Pauwels, Karl and Kragic, Danica, "Integrated On-line Robot-camera Calibration and Object Pose Estimation", IEEE International Conference on Robotics and Automation, Stockholm, Sweden, 2016.

Integrated On-line Robot-camera Calibration and Object Pose Estimation

Karl Pauwels and Danica Kragic

***Abstract*— We present a novel on-line approach for extrinsic robot-camera calibration, a process often referred to as *hand-eye calibration*, that uses object pose estimates from a real-time model-based tracking approach. While off-line calibration has seen much progress recently due to the incorporation of bundle adjustment techniques, on-line calibration still remains a largely open problem. Since we update the calibration in each frame, the improvements can be incorporated immediately in the pose estimation itself to facilitate object tracking. Our method does not require the camera to observe the robot or to have markers at certain fixed locations on the robot. To comply with a limited computational budget, it maintains a fixed size configuration set of samples. This set is updated each frame in order to maximize an observability criterion. We show that a set of size 20 is sufficient in real-world scenarios with static and actuated cameras. With this set size, only 100 microseconds are required to update the calibration in each frame, and we typically achieve accurate robot-camera calibration in 10 to 20 seconds. Together, these characteristics enable the incorporation of calibration in normal task execution.**

I. INTRODUCTION

To enable a robot to interact with objects observed by a camera, and to learn from these interactions, it is invaluable to know the pose of the camera in the robot's reference frame. This allows for the robot to situate perceived objects with respect to itself, as illustrated in Fig. 1. The camera pose can be obtained using a robot-camera calibration procedure, often referred to as *hand-eye calibration*. Typically such calibration requires a time-intensive procedure in which first a fixed sequence of robot configurations are determined and executed while a video is recorded. This video is then processed off-line with a vision-based pose estimation method in order to detect the pose of a calibration object or markers attached to the robot. Finally a batch optimization is performed to determine the calibration parameters. The hand-eye calibration problem is not simply a pose estimation problem since both the position or pose of the calibration object and the camera pose with respect to the robot are unknown. So as a side product of the calibration, the object's pose or position in world coordinates is obtained as well.

In many real-world situations, the camera pose relative to the robot may change over time due to changing temperature, humidity, or due to a contact or collision with the environment. It may also be necessary to quickly add cameras to the robot, or even for the robot itself to pick up a task-specific camera and put it back after completing the task. The flexible use of multiple cameras can also simplify object pose estimation [1] and arises naturally in multi-agent collaboration scenarios. An efficient on-line approach to camera calibration can be extremely useful in all these situations.

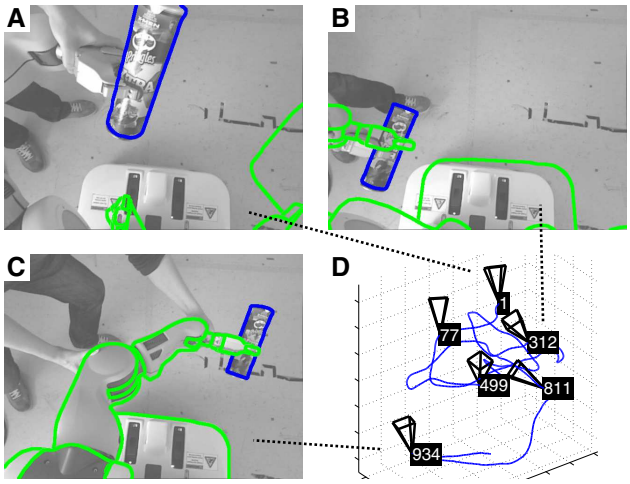

Fig. 1. Our approach improves the robot-camera calibration (green in A–C), in an on-line fashion by tracking natural objects in real-time (blue in A–C). While the object is moved (blue trajectory in D), a fixed size configuration set is updated to maximize the observability of the calibration parameters. (D) shows the final set labeled by frame number with example corresponding images in (A–C). The *Supplemental Material Video*, available at `https://youtu.be/Te_bMIK2TPs`, shows the evolution of the calibration and configuration set over time.

If the calibration can be fully integrated with other on-line visual processing, such as object pose estimation, the calibration procedure can be simplified and become an inherent part of the task execution itself, alleviating the need for a specific calibration process. Current state-of-the-art methods still require around five minutes for a complete calibration [2] and are therefore not directly applicable to on-line calibration, where only a few milliseconds are available in each frame to incorporate the new information.

Speed is critical for integrating pose tracking and calibration, and to allow both to complement each other. Our proposed method achieves this by using a simplified problem formulation and by leveraging a model-based pose estimation method. We also propose to limit the amount of samples used for calibration, and to select useful samples on-line based on an observability measure, as illustrated in Fig. 1.

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, {kpauwels,dani}@kth.se. The authors gratefully acknowledge the European projects RoboHow (FP7-ICT-288533) and RobDREAM (H2020-645403). The GPUs used for this research were donated by the NVIDIA Corporation.

The model-based pose estimation method used in our approach enables the use of everyday objects for calibration and fully integrating the procedure in normal task execution. It also allows for the use of more sophisticated calibration objects and powerful matching approaches to simplify detecting and tracking of these patterns.

II. RELATED WORK

The calibration problem considered in this work becomes much simpler if the pose of the robot can be observed directly from the camera. It is then straightforward to map the camera into the robot's reference frame. Although much progress has been made on this problem [3]–[5], visual robot pose estimation is still very difficult. The problem can be somewhat simplified by attaching markers to the robot [2], [6], [7] but this may not always be feasible and it is still difficult to estimate the 3D position of the marker in the camera frame accurately. For this reason, the less informative 2D image projection of the marker is typically used in the error formulation. Even with markers applied, the poses of the markers with respect to the robot still need to be determined and this also requires a hand-eye calibration method.

The classical methods for hand-eye calibration use a set of configurations consisting of two frames: the object pose in the camera frame, and the robot frame of the last link on which the camera is mounted. The relation between the camera and robot can then be obtained by exploiting multiple pairs of such observations [8], [9]. These methods are non-iterative and very fast, but formulate the error in terms of pose, which requires an arbitrary weighting between the translation and rotation components. Classical approaches also impose certain requirements on the configuration set which makes it difficult to apply them in an on-line fashion. These approaches however are still used to provide initial estimates for the more recent iterative approaches. The latter minimize a feature reprojection error in a similar way as bundle adjustment approaches in computer vision [2], [7], [10]. Our approach here relates to bundle adjustment in that the scene structure is part of the parameter space. However, instead of the 2D image position of a detected marker, we use the 3D position of the calibration object, as provided by a model-based pose estimation method. This simplifies the error formulation. We do not aim to do a full calibration of the camera's intrinsics and lens distortion since these factors are independent of the problem studied here. The simplified error function allows us to easily formulate the Jacobian so that we do not have to resort to costly numerical estimation methods as done by the recent approaches [2], [10].

Recently there has also been work focusing specifically on the problem of selecting a set of configurations that maximizes the accuracy of the calibration while minimizing the calibration time [7], [11], [12]. These works do not consider on-line calibration and instead generate a batch of configurations that is checked for marker visibility and collision-free, and then subsample this batch before executing it on the robot. In order to evaluate marker visibility, its approximate location needs to be known. Typically this involves markers attached to the robot. These methods cannot be used in on-line scenarios that involve static objects in unknown positions, or where the robot is executing a specific task.

There has been some work on on-line camera *registration* [6], [13] but here the marker pose with respect to the robot is assumed to be known, and only the rigid camera to body transform is estimated and filtered over time. The term on-line has also been used in the context of hand-eye calibration but referring to the use of natural scene features to estimate the camera motion, using structure-from-motion rather than object pose estimation methods [14]. Such natural scene feature tracking is less robust in the small object tracking scenario shown in Fig. 1. Apart from using a formulation similar to the early work on hand-eye calibration, in that work the configuration set still consists of pairs of observations that need to cover the configuration space. The problem of how to exploit a steady stream of trajectory information is not dealt with by these methods.

We integrate our calibration method with a real-time model-based pose estimation method. It is critical that the calibration has minimal computational requirements. We show that we require only 100 microseconds per frame. This allows us to improve the accuracy in an on-line fashion and arrive at highly accurate results in less than 20 seconds.

III. METHODOLOGY

Our approach relies on a real-time interaction between calibration and model-based object pose estimation. The object motion, estimated in the camera frame, is correlated with the motion of robot parts, as obtained from forward kinematics. We consider the two common scenarios depicted in Fig. 2 where the camera can be either actuated (A) or static (B). The calibration is updated every frame in our method so that the improvements can be immediately exploited in the object pose estimation. This allows for a more accurate prediction of the object motion in response to the robot motion. The improved object pose estimates in turn help to further improve the estimated calibration. We first give a brief overview of our pose estimation method. We then present our proposed formulation of and solution to the calibration problem, and explain how this can be accomplished on-line and with minimal computational effort.

A. Model-based Pose Estimation

We rely on our previous work[1] to estimate the pose of known objects in the scene [1], [15]. This model-based approach computes and integrates a variety of visual cues in order to estimate and track the 6-DOF pose of multiple rigid objects in real-time. It exploits different motion and depth cues, as well as visual feature descriptors. The objects are represented by textured meshes, which allows the method to synthesize images according to currently hypothesized object poses, see Fig. 3B. The pose is then updated to minimize

[1]ROS-module available at `https://github.com/karlpauwels/simtrack`

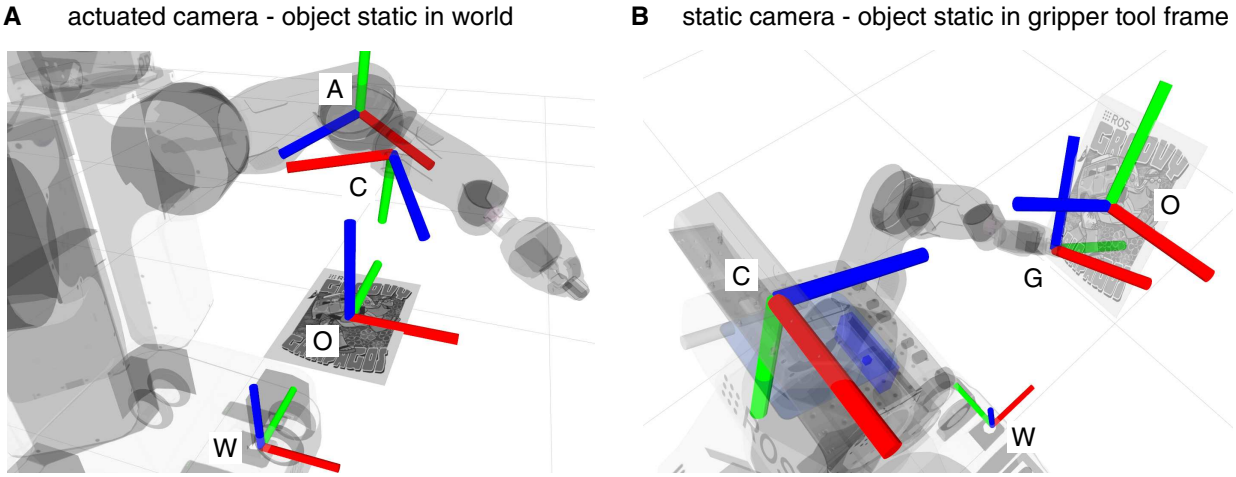

Fig. 2. Coordinate frames referred to in the (A) actuated and (B) static camera scenarios: A = arm, C = camera, O = calibration object, W = world or robot, and G = gripper tool frame. In scenario (A) the goal of the calibration procedure is to estimate the transform between camera and arm, $\mathbf{T}_{AC}$, and in scenario (B) the goal is to estimate the transform between camera and world, $\mathbf{T}_{WC}$.

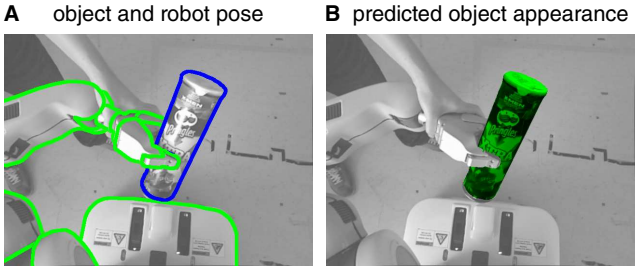

Fig. 3. (A) Blue outline shows the object pose as estimated through vision-based pose estimation and green outline shows the robot pose as obtained from the joint angle measurements. (B) Object appearance as predicted from the estimated object pose and a textured wireframe model of the object. The texture is overlaid in green and does not contain the specular reflections in (A). The region occluded by the robot is also correctly discarded.

the difference between the actual observed image and this synthesized image. Since the robot motion is known, this prediction can be improved with improved calibration accuracy. The method can also incorporate mesh models of the robot itself in the scene when synthesizing the images. The more accurate the calibration, the more precise robot-object occlusions can be predicted. This is especially relevant in the static camera scenario of Fig. 2B where the calibration object is occluded by the gripper, see also Fig. 3. The estimated object pose and the robot pose from forward kinematics are shown in Fig. 3A and the predicted object appearance is overlaid in green in Fig. 3B. The more accurate the calibration, the better the gripper occlusion can be predicted and accounted for.

B. Robot-camera Calibration

For simplicity, we only use the estimated 3D position of the calibration object in our calibration method, and not its orientation. Since orientation and position are measured in different units, it is difficult to appropriately scale their influence on the error.

We don't distinguish between robot and world frames since the robot is assumed static in the world, only its arms move. In the first scenario, Fig. 2A, we assume that the object's pose is static with respect to the world frame (W), whereas in the second we assume it is static with respect to the gripper tool frame (G). The derivation proceeds in the same way in both cases. In this section we focus only on the actuated camera scenario, but we elaborate more on the static scenario in Section IV-C.

The arm frame (A) is on the last link to the camera and there thus exists an unknown static transformation $\mathbf{T}_{AC}$ that connects the camera to the robot. For each image frame i of the video sequence, the visual object tracker provides us with the calibration object's 3D position in the camera frame, $\mathbf{p}_C^i$. We also obtain the arm pose, $\mathbf{T}_{WA}^i$, from the joint angle measurements and forward kinematics. Since the object's unknown position in the world frame, $\mathbf{p}_W$, is assumed static, we can express the 3D measurement error in the camera frame as follows:

$$\mathbf{e}_i = \mathbf{p}_C^i - \mathbf{T}_{CA}\,\mathbf{T}_{AW}^i\,\mathbf{p}_W \ . \tag{1}$$

The unknowns are $\mathbf{T}_{CA} = \mathbf{T}_{AC}^{-1}$ and $\mathbf{p}_W$ and appear as a product in the error function. Note that additional static scene information does not directly help in disambiguating the problem since for every additional measurement $\mathbf{p}_C$ an additional variable $\mathbf{p}_W$ needs to be estimated. Instead, measurements need to be combined across multiple frames to resolve the calibration problem. For simplicity, and as commonly done in this work [11], we assume that the joint angle errors are negligible and that the $\mathbf{T}_{AW}^i$, obtained through the forward kinematics, are exact. Assuming Gaussian noise in the measurements $\mathbf{p}_C^i$, we can obtain the maximum likelihood estimates as the solution to the following least-squares problem:

$$F(\boldsymbol{\theta}) = \sum_i \mathbf{e}_i{}^\top \mathbf{e}_i = \mathbf{f}(\boldsymbol{\theta})^\top \mathbf{f}(\boldsymbol{\theta}) \ , \tag{2}$$

where

$$\mathbf{f}(\boldsymbol{\theta}) = \left[\mathbf{e}_1{}^\top \dots \mathbf{e}_n{}^\top\right]^\top \ , \tag{3}$$

and the parameter space $\boldsymbol{\theta} = \{\mathbf{T}_{CA}, \mathbf{p}_W\}$. The error function is highly nonlinear as it involves products of the unknown parameters. To find the least-squares estimate, we can linearize $\mathbf{f}$ using the Jacobian $\mathbf{J}(\tilde{\boldsymbol{\theta}}) = \frac{\delta \mathbf{f}}{\delta \boldsymbol{\theta}}(\tilde{\boldsymbol{\theta}})$ at the linearization point $\tilde{\boldsymbol{\theta}}$:

$$\mathbf{f}(\boldsymbol{\theta}) = \mathbf{f}(\tilde{\boldsymbol{\theta}} + \Delta\boldsymbol{\theta}) \approx \mathbf{f}(\tilde{\boldsymbol{\theta}}) + \mathbf{J}(\tilde{\boldsymbol{\theta}})\Delta\boldsymbol{\theta} \ . \tag{4}$$

To facilitate the derivation of the Jacobian we first expand the measurement error (1) as follows:

$$\mathbf{e}_i = \mathbf{p}^i_C - \mathbf{R}_{CA}\left(\mathbf{R}^i_{AW}\,\mathbf{p}_W + \mathbf{t}^i_{AW}\right) - \mathbf{t}_{CA} \ , \tag{5}$$

with $\mathbf{R}$ and $\mathbf{t}$ the rotation matrix and translation vector. Note that we freely interchange here between homogeneous and Cartesian coordinates. It can now be seen that each measurement contributes to the Jacobian in the following way:

$$\frac{\delta \mathbf{e}_i}{\delta \mathbf{p}_W} = -\mathbf{R}_{CA}\mathbf{R}^i_{AW} \ , \tag{6}$$

$$\frac{\delta \mathbf{e}_i}{\delta \mathbf{t}_{CA}} = -\mathbf{I}_3 \ . \tag{7}$$

Regarding the rotation update, we express the Jacobian in terms of the incremental rotation, linearized using the small-angle approximation of the rotation matrix:

$$\mathbf{e}_i(\boldsymbol{\omega}_{CA}) = -e^{[\boldsymbol{\omega}_{CA}]_\times}\mathbf{R}_{CA}\mathbf{p}^i_A + \mathbf{p}^i_C - \mathbf{t}_{CA} \tag{8}$$

$$\approx -\left(\mathbf{I}_3 + [\boldsymbol{\omega}_{CA}]_\times\right)\mathbf{R}_{CA}\mathbf{p}^i_A + \mathbf{p}^i_C - \mathbf{t}_{CA} \tag{9}$$

$$= \left[\mathbf{R}_{CA}\mathbf{p}^i_A\right]_\times \boldsymbol{\omega}_{CA} + \dots \ , \tag{10}$$

where $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]^\top$ corresponds to the incremental rotation axis and angle, and $[\boldsymbol{\omega}]_\times$ is the skew-symmetric matrix corresponding to the vector cross-product. We have compacted $\mathbf{p}^i_A = \mathbf{R}^i_{AW}\,\mathbf{p}_W + \mathbf{t}^i_{AW}$ to condense the notation. The partial derivative to the incremental rotation is now:

$$\frac{\delta \mathbf{e}_i}{\delta \boldsymbol{\omega}_{CA}} = \left[\mathbf{R}_{CA}\mathbf{p}^i_A\right]_\times \ . \tag{11}$$

The linearized error function can be used in a Gauss-Newton or Levenberg-Marquardt algorithm to iteratively update the parameters:

$$\Delta\boldsymbol{\theta} = -\left[\mathbf{J}(\tilde{\boldsymbol{\theta}})^\top\mathbf{J}(\tilde{\boldsymbol{\theta}})\right]^{-1}\mathbf{J}(\tilde{\boldsymbol{\theta}})^\top\mathbf{f}(\tilde{\boldsymbol{\theta}}) \ . \tag{12}$$

At each iteration the incremental rotation matrix is obtained using the exponential map, and incorporated into the previous estimate at iteration $k-1$ to obtain the new estimate at iteration k:

$$\mathbf{R}^k_{CA} = e^{[\boldsymbol{\omega}^k_{CA}]_\times}\,\mathbf{R}^{k-1}_{CA} \ . \tag{13}$$

So far we have ignored the reliability of the object pose estimates. The reliability in most visual 3D estimation methods is inversely proportional to distance from the camera since a larger triangulation baseline is obtained the closer the object is to the image. Since we are dealing with a single object here throughout the entire calibration sequence, we simply perform a weighted least-squares optimization where each sample is weighted by the inverse distance. So instead of minimizing (2) directly, we use the weighted version:

$$\mathbf{e}^w_i = w_i\mathbf{e}_i \ , \tag{14}$$

where $w_i = 1/z(\mathbf{p}^i_C)$ with $z(\mathbf{p})$ the depth-component of $\mathbf{p}$. Note that this is easier than formulating the projection in (2) and is made possible due to the availability of 3D object position information. This weighting can also be used to incorporate other sources of reliability information, or to implement an iteratively reweighted least-squares procedure that can handle large outliers in the pose estimation [16].

C. On-line Calibration

In order to perform the calibration on-line with a limited computational budget, we maintain and optimize a fixed size set of samples over time. We refer to this as the *configuration set*, $\mathcal{S}$, in the remainder. Each sample j in this set consists of the estimated object pose in the camera frame, $\mathbf{p}^j_C$, and the arm pose obtained through forward kinematics, $\mathbf{T}^j_{AW}$. At each time instance, the newly arriving sample is evaluated against all other samples in the configuration set in order to greedily maximize an observability index. In accordance with [7], [12] we use the following index:

$$\Omega = \left(\prod_j^{n_c}\sigma_j\right)^{\frac{1}{n_c}}(n_r)^{-\frac{1}{2}} \ , \tag{15}$$

where σ_j are the singular values obtained from the Singular Value Decomposition of the Jacobian $\mathbf{J}$, and n_r and n_c the number of rows and columns of the Jacobian, equal to three times the number of samples and nine respectively. This observability index can be computed efficiently using the following:

$$\prod_j^{n_c}\sigma_j = \sqrt{|\mathbf{J}^\top\mathbf{J}|} \ , \tag{16}$$

where $|\mathbf{A}|$ is the determinant of $\mathbf{A}$. Maximizing the observability index, or the product of the singular values, is equal to minimizing the Eigenvalues of $\left(\mathbf{J}^\top\mathbf{J}\right)^{-1}$. The latter corresponds to the covariance matrix of the parameter estimates. By maximizing the index, we thus reduce the uncertainty of the parameters. This determinant-based criterion has also been shown to result in the most precise calibration from a theoretical perspective [17].

Our proposed approach for maximizing the observability on-line is summarized in Algorithm 1. Starting from an initial estimate of the calibration parameters, we fill up the configuration set with the samples corresponding to the first image frames and update the estimate and index. We discuss the initial estimate in more detail in Section IV. For each new image frame, we evaluate the observability index obtained by replacing each sample from the set with the new sample, one sample at a time. We then perform the replacement for the maximal index if it improves the index obtained in the previous frame. We finally update the calibration estimate and proceed with the next image frame.

Algorithm 1: On-line updating the configuration set

Input: N: configuration set size, L: sequence length,
initial estimate $\{\mathbf{T}_{CA}^0, \mathbf{p}_W^0\}$,
$\mathbf{x}_t = \{\mathbf{p}_C^t, \mathbf{T}_{AW}^t\}$:measurements at time t
Output: final estimate: $\{\mathbf{T}_{CA}^L, \mathbf{p}_W^L\}$

```
// initialization
```
for $t = 1$ **to** N **do**
  $\mathbf{s}_t = \mathbf{x}_t$
$\mathcal{S}_N^* = \{\mathbf{s}_1, \ldots, \mathbf{s}_N\}$
// $g()$ computes the observability index
$\Omega_N^* = g(\mathcal{S}_N^*)$
// $h()$ updates the calibration estimate
$\{\mathbf{T}_{CA}^N, \mathbf{p}_W^N\} = h\left(\mathcal{S}_N^*\right)$
// on-line updating
for $t = N+1$ **to** L **do**
  $\mathcal{S}_\cup = \mathcal{S}_{t-1}^* \cup \mathbf{x}_t$
  $j^* = \text{argmax}_j\, g(\mathcal{S}_\cup \setminus \mathbf{s}_j)$
  $\Omega = g(\mathcal{S}_\cup \setminus \mathbf{s}_{j^*})$
  if $\Omega > \Omega_{t-1}^*$ **then**
    $\mathcal{S}_t^* = \mathcal{S}_\cup \setminus \mathbf{s}_{j^*}$
    $\Omega_t^* = \Omega$
    $\{\mathbf{T}_{CA}^t, \mathbf{p}_W^t\} = h\left(\mathcal{S}_t^*\right)$
  else
    $\mathcal{S}_t^* = \mathcal{S}_{t-1}^*$
    $\Omega_t^* = \Omega_{t-1}^*$

IV. EXPERIMENTS

This section contains real-world calibration results obtained in the actuated and static camera scenarios. Although we do not consider algorithm initialization in this work, we have found the method to be very robust to large errors in the initialization values. In the examples below we initialize the algorithm randomly and far from the solution. It is usually easy to obtain a rough initial guess, or to improve upon a previous calibration. In the absence of any prior knowledge about the setup, a classical hand-eye calibration method can also be used for initialization.

We also attempted a comparison with an iterated extended Kalman filter using the measurement function Jacobian as derived above (6),(7),(11). This problem is hard since multiple configurations are required in order to fully observe the state and the problem is highly nonlinear. Due to the linearization, the filter quickly diverged, unless the initial state estimate was very close to the solution. The nonlinearity of the problem appears better represented in the set of configurations maintained by our method.

A. Computational Requirements

The computation time of the observability index (16) scales linearly with increasing configuration set size, but since it needs to be evaluated for each sample in the set, the combined effect is superlinear. Very little time is actually required with a small number of configurations, as illustrated in Fig. 4. Even when maintaining 100 configurations, only around one millisecond is required per frame to update the calibration. These times were obtained with a C++ implementation running on a single core of an Intel Core i7-4790K CPU. In the remainder we always used a configuration set of size 20, which only required 100 microseconds.

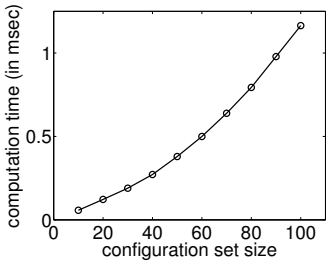

Fig. 4. Computation times scale superlinearly with configuration set size. Accurate results can be obtained with as little as 20 configurations, and require around 100 microseconds per frame.

B. Actuated Camera

We first consider the scenario with a camera rigidly attached to the robot as shown in Fig. 2A. For this example we used the left arm camera of the PR2 robot. The calibration object is assumed static in the scene. Determining whether or not this is actually the case is outside the scope of this work.

For initialization, we set $\mathbf{p}_W = [0,\, 0,\, 0]^\top$ and randomly displaced $\mathbf{T}_{CA}$ from the PR2's factory calibration, which is not very accurate, according to a randomly generated rigid transform with translation magnitude of one meter and a rotation angle of 90 degrees applied to a random rotation axis. This initial configuration is shown in Fig. 5A, overlaid in green, and is clearly far from the solution. Since we used a configuration set of size 20, the first estimate relied on the first 20 frames. This already greatly improved the estimate, see Fig. 5B, but the hand and base were still misaligned. We continued by manually moving the camera above and away from the object. The final estimate of the complete camera path over the course of the calibration is illustrated in Fig. 6B. This figure also highlights the camera positions that were part of the final configuration set. Note how this set efficiently summarizes the camera's trajectory, and retains frames that correspond to widely-separated configurations. Interestingly, the final configuration typically contains short subsequences. This can be related to the critical factors and criteria for improving hand-eye calibration accuracy identified in [18]. One of the criteria states that small translational movements of the hand can improve the error in the translation component of the calibration [19].

Fig. 6A contains the evolution of the root mean squared error (RMSE) of the error function (2), evaluated over the entire sequence, together with the observability index (16). The RMSE is computed using the weighted version (14), but the weights are normalized, so that the error still corresponds to a distance measure. The measurements are only shown in Fig. 6A when the configuration set is updated. The RMSE quickly drops below one centimeter. For comparison, we also show the RMSE using an exhaustive estimation

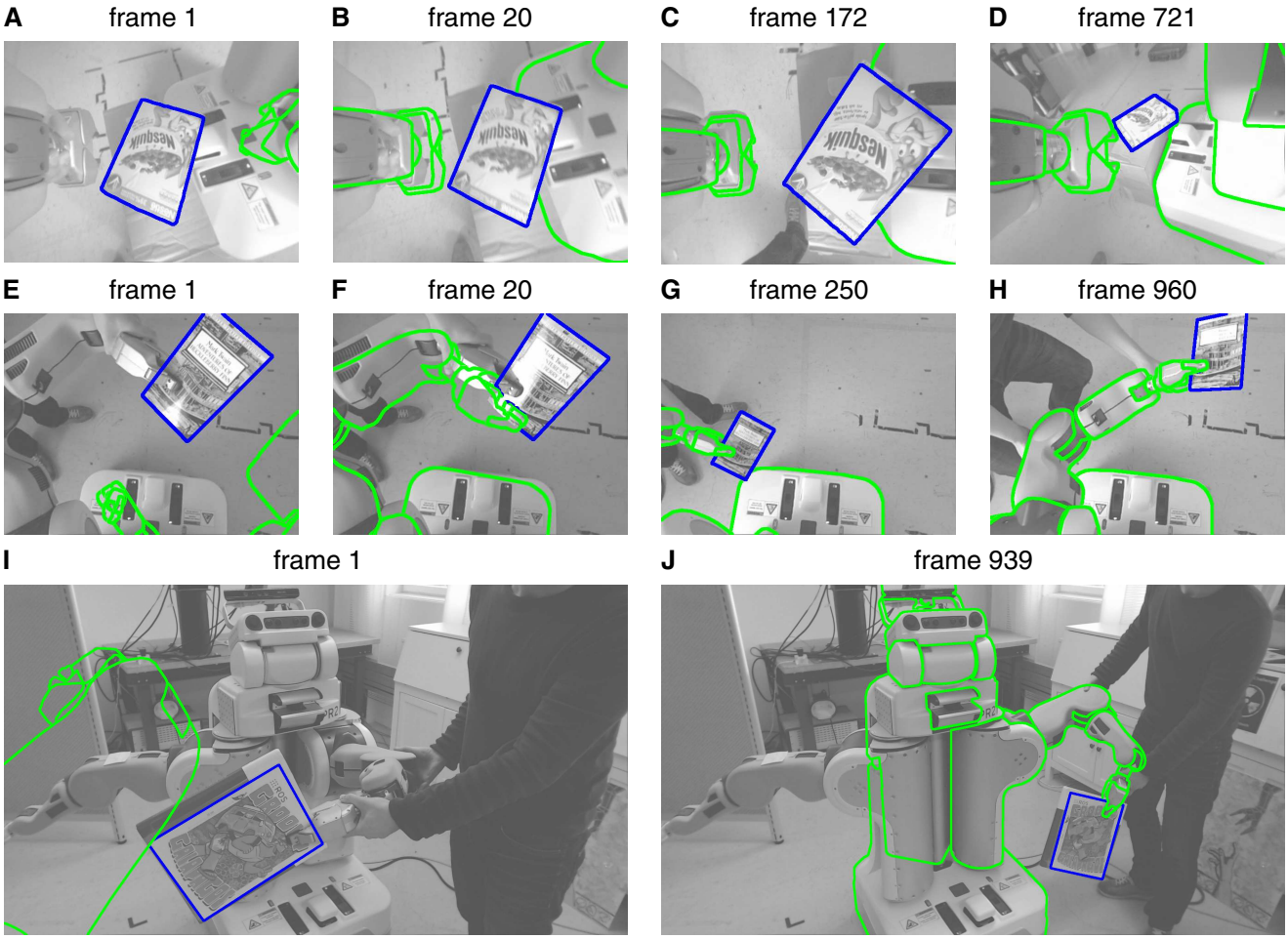

Fig. 5. Snapshots from the on-line calibration procedure applied to real-world sequences involving an actuated camera (A–D), a static camera on the robot (E–H), and a static camera external to the robot (I–J). The robot's right arm is not shown in panel (J) since it was broken at the time of the experiment and the joint encoder offsets were unknown. The full sequences are shown in the *Supplemental Material Video*.

that incorporates all measurements up to the current frame, see solid line in Fig. 6A, and which can be considered optimal for this problem. The proposed on-line estimation behaves very similar to the exhaustive estimation, but uses far fewer computational resources. The observability measure stabilizes after about 400 frames. This stabilization can be detected and used as a stopping criterion, but we did not consider this in this work. Note also how larger gaps occur later in the sequence indicating that the calibration is stabilizing and the configuration set does not require further updates. After frame 800, the new samples no longer improve the observability. We show two more frames from the sequence in Fig. 5C,D. Fig. 5C shows the improvements after about six seconds, and Fig. 5D shows the end of the sequence, where the calibration has stabilized. The latter shows how both the gripper and robot base and body are very accurately projected in the image. It also shows the high dynamic range over which the pose estimation algorithm can track the object. The evolution of tracking and calibration for the entire sequence is shown in the *Supplemental Material Video*.

C. Static Camera

The second scenario requires a static camera, either connected to or external to the robot, to be calibrated with respect to the robot. This can be accomplished by rigidly attaching the calibration object to the robot, as illustrated in Fig. 2B, and presenting it to the camera in various poses. Since the object is now being moved by the robot, it is no longer static in the world frame, as in the actuated camera scenario. It is however static with respect to the robot gripper. Using the frames illustrated in Fig. 2B we can thus reformulate the (unweighted) measurement error as:

$$\mathbf{e}_i = \mathbf{p}_C^i - \mathbf{T}_{CW}\,\mathbf{T}_{WG}^i\,\mathbf{p}_G \ . \tag{17}$$

This has exactly the same form as (1) with regard to parameters and measurements, and also the Jacobian can be derived in the same way by replacing the corresponding frames.

We show results for two different setups in this section. In the first we used the PR2's head-mounted Kinect camera and considered it static, see Fig. 5E–H. We did not use the depth signal for object tracking to make the results more comparable across the different experiments. For the second

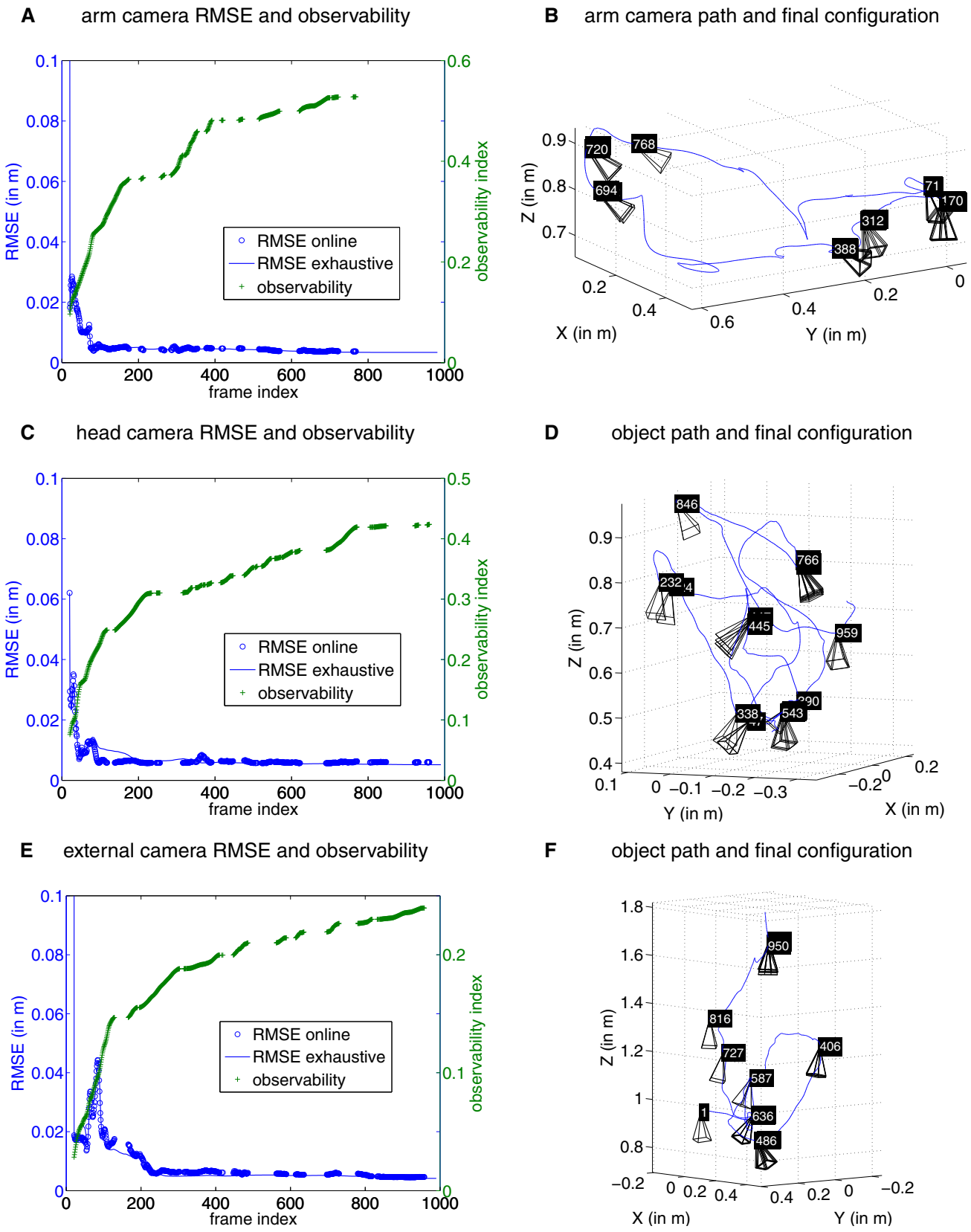

Fig. 6. Calibration results using the PR2's arm camera (A,B), head-mounted Kinect (C,D), and an external camera (E,F). (A,C,E) contain the evolution of the RMSE for the on-line procedure and the exhaustive estimation (using all preceding frames), together with the observability measure. The RMSE is always evaluated on the complete sequence. (B) Camera path for the actuated camera scenario together with the samples contained in the final configuration set, as marked by the camera symbols and labeled with frame number. (D,F) Same as (B) for the static camera scenarios but showing the object pose trajectory in the camera frame.

setup we used a Logitech C920 HD webcam, mounted on a tripod external to the robot, see Fig. 5I–J. This camera provided 1920×1080 resolution video at 30 Hz to the tracker.

In the first setup, the initial state is again randomly disturbed with translation magnitude one meter and 90 degrees rotation offset and, as illustrated in Fig. 5E, clearly far from the solution. After accumulating the first 20 frames in the initial configuration set, the estimate shown in Fig. 5F was obtained. Although much more accurate, the projected gripper pose was further off than in the previous scenario. The particular object used here is very reflective as can be seen in Fig. 5E,F. This highlights the importance of using a robust object tracking framework for calibration. Fig. 6C shows the evolution of the RMSE and observability over the course of the sequence. The observability index again stabilizes towards the end of the sequence and fewer updates occur later in the sequence. Fig. 6D now shows the object path in the camera frame, as estimated by the vision algorithm, with the camera symbols highlighting the object poses of the samples in the final configuration set. As before, the set compactly represents the trajectory of the calibration object over the entire sequence. Two more frames are shown in Fig. 5G,H acquired in the middle and at the end of the sequence. Both show a precise alignment of the projected robot pose in the camera image. See the *Supplemental Material Video* for the evolution of the calibration over the complete sequence.

We also obtained highly accurate results in the second setup with an external camera, see Fig. 5J. Fig. 5I contains the initial state, which was now randomly disturbed from the zero position. The PR2's right arm is not shown in Fig. 5J because it was broken at the time of the experiment. This made it impossible to calibrate the joint encoder offsets and rely on the forward kinematics. Fig. 6E,F again show a similar evolution of the RMSE, observability, and object path during calibration.

Fig. 5 also demonstrates the usefulness of integrating calibration and tracking for occlusion detection. At the initial state, Fig. 5F, it is difficult to correctly discard the region occluded by the gripper in tracking. In fact, the predicted robot occlusion covers up a visible part of the object. We only consider occlusions later on in the calibration. Once the calibration is more accurate, Fig. 5G, the occluded region can be ignored and the grasped object can be tracked more easily.

V. CONCLUSIONS

We have proposed a highly efficient on-line robot-camera calibration method that is embedded in a model-based pose estimation algorithm. Our approach is simple and can be used to calibrate actuated as well as static cameras. We have demonstrated experimentally that precise results can be obtained with a modular approach where intrinsic camera calibration and pose estimation are performed separately from robot-camera calibration, as opposed to optimizing all parameters simultaneously. Our approach allows using everyday objects for calibration, provided their models are available, and can extract calibration information from natural pose trajectories. This opens up the possibility to better integrate calibration with task execution, *e.g.* by automatically switching to a calibration mode for the actuated cameras once the scene is known to be static. This could be determined on the basis of an independent sensor, such as a static head camera. In turn, when an object is known to be grasped rigidly, *e.g.* based on tactile sensor feedback, a calibration mode can be activated for the static cameras as well. Future work will concentrate on using the observability obtained on-line to guide the robot motion itself towards regions that are more likely to improve calibration.

REFERENCES

[1] K. Pauwels and D. Kragic, "Simtrack: A simulation-based framework for scalable real-time object pose detection and tracking," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Hamburg, Germany, 2015, pp. 1300–1307.

[2] O. Birbach, U. Frese, and B. Buml, "Rapid calibration of a multi-sensorial humanoids upper body: An automatic and self-contained approach," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 420–436, Apr. 2015.

[3] P. Hebert, N. Hudson, J. Ma, and J. Burdick, "Dual arm estimation for coordinated bimanual manipulation," in *ICRA*, 2013, pp. 120–125.

[4] X. Gratal, C. Smith, M. Björkman, and D. Kragic, "Integrating 3D features and virtual visual servoing for hand-eye and humanoid robot pose estimation," in *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Oct. 2013, pp. 240–245.

[5] T. Schmidt, R. Newcombe, and D. Fox, "DART: Dense articulated real-time tracking," *Proceedings of Robotics: Science and Systems, Berkeley, USA*, 2014.

[6] N. Dantam, H. B. Amor, H. Christensen, and M. Stilman, "Online camera registration for robot manipulation (presented)," in *International Symposium on Experimental Robotics*, 2014.

[7] D. Maier, S. Wrobel, and M. Bennewitz, "Whole-body self-calibration via graph-optimization and automatic configuration selection," in *ICRA*, May 2015, pp. 5662–5668.

[8] R. Tsai and R. Lenz, "Real time versatile robotics hand/eye calibration using 3D machine vision," in *ICRA*, Apr. 1988, pp. 554–561 vol.1.

[9] K. Daniilidis, "Hand-Eye Calibration Using Dual Quaternions," *The International Journal of Robotics Research*, vol. 18, no. 3, pp. 286–298, Mar. 1999.

[10] V. Pradeep, K. Konolige, and E. Berger, "Calibrating a Multi-arm Multi-sensor Robot: A Bundle Adjustment Approach," in *Experimental Robotics*, O. Khatib, V. Kumar, and G. Sukhatme, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, vol. 79, pp. 211–225.

[11] Y. Sun and J. Hollerbach, "Active robot calibration algorithm," in *ICRA*, May 2008, pp. 1276–1281.

[12] H. Carrillo, O. Birbach, H. Taubig, B. Bauml, U. Frese, and J. Castellanos, "On task-oriented criteria for configurations selection in robot calibration," in *ICRA*, May 2013, pp. 3653–3659.

[13] N. Dantam, H. B. Amor, H. Christensen, and M. Stilman, "Online multi-camera registration for bimanual workspace trajectories (presented)," in *International Conference on Humanoid Robots*, 2014.

[14] N. Andreff, R. Horaud, and B. Espiau, "On-line hand-eye calibration," in *3-D Digital Imaging and Modeling, 1999. Proceedings. Second International Conference on*, 1999, pp. 430–436.

[15] 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.

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

[17] Y. Sun and J. Hollerbach, "Observability index selection for robot calibration," in *ICRA*, May 2008, pp. 831–836.

[18] R. Tsai and R. Lenz, "A new technique for fully autonomous and efficient 3D robotics hand/eye calibration," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 3, pp. 345–358, June 1989.

[19] J. Schmidt and H. Niemann, "Data Selection for Hand-eye Calibration: A Vector Quantization Approach," *The International Journal of Robotics Research*, vol. 27, no. 9, pp. 1027–1053, Sept. 2008.