ICT Call 7
ROBOHOW.COG
FP7-ICT-288533

Deliverable D4.3:

Tracking and State Estimation of Manipulated Objects

February 2, 2015

Project acronym:	ROBOHOW.COG
Project full title:	Web-enabled and Experience-based Cognitive Robots that Learn Complex Everyday Manipulation Tasks
Work Package:	WP 4
Document number:	D4.3
Document title:	Tracking and State Estimation of Manipulated Objects
Version:	1.0
Delivery date:	January 31st, 2015
Nature:	Report
Dissemination level:	Public
Authors:	Christian Smith (KTH) Yiannis Karayiannidis (KTH) Yasemin Bekiroğlu (KTH) Karl Pauwels (KTH) Danica Kragic (KTH)

The research leading to these results has received funding from the European Union Seventh Framework Programme FP7/2007-2013 under grant agreement n^{o}288533 ROBOHOW.COG.

Contents

Summary

The contribution in this deliverable is on tracking and state estimation of manipulated objects, as detailed in Tasks T4.2, T4.3, T4.4, and to some extent T4.5. The ability to manipulate objects depends on knowledge of the objects' extrinsic properties such as geometry, pose, and how they interact with other objects and the environment, as well as intrinsic properties such as object boundaries, and how they react to applied forces.
We explore different perceptual methods exploiting different sensory modalities, to gain understanding of these intrinsic and extrinsic object properties. Algorithms have been developed both for detecting and tracking these object properties.

Chapter 1
Published Results

The results for this deliverable have been accepted for publication in peer-reviewed venues. This section contains a short description of the contributions, and references to the published reports, which are appended to this document.

1.1 Tracking of pose and interaction forces

As detailed in tasks T4.2 and T4.3, we study perception of the pose of different objects, to enable the robot to sense and understand its environment. By using known object models, and exploiting the inherent parallelism in the problem, we show a method to simultaneously track the pose of more than a hundred different objects in real time, using 3D vision. Also, as detailed in tasks T4.3 and T4.4, we show how to track contact interaction between an unmodelled tool held by a robot and the environment. By using force and torque measurements, we demonstrate tracking of contact points and surface normals, even for visually occluded cases. The results are published in [1, 2].

1.2 Understanding object properties

In the context of tasks T4.2 and T4.4, we study the problem of disambiguating between objects and environment, and between separate objects. By using a probabilistic learning framework, we show how to enable a robot to determine object separation by observing the effects of touching the object. We also show how the robot can autonomously learn the optimal actions for achieving efficient disambiguation between possible hypotheses. The results are published in [3].
In the context of task T4.4 and T4.5, we explore different methods to determine container contents by observing their deformation under applied grasps. We show how the different modalities of touch and vision can be used to infer content information, and that similar performance can be achieved with either modality. The results are published in [4]

Bibliography

[1] K. Pauwels, L. Rubio, and E. Ros, "Real-time pose detection and tracking of hundreds of objects," *IEEE Transactions on Circuits and Systems for Video Technology*, 2015, (In submission).

[2] Y. Karayiannidis, C. Smith, F. Vina, and D. Kragic, "Online contact point estimation for uncalibrated tool use," in *IEEE International Conference on Robotics and Automation*, 2014, pp. 2488–2493.

[3] M. Björkman and Y. Bekiroğlu, "Learning to disambiguate object hypotheses through self-exploration," in *IEEE-RAS International Conference on Humanoid Robots*, Madrid Spain, 2014, pp. 560–565.

[4] P. Guler, Y. Bekiroglu, X. Gratal, K. Pauwels, and D. Kragic, "What's in the container? classifying object contents from vision and touch," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, Sept 2014, pp. 3961–3968.

Real-time Pose Detection and Tracking of Hundreds of Objects

Karl Pauwels, Leonardo Rubio, and Eduardo Ros

***Abstract*—We propose a novel model-based method for tracking the six-degrees-of-freedom (6DOF) pose of a very large number of arbitrarily shaped rigid objects in real-time. By combining dense motion and depth cues with sparse keypoint correspondences, and by feeding back information from the modeled scene to the cue extraction process, the method is both highly accurate and robust to noise and occlusions. A tight integration of the graphical and computational capability of graphics processing units (GPUs) allows the method to simultaneously track hundreds of objects in real-time. We achieve pose updates at framerates around 40 Hz when using 500,000 data samples to track 150 objects. We introduce a synthetic benchmark dataset with varying objects, background motion, noise and occlusions that enables the evaluation of stereo-vision-based pose estimators in complex scenarios. Using this dataset and a novel evaluation methodology, we show that the proposed method greatly outperforms state-of-the-art methods. Finally, we demonstrate excellent performance on challenging real-world sequences involving multiple objects being manipulated.**

***Index Terms*—model-based object pose estimation, optical flow, stereo, real time, graphics processing unit (GPU), benchmarking.**

I. Introduction

ESTIMATING and tracking the 6DOF (three translation and three rotation) pose of multiple rigid objects is critical for robotic applications involving object grasping and manipulation, and also for inspection tasks, activity interpretation, or even path planning. In many situations, models of the objects of interest can be obtained quickly and with relative ease, either off-line [1], or on-line in an exploratory stage [2], [3].

A. Related Work

Since real-time pose detection and tracking is such an important ability of robotic systems, a wide variety of methods have been proposed in the past. We only provide a brief overview of the major classes of methods here. Many more examples exist for each class. An important distinction exists between pose *detection* and pose *tracking* methods. Unlike pose tracking methods, pose detection methods do not exploit temporal information. Our work mostly focuses on pose tracking although we rely on pose detection to (re-)initialize tracking and thus also consider the interaction between both. Therefore we also include a brief overview of these methods here.

Pose detection approaches can recover the pose from a single image, without requiring an initial estimate. Many such methods rely on sparse keypoints and descriptors to match 2D image features to 3D model points [4], [5]. It has recently been shown how these methods can scale to a very large number of objects [6], [7]. Template methods on the other hand match images to a set of stored templates covering different views of an object [8]. These templates can contain various features and recently also RGB-D-input has been exploited [9]. When complete 3D object models are available, the estimates provided by such methods can also be subsequently refined using standard depth-based iterative closest point (ICP) procedures [10], [11]. Template-based methods are related to learning-based methods for multi-view object class detection, but the latter typically provide only coarse viewpoint estimates [12].

Pose tracking methods refine a pose estimate, usually obtained at the previous time step, based on the measurements obtained at the current time. The most efficient tracking methods to date match expected to observed edges by projecting a 3D wireframe model in the image [13]. Many extensions have been proposed that exploit also texture information [14], [15], optical flow [16] or particle filtering [5], [17], [18] in order to reduce sensitivity to background clutter and noise. Robustness has also been improved by considering multiple hypotheses [19]. A different class of approaches relies on level-set methods to maximize the discrimination between statistical foreground and background appearance models [20]. These methods can include additional cues such as optical flow and sparse keypoints, but at a large computational cost [21]. Recently, also methods that combine both edge- and region-based information have been proposed [22].

Some of these methods also combine tracking with detection to enable automatic initialization or recovery in case of severe occlusions [4], [18], [23], but the multiple object case is not considered explicitly as this requires accounting for inter-object occlusion and having scalable computational requirements.

Most of the above-mentioned tracking approaches can exploit multi-view information or blob-based stereo triangulation [24], but dense depth information is rarely used. Particle filtering has been used recently for RGB-D object tracking [25] but the dense depth information is only used there to evaluate the particle hypotheses, rather than to compute a pose update, as done in ICP-based methods, such as the one proposed here. An ICP approach [10] can be used provided the object has sufficiently salient shape features. This has been

K. Pauwels is with the Computer Vision and Active Perception lab, Royal Institute of Technology (KTH), Stockholm, Sweden, e-mail: kpauwels@kth.se.

L. Rubio is with Fuel 3D Technologies Limited, Oxford, UK, e-mail: leo@fuel-3d.com.

E. Ros is with the Computer Architecture and Technology Department, University of Granada, Spain, e-mail: eros@ugr.es.

applied with great success in related problems such as online scene modeling [2] where the whole scene is considered as a rigid object. Recently, due to the prevalence of cheap depth sensors, depth information is also being applied in other related problems, such as articulated body pose estimation [26] and visual servoing [27].

We show here that incorporating some of these advances in real-time object tracking, and extending them with additional cues, yields great improvements.

B. Novelty

Unlike most model-based methods that exploit only salient parts of the model (keypoints, silhouettes, edges, shape features), we instead aim to retain all of the model's shape and appearance information, and match it to the observed dense visual features. In this way, the useful feature-set is constrained by the dense algorithms at perception time, rather than at model creation time.

The main contributions can be summarized as follows. Firstly, we introduce a model-based 6DOF pose detection and tracking method that combines dense motion and stereo disparity measurements with sparse keypoint features. It exploits feedback from the modeled scene to the cue extraction level and provides an effective measure of reliability. Although dense motion and stereo have been combined before in model-free 6DOF camera pose estimation [28], we show here how the model information can be used to keep the motion and stereo measurements separate while still jointly minimizing their cost in the error function. For the stereo component we use an ICP approach rather than the 'disparity flow' used by [28]. We show that by iteratively re-rendering the scene, we can move from the differential pose updates typically obtained from linearized edge- or motion-based energy functions, to the full discrete pose update. Secondly, the method has been designed specifically for high-performance operation ($\pm$ 40 Hz with 150 objects). To achieve this, every aspect of the algorithms and the system has been developed with GPU acceleration in mind. Both the graphics and computation pipelines of a modern GPU are extensively used and tightly integrated in both the low-level cue extraction and pose tracking stages. Finally, an extensive benchmark dataset and evaluation methodology have been developed and used to show increased performance (in accuracy, robustness, and speed) as compared to state-of-the-art methods.

Parts of this work have been presented earlier [29]. We introduce here an extension to the multi-object case, together with more method and implementation details, a more detailed evaluation, and additional real-world results.

II. Proposed Method

A concise overview of the method is shown in Fig. 1. Different visual cues are extracted from a stereo video stream and combined with model information (known a priori) to estimate the 6DOF pose of multiple objects. In turn, scene information (related to the joint appearance and shape of the tracked models) is fed back to facilitate the cue extraction itself.

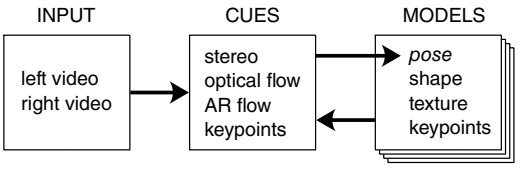


Fig. 1. Method overview illustrating the different visual cues (AR = augmented reality, see II-B2) and model components. The cues are combined to estimate the objects poses and scene information is fed back to facilitate the cue extraction.

A. Scene Representation

The proposed method supports 3D models of arbitrary shape and appearance of the objects to track. The models' surface geometry and appearance are represented by a triangle mesh and color texture respectively. For a given set of model poses, the color (Fig. 2B), distance to the camera (Fig. 2E), surface normal (Fig. 2F), and object identity (Fig. 2D) can be obtained efficiently at each pixel through OpenGL rendering. As a result also self-occlusions and occlusions between different modeled objects are handled automatically through OpenGL's depth buffer. In a training stage, SIFT (Scale-Invariant Feature Transform) features [30] are extracted from keyframes of rotated versions of each model (30° separation), and mapped onto the model's surface.

B. Visual cues

The dense motion and stereo cues are obtained using coarse-to-fine GPU-accelerated phase-based algorithms [31], [32] (modified as discussed next), and the SIFT features are extracted and matched using a GPU library [33].

1) Model-based dense stereo: Coarse-to-fine stereo algorithms, although highly efficient, support only a limited disparity range and have difficulties detecting fine structures [34]. To overcome these problems we feed the object pose estimates obtained in the previous frame back as a prior in the stereo algorithm. Figure 3A,B show an example real-world stereo image pair of a box being manipulated. Using the box's previous frame pose estimate, stereo priors are generated for the current frame pair by converting OpenGL's Z-buffer to disparities for both the left and right cameras. These disparity values are downsampled (Fig. 3C,D) and introduced at the lowest scale used by the stereo algorithm. Stereo disparity is then computed twice, once with respect to the left and once with respect to the right image (essentially by swapping left and right images), each time processing the entire pyramid. This results in two separate disparity maps, one from the left to the right image, δ_L^R, and one from the right to the left image, δ_R^L. A left/right consistency check is then used to remove unreliable estimates. Concretely, δ_L^R is used to find the corresponding pixel in δ_R^L and the following error is evaluated in each pixel:

$$e_\delta(\mathbf{x}) = |\delta_L^R(\mathbf{x}) + \delta_R^L\big(\mathbf{x} + \delta_L^R(\mathbf{x})\big)| \ . \tag{1}$$

Disparity estimates with an error exceeding 0.5 pixel are then considered unreliable (see [34] for further details). The reliable estimates for the current frame obtained with and without

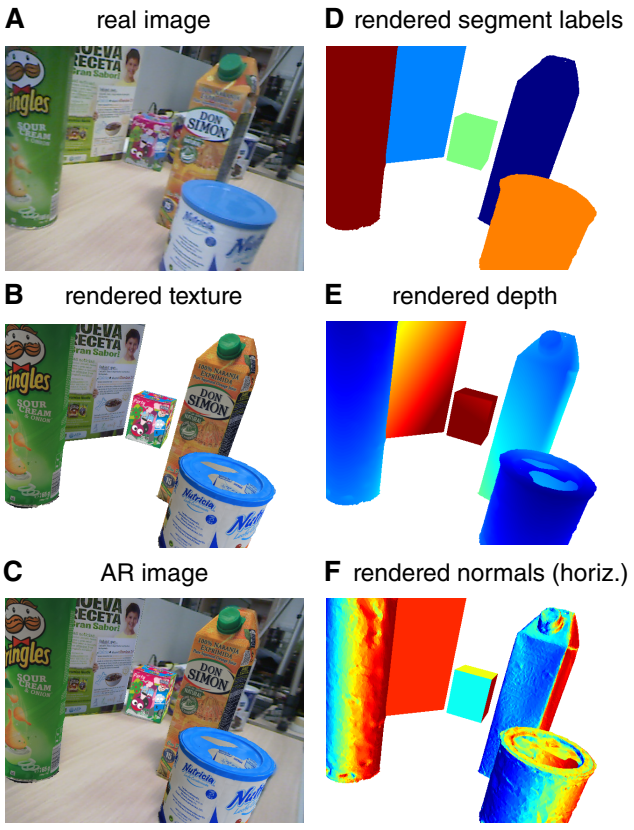


Fig. 2. (A) Original input image, (B) color image obtained by rendering the textured models according to the current pose estimates, (C) AR image obtained by merging A and B (see II-B2), (D) segment (object identity) labels accounting for occlusions, (E) depth-buffer, and (F) horizontal component of the normals. Note the model errors in the lid of the *Nutricia* object.

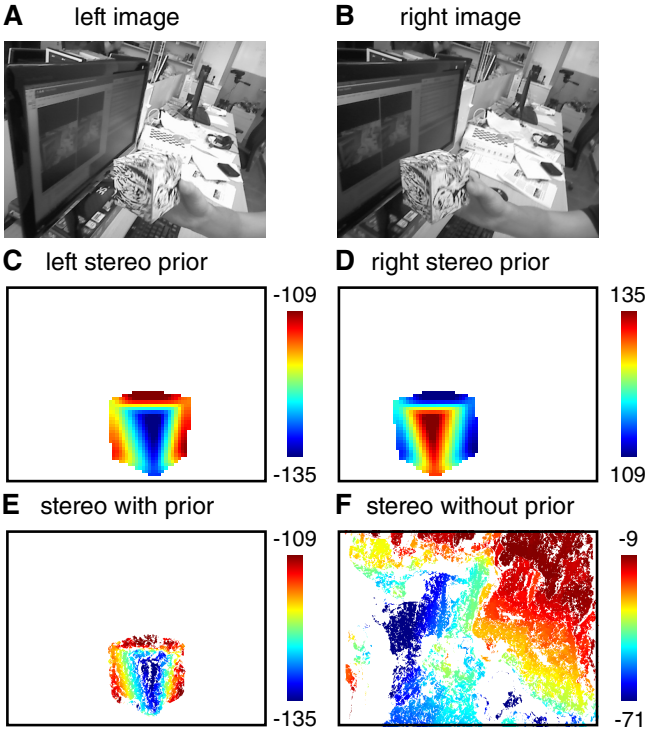


Fig. 3. (A) Left and (B) right input images and low-resolution (C) left and (D) right stereo priors generated using the previous frame pose estimate of the manipulated box. (E) Stereo obtained using four scales and initializing with the prior, and (F) stereo obtained using six scales and without using the prior.

the priors are shown in panels E and F respectively. For the prior-less algorithm, we used the maximal amount of scales (six) that allow us to fit the filter kernels (11×11) at the lowest resolution. Due to the large range of disparities in this particular scene, without the prior, the focus is on the background rather than on the object of interest (Fig. 3F). Note that the prior corresponds to the *previous* frame pose estimate. With prior, the stereo algorithm thus only has to correct the disparity that results from the pose difference between the current and previous frame. This is much smaller than the absolute disparity and is instead of the order of the correspondences obtained in an optical flow scenario. In Fig. 3E we show that four scales are sufficient to obtain detailed estimates. In certain scenarios (*e.g.* complete loss of tracking) it is possible that a completely wrong prior is generated. In this case the pose selection mechanism (see II-C2) will signal this and either switch to the sparse detector's pose estimate, or mark both the tracker and detector estimates unreliable.

2) Dense motion cues: The optical flow algorithm integrates the temporal phase gradient across different orientations and also uses a coarse-to-fine scheme to increase the dynamic range [35]. Unlike [35], we estimate the temporal phase gradient using two instead of five frames, more specifically the frames I_t and I_{t+1}, at times t and $t+1$. This results in noisier estimates, but reduces the latency. This trade-off can be made depending on the expected dynamics of the considered scenario. In a similar fashion as in the stereo algorithm, a simple consistency check is used to discard unreliable estimates (in this case a forward/backward as opposed to left/right consistency check in the stereo case). A prior is not required here since displacements in the motion scenario are much smaller than in the stereo scenario.

Figure 4C contains the (subsampled and scaled) optical flow vectors from Fig. 4A to the next image in a complex real-world scenario with both object and camera motion. Besides the optical flow, we also extract a second type of motion which we refer to as augmented reality (AR) flow that incorporates scene-feedback. The models' textures are rendered at their current pose estimates and overlaid on I_t, resulting in an 'augmented image' $\hat{I}_t$. An example is shown in Fig. 4B using a single object (see Fig. 2C for a multi-object example). The motion is then computed from $\hat{I}_t$ to the next real image (I_{t+1}), and shown in Fig. 4D for the example presented here. Because of the erroneous pose estimate (deliberately large in this case to better illustrate the concept) this motion is quite different from the optical flow. It allows the tracker to recover from such errors by effectively countering the drift that results from tracking based on optical flow alone.

Note that the appearance (especially the brightness) of the model texture will always be somewhat different from the real

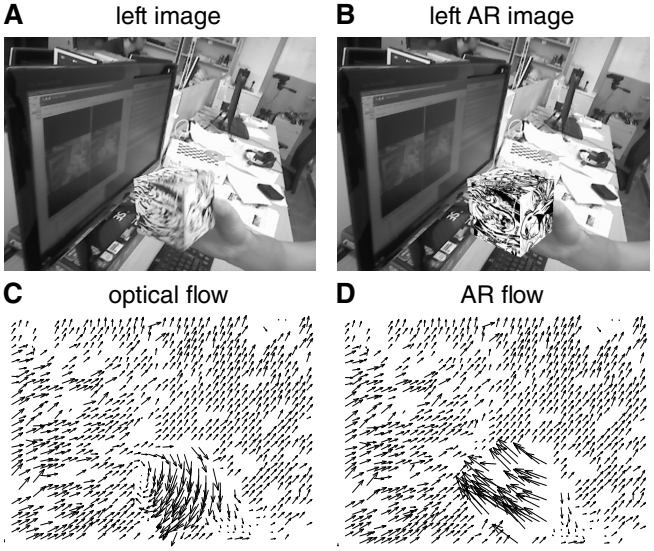


Fig. 4. (A) Left image and (B) left image with rendered model (at incorrect pose) superimposed. (C) Real optical flow from real image in A to the next real image. (D) AR flow from AR image in B to the next real image. Both flow fields are subsampled (15×) and scaled (5×).

scene. It is therefore critical that the optical flow algorithm is largely invariant to this. Since the phase-based algorithm used here relies on a *phase*-constancy rather than a *brightness*-constancy assumption, it is inherently invariant to intensity differences. Consider the more extreme (but not unrealistic) example in Fig. 5. Here, the images were recorded with short exposure, resulting in large intensity differences between the real scene and the model texture. We compare the optical flow of our method to a standard pyramidal Lucas & Kanade algorithm (L&K) [36]. Note that the optical flow (from 5A to 5B) is similar for both methods (although L&K is more noisy). L&K however fails completely on the AR flow (from 5C to 5B) where the large intensity differences result in large flow vectors (5E). The phase-based algorithm is completely insensitive to this and correctly perceives the rotation of the AR image (5G). The same default parameters were used in (D,E) and (F,G).

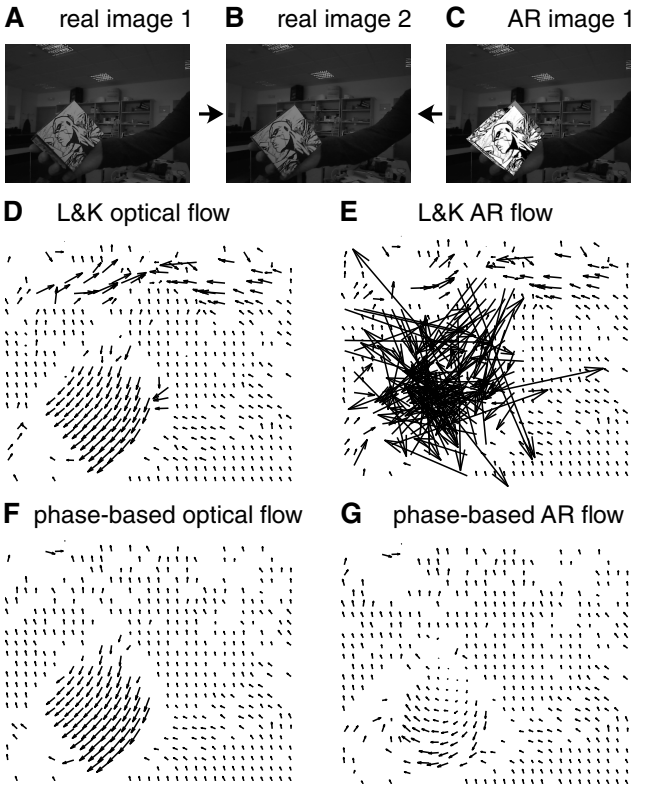


Fig. 5. Importance of intensity invariance for AR flow computation. (A) and (B) are consecutive images obtained with short exposure. The AR image (C) is composed by superimposing the model (with much brighter texture) at an incorrect pose on image (A). Optical flow from A to B computed with (D) pyramidal Lucas & Kanade and (F) our phase-based algorithm. AR flow from C to B with (E) pyramidal Lucas & Kanade and (G) phase-based. All flow fields are subsampled (20×) but unscaled.

C. Pose detection and Tracking

The proposed tracking method incorporates the differential rigid motion constraint into a fast variant of the ICP algorithm [10] to allow all the dense cues to simultaneously and robustly minimize the pose errors. A selection procedure (II-C2) is used to re-initialize the tracker with estimates obtained from a sparse keypoint-based pose detection approach, if required.

1) Dense Pose Tracking: In the following we will consider the single object case only. Multiple objects can be treated independently and in parallel since we are not considering interactions between the objects. Note that occlusions are handled automatically at the rendering stage. In Section III we will discuss in more detail how this parallel optimization is performed.

Our aim is to recover the rigid rotation and translation that best explains the dense visual cues and transforms each model point $\mathbf{m} = [m_x,\, m_y,\, m_z]^{\mathrm{T}}$ at time t into point $\mathbf{m}'$ at time $t+1$:

$$\mathbf{m}' = \mathrm{R}\,\mathbf{m} + \mathbf{t}\ , \tag{2}$$

with R the rotation matrix and $\mathbf{t} = [t_x,\, t_y,\, t_z]^{\mathrm{T}}$ the translation vector. The rotation matrix can be simplified using a small angle approximation:

$$\mathbf{m}' \approx (\mathbf{1} + [\boldsymbol{\omega}]_\times)\,\mathbf{m} + \mathbf{t}\ , \tag{3}$$

with $\boldsymbol{\omega} = [\omega_x,\, \omega_y,\, \omega_z]^{\mathrm{T}}$ representing the rotation axis and angle. Each stereo disparity measurement, d', at time $t+1$ can be used to reconstruct an approximation, $\mathbf{s}'$, to $\mathbf{m}'$. Since we are using a rectified camera configuration, this is quite straightforward:

$$\mathbf{s}' = \begin{bmatrix} s'_x \\ s'_y \\ s'_z \end{bmatrix} = \begin{bmatrix} x\, s'_z/f \\ y\, s'_z/f \\ -f\, b/d' \end{bmatrix}\ , \tag{4}$$

with $\mathbf{x} = [x, y]^{\mathrm{T}}$ the pixel coordinates (with nodal point as origin), f the focal length, and b the baseline of the stereo rig. To use this reconstruction in (2), the model point $\mathbf{m}$ corresponding to $\mathbf{s}'$ needs to be found. We deliberately avoid using the motion cues to facilitate this correspondence

search (as done in many ICP extensions) since this requires both valid stereo and valid motion measurements at the same pixel. Instead we use the efficient projective data association algorithm [37] that corresponds stereo measurements to model points that project to the same pixel. These correspondences can be obtained instantly, but they are less accurate. Therefore, a *point-to-plane* as opposed to a *point-to-point* distance needs to be minimized [38], which is obtained by projecting the error on $\mathbf{n} = [n_x,\, n_y,\, n_z]^{\mathrm{T}}$, the model normal vector in $\mathbf{m}$:

$$e = \Big[(\mathrm{R}\,\mathbf{m} + \mathbf{t} - \mathbf{s}') \cdot \mathbf{n}\Big]^2 \ . \tag{5}$$

Note that the normal, obtained by rendering the scene through OpenGL, is already in the camera frame (Fig. 2F). This error expresses the distance from the reconstructed points to the plane tangent to the model. By linearizing this measure as in (3) and summing over all correspondences $\{\mathbf{m}_i, \mathbf{s}'_i\}$, we arrive at a strictly shape-based error measure that is linear in the parameters of interest $\mathbf{t}$ and $\boldsymbol{\omega}$:

$$e_S(\mathbf{t}, \boldsymbol{\omega}) = \sum_i \left(\Big[(\mathbf{1} + [\boldsymbol{\omega}]_\times)\, \mathbf{m}_i + \mathbf{t} - \mathbf{s}'_i \Big] \cdot \mathbf{n}_i \right)^2 \ . \tag{6}$$

We next discuss the incorporation of dense motion estimates into the pose tracking procedure. By rearranging (3), we obtain the following:

$$\mathbf{m}' - \mathbf{m} \approx \mathbf{t} + \boldsymbol{\omega} \times \mathbf{m} \ . \tag{7}$$

Note that this looks very similar to the differential motion equation from classical kinematics that expresses the 3D motion of a point, $\dot{\mathbf{m}}$, in terms of its 3D translational and rotational velocity:

$$\dot{\mathbf{m}} = \mathbf{t} + \boldsymbol{\omega} \times \mathbf{m} \ . \tag{8}$$

We deliberately retain the $(\mathbf{t}, \boldsymbol{\omega})$ notation since the displacements in (3) are expressed in the same time unit. This can now be used with the optical flow to provide additional constraints on the rigid motion. Unlike in the stereo case (4), $\dot{\mathbf{m}}$ cannot be reconstructed from the optical flow and instead (8) needs to be enforced in the image domain. After projecting the model shape:

$$\mathbf{x} = f \begin{bmatrix} m_x/m_z \\ m_y/m_z \end{bmatrix} \ , \tag{9}$$

the expected pixel motion $\dot{\mathbf{x}} = [\dot{x}, \dot{y}]^{\mathrm{T}}$ becomes:

$$\dot{\mathbf{x}} = \frac{\delta \mathbf{x}}{\delta t} = \frac{f}{m_z^2} \begin{bmatrix} \dot{m}_x m_z - m_x \dot{m}_z \\ \dot{m}_y m_z - m_y \dot{m}_z \end{bmatrix} \ . \tag{10}$$

Combining (10) with (8) results in the familiar equations [39]:

$$\dot{x} = \frac{(f\, t_x - x\, t_z)}{m_z} - \frac{x\, y}{f}\, \omega_x + (f + \frac{x^2}{f})\, \omega_y - y\, \omega_z \ , \tag{11}$$

$$\dot{y} = \frac{(f\, t_y - y\, t_z)}{m_z} - (f + \frac{y^2}{f})\, \omega_x + \frac{x\, y}{f}\, \omega_y + x\, \omega_z \ , \tag{12}$$

which are linear in $\mathbf{t}$ and $\boldsymbol{\omega}$ provided the depth of the point is known. We obtain this depth m_z by rendering the model at the current pose estimate rather than using the stereo measurement (9). This has the advantage of keeping the motion and stereo measurements strictly separate. Since we have two sources of pixel motion, we have two error functions:

$$e_O(\mathbf{t}, \boldsymbol{\omega}) \ = \ \sum_i \|\dot{\mathbf{x}}_i - \mathbf{o}_i\|^2 \ , \tag{13}$$

$$e_A(\mathbf{t}, \boldsymbol{\omega}) \ = \ \sum_i \|\dot{\mathbf{x}}_i - \mathbf{a}_i\|^2 \ , \tag{14}$$

with $\mathbf{o} = [o_x,\, o_y]^{\mathrm{T}}$ and $\mathbf{a} = [a_x,\, a_y]^{\mathrm{T}}$ the observed optical and AR flow respectively.

Both the linearized point-to-plane distance in the stereo case and the differential motion constraint in the optical and AR flow 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:

$$E(\mathbf{t}, \boldsymbol{\omega}) = e_S(\mathbf{t}, \boldsymbol{\omega}) + e_O(\mathbf{t}, \boldsymbol{\omega}) + e_A(\mathbf{t}, \boldsymbol{\omega}) \ . \tag{15}$$

We let each cue contribute equally in this error function. The optimal balance between optical and AR flow depends on model quality and is therefore hard to decide in advance. Since the system as a whole is quite robust due to the multiple cues, this issue is not critical. More important though is the relative weighting of stereo and motion. For this purpose we evaluated different weight settings on the entire benchmarking dataset of Section IV and observed only marginal effects in a range around equal weight. Setting weights to zero does have large effects though, as will be shown in Section V-A.

To increase robustness, an M-estimation scheme is used to gradually reduce the influence of outliers and ultimately remove them from the estimation [40]. In the case of large rotations the linearized constraints used in (15) are only crude approximations, and many of the shape correspondences obtained through projective data association will be wrong. Therefore, the minimization of (15) needs to be iterated a number of times, at each iteration updating the pose, the shape correspondences, and the unexplained part of the optical and AR flow measurements.

At iteration k, an incremental pose update is obtained by minimizing $E(\Delta\mathbf{t}^k, \Delta\boldsymbol{\omega}^k)$, and accumulated into the pose estimate at the previous iteration $k-1$:

$$\mathrm{R}^k \ = \ \Delta\mathrm{R}^k\, \mathrm{R}^{k-1} \ , \tag{16}$$

$$\mathbf{t}^k \ = \ \Delta\mathrm{R}^k\, \mathbf{t}^{k-1} + \Delta\mathbf{t}^k \ , \tag{17}$$

where $\Delta\mathrm{R}^k = e^{[\Delta\boldsymbol{\omega}^k]_\times}$. The model is updated as:

$$\mathbf{m}^k = \mathrm{R}^k\, \mathbf{m} + \mathbf{t}^k \ , \tag{18}$$

and used to obtain the new (projective) shape correspondences and the part of the optical and AR flow explained thus far, $\Delta\mathbf{x}^k = [\Delta x^k, \Delta y^k]^{\mathrm{T}}$:

$$\Delta x^k \ = \ f(m_x^k/m_z^k - m_x/m_z) \ , \tag{19}$$

$$\Delta y^k \ = \ f(m_y^k/m_z^k - m_y/m_z) \ . \tag{20}$$

This explained flow is subtracted from the observed optical and AR flow:

$$\mathbf{o}^k \ = \ \mathbf{o} - \Delta\mathbf{x}^k \ , \tag{21}$$

$$\mathbf{a}^k \ = \ \mathbf{a} - \Delta\mathbf{x}^k \ . \tag{22}$$

The next iteration incremental pose updates are then obtained by minimizing $E(\Delta\mathbf{t}^{k+1}, \Delta\boldsymbol{\omega}^{k+1})$, which operates on $\mathbf{o}^k$, $\mathbf{a}^k$, $\mathbf{m}^k$, and $\mathbf{s}'$. This cycle is repeated a fixed number of times (we use three internal iterations for the M-estimation, and three external iterations). Note that even though the updates in each iteration are estimated from linearized equations, the correct accumulated discrete updates are used in between iterations (18–20).

2) Combined Sparse and Dense Pose Estimation: A RANSAC-based monocular perspective-n-point pose estimator is used to robustly extract the 6DOF object pose on the basis of correspondences between image (2D) and model codebook (3D) SIFT keypoint descriptors [36], [41]. Exhaustive matching is used and therefore, unlike the dense tracking component of the proposed method, this sparse estimator provides a pose estimate that does not depend on the previous frame's estimate.

Due to the nonlinear and multimodal nature of the sparse and dense components, directly merging them using for instance a Kalman filtering framework is not suitable here. Instead we *select* either the sparse or dense estimate based on the effectiveness of its feedback on cue extraction, as measured by the proportion of valid AR flow vectors in the projected (visible, unoccluded) object region. An example of these regions is shown in Fig. 2D. The pose estimate that leads to the largest proportion valid AR flow wins. As mentioned above, a flow vector is considered valid if it passes a simple forward/backward consistency check (II-B2). Note that the accuracy of AR flow is affected by model inaccuracies, but since we are comparing (dense-pose-based) AR flow to (sparse-pose-based) AR flow, both will be equally affected. A very important characteristic of this measure is that, unlike optical-flow-based or stereo-based measures, AR flow *is* affected by occlusions (unless the occlusion is due to another tracked object). When the occluder becomes dominant, the dense tracker will start using the occluder's motion and stereo measurements for the pose update, rather than those of the object-of-interest. At this point the sparse pose update will be selected since its more accurate pose estimate will likely result in a larger AR flow density. This will remain active until the occlusion becomes too large. Then, the small proportion of valid AR flow will signal the problem and the object will be considered lost (occluded). It then needs to be re-detected for tracking to resume. We show in Section V that this simple measure is adequate for selecting and determining the reliability of the pose estimate.

Figure 6A shows how the previous frame dense and sparse poses are used to generate two AR images. From these, the dense and sparse AR flow are extracted and the winner pose is selected based on the proportion of valid AR flow vectors in the object region (Fig. 6B). This pose is then used to update the stereo priors and obtain the model-based stereo at time t. Finally, all cues are combined in the tracker resulting in the dense pose at time t.

This approach can be extended to the multi-object case as follows. Since the sparse estimator does not scale to multiple objects like the tracker, we let it operate on a single object at each time instance. This object is selected probabilistically according to the current reliability of each object's pose.

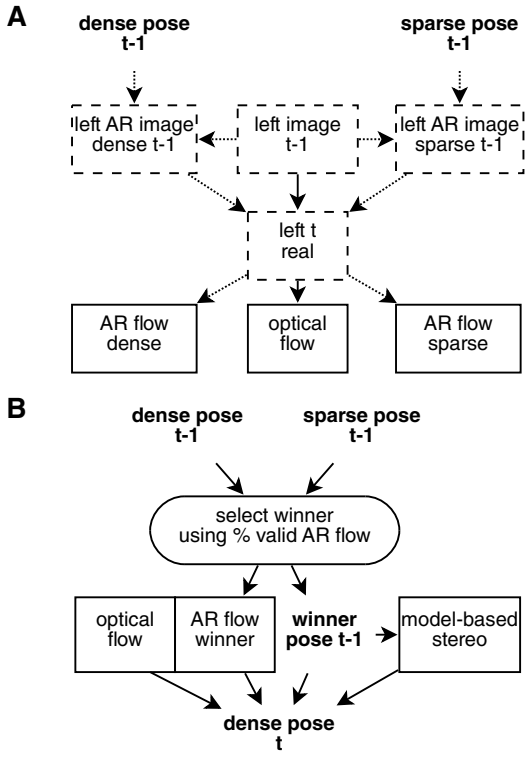


Fig. 6. Combined sparse pose detection and dense pose tracking. Dashed boxes refer to images and solid boxes to extracted cues. (A) Processing sequence for obtaining the three different motion cues. (B) Sequence performed to obtain the dense cues and select the winner pose. Note that model-based stereo only needs to be computed once since it is not used in the evaluation.

Concretely, each object o has the following probability of being selected for a sparse update:

$$p(o) = \frac{1 - r(o)}{\sum_i \big(1 - r(i)\big)} \ , \tag{23}$$

with $r(o)$ the reliability (*i.e.* the proportion valid AR flow). This functions as an attention mechanism, focusing the limited resources on the least reliable estimates. With multiple objects present, it is critical to consider the reliability of the object poses at rendering time. Rendering an unreliable object at an incorrect pose (for example close to the camera) can dramatically disturb the processing of the other objects due to the occlusion handling in the Z-buffering. Therefore, unreliable objects are not considered in the tracking stage. They instead receive a higher probability of becoming the sparse pose estimator's current target.

The sparse/dense selection is performed in a similar way as in the single object scenario (considering the reliability of the updated object) but now also ensuring that the sparse update does not negatively affect the reliability of any other currently tracked object.

III. GPU Implementation

An overview of all the processing steps in the multi-object case is provided in Algorithm 1. The entire system has been

developed using OpenGL and NVIDIA's CUDA framework [42].

Algorithm 1: GPU multiple pose tracking update

```
input : initial poses; low-level visual cues
        (Section III-A)
output: pose updates
for ICP iterations do
    begin pre-process (Section III-B)
        mark valid samples with OpenGL segment labels
        (Fig. 2D)
        radix sort indices of valid samples based on
        segment labels
        if #samples > max_samples then
            subsample valid indices
        gather residual optical and AR flow, disparity,
        and OpenGL depth and normals
    end
    begin ordinary least squares (Section III-C)
        compose normal equations optical and AR flow
        compose normal equations disparity
        solve systems (on CPU)
    end
    for reweighting iterations do
        begin robust least squares iteration
        (Section III-D)
            compute abs. residuals optical and AR flow
            compute abs. residuals disparity
            compute approx. median abs. residuals
            compose weighted normal equations optical
            and AR flow
            compose weighted normal equations disparity
            solve systems (on CPU)
        end
    compute residual optical and AR flow (21,22)
    update poses
    render scene at updated poses
```

A. Low-level Vision

As shown in Fig. 6 the AR flow is computed twice, based on the previous frame's sparse and dense pose estimates. Consequently, four Gabor pyramids need to be computed (left image, right image, sparse left AR image, dense left AR image). A number of rendering steps are also required to create the stereo priors and AR images. A detailed discussion of the GPU implementation of the low-level component of the system (but without the feedback components introduced here) can be found in [31].

B. Pre-processing

The pre-processing component aims to re-organize the pixel-based low-level vision and model-based cues to enable efficient processing in subsequent stages. Concretely this involves assigning all valid measurements (residual optical and AR flow, disparity, Z-buffer and normals) to their respective segments, in accordance with the current pose estimates. A unique index or label is associated with each object and by rendering the scene according to the current pose estimates, the corresponding segment label (or a label signaling no object) is assigned to each pixel (Fig. 2D). All OpenGL data written by the shaders is directly accessible through CUDA. The most expensive operation at this stage is sorting the label indices. Since every image pixel has a label, a general sorting operation is infeasible. However, the number of labels is limited and an efficient radix sort can be used instead [43]. After sorting, the indices are subsampled if the total available measurements exceeds a threshold ($max\_samples$). All data is then gathered and a compacted stream results.

C. Ordinary Least Squares

Least squares estimation involves composing and solving the normal equations corresponding to the linear least squares system of (15). We can rewrite this as follows to expose the linearity:

$$\begin{aligned} E(\boldsymbol{\alpha}) &= (\mathbf{F}_S\boldsymbol{\alpha} - \mathbf{d}_S)^{\mathrm{T}}(\mathbf{F}_S\boldsymbol{\alpha} - \mathbf{d}_S) + \\ &\quad (\mathbf{F}_M\boldsymbol{\alpha} - \mathbf{d}_M)^{\mathrm{T}}(\mathbf{F}_M\boldsymbol{\alpha} - \mathbf{d}_M) \ , \end{aligned} \tag{24}$$

where the rigid motion parameters are stacked into a screw vector $\boldsymbol{\alpha} = \begin{pmatrix} \boldsymbol{\omega} \\ \mathbf{t} \end{pmatrix}$ for convenience, and $\mathbf{F}_S$, $\mathbf{d}_S$ and $\mathbf{F}_M$,$\mathbf{d}_M$ are obtained by gathering the sensor data according to (6) and (11,12) respectively (for compactness, we no longer distinguish between optical and AR flow at this point). This can be solved in the least-squares sense using the normal equations:

$$(\mathbf{F}_S^{\mathrm{T}}\mathbf{F}_S + \mathbf{F}_M^{\mathrm{T}}\mathbf{F}_M)\boldsymbol{\alpha} = (\mathbf{F}_S^{\mathrm{T}}\mathbf{d}_S + \mathbf{F}_M^{\mathrm{T}}\mathbf{d}_M) \ . \tag{25}$$

The construction of the matrices in the lefthand-side of (25) involves a substantial increase of data at each sample. A sample here refers to a valid optical or AR flow vector, or a valid stereo disparity measurement. For example, for the motion case (11,12) each sample contains the motion vector (2D), the depth-buffer (1D), and the (linearized) pixel location (1D), resulting in a total of 4 input values. The flow normal equations associated with this sample however contain 23 unique values (due to symmetry etc.). For the stereo case (6) the expansion goes from 6 input values (1 disparity, 1 depth, 3 normals, 1 pixel location) to 27 unique values in the normal equations. To limit the data stream, this composition is combined with an initial compaction operation. This is then followed by an additional GPU kernel (a CUDA processing step) to completely reduce the normal equations. These normal equations are then solved on the CPU.

D. Reweighted Least Squares

Tukey's M-estimation scheme [40] (an iteratively reweighted least squares approach) is used to robustly solve the normal equations in the presence of outliers. At each iteration the least squares problem from the previous section is solved, but now weighting each constraint inversely proportional to the error obtained with the current estimate. Samples whose error exceeds a certain threshold are

completely removed from the estimation. Critical here is the scale of the absolute residual distribution, which is directly related to this outlier rejection threshold. This involves computing the median for each segment (object) at each iteration. Since computing the exact median requires too many sorting operations, we instead use an approximation to the median, as successfully used previously in [44]. The particular algorithm used performs a recursive reduction operation on triplets, at each stage replacing each triplet by its center value [45]. This algorithm is very suitable for GPU implementation. We limit the maximum number of elements for determining this approximation to $3^9 = 19,683$ per segment. We did not find any significant reduction in accuracy when using the approximate rather than exact median on the entire benchmarking dataset of Section IV. Concretely, the absolute residuals are computed, the scale is determined, and then the normal equations are computed as in the previous section, but now weighting each sample according to the scale and the residual. The final systems are again solved on the CPU.

E. Processing Times

We have created a synthetic problem dataset to evaluate how the computation times scale as a function of the number of objects being tracked. In this experiment the images were of resolution 640×480. The same object was rendered multiple times in a regular grid. The left camera images for the first frame for problems involving 4, 36 and 144 objects are shown in Fig. 7(A–C). The objects undergo small pose changes in this experiment. Note that here the pose estimation difficulty is not important, rather we want to guarantee a very large number of valid optical flow and stereo samples for each segment so as to maximize the computational complexity.

Three internal (M-estimation) and three external (ICP) iterations were performed. The component and total times for a 6DOF tracking problem ranging from one to 160 objects are shown in Fig. 7. All times were obtained employing a single GPU of a Geforce GTX 590. As expected, the pre-processing stage (dominated by the radix sort) is the most time-consuming. A substantial increase occurs at 16 objects since a switch is made from 4 to 8 bit radix sorting. More efficient and gradual implementations can be used here instead [46]. The times required to compose and solve the normal equations (including CPU–GPU transfers), and for rendering, increase linearly (but slowly) with number of objects. In general this increase is controlled and we also see excellent scaling in terms of the number of samples used from 50,000 (Fig. 7D) to 500,000 (Fig. 7E). With the number of objects considered here, rendering and normal equations solving times are negligible.

A breakdown of the time required to compute the low level dense cues for a single 640×480 image frame is provided in Table I (once again using one GPU of a Geforce GTX 590). As discussed in Section III-A, four Gabor pyramids need to be computed (left, right, 2× left AR). A number of rendering steps are also required to create the stereo priors and AR images. Note that the low-level component computes dense

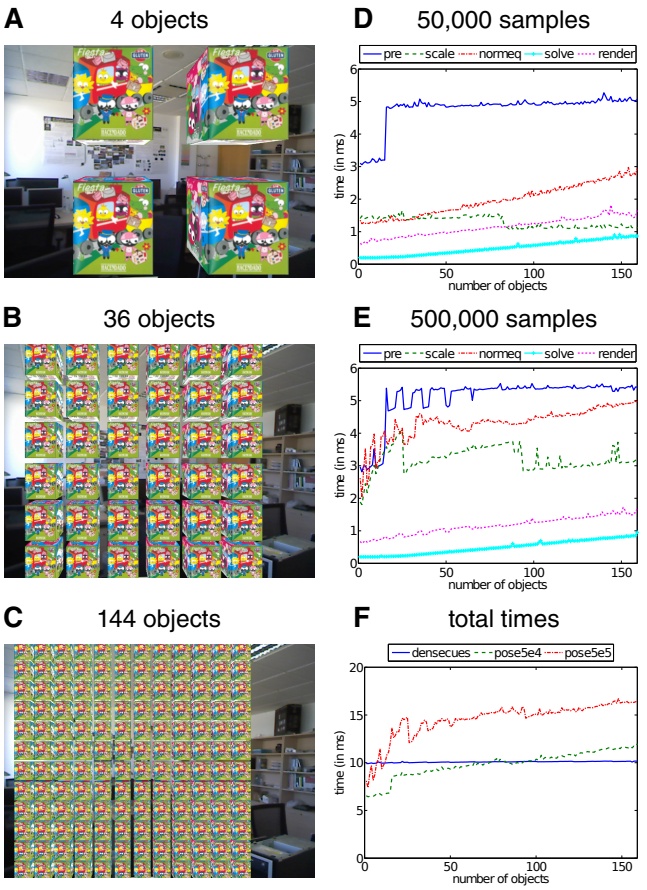


Fig. 7. (A,B,C) Example synthetic images used to evaluate computational performance as a function of number of objects tracked. Computation times of the pose update component for (D) 50,000 and (E) 500,000 samples. The legend refers to the following stages: pre-processing *(pre)*, absolute residuals and scale *(scale)*, composition and reduction of the normal equations *(normeq)*, solving the normal equations *(solve)* and rendering *(render)*. The total computation times for the dense cue extraction *(densecues)* and pose estimation using 50,000 *(pose5e4)* and 500,000 *(pose5e5)* samples are shown in (F). These times are lower than the total times in (D) and (E) due to the reduced overhead of timing the components.

stereo and three times optical flow altogether at ±100 Hz. Table II contains the framerates achieved by the complete tracking system (low-level and pose updates) for a number of different configurations. These times correspond to those reported in Fig. 7F. Note that the complete tracking system operates at 38 Hz when tracking 150 objects using 500,000 samples.

The sparse detection component runs independently on the second GPU of the Geforce GTX 590 so that it does not affect the tracker's speed. Its estimates are employed when available and so its speed mainly determines the recovery latency in case tracking is lost. Our current implementation runs at 20 Hz. This can be increased by reducing the model size and/or using more efficient keypoints and descriptors, but at the cost of reduced accuracy. An alternative is to use FPGA-accelerated SIFT implementations [47].

TABLE I
PROCESSING TIMES DENSE LOW-LEVEL CUES (IN MS)

dense cues *(640×480 image size)*	
– image transfer CPU → GPU	0.4
– Gabor pyramid (4×)	3.3
– optical flow	1.5
– AR flow (2×)	3.0
– model-based stereo	1.2
– rendering	0.9
total	10.3

TABLE II
TRACKING FRAME RATES (IN HZ)

	# samples	
# objects	50,000	500,000
1	61	55
20	54	43
150	46	38

IV. SYNTHETIC BENCHMARK DATASET

We have constructed an extensive synthetic benchmark dataset to evaluate pose trackers under a variety of realistic conditions. Its creation is discussed in detail next, but Fig. 11 already shows some representative examples that illustrate the large distance range, background and object variability, and the added noise and occlusions. The complete dataset is available on-line.[1] Note that we only focus on the single object case here to facilitate the comparison with alternative methods. We show many real-world examples involving multiple objects in Section V-C and *Supplemental Material Video 3*.[2]

A. Object Models

We have selected four objects from the publicly available KIT object models database [48]. Snapshots of these models are shown in Fig. 8 (*soda*, *soup*, *clown*, and *candy*) and provide a good sample of the range of shapes and textures available in the database. We also included two cube-shaped objects, one richly textured (*cube*), and the other containing only edges (*edge*).

B. Object and Background Motion

Using the proposed system, we recorded a realistic and complex motion trace by manually manipulating an object similar to *cube*. This, possibly erroneous, motion trace was then used to generate synthetic sequences and so it is, by definition, the ground-truth object motion. The trace is shown in Fig. 9 and covers a high dynamic range, varying all six degrees of freedom. The realism and complexity of the sequences is further increased by blending the rendered objects into a real-world stereo sequence recorded with a moving camera in a cluttered office environment. Some examples are shown in Fig. 11 but, to fully appreciate the complexity, we refer to *Supplemental Material Video 1*.

[1] http://www.karlpauwels.com/datasets/rigid-pose/
[2] All supplemental material videos are available at http://www.karlpauwels.com/ieee-tcsvt-supplemental-material/

Fig. 8. Object models used in the synthetic sequences.

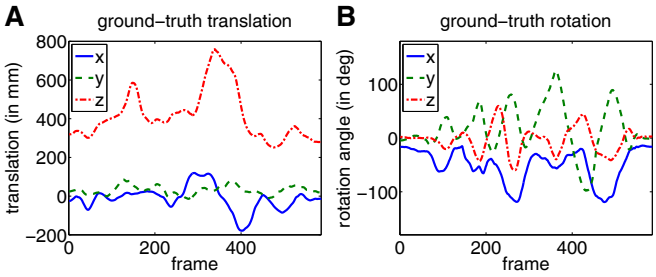


Fig. 9. Ground-truth object motion in synthetic sequences.

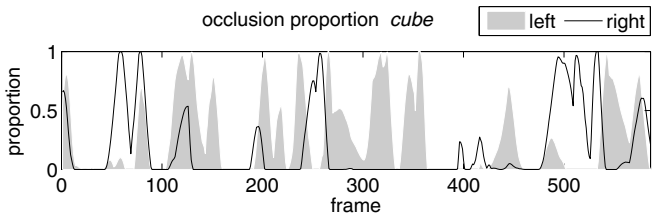


Fig. 10. Proportion occlusion in the *cube* sequence.

C. Noise and Occluder

To further explore the limitations of pose tracking methods, different sequences are created corrupted either by noise or an occluding object. For the noisy sequences, Gaussian noise (with σ equal to one tenth of the intensity range) is added separately to each color channel, frame, and stereo image (Fig. 11B). To obtain realistic occlusion (with meaningful motion and stereo cues), we added a randomly bouncing 3D teddy bear object to the sequence (Fig. 11C,D and *Supplemental Material Video 1*). The occlusion proportion of the *cube* object over the sequence is shown in Fig. 10. Although this differs for the left and right sequences, none of the methods evaluated here exploit this (*e.g.* our dense stereo cue is affected by *either* left or right occlusions).

D. Performance Evaluation

Pose trackers are usually evaluated by comparing the estimated to the ground-truth or approximate ground-truth pose across an entire sequence [14], [15], [18], [20]–[22]. However, once a tracker is lost, the subsequent estimates (and their errors) become irrelevant. For example, if tracking is lost early in the scene, the average error will typically be very large, but this doesn't mean the tracker cannot track the object in the remainder of the sequence, if re-initialized.

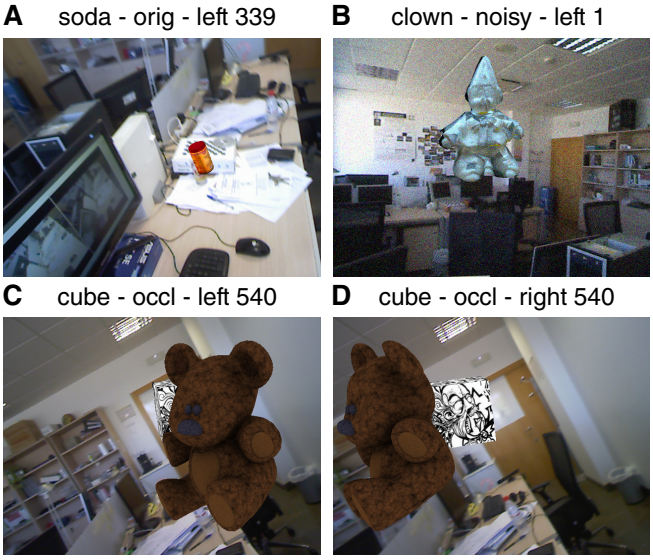


Fig. 11. Indicative samples from the different synthetic sequences illustrating (A) large distance range, (B) added noise, and (C,D) different occlusion proportions in the left and right sequences (orig = noise free; occl = occluded; number = frame index).

For this reason, we propose to instead measure the *proportion* of a sequence that can be tracked successfully. Since we use synthetic sequences, we can continuously monitor tracking accuracy and automatically reset when tracking is lost. But how to decide if tracking is lost? Rotations and translations affect an object's position in very different ways and are therefore hard to summarize into a single error. Instead we use the largest distance between corresponding vertices, $\mathbf{v}_j$, of the object transformed according to the ground-truth $(\mathrm{R}, \mathbf{t})$ and estimated pose $(\hat{\mathrm{R}}, \hat{\mathbf{t}})$:

$$e_P = \max_j \|(\hat{\mathrm{R}}\,\mathbf{v}_j + \hat{\mathbf{t}}) - (\mathrm{R}\,\mathbf{v}_j + \mathbf{t})\| \ . \qquad (26)$$

This measure is easy to interpret and is sensible from a task-based perspective. Considering for example a grasping or collision-avoidance task, it is important to understand the positional accuracy of the entire object, incorporating both its shape and the pose. When this distance exceeds a threshold (e.g. 10 mm), the tracker is reset to the ground-truth. The proportion of the sequence tracked correctly then constitutes a scalar performance measure for the entire sequence. To put this measure in perspective, a *static* error is also computed for each sequence using a 'naive tracker'. This 'tracker' simply never updates the pose. As a consequence all resets are triggered by the object motion alone and the error provides an indication of the sequence complexity. For example, in a sequence with a static object, perfect performance will be achieved.

V. Results

A. Stereo and Optical Flow Synergy

Table III shows results on the least textured sequences *soda* and *edge*. The *static* performance is around 50% in both, which means that, without tracking, a reset is required approximately every other frame. Due to the low texture, shape-symmetry (*soda*) and shape-planarity (*edge*), stereo-only performance is quite bad in these sequences and even below *static* in the noisy and occluded scenarios. Optical-flow-only performance is better but, when both are combined, great improvements can be observed. This highlights the importance of combining multiple cues, particularly in low-texture situations. The AR flow and sparse cues further improve the results, but these are discussed in the next section (and shown in Table IV).

TABLE III
Tracking success rate (in %) – stereo and optical flow

	soda			*edge*		
	orig	noisy	occl	orig	noisy	occl
static	53	53	53	50	50	50
stereo	77	47	42	59	50	32
optical flow	93	81	57	78	81	40
stereo+flow	100	96	63	92	93	52

B. State-of-the-art Methods

The Blocks World Robotic Vision Toolbox (BLORT) [5] provides a number of object learning, recognition, and tracking components that can be combined to robustly track object pose in real-time. We only evaluate the particle-filter-based tracking module here since the recognition module is very similar to our sparse-only method. Each particle represents a pose estimate and is used to render an edge model (combining edges from geometry and texture) that is matched against edges extracted from the current image. We evaluate BLORT's tracking module with the default (real-time) setting with 200 particles and a high precision variant with 10,000 particles. Due to an inefficient rendering procedure, the current tracker implementation of BLORT can not handle models with a high vertex count. We therefore limited the geometrical complexity of the models to 800 triangles in all the sequences.

We also evaluate a state-of-the-art real-time-capable region-based tracker. The PWP3D method [20] uses a 3D geometry model to maximize the discrimination between statistical foreground and background appearance models, by directly operating on the 3D pose parameters. To ensure the best possible performance, we used very small gradient descent step sizes (0.1 degrees for rotation and 1 mm for translation). Together with a large number of iterations (100), this ensures stable convergence (although no longer in real-time). Furthermore, we initialized the PWP3D method at each frame with the ground-truth color histogram of the actual (or unoccluded) frame being processed so that also inaccuracies here do not affect performance.

Table IV summarizes all the results obtained with a tracking reset threshold equal to 10 mm. To better quantify the precision obtained, we have also included the Root Mean Squared (RMS) errors, obtained on the successful frames, for the three translational ($e_T^{\{x,y,z\}}$) and three rotational ($e_R^{\{x,y,z\}}$) components of the pose (using Euler angles for the rotation components) and for the maximal distance error (26). For each condition, we highlight the best result, considering first the Success Rate (SR) and then the RMS distance error.

TABLE IV
TRACKING ERRORS ($e_T^{\{x,y,z\}}$ AND e_P IN MM, $e_R^{\{x,y,z\}}$ IN DEGREES) AND SUCCESS RATE (SR IN %)

	soda – noise free								soup – noise free							
	e_T^x	e_T^y	e_T^z	e_R^x	e_R^y	e_R^z	e_P	SR	e_T^x	e_T^y	e_T^z	e_R^x	e_R^y	e_R^z	e_P	SR
sparse-and-dense	0.3	0.2	1.0	0.4	0.6	0.5	1.5	98.6	0.1	0.1	0.4	0.1	0.1	0.1	0.6	98.3
dense-only	**0.3**	**0.2**	**0.6**	**0.4**	**0.7**	**0.5**	**1.1**	**100.0**	**0.2**	**0.1**	**0.2**	**0.1**	**0.3**	**0.1**	**0.6**	**99.7**
sparse-only	0.7	0.4	3.1	1.0	1.4	0.6	4.2	61.1	0.7	0.6	1.7	0.6	0.7	0.4	2.8	93.0
part. filt. 10,000	1.2	0.7	3.9	1.3	2.8	1.3	5.7	75.5	1.4	1.1	3.4	1.2	1.9	1.0	5.7	76.9
region-based	0.5	1.1	2.2	1.9	7.2	4.8	6.2	84.4	0.5	1.3	1.8	0.7	1.7	0.6	3.8	96.2
part. filt. 200	1.3	1.0	3.9	1.9	3.1	1.8	6.1	57.5	1.6	1.3	3.4	1.7	2.5	1.3	6.6	46.7
static	3.1	1.9	3.5	1.6	2.8	1.9	6.8	53.1	2.8	1.9	3.2	1.4	2.3	1.6	7.3	45.0
	soda – noisy								soup – noisy							
	e_T^x	e_T^y	e_T^z	e_R^x	e_R^y	e_R^z	e_P	SR	e_T^x	e_T^y	e_T^z	e_R^x	e_R^y	e_R^z	e_P	SR
sparse-and-dense	0.9	0.7	1.8	1.0	2.3	2.2	3.5	96.9	0.4	0.6	0.5	0.3	0.5	0.4	1.4	98.6
dense-only	**1.0**	**0.7**	**1.8**	**1.0**	**2.4**	**2.3**	**3.6**	**98.5**	**0.4**	**0.6**	**0.5**	**0.3**	**0.5**	**0.3**	**1.4**	**99.0**
sparse-only	1.2	0.8	3.1	1.7	2.4	1.2	5.3	36.5	0.7	0.6	2.0	0.9	1.1	0.5	3.6	74.0
part. filt. 10,000	1.5	0.8	4.4	1.3	2.8	1.2	6.2	65.2	1.6	1.2	3.3	1.5	2.1	1.0	6.1	65.8
region-based	0.5	1.1	2.2	2.2	6.6	4.9	6.1	84.4	0.5	1.2	2.0	0.7	1.8	0.6	4.0	95.5
part. filt. 200	1.4	1.0	4.1	1.8	3.3	1.9	6.4	59.6	1.6	1.4	3.4	1.8	2.6	1.6	6.8	53.9
static	3.1	1.9	3.5	1.6	2.8	1.9	6.8	53.1	2.8	1.9	3.2	1.4	2.3	1.6	7.3	45.0
	soda – occluded								soup – occluded							
	e_T^x	e_T^y	e_T^z	e_R^x	e_R^y	e_R^z	e_P	SR	e_T^x	e_T^y	e_T^z	e_R^x	e_R^y	e_R^z	e_P	SR
sparse-and-dense	**0.7**	**0.6**	**1.7**	**0.9**	**1.8**	**1.2**	**3.1**	**68.3**	**0.5**	**0.6**	**0.9**	**0.7**	**0.7**	**0.4**	**2.2**	**80.0**
dense-only	0.7	0.6	1.3	1.0	1.9	1.3	2.9	66.8	0.9	0.6	0.7	0.7	0.6	0.6	2.3	73.5
sparse-only	0.8	0.4	3.2	1.1	1.5	0.8	4.3	44.0	0.7	0.6	1.6	0.7	1.0	0.4	2.8	76.5
part. filt. 10,000	1.7	1.1	3.9	1.2	3.0	1.1	6.0	53.8	1.6	1.3	3.2	1.3	2.3	0.7	5.9	62.5
region-based	0.7	1.1	2.4	1.9	7.2	5.3	6.3	44.0	0.9	1.2	2.2	1.2	2.2	0.9	4.8	44.0
part. filt. 200	1.7	1.2	4.2	1.9	3.5	2.1	6.8	45.2	1.6	1.4	3.6	1.7	2.5	1.3	6.7	40.2
static	3.1	1.9	3.5	1.6	2.8	1.9	6.8	53.1	2.8	1.9	3.2	1.4	2.3	1.6	7.3	45.0
	clown – noise free								candy – noise free							
	e_T^x	e_T^y	e_T^z	e_R^x	e_R^y	e_R^z	e_P	SR	e_T^x	e_T^y	e_T^z	e_R^x	e_R^y	e_R^z	e_P	SR
sparse-and-dense	0.2	0.1	0.5	0.2	0.3	0.2	0.9	100.0	0.1	0.1	0.3	0.1	0.2	0.1	0.6	99.8
dense-only	**0.1**	**0.2**	**0.2**	**0.2**	**0.3**	**0.2**	**0.7**	**100.0**	**0.1**	**0.1**	**0.1**	**0.1**	**0.2**	**0.1**	**0.3**	**100.0**
sparse-only	0.5	0.4	2.0	0.7	0.8	0.5	3.1	92.0	0.6	0.4	1.7	0.5	0.8	0.4	2.7	95.4
part. filt. 10,000	1.1	0.9	3.2	1.4	1.9	1.1	5.6	87.5	1.4	1.0	3.1	1.2	2.1	1.1	5.4	76.7
region-based	0.4	1.2	1.9	0.9	2.2	1.0	4.3	96.4	0.6	1.3	2.4	1.2	3.4	2.1	5.4	83.6
part. filt. 200	1.3	1.2	3.6	1.8	2.6	1.4	6.7	56.2	1.6	1.3	3.5	1.6	2.6	1.5	6.6	45.9
static	2.9	1.7	3.2	1.5	2.4	1.7	7.2	47.1	2.8	2.0	3.2	1.3	2.4	1.7	7.2	46.4
	clown – noisy								candy – noisy							
	e_T^x	e_T^y	e_T^z	e_R^x	e_R^y	e_R^z	e_P	SR	e_T^x	e_T^y	e_T^z	e_R^x	e_R^y	e_R^z	e_P	SR
sparse-and-dense	0.3	0.4	0.9	0.4	0.6	0.4	1.6	98.3	0.2	0.1	0.4	0.1	0.3	0.2	0.7	99.7
dense-only	**0.3**	**0.3**	**0.6**	**0.4**	**0.6**	**0.4**	**1.4**	**99.8**	**0.1**	**0.1**	**0.2**	**0.1**	**0.2**	**0.1**	**0.5**	**100.0**
sparse-only	0.7	0.7	2.4	1.1	1.3	0.9	4.2	71.1	0.7	0.7	1.8	0.7	0.9	0.5	3.2	90.4
part. filt. 10,000	1.4	1.1	3.3	1.5	2.4	1.5	6.3	82.4	1.4	1.2	3.0	1.3	2.2	1.0	5.6	76.4
region-based	0.6	1.3	3.0	1.2	2.1	1.2	5.3	89.4	0.7	1.4	2.6	1.2	3.7	2.2	5.8	83.9
part. filt. 200	1.4	1.2	3.6	1.8	2.6	1.7	6.8	62.2	1.7	1.4	3.6	1.6	2.6	1.6	6.6	48.8
static	2.9	1.7	3.2	1.5	2.4	1.7	7.2	47.1	2.8	2.0	3.2	1.3	2.4	1.7	7.2	46.4

The proposed *dense-only* tracking method obtains an SR close to 100% regardless of model shape, texture (see *edge*), or sequence noise (although e_P increases with noise as can be expected). In the occluded scenario however, it is frequently outperformed by the *sparse-only* and high quality *particle filtering* methods. But, when combined with the sparse method (II-C2) the synergy of both modules is confirmed. *Sparse-and-dense* retains the excellent performance of the *dense-only* method with greatly improved robustness to occlusions, even outperforming *sparse-only* on most sequences. The slight decrease in performance of *sparse-and-dense* with respect to *dense-only* is due to the limits of the selection mechanism. At times, a less accurate sparse estimate can result in a larger AR-flow proportion. A potential improvement is to also consider the magnitude of the AR flow, but we found this to be less robust. More important though is that, unlike *dense-only*, *sparse-and-dense* also enables recovery from tracking failures, which is critical in real-world applications. The *particle filter* method performs very well provided a very large number of particles are used. In the real-time setting (200 particles) the performance is not much better than *static*. The *sparse-only* method performs badly on *soda* due to the weak texture, and fails on *edge* due to the complete lack of texture. The *region-based* tracker performs consistently well on the noise-free and noisy sequences, but fails dramatically in the presence of an occluder. Although it can handle certain types of occlusions, large failures occur when an entire side of the object contour is occluded [20].

C. Real-world Sequences

The proposed method also yields excellent results in real-world scenarios. Some example single object results with a cluttered scene, occluders, and camera motion are shown in Fig. 12. The dense estimate is selected as winner in Fig. 12A,B. This usually occurs when the object is far away (A) or suffering from severe motion blur (B). The sparse estimate is usually selected when only a small part of the object is visible (C). Figure 12D finally shows some tracking

TABLE IV
CONTINUED

	clown – occluded								candy – occluded							
	e_T^x	e_T^y	e_T^z	e_R^x	e_R^y	e_R^z	e_P	SR	e_T^x	e_T^y	e_T^z	e_R^x	e_R^y	e_R^z	e_P	SR
sparse-and-dense	**0.6**	**0.5**	**1.1**	**0.7**	**0.8**	**0.6**	**2.4**	**77.4**	**0.5**	**0.5**	**1.0**	**0.6**	**0.9**	**0.5**	**2.1**	**81.2**
dense-only	1.1	0.7	0.8	0.8	1.0	1.1	3.0	69.5	0.5	0.7	0.6	0.6	0.8	0.5	2.0	74.7
sparse-only	0.4	0.4	1.9	0.7	0.8	0.4	2.9	74.1	0.6	0.5	1.7	0.5	0.9	0.4	2.8	80.0
part. filt. 10,000	1.3	1.0	3.5	1.3	2.3	1.2	6.2	76.4	1.5	1.3	3.2	1.2	2.2	1.0	5.7	63.9
region-based	0.8	1.2	2.3	1.0	1.9	0.8	4.5	43.8	0.9	1.3	2.3	1.5	3.2	1.9	5.5	39.2
part. filt. 200	1.4	1.4	3.4	2.0	2.4	1.6	6.7	48.1	1.7	1.5	3.1	1.7	2.7	1.9	6.7	40.9
static	2.9	1.7	3.2	1.5	2.4	1.7	7.2	47.1	2.8	2.0	3.2	1.3	2.4	1.7	7.2	46.4
	cube – noise free								edge – noise free							
	e_T^x	e_T^y	e_T^z	e_R^x	e_R^y	e_R^z	e_P	SR	e_T^x	e_T^y	e_T^z	e_R^x	e_R^y	e_R^z	e_P	SR
sparse-and-dense	0.1	0.1	0.3	0.1	0.2	0.1	0.5	100.0	**1.4**	**0.9**	**0.8**	**1.1**	**1.5**	**1.6**	**3.9**	**98.3**
dense-only	**0.1**	**0.1**	**0.1**	**0.1**	**0.1**	**0.1**	**0.2**	**100.0**	**1.4**	**0.9**	**0.8**	**1.1**	**1.5**	**1.6**	**3.9**	**98.3**
sparse-only	0.5	0.4	1.7	0.6	0.7	0.4	2.7	97.6	–	–	–	–	–	–	–	0.0
part. filt. 10,000	1.1	0.9	3.0	1.3	1.7	1.0	5.2	93.2	1.3	0.9	3.1	1.1	1.7	1.0	5.2	72.4
region-based	0.7	1.2	2.8	1.7	1.6	0.9	5.1	83.9	0.7	1.1	2.3	1.7	1.5	0.8	4.6	84.6
part. filt. 200	1.5	1.1	3.7	1.9	2.2	1.5	6.8	53.1	1.6	1.2	3.5	1.7	2.1	1.6	6.5	63.2
static	3.0	2.0	3.2	1.6	2.5	1.7	7.2	50.2	3.0	2.0	3.2	1.6	2.5	1.7	7.2	50.2
	cube – noisy								edge – noisy							
	e_T^x	e_T^y	e_T^z	e_R^x	e_R^y	e_R^z	e_P	SR	e_T^x	e_T^y	e_T^z	e_R^x	e_R^y	e_R^z	e_P	SR
sparse-and-dense	0.1	0.1	0.1	0.1	0.2	0.1	0.4	100.0	**1.2**	**0.9**	**1.0**	**1.3**	**1.8**	**1.5**	**4.1**	**98.6**
dense-only	**0.1**	**0.1**	**0.1**	**0.1**	**0.1**	**0.1**	**0.3**	**100.0**	**1.2**	**0.9**	**1.0**	**1.3**	**1.8**	**1.5**	**4.1**	**98.6**
sparse-only	0.5	0.4	1.8	0.7	0.7	0.4	2.8	95.9	–	–	–	–	–	–	–	0.0
part. filt. 10,000	1.3	0.9	2.8	1.2	1.9	1.1	5.2	93.7	1.4	1.0	3.2	1.3	1.8	0.9	5.5	91.3
region-based	0.7	1.2	3.2	1.6	1.9	1.2	5.6	74.3	0.6	1.1	2.5	1.8	1.7	1.0	4.9	83.7
part. filt. 200	1.5	1.2	3.6	2.0	2.1	1.4	6.7	53.6	1.7	1.2	3.8	1.8	2.1	1.4	6.8	62.5
static	3.0	2.0	3.2	1.6	2.5	1.7	7.2	50.2	3.0	2.0	3.2	1.6	2.5	1.7	7.2	50.2
	cube – occluded								edge – occluded							
	e_T^x	e_T^y	e_T^z	e_R^x	e_R^y	e_R^z	e_P	SR	e_T^x	e_T^y	e_T^z	e_R^x	e_R^y	e_R^z	e_P	SR
sparse-and-dense	0.6	0.7	1.3	0.7	0.9	0.7	2.6	76.2	1.6	1.3	1.3	1.6	1.9	2.0	4.9	57.0
dense-only	0.6	0.7	0.7	0.7	0.9	0.7	2.3	71.1	1.6	1.3	1.3	1.6	1.9	2.0	4.9	57.0
sparse-only	**0.6**	**0.5**	**1.9**	**0.8**	**0.8**	**0.4**	**2.9**	**78.8**	–	–	–	–	–	–	–	0.0
part. filt. 10,000	1.3	1.2	3.0	1.5	1.8	1.2	5.7	76.2	**1.5**	**1.1**	**3.1**	**1.5**	**1.9**	**1.0**	**5.7**	**68.2**
region-based	0.9	1.2	2.9	1.5	1.9	1.0	5.3	37.8	0.8	1.2	2.4	1.4	1.7	0.9	4.6	38.9
part. filt. 200	1.5	1.4	3.4	2.1	2.2	1.5	6.7	38.9	1.8	1.2	3.6	1.8	2.3	1.4	6.6	49.7
static	3.0	2.0	3.2	1.6	2.5	1.7	7.2	50.2	3.0	2.0	3.2	1.6	2.5	1.7	7.2	50.2

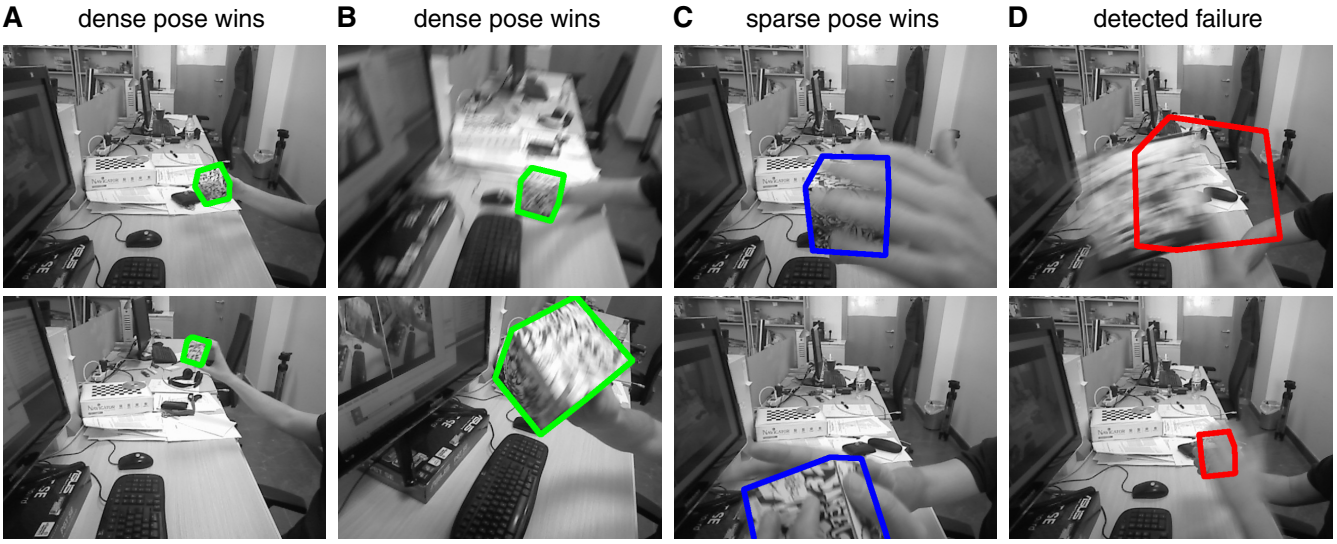


Fig. 12. Indicative real-world single object pose estimation results, showing how the dense pose is selected when (A) the object is far and/or (B) motion-blurred, how the sparse pose is selected in case of (C) strong occlusions, and how (D) failures can be detected correctly.

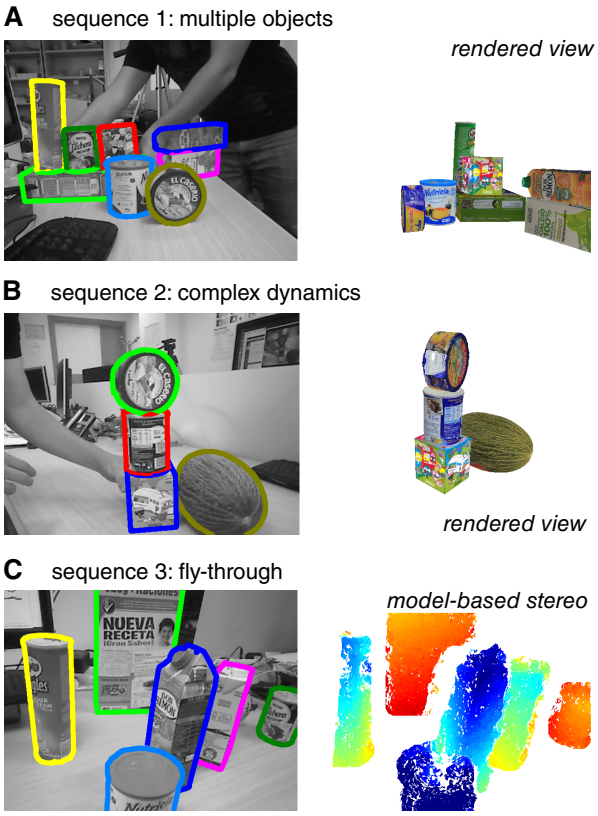


Fig. 13. Real world pose detection and tracking results involving multiple objects and complex dynamics. These images are representative snapshots of much longer sequences (see *Supplemental Material Video 3*).

failures that are detected correctly by the reliability measure (proportion AR flow < 0.15). See *Supplemental Material Video 2* for more single object real-world results.

Figure 13 contains three snapshots of longer sequences contained in *Supplemental Material Video 3*. This video demonstrates the proposed method in various complex scenarios involving multiple interacting objects undergoing manipulation. The 3D models were obtained using a publicly available solution [1]. In some of these scenarios (Fig. 13A,B) we also show an image rendered from a viewpoint different from the camera, to more clearly show the precision obtained by our system. Note how in Fig. 13A,B the objects are aligned quite precisely even though we do not prevent them from intersecting each other. Our approach can handle a great variety in shape and appearance (*e.g.* the *melon* object in Fig. 13B). The method is also quite robust to inaccuracies at the modeling stage (*e.g.* due to limits of the acquisition process the bottoms of the objects are never modeled, see Fig. 13B *caserio* object, but see also Fig. 2). Occlusions between the different objects are also handled automatically through the rendering process and subsequent label assignment. Figure 13C shows how the model-based stereo handles multiple objects and respective occlusions.

VI. Discussion

Although highly robust, the edge-based particle filter method requires a large number of particles to achieve high accuracy. Since each particle requires a rendering step, the performance critically depends on model complexity. This method, and other related particle filter methods [25] are also difficult to extend to the articulated scenario due to the increased dimensionality of the problem. The sparse keypoint-based method is highly robust to occlusions and provides excellent synergy with the dense methods proposed here. The region-based method does not require edges or texture and performs very well. It does have problems with symmetric objects, is slow to converge, and fails on certain types of occlusions. This requires the use of multiple cameras or explicit modeling of the occlusions. There is much potential for incorporating color-based (region-based) cues into the proposed method. However it is not straightforward to process these in a GPU-friendly manner.

Note that the proposed method also supports depth cues other than stereo (e.g. from a Kinect sensor), and conversely, enables for the incorporation of motion cues in current depth-only applications [2]. Current Kinect versions however do not provide the high shape detail close to the camera, nor the high framerates achieved by our model-based stereo algorithm [49].

The scalability results shown here can be considered somewhat artificial. Nonetheless, there are many situations where this approach is useful. Multiple (not necessarily overlapping) camera views can be combined to simultaneously track a very large number of objects. Also articulated or non-rigid objects could be decomposed or approximated by a large number of rigid components that can be jointly tracked using this approach. Specifically, it has been shown how the parts can be considered individually as rigid objects, and the constraints enforced later [13], [50]. Additional constraints can be obtained from physics simulations and jointly incorporated in a temporal filtering framework considering also (multi-)object persistency, manipulator feedback, etc.

VII. Conclusion

We have presented a novel model-based multi-cue approach for simultaneously tracking the 6DOF poses of a very large number of rigid objects of arbitrary shapes that exploits dense motion and stereo cues, sparse keypoint features, and feedback from the modeled scene to the cue extraction. The method is inherently parallel and efficiently implemented using GPU acceleration. We have introduced an evaluation methodology and benchmark dataset specifically for this problem. Using this dataset we have shown improved accuracy, robustness, and speed of the proposed method as compared to state-of-the-art real-time-capable methods.

Acknowledgment

The authors gratefully acknowledge the European FP7 project RoboHow (FP7-ICT-288533) and the Spanish National Project NEUROPACT (TIN2013-47069-P). The GPU used for this research was donated by the NVIDIA Corporation.

REFERENCES

[1] Autodesk, "123D Catch," http://www.123dapp.com/catch/, 2014.

[2] R. Newcombe, A. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon, "KinectFusion: real-time dense surface mapping and tracking," in *Proc. IEEE Int. Symp. on Mixed and Augmented Reality*, Oct. 2011, pp. 127–136.

[3] M. Krainin, P. Henry, X. Ren, and D. Fox, "Manipulator and object tracking for in-hand 3D object modeling," *Int. J. Robot. Res.*, 2011.

[4] L. Vacchetti, V. Lepetit, and P. Fua, "Stable real-time 3D tracking using online and offline information," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 10, pp. 1385 –1391, Oct. 2004.

[5] T. Mörwald, J. Prankl, A. Richtsfeld, M. Zillich, and M. Vincze, "BLORT - The Blocks World Robotic Vision Toolbox," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2010.

[6] A. Collet, M. Martinez, and S. S. Srinivasa, "The MOPED framework: Object recognition and pose estimation for manipulation," *Int. J. Robot. Res.*, vol. 30, no. 10, pp. 1284–1306, Apr. 2011.

[7] 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 *Proc. IEEE Int. Conf. on Computer Vision and Pattern Recognition*, June 2013.

[8] S. Holzer, S. Hinterstoisser, S. Ilic, and N. Navab, "Distance transform templates for object detection and pose estimation," in *Proc. IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2009, pp. 1177–1184.

[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 *Proc. Int. Conf. on Computer Vision*, 2011, pp. 858–865.

[10] P. Besl and N. McKay, "A method for registration of 3D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, pp. 239–256, 1992.

[11] 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 *Proc. Asian Conf. on Computer Vision*. Springer, 2013, pp. 548–562.

[12] J. Liebelt and C. Schmid, "Multi-view object class detection with a 3D geometric model," in *Proc. IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2010, pp. 1688–1695.

[13] T. Drummond and R. Cipolla, "Real-time visual tracking of complex structures," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 932–946, 2002.

[14] V. Kyrki and D. Kragic, "Tracking rigid objects using integration of model-based and model-free cues," *Mach. Vision Appl.*, vol. 22, pp. 323–335, 2011.

[15] M. Pressigout and E. Marchand, "Real-time hybrid tracking using edge and texture information," *Int. J. Robot. Res.*, vol. 26, pp. 689–713, 2007.

[16] M. Pressigout, E. Marchand, and E. Memin, "Hybrid tracking approach using optical flow and pose estimation," in *Proc. IEEE Int. Conf. on Image Processing*, Oct. 2008, pp. 2720–2723.

[17] P. Azad, D. Munch, T. Asfour, and R. Dillmann, "6-DoF model-based tracking of arbitrarily shaped 3D objects," in *Proc. IEEE Int. Conf. on Robotics and Automation*, May 2011, pp. 5204–5209.

[18] C. Choi and H. Christensen, "Robust 3D visual tracking using particle filtering on the special euclidean group: A combined approach of keypoint and edge features," *Int. J. Robot. Res.*, vol. 31, no. 4, pp. 498–519, 2012.

[19] C. Teuliere, E. Marchand, and L. Eck, "Using multiple hypothesis in model-based tracking," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2010, pp. 4559–4565.

[20] V. Prisacariu and I. Reid, "PWP3D: Real-time segmentation and tracking of 3D objects," *Int. J. Comput. Vision*, vol. 98, pp. 335–354, 2012.

[21] T. Brox, B. Rosenhahn, J. Gall, and D. Cremers, "Combined region and motion-based 3D tracking of rigid and articulated objects," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 3, pp. 402–415, 2010.

[22] A. Petit, E. Marchand, and K. Kanani, "A robust model-based tracker combining geometrical and color edge information," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2013.

[23] Q. Wang, W. Zhang, X. Tang, and H.-Y. Shum, "Real-time bayesian 3-D pose tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 12, pp. 1533–1541, Dec 2006.

[24] P. Azad, T. Asfour, and R. Dillmann, "Accurate shape-based 6-DoF pose estimation of single-colored objects," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Oct. 2009, pp. 2690–2695.

[25] C. Choi and H. Christensen, "RGB-D object tracking: A particle filter approach on GPU," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2013, pp. 1084–1091.

[26] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *Proc. IEEE Int. Conf. on Computer Vision and Pattern Recognition*, Jun. 2011, pp. 1297–1304.

[27] C. Teuliere and E. Marchand, "Direct 3D servoing using dense depth maps," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2012, pp. 1741–1746.

[28] A. Talukder and L. Matthies, "Real-time detection of moving objects from moving vehicles using dense stereo and optical flow," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, vol. 4, Sep. 2004, pp. 3718–3725 vol.4.

[29] 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 *Proc. IEEE Int. Conf. on Computer Vision and Pattern Recognition*, Portland, June 2013.

[30] D. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[31] K. Pauwels, M. Tomasi, J. Díaz, 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 T. Comput.*, vol. 61, no. 7, pp. 999–1012, 2012.

[32] M. Tomasi, M. Vanegas, F. Barranco, J. Daz, and E. Ros, "Massive parallel-hardware architecture for multiscale stereo, optical flow and image-structure computation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 2, pp. 282–294, 2012.

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

[34] S. Sabatini, G. Gastaldi, F. Solari, K. Pauwels, M. Van Hulle, J. Díaz, E. Ros, N. Pugeault, and N. Krüger, "A compact harmonic code for early vision based on anisotropic frequency channels," *Comput. Vis. Image Und.*, vol. 114, no. 6, pp. 681–699, 2010.

[35] K. Pauwels and M. Van Hulle, "Optic flow from unstable sequences through local velocity constancy maximization," *Image Vision Comput.*, vol. 27, no. 5, pp. 579–587, 2009.

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

[37] G. Blais and M. Levine, "Registering multiview range data to create 3D computer objects," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 8, pp. 820–824, 1995.

[38] C. Yang and G. Medioni, "Object modelling by registration of multiple range images," *Image Vision Comput.*, vol. 10, no. 3, pp. 145–155, 1992.

[39] H. C. Longuet-Higgins and K. Prazdny, "The interpretation of a moving retinal image," *P. Roy. Soc. B-Biol. Sci.*, vol. 208, pp. 385–397, 1980.

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

[41] V. Lepetit and P. Fua, "Monocular model-based 3D tracking of rigid objects," *Foundations and Trends in Computer Graphics and Vision*, vol. 1, pp. 1–89, 2005.

[42] E. Lindholm, J. Nickolls, S. Oberman, and J. Montrym, "NVIDIA Tesla: A unified graphics and computing architecture," *IEEE Micro*, vol. 28, no. 2, pp. 39–55, 2008.

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

[44] K. Pauwels, N. Krüger, M. Lappe, F. Wörgötter, and M. Van Hulle, "A cortical architecture on parallel hardware for motion processing in real time," *J. Vision*, vol. 10, no. 10, 2010.

[45] S. Battiato, D. Cantone, D. Catalano, G. Cincotti, and M. Hofri, "An efficient algorithm for the approximate median selection problem," in *Algorithms and Complexity*, ser. Lecture Notes in Computer Science, G. Bongiovanni, R. Petreschi, and G. Gambosi, Eds. Springer Berlin Heidelberg, Jan. 2000, no. 1767, pp. 226–238.

[46] D. Merrill and A. Grimshaw, "High performance and scalable radix sorting: A case study of implementing dynamic parallelism for GPU computing," *Parallel Processing Letters*, vol. 21, no. 02, pp. 245–272, 2011.

[47] J. Jiang, X. Li, and G. Zhang, "SIFT hardware implementation for real-time image feature extraction," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 7, pp. 1209–1220, 2014.

[48] A. Kasper, Z. Xue, and R. Dillmann, "The KIT object models database: An object model database for object recognition, localization and manipulation in service robotics," *Int. J. Robot. Res.*, vol. 31, no. 8, pp. 927–934, 2012.

[49] Wikipedia, "Kinect," 2014.

[50] K. Pauwels, L. Rubio, and E. Ros, "Real-time model-based articulated object pose detection and tracking with variable rigidity constraints," in *Proc. IEEE Int. Conf. on Computer Vision and Pattern Recognition*, Columbus, Ohio, 2014.

2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Online Contact Point Estimation for Uncalibrated Tool Use

Yiannis Karayiannidis, Christian Smith, Francisco E. Viña, and Danica Kragic

***Abstract*— One of the big challenges for robots working outside of traditional industrial settings is the ability to robustly and flexibly grasp and manipulate tools for various tasks. When a tool is interacting with another object during task execution, several problems arise: a tool can be partially or completely occluded from the robot's view, it can slip or shift in the robot's hand - thus, the robot may lose the information about the exact position of the tool in the hand. Thus, there is a need for online calibration and/or recalibration of the tool. In this paper, we present a model-free online tool-tip calibration method that uses force/torque measurements and an adaptive estimation scheme to estimate the point of contact between a tool and the environment. An adaptive force control component guarantees that interaction forces are limited even before the contact point estimate has converged. We also show how to simultaneously estimate the location and normal direction of the surface being touched by the tool-tip as the contact point is estimated. The stability of the the overall scheme and the convergence of the estimated parameters are theoretically proven and the performance is evaluated in experiments on a real robot.**

I. Introduction

The ability to robustly grasp and manipulate tools intended for human use and employ these for various tasks (as in Fig. 1) remains one of the big challenges for robots working outside of traditional industrial settings. The application areas range from flexible industrial robots working with tools intended for human use to domestic service robots that perform household chores [1]. In order to provide the input for the control loop guiding the execution of a task, the knowledge of the position of the tool-tip is necessary.

In contrast to classical industrial or other robots with fixed and a priori known tools, it is not realistic to assume that service robots have precise beforehand calibrations of the positions of the tool-tips. Even if they did, the tool may slip and move relative to the gripper while it is being used. Therefore, there is a need for online calibration and/or recalibration of the position of the tip of the tool the robot is using. Current approaches mostly rely on vision-based methods for calibrating the pose and are therefore not applicable in scenarios where the tools or relevant parts of it are occluded.

In this paper, we present an online tool-tip calibration method that uses force/torque measurements and an adaptive estimation scheme to find the point of contact between a tool and the environment. This estimation can be carried out in real-time while the robot is using the tool to perform some arbitrary task, and does not require any predefined model of the shape, size or initial position of the tool being used.

The authors are with the Computer Vision and Active Perception Lab., Centre for Autonomous Systems, School of Computer Science and Communication, Royal Institute of Technology (KTH), SE-100 44 Stockholm, Sweden. e-mail: {yiankar|ccs|fevb|dani}@kth.se

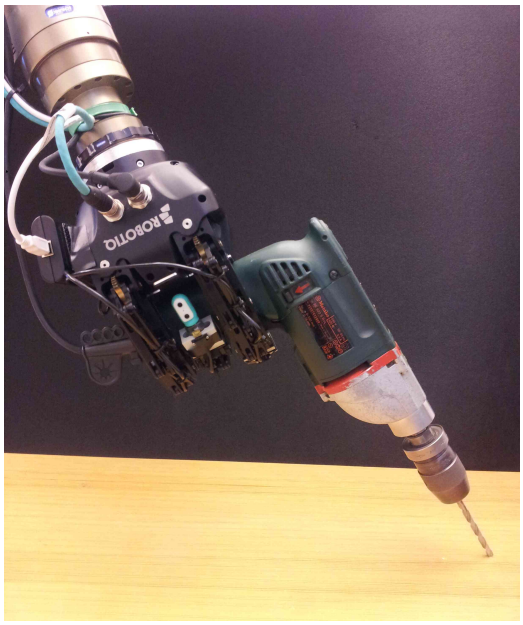


Fig. 1 : Robot manipulating a tool intended for human use.

An adaptive force control component guarantees that interaction forces are limited even before the contact point estimate has converged. We also show how to simultaneously estimate the location and normal direction of the surface being touched by the tool-tip as the contact point is estimated.

The paper has the following structure: Section II reviews the state of the art in related work, Section III formalizes the problem in terms of statics and kinematics, Section IV describes the proposed approach and motivates it theoretically, giving stability and convergence proofs, while Section V describes the experimental implementation of the method on a real robot and the experimental results. Finally, conclusions are presented in Section VI.

II. Related Work

The problem of resolving uncertainties in the end-effector or the tool positioning is well-studied and has been a relevant topic in robotics since the advent of the first manipulators. Early work focused on solving the problem of calibrating the position of the manipulators and end-effectors themselves [2], and this has been extended to also include objects grasped by the robot [3].

Some works treat the problem without explicitly modelling the position of a tool or its point of interaction with the

978-1-4799-3684-7/14/$31.00 ©2014 IEEE

environment, but focus rather on robust performance of a well-defined task under positioning uncertainties. Examples of this includes work by Bruyninckx *et al.* that estimates the alignment error between the peg and the hole for an insertion task [4], work by Hovland and McCarragher that uses a neural network approach to estimate the contact states between two work pieces [5], and work by Koeppe and Hirzinger that learns the appropriate interaction forces for a peg-in-hole task [6].

Some work treats tool-tip position estimation as a calibration problem that can be performed offline prior to using the tool. Yang *et al.* use an iterative least squares approach to calibrate the relationship between tool-tip position and joint angles, using measurements from a known external reference [7]. Cheah *et al.* use adaptive control for tracking a kinematically uncertain manipulator chain, including a tool grasped by the end-effector, but only estimate the kinematics — the position of the end-effector and the tool are measured externally [3].

Others use a particle filter approach to fit a model to the object pose by collecting measurements from touching the object before grasping it [8], [9]. Hebert *et al.* use a fusion approach with vision, force/torque measurements and proprioception to estimate the position of an object with a known model, held in the end-effector [10]. Păiş *et al.* learn relations between a held object and an external tool it interacts with using gaussian mixture models [11].

Atkeson *et al.* have proposed a method for estimating inertial parameters of a grasped object based on force/torque measurements [12], and Kubus *et al.* have used sensor fusion combining measurements of acceleration, velocities, position, forces, and torques to estimate inertial parameters and principal moments of inertia of a grasped object, fitting parameters to estimate object pose [13]. Both these approaches require free motion in a prespecified trajectory, and can not be applied to estimate the contact point of a tool online as it is being used.

Muto and Shimokura have shown a method for estimating contact points given a known tool fixed in the end-effector, using force measurements [14]. Lei *et al.* propose a method that learns model parameters to estimate the position and external force load on a specific non-rigid grinding tool, using proprioception and force/torque measurements [15].

The literature on vision-based object pose estimation and tracking is far too vast to summarize here, but some examples of tool pose estimation based on computer vision methods include work by Kemp and Edsinger who use visual gradients to detect tool-tip positions [16], Krainin *et al.* who simultaneously build an object model and track it in the robot hand [17], and Beetz *et al.*, who use repeated visual template matching to find the location of a spatula in the robot hand [18].

In our previous work, we have shown how an adaptive estimator, integrating proprioception and force/torque measurements can be used to estimate the location of hinges on doors, or the direction of unconstrained motion for drawers that the robot is opening [19], and to estimate slopes on surfaces the robot is in contact with [20]. We have also shown how to learn manipulation affordances and slippage behaviors of held objects by using combined measurements of wrist-based force/torque measurements and grasp forces [21].

In the present paper, we build on these ideas and propose an adaptive controller that estimates the point of contact of the tool with the environment, along with estimating the contact surface normal. This enables the robot to execute a task with the held tool while performing the estimation. The proposed controller limits the interaction forces, to avoid damage to the tool or workpiece while estimates are converging. The proposed method uses force/torque measurements from a wrist-mounted sensor, it is inherently online and fast enough to react to changes, and can thus track the contact point even if it moves relative to the robot hand while executing the task.

III. Kineto-Statics Formulation

Before describing the proposed method for estimating the contact point and the surface normal, we define notation and formalize the relevant first-order differential kinematics and the statics. We assume a system that consists of a robot which performs a task with its tool on a surface; the task requires motion control of the tool contact point and control of the contact forces of the tool on a surface.

A. Notation

First, we introduce the following notation and definitions that will be used throughout this paper:

Bold small letters denote vectors and bold capital letters denote matrices. $\mathbf{I}_\nu$, $\mathbf{O}_\nu \in \mathbb{R}^{\nu\times\nu}$ denote an identity and a square matrix of zeros respectively while $\mathbf{0}_\nu \in \mathbb{R}^\nu$ denotes a column vector of zeros. Hat $\hat{\cdot}$ and tilde $\tilde{\cdot}$ denote estimates and the errors between control variables and their corresponding desired values/vectors respectively. $\mathbf{R}_a$ denotes a rotation matrix that describes the orientation of a frame with respect to the global frame. A left superscript (e.g. ${}^a\mathbf{b}$) denotes the frame (e.g. $\{a\}$) in which a three-dimensional vector (e.g. $\mathbf{b} \in \mathbb{R}^3$) is expressed, e.g. ${}^a\mathbf{b} = \mathbf{R}_a^\top \mathbf{b}$, and it is omitted in case of the global frame. The projection matrix on the orthogonal complement space of a unit three dimensional vector $\mathbf{a}$ is denoted by $\bar{\mathbf{P}}(\mathbf{a}) \triangleq \mathbf{I}_3 - \mathbf{a}\mathbf{a}^\top$. $\mathbf{S}(\mathbf{b})$ denotes the skew-symmetric matrix produced by $\mathbf{b} \in \mathbb{R}^3$ to perform a cross product operation with any three-dimensional vector $\mathbf{a} \in \mathbb{R}^3$ i.e. $\mathbf{b} \times \mathbf{a} = \mathbf{S}(\mathbf{b})\mathbf{a}$.

B. First Order Differential Kinematics

Consider a robot end-effector equipped with a force/torque sensor on its wrist that is grasping a tool which is in contact with a surface, as shown in Fig. 2. We denote with $\{e\}$ a frame attached at a kinematically known position of the end-effector (e.g. center of force/torque sensor) denoted by $\mathbf{p}_e$. We assume that the contact between the tool and the surface is modeled as a contact point that can slide along the surface when the end-effector is moving. At the contact point position $\mathbf{p}_c$, we attach a frame $\{c\}$ with orientation described by $\mathbf{R}_c = [\mathbf{n}_c\ \mathbf{o}_c\ \mathbf{a}_c]$, with $\mathbf{n}_c$ being a unit vector

which is normal to the surface while $\mathbf{o}_c$, $\mathbf{a}_c$ being vectors that span the flat surface and can be arbitrarily chosen. Let $\mathbf{r}$ be the relative position of $\{e\}$ and $\{c\}$ defined as follows:

$$\mathbf{r} = \mathbf{p}_c - \mathbf{p}_e \tag{1}$$

Note that ${}^e\mathbf{r} \triangleq \mathbf{R}_e^\top \mathbf{r}$ is constant if we assume that the end-effector is rigidly grasping the tool. Assuming additionally that only sliding motion is performed i.e. $\dot{\mathbf{p}}_c = \dot{\mathbf{p}}_e$ then $\mathbf{r}$ is constant too and the velocity of the end-effector frame is constrained as follows:

$$\mathbf{n}_c^\top \dot{\mathbf{p}}_e = 0 \tag{2}$$

Note that the assumption of pure sliding motion that simplifies the estimation of the surface slope is not necessary for the main objective of this work which is the contact point estimation. Differentiating (1) implies that the contact point velocity is related to the end-effector velocity as follows:

$$\dot{\mathbf{p}}_c = \mathbf{J}_t \begin{bmatrix} \dot{\mathbf{p}}_e \\ \boldsymbol{\omega}_e \end{bmatrix}, \text{ with } \mathbf{J}_t \triangleq \begin{bmatrix} \mathbf{I}_3 & -\mathbf{S}(\mathbf{r}) \end{bmatrix} \in \mathbb{R}^{3\times 6} \tag{3}$$

being the tool Jacobian matrix and $\boldsymbol{\omega}_e$ being the end-effector rotational velocity i.e. $\mathbf{S}(\boldsymbol{\omega}_e) = \dot{\mathbf{R}}_e \mathbf{R}_e^\top$. By commanding zero rotational velocity and assuming only sliding motion, we can omit the tool Jacobian when mapping the contact point velocities to the joint space. This means that the first order inverse kinematics are given by:

$$\dot{\mathbf{q}} = \mathbf{J}^+ \begin{bmatrix} \mathbf{u} \\ \mathbf{0}_3 \end{bmatrix} \tag{4}$$

where $\mathbf{u}$ is a commanded end-effector or contact point velocity control law and $\mathbf{J}^+$ is the right pseudo-inverse of the end-effector Jacobian with $\mathbf{J}^+ \triangleq \mathbf{J}^\top(\mathbf{J}\mathbf{J}^\top)^{-1}$.

C. Statics

While the end-effector presses with the tool on the surface, the normal force arising (with magnitude denoted by f_n) can – in case of rigid contact – be regarded as a Lagrange multiplier of the controlled system associated to the constraint (2). While the contact point is moving along the surface following the motion of the end-effector, tangential forces $\mathbf{f}_t$ arise owing to dynamic friction components that depend on the sliding velocity of the contact point. The total contact force applied at the contact point is mapped to the end-effector as a wrench consisting of a force vector $\mathbf{f}_c$ and a torque vector $\boldsymbol{\tau}_c$:

$$\mathbf{f}_c = \mathbf{n}_c f_n + \mathbf{f}_t, \tag{5}$$

$$\boldsymbol{\tau}_c = \mathbf{r} \times \mathbf{f}_c. \tag{6}$$

The total force $\mathbf{f}_m$ and torque $\boldsymbol{\tau}_m$ measured by the force/torque sensor (assuming noise-free measurements, and no acceleration of the end-effector) is given by:

$$\mathbf{f}_m = \mathbf{f}_c + \mathbf{f}_g, \tag{7}$$

$$\boldsymbol{\tau}_m = \mathbf{r} \times \mathbf{f}_c + \mathbf{r}_g \times \mathbf{f}_g, \tag{8}$$

where $\mathbf{f}_g$ is the gravity force acting at the center of mass of the object and $\mathbf{r}_g$ is the position of the center of mass with regard to $\mathbf{p}_e$. Note that if $\mathbf{f}_g$ and $\mathbf{r}_g$ are known, gravity compensation can be performed by subtracting them from (7) and (8) to obtain $\mathbf{f}_c$ and $\boldsymbol{\tau}_c$, and thus we can use (5) and (6) to identify $\mathbf{r}$ as proposed in Section IV-A. This gravity compensation can be achieved either by assuming an object with known mass and position with respect to the end-effector, or by considering identification of the gravity effects $\mathbf{f}_g$ and $\mathbf{r}_g$ in a prior free-motion phase where the only force acting on the object is inertial, and $\mathbf{f}_c = \mathbf{0}_3$. In the latter case, proper rotation of the object by the end-effector can generate signals that can be used in the algorithm in Section IV-A to identify $\mathbf{r}_g$ to allow compensation for the gravity during the main contact phase. In case of a lightweight tool, such that $\mathbf{f}_g \ll \mathbf{f}_c$, we can assume that the effects of gravity are within the limits of the measurement errors, and do not need to be compensated for.

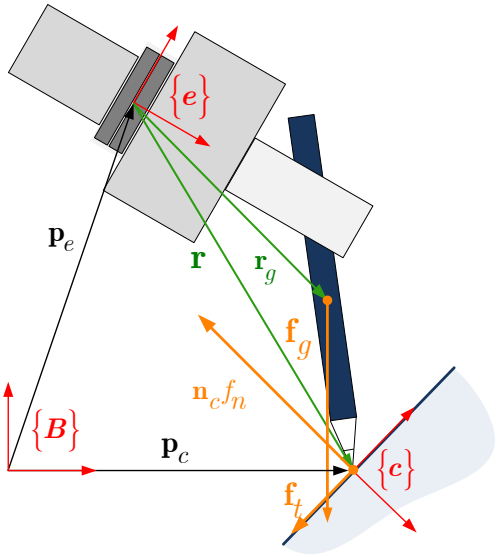


Fig. 2 : A robot end-effector equipped with a force/torque sensor on its wrist that is grasping a tool which is in contact with the surface. Frames are illustrated with red lines. Forces are depicted with orange. Absolute position vectors with respect to the base frame $\{B\}$ are depicted with black lines. Relative positions with respect to the end-effector (sensor) frame are depicted with green lines.

IV. Methodology

In this section we propose the adaptive laws for estimating the contact point and the surface orientation as well as the force/motion control which is based on these estimates. The overall control scheme effectiveness is theoretically justified and the formal proofs of the results are given in the Appendix.

A. Contact Point Estimation

First we design the adaptive law for estimating ${}^e\mathbf{r}$ assuming that ${}^e\mathbf{r}$ is piecewise constant or slowly varying compared to the rate of the estimation. An example of an estimation

rate that can be achieved is demonstrated in Section V. The estimates can also be used to estimate $\mathbf{r} \triangleq \mathbf{R}_e {}^e\mathbf{r}$ in the global frame. The proposed adaptive law utilizes measurements of forces and torques expressed in the end-effector frame, which is assumed to coincide with the force/torque sensor frame, as this can trivially be achieved through known transformations. The law is given by the following equations:

$$ {}^e\dot{\hat{\mathbf{r}}} = -\mathbf{\Gamma}_r \left[\mathbf{L}_r(t) {}^e\hat{\mathbf{r}} - \mathbf{c}_r(t) \right] \tag{9} $$

with

$$ \dot{\mathbf{L}}_r = -\beta_r \mathbf{L}_r - \mathbf{S}({}^e\mathbf{f})\mathbf{S}({}^e\mathbf{f}) \quad \text{with } \mathbf{L}_r(0) = \mathbf{O}_3 \tag{10} $$

$$ \dot{\mathbf{c}}_r = -\beta_r \mathbf{c}_r + \mathbf{S}({}^e\mathbf{f}) {}^e\boldsymbol{\tau} \quad \text{with } \mathbf{c}_r(0) = \mathbf{0}_3 \tag{11} $$

where $\mathbf{\Gamma}_r$ is a positive definite matrix affecting the speed of convergence, β_r is a positive design constant acting as forgetting factor and ${}^e\mathbf{f}$, ${}^e\boldsymbol{\tau}$ are either the measured force and torque after gravity compensation used to estimate ${}^e\mathbf{r}$ during the contact phase, or the measured force and torque owing to gravity used to estimate the center of mass ${}^e\mathbf{r}_g$ in the free-motion phase.

Proposition 1: The adaptive estimation law (9)-(11) guarantees that:

(i) the torque estimation error, the estimate ${}^e\hat{\mathbf{r}}$ and its derivative are bounded i.e. ${}^e\boldsymbol{\tau} - {}^e\hat{\boldsymbol{\tau}}$, ${}^e\hat{\mathbf{r}}$, ${}^e\dot{\hat{\mathbf{r}}} \in \mathcal{L}_\infty$,
(ii) the torque estimation error and and the estimation rate ${}^e\dot{\hat{\mathbf{r}}}$ are square integrable, i.e. ${}^e\boldsymbol{\tau} - {}^e\hat{\boldsymbol{\tau}}$, ${}^e\dot{\hat{\mathbf{r}}} \in \mathcal{L}_2$,
(iii) $\lim_{t\to\infty} {}^e\hat{\boldsymbol{\tau}} = {}^e\boldsymbol{\tau}$ and $\lim_{t\to\infty} \|\dot{\hat{\mathbf{r}}}\| = 0$, and
(iv) if $\mathbf{S}({}^e\mathbf{f})$ is persistently excited (PE) ${}^e\hat{\mathbf{r}}$ converges exponentially to ${}^e\mathbf{r}$. By choosing $\mathbf{\Gamma}_r = \gamma_r \mathbf{I}$, with γ_r being positive constant, the speed of convergence can be arbitrarily increased by increasing γ_r.

The proposed estimator (9)-(11) is an integral adaptive control law since its design is based on the minimization of an integral cost function of the error between the actual and the estimated torque [22]; the proof of the Proposition 1 is following the proof of the integral adaptive control for identifying the parameters in a multiple inputs-single output parametric model and is based on the use of a quadratic Lyapunov function $V({}^e\tilde{\mathbf{r}}) = \frac{1}{2} {}^e\tilde{\mathbf{r}}^\top \mathbf{\Gamma}_r^{-1} {}^e\tilde{\mathbf{r}}$.

As is demonstrated in Section V, convergence to the actual parameters can be achieved by varying the direction of $\mathbf{f}_e$ in order to span some surface in the Cartesian space, which is an identification condition arising from the problem formulation.

The contact point estimate $\hat{\mathbf{p}}_c$ can be calculated using proprioception and the estimate ${}^e\mathbf{r}$ produced by exploiting force/torque measurements:

$$ \hat{\mathbf{p}}_c = \mathbf{p}_e + \mathbf{R}_e {}^e\hat{\mathbf{r}} \tag{12} $$

Note that it is also possible to use the contact point estimation together with an accurate model of the grasped tool to determine which of a possible set of points is in contact. In this case, we can additionally infer the orientation of the tool given that the grasping point is obtained through tactile sensing.

The adaptive law can also be used to identify the center of mass in case of free-space motion. The parameters are identified exponentially fast given that ${}^e\mathbf{f} = \mathbf{R}_e^\top \mathbf{f}$ is PE. Note that $\mathbf{f} = \mathbf{f}_g$ is constant and thus the identification is excited by the rotational motion of the object.

B. Surface Normal Estimation

In order to estimate the surface normal direction we design the following adaptive law:

$$ \dot{\hat{\mathbf{n}}}_c = -\gamma_n \bar{\mathbf{P}}(\hat{\mathbf{n}}_c) \mathbf{L}_n(t) \hat{\mathbf{n}}_c \tag{13} $$

$$ \dot{\mathbf{L}}_n = -\beta_n \mathbf{L}_n + \tfrac{1}{1+\|\dot{\mathbf{p}}_e\|^2} \dot{\mathbf{p}}_e \dot{\mathbf{p}}_e^\top \quad \text{with } \mathbf{L}_n(0) = \mathbf{O}_3 \tag{14} $$

where γ_n is a positive constant for tuning the speed of convergence and β_n is a positive forgetting factor.

Proposition 2: The adaptive law (13)-(14) guarantees that:

(i) the norm of the estimate $\hat{\mathbf{n}}_c(t)$ is invariant i.e. given that $\|\hat{\mathbf{n}}_c(0)\| = 1$, $\|\hat{\mathbf{n}}_c(t)\| = 1, \forall t > 0$,
(ii) if $\vartheta(0) \in (-\frac{\pi}{2}, \frac{\pi}{2})$ then $\vartheta(t) \in (-\frac{\pi}{2}, \frac{\pi}{2}), \forall t > 0$ where ϑ is the angle formed between $\mathbf{n}_c$ and $\hat{\mathbf{n}}_c$,
(iii) $\lim_{t\to\infty} \|\dot{\hat{\mathbf{n}}}_c\| = 0$, and
(iv) if $\dot{\mathbf{p}}_e$ is persistently excited (PE) ϑ converges to zero exponentially which implies that $\hat{\mathbf{n}}_c$ converges exponentially to $\mathbf{n}_c$ with a rate that can be tuned by γ_n.

The proposed estimator (13)-(14) is an integral adaptive control – in contrast to those used in our previous work [19], [20]– with normalized input but here is modified in order to produce unit and well-defined estimates of the normal direction as the problem in hand requires. The proof of Proposition 2 can be found in the Appendix, and is based on defining the Lyapunov function in the domain of the *estimation error angle* ϑ formed between $\mathbf{n}_c$ and $\hat{\mathbf{n}}_c$.

Measurements of the contact force $\mathbf{f}_c$ alone cannot in general be used together with (5) to identify the surface normal if the contribution from the tangential force $\mathbf{f}_t$ due to friction is unknown. However, we can use the force measurements in order to initialize the proposed estimator when contact is detected i.e. $\hat{\mathbf{n}}_c(0) = \mathbf{f}(0)/\|\mathbf{f}(0)\|$. Given that the gravity is compensated in $\mathbf{f}(0)$, the initial angle error will be within the cone of friction which implies that $|\vartheta(0)| < \pi/2$ and consequently that the estimator is properly initialized. If there is no rotational motion of the end-effector, the sliding velocity of the tool-tip on the surface is equal to the end-effector velocity, and thus the latter can be directly used to estimate the surface normal direction, independent of the accuracy of the contact point estimate.

C. Force/motion Control

The control objective is to follow a velocity trajectory $\mathbf{v}_d(t)$ and to press upon the surface with a desired force f_d. In this way, we can perform a meaningful task and simultaneously generate signals ${}^e\mathbf{f}$ and $\dot{\mathbf{p}}_e$. In particular, the motion along the surface not only generates $\dot{\mathbf{p}}_e$ that span the orthogonal complement of the normal direction required in (13) but gives rise to tangential forces owing to dynamical friction that can be added to the normal interaction forces,

see (5), in order to generate an appropriate signal ${}^e\mathbf{f}$ to excite (9) by spanning a surface in the Cartesian space.

The velocity control design is based on decomposing the motion and force control directions according to hybrid force/motion control methodology by using however the estimates $\hat{\mathbf{n}}_c$ – like the kinematic loop of [20]. The proposed kinematic controller is given by the following equation:

$$\mathbf{u} = \bar{\mathbf{P}}(\hat{\mathbf{n}}_c)\mathbf{v}_d - \hat{\mathbf{n}}_c v_f \tag{15}$$

where v_f is a PI control loop of the estimated force error $\tilde{\hat{f}}_n = \hat{f}_n - f_d$. Note that the estimated $\hat{f}_n$ can be calculated based on force measurements and the online estimates $\hat{\mathbf{n}}_c$. In particular:

$$v_f = \alpha_I \int_0^t (\hat{f}_n - f_d)d\tau + \alpha_P(\hat{f}_n - f_d), \quad \hat{f}_n = \hat{\mathbf{n}}_c^\top \mathbf{f} \tag{16}$$

with α_I and α_P are positive control gains.

The velocity trajectory $\mathbf{v}_d(t)$ can be either defined a priori in a feedforward fashion or designed appropriately by using feedback of control errors such as end-effector or contact point position errors. A simple way to define $\mathbf{v}_d(t)$ is the following:

$$\mathbf{v}_d(t) = \dot{\mathbf{p}}_d - \alpha(\mathbf{p} - \mathbf{p}_d) \tag{17}$$

where α is positive control gain and $\mathbf{p}$ can be either the end-effector position or the contact point estimate depending on the definition of the desired position $\mathbf{p}_d$. Note that $\mathbf{p}_d$ can be defined as follows:

1) directly and a priori in the robot workspace e.g. by using vision and mapping a desired trajectory from the image space to robot space. In this case the feedback control is designed using the contact point position which is however based on estimates obtained by (9) i.e. $\mathbf{p} := \hat{\mathbf{p}}_c$.
2) locally at the surface as $\xi_d \in \mathbb{R}^2$ and then mapped online to the robot workspace through a transformation $(\mathbf{p}_e(0), \hat{\mathbf{R}}_c)$. Details on the motivation behind this selection and the construction of $\hat{\mathbf{R}}_c$ can be found in [20]. In this case the design of $\mathbf{v}_d(t)$ is based on $\mathbf{p}_e$ instead of $\hat{\mathbf{p}}_c$. This can be explained by the following observation: the objective of drawing a circle with center around the initial contact point is equivalent of drawing a virtual circle around the end-effector's initial position.

In more complicated scenarios where both sliding and rolling motion of the tool take place the estimated contact point position must be used in $\mathbf{v}_d(t)$ even when the target is defined based on local coordinates.

Analysis of the closed loop when $\mathbf{u}$ (15)–(17) is applied (briefly sketched in the Appendix) yields to the following theorem:

Theorem 1: The control law (15)–(17) applied as a translational velocity controller to a robot firmly grasping a tool which is in contact with a flat surface, together with the adaptive laws (9) and (13) used for estimating the contact parameters such as contact point position and surface orientation ensure the boundedness of the contact force and the velocity along the unconstrained directions as well as the convergence of the force/motion errors to zero and the identification of the uncertain parameters given that $\mathbf{u}$ and ${}^e\mathbf{f}$ are persistently excited.

V. Experiments

Our adaptive control framework for tool and surface calibration was evaluated with a robot setup consisting of a 7-DOF velocity controlled manipulator controlled at 130 Hz. The manipulator also includes a wrist mounted ATI Mini45 6-axis force-torque sensor used for the force feedback control and the contact point estimator.

For more details on the experimental setup, see e.g. [23]. The end-effector velocities used for the estimation of the surface normal were calculated using joint velocities filtered through joint position measurements.

Fig. 3 : Experimental setup used for evaluating our adaptive control scheme for contact point and surface normal estimation.

To perform our experiments we attached a tool rigidly to the robot's gripper as shown in Fig. 3. Attaching the tool rigidly to the end-effector allowed us to have a consistent ground truth with which to compare the controller's estimation of the contact point. Furthermore, we tested the controller over a flat table which was previously calibrated to obtain the ground truth of the surface normal. A circular trajectory of 4 cm radius and 5 second period was commanded to the manipulator during the experiment.

Fig. 4 shows the normal force error $|\tilde{f}_n| = |f_n - f_d|$ which indicates that the adaptive controller manages to regulate the normal contact force.

Fig. 5 and 6 show the estimation errors of the contact point and the surface normal respectively. The contact point converges with an error of approximately 5 mm, which, given

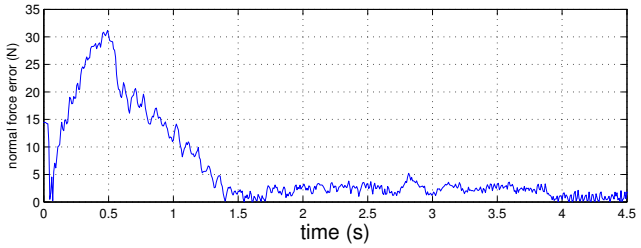


Fig. 4 : Normal force error $|\tilde{f}_n|$.

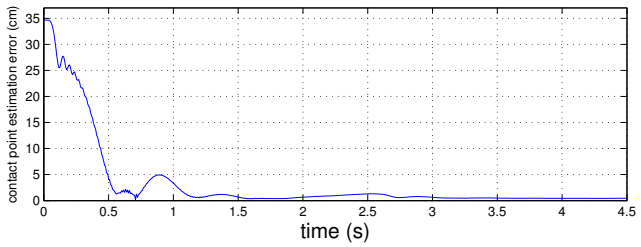


Fig. 5 : Contact point estimation error.

the 30.8 cm distance from the force-torque sensor to the tool-tip is within the error margins of the setup. Moreover, the surface normal estimate converges with an approximately 1.5 degree error with respect to the ground truth normal.

For a set of contact forces that share the same direction and only vary in magnitude, the contact point estimate $\hat{\mathbf{p}}_c$ will converge to a point on a line passing through the actual contact point $\mathbf{p}_c$, parallel to the force direction. As the direction of contact forces vary, $\hat{\mathbf{p}}_c$ will converge to the intersection point of a set of such lines, which will be $\mathbf{p}_c$. In the experimental convergence of $\hat{\mathbf{p}}_c$, as seen in Fig. 5, we see an initial convergence to a point on such a line after approx 0.1 s, and then, as the direction of the contact force starts to change as the tool-tip slides on the surface, we see convergence to the intersection point, or $\mathbf{p}_c$. For the setup in the experiment we see that we have good convergence for forces that spread over approximately 14 degrees with respect to the surface normal. For faster convergence with the same setup, we would need contact forces spanning that angle variation in shorter time.

VI. Conclusions

In this paper, we have proposed a method for simultaneous online estimation of the point of contact of a tool held by a robot, and the normal of the surface it is interacting with. The method is based on adaptive estimation and a hybrid force/motion controller, and uses force and torque measurements from a wrist-mounted sensor.

The fast convergence of the contact point estimate makes it suitable for real time tracking of the endpoint of a tool that may slip and move in the robot's hand as it is being used for a task execution. The method also guarantees stable control of contact forces even before the estimates converge. For non-contact motion, the method can be used to estimate the center of mass of the end-effector and/or a grasped object.

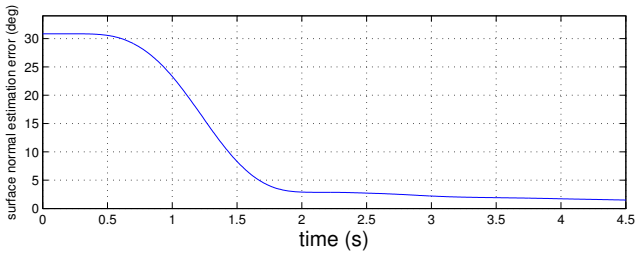


Fig. 6 : Surface normal estimation error.

This enables tool use with unmodelled and/or uncalibrated tools.

The strength of the method lies in the fact that it uses force and torque measurements and it is therefore complimentary to vision based approaches where occluded or bad lighting conditions make affect the estimation. An interesting future extension is to combine the proposed method with other methods for object tracking. One possibility is to combine it with model based vision methods or tactile sensors to use the tracked contact point to improve pose tracking of an object.

Appendix

Proof of Proposition 2: (i) Note that $\frac{d}{dt}\left(\|\hat{\mathbf{n}}_c\|^2\right) = -\gamma_n[\bar{\mathbf{P}}(\hat{\mathbf{n}}_c)\hat{\mathbf{n}}_c]^\top \mathbf{L}_n(t)\hat{\mathbf{n}}_c = 0$; thus the norm of the estimate is invariant and consequently bounded. (ii) Note that (14) and the constraint (2) implies:

$$\frac{d}{dt}\left(\mathbf{L}_n\mathbf{n}_c\right) = -\beta_n\mathbf{L}_n\mathbf{n}_c$$

which in turn, for $\mathbf{L}_n(0) = \mathbf{O}_3$, implies that $\mathbf{n}_c$ belongs in the nullspace of $\mathbf{L}_n$. Note also that $\mathbf{L}_n$ is positive semidefinite. Consider the following Lyapunov function:

$$V(\theta) = -\ln(\cos\vartheta), \quad V : (-\tfrac{\pi}{2}, \tfrac{\pi}{2}) \to \mathbb{R} \tag{18}$$

Its time derivative along the systems trajectories (13)-(14) is given by:

$$\dot{V}(\theta, t) = -\gamma_n\tilde{\mathbf{n}}_c^\top\mathbf{L}_n(t)\tilde{\mathbf{n}}_c \tag{19}$$

From (18) and (19) we conclude that $V(\theta)$ is bounded which implies $\vartheta(t) \in (-\frac{\pi}{2}, \frac{\pi}{2}), \forall t > 0$ for $\vartheta(0) \in (-\frac{\pi}{2}, \frac{\pi}{2})$. (iii) Clearly (i) and (ii) imply that $\hat{\mathbf{n}}_c \in \mathcal{L}_\infty$. Furthermore, since $\frac{1}{1+\|\dot{\mathbf{p}}_e\|^2}\dot{\mathbf{p}}_e\dot{\mathbf{p}}_e^\top$ is bounded by construction, (14) implies that $\mathbf{L}_n$ and $\dot{\mathbf{L}}_n$ are bounded too. By integrating both sides of (19) in $t \in [0, \infty)$ and taking into account that $V(\infty)$ is bounded from (ii), we get that $\mathbf{L}_n^{1/2}\tilde{\mathbf{n}}_c \in \mathcal{L}_2$. The update law (13) implies: (a) $\|\dot{\hat{\mathbf{n}}}_c\| \leq \gamma_n\|(\mathbf{L}_n^\top)^{1/2}\|\|\mathbf{L}_n^{1/2}\tilde{\mathbf{n}}_c\|$ which implies that $\dot{\hat{\mathbf{n}}}_c \in \mathcal{L}_\infty \bigcap \mathcal{L}_2$ as well as (b) $\ddot{\hat{\mathbf{n}}}_c$ given the boundedness of $\dot{\hat{\mathbf{n}}}_c$ and $\dot{\mathbf{L}}_n$. Clearly (a) and (b) yield $\lim_{t\to\infty}\|\dot{\hat{\mathbf{n}}}_c\| = 0$. (iv) The PE condition is satisfied given that there exists α_0, T_0 such that $\int_t^{t+T_0}\dot{\mathbf{p}}_e\dot{\mathbf{p}}_e^\top d\tau \geq \alpha_0\mathbf{I}_3$. Using the integral expression of $\mathbf{L}_n = \int_0^t \exp(-\beta_n(t-\tau))\frac{1}{1+\|\dot{\mathbf{p}}_e\|^2}\dot{\mathbf{p}}_e\dot{\mathbf{p}}_e^\top d\tau$ implied by (14), it can be found that $\mathbf{L}_n(t) \geq \lambda\exp(-\beta_n T_0)\mathbf{I}_3$ given that the PE condition is satisfied. Then we consider a quadratic Lyapunov function $U = \frac{1}{2}\|\tilde{\mathbf{n}}_c\|^2$ which in our case depends

only on the estimation error angle i.e. $U = 1 - \cos\vartheta$. It can be easily proved that $\dot{U} \leq -2\lambda\gamma_n \exp(-\beta_n T_0)U$

$$U(t) \leq \exp\left(-2\lambda\gamma_n e^{-\beta_n T_0} t\right) U(T_0) \tag{20}$$

Note also that $\frac{4}{\pi^2}\|\vartheta\|^2 \leq U(\theta) \leq \frac{1}{2}\|\vartheta\|^2$ and thus (21) implies that the angle between the actual and the estimated vector $\hat{\mathbf{n}}_c$, converges exponential to zero as follows:

$$|\vartheta(t)| \leq \frac{\pi\sqrt{2}}{4} \exp\left(-\lambda\gamma_n e^{-\beta_n T_0} t\right) |\vartheta(T_0)| \tag{21}$$

Proof of Theorem 1: Let us consider the case of a bounded input $\mathbf{v}_d$. This assumption is valid even when $\mathbf{v}_d$ is defined using feedback given that $\mathbf{p}_d$, $\dot{\mathbf{p}}_d$ are bounded and by considering a bounded robot workspace. Saturation on the position error can be used in order to construct a bounded $\mathbf{v}_d$. Substituting the control law in to the system first order differential kinematics (4) we get: $\dot{\mathbf{p}}_e = \mathbf{u}$. By projecting the aforementioned equation along the surface normal we get: $v_f = \frac{1}{\cos\vartheta}\mathbf{n}_c^\top \bar{\mathbf{P}}(\hat{\mathbf{n}}_c)\mathbf{v}_d$. Since $\hat{\mathbf{n}}_c$ is bounded and $\vartheta \neq \pi/2$ from (i) and (ii) of Proposition 2, v_f is bounded. The boundedness of v_f implies $\dot{\mathbf{p}}_e$ is bounded and additionally that $\int_0^t (\hat{f}_n - f_d)d\tau$, $\hat{f}_n$ are bounded. Hence ${}^e\mathbf{f}$ is bounded and thus the update law for ${}^e\hat{\mathbf{r}}$ (9)-(11) is well-defined. The boundedness of $\mathbf{p}_e$ can be proved by using the boundedness of the estimates ${}^e\hat{\mathbf{r}}$, $\hat{\mathbf{n}}_c$ and their derivatives ${}^e\dot{\hat{\mathbf{r}}}$, $\dot{\hat{\mathbf{n}}}_c$. Ultimate bounds can also be found by exploiting $\lim_{t\to\infty} {}^e\dot{\hat{\mathbf{r}}} = \mathbf{0}_3$, $\lim_{t\to\infty} \dot{\hat{\mathbf{n}}}_c = \mathbf{0}_3$ (Proposition 1 and 2); analytic derivations are omitted. Given that ${}^e\mathbf{f}$, $\dot{\mathbf{p}}_e$ (or $\mathbf{v}_d$) satisfy the PE condition the estimation error converges to zero exponentially fast and thus v_f converges exponential fast to zero which implies $\int_0^t (\hat{f}_n - f_d)d\tau \to 0$ and $f_n \to f_d$. Furthermore, $\dot{\mathbf{p}}_e \to \bar{\mathbf{P}}(\mathbf{n}_c)\mathbf{v}_d$ with implies that $\bar{\mathbf{P}}(\mathbf{n}_c)(\mathbf{p} - \mathbf{p}_d) \to \mathbf{0}_3$.

ACKNOWLEDGMENT

This work has been supported by the Swedish Research Council (VR), the European Union FP7 project RoboHow.Cog (FP7-ICT-288533), and the Swedish Foundation for Strategic Research. The authors gratefully acknowledge the support.

REFERENCES

[1] C. Kemp, A. Edsinger, and E. Torres-Jara, "Challenges for robot manipulation in human environments [grand challenges of robotics]," *IEEE Robotics Automation Magazine*, vol. 14, no. 1, pp. 20–29, 2007.

[2] Z. S. Roth, B. Mooring, and B. Ravani, "An overview of robot calibration," *IEEE Journal of Robotics and Automation*, vol. 3, no. 5, pp. 377–385, 1987.

[3] C. Cheah, C. Liu, and J. Slotine, "Adaptive jacobian tracking control of robots with uncertainties in kinematic, dynamic and actuator models," *Automatic Control, IEEE Transactions on*, vol. 51, no. 6, pp. 1024–1029, 2006.

[4] H. Bruyninckx, S. Dutre, and J. De Schutter, "Peg-on-hole: a model based solution to peg and hole alignment," in *IEEE International Conference on Robotics and Automation*, vol. 2, 1995, pp. 1919–1924.

[5] G. Hovland and B. J. McCarragher, "Combining force and position measurements for the monitoring of robotic assembly," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, 1997, pp. 654–660.

[6] R. Koeppe and G. Hirzinger, "Sensorimotor compliant motion from geometric perception," in *IEEE International Conference on Intelligent Robots and Systems*, vol. 2, 1999, pp. 805–811.

[7] G. Yang, I.-M. Chen, S. H. Yeo, and W. K. Lim, "Simultaneous base and tool calibration for self-calibrated parallel robots," *Robotica*, vol. 20, pp. 367–374, 7 2002.

[8] A. Petrovskaya, O. Khatib, S. Thrun, , and A. Y. Ng, "Touch based perception for object manipulation," in *Robotics Science and Systems Conference, Robot Manipulation Workshop*, Atlanta, GA, 2007.

[9] C. Corcoran and R. Platt, "A measurement model for tracking hand-object state during dexterous manipulation," in *IEEE International Conference on Robotics and Automation*, 2010, pp. 4302–4308.

[10] P. Hebert, N. Hudson, J. Ma, and J. Burdick, "Fusion of stereo vision, force-torque, and joint sensors for estimation of in-hand object location," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 5935–5941.

[11] L. Păiş, K. Umezawa, Y. Nakamura, and A. Billard, "Learning robot skills through motion segmentation and constraints extraction," *ACM/IEEE International Conference on Human-robot Interaction, Workshop on Collaborative Manipulation*, 2013.

[12] C. Atkeson, C. An, and J. Hollerbach, "Rigid body load identification for manipulators," in *24th conf on Decision and Control*, Fort Lauderdale, Fl, Dec 1985, pp. 996–1003.

[13] D. Kubus, T. Krüger, and F. M. Wahl, "On-line rigid object recognition and pose estimation based on inertial parameters," in *EEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, CA, 2007, pp. 1402–1408.

[14] S. Muto and K. Shimokura, "Teaching and control of robot contour-tracking using contact point detection," in *IEEE International Conference on Robotics and Automation*, 1994, pp. 674–681.

[15] Y. Lei and S. F. Miller, "Pose estimation and machining efficiency of an endoscopic grinding tool," *The International Journal of Advanced Manufacturing Technology*, pp. 1–11, 2013.

[16] C. C. Kemp and A. Edsinger, "Robot manipulation of human tools: Autonomous detection and control of task relevant features," in *5th IEEE International Conference on Development and Learning (ICDL-06*, 2006.

[17] M. Krainin, P. Henry, X. Ren, and D. Fox, "Manipulator and object tracking for in hand model acquisition," in *Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation at the Int. Conf. on Robotics & Automation (ICRA), Anchorage, Alaska*, 2010.

[18] M. Beetz, U. Klank, I. Kresse, A. Maldonado, L. Mosenlechner, D. Pangercic, T. Ruhr, and M. Tenorth, "Robotic roommates making pancakes," in *IEEE-RAS International Conference on Humaoid Robots*, 2011, pp. 529–536.

[19] Y. Karayiannidis, C. Smith, F. Viña, P. Ögren, and D. Kragic, "Model-free robot manipulation of doors and drawers by means of fixed-grasps," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 4470–4477.

[20] Y. Karayiannidis and Z. Doulgeri, "Adaptive control of robot contact tasks with on-line learning of planar surfaces," *Automatica*, vol. 45, no. 10, pp. 2374–2382, 2009.

[21] F. Viña, Y. Bekiroglu, C. Smith, Y. Karayiannidis, and D. Kragic, "Predicting slippage and learning manipulation affordances through gaussian process regression." in *IEEE-RAS International Conference on Humanoid Robots*, 2013.

[22] P. A. Ioannou and J. Sun, *Robust Adaptive Control*. Upper Saddle River, NJ:Prentice Hall, 1996.

[23] C. Smith and Y. Karayiannidis, "Optimal command ordering for serial link manipulators," in *IEEE-RAS International Conference on Humanoid Robots*, 2012, pp. 255–261.

2014 14th IEEE-RAS International Conference on
Humanoid Robots (Humanoids)
November 18-20, 2014. Madrid, Spain

Learning to Disambiguate Object Hypotheses through Self-Exploration

Mårten Björkman and Yasemin Bekiroglu

***Abstract*— We present a probabilistic learning framework to form object hypotheses through interaction with the environment. A robot learns how to manipulate objects through pushing actions to identify how many objects are present in the scene. We use a segmentation system that initializes object hypotheses based on RGBD data and adopt a reinforcement approach to learn the relations between pushing actions and their effects on object segmentations. Trained models are used to generate actions that result in minimum number of pushes on object groups, until either object separation events are observed or it is ensured that there is only one object acted on. We provide baseline experiments that show that a policy based on reinforcement learning for action selection results in fewer pushes, than if pushing actions were selected randomly.**

I. Introduction

One of the core challenges in the field of robotics is to equip robots with the ability to intelligently interact with the world, using sensorimotor capabilities comparable to those of humans. To achieve this, a robot needs to perceive and interpret the environment and act according to the situations it is engaged in. The robot should be able to gather and interpret sensory information in new, unforeseen situations being provided with only minimal prior knowledge. In many practical scenarios, a key requirement for a robot is the ability to detect, recognize and manipulate objects, autonomously or in collaboration with humans or other robots. To develop cognitive and behavioral capabilities in such a context, it is essential for the robot to form useful object representations or categories by being actively engaged in the environment.

Theories of cognitive development describe basic overlapping stages that infants go through as they mentally mature, stages during which they try to make sense of the world actively, rather than just passively observe [1]. They construct their own knowledge from experimenting on the world and independently learn concepts without intervention of or rewards from adults. Similar theories have inspired a field known as developmental robotics, in which emphasis is placed on the necessity for robots to acquire skills through interaction and only rely on knowledge that it can itself verify [2]. Concepts acquired thus emerge as a result of the robot's own interaction with the world and its embodiment.

Our goal is to develop an embodied cognitive system for a robot to continuously learn low-level sensorimotor abilities to understand its environment by exploration. Given a robot interacting with objects placed on a table-top, we present a reinforcement learning approach, where pushing actions are chosen to maximize rewards that are given when events of object separation are observed, this in order to deduce the number of objects being present. The robot gathers data by applying random pushing actions on groups of objects and observing the outcome of the actions based on visual sensing, see Fig 1. After completing the learning phase on the collected observations, the robot builds initial hypotheses of objects in a given scene, hypotheses are then refined by a more strategic manipulation.

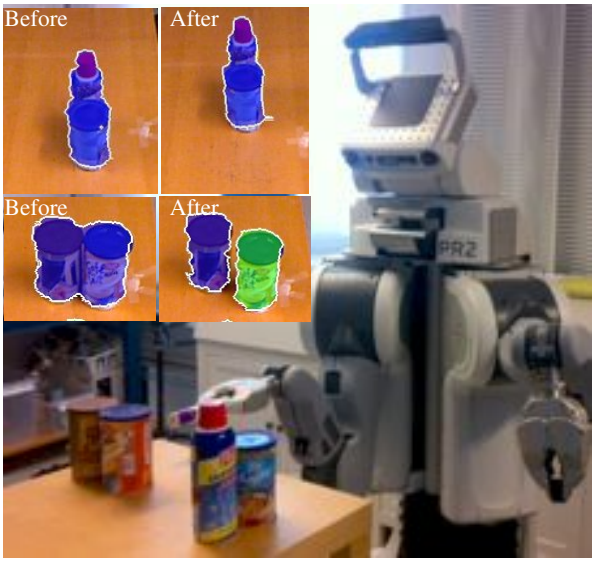


Fig. 1: *Difference in outcomes of an applied pushing action: A PR2 robot explores a given scene by pushing to understand how many objects are present. Depending on object configurations, the same action can yield different outcomes. The two objects in the top row cannot be separated and are thus grouped together as one object after the action, while the other two objects can be separated after the same action.*

II. Related work and Contributions

Our work is related to interactive object segmentation and object exploration through manipulation. In this section we will provide a brief overview regarding these research areas and present our contributions.

Segmentation in static images has been studied extensively for many years [3], [4]. Segmentation methods provide partitioning of the image into regions, typically using similarity in color and position. However, resulting regions do not necessarily correspond to physical objects. The goal of the object segmentation methods is to distinguish objects from

The authors are with the Centre for Autonomous Systems and the Computer Vision and Active Perception Lab, CSC, KTH Royal Institute of Technology, Stockholm, Sweden. Email: {celle|yaseminb}@kth.se. This work was supported by the Swedish Research Council and the EU projects eSMCs (FP7-IST-270212) and RoboHow.Cog (FP7-ICT-288533).

the background they are placed in-front. A common approach is to allow user feedback that provides information about regions of interest [5] to include target objects and thus improve segmentation. Bergström et al. [6] follow such an approach and present a Markov random field based segmentation framework that involves a human operator to guide gradual splitting of segments without explicitly stating how. A human operator helps for interaction with the scene to disambiguate the hypotheses. There are also studies where a robotic arm manipulation is used for interactive segmentation [7]–[9]. Kenny et al. [7] and Fitzpatrick [8] address segmentation of rigid objects based on a video stream of objects being moved by a robot and the segmentation method is based on a graph cut algorithm. Hausman et al. [9] use 3D data to segment objects from a background. Their system includes a static pre-segmentation from geometrical categorization, RGBD features for tracking textureless objects and a graph-based trajectory clustering method. Van Hoof et al. [10] propose a probabilistic approach by quantifying the expected informativeness of actions in information-theoretic terms to select actions. Chang et al. [11] present a framework for representing interaction strategies for singulation of objects. The strategies allow for object selection, manipulation primitive selection and a scene state update. A set of heuristics is used for the manipulation primitive selection step. They determine a likelihood ratio of a target being a single item or multiple items based on the magnitude of the transform motion and the percentage of dense point matches. Hermans et al. [12] achieve singulation of objects through pushing actions. The approach explicitly hypothesizes about object orientation and location of potential separation boundaries between possible objects using input from an RGBD camera.

Pushing actions have been used for learning different object attributes during scene exploration. Ridge et al. [13] propose to ground features of objects with respect to their manipulation in order to compare unknown objects and the effects of manipulation. Object push affordances are learned based on pre-push and post-push object features and push action trajectories. Pushing experiments are performed by a human. Högman et al. [14] discover functional categories based on how objects move during pushing experiments and use an entropy-based action selection approach for object classification. Sanchez-Fibla et al. [15] represent objects in terms of affordance gradients that capture both the shape of objects and the effect of a push applied to different positions of the objects. Kopicki et al. [16] study the problem of learning to predict the interactions of rigid bodies in a probabilistic framework. A robotic arm applies random pushes to objects, observes the resulting motion with a vision system and learns the relation between push actions and the object motions.

Our work differs from the state of the art since our method:

1) does not rely on human feedback for refinement of object segmentation,
2) uses Gaussian process models to learn the effects of pushing actions applied to object groups, and
3) applies reinforcement learning to autonomously learn how to most effectively disambiguate between objects initially grouped into single hypotheses.

III. Learning Framework

We assume that the robot has some ability to generate hypotheses of objects in the scene without having any stored knowledge of categories or instances of objects. Naturally, given no prior knowledge, multiple objects standing close by and overlapping in the visual field, may look like a single self-occluding articulated object. Thus, there is no guarantee that the separation of these objects will in fact be successful. The task the robot is then facing is to learn how to interact with the scene an objects by pushing given as few pushes as possible until a separation of objects can be detected.

A. Forward models

The learning framework is based on the notion of object-related sensorimotor contingencies [17], which are relations that describe predicted sensory changes due to actions applied to physical objects. Given some object-related sensory space $\mathcal{S}$ and an action $a \in \mathcal{A}$, where A is here limited to pushing actions, the observation $s \in \mathcal{S}$ of some manipulated objects changes to a new observation $s' \in \mathcal{S}$ as a result of applying a. Such a relation can be described by a forward model [18] that can often be represented by some function $f : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$. Since visual observations are typically subjected to considerable noise, the observed outcome of an action can hardly be expected to be deterministic. Instead we apply probabilistic models for the same purpose and assume these to be Gaussian, where the mean and variance depend on s and a, that is

$$P(s'|s,a) = \mathcal{N}(\mu(s,a), \sigma^2(s,a)).$$

These distributions are found by means of Gaussian process regression [19] using squared-exponential covariance functions

$$k(s_i, s_j) = \sigma_f^2 \exp(-\frac{1}{2}(s_i - s_j)^\top \Lambda^{-1}(s_i - s_j)) + \sigma_n^2 \delta_{ij},$$

where s_i and s_j are two different observations, σ_f^2 is the signal variance, σ_n^2 is the noise variance and Λ is a diagonal matrix of length scales for the different sensory components of s. The unknown parameters are found through maximum likelihood optimization, using a previously recorded set of observations of pushes, represented as tuples (s, a, s').

B. Sensory and action spaces

For the experiments in Section IV, we use a sensory space $\mathcal{S}$ with measurements derived from physical objects observed in RGBD images. These objects, or possibly groups of objects, are segmented from the rest of the scene by first detecting a table-top plane and removing 3D points located below that plane. Using 2D connectivity, points within a reachable distance are then grouped into segments, where segments belonging to the robot itself are identified using the robot's known forward kinematics. These segments are pruned from further consideration, resulting in a remaining set of segments for the objects placed on the table-top.

For each such segment, a vector of four measurements

$$s = (s_{shape}, s_{fragm}, s_{circum}, s_{orient})^\top$$

is computed, measurements that are directly derived from the sensory data. These can be seen as a low-dimensional representation of the raw data, without any assumptions made on what is actually observed. The representation is respectively given by the shape of the segment's image projection computed as the length of minor axis divided by the major one (s_{shape}), the spread of depth values computed as an entropy (s_{fragm}), the circumference divided by the square root of the area of the segment in question (s_{circum}) and the orientation of the 3D point cloud projected down to the 2D table-top (s_{orient}). In practice, the orientation is represented by two values

$$s_{orient} = (\sin(2\alpha), \cos(2\alpha))^\top,$$

where α is the angle between the lateral direction of the robot and dominating one of the point cloud. Due to the $180°$ periodicity that occurs when the frontside of an object is replaced by the backside as it is turned, the double angle is used instead. While some sensory dimensions are believed to change as a function of the push applied, others should give an indication of object separation that can be used for action planning.

To simplify analysis of data and speed up the learning process, we use an constrained action space $\mathcal{A} = \{a_l, a_f, a_r\}$ of three possible pushes; left-side, forward and right-side pushes. Each push is done towards the centroid of the 3D point cloud representing the object group under consideration and is extended 2 cm beyond the centroid to guarantee contact with the robot's end-effector. The sideways pushes are at an angle of $40°$ with respect to the forward direction.

C. Reinforcement learning

By collecting multiple observations of pushes applied to different configurations of object groups, reinforcement learning is used to learn the best push, or sequence of pushes, to separate objects within these groups. For that purpose we use the Gaussian process approximate Q-learning method proposed in [20]. Given a probabilistic forward model $P(s'|s,a)$ and a reward function $R(s')$, both modelled by Gaussian processes as explained in Section III-A, a Q-function $\mathcal{Q}(s,a)$ is approximated with another Gaussian process model. In practice, however, since our action space is discrete, we use three functions $\mathcal{Q}_a(s)$ to represent $\mathcal{Q}(s,a)$, one for each possible action $a \in \{a_l, a_f, a_r\}$. Rewards are given when the system observes that an object group segment has been split in two, indicating a successful separation.

The method assumes a set of control points $\{(s_i, a_i)\}$, each with a corresponding Q-value q_i that are used to approximate $\mathcal{Q}(s,a)$. As control points we use actual observations collected during experimentation, without clustering or any other method to reduce the size of the data set. The expected reward at a control point is given by

$$r_i = \int R(s')P(s'|s_i,a_i)ds',$$

Fig. 2: *Objects used in our experiments.*

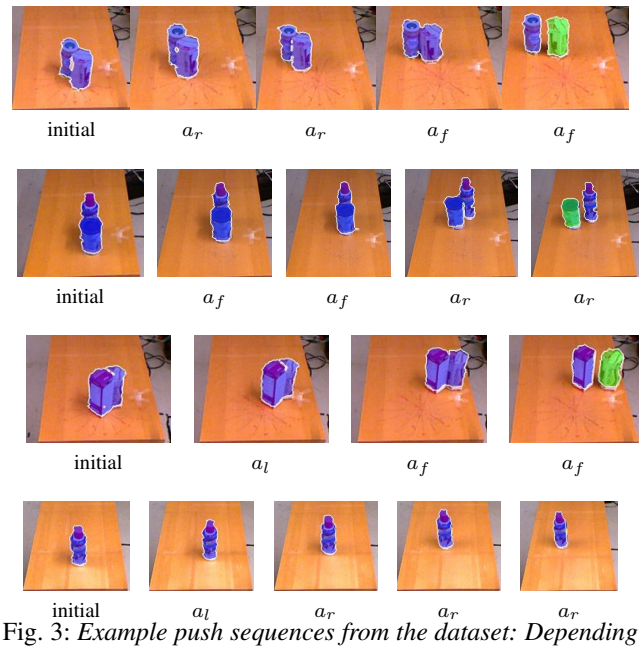


Fig. 3: *Example push sequences from the dataset: Depending on the initial configurations and the randomly chosen actions, sequences have different lengths. Segmentation results after each action from $\mathcal{A}$ is given. Experiments using only one object are also available in the dataset (an example can be seen in the last row).*

while the approximate Q-function is obtained by first updating the q_i values,

$$q_i \leftarrow r_i + \gamma \max_{a' \in \mathcal{A}} \int \mathcal{Q}(s',a')P(s'|s_i,a_i)ds',$$

and then re-estimating $\mathcal{Q}(s,a)$ using Gaussian process regression. These two processes are interleaved and iterated until convergence. Here the discount γ $(0 \leq \gamma \leq 1)$ is used to make a trade-off between sooner and later rewards. Once convergence is reached a policy can be found by maximizing the Q-function over the possible actions,

$$\hat{a}(s) = \arg\max_{a \in \mathcal{A}} \mathcal{Q}(s,a).$$

In the next section we will study how well following such a policy will speed up object separation through pushing.

IV. Experimental Evaluation

As a platform for experimentation, a PR2 humanoid robot was used, with object groups selected from the nine everyday

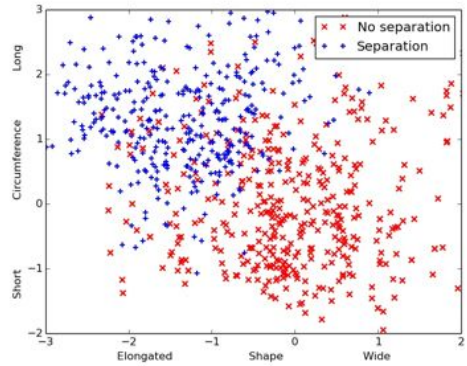


Fig. 4: *Shape and circumference of object group segments for which object separation have (+) and have not (x) been observed. Scales refer to standard deviations.*

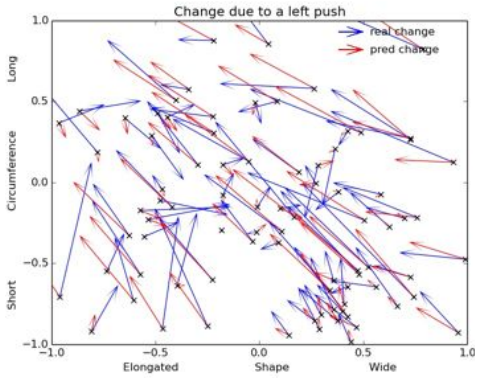


Fig. 5: *Real (blue) and predicted (red) changes in shape and circumference due to a push from the left.*

objects shown in Fig. 2 and the segmentation system described in Section III-B. Training data for the reinforcement based learning algorithm are collected by applying randomly chosen pushing actions until the objects are separated or cannot be reached, see Fig. 3. The total number of pushing sequences is 415 with 1.87 pushes on average per sequence. Using this set of data the models and classifiers were learned and tested, as described in the following subsections.

A. Feasibility of representation

The hypothesis is that the low-dimensional representation of sensory data given by $\mathcal{S}$ is sufficient for planning of pushing actions, to maximize object separation in a reinforcement learning framework. To test the feasibility of $\mathcal{S}$ for prediction of object separation, Gaussian process based classifiers were trained using the set of 771 observed pushes. Examples of shape and circumference of object group segments observed after a push can be seen in Fig. 4. With the full set of sensory measurements, separation rewards can be predicted with a probability of 91.7%[1] from (s, a, s')-tuples. Comparing this

[1]All classification results are given at equal error rate with training done using data divided with 5-fold cross validation.

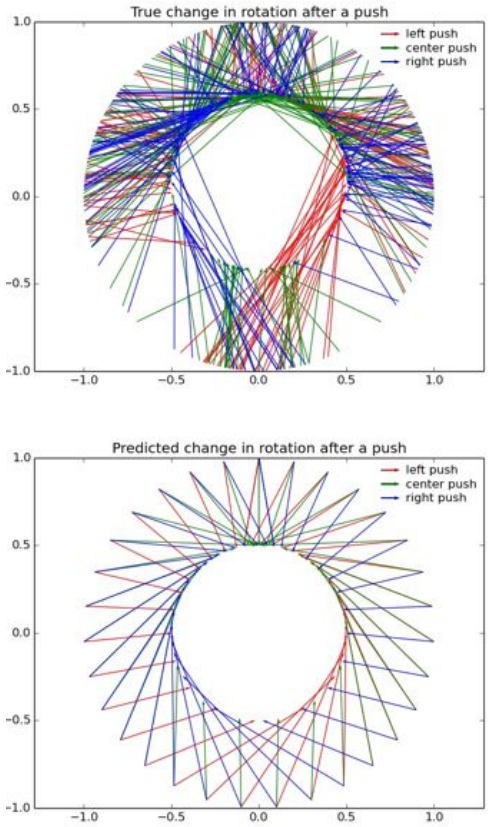


Fig. 6: *Real (upper) and predicted (lower) changes in orientations for left (red), center (green) and right (blue) pushes. The outer circle illustrates orientations prior and the inner center after a push. The lower side of circles represents orientations when objects are in-front and the upper side objects placed next to each other.*

rate to the prediction rate of 89.8%, using a similar classifier based on (s, a), it can be concluded that prediction of object separation is feasible given only information available prior to a push. For a classifier based on only s, the prediction rate is significantly lower, 72.1%, showing a dependence between the pushing action applied and the current state. By excluding one sensory dimension at the time, the prediction rate decreases with respectively 1.9%, 1.3%, 2.5% and 26.9% for s_{shape}, s_{fragm}, s_{circum} and s_{orient}. Thus each dimension is of importance, but the orientation is significantly more important than the other dimensions.

B. Analysis of forward model

As mentioned earlier the expected change in sensory data is captured by forward models $P(s'|s, a)$ that are modelled through Gaussian processes. Given the high dimensionality, studying the behavior of the forward models is complicated and is better done in projections. Fig. 5 shows the real and predicted changes in shape and circumference for a selected number of samples. Since samples have different orientations

it is hard to see a pattern in the predictions. Despite this one can conclude that predictions are similar to the real changes.

More distinct patterns can be seen when studying changes in orientation, as shown in Fig. 6. Even if it can be observed also in the real data, the regularity is particularly clear in the predictions. If objects are placed in-front of each other (see the lower side of circles) there is a distinct difference in how a group rotates, with less change for center pushes, than if pushes are from the side. Another observation is that objects tend to be pushed so that they eventually end up next to each other. If objects are originally placed in-front of each other, it might take a number of pushes though.

Analyzing the performance of the forward models is easier, if models are used for a purpose. One such is classification, such as distinguishing between groups of one or two objects. Using a set of 91 recorded pushes applied to single object groups, a forward model was created, similar to the one based on groups of two objects. Using these two models a naïve Bayesian classifier was created. With cross-validation it was concluded that the number of objects in a group can be classified with a probability of 94.0% using (s, a, s')-tuples or 85.9% with just s. In practice this means that a system might try to separate objects assuming that a group contains two objects, but stop shortly after a single push, if the observed effect indicates that the group is just one object.

C. Policy based pushes

From $\mathcal{Q}(s, a)$, learned with Gaussian process approximate Q-learning as described above, a pushing policy can be found by maximizing $\mathcal{Q}(s, a)$ with respect to the action a, i.e. the action that will most likely lead to a future separation of objects. The question is how such a policy differs from a random policy, where sensory information is not taken into account. Another question is how the policy changes, if more emphasis is placed on future separation events, compared to a greedy policy, for which only the next push is considered.

The upper graphs in Fig. 7 show the probability of receiving an immediate reward for different orientations (s_{orient}) and pushing directions. From the graphs it is clear that the best push is always a push from the side, with the dominating side of the object facing the pushing direction. The exceptions are in the two extreme orientations, where the direction does not matter. Regardless of push, however, you are unlikely to immediately separate the objects, if they are placed in-front of each other. The picture changes slightly with Q-learning and a discount factor $\gamma = 0.5$, as can be seen in the lower graphs of Fig. 7. If objects are side-by-side, a center push is preferable from a sideways push. For objects placed in-front of each other, a sidewise push will more likely lead to a situation where objects can be separated in the push that follows, even if the first push fails to do so.

Using a series of 100 experiments performed online with different configurations of two object groups, the benefit of the computed policy was also tested by comparing the results to those of random pushes. The same original configurations

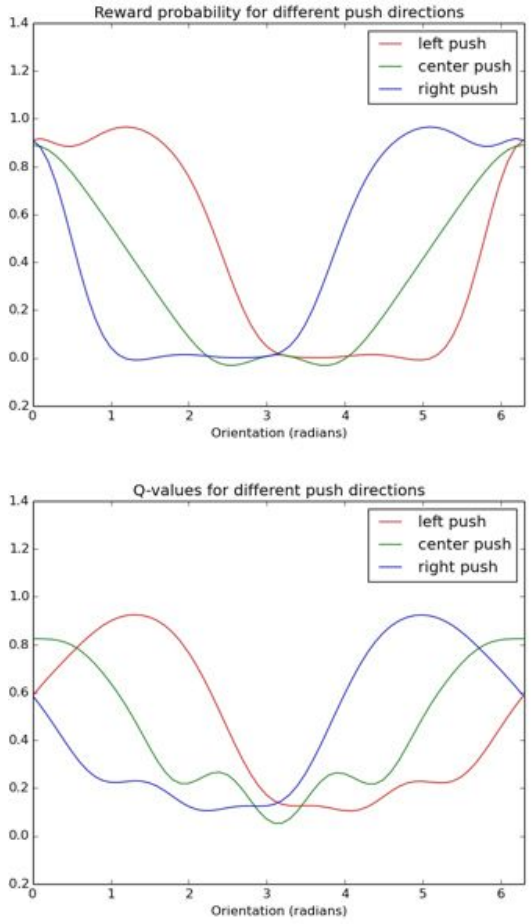


Fig. 7: *Predicted immediate rewards (upper) and Q-values (lower) for different orientations (s_{orient}) and pushes from the left (red), center (green) and right (blue). Orientations of 0 and π radians respectively correspond to two objects placed next to and in-front of each other. For all sensory dimensions, but s_{orient}, the mean values are assumed.*

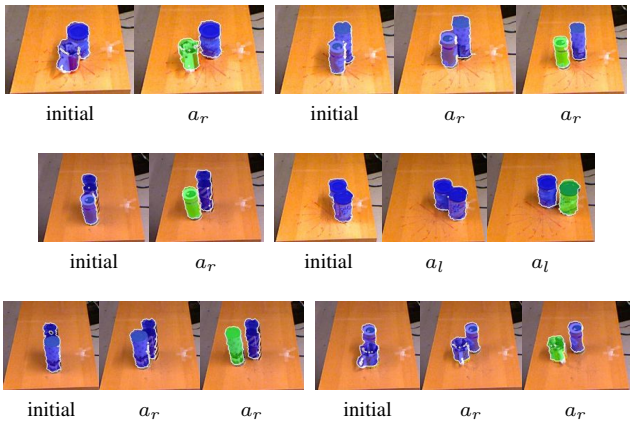


Fig. 8: *Example push sequences using the policy.*

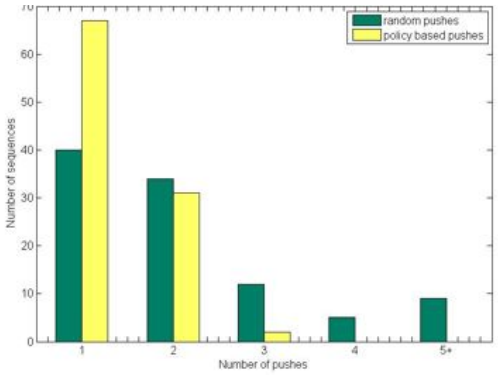


Fig. 9: *Number of pushes required for object separation to be detected using either random or policy based pushes.*

were used for both policy based and random pushes. Example policy based pushes can be seen in Fig. 8. Histograms of the number of pushes required for object separation rewards to be received can be seen in Fig. 9. With policy based pushes you rarely need more than two pushes to separate two objects, unlike random pushes that often requires more.

It is also possible to separate objects with only one push in 67% of the cases with policy based pushes, instead of 40% for random ones. The average number of pushes varies depending on the orientation of objects. For objects side-by-side you need 1.3 policy based pushes or 1.7 random ones, while the averages increase to 2.0 and 3.1 respectively for objects place in-front of each other.

V. Conclusions

This paper addressed the problem of learning through exploration to verify object hypotheses initialized from segmentation. Our robot explored scenes with segmented objects through randomly generated pushing actions, resulting in more than 700 observed pushes in total. Nine objects were used in multiple configurations of two object combinations to acquire sensory data along with corresponding action parameters.

Bottom-up sensory features for learning were defined, features that capture how the observed 3D point clouds change due to pushing actions, something that was modelled using Gaussian processes. We presented a framework based on Q-learning to learn what pushes to apply to receive rewards due to successful separation of grouped objects. Experimental results demonstrated that the proposed learning method is capable of separating objects with fewer pushes, than if pushes where randomly generated, assuming the same initial conditions and objects.

In the presented framework, sensory and action spaces were intentionally kept small to facilitate learning with limited sets of training examples. Despite this the system managed to autonomously learn how to push object groups to separate objects. It is likely to believe, however, that by always pushing at the centroid of the observed point cloud, there are inherent limitations to what combinations of objects can be separated. We thus like to extend the action space, first to become continuous and then to allow pushes at different locations. Furthermore, in the presented study we used a four dimensional sensory space, dimensions that were predefined.

In the future work, we intend to apply unsupervised learning on the raw 3D measurements to autonomously find low-dimensional representations directly given by the data, this in order to avoid any bias that might exist by design.

References

[1] J. Piaget, "Piaget's theory," in *Handbook of Child Psychology*, P. Mussen, Ed., vol. 1. New York: Wiley, 1983.

[2] A. Stoytchev, "Some basic principles of developmental robotics," *IEEE Transactions on Autonomous Mental Development*, vol. 1, no. 2, pp. 122–130, 2009.

[3] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, no. 9, pp. 1124–1137, Sept 2004.

[4] H. Zhang, J. E. Fritts, and S. A. Goldman, "Image segmentation evaluation: A survey of unsupervised methods," *Computer Vision and Image Understanding*, vol. 110, no. 2, pp. 260–280, 2008.

[5] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut -interactive foreground extraction using iterated graph cuts," *ACM Transactions on Graphics (SIGGRAPH)*, August 2004.

[6] N. Bergström, M. Björkman, and D. Kragic, "Generating object hypotheses in natural scenes through human-robot interaction," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, sept. 2011, pp. 827–833.

[7] J. Kenney, T. Buckley, and O. Brock, "Interactive segmentation for manipulation in unstructured environments," in *IEEE Int. Conf. Robotics and Automation (ICRA)*, 2009, pp. 1377–1382.

[8] P. Fitzpatrick, "First contact: an active vision approach to segmentation," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2003, pp. 2161–2166.

[9] K. Hausman, F. Balint-Benczedi, D. Pangercic, Z.-C. Marton, R. Ueda, K. Okada, and M. Beetz, "Tracking-based interactive segmentation of textureless objects," in *IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2013, pp. 1122–1129.

[10] H. van Hoof, O. Kroemer, and J. Peters, "Probabilistic segmentation and targeted exploration of objects in cluttered environments," *IEEE Transactions on Robotics (T-RO)*, accepted.

[11] L. Y. Chang, J. R. Smith, and D. Fox, "Interactive singulation of objects from a pile," in *IEEE Int. Conf. Robotics and Automation (ICRA)*, 2012, pp. 3875–3882.

[12] T. Hermans, J. Rehg, and A. Bobick, "Guided pushing for object singulation," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2012, pp. 4783–4790.

[13] B. Ridge and A. Ude, "Action-grounded push affordance bootstrapping of unknown objects," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2013, pp. 2791–2798.

[14] V. Högman, M. Björkman, and D. Kragic, "Interactive object classification using sensorimotor contingencies," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Nov 2013, pp. 2799–2805.

[15] M. Sanchez-Fibla, A. Duff, and P. F. Verschure, "The acquisition of intentionally indexed and object centered affordance gradients: a biomimetic controller and mobile robotics benchmark," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2011, pp. 1115–1121.

[16] M. S. Kopicki, S. Zurek, R. Stolkin, T. Morwald, and J. L. Wyatt, "Learning to predict how rigid objects behave under simple manipulation." in *IEEE Int. Conf. Robotics and Automation (ICRA)*, 2011, pp. 5722–5729.

[17] K. O'Regan and A. Noë, "What it is like to see: A sensorimotor theory of perceptual experience," *Synthese*, vol. 129, no. 1, pp. 79–103, 2001.

[18] M. Kawato, "Internal models for motor control and trajectory planning," *Current Opinion in Neurobiology*, vol. 9, no. 6, pp. 718–727, 1999.

[19] C. E. Rasmussen, "Gaussian processes in machine learning," in *Advanced Lectures on Machine Learning*. Springer, 2004, pp. 63–71.

[20] M. Kuss, "Gaussian process models for robust regression, classification, and reinforcement learning," Ph.D. dissertation, TU Darmstadt, 2006.

2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
September 14-18, 2014. Chicago, IL, USA,

What's in the Container? Classifying Object Contents from Vision and Touch

Püren Güler, Yasemin Bekiroglu, Xavi Gratal, Karl Pauwels and Danica Kragic

***Abstract*— Robots operating in household environments need to interact with food containers of different types. Whether a container is filled with milk, juice, yogurt or coffee may affect the way robots grasp and manipulate the container. In this paper, we concentrate on the problem of identifying what kind of content is in a container based on tactile and/or visual feedback in combination with grasping. In particular, we investigate the benefits of using unimodal (visual or tactile) or bimodal (visual-tactile) sensory data for this purpose. We direct our study toward cardboard containers with liquid or solid content or being empty. The motivation for using grasping rather than shaking is that we want to investigate the content *prior to* applying manipulation actions to a container. Our results show that we achieve comparable classification rates with unimodal data and that the visual and tactile data are complimentary.**

I. Introduction

Visual and haptic perception of robots has evolved significantly during the last couple of years. Using multiple sensory modalities comes natural to us humans and we are able to shift between them in cases when one of them is not functioning. The study by Norman et al. [1] investigates the role of sensory modalities in understanding object shape and shows that vision and touch provide complimentary information. In this study, human subjects are presented two objects and asked to match them by unimodal (visual-visual, haptic-haptic) and cross-modal (visual-haptic, haptic-visual) shape comparison where cross-modal shape comparisons led to best results. Heller [2] studies the use of vision and haptic in humans and shows that visual observation of hand movements improve surface texture perception. In addition, it is shown that touch is used for gathering information about the texture during interaction with an object while vision is used to monitor the hand, see Figure 1.

In robotics, visual and haptic/tactile data have been utilized for perceiving object properties during interaction/grasping. Haptic and force-torque sensors have been studied extensively for control of grasping and manipulation [3] to avoid slippage and prevent damaging objects. Vision [4], [5] has typically been used to plan grasping actions and to update action parameters when objects move to compensate for manipulator positioning inaccuracies and sensor noise. Integration of these modalities results in a richer observation space: through haptic sensing, we extract the information at the contact regions between the fingers and the object while the change in object deformation around the finger may be extracted through visual sensing. This way, the resolution of haptic sensors which is still inferior compared to human skin can be supported and used to develop better control algorithms for grasping. For example, grasping a cardboard container will require different forces depending on whether it is filled with milk or cereal.

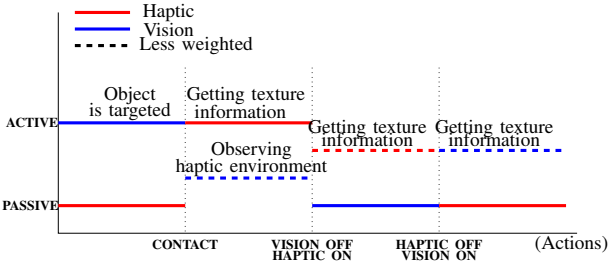


Fig. 1. According to Heller [2], there is a "division of labor" between senses. While making bimodal judgments, using vision and touch together, we rely on different contributions from both: vision is for controlling the hand and exploring the haptic environment (participants monitored their hand not the surfaces being touched) and haptic is for gathering information about surface texture. In the absence of one, either vision or touch leads to similar levels of accuracy but higher accuracy is achieved when these are combined.

In this paper, we study the feasibility of using unimodal (visual or tactile) and bimodal (visual and tactile) data to identify object content by applying grasping or squeezing actions on it. The aim is to enable proper manipulation of objects both in terms of applied forces and subsequent actions such as pouring or lifting. The main focus is thus to investigate to what extent vision and haptic modalities individually or integrated are useful for this purpose by comparing different learning methods: k-means, Quadratic Discriminant Analysis (QDA), k-nearest neighbors(kNN) and support vector machine (SVM).

This paper is organized as follows. In Section II, we review the related work and summarize contributions of our work. We present the classification methods and the methodology in Section III. In Section IV, we present the experimental evaluation and summarize the paper in Section V.

II. Related Work and Contributions

In robotics, modeling and recognition of object properties, e.g., material [6], shape [7] and pose [8], have been studied primarily by using touch sensing. Chitta et al. [9] estimate the internal state of objects, i.e., if a bottle is closed, open, full or empty, by using tactile information during grasping. Chu

The authors are with the Computer Vision and Active Perception Lab, Center for Autonomous Systems, School of Computer Science and Communication, KTH Royal Institute of Technology, Stockholm, Sweden, {puren,yaseminb,javiergm,dani}@kth.se and Computer Architecture and Technology Department, University of Granada, Spain, {kpauwels}@ugr.es. . This work was supported by the Swedish Foundation for Strategic Research and the EU project RoboHow.Cog (FP7-ICT-288533).

et al. [10] use tactile information obtained from exploratory procedures on objects, e.g., tapping, squeezing, holding, and both slow and fast sliding, to learn haptic adjectives such as rough, hard and compact. There are also studies which use different actions rather than grasping such as rolling and scratching to identify the properties of objects, e.g., stiffness or texture [3]. Object shape estimation has also been studied with unimodal data, i.e., only visual [11] or tactile [12] sensing.

Studies regarding integration of sensory modalities concentrate mostly on extracting 3D object shape. Dragiev et al. [13] include laser data in addition to tactile measurements. Björkman et al. [14] utilize both visual and tactile sensing to model unknown objects by touching them strategically at parts that are uncertain in terms of shape without exhaustive exploration. There are also studies that focus on object recognition without explicitly modeling the full 3D shape, but rather representing the objects based on visual [15], tactile [16], [7] or both features [17].

In all these studies, objects have different appearance and texture can therefore be used as an informative visual property. Our aim in this paper is to exploit to what extent we can go beyond the classical object recognition, categorization and matching approaches and concentrate solely on local information, as shown in Figure 3. By local we consider visual information resulting from the fact that a container will deform differently around the grasping point dependent on whether it is empty or it is filled by different types of material. Differently from the aforementioned related work, we focus on exploring to what extent tactile and visual sensing can provide information about this - either individually or integrated.

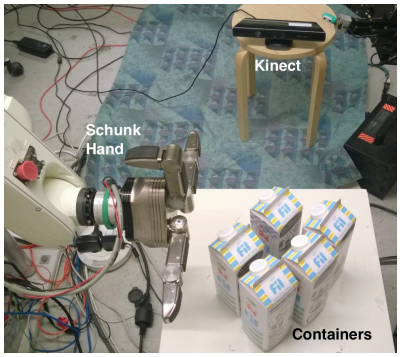


Fig. 2. The containers used in the experiments and the experimental robot platform, composed of a Kuka industrial arm, a three-finger Schunk Dextrous hand equipped with tactile sensing arrays, and a Kinect camera. The Kinect camera is placed approximately one meter away from the robot.

In our approach, we let a robot (Figure 2) explore containers filled with different types of content by squeezing them and observing the effects of the action on the container using visual (deformation) and tactile cues (pressure). We use squeezing/grasping action for detection since previous studies [10] show that this action can be equally informative as other actions for object classification, e.g, shaking. The robot assesses the degree of deformation in the contact region and in the area around it. It then infers the type of content inside the container based on the experience gathered through an offline learning process. For this purpose, we investigate and compare supervised and unsupervised learning methods.

III. Methodology

The system employs an offline learning step and an online inference step. The offline step is used to generate the experience based on training data. To generate the training data, we first let the robot squeeze/grasp containers with different contents. Grasps are applied around the middle part of the object, see Figure 3. We employ a model based tracking [18] to acquire the visual data which is composed of the depth values of the container. We capture the change in depth between the first and the final grasping frame and use it as our visual input. Tactile data from the fingertip sensors in the final grasping position are also stored. Examples of training data for different sensory modalities are shown in Figure 6 and Figure 7. In our analysis, there are two questions we want to answer:

1) "Is it possible to identify the content of a container given the available sensory modalities?"
2) "Should the problem be modeled using parametric or non-parametric learning techniques?"

Thus, the first step of our analysis investigates the viability of discriminating between the different contents. For the analysis, we apply an unsupervised learning or clustering method. Secondly, we learn the relations between the tactile/visual data and the content types using two kinds of supervised learning methods: parametric and non-parametric. The choice of a learning method is dependent on the characteristics of the data: in particular, the data is high-dimensional but the available training data is limited. Thus, the assumptions inherit to the learning methods, e.g. that the data is normally distributed, may not hold.

A. Training Data

Our training dataset includes tactile and visual features and the class labels for each grasping experiment denoted by $D_o = \left\{(x_o^i, l^i)\right\}_{i=1,\ldots,n}$. Here, each pair (x_o^i, l^i) is composed of perceptual readings $x_o^i \in \mathbb{R}^d, o \in \{t, v, tv\}$ and discrete content labels l^i. A vector x representing perceptual observations includes change in depth values x_v^i, tactile readings x_t^i and tactile and visual data combined x_{tv}^i. More information about the size of the dataset is provided in Section IV.

B. Learning Methods

The first method we apply is k-means clustering [19]. The goal is to identify the underlying similarities or clusters in the dataset without making any assumptions about the class to which the data samples belong to. Let us denote $C = \{C_1, C_2, ..., C_k\}$ to be the partitions (clusters) of the datasets D_t, D_v, D_{tv} and centroid $c_i, i = 1, 2, ..., k$ as the conditional mean of p, which is the probability mass function for the dataset, over the set C_i. Then, $w^2(C) = \sum_{i=1}^{k} \int_{C_i} |x - c_i|^2 dp(x)$ is expected to be low for the partitions in C which are generated as a result of the method. In each step of the

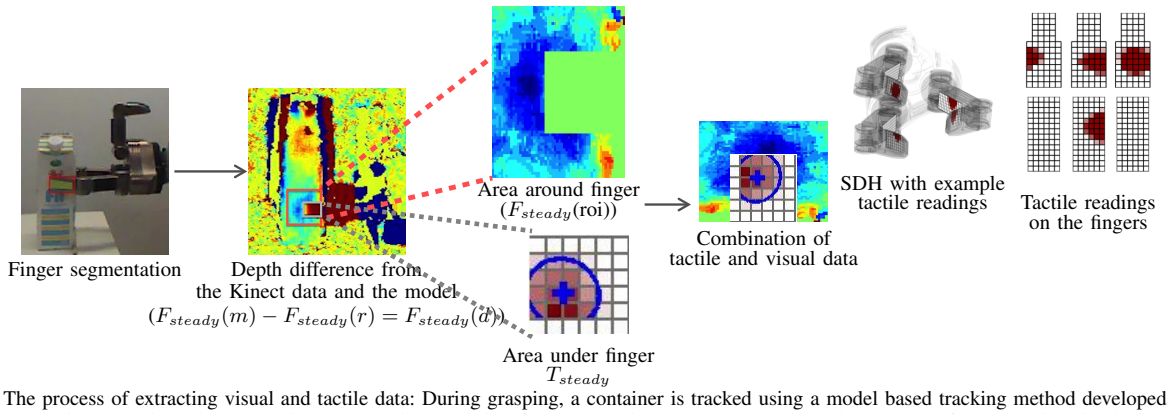


Fig. 3. The process of extracting visual and tactile data: During grasping, a container is tracked using a model based tracking method developed in [18]. We track the object pose in order to localize the container and therefore capture its depth values. Then, the contour of the container is obtained. When the grasp reaches the steady state, the finger is segmented from the container using color filtering. The segmented finger is marked with green color in the left figure. The difference between the depth values corresponding to the model of the container and the Kinect data at the steady state of the grasp is calculated. Finally, a 61x61 pixel area around the finger is extracted. At the steady state, tactile readings are also saved. Example readings that correspond to the available pressure at the contact regions between the hand and the object are illustrated with the red areas on the tactile readings in the image. Blue lines show the orientation of the contact pressure and the cross shows the center of the pressure. Best viewed in color.

algorithm a new c is calculated and this calculation continues until c does not vary.

The second method we apply is kNN [20] which is a widely used non-parametric technique for multi-class classification that identifies the closest training examples given a query. The nearest neighbors of an observation is defined using Euclidean distance. We assume that we have a finite set of classes $C \in \{c_1, c_2, ..., c_s\}$ in datasets D_t, D_v, D_{tv}. For the classification of a new instance $x_o^q \in R^d$, the distance between two instances is calculated as $d(x_o^q, x_o^j) = \sqrt{\sum_{r=1}^{d} \left(x_o^q(r) - x_o^j(r)\right)^2}$, where $r = 1, 2, ...d$. The target function to learn the discrete class labels through real-valued observations can be formalized as $f : R^d \rightarrow C$. Then x_o^q is assigned to a class c_i using

$$\widehat{f}(x_o^q) = \arg\max_{c_s \in C} \sum_{j=1}^{k} \delta(c_s, f(x_o^j)),$$

where $\delta(m, n) = 1$, if $m = n$ and it is zero if $m \neq n$ and $\widehat{f}(x_o^q)$ returns a value which is the most common value of f among the k training examples nearest to x_o^q.

The third method is QDA [21] which is a parametric method for multi-class classification. It assumes that the data has a Gaussian mixture distribution with means and covariances which vary for each class. In the training stage, the datasets D_t, D_v, D_{tv} are fitted into a matrix M with size of $N-by-K$, where N is the number of data points and K is the number of the classes. M_{nk} is set to 1 if the data pointx_n, $n \in N$ belongs to the class $C_k, k = 1, ..., K$, otherwise it is $M_{nk} = 0$. In the inference stage, the following expectation classification cost is minimized:

$$\hat{y} = \arg\min_{y=1,...,K} \sum_{k=1}^{K} \hat{P}(C_k|x) S(C_y|C_k) \quad (1)$$

where $\hat{P}(C_k|x)$ is the posterior probability of the class C_k for the data point x and $S(C_y|C_k)$ is the cost of classification of a data point as C_y while the actual class is C_k. The posterior probability is composed according to Bayes rule and the likelihood is expressed as the density of the multivariate normal with the mean μ_k and the covariance Σ_k, while the prior probability of the class C_k is a uniform distribution over the total number of the classes. The parameters $\hat{\mu}$ and $\hat{\Sigma}$ are calculated as follows:

$$\hat{\mu_k} = \frac{\sum_{n=1}^{N} M_{nk} x_n}{\sum_{n=1}^{N} M_{nk}}, \quad (2)$$

$$\hat{\Sigma} = \frac{\sum_{n=1}^{N} \sum_{k=1}^{K} M_{nk} (x_n - \hat{\mu_k})(x_n - \hat{\mu_k})^T}{N - K}. \quad (3)$$

The final method we apply is SVM [22]. SVM is designed for two-class classification problems. It fits a hyperplane between two classes. The aim is to find the best separating hyperplane so that the maximum margin between the classes is obtained. An optimization method is applied to calculate the weights w_i and the bias b to classify $x_j, j = 1, ..., N$ using the following formula:

$$y_j = \sum_i w_i K(x_i, x_j) + b, \quad (4)$$

where x_i is the support vector and if $y_j \geqslant 0$, then x_j is classified as 1 which can be called as the first class, otherwise -1 (second class). We used a quadratic kernel:

$$K(x_i, x_j) = ({x_i}^T x_j + b)^2. \quad (5)$$

In our use, we have a multi-class classication problem. We assume that there are $N_k, k = 1, 2, .., 5$ points in the class k in the datasets D_t, D_v, D_{tv}. We have the training and the test data points $x_j, j = 1, 2, ..., N$. In the training phase, for each k, the parameters w_k and b_k are predicted using optimization by setting the label of the data point, y_j of the class k as 1 and the rest as -1. Thus, the problem is converted into a two-class classification problem. In the test phase, the learned parameters are used when we want to classify a new

point x_j from the test data. For the classification, a score is calculated for each class

$$s_k(x_j) = \sum_{i=1}^{n} w_i^k y_i < x_i, x_j > + b_k \quad (6)$$

where $(w_k^1, ..., w_k^n, b_k)$ are the SVM parameters predicted for each class. The dot product is between x_j and the support vectors x_i. y_i is the training set labels. Later, the posterior probability is approximated as a function of the score [23].

$$P(y_j = 1|x_j) = \frac{1}{1 + exp(As_k(x_j) + B)} \quad (7)$$

where A and B are the slope and intercept parameters of the sigmoid function. x_j is assigned to the class k with the highest probability.

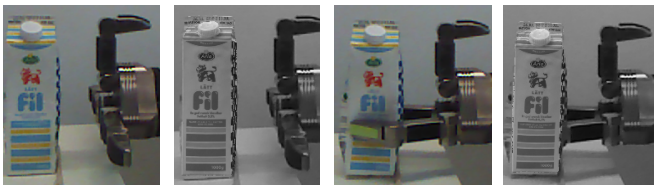
Before applying the learning algorithms, as a standard first step to data preprocessing, each feature vector in the dataset is normalized to zero-mean and unit variance. Since the dataset is high dimensional, we employ Principal Component Analysis (PCA) [24]. The principal components which contain 80% of the variance of the dataset are used in the training phase and the test phase of the analysis. The principal components are calculated separately for D_v, D_t, D_{tv}. We also reduce the resolution of the visual data 10 times to cope with the limited amount of the data.

C. Test for Normal Distribution

To test whether the data is normally distributed or not, we applied Kolmogrov-Smirnov (KS) test [25]. In this test, it is assumed that a population $z \in R^N$ has a cumulative distribution function (cdf) $F_0(z)$. This cdf implies that for every $z_i, i = 1, ..., N$ in any population with the sample size N, $F_0(z_i)$ gives the proportion of the ones whose measurements (corresponding to the tactile and visual features in our case) which are less than or equal to z_i in the population. $F_0(z_i)$ is tested against a cumulative step function (csf) $S(z_i) = \frac{k}{N}$ where k is the number of the observations less than or equal to z_i. The maximum distance between $F_0(z_i)$ and $S(z_i)$ is then

$$d = max|F_0(x) - S(x)|.$$

According to the comparison of d and an upper bound α (significance level), a decision is made: the null hypothesis of the data being normally distributed is rejected, if $d > \alpha$, where α is set to 0.05 in our experiments. We applied the test separately for D_v, D_t, D_{tv} on each feature vector, after dimensionality reduction, by dividing the dataset into partitions with 10, 20, 30, 40 and 50 samples. In D_v, for all of the feature vectors the null hypothesis is rejected while for D_t and D_{tv}, for only one feature vector, the hypothesis is accepted the partitions with 10 and 30 samples respectively. These results thus indicate that our data may not lead to a better classification performance with the parametric method in comparison to the non-parametric one. We will investigate this further in the next section.



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

Fig. 4. The original image captured by the Kinect: (a) before grasping and (c) after grasping. The synthetic model of the container overlaid on the container: (b) before grasping and (d) after grasping. The synthetic model is used by the model based tracker to track the pose of the object. We can observe that there is a clear difference between the original image which shows the deformed container and the estimate provided by the model based tracker. The estimated depth difference is based on this observed depth difference obtained through subtraction.

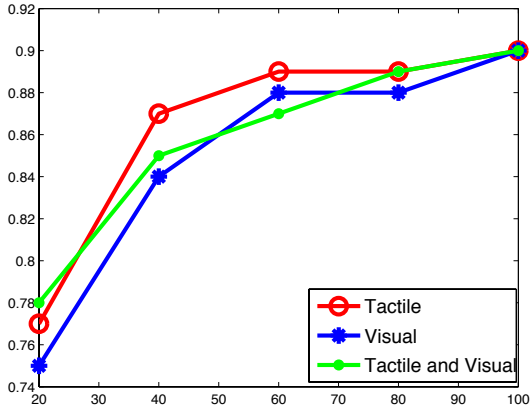


Fig. 5. We apply the knn classification on the data with different sizes with 20%, 40%, 60%, 80%, 100% of all data. The accuracy rate is shown against the data size for tactile, visual and, tactile and visual data.

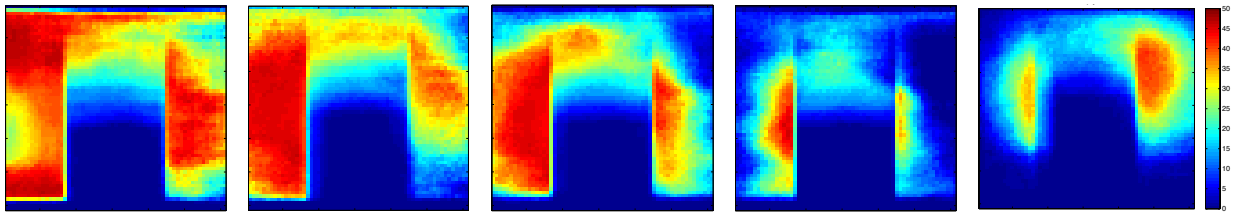
IV. Experimental Evaluation

We first describe the data collection procedure together with the data selection choices. We then present and discuss the results of the employed learning techniques on the data.

A. Data Acquisition

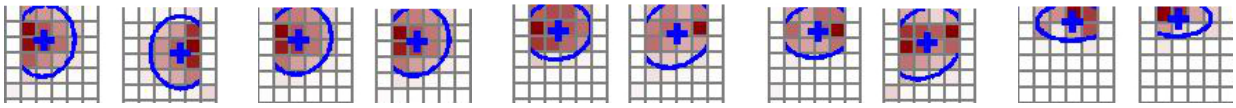
Our method relies on the visual data acquired by a Kinect camera. The tactile data is acquired with a seven degree-of-freedom Schunk Dextrous Hand (SDH) mounted on an industrial KUKA arm. The SDH has two pressure sensitive tactile sensor arrays on each of its three fingers, see Figure 2 and 3. We first generate random grasping positions on the object around its middle region. Middle region on the container along its longest edge is chosen for grasping as it is considered as the most informative region for containers. Each container is filled with the same amount of content which is about 90% of the container.

In our experiments we use five content classes denoted $l_i \in \{$*empty*, *water*, *yogurt*, *flour*, *rice*$\}$. We conduct the experiments using several identical containers with different contents to capture the actual behavior during squeezing and avoid introducing differences in the data due to variations in the actual container material. The number of grasps executed is 150 for each content type. 30 grasps are removed from *yogurt* data due to corruption. The same amount of force



(a) Rice (b) Flour (c) Yogurt (d) Water (e) Empty

Fig. 6. The depth values of the area around the finger are obtained for the containers filled with different content. The average of these images with 61x61 pixels is calculated for each content by removing the finger in the middle. Different levels of the change in depth values is observed starting from the hardest content (*rice*) to the softest one (*empty*). The middle blue area shows the location of the finger. Best viewed in color.



(a) Rice (b) Flour (c) Yogurt (d) Water (e) Empty

Fig. 7. The tactile data resulting from the contact between the finger and the object for the containers filled with different content - same force is applied to all containers. The average of these 6x6 tactile images is calculated for each content. The two tactile arrays which are active during grasping are used and shown side by side in the image. Different tactile signatures are obtained for different materials. Best viewed in color.

is applied for every grasp. To avoid saturation of the tactile data, we run the hand with the lowest current value. Figures 6 and 7 show, for each content, the average image of the area around the finger and the tactile readings under the finger respectively over all grasps with the same scale. As seen from these average images, the tactile and visual data vary dependent on the content of the container. Classification using the same type of objects with different contents is a different problem than categorizing objects according to their surface texture [10]. It is likely to think that chosen categories may not be "felt" distinctive enough by the tactile and the visual sensors. Saturation in visual depth values could happen due to applying too large or too small force. We first conduct experiments with three classes: *liquid*, *granular* and *empty*. Then we extend the set by two additional classes: *liquid* (*water*,*yogurt*), *granular* (*flour*,*rice*) and *empty*.

As stated in [2], the deformation around the contact region provides valuable information for object perception. We therefore select the area around the finger as our visual input. The camera observes the grasped side of the container and we record frames starting when the first contact (F_{first}) with the surface occurs until the grasp reaches a steady state (F_{steady}). By steady state we consider the pose of the fingers where the joints have stopped moving while applying a constant current to the motors. In order to capture the change around that area, we compare the depth values of the container ($F_{steady}(r)$) at F_{steady} and the depth values of the model of the container ($F_{steady}(m)$) obtained from the rigid-object-tracker [1] [18], see Figure 4.

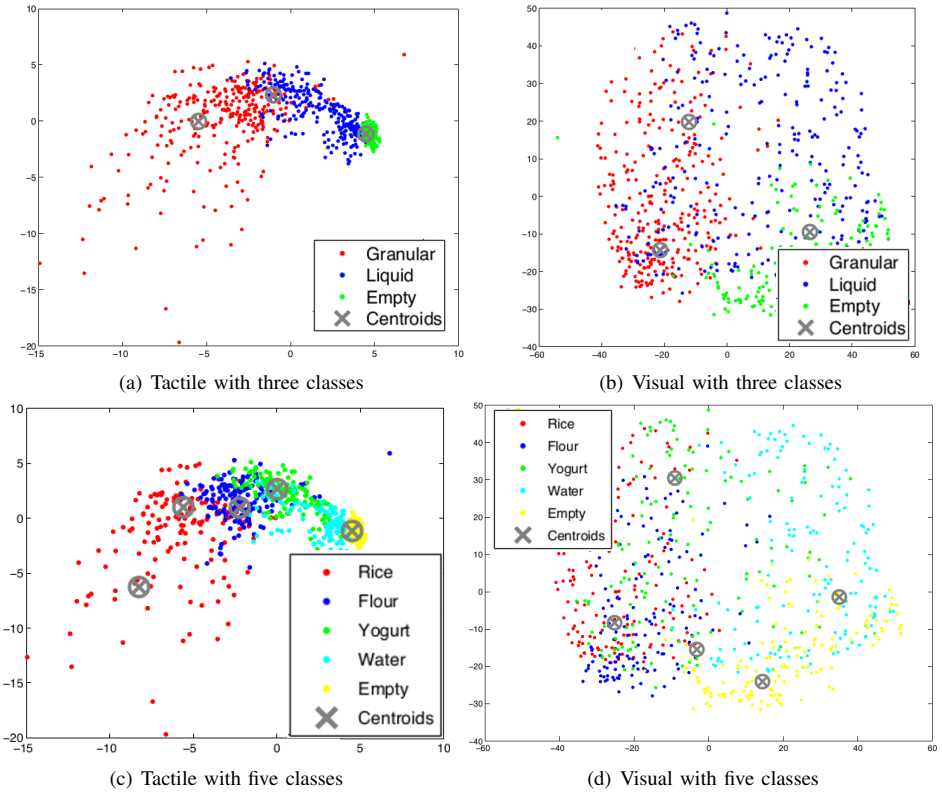
$F_{steady}(m)$ represents the object before grasping. It can be seen as an ideal image of the container. After grasping, we differentiate from $F_{steady}(m)$ and with the comparison of the ($F_{steady}(m)$ and $F_{steady}(r)$), we aim to measure how much the deformed object deviates from its original shape. This comparison provides a depth-difference image ($F_{steady}(d)$) on the area with a size of 61x61 pixels (F_{steady}(roi)) around the finger instead of the whole container surface. This area captures most of the changes occurring on the object surface during grasping. Eventually, the resulting depth image, F_{steady}(roi) includes our raw features denoted as $f \in \mathbb{R}^{61\text{x}61}$ that is used for the classification of $D_v = \{(x_v^i = F_{steady}(\text{roi}), l^i)\}$ for the ith grasp. Before training and testing PCA is applied on D_v. In addition, we also investigate the limits of using raw data. However, due to the high dimensionality of the visual data with respect to the dimensionality of the tactile data and the amount of the data points, we do use the raw tactile and the visual data together. The number of pixels is reduced from 61x61 to 11x11. Thus, this time, F_{steady_small}(roi) includes our raw features denoted as $f \in \mathbb{R}^{11\text{x}11}$ that is used for the classification of $D_{v\_small} = \{(x_v^i = F_{steady\_small}(\text{roi}), l^i)\}$ for the ith grasp. We do not apply PCA on this data.

Tactile data is obtained from the pressure-sensitive arrays on the fingers. A specific two-fingered preshape is used for grasping, see Figure 3. Each finger is composed of two segments which yields a total of four tactile arrays. Since the hand achieves contact with the fingertips, we use only the fingertip segments of the tactile arrays. The first 36 cells of each array are used since these are in contact and therefore most informative regions. The resulting tactile data T_{steady} denoted by $f \in \mathbb{R}^{2\text{x}36}$ is used for the classification of $D_t = \{(x_t^i = T_{steady}, l^i)\}$ for the ith grasp. For the experiments combining visual and tactile data we use $D_{tv} = \{(x_{tv}^i = T_{steady}, F_{steady}(\text{roi}), l^i)\}$ and $D_{tv\_small} = \{(x_{tv}^i = T_{steady}, F_{steady\_small}(\text{roi}), l^i)\}$. The process of data collection is illustrated in Figure 3.

B. Natural Groupings through Clustering

We first analyze the data with k-means in order to show the natural groupings of *empty*, *liquid* and *granular*. We set thus the number of clusters to three. The clustering is repeated 10 times and average results are reported. The data shown

[1]The tracking module exceeds 60 Hz and the frame size is 640x480.



(a) Tactile with three classes (b) Visual with three classes

(c) Tactile with five classes (d) Visual with five classes

Fig. 8. Projection of the data points onto the two principal components with the largest Eigen values for (a) three clusters and tactile data (b) visual data, (c) five clusters and tactile data and (d) visual data. Best viewed in color.

in Figure 8(a) is the tactile data projected on the first two principal components. In the figure, the data is colored based on their original labels and the centroids are calculated using k-means. We can see from the Figure 8(a) that there are natural clusters for *empty*, *liquid* and *granular*. However, the centroid of *liquid* cluster is overlapping with a small portion of the *granular* data points, which reveals their expected similarities. Hence, there can be some misclassification between *granular* and the *liquid* classes.

Another clustering result is shown for visual data in Figure 8(b). In the projected data, we can observe that there is more overlap than for the tactile data. We can also observe the similar trend to the tactile data in terms of natural clustering for three content types. As for clustering with $k = 5$, we observe overlap between similar materials both for visual and tactile data, see Figure 8(c) and 8(d).

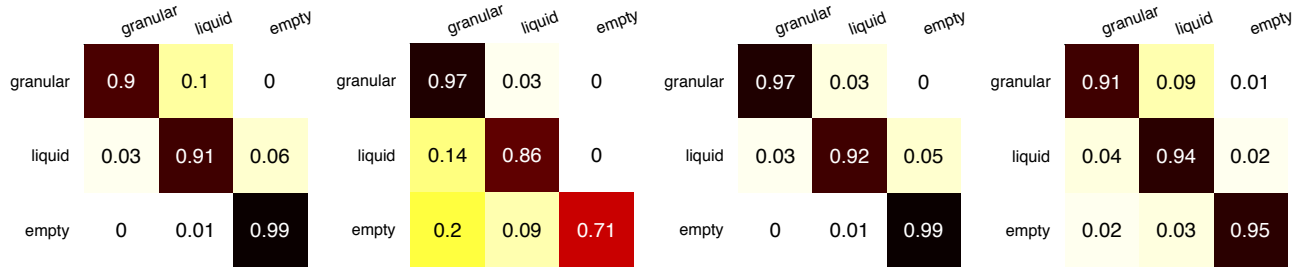
C. Learning the Association between the Data and Content Classes

We set up classification experiments to evaluate if the associations between the visual/tactile data and the corresponding labels could be learned from the training data using kNN, QDA and SVM. We test whether the same classification performance can be acquired using smaller amount of data. As seen in Figure 5, the increase of the data size has a positive effect on the classification. We thus report the results using all the available training data.

We choose on random 70% of the data for training and use the rest for testing. We repeat the classification experiments with randomly chosen train and test splits 200 times and report the average accuracy rates. We set the number of nearest neighbors for kNN to three and five for three classes and five classes respectively and use Euclidean distance for measuring the similarities between candidate neighbors.

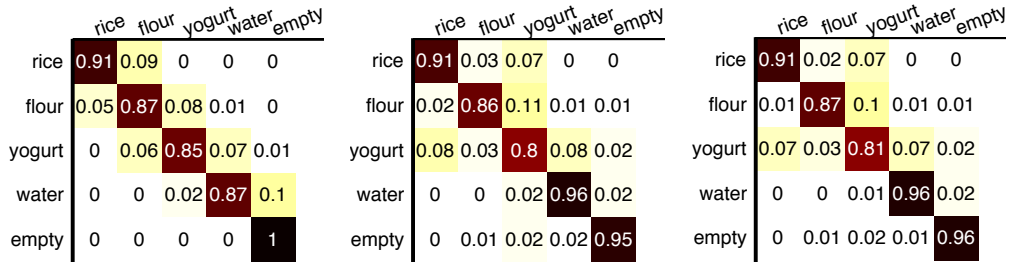
As seen in Figure 9, kNN gives higher accuracy rates in general except in case of the visual data for the *granular* class. The higher accuracy rata may be due to the fact that the data is not normally distributed generally and kNN is not relying on that assumption. Since QDA does not improve the results for the other two classes, we can conclude that kNN outperforms the QDA classifier. kNN classifier performs equally well for the three-class classification, while tactile data leads to a better accuracy for the *granular* class compared to visual data, which shows that this content is identified better with the tactile sensors.

As for the classification results on the tactile data in Figure 9(c), the highest misclassification is observed for liquid which is consistent with the findings of clustering results. Since liquid is a transition content between *empty* and *granular*, the corresponding pressure readings are mistaken for the other classes. When we look at the kNN classification values for visual data (Figure 9(d)), the worst result is obtained for the granular class using visual data and most of the misclassified *granular* data is classified as *liquid*. In the clustering results shown in Figure 8(b), we see that these two classes experience significant overlap for visual data.



(a) Tactile, QDA (b) Visual, QDA (c) Tactile, kNN (d) Visual, kNN

Fig. 9. Confusion matrices for QDA analysis of (a) tactile, (b) visual data and kNN analysis of (c) tactile and (d) visual data for three types of contents.



(a) Tactile, kNN (b) Visual, kNN (c) Tactile and Visual, kNN

Fig. 10. Confusion matrices of kNN analysis of (a) tactile, (b) visual data and (c) tactile and visual data together for five types of materials.

The visual difference between *granular* and the other types of contents is found not to be too different as compared to the other classes.

D. Classifying Five Materials with Uni-modal and Bi-modal Data

In this section, we analyze classification performance for the five classes (*empty*, *water*, *yogurt*, *flour*, *rice*) for cases when tactile and visual data are integrated. Since we achieved better results for kNN in the previous section, we performed the evaluation using this method.

As seen in Figure 10(a), for the tactile data, five classes are classified correctly up to 90% with the highest misclassification rate for *water* compared to visual data. It is classified as either *yogurt* or *empty*. This is consistent with the results from the previous experiment and also with the clustering results where we observe a large overlap between these, see Figure 8(c). As the similarities in deformation increase in the order of: (*empty*, *water*, *yogurt*, *rice*, *flour*), the classes are mostly misclassified as their neighbors which can also be seen in clustering results, Figures 8(c) and 8(d).

However, according to visual data based classification results (Figure 10(b)), visual data is found to be most informative for the *water* class while for the other classes visual data provides similar accuracy rates to the tactile data. In Figure 10(b), we see that *rice* and *flour* are mostly misclassified as *yogurt* (*liquid*), which is consistent with the clustering of the visual data based on k-means.

Visual data also increases the accuracy rate for water if we combine the tactile and visual data. We show the result of this combination in Figure 10(c), the accuracy rate for *water* increases as we observe in the previous experiment.

Nevertheless, the visual data decreases the accuracy rate of *yogurt* and *empty* of the tactile data with 4% and 5% respectively. Therefore, we conduct another experiment using raw D_{v_small} and D_{tv_small} with lower visual resolution. In the previous experiments, applying PCA may have reduced the performance of the tactile and the visual data. Thus, in this experiment, we try to analyze what kind of performance improvement we gain when we combine raw tactile and raw visual data. SVM is applied for classification because the overall accuracy rate of SVM is higher than kNN's result, i.e. for the tactile-visual data, 90% with kNN, 95% with SVM. In Figure 11, we observe 3% performance improvement for *water* and *flour*. For other materials, adding the visual data does not degrade the accuracy of the tactile data. Although the effect of the visual data is less visible than the kNN experiments, we see that visual data still add some improvement even when:

- we distort the visual data (resolution decrease).
- we use the raw tactile data without any pre-processing which may cause to lose some valuable information.
- we do not apply any dimensionality reduction algorithm while using tactile and visual data together.

In summary, combining the two modalities leads to improvements where one of them is weaker.

V. Conclusion

In this paper, we set out to investigate the feasibility of using visual and/or tactile data for classification of object contents based on container deformation observed when the container is squeezed/grasped by a robot. We perform experimental evaluation with five classes and present results using tactile and visual data individually or combined. The

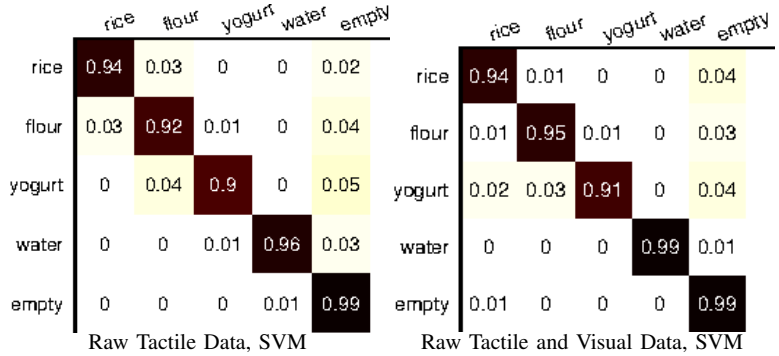


Fig. 11. Confusion matrices of SVM analysis for raw tactile in the left and tactile and visual in the right. PCA is not applied.

experimental results show that visual and tactile sensing can be utilized for identifying object content individually. In addition, when fused, they can provide complementary information with classification accuracy improvements where one of them is weaker.

The current limitations of the system lie in the fact that the tracking system relies on a model of the object that assumes objects are rigid. If the actual deformation is tracked, it may be used directly as the measurement. The container model would then need to be continuously updated, providing a better match with the observed scene, which would also allow us to model the temporal aspect of the problem more efficiently. The sensitivity level of the classifiers in terms of dealing with finer deformation changes can potentially be improved by adding more information than the pressure from the tactile sensors, such as vibration with a different action (shake/tap) and by also explicitly modeling temporal changes of the sensory data such as using Hidden Markov Models. We also plan to experiment with more objects with different contents and explore different sensor fusion methods such as decision making based on the ambiguity between the different sensory modalities.

REFERENCES

[1] J. Norman, H. Norman, A. Clayton, J. Lianekhammy, and G. Zielke, "The visual and haptic perception of natural object shape," *Perception and Psychophysics*, vol. 66, no. 2, pp. 342–351, 2004.

[2] M. A. Heller, "Visual and tactual texture perception: Intersensory cooperation," *Journal of Perception and Psychophysics*, vol. 4, no. 31, pp. 339–344, 1982.

[3] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesis–a survey," *Robotics, IEEE Transactions on*, vol. 30, no. 2, pp. 289–309, April 2014.

[4] D. Kragic, A. Miller, and P. Allen, "Real-time tracking meets online grasp planning," *IEEE International Conference on Robotics and Automation, ICRA'01*, pp. 2460–2465, 2001.

[5] B. Yoshimi and P. Allen, "Closed-loop visual grasping and manipulation," in *IEEE International Conference on Robotics and Automation*, 1996.

[6] N. Jamali and C. Sammut, "Majority voting: Material classification by tactile sensing using surface texture," *IEEE Transactions on Robotics*, vol. 27, no. 3, pp. 508–521, 2011.

[7] Z. Pezzementi, E. Plaku, C. Reyda, and G. Hager, "Tactile-object recognition from appearance information," *IEEE Transactions on Robotics*, vol. 27, no. 3, pp. 473–487, 2011.

[8] A. Petrovskaya and O. Khatib, "Global localization of objects via touch," *IEEE Transactions on Robotics*, vol. 27, no. 3, pp. 569–585, June 2011.

[9] S. Chitta, M. Piccoli, and J. Sturm, "Tactile object class and internal state recognition for mobile manipulation," in *IEEE International Conference on Robotics and Automation*, 2010, pp. 2342–2348.

[10] V. Chu, I. Mcmahon, L. Riano, C. G. Mcdonald, Q. He, J. M. Perez-tejada, M. Arrigo, N. Fitter, J. C. Nappo, T. Darrell, and K. J. Kuchenbecker, "Using Robotic Exploratory Procedures to Learn the Meaning of Haptic Adjectives," in *ICRA, IEEE Conference on Robotics and Automation*, 2013, pp. 3048–3055.

[11] M. Krainin, P. Henry, X. Ren, and D. Fox, "Manipulator and object tracking for in-hand 3d object modelling," *Int. J. Robotics Research*, vol. 30, no. 11, pp. 1311–1327, September 2011.

[12] M. Meier, M. Schopfer, R. Haschke, and H. Ritter, "A probabilistic approach to tactile shape reconstruction," *IEEE Trans. Robot.*, vol. 27, no. 3, pp. 630–635, June 2011.

[13] S. Dragiev, M. Toussaint, and M. Gienger, "Gaussian process implicit surfaces for shape estimation and grasping," in *IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2011, pp. 2845–2850.

[14] M. Bjorkman, Y. Bekiroglu, V. Hogman, and D. Kragic, "Enhancing Visual Perception of Shape through Tactile Glances," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013.

[15] A. Ude, D. Omrcen, and G. Cheng, "Making object learning and recognition an active process," *Int. J. Humanoid Robotics*, vol. 5, no. 2, pp. 267–286, 2008.

[16] A. Schneider, J. Sturm, C. Stachniss, M. Reisert, H. Burkhardt, and W. Burgard, "Object identification with tactile sensors using bag-of-features," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Oct. 2009, pp. 243–248.

[17] P. Allen, "Integrating vision and touch for object recognition tasks," Dept. of Computer Science, Columbia University, Tech. Rep., 1986.

[18] 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 *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Portland, 2013, pp. 2347–2354.

[19] J. Macqueen, "Some methods for classification and analysis of multivariate observations," in *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*, 1967, pp. 281–297.

[20] T. M. Mitchell, *Machine Learning*, 1st ed. New York, NY, USA: McGraw-Hill, Inc., 1997.

[21] R. A. FISHER, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936. [Online]. Available: http://dx.doi.org/10.1111/j.1469-1809.1936.tb02137.x

[22] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines: And Other Kernel-based Learning Methods*. New York, NY, USA: Cambridge University Press, 2000.

[23] J. C. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," in *Advances in Large Margin Classifiers*. MIT Press, 1999, pp. 61–74.

[24] J. Shlens, "A tutorial on principal component analysis," *Systems Neurobiology Laboratory, University of California at San Diego*, 2005.

[25] F. J. Massey Jr, "The kolmogorov-smirnov test for goodness of fit," *Journal of the American statistical Association*, vol. 46, no. 253, pp. 68–78, 1951.