In: Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS 2014), Chicago (2014).

Real-time Object Pose Recognition and Tracking with an Imprecisely Calibrated Moving RGB-D Camera

Karl Pauwels[1], Vladimir Ivan[2], Eduardo Ros[1] and Sethu Vijayakumar[2]

***Abstract*— We introduce a real-time system for recognizing and tracking the position and orientation of a large number of complex real-world objects, together with an articulated robotic manipulator operating upon them. The proposed system is fast, accurate and reliable and yet does not require precise camera calibration. The key to this high level of performance is a continuously-refined internal 3D representation of all the relevant scene elements. Occlusions are handled implicitly in this approach and a soft-constraint mechanism is used to obtain the highest precision at a specific region-of-interest. The system is well-suited for implementation on Graphics Processing Units and thanks to a tight integration of the latter's graphical and computational capability, scene updates can be obtained at framerates exceeding 40 Hz. We demonstrate the robustness and accuracy of this system on a complex real-world manipulation task involving active endpoint closed-loop visual servo control in the presence of both camera and target object motion.**

I. Introduction

The task of recognizing known objects and estimating their position and orientation is a well-studied problem. It is also a crucial component of a robotic system designed for manipulating objects. Even when the visual appearance of these objects is known, recognition and pose estimation become increasingly difficult in scenes with a large number of objects, clutter, varying lighting conditions or when the camera and/or the objects move at high speed. On top of that, tedious and sensitive calibration procedures are required in order to extract precise 3D information.

We propose a method capable of real-time 3D position and orientation estimation of textured rigid and articulated objects. Our technique makes use of highly parallelized computation on Graphics Processing Units (GPUs). This allows it to perform object pose recognition and pose tracking at rates exceeding 40 Hz and it also scales the method to a large number of tracked objects. Manipulation tasks (especially in cluttered environments) rely heavily on accurate and fast pose estimation [1]. Manipulating objects and tools [2] or passing objects between human and a robot [3] are still hard problems that are being actively researched. The method proposed here achieves high pose tracking accuracy at speeds exceeding typical camera frame rates, enabling practical deployment in real-time robotics applications operating in unstructured environments with significant and intermittent occlusion.

We demonstrate our system on a manipulation task that involves visual servoing. We combine multi-object tracking of complex objects in real-time with proprioception and the kinematic constraints of the robot to improve the accuracy of tracking by constraining the motion of tracked objects. Self-occlusion is handled implicitly while the robustness against occlusion from objects that are not being tracked is high.

A. Main Contributions

As compared to our previous work on real-time combined pose recognition and tracking [4] we present three main contributions.

1) Multi-object tracking: We extend the method to simultaneously track a (potentially very) large number of objects. Due to the 3D scene representation used, self-occlusions and occlusions between tracked objects are handled implicitly by the approach.

2) Joint Object–Manipulator Tracking: By incorporating real-time proprioceptive information and kinematic constraints, we extend the approach to allow highly accurate manipulator tracking as well. Using a unique two-way consistency resolution paradigm, we exploit the information provided by grasped objects to facilitate manipulator tracking and vice versa.

3) Imprecise Calibration: We no longer rely on a precisely calibrated stereo configuration as in [4], but instead employ an imprecisely calibrated RGB-D camera (Kinect). We present two mechanisms to achieve reliability and robustness to this and the various other noise sources present in the system. Rather than attempting to exactly match the entire scene to our internal representation, we focus on a specific *region-of-interest* where high precision is required (*e.g.* where the interaction takes place) and use a *soft-constraint* mechanism to incorporate information obtained from other areas in the scene.

B. Related Work

There is considerable recent work focusing on object pose or viewpoint recognition, and in particular on scaling this problem (in real-time) to a very large number of objects [5], [6]. Our primary focus here is on pose *tracking* rather than *detection*. Although we discuss how both are combined, we rely on standard methods for the recognition element.

With regard to real-time pose tracking, many efficient model-based methods exploit edge- or region-based information or both [7] and often use particle filtering to enhance robustness [8]. Recently, also dense depth information is being

*This work has been supported by grants from the European Commission (Marie Curie FP7-PEOPLE-2011-IEF-301144 and TOMSY FP7-270436) and CEI BioTIC GENIL (PYR-2014-6 and PYR-2014-16). The GPU used for this research was donated by the NVIDIA Corporation.

[1]Computer Architecture and Technology Department, University of Granada, Spain {kpauwels,eros}@ugr.es

[2]School of Informatics, University of Edinburgh, UK {v.ivan,sethu.vijayakumar}@ed.ac.uk

considered, *e.g.* in visual servoing [9]. The method presented here extends a method that combines sparse keypoints with dense motion and depth information [4].

Another group of work also considers the manipulator in tracking [10]–[13]. All these methods involve only a single object and rely on precisely calibrated sensors, without allowing the camera(s) to move. Calibrating multi-camera systems is non-trivial and RGB-D sensors, such as Kinect, are known to exhibit depth distortions that increase with range [14].

Model-based tracking has been widely used for manipulation and tool handling in robotics [2]. While solving most such tasks can be improved by accurate pose tracking, more complex tasks require robust detection and tracking capabilities, *e.g.* inferring object properties [15], manipulation in clutter [1] or interaction with humans [3]. Analyzing interactions often involves extraction of semantic information from the tracked data [16], [17], which also relies on pose estimation in difficult conditions with occlusions.

II. Proposed Method

An overview of the proposed method is shown in Fig. 1. We consider an environment with multiple freely moving objects (among which the servoing target object) observed by a freely moving Kinect camera. This sensor provides RGB color video together with dense depth data at a 30 Hz framerate. In addition, encoder values are sampled from the robot arm and hand synchronous with the Kinect data. Sparse keypoints (Scale-Invariant Feature Transform or SIFT features) [18] together with dense motion (*cf.* optical flow) are extracted from the images (Section II-B). The keypoints are used exclusively to initialize or reset the tracker in case objects are lost. The other cues are used for tracking. The tracker continuously updates the internal state in order to maximize the consistency between these cues and a detailed 3D representation of the environment (Section II-A). The tracker is the core of the proposed method and is explained in detail in Section II-C. The internal state is used here to control the robot in a classic visual servoing task (Section II-E).

A. Scene Representation

The modeled scene consists of rigid objects, the robot arm, and the robot hand (Fig. 1B). The objects are represented by 3D textured wireframe models. A commercial solution was used to construct detailed models from a collection of pictures taken from different viewpoints [19]. Once the models are constructed, SIFT features are extracted from different viewpoints and mapped onto the model surface. The robot arm (KUKA LWR4) and hand (Schunk Dexterous Hand 2.0) are represented by colored 3D wireframe models that were generated from CAD models provided by their respective manufacturers. They do not contain useful texture for tracking.

In order to update the state in response to the arrival of new sensory data, the scene needs to be synthesized (rendered) multiple times (Section II-C). We use a collection of custom OpenGL shader-code for this. By rendering the scene through OpenGL, self-occlusions and occlusions between different objects and the robot arm or hand are managed automatically using the z-buffering mechanism.

Figure 2 shows some of the information that can be extracted from a successfully tracked scene. By rendering the scene we can obtain textured views (B,F) from arbitrary viewpoints, object and robot part segments (D), together with depth (G) and normals (H). Note that the base of the robot (the textured table, yellow in D) is also considered as an independently moving rigid object. The robot arm and hand are composed of multiple separate rigid models to allow for their articulation.

B. Sensor Data and Visual Features

The input to the algorithm consists of images (Fig. 2A) and depth (*i.e.* Kinect disparity, Fig. 2E) provided by a Kinect sensor at 30 Hz. We rely on OpenCV for mapping the depth image onto the RGB image and do not perform any additional processing (*e.g.* undistortion). We do not manually calibrate the Kinect camera, but use inaccurate default parameters. Specifically we use a focal length equal to 525 pixels, a baseline equal to 8 cm, and assume that the nodal point is located in the image center. We also sample the motor encoder values at the same framerate. From these we obtain precise position information ($\pm$0.05 mm repeatability) for the different components of the robot arm and hand in the robot's frame of reference.

We use efficient GPU libraries to extract dense phase-based motion cues [20] and SIFT features [21] from the images. The optical flow algorithm used integrates the temporal phase gradient across different orientations and uses a coarse-to-fine scheme to increase the dynamic range. Rather than extracting optical flow directly from the images we instead compute optical flow between a (partially) synthetic image, generated on the basis of the current tracking hypothesis (*i.e.* the current pose estimates), and the next image. When used for tracking, this type of motion information is insensitive to drift since it measures the difference between the current scene hypothesis and the observed scene (rather than simply the image motion). For the same reason it can be used to measure the reliability of tracking. We refer to this as Augmented Reality (AR) flow since the partially synthetic image combines rendered information with the real image. An example is shown in Fig. 2C. When compared to the actual image (Fig. 2A) lighting differences (specularities on the bottle and shading inside the mug) and non-modeled occlusions (the thumb) are visibly different. The inherent robustness of phase-based measures to intensity changes is critical here. Note that we do not blend the robot since its texture is not useful for tracking (occlusions are however considered implicitly, *e.g.* at the robot finger). See [4] for more details on AR flow.

C. Object and Manipulator Pose Tracking

Considering a single object model for now, we aim to estimate the rigid rotation and translation that transforms

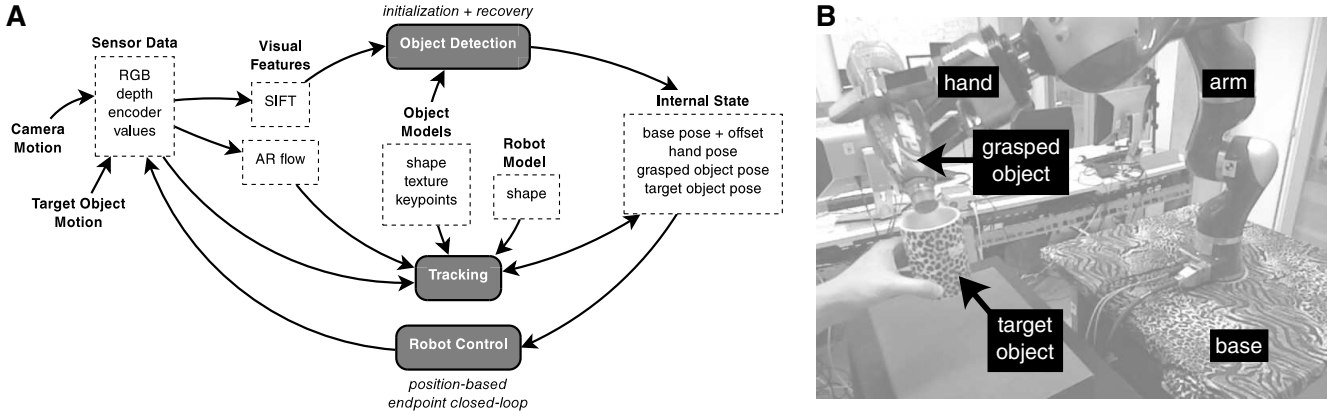

Fig. 1. (A) Method overview (AR = Augmented Reality) and (B) scene elements of interest.

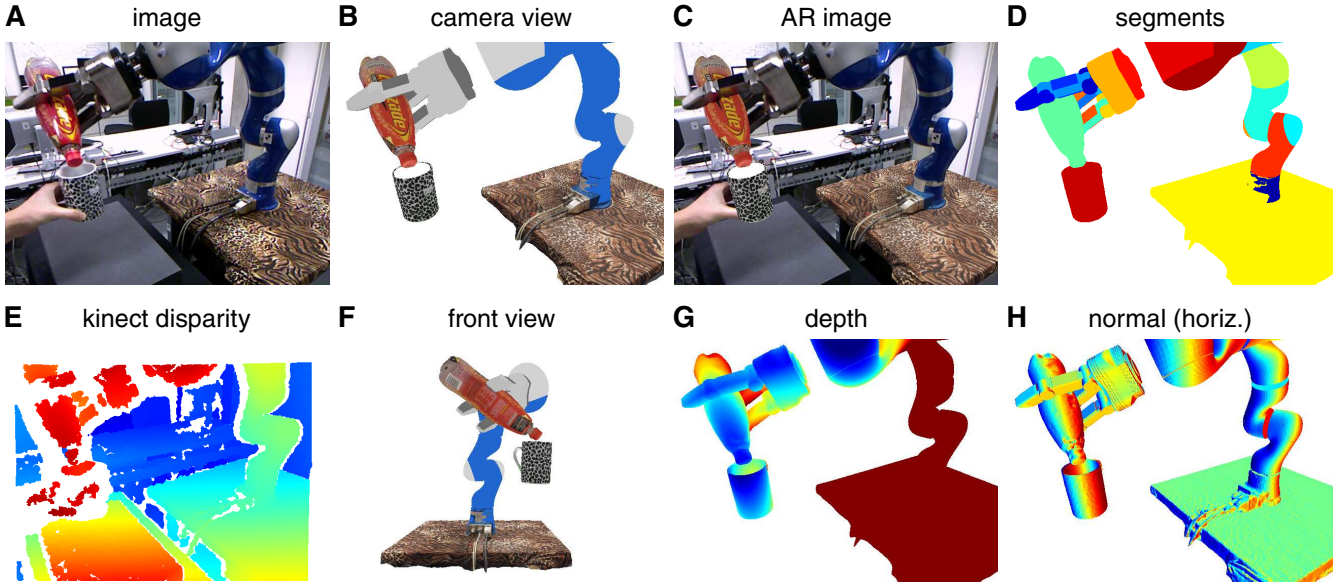

Fig. 2. Scene representation and sensory cues. (A) RGB image and (E) disparity obtained from Kinect sensor, tracker state rendered from (B) camera point-of-view and (F) alternative point-of-view. (C) Augmented Reality image obtained by blending the object parts (but not the robot) of (B) with (A). Additional information obtained from scene rendering: (D) object and part segments, (G) depth (scaled here to highlight region-of-interest), and (H) normals (only horizontal component shown).

each model point $\mathbf{m} = [m_x,\, m_y,\, m_z]^\top$ at time t into point $\mathbf{m}'$ at time $t+1$:

$$\mathbf{m}' = \mathrm{R}\,\mathbf{m} + \mathbf{t}\ , \tag{1}$$

according to the rotation matrix R and the translation vector $\mathbf{t} = [t_x,\, t_y,\, t_z]^\top$ that best explains the observed motion and/or depth cues. The rotation matrix can be simplified using a small angle approximation:

$$\mathbf{m}' \approx (\mathbf{1} + [\boldsymbol{\omega}]_\times)\,\mathbf{m} + \mathbf{t}\ , \tag{2}$$

with $\boldsymbol{\omega} = [\omega_x,\, \omega_y,\, \omega_z]^\top$ the rotation axis and angle, and $[\boldsymbol{\omega}]_\times$ the skew-symmetric cross product matrix.

For the *depth cues*, an efficient Iterative Closest Point (ICP) approach is used. The correspondences are obtained using projective data association [22], effectively matching the Kinect depth measurements to model points that project to the same pixel. The point cloud at time $t+1$ is reconstructed from the Kinect disparity d' as follows:

$$\mathbf{s}' = [s'_x,\, s'_y,\, s'_z]^\top = [x\, s'_z/f,\, y\, s'_z/f,\, f\, b/d']^\top\ , \tag{3}$$

with pixel coordinates $\mathbf{x} = [x, y]^\top$ (with nodal point as origin), focal length f and b Kinect baseline. Recall that these calibration parameters are only approximate. A linearized version of the point-to-plane distance is then used [23] as target of the optimization. This results in the following error function:

$$e_D(\mathbf{t}, \boldsymbol{\omega}) = \sum_i \left(\left[(\mathbf{1} + [\boldsymbol{\omega}]_\times)\,\mathbf{m}_i + \mathbf{t} - \mathbf{s}'_i \right] \cdot \mathbf{n}_i \right)^2\ , \tag{4}$$

with $\mathbf{n}_i$ the normal at sample i obtained from rendering the scene (Fig. 2H).

For the *motion cues*, the differential motion equation is used [24]. It relates the 3D motion of a point, $\dot{\mathbf{m}}$, to its 3D

translational and rotational velocity:

$$\dot{\mathbf{m}} = \mathbf{t} + [\boldsymbol{\omega}]_\times \mathbf{m} \ . \tag{5}$$

By projecting this 3D motion into the image, the pixel motion $\dot{\mathbf{x}} = [\dot{x}, \dot{y}]^\top$ can be expressed as follows [24]:

$$\dot{x} = \frac{(f\, t_x - x\, t_z)}{m_z} - \frac{x\, y}{f}\, \omega_x + (f + \frac{x^2}{f})\, \omega_y - y\, \omega_z \ , \tag{6}$$

$$\dot{y} = \frac{(f\, t_y - y\, t_z)}{m_z} - (f + \frac{y^2}{f})\, \omega_x + \frac{x\, y}{f}\, \omega_y + x\, \omega_z \ , \tag{7}$$

which is linear in $\mathbf{t}$ and $\boldsymbol{\omega}$, provided that the depth of the point (m_z) is known. To maximize complementarity of the cues, we do not use the Kinect depth (s_z) here but rather the depth obtained from rendering the model at the current pose estimate (Fig. 2G). In this way motion information can still be exploited in the absence of Kinect depth (*e.g.* when the object is close to the camera). We then have the following error function for motion:

$$e_M(\mathbf{t}, \boldsymbol{\omega}) \quad = \quad \sum_i \| \dot{\mathbf{x}}_i - \mathbf{a}_i \|^2 \ , \tag{8}$$

with $\mathbf{a} = [a_x,\, a_y]^\top$ the observed AR flow.

Both the linearized point-to-plane distance in the depth case and the differential motion constraint in the motion case now provide linear constraints on the same rigid motion representation $(\mathbf{t}, \boldsymbol{\omega})$ and can thus be minimized jointly using the following error function:

$$E(\mathbf{t}, \boldsymbol{\omega}) = e_D(\mathbf{t}, \boldsymbol{\omega}) + e_M(\mathbf{t}, \boldsymbol{\omega}) \ . \tag{9}$$

The samples used to compose (4) and (8) can be different. In the case of large rotations these linearized constraints are only crude approximations, and many of the shape correspondences obtained through projective data association will be wrong. Therefore, the minimization of (9) needs to be iterated a number of times, at each iteration updating the pose, the shape correspondences, and the unexplained part of the AR flow measurements. An M-estimation scheme is also used to increase robustness to outliers [25]. See [4] for more details on this procedure.

We employ different strategies here for tracking the different scene parts (Fig. 1B).

1) Target Object and Robot Base: Since the rigid objects and the robot base are assumed textured, both motion and depth cues can be used in their update. Tracking the base provides an indication of camera motion whereas the tracked target guides the servoing.

2) Robot Arm and Hand: The robot's arm and hand are articulated objects, but since we have access to the encoder values, we update the articulation at the start of each frame and consider the robot rigid. What remains then, is to correctly position and orient it with respect to the (moving) camera. This may appear to be a trivial problem since tracking the robot base already gives us an indication of the camera motion. However, at this point various sources of error manifest themselves. Specifically we are dealing with:

- imprecise internal calibration of RGB and depth cameras (focal length, baseline, nodal point, depth-to-RGB mapping, lens distortion, ...)
- imprecise tracking of the base (*i.e.* external calibration)
- imprecise geometry and/or texture of object, robot, and base models
- imprecise model connection between the scanned base and the arm CAD model
- inaccurate encoder values
- lags due to communication delays

Visually tracking (parts of) the robot itself allows us to compensate for these errors. Since the robot's arm and hand are glossy and uniformly colored, it is difficult to extract good quality visual motion at their location. We therefore only rely on the Kinect's depth signal to refine the robot's rigid pose. It is difficult to precisely match the robot in its entirety due to the imprecise internal calibration. We therefore only use the parts we are interested in for tracking, our *region-of-interest*. In the servoing task considered here, the hand is most interesting.

The tracked base is still useful for stabilizing this estimation and for dealing with rapid camera motion. For this reason we connect the rigid pose updates of the robot hand and the robot base using a soft-constraint mechanism. We can rewrite (9) as follows to expose the linearity:

$$E(\boldsymbol{\alpha}) = (\mathbf{F}\boldsymbol{\alpha} - \mathbf{c})^\top (\mathbf{F}\boldsymbol{\alpha} - \mathbf{c}) \ , \tag{10}$$

where the rigid motion parameters are stacked into a vector $\boldsymbol{\alpha} = \begin{pmatrix} \boldsymbol{\omega} \\ \mathbf{t} \end{pmatrix}$ for convenience, and $\mathbf{F}$ and $\mathbf{c}$ are obtained by gathering the sensor data according to (4),(6),(7). This can be solved in the least-squares sense using the normal equations:

$$\mathbf{F}^\top \mathbf{F} \boldsymbol{\alpha} = \mathbf{F}^\top \mathbf{c} \ . \tag{11}$$

Instead, the solution can be biased towards a specific solution, $\boldsymbol{\alpha}_p$, as follows:

$$\left(\mathbf{F}^\top \mathbf{F} + \boldsymbol{\lambda}\right) \boldsymbol{\alpha} = \mathbf{F}^\top \mathbf{c} + \boldsymbol{\lambda} \boldsymbol{\alpha}_p \ . \tag{12}$$

This is similar to ridge regression but instead of penalizing the size of the coefficients, it penalizes their difference from the prior $\boldsymbol{\alpha}_p$ [26]. We let $\boldsymbol{\lambda}$ scale with the diagonal of $\mathbf{F}^\top \mathbf{F}$ to fix the relative contribution of data and prior.

3) Grasped Object: In the scenarios considered here the grasped object is rigidly manipulated by the robot. As a result it can guide the hand pose update and, conversely, the hand pose update can guide the object pose update. For this reason we update both jointly. This is simply done by adding the depth and AR flow measurements of the grasped object to $\mathbf{F}$ and $\mathbf{c}$ in the soft-constraint normal equations (12).

D. Object and Manipulator Pose Detection

So far we have only discussed pose tracking. This does not enable initializing the pose or recovery in case of severe occlusions. For this purpose we incorporate a standard RANSAC-based monocular perspective-n-point pose detector [27]. This detector matches SIFT keypoint descriptors between the image (2D) and the model codebook (3D).

To improve efficiency and accuracy, only one object is considered at a time. The object-of-interest is determined probabilistically, depending on the tracking reliability of that part. Note that only the objects and the robot base are considered here. The robot itself is insufficiently textured for this purpose. Instead, the base is used to re-initialize its location. In the same way as in [4], we *select* either the sparse or dense estimate based on the proportion of valid AR flow that can be extracted in their region. This measures how well the synthetic image matches the observed image. It also signals non-modeled occlusions (such as hands) and will thus trigger a switch to the detector.

E. Robot Control

The robot used in our experiments is a KUKA LWR4 arm with a Schunk Dexterous Hand 2.0 mounted as the end-effector. This system allows us to control seven Degrees-Of-Freedom (DOFs) of the arm to grasp and fully position and orient an object, controlling all six DOFs of its movement.

The proposed system is capable of tracking the end-effector (the robot hand holding the object) as well as the manipulated objects. This allows us to perform Endpoint Closed Loop control (ECL) [28]. The corrections in the Cartesian space are computed using an iterative Inverse Kinematics (IK) with weighted pseudo-inverse method. Here, we specify the optimization criteria of the IK solver as: the position of the tip of the manipulated object (the bottle cap) and the alignment of the object (alignment of the central axis of the bottle). The position target is set to the tracked position of the target object (the mug) and the axis alignment is controlled with respect to the gravity vector to simulate a pouring motion. The alignment is controlled manually by the operator, since its target does not depend on the tracked objects. We let the IK solver converge until the position error is smaller than 0.001 mm (which is below the robot's repeatability of ± 0.05 mm). This kinematic motion has been arbitrarily chosen to mimic a real-life task.

F. Implementation Aspects

The timings mentioned in this section were obtained using an NVIDIA Geforce GTX 590 and an Intel Core i7. The system relies heavily on the integration between OpenGL and NVIDIA's CUDA framework in order to achieve real-time performance.

The AR flow is computed using a highly efficient coarse-to-fine algorithm [4]. Since we rely on dense visual cues (easily exceeding 100,000 measurements), the bulk of the computational effort is spent on the composition of the normal equations (11),(12). This can however be performed independently and in parallel for each segment (free objects, base, grasped object, hand) making it ideally suited for GPU acceleration. Apart from this, the most time-consuming step is assigning the valid measurements to their respective segments (in accordance with the current pose estimates). For this, the segment indices (Fig. 2D) need to be sorted. Since the number of segments is limited, an efficient radix sort can be used here [29].

TABLE I
POSE TRACKING FRAME RATES

	# motion/depth samples	
# segments	50,000	500,000
1	62 Hz	57 Hz
20	55 Hz	44 Hz
150	47 Hz	38 Hz

Table I shows the achieved frame rates for different numbers of tracked parts as a function of number of data samples (AR flow and Kinect depth) used. In the servoing scenario considered here we are only tracking four distinct parts; the robot base, target object, grasped object, and hand. We achieve tracking framerates around 50 Hz for this scenario. The SIFT-based part detection runs independently on the second GPU of the GTX 590. Its estimates are employed when available so that it does not slow down the tracker. In our current implementation it provides a pose hypothesis at 20 Hz.

III. EXPERIMENTS

See our previous work for an extensive comparative evaluation of the core single object pose tracking and detection component [4]. In the remainder we only evaluate the extensions presented in this work.

A. Sequences

We have evaluated the proposed method on two video sequences, one involving visual servoing and limited camera motion, and the other involving extensive camera motion. Both sequences are shown in their entirety, together with the tracking results, in the *Supplemental Material* video.

1) Visual Servoing: The first sequence corresponds to the scenario depicted in Fig. 1B with active visual servoing. There are two freely moving objects (the robot base and a coffee mug) and one object (the bottle) is firmly grasped by the robot hand. It is considered rigidly attached to the robot hand, as explained in Section II-C.3. Although the sequence contains only limited camera motion, the mug and robot exhibit fast motion over a large area. The hand and grasped object occlude each other strongly. There are many instances where the grasped object is difficult to see due to its distance from and pose with respect to the camera (Fig. 4A). The sequence also contains many frames where the objects are very close to the camera and thus outside the range of the Kinect depth camera (Fig. 4B).

2) Camera Motion: In the second sequence, the target object has been omitted and no visual servoing occurs. Instead, the grasped object is manipulated faster and in more complex ways. The camera undergoes high speed motion over a large area since the recording was made while walking around the robot. The hand is also further from the camera (Fig. 4C) and there is an instance where it is completely occluded by the robot arm (Fig. 4D).

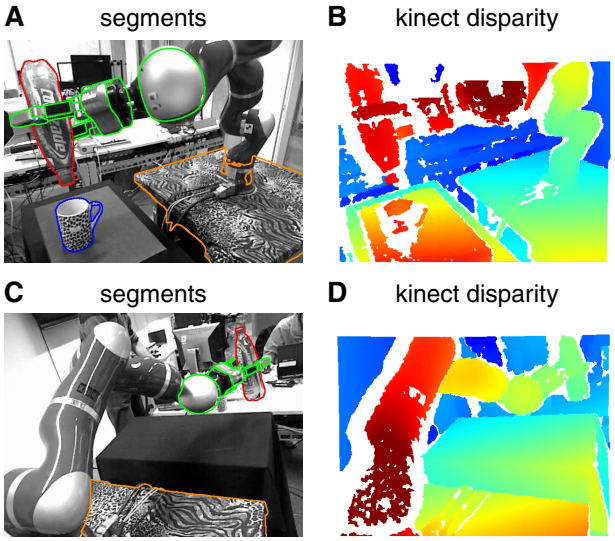

Fig. 3. Frames used to initialize tracking in the *'visual servoing'* (A,B) and *'camera motion'* sequences (C,D).

B. Tracker Initialization

For simplicity we only consider pose tracking here. The SIFT-based pose detector is used at the start of each sequence only. In this way the results are not obscured by the detector's ability to recover from tracking errors. The interaction between tracker and detector has been extensively evaluated in [4]. The tracker is initialized on a frame where sufficient information is available to detect the objects and align the hand with the grasped object. Figure 3 shows these initialization frames. The segments in Fig. 3A,C correspond to the result of initialization. The detector was used to initialize the pose of the bottle, mug, and table. A fixed initial relative pose between base and robot was used to initialize the robot pose. Next, AR flow and depth were used to refine the objects' poses, and depth was used to refine the hand pose (using the green areas in Fig. 3A,C only). The relative pose between hand and grasped object was then fixed for the remainder of the sequence (Section II-C.3).

C. Results

The proposed method successfully tracks all the objects-of-interest over the course of the entire sequences without requiring detector re-initialization. This amounts to 2800 consecutive frames in the *'visual servoing'* sequence and 1400 consecutive frames in the *'camera motion'* sequence. These results are shown in their entirety in the *Supplemental Material* video. Some indicative results are shown in Fig. 4. All the actively tracked segments precisely match the scene objects and robot hand. The front-view-rendering in rows (A) and (B) give an indication of the precision obtained while servoing. The misalignment between robot base and arm is the result of not actively tracking the arm. Recall that the effects of the various noise sources manifest themselves as this misalignment. The relative pose between base and arm is set at initialization time but is then allowed to change according to the soft-constraint mechanism. The evolution of this time-varying base offset over the course of the *'visual servoing'* sequence is shown in Fig. 5A. The changes in this offset correlate with the distance between the hand and the camera, and are mainly due to the inaccurate camera calibration.

D. Comparative Evaluation

We next compare the proposed method to a number of alternatives. Since ground-truth is not available for the complex real-world sequences used, we instead evaluate the results in terms of how well the disparity obtained by synthesizing the scenes according to the tracker state matches the disparity obtained from the Kinect sensor. We allow for a maximal error of one pixel and compute the proportion of explained measurements within the image region occupied by the combined hand/grasped object segment. The results are summarized in Fig. 6 and discussed next.

1) Fixed Robot–Base Connection: This approach does not track the hand and grasped object, but instead relies on a fixed relative pose between robot base and arm. In this way, the poses can be derived directly from the estimated base pose, encoder values and forward kinematics. The connection between base and robot is optimized here. Using the successful pose trace estimated with the proposed method, we estimated the fixed relative pose that minimizes the least-squares 3D distance error between corresponding vertices at the hand and bottle object over the course of the entire two sequences. The results obtained with this approach are depicted with the red dash-dotted lines in Fig. 6 and are significantly worse than the other two approaches.

2) Tracking Without Base Prior: In this approach we use the hand and grasped object features directly to compute the pose update, without including the prior as in (12). This approach works very well as can be seen in Fig. 6A (green dashed curve). It slightly outperforms the proposed method on many occasions, which is to be expected since this method is strictly data-driven and does not need to satisfy the prior. It is however significantly outperformed around frames 1200 and 2700 in (A) due to an ambiguous situation for depth-only ICP (near-planarity). An example frame (1203) is shown in Fig. 7A. Note how the proposed method does not suffer from this problem (Fig. 7B). The robot–base connection also moves very erratically without prior (Fig. 5B) which can be problematic for control. Note how the tracking errors around frames 1200 and 2700 in (A) also manifest themselves in the base offset. The problems with this method increase with more severe camera motion and occlusions, and it fails completely on the *'camera motion'* sequence (Fig. 6B) around frame 320 (corresponding to the situation in Fig. 4D).

3) Tracking Objects Only, Ignoring Manipulator: This method can not exploit the hand pose to improve the grasped object pose and ignores their mutual occlusions. We do not include it in the quantitative comparison since tracking is lost quickly. For example, in the scenario depicted in Fig. 4A there are not enough measurements at the bottle for reliable pose tracking.

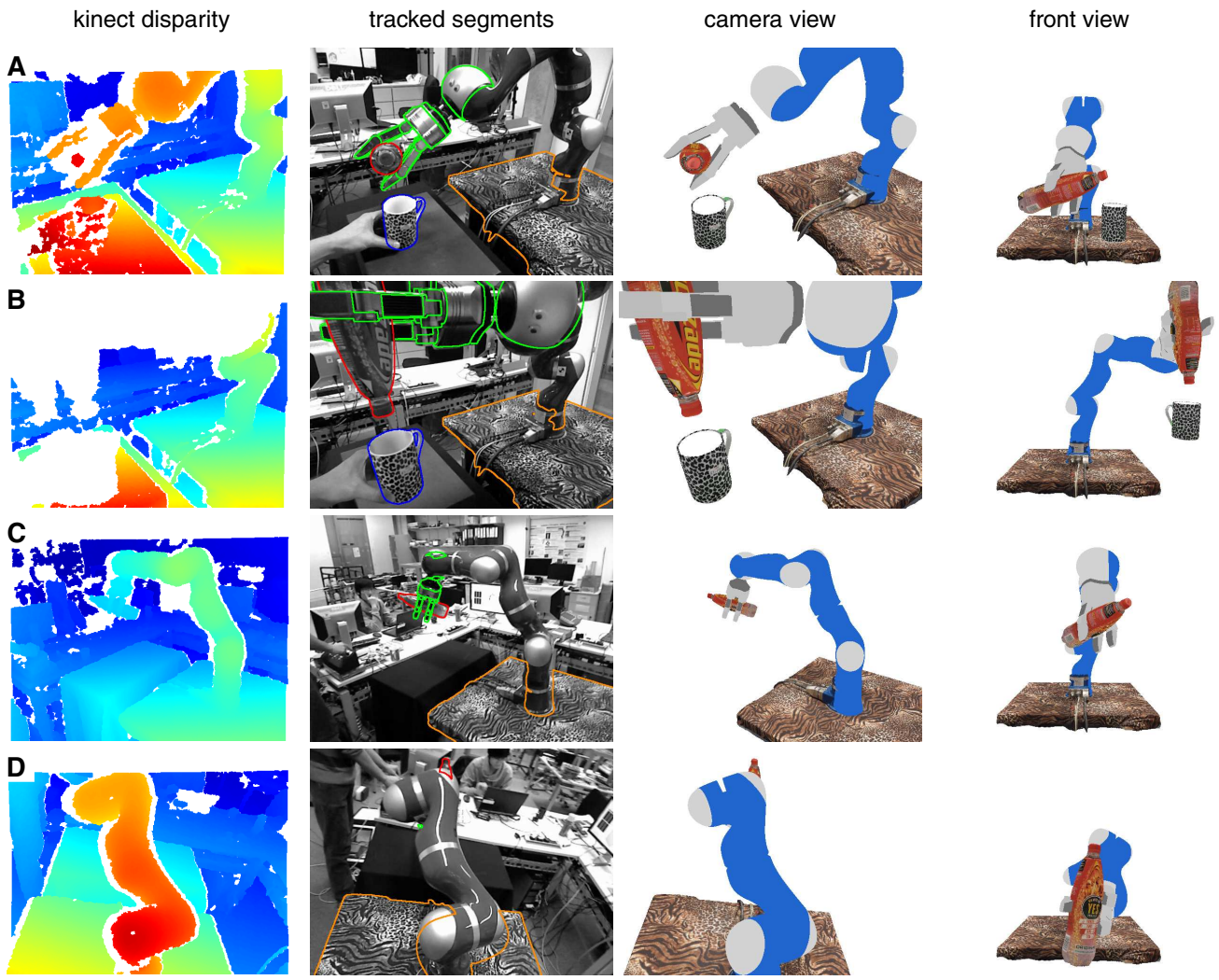

Fig. 4. Tracking results obtained with the proposed method. Only the actively tracked segments are highlighted.

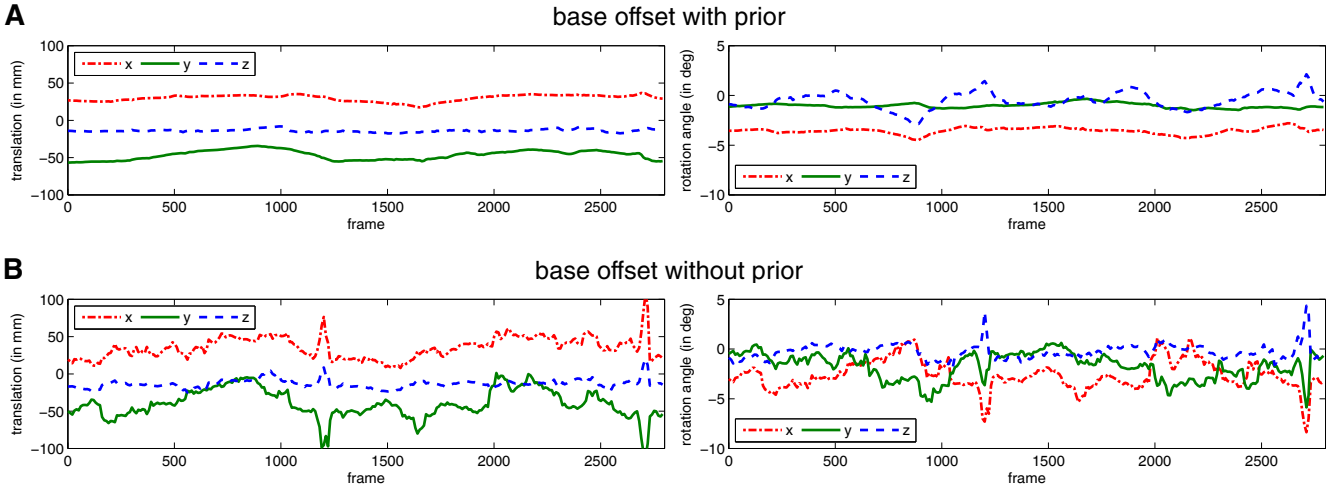

Fig. 5. Evolution of the position (left) and orientation (right) of the base offset with respect to the initial relative pose between robot base and arm when (A) including the base pose update as a prior in the tracker, and (B) when using only hand and grasped object to establish the robot pose. We use a left-handed coordinate system with X-axis horizontal, Y-axis vertical, and Z-axis in-depth.

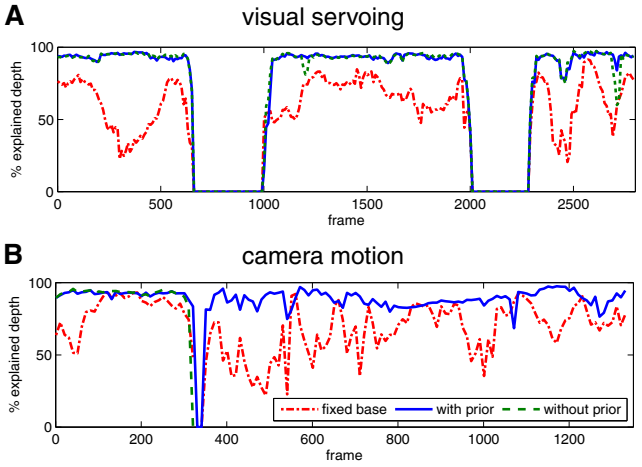

Fig. 6. Proportion of the valid Kinect disparity measurements (within the hand + grasped object region) that match the disparity as predicted by the current tracker state for the fixed base (dash-dotted red), proposed (solid blue), and prior-less method (dashed green). The hand is too close to the depth sensor around frames 650–1000 and 2000–2300 in (A).

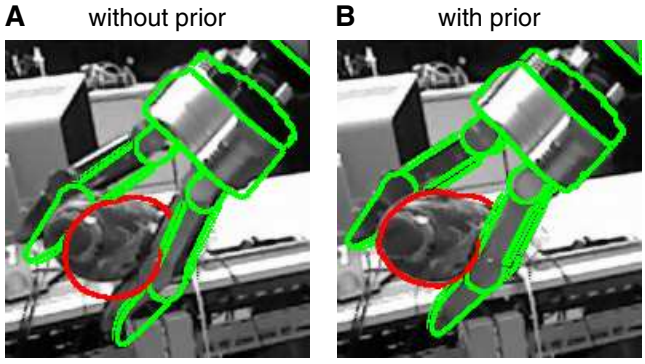

Fig. 7. Failure of tracking without base prior (A) due to ambiguity of depth-only ICP (frame 1203 in Fig. 6A). Relying on the base update resolves the ambiguity (B).

4) Tracking Manipulator Only, Ignoring Objects: We also do not include quantitative results here since tracking fails on many occasions, such as in the absence of Kinect depth measurements (Fig. 4B).

IV. Conclusion and Future Work

We have presented a real-time system for joint multi-object and manipulator detection and tracking in complex, dynamic scenarios involving imprecise calibration. The method achieves a high degree of accuracy and reliability by constantly updating a detailed 3D scene representation on the basis of large amounts of dense visual data. Although it could easily be incorporated, a temporal filtering component is not strictly required.

Future work will explore softening the connection between hand and grasped object, to allow for vision-based grasp stability assessment and enable more complex manipulations. Recently, our approach has also been shown suitable for explicitly tracking articulated objects [30], which can reduce the dependence on precise proprioception.

References

[1] M. Dogar, K. Hsiao, M. Ciocarlie, and S. Srinivasa, "Physics-based grasp planning through clutter," in *R:SS*, July 2012.

[2] L. Righetti, M. Kalakrishnan, P. Pastor, J. Binney, J. Kelly, R. Voorhies, G. Sukhatme, and S. Schaal, "An autonomous manipulation system based on force control and optimization," *Autonomous Robots*, vol. 36, no. 1-2, pp. 11–30, 2014.

[3] V. Micelli, K. Strabala, and S. Srinivasa, "Perception and control challenges for effective human-robot handoffs," in *R:SS*, June 2011.

[4] K. Pauwels, L. Rubio, J. Diaz Alonso, and E. Ros, "Real-time model-based rigid object pose estimation and tracking combining dense and sparse visual cues," in *CVPR*, June 2013, pp. 2347–2354.

[5] A. Collet Romea, M. Martinez Torres, and S. Srinivasa, "The MOPED framework: Object recognition and pose estimation for manipulation," *R:SS*, vol. 30, no. 10, pp. 1284 – 1306, 2011.

[6] Q. Hao, R. Cai, Z. Li, L. Zhang, Y. Pang, F. Wu, and Y. Rui, "Efficient 2D-to-3D correspondence filtering for scalable 3D object recognition," in *CVPR*, 2013.

[7] A. Petit, E. Marchand, and K. Kanani, "A robust model-based tracker combining geometrical and color edge information," in *IROS*, 2013.

[8] C. Choi and H. I. Christensen, "RGB-D object tracking: A particle filter approach on GPU," in *IROS*, 2013, pp. 1084–1091.

[9] C. Teuliere and E. Marchand, "Direct 3D servoing using dense depth maps," in *IROS*, 2012, pp. 1741–1746.

[10] M. Wuthrich, P. Pastor, M. Kalakrishnan, J. Bohg, and S. Schaal, "Probabilistic object tracking using a range camera," in *IROS*, Nov 2013, pp. 3195–3202.

[11] P. Hebert, N. Hudson, J. Ma, and J. Burdick, "Dual arm estimation for coordinated bimanual manipulation," in *ICRA*, 2013.

[12] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, "Full DOF tracking of a hand interacting with an object by modeling occlusions and physical constraints," in *ICCV*, 2011, pp. 2088–2095.

[13] M. Krainin, P. Henry, X. Ren, and D. Fox, "Manipulator and object tracking for in-hand 3D object modeling," *IJRR*, vol. 30, no. 11, pp. 1311–1327, 2011.

[14] A. Teichman, S. Miller, and S. Thrun, "Unsupervised intrinsic calibration of depth sensors via SLAM," in *R:SS*, 2013.

[15] P. Güler, Y. Bekiroglu, K. Pauwels, and D. Kragic, "What's in the container? Classifying object contents from vision and touch," in *IROS*, Chicago, Illinois, 2014.

[16] Z. Jia, A. Gallagher, A. Saxena, and T. Chen, "3D-based reasoning with blocks, support, and stability," in *CVPR*, 2013.

[17] A. Pieropan, G. Salvi, K. Pauwels, and H. Kjellström, "Audio-visual classification and detection of human manipulation actions," in *IROS*, Chicago, Illinois, 2014.

[18] D. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, no. 2, pp. 91–110, 2004.

[19] Autodesk, "123d catch," http://www.123dapp.com/catch/.

[20] K. Pauwels, M. Tomasi, J. Diaz Alonso, E. Ros, and M. Van Hulle, "A comparison of FPGA and GPU for real-time phase-based optical flow, stereo, and local image features," *IEEE Transactions on Computers*, vol. 61, no. 7, pp. 999–1012, July 2012.

[21] C. Wu, "SiftGPU: A GPU implementation of scale invariant feature transform (SIFT)," http://cs.unc.edu/ ccwu/siftgpu, 2007.

[22] G. Blais and M. Levine, "Registering multiview range data to create 3D computer objects," *IEEE PAMI*, vol. 17, no. 8, pp. 820–824, 1995.

[23] C. Yang and G. Medioni, "Object modelling by registration of multiple range images," *Image Vision Comput.*, vol. 10, pp. 145–155, 1992.

[24] H. C. Longuet-Higgins and K. Prazdny, "The interpretation of a moving retinal image," *P. Roy. Soc. B-Biol. Sci.*, pp. 385–397, 1980.

[25] F. Mosteller and J. Tukey, *Data analysis and regression: A second course in statistics*. Mass.: Addison-Wesley Reading, 1977.

[26] H. Theil, "On the use of incomplete prior information in regression analysis," *JASA*, vol. 58, no. 302, pp. 401–414, 1963.

[27] V. Lepetit and P. Fua, "Monocular model-based 3D tracking of rigid objects," *FTCGV*, vol. 1, pp. 1–89, 2005.

[28] S. Hutchinson, G. Hager, and P. Corke, "A tutorial on visual servo control," *IEEE T. Robotic. Autom.*, vol. 12, no. 5, pp. 651–670, 1996.

[29] J. Hoberock and N. Bell, "Thrust: A parallel template library," 2010, version 1.7.0. [Online]. Available: http://thrust.github.io/

[30] K. Pauwels, L. Rubio, and E. Ros, "Real-time model-based articulated object pose detection and tracking with variable rigidity constraints," in *CVPR*, Columbus, Ohio, 2014.