2D-3D Collaborative Tracking (23CT): Towards Stable Robotic Manipulation

Sorin M. Grigorescu
Department of Automation
Transilvania University of Brasov
Brasov, Romania 500240
s.grigorescu@unitbv.ro

Dejan Pangercic
Robert Bosch LLC
Palo Alto, CA, USA
dejan.pangercic@us.bosch.com

Michael Beetz
Intelligent Autonomous Systems/Artificial Intelligence
Department of Computer Science and Centre for Computing Technologies (TZI)
University of Bremen, Germany
beetz@informatik.uni-bremen.de

***Abstract*—In this paper, the 2D-3D *Collaborative Tracking* (23CT) system for tracking rigid bodies in the context of mobile robotic manipulation is presented. The tracking approach is based on a collaborative tracking framework developed around two trackers: a 2D multi-class region of interest tracker and a 3D model-based tracker, where both trackers benefit from each other. The goal of this work is to improve the motion planning and the object handling capabilities of service robotics platforms that operate in complex and cluttered household environments and perform highly dexterous tasks such as mobile pick and place, pouring, flipping (of pancakes), etc. For performance evaluation, the proposed system was compared with a marker-based tracking system. The 23CT approach is using a visual data stream supplied by an RGB-D sensor, such as the MS Kinect$^{®}$.**

I. Introduction

One important task for service robots such as the PR2 (*Personal Robot* 2) [1], or the FRIEND (*Functional Robot with dexterous arm and user-frIENdly interface for Disable people*) [2] that operate in household environments, is to reliably handle common household objects (such as plates, mugs, pots, bottles, boxes) usually placed in heavy cluttered scenes. An example scene which falls in the above category can be seen in Fig. 1, where different objects placed on a table in front of a PR2 robot are to be handled.

Currently, a commonly used approach in many mobile manipulation scenarios is to recognize the objects of interest and obstacles in the scene, calculate the poses of the objects and perform the manipulation actions [3]. Throughout the manipulation procedure, there is usually no visual information available with respect to the dynamics of the scene, that is, to the motion of the grasped objects and to the obstacles. This increases the risk of failures if the state of the environment changes, namely, if the poses of the objects, as well as of different obstacles present in the scene, vary.

Although object tracking is a well-developed research area in the computer vision community, its application to mobile manipulation, and robotics in general, is rather unexplored. Recently, Krainin et al. [4], [5], applied the concept of object tracking during manipulation for building online 3D models of objects. Ueda et al. implemented a model-based tracker using a particle filtering algorithm and the KLD-sampling method which is available in Point Clouds Library [6]. Related to robotics, a hand tracking and modeling approach has been proposed in various works [7], [8].

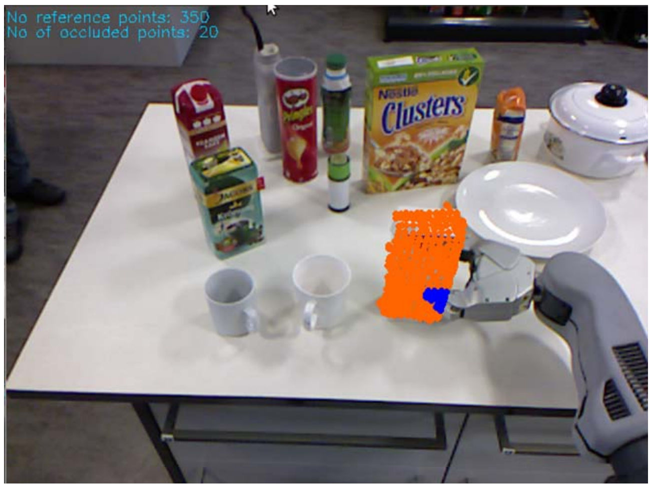

Fig. 1. Example tabletop scene consisting of the objects to be tracked during mobile manipulation.

Teichman and Thrun [9] proposed a semi-supervised approach to the problem of track classification in dense 3D range data. The method uses a boosting classification system on top of a series of 2D and 3D features, such as *Spin images*, *Histogram of Oriented Gradients* (HOG) and the object's *oriented bounding box* size.

A novel paradigm for training a binary classifier in the context of shape following has been proposed in [10]. The learning process is guided by *positive* (P) and *negative* (N) constraints which restrict the labeling of the dataset. P-N learning evaluates the classifier on unlabeled data, identifies examples that have been classified in contradiction with structural constraints and augments the training set with the corrected samples in an iterative process. An online boosting tracking technique has also been proposed by Grabner and Bischof [11].

In a number of recent papers, such as the one of Choi et al. [12], the output result from more trackers is fused together using a weighting scheme for the purpose of improving the performance of the overall tracking procedure.

Tracking has been heavy investigated also for the case of camera pose and dense 3D reconstruction of human environments. The DTAM (*Dense Tracking and Mapping*) system, proposed by Newcombe et al. [13], relies not on feature extraction, but on dense pixel-wise processing. As a single hand-held RGB-D camera is moved over a static scene, detailed textured depth maps at selected keyframes are estimated in order to produce a surface patchwork with millions of vertices.

The main contribution of the work presented in this paper is summarized as follows:

- the development of a baseline collaborative 2D-3D tracking framework that efficiently combines two trackers: a 2D online boosting tracker and a 3D model-based tracker;
- the application of this framework to the mobile manipulation tasks performed by the PR2 robot.

The rest of the paper is organized as follows. In Section II, the formulation of the tracking challenge that herein proposed system solves is given in the context of mobile manipulation. The deployed algorithms are described in Section III, followed by experimental results presented in Section IV. Finally, conclusions are stated in Section V.

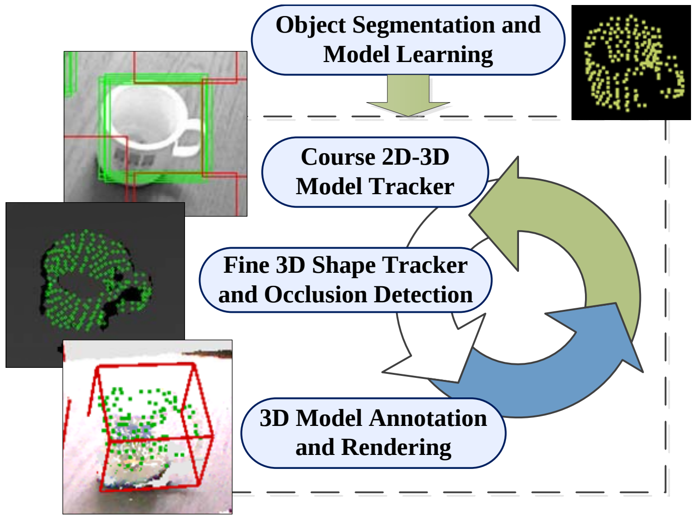
Object Segmentation and Model Learning
Course 2D-3D Model Tracker
Fine 3D Shape Tracker and Occlusion Detection
3D Model Annotation and Rendering

Fig. 2. The flow chart of the 2D-3D Collaborative Tracking system (23CT) consists of four steps: model generation through tabletop clustering, online multiclass boosting classification, occlusion detection and 3D-based fine transform calculation.

II. Problem Formulation

The goal of the proposed tracking system is to track the 3D poses ϕ of a set of rigid bodies C from RGB-D data streams, as obtained by Kinect, as accurate as possible:

$$\hat{\phi} = \arg\max_{\phi} P(\mathbf{C}, \phi | I, D); \tag{1}$$

where ϕ represents an affine 3D transformation which maps the 3D pose of an object between two frames and P is the likelihood of $\mathbf{C}$ and ϕ given I and D. I and D are the RGB and the depth information, respectively, delivered by the sensor. The tracked objects are defined as 3D point clusters, or *Point Distribution Models* (PDMs [14]), $\mathbf{C} = \{c_0, c_1, \ldots, c_K\}$, where K is the total number of tracked clusters. The pose of each cluster model c_k is given at every frame by its corresponding relative affine transformation $\hat{\phi}_k$. The position of the clusters is related to their centroids.

III. Methodology: The 23CT Tracking Framework

The flow chart of the 23CT collaborative tracking framework is illustrated in Fig. 2 for the case of a single object tracking. The tracking is initialized through tabletop object segmentation, which calculates the 3D reference model, or cluster, c_k of the object of interest. Once c_k is known, its shape is projected into the 2D image, where a classifier for tracking is trained on the resulted projection. Inside the tracking loop, the classifier establishes 2D correspondences between the consecutive frames $t-1$ and t, where t represents the discrete time. These 2D image matches are used to select the 3D point correspondences $\hat{m}_k$ between the corresponding point clouds $D[t-1]$ and $D[t]$ in the Kinect data. From $\hat{m}_k$, an initial course transform A_{course} for the reference cluster is calculated. Although, depending on the quality of the $\hat{m}_k$ matches, A_{course} can, to some extent, provide good tracking results, it fails to precisely map c_k onto the current cloud $D[t]$. For this reason, a second transform A_{fine} is determined using an *Iterative Closest Point* (ICP) [15] algorithm applied on non-occluded object points. Thus, the final object model transform $\hat{\phi}$, which tracks the pose of a 3D object between consecutive frames, can be written as:

$$\hat{\phi} = < A_{course}, A_{fine} > . \tag{2}$$

In the following, the calculation of the two affine transforms will be detailed.

A. Initialization: Object Segmentation

The 3D model is generated through a tabletop object segmentation. The latter is has been implemented in our earlier work [16] and is realized by grouping 3D points into separate object clusters from a point cloud $D[0]$. Firstly, the supporting plane on which the objects reside is segmented out. Secondly, points are grouped together into a cluster if the Euclidean distance between them is smaller than $0.02m$. The output of this operation is represented by a vector of object clusters $\mathbf{C}$ representing the reference object models defined as PDMs.

Along with the cluster representation, the 2D models are defined as regions of interest (ROIs) in the 2D image plane. The ROIs are calculated through the 2D projection of the clusters in $\mathbf{C}$ using the parameters of the sensor model (e.g. optical center and focal length).

B. Online Multiclass Object Tracking

Boosting is a machine learning technique used in a variety of computer vision applications such as image segmentation, text and object recognition, natural language processing, medical diagnostic, etc. In this paper, an *Online Multi-Class Boosting* (OMCB) approach [17] has been used to track the objects of interest in the 2D image domain.

The nature of multiclass online boosting, that is, of being able to learn online multiple object instances, made it a proper candidate for the 2D tracking task in the sense that the procedure does not only track multiple objects, but also models the scene's background as an extra class. In our approach, we have considered each cluster c_k to have a rectangular patch projection onto the image plane. This projection, or patch, is considered to be the so-called cluster's class $y(c) \in \{-1, 0, 1, 2, \ldots, K\}$. The value -1 represents the extra class through which the background of the scene is modeled. By modeling the background, the discriminative power of the classifier can be significantly improved. In Fig. 3(a), the background model is illustrated using a different color with respect to the objects classes.

The classifier maintains a model $f : \Re^d \rightarrow \Re^K$, which is a mapping from the input features space to the multiclass hypothesis domain [17]. The feature vector $\mathbf{x}$ used for training the classifier is composed of the following features:

- 6 types of *Haar-like features* [18] extracted from the rectangular patches, resulting in a number of 192 feature values normalized to the interval $[-1, 1]$, as proposed in [19];
- *color features* obtained from $16 \times 16px$ downsampled projected patches (see example patches in Fig. 3(a)), ranging in the interval $[0, 1]$; color was considered as the hue plane of the HSV (*Hue*, *Saturation*, *Value*) color space.

For each cluster class c_k, the classifier provides a confidence measure $f_k(\mathbf{x})$. In the multiclass space, the function f is defined as the vector:

$$f(\mathbf{x}) = \begin{bmatrix} f_{-1}(\mathbf{x}) & f_0(\mathbf{x}) & \ldots & f_K(\mathbf{x}) \end{bmatrix} \quad (3)$$

The new 2D image positions of the objects patches, relative to the previous frame, are determined as $\arg\max f(\mathbf{x})$.

C. Reference Model Tracking and Fitting

Once the new 2D positions of the objects patches have been determined with the help of multiclass boosting, the 3D pose of each reference cluster has to be calculated in the real-world 3D Cartesian space. This procedure is performed in three sequential steps. Firstly, the A_{course} transform is determined using a so-called *landmarks tracker*. The reference cloud is translated and rotated to the new pose given by A_{course}. Secondly, possible occlusions are calculated and finally the reference cluster is aligned with the current point cloud data using an ICP algorithm.

1) The landmarks tracker: represents the key component which connects the 2D boosting tracker with the 3D one. Namely, it is used to get 3D point correspondences from which A_{course} will be estimated. In this work, the tracker is represented by a combination of *Lucas-Kanade* (LK) optical flow and Shi-Tomasi features [20] which accurately determine correspondence keypoints between current and previous object patches. Once a set of matches has been calculated in the 2D image domain, their corresponding 3D values m_k are obtained by a direct mapping between the 2D point matches and their 3D correspondences in the current and previous point clouds. The mapping procedure is applied between the RGB and depth images delivered by the sensor.

An important challenge in determining point correspondences is the presence of outliers at the end of the procedure. In order to cope with this problem, a *Maximum Likelihood Estimator* (MLE) Φ was used to filter out bad matches [21]:

$$\hat{\Phi} = \arg\max_{\Phi} L(\Phi | \mathbf{M}(m_k)), \quad (4)$$

where $\hat{\Phi}$ is the maximum likelihood estimate for the Gaussian *Probability Density Function* (PDF) $P(\mathbf{M}|\Phi_\mu, \Phi_\sigma)$ describing the 3D orientation distribution of the lines connecting the 3D correspondence points. The inliers $\hat{m}_k$ are obtained from the set of 3D matches m_k using the mean Φ_μ and variance Φ_σ of the MLE. The A_{course} affine transformation which maps the reference cluster c_k to its approximate true pose in the new input point cloud is obtained through a *Singular Value Decomposition* (SVD) rigid body transform estimation. The SVD estimation can be used only if $\hat{m}_k \geq 4$. Although the new pose of c_k is correlated to the real object's pose, the reference cluster is not yet aligned with the input point cloud. This alignment is achieved using an SVD-based ICP algorithm which determines the A_{fine} transformation between the object cluster and the point cloud. One main drawback of ICP methods is that they often converge to the wrong local minima if the reference model to be aligned does not come with a good initial pose guess.

2) Occlusion detection: aims at filtering out the points that occlude the reference cluster transformed using A_{fine}. The goal of the method is to generate an non-occluded estimate $\hat{c}_k$ of the real reference cluster model c_k. In order to detect occlusions in real-time, the ray-casting approach has been used. Namely, the reference point features in c_k are projected at every frame based on the RGB-D sensor's intrinsic and extrinsic parameters. The obtained 2D projections are used to search the current point cloud for occlusions. Firstly, each point in c_k is projected onto the image plane. At each 2D location given by the projection, a corresponding point from the current depth image D is found, according to the ray-casting methodology. A point is said to be occluded if the point in D has a camera-point distance smaller than the one corresponding to the projected point in c_k. In this case, the point in the tracked cluster is occluded and thus is discarded from $\hat{c}_k$.

It is important to mention that the ICP determined A_{fine} transform provides proper 3D tracking results only if the number of non-occluded points is above a certain threshold, corresponding to a heuristically determined value of minimum 30% of the points in the tracked cluster. This means that the tracking results are stable for partial occlusions of the objects of interest. Algorithm 1 presents the whole 23CT tracking

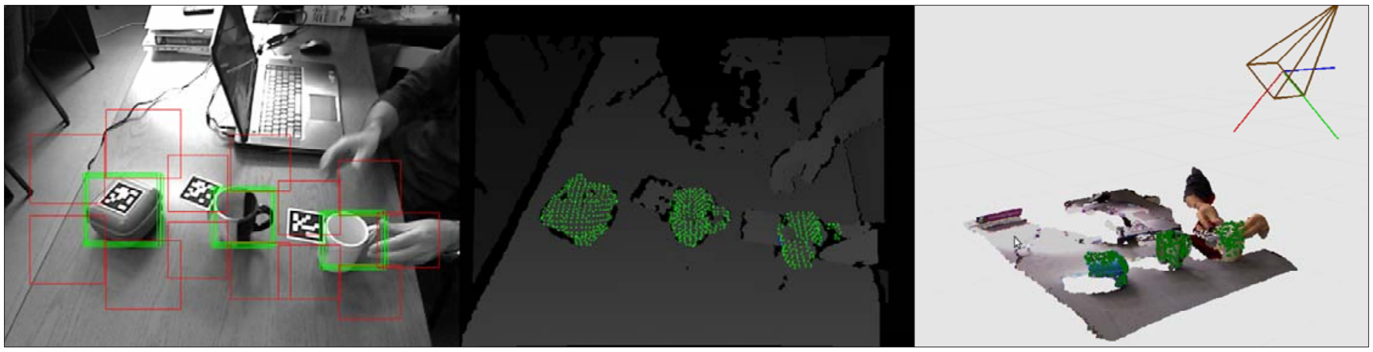
(a) (b) (c)

Fig. 3. Snapshot from the 23CT tracking loop (best viewed in color). (a) 2D multiclass object tracking; The green patches represent object classes, while the red ones the background model. (b) Aligned reference cluster models (shown in green) onto current depth math. (c) Perspective view on the tracked scene.

strategy as pseudocode[1].

```
Data: Input RGB-D stream.
Result: 3D poses of the object clusters at each input RGB-D frame.
1  while initialization do
2      Segment the input point cloud $D$ into distinct object cluster models $\mathbf{C}$;
3      Initialize the OMCB classifier with ROIs corresponding to the projections of $\mathbf{C}$;
4  end
5  while input RGB-D stream active do
6      Sample, evaluate and chose the best hypotheses $f(\mathbf{x})$ for the clusters ROIs;
7      foreach cluster $c_k$ in $\mathbf{C}$ do
8          Determine 3D landmarks $m_k(c_k[t], c_k[t-1])$;
9          Estimate the landmarks inliers $\hat{m}_k \subset m_k$;
10         if $\hat{m}_k \geq 4$ then
11             Determine the course transform $A_{course}$ using SVD($\hat{m}_k$);
12             Transform $c_k = c_k \cdot A_{course}$;
13             Calculate and remove occlusions $\hat{c}_k \subset c_k$;
14             ICP based fine transform $A_{fine}$ estimation between $\hat{c}_k$ and the input cloud $D$;
15             Transform $c_k = c_k \cdot A_{fine}$;
16             Reproject the transformed cluster $c_k$ onto the 2D image and update the cluster's ROI position and size;
17         end
18     end
19     Update the OMCB classifier using the new clusters ROIs.
20 end
```

Algorithm 1: Pseudo code of the 23CT multiple objects tracking approach.

IV. Experimental Evaluation

The evaluation of the overall visual tracking system has been performed with respect to the 3D ground-truth poses (3D position plus 3D orientation) of the objects of interest. For image acquisition, a MS Kinect® RGB-D camera, delivering $640px \times 480px$ size color and depth images, has been used. The ground-truth 3D poses were determined using the following setup. In the scenes, a visual marker was installed in such a way that the poses of the objects could be easily measured with respect to the marker. The 3D pose of the marker was detected using the ARToolKitPlus library [22] which provides subpixel accuracy estimation of the marker's location with an average error of approximately $5mm$. By calculating the markers 3D pose, a ground-truth reference value for the objects' position and orientation estimation was obtained. Further, the positions of the objects' features were calculated using the proposed system. Both results, that is the 2D and the 3D poses, were compared to the ground-truth data provided by the ARToolKitPlus marker tracking. The 2D image marker pose was calculated by projecting the marker's 3D pose onto the 2D image plane.

The pose evolution for an object tracking experiment, including 3 tracked objects over a sequence of 413 frames, is illustrated in Fig. 4[2]. As it can be seen from the diagrams, the positions and orientations of the tracked object follow relatively closely the ones delivered by the marker based recognition system. The calculated orientation error is low enough to ensure a stable tracking procedure. This correlation can be easily observed when analyzing the statistical error results between the two approaches, given in Tab. I. There are certain sample intervals in the diagrams from Fig. 4, such as $[252, 271]$ and $[347, 413]$, where the proposed 23CT tracker outperforms the traditional marker-based system. Namely, the ARToolKitPlus detection accuracy is strictly dependent on the adaptive thresholding procedure. Since in the mentioned intervals certain reflections influence the thresholded pattern, the

[1]The source code of the 23CT approach will be released soon under an open-source license.

[2]The accompanying video of this paper is also available online at http://youtu.be/tbLvyeCNuDE.

	X_e [m]	Y_e [m]	Z_e [m]	$Roll_e$ [deg]	$Pitch_e$ [deg]	Yaw_e [deg]
Max error	0.1013	0.0853	0.7039	12.0380	24.2644	18.2148
Mean	0.0760	0.0331	0.0143	4.0028	0.3394	7.9535
Std. dev.	0.0096	0.0190	0.0499	10.4121	16.7426	17.0336

TABLE I
STATISTICAL RESULTS OF ERRORS BETWEEN THE PROPOSED 23CT AND MARKER BASED 3D OBJECT TRACKING.

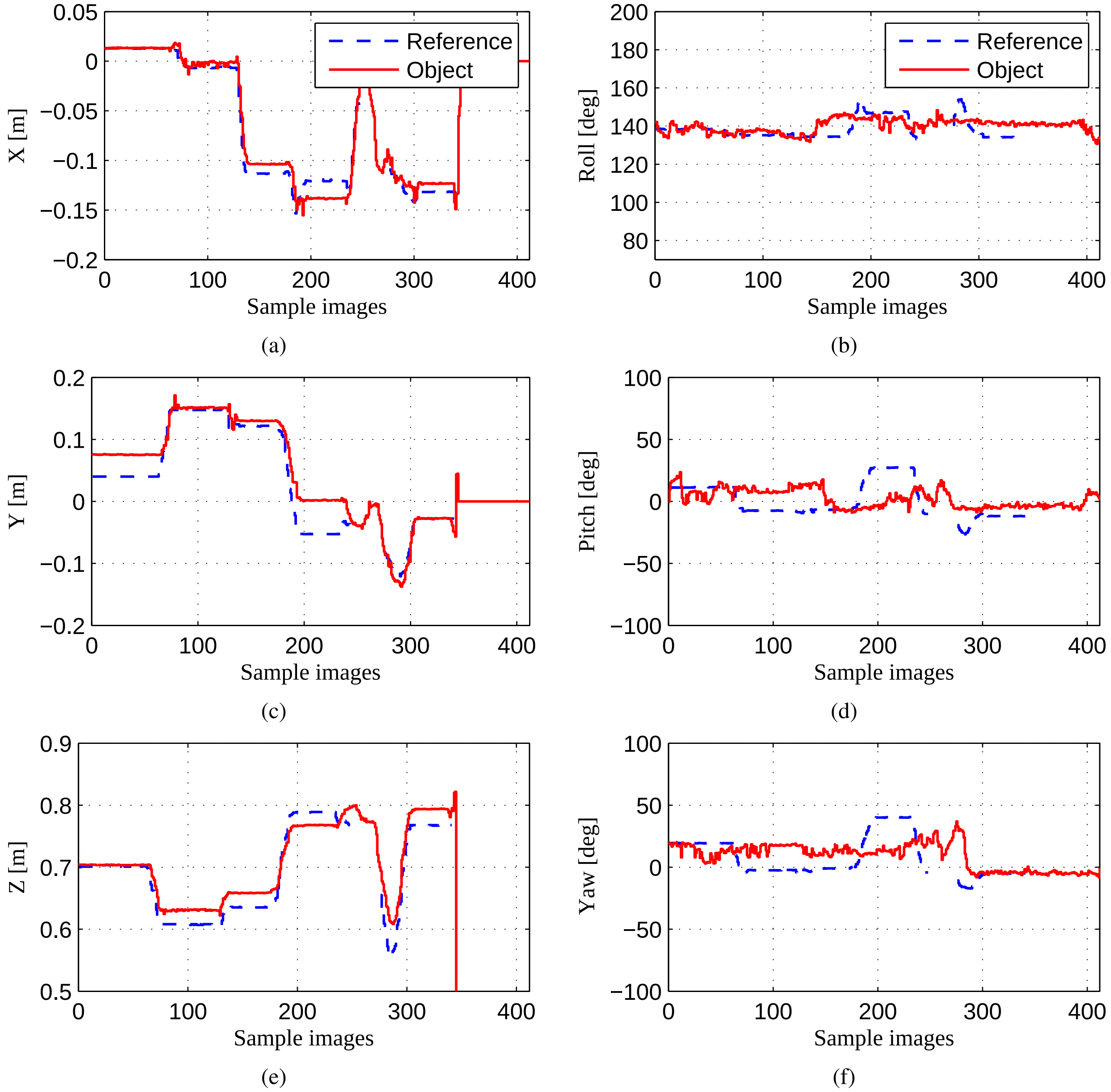

Fig. 4. Real (reference) and estimated poses of a tracked object of interest in the Cartesian space, corresponding to the sequence with the snapshot example from Fig. 3.

2D marker cannot be detected, hence the lack of ground-truth information in the mentioned intervals. An object occlusion example can be seen in the video accompanying this paper, between seconds 22-26.

The computation time required for tracking is dependent on the number of tracked objects. Due to the sequential implementation of the processing loop, the average tracking time is approx. $1-2$ *Frames per Seccond* (FPS). We expect a significant improvement through the implementation of the tracker into parallel processors.

V. Conclusions

In this paper, an object tracking algorithm, coined 23CT, with the purpose of stabilizing the mobile manipulation capabilities of service robots, has been proposed. One of the main contributions of the paper lies in the nature of the tracking loop, which, in comparison to many state of the art methods, includes the 3D visual information instead of 2D image data only. As seen from the experimental results section, the proposed approach offers stable tracking which can be used for the mobile manipulation tasks such as online adaptation of path planning, object slippage detection, physics-based projections, etc.

As future work, the authors plan to extend the proposed collaborative tracking framework with the inclusion of a motion model, the inclusions of the motion priors from the robot's end-effector, the inclusions of the simplified physics for the stability analysis and the development of new 2D/3D visual features which could be used in the tracking loop. Also, an execution time improvement can be achieved by relocating the sampling and features calculation functions to parallel computation devices such as *Graphical Processors* (GPUs) or *Field Programmable Gate Arrays* (FPGAs).

ACKNOWLEDGEMENTS

This paper is supported by the Sectoral Operational Program Human Resources Development (SOP HRD), financed from the European Social Fund and by the Romanian Government under the project number POSDRU/89/1.5/S/59323.

References

[1] M. Beetz, F. Stulp, P. Esden-Tempski, A. Fedrizzi, U. Klank, I. Kresse, A. Maldonado, and F. Ruiz, "Generality and legibility in mobile manipulation," *Auton. Robots*, vol. 28, no. 1, pp. 21–44, Jan. 2010.

[2] S. Grigorescu, T. Lüth, C. Fragkopoulos, M. Cyriacks, and A. Gräser, "A bci-controlled robotic assistant for quadriplegic people in domestic and professional life," *Robotica*, vol. 30, no. 3, p. 419, 2012.

[3] S. Chitta, E. G. Jones, M. Ciocarlie, and K. Hsiao, "Perception, planning, and execution for mobile manipulation in unstructured environments," *IEEE Robotics and Automation Magazine, Special Issue on Mobile Manipulation*, 2012.

[4] M. Krainin, P. Henry, X. Ren, and D. Fox, "Manipulator and object tracking for in-hand 3d object modeling," *Int. J. Rob. Res.*, vol. 30, pp. 1311–1327, September 2011.

[5] J. Krainin, B. Curless, and D. Fox, "Autonomous generation of complete 3d object models using next best view manipulation planning," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 2011.

[6] R. Ueda, "Point clouds library: Tracking," in pointclouds.org. [Online]. Available: http://docs.pointclouds.org/1.6.0/classpcl_1_1tracking_1_1_tracker.html

[7] L. Sun, U. Klank, and M. Beetz, "Eyewatchme - 3d hand and object tracking for inside out activity analysis," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009.*, June 2009, pp. 9–16.

[8] I. Oikonomidis, N. Kyriazis, and A. Argyros, "Tracking the articulated motion of two strongly interacting hands," in *CVPR 2012*. IEEE, June 2012.

[9] A. Teichman and S. Thrun, "Tracking-based semi-supervised learning," in *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2011.

[10] Z. Kalal, J. Matas, and K. Mikolajczyk, "P-n learning: Bootstrapping binary classifiers by structural constraints," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, San Francisco, CA, USA, June 2010.

[11] H. Grabner and H. Bischof, "On-line boosting and vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, New York, NY, USA, June 2006.

[12] W. Choi, C. Pantofaru, and S. Savarese, "Detecting and tracking people using an rgb-d camera via multiple detector fusion," in *Workshop on Challenges and Opportunities in Robot Perception, at the International Conference on Computer Vision (ICCV)*, Barcelona, Spain, November 2011.

[13] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Barcelona, Spain, November 2011.

[14] T. Cootes and C. Taylor, "Statistical models of appearance for computer vision," 2000.

[15] P. Besl and N. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, pp. 239–256, 1992.

[16] U. Klank, D. Pangercic, R. B. Rusu, and M. Beetz, "Real-time CAD Model Matching for Mobile Manipulation and Grasping," in *9th IEEE-RAS International Conference on Humanoid Robots*, Paris, France, December 7-10 2009, pp. 290–296.

[17] A. Saffari, M. Godec, T. Pock, C. Leistner, and H. Bischof, "Online Multi-Class LPBoost," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[18] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," 2001, pp. 511–518.

[19] S. Hare, A. Saffari, and P. H. S. Torr, "Struck: Structured output tracking with kernels." in *ICCV*, D. N. Metaxas, L. Quan, A. Sanfeliu, and L. J. V. Gool, Eds. IEEE, 2011, pp. 263–270.

[20] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[21] S. M. Grigorescu, T. T. Cocias, G. Macesanu, and F. Moldoveanu, "Stereo vision-based 3d camera pose and object structure estimation - an application to service robotics." in *VISAPP (2)*, G. Csurka and J. Braz, Eds. SciTePress, 2012, pp. 355–358.

[22] D. Wagner and D. Schmalstieg, "ARToolKitPlus for Pose Tracking on Mobile Devices," Institute for Computer Graphics and Vision, Graz University of Technology, Tech. Rep., Feb. 2007.