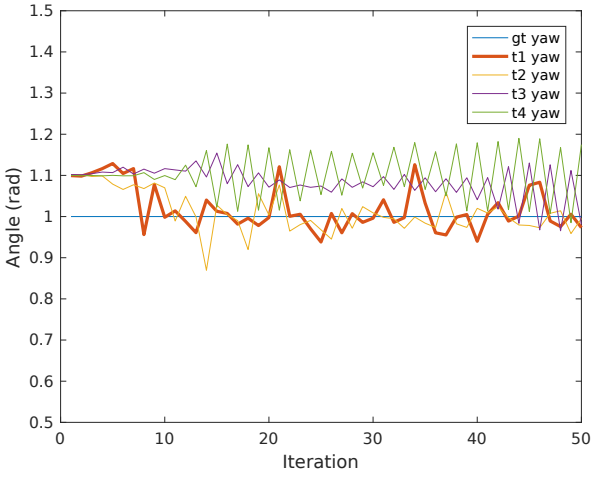
(c) Translation Y



(d) Rotation Pitch

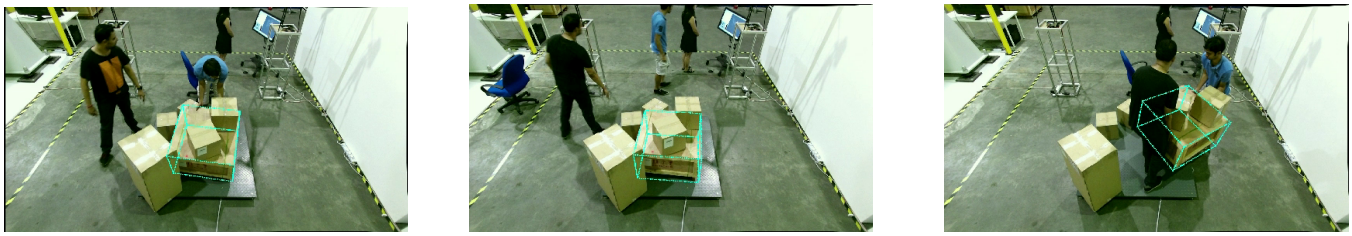


(e) Translation Z



(f) Rotation Yaw

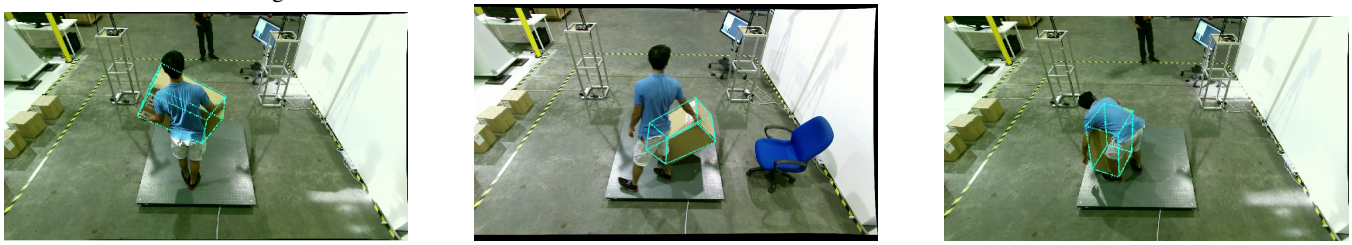
Fig. 7: Convergence of trackers from an initial pose, for the object Juice. The four trackers mentioned in Table III have been tested: t1 refers to “color+3D+PSC”, t2 refers to “3D+PSC”, t3 refers to “color+3D”, t4 refers to “’Only 3D”.



(a) WoodBox: Full 6d tracking

(b) WoodBox: Tracking in a cluttered scene

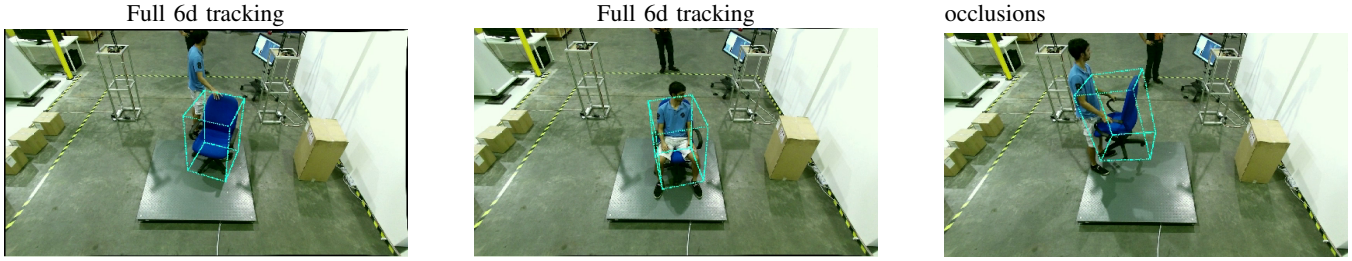
(c) WoodBox: Full 6d tracking in a cluttered scene with occlusions



(d) Box: Full 6d tracking

(e) Box: Full 6d tracking

(f) Box: Tracking in the presence of large occlusions



(g) Chair: Tracking of a more complicated object

(h) Chair: Tracking in the presence of occlusions

(i) Chair: Full 6d tracking

Fig. 8: Constraint-based tracking for 3 objects in our experimental setup.

TABLE IV: Evaluation of TPSC in a logistics scenario

Object	**Tracker**	**#Particles**	**RMS Errors (mm)**			**RMS Errors (deg)**		
			X	Y	Z	Roll	Pitch	Yaw
WoodBox	Only 3D	50	26.1	23.9	27.4	2.91	2.79	2.75
	Only 3D	100	21.4	25.0	23.2	2.11	2.29	1.82
	Only 3D	200	14.9	19.5	19.7	1.45	1.53	1.35
	3D+PSC	50	**14**	**14.1**	**16.3**	**1.69**	**1.57**	**1.40**
	3D+PSC	100	**14**	**15.2**	**13.6**	**1.21**	**1.29**	**1.08**
	3D+PSC	200	**10.1**	**12.7**	**14.3**	**1.097**	**0.87**	**0.85**
Box	Only 3D	50	15.4	17.3	56.2	5.74	19.05	7.36
	Only 3D	100	11.8	24.6	23.9	1.97	1.65	2.00
	3D+PSC	50	**14.3**	**12.6**	**34.0**	**5.01**	**18.87**	**7.13**
	3D+PSC	100	**10.3**	**19.4**	**16.1**	**1.25**	**1.07**	**1.28**

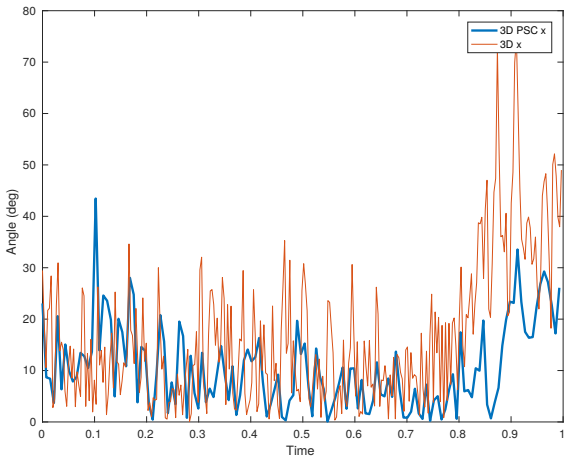
In principle, our approach can be used in higher dimensional spaces and is not specialized for 6D space. We focused our experiments towards 6D rigid objects, but this can be extended to articulated objects (e.g. hands, skeletons) in the future. Also, in many practical tracking scenarios the tracker complexity is also influenced heavily by the number of objects to be simultaneously tracked. If each object requires a smaller (and lower dimensional) search subspace based on our method, more objects can be tracked simultaneously with the same number of particles.

VII. Acknowledgment

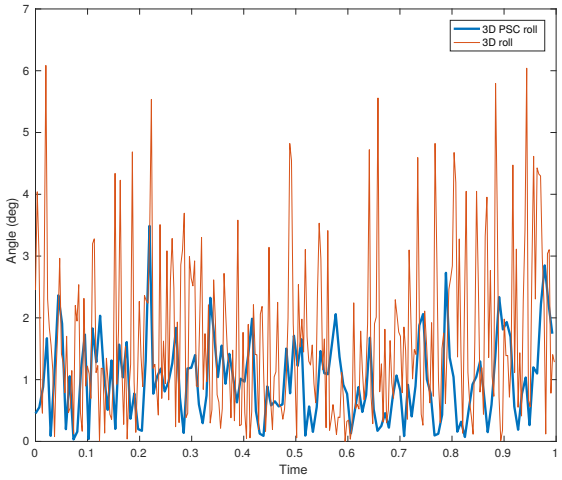
This work was financially supported by the Singapore National Research Foundation under its Campus for Research Excellence And Technological Enterprise (CREATE) programme.

References

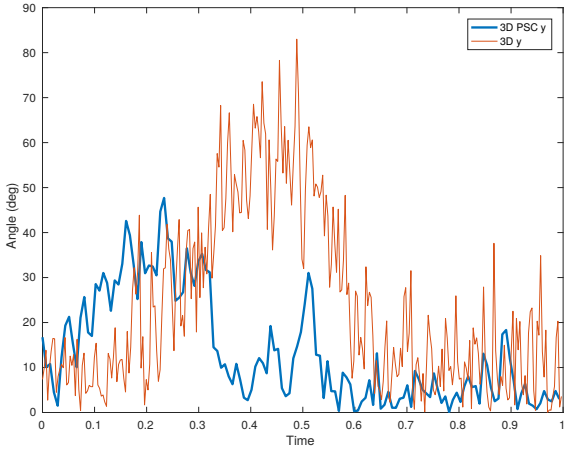
[1] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Computing Surveys*, vol. 38, no. 4, dec 2006. [Online]. Available: http://doi.acm.org/10.1145/1177352.1177355



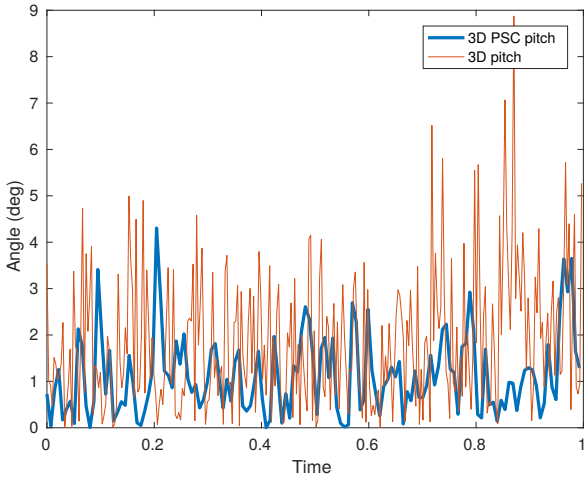
(a) Translation X



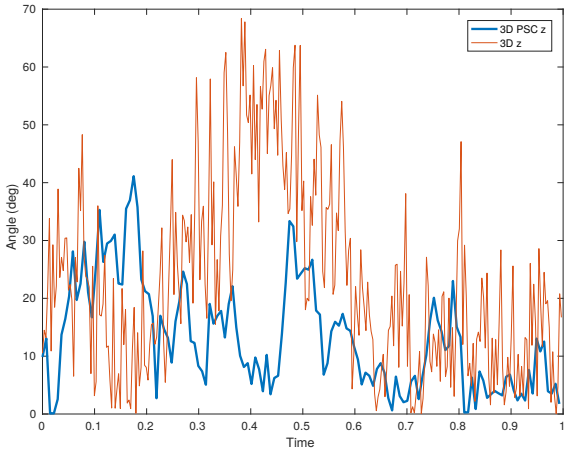
(b) Rotation Roll



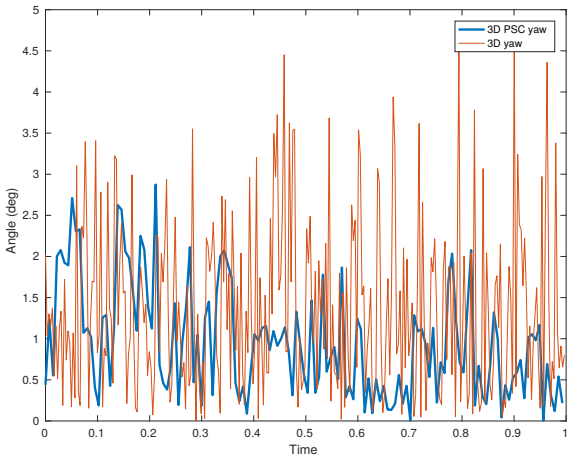
(c) Translation Y



(d) Rotation Pitch



(e) Translation Z

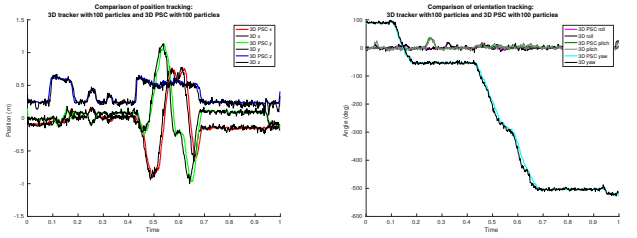


(f) Rotation Yaw

Fig. 10: Comparison of tracking errors on synthetic data from a pre-defined trajectory for the object WoodBox.

[2] Z. Chen, "Bayesian filtering: From kalman filters to particle filters, and beyond," *Statistics*, vol. 182, no. 1, pp. 1–69, 2003.

[3] Y. Bar-Shalom, *Tracking and data association*. Academic Press Professional, Inc., 1987.



(a) Trajectory for translation (b) Trajectory for orientation

Fig. 9: Comparison of trajectories for object WoodBox using our live experimental setup

[4] S. Nair, G. Panin, M. Wojtczyk, C. Lenz, T. Friedelhuber, and A. Knoll, "A multi-camera person tracking system for robotic applications in virtual reality tv studio," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2008.

[5] S. Nair, G. Panin, T. Röder, T. Friedelhuber, and A. Knoll, "A distributed and scalable person tracking system for robotic visual servoing with 8 dof in virtual reality tv studio automation," in *Proceedings of the 6th International Symposium on Mechatronics and its Applications (ISMA09)*. IEEE, 2009.

[6] C. Choi and H. I. Christensen, "RGB-D object tracking: A particle filter approach on GPU," *IEEE International Conference on Intelligent Robots and Systems*, pp. 1084–1091, 2013.

[7] G. Panin, C. Lenz, M. Wojtczyk, S. Nair, E. Roth, T. Friedlhuber, and A. Knoll, "A unifying software architecture for model-based visual tracking," pp. 681 303–681 303–14, 2008. [Online]. Available: http://dx.doi.org/10.1117/12.784609

[8] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes," in *Asian conference on computer vision*. Springer, 2012, pp. 548–562.

[9] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit, "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes," in *International Conference on Computer Vision*. IEEE, 2011, pp. 858–865.

[10] J. A. Brown and D. W. Capson, "A framework for 3d model-based visual tracking using a gpu-accelerated particle filter," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 1, pp. 68–80, 2012.

[11] S. Melax, L. Keselman, and S. Orsten, "Dynamics based 3d skeletal hand tracking," in *Proceedings of Graphics Interface 2013*, ser. GI '13. Toronto, Ont., Canada, Canada: Canadian Information Processing Society, 2013, pp. 63–70. [Online]. Available: http://dl.acm.org/citation.cfm?id=2532129.2532141

[12] C. Qian, X. Sun, Y. Wei, X. Tang, and J. Sun, "Realtime and robust hand tracking from depth," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2014, pp. 1106–1113.

[13] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, "Efficient model-based 3d tracking of hand articulations using kinect." in *British Machine Vision Conference (BMVC)*, vol. 1, no. 2, 2011, p. 3.

[14] H. Hamer, K. Schindler, E. Koller-Meier, and L. Van Gool, "Tracking a hand manipulating an object," in *Computer Vision, 2009 IEEE 12th International Conference On*. IEEE, 2009, pp. 1475–1482.

[15] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman filter: Particle filters for tracking applications*. Artech house Boston, 2004, vol. 685.

[16] J. Giebel, D. M. Gavrila, and C. Schnörr, "A bayesian framework for multi-cue 3d object tracking," in *European Conference on Computer Vision*. Springer, 2004, pp. 241–252.

[17] D. Fox, "Adapting the sample size in particle filters through kld-sampling," *The International Journal of Robotics Research*, vol. 22, no. 12, pp. 985–1003, 2003. [Online]. Available: http://dx.doi.org/10.1177/0278364903022012001

[18] G. Panin, *Model-based Visual Tracking: The OpenTL Framework*, ser. IT Pro. Wiley, 2011. [Online]. Available: https://books.google.com.sg/books?id=voYW0kGib8EC

[19] L. Zhang, J. Sturm, D. Cremers, and D. Lee, "Real-time human motion tracking using multiple depth cameras," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2389–2395, 2012.

[20] J. Behley, V. Steinhage, and A. B. Cremers, "Efficient Radius Neighbor Seach in Three-dimensional Point Clouds," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[21] A. Perzylo, N. Somani, M. Rickert, and A. Knoll, "An ontology for CAD data and geometric constraints as a link between product models and semantic robot task descriptions," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[22] N. Somani, A. Perzylo, C. Cai, M. Rickert, and A. Knoll, "Object detection using boundary representations of primitive shapes," in *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Zhuhai, China, December 2015.

[23] N. Somani, M. Rickert, and A. Knoll, "An exact solver for geometric constraints with inequalities," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1148–1155, April 2017, accepted for presentation at ICRA 2017.

[24] G. Pass and R. Zabih, "Comparing images using joint histograms," *Multimedia Systems*, vol. 7, no. 3, pp. 234–240, may 1999. [Online]. Available: http://dx.doi.org/10.1007/s005300050125

[25] A. Bhattacharyya, "On a measure of divergence between two multinomial populations," *Sankhyā: the indian journal of statistics*, pp. 401–406, 1946.

[26] M. Isard and A. Blake, "Condensation – conditional density propagation for visual tracking," *International Journal of Computer Vision (IJCV)*, vol. 29, no. 1, pp. 5–28, 1998.

[27] T. Wiedemeyer, "IAI Kinect2," https://github.com/code-iai/iai"_kinect2, Institute for Artificial Intelligence, University Bremen, 2014, accessed June 12, 2015.

[28] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, "Computing and rendering point set surfaces," *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 1, pp. 3–15, 2003.