IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, May 2016.

Focused Online Visual-Motor Coordination for a Dual-Arm Robot Manipulator

Seongyong Koo and Sven Behnke

***Abstract*— Coordination between visual sensors and robot manipulators is necessary for successful manipulation. This paper proposes a novel visual-motor coordination method that performs online parameter estimation of an RGB-D camera mounted in a robot head without any external markers. Through self-observation of a dual-arm robot manipulator, the method updates parameters to reduce the discrepancy between observed point cloud data and 3D mesh models of the current robot configuration. With the estimated parameters at each time step, visual data is adjusted to the focused workspace of the 14 DOF dual-arm robot manipulator. The online and real-time algorithm was developed by using a GPU-based particle filtering method. Experimental results show that our method outperforms state-of-the-art offline registration methods in terms of accuracy and computation time. We also analyzed the dependence of the results on prior parameters to demonstrate the online capability of our method.**

I. Introduction

The ability to control hand movements guided by vision enables robots to manipulate objects precisely. Visual-motor coordination (eye-hand coordination), which refers to integrating vision information with the movements of the body or parts of the body [1], is a prerequisite step to generate visually accurate, energy and time-efficient robot movements. However, as shown in the top of Fig. 1, self-observation and simulation of the robot body are frequently mismatched due to kinematic errors between robot manipulators and the vision sensor. With the discrepancy between the expected and observed bodies, the robot cannot perform accurate manipulation tasks. In Fig 1, for example, the two vectors from both end-effectors to the target object estimated from the uncalibrated scene are inconsistent with the ones from the calibrated scenes shown at the bottom. This calls for robot calibration methods that estimate uncertain parameters of kinematic chains from external sensors [2].

Traditionally, robot calibration is performed once as an initial step by using marker-based computer vision techniques [3], but this approach has three issues: First, the marker-based camera calibration results, which are used as a reference input for the robot calibration, include errors that cannot be ignored for manipulation. This error propagates to the robot calibration step, which in turn estimates incorrect robot kinematic information. Second, in the case of pose changes of sensors by unwanted physical disturbances, the calibration step should be conducted again. This frequently happens to the sensors mounted on the robot while the robot operates. Third, a small perturbation of the kinematic model at the calibration phase can cause large position error in different robot configurations. This restricts robot manipulability unless the calibration process is performed in the entire workspace, which is inefficient and sometimes infeasible for unseen areas.

The authors are with the Autonomous Intelligent Systems Group, Computer Science Institute VI, University of Bonn, Germany. Email: koosy@ais.uni-bonn.de, behnke@cs.uni-bonn.de

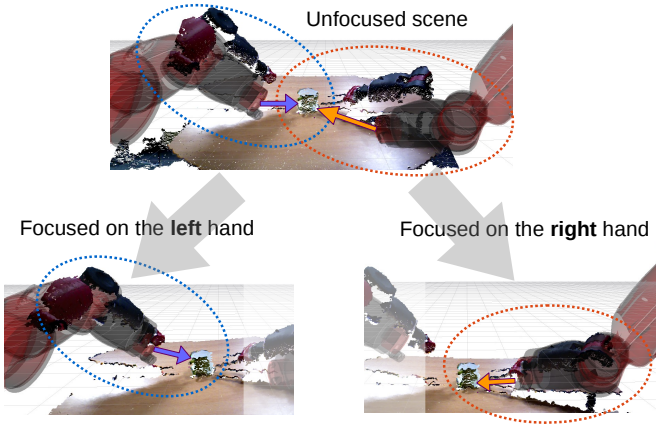


Fig. 1: Example of the focused visual-motor coordination of 3D models of a dual-arm robot manipulator and the measured point cloud data. From the unfocused visual scene (top), the method calibrates visual measurements such that the observed body parts best match the 3D mesh body model, focused on the given body coordinate. The focus frame can be changed according to the task, e.g. grasping a cup by the left hand (bottom left) or by the right hand (bottom right).

This paper proposes a novel visual-motor coordination method that performs online adjustment of visual measurements such that they are fitted to the expected robot body of interest without any external markers. In order to overcome the aforementioned difficulties of robot calibration, it features:

- estimating extrinsic parameters of an RGB-D camera on the robot head by comparing a 3D mesh model of the manipulator with measured point cloud data of the robot body,
- online estimation that is robust to external disturbances and model uncertainty, and
- continuous parameter updates adaptive to the manipulation area of interest.

II. Related Work

Robot calibration has been considered as a part of the system development process by trying to find true parameter

values accurately at the initial step [4]. Marker-based computer vision techniques have been widely used to estimate not only the 6D pose of a camera, but also its kinematic relation to the robot coordinates. Asfour et al. [5] proposed a head-eye calibration method to estimate a stereo camera pose relative to the joints of a humanoid head by using a marker and a least-squares optimization approach. Hubert et al. [6] developed a Bayesian approach to calibrate hand-eye kinematics of a 7 DOF arm and 2 DOF neck. They introduced an efficient method that incorporates prior model knowledge to optimize a multitude of parameters from only a few observations. More recently, a similar marker-based approach was applied to calibrate multiple sensors [7] and multiple kinematic chains [8]. Birbach et al. [7] used specially designed markers on wrists to calibrate a pair of cameras, an RGB-D camera, and an Inertial Measurement Unit mounted on a humanoid head, with respect to its kinematic chain. Maier et at. [8] calibrated a whole-body kinematic model of a humanoid robot. They used markers attached to the four end-effectors and a least-square optimization method to estimate all the parameters.

With the assumption that the initially estimated parameters possibly contain uncertain values due to the measurement error and external disturbances, online calibration methods have been studied to compensate the uncertainty. In many cases of the online approach, marker-based methods cannot be applied due to their requirements of special spaces and procedures for setup. Park et al. [9] proposed a structured laser module attached to the end-effector, and its accurate position was detected by a stationary camera. With the measured position, Extended Kalman Filtering was used to estimate kinematic parameters of a 7 DOF humanoid manipulator. In order to compensate kinematic parameter errors of a humanoid robot head, Moutinho et al. [10] proposed an online estimation process from relative encoders, inertial sensors, and visual data. This method showed robustness to noise and abrupt changes in the parameters, and it was further extended to the stereo camera calibration. Vicente et al. [11] more recently proposed an online hand-eye calibration method with GPU acceleration. They utilized a 3D CAD model of a robot hand and a particle filter approach to estimate its pose and a set of joint offsets to calibrate the arm.

While these online methods estimated the true pose of a robot body part (a head in [10] and a hand in [11]), our approach introduces a novel concept of *virtual coordinate* that moves around a *focus frame* to coordinate visual scene around the frame to the best of nearby body parts. The online estimation of the virtual coordinate is more accurate and flexible for dual-arm robot manipulators that dynamically change the manipulator in a wide working space. Moreover, the markerless approach is not only more precise than marker-based approaches, but it is also easier to configure the robot system without an initial complex calibration step.

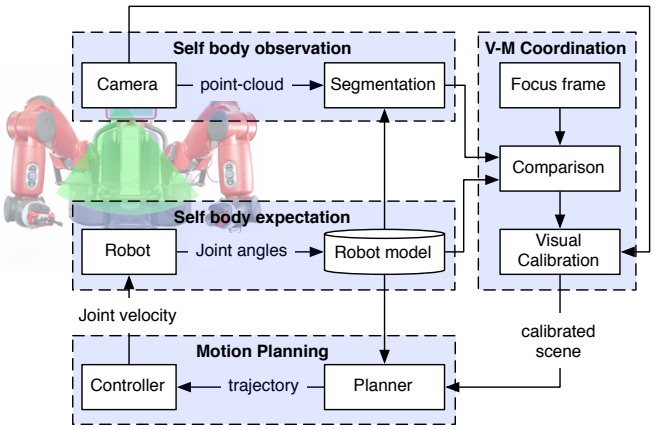


Fig. 2: Architecture of the focused online visual-motor coordination. The coordination is the process of reducing the discrepancy between two sets of point cloud data from the *self body observation* and *self body expectation* modules, concentrated on the given focus frame. It produces a calibrated scene to be used in the motion planning module at each time step.

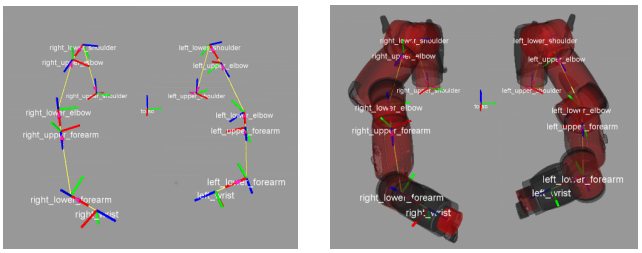
III. Focused Online Visual-motor Coordination

Fig. 2 gives an overview of the proposed visual-motor coordination framework. Visual measurements come from an RGB-D camera on the robot head and motor information is given by measured joint angles and robot descriptions including kinematic structure and 3D mesh data. The *Self body observation* module extracts body-related visual information which will be compared with expected body information from the *self body expectation* module. The visual data captured from the camera is adjusted to reduce the discrepancy between the two body representations, and the calibrated visual information will be used for motion planning with the robot model. This process is performed every time any part of the robot body is observed.

A. Self Body Expectation

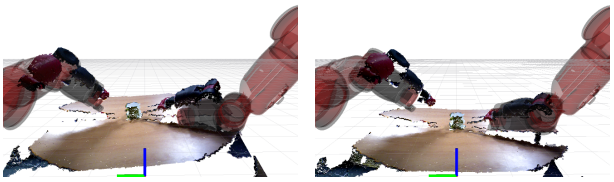
At each time step t, joint angles of the robot $\mathbf{q}_t$ yield a configuration of the robot body posture with a predefined robot model that consists of kinematic chain parameters and 3D mesh models for each link $\mathcal{R}_t = \{\mathbf{\Theta}_t, \mathcal{M}\}$. Here, the kinematic chain is represented as a tree structure, $\mathbf{\Theta} = \{\theta_i^j | i < j \leq n-1\}$, where n is the total number of degrees of freedom (DOF) and i, j are the indices of each joint starting from the base (0) to the end effector $(n-1)$. Each $\theta_i^j = \{\mathbf{e}_i^j, \mathbf{x}_i^j\}$ refers to the relative pose of the frame j to the frame i, which is defined by Euler angles $\mathbf{e} = \{r, p, y\}$ and link parameters $\mathbf{x} = \{x, y, z\}$. Given joint angles $\mathbf{q}_t$, corresponding Euler angles can be specified, but the parameters of the robot link $\mathcal{X} = \{\mathbf{x}_i^j | i < j \leq n-1\}$ should be predefined. Fig. 3a shows the $n = 14$ DOF kinematic chain of the Baxter dual-arm robot.

Another component of the robot model, $\mathcal{M}$ consists of a 3D mesh model for each link, $\mathcal{M} = \{\mathbf{M}_i\}_{i=0}^n$. Here, we use



(a) The kinematic chain of our 14 DOF dual-arm robot manipulator (b) An example of the expected body with 3D CAD models

Fig. 3: Expected body given joint angles, a robot model can be represented by a kinematic chain (a) and 3D models for each link (b).



(a) Visual focus on the left arm (b) Visual focus on the right arm

Fig. 4: Visual data coordinated to two different focus frames. Discrepancies between the model and the visual data increase with the distance from the focus frame.

triangle meshes that comprise a set of triangle faces with a face normal, $\mathbf{M}_i = \{\mathbf{f}_j, \mathbf{n}_j\}_{j=0}^{n_i}$, where n_i is the number of triangle faces in the mesh model of link i. Each face has three vertices $\mathbf{f} = \{\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2\}$, where $\mathbf{v}, \mathbf{n} \in \mathbb{R}^3$. Given $\mathbf{\Theta}_t$, all elements $\mathbf{M}_i$ can be transformed into the Cartesian space with each corresponding link coordinate to the robot base, θ_o^i. Let $\widehat{\mathcal{M}}_{o,t}$ denote the transformed mesh model at time t, which we refer to as the *expected body*. Fig. 3b shows an example of the expected body.

B. Self Body Observation

From an RGB-D camera, point cloud data is obtained at each time step, $\mathbf{P}_t = \{\mathbf{p}_1, ..., \mathbf{p}_n\}$, each of which contains the 3-dimensional position and RGB color of points on the object surface, $\mathbf{p}_i = \{\mathbf{x}, \mathbf{c}\}$, where $\mathbf{x}, \mathbf{c} \in \mathbb{R}^3$. With the assumption that the camera pose relative to the robot base frame in the previous time step $\theta_{o,t-1}^c$ has no considerable errors, the measurement point cloud, $\mathbf{P}_t$ can be transformed into the Cartesian space of the robot base frame, thus, producing $\mathbf{P}_o$, each point of which is transformed to become $\mathbf{p}_o^i = \{T(\mathbf{x}, \theta_{o,t-1}^c), \mathbf{c}\}$. Here, $T(\mathbf{x}, \theta)$ indicates a homogeneous transformation of the point $\mathbf{x} \in \mathbb{R}^3$, given the 6-dimensional parameters θ. Because the parameters $\theta_{o,t}^c$ can be continuously updated in the online estimation process, which will be explained in the visual calibration step in the next section, the assumption holds only with a proper selection of the initial parameters θ_{0,t_0}^c. Note that to improve readability in the remainder of this paper, notations without any specified time variable indicate that the value is at the current time t.

Self body observation is the process of extracting the part of the point cloud that most likely contains the robot body itself at each time t, the so called *observed body* $\mathbf{Z}_o$. This set can be computed by comparing a distance of each point in $\mathbf{P}_o$ to the expected body $\widehat{\mathcal{M}}_o$, because both are represented in the robot base frame. Here, we define the point cloud of the observed body to be a set of points within a certain distances from the mesh model:

$$\mathbf{Z}_o = \{\mathbf{p}_i | d(\mathbf{p}_i, \widehat{\mathcal{M}}_t) \leq \delta, \mathbf{p}_i \in \mathbf{P}_o\}, \quad (1)$$

$$d(\mathbf{p}_i, \widehat{\mathcal{M}}_o) = \min_{\mathbf{v}_j \in \widehat{\mathcal{M}}_o} ||\mathbf{p}_i - \mathbf{v}_j||. \quad (2)$$

The distance of a point to the mesh model is calculated by comparison of vertices, instead of faces, for computational simplicity.

C. Virtual Coordinate in the Focus Frame

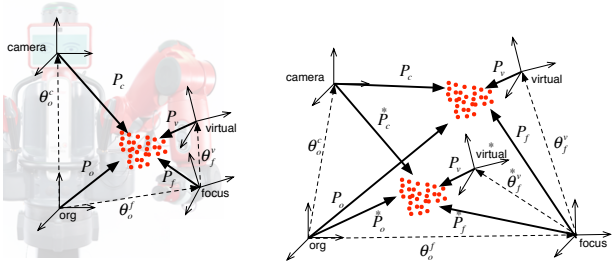
With inperfect models, measurement errors in the joint angles in $\mathcal{R}_t$, and camera intrinsic parameter estimation errors, the visual information and robot models cannot be perfectly matched. As shown in Fig. 4, the visual measurements and the robot model produce a discrepancy, which has different errors in different parts of the workspace. Due to intrinsic camera calibration errors, it is frequently not possible to achieve a perfect alignment everywhere. To compensate for this, we introduce here the concept of a focus frame where the errors should be minimized so as to improve the manipulability near the frame origin. In most cases, the frame for manipulation is far from the camera frame, which makes it hard to coordinate the visual measurements close to the focus frame by calibrating the camera pose. We introduce a virtual coordinate moving in the focus frame to calibrate the visual data directly from the frame. Fig. 5 shows an example of the concept. Using the kinematic chains of four frames, the visual data will be calibrated according to the estimation of the virtual frame with the following relations:

$$\mathbf{P}_o = T(\mathbf{P}_c, \theta_o^c), \quad (3)$$

$$\mathbf{P}_f = T(\mathbf{P}_o, \theta_f^o), \quad (4)$$

$$\mathbf{P}_v = T(\mathbf{P}_f, \theta_v^f). \quad (5)$$

Fig. 5b shows the case of virtual coordinate updates



(a) Visual coordinate (b) Moving virtual coordinate

Fig. 5: Virtual coordinate on the focus frame.

from θ_f^v to $\overset{*}{\theta}_f^v$, relative to the focus frame. After estimating the updated virtual coordinate, where the captured visual measurements are now attached, the calibrated visual data can be obtained based on the base frame and the camera frame as follows:

$$\overset{*}{\mathbf{P}}_f = T(\mathbf{P}_v, \overset{*}{\theta}_f^v), \tag{6}$$

$$\overset{*}{\mathbf{P}}_o = T(\overset{*}{\mathbf{P}}_f, \theta_o^f), \tag{7}$$

$$\overset{*}{\mathbf{P}}_c = T(\overset{*}{\mathbf{P}}_o, \theta_c^o). \tag{8}$$

Initially, the virtual frame is located at the same pose as the focus frame. As the estimation proceeds at each time step, θ_f^v is updated with the newly observed body points $\mathbf{Z}_v$ to find the best match of $\mathbf{Z}_f$ to the model $\widehat{\mathcal{M}}_o$, which results in $\overset{*}{\theta}_f^v$ and $\overset{*}{\mathbf{P}}_f$. Note that while the robot moves, the movement of the virtual frame would be feasibly small to update since it is attached to the moving focus frame.

The body model $\widehat{\mathcal{M}}_o$ should also be transformed to the focus frame to be fairly comparable with the visual measurements. In order to compute the comparison efficiently, we applied an occupancy grid method to represent the presence of the surface of the robot body in the 3D space, similar to [12]. This is an efficient method to compare multiple hypotheses of θ_f^v with the constructed grid model at once, without expensive correspondence searches for each hypothesis. The evenly spaced field of the grid, which can be accessed easily by its ordered index, contains position information by grid index itself, and surface normal information where the grid is involved.

Fig. 6 shows an example of the occupancy grid from triangle mesh faces. First, a 3D grid space is constructed from the mesh model transformed to the focus frame:

$$\mathbf{G}_f \in \mathbb{R}^{3 s_x s_y s_z}. \tag{9}$$

The size of the space is defined by s_x, s_y, s_z, each of which is the length of the model divided by the given grid size, λ_g. Each entity of the grid, $\mathbf{g}(x, y, z) \in \mathbb{R}^3$, which is initialized with the zero vector, is filled by a three-dimensional surface normal vector if it is involved in one of the triangle faces in the transformed model $\widehat{\mathcal{M}}_f = T(\widehat{\mathcal{M}}_o, \theta_f^o)$. First, for each face $\mathbf{f}_i$, a set of indices of grids in the 3D area $X(\mathbf{f}_i), Y(\mathbf{f}_i), Z(\mathbf{f}_i)$ that completely covers the 3D triangle face are searched with the position values of the three vertices in the face. Each grid cell is assigned with the normal vector of the closest vertex:

$$\mathbf{g}(x, y, z | x \in X(\mathbf{f}_i), y \in Y(\mathbf{f}_i), z \in Z(\mathbf{f}_i)) = \mathbf{n}(\mathbf{v}_{cls}), \tag{10}$$

$$\mathbf{v}_{cls} = \underset{\mathbf{v}_i \in \mathbf{f}_i}{\arg\min} \, ||\{x(\mathbf{v}_i), y(\mathbf{v}_i), z(\mathbf{v}_i)\}^\top - \{x, y, z\}^\top||. \tag{11}$$

D. Particle Filtering for Visual Calibration

With the expected body $\widehat{\mathcal{M}}_o$ as motor-related, and the observed body $\mathbf{Z}_t$ as visual-related data, the visual calibration process estimates the virtual frame $\theta = \{r, p, y, x, y, z\}$ at each time step so that the coordinate frame of the visual

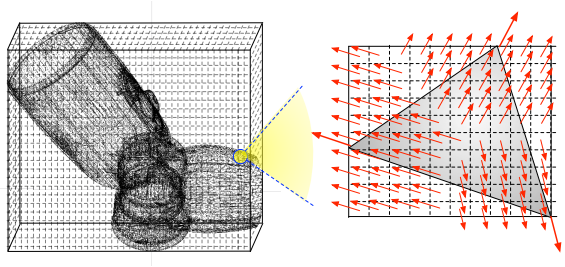


Fig. 6: Example occupancy grid from triangle mesh faces.

data matches that of the robot. Note that for the remainder of the paper, we use only θ to denote the virtual frame θ_f^v. In order to estimate the six-dimensional parameters in real-time, we used a GPU-based particle-filtering approach with a likelihood function of $p(\mathbf{G}_f | \theta, \mathbf{Z}_f)$ as in [12], [13].

Following a particle filtering framework, a set of weighted particles $\mathcal{S}_t = \{\theta_t^{(n)}, \pi_t^{(n)}\}_{n=1}^N$ represent the posterior probability density function of the virtual frame parameters, $p(\theta_t | \mathbf{Z}_{1:t}, \widehat{\mathcal{M}}_{1:t})$. After the weighted particles have been obtained, the virtual frame pose can be estimated by summing all particles with their weights:

$$\overset{*}{\theta}_t = \sum_{n=1}^{N} \pi_t^{(n)} \theta_t^{(n)}. \tag{12}$$

At each time step, the particles $\{\theta_t^{(n)}\}_{n=1}^N$ are generated by following the Sampling Importance Resampling (SIS) method [14]. The most important part is to define a proper weight evaluation function that is proportional to the likelihood of the expected body, given the observed body and the particle $\theta_t^{(n)}$,

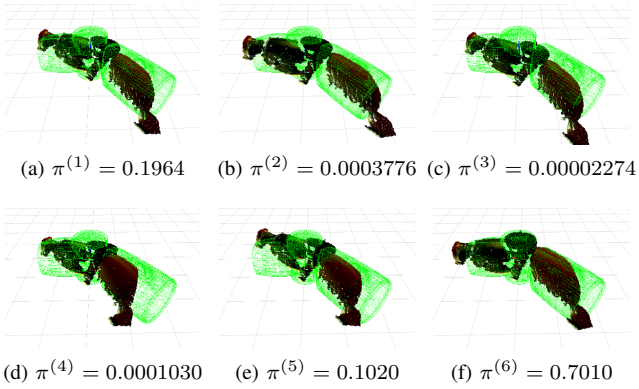
$$\pi_t^{(n)} \propto p(\mathbf{G}_f | \theta, \mathbf{Z}_f), \tag{13}$$

satisfying the normalization condition, $\sum_{i=1}^N \pi_t^{(n)} = 1$. The evaluation function means that a hypothesis of the virtual frame pose that transforms the observed body close to the expected body scores a high value of probability to generate the estimated parameter $\overset{*}{\theta}_t$. Thus, the likelihood function should be defined according to the closeness of $\mathbf{Z}_f$ and $\mathbf{G}_f$, given the camera pose hypothesis $\theta_t^{(n)}$.

The likelihood of a set of points $\mathbf{Z}_f$ to the mesh model $\mathbf{G}_f$ can be expressed as the product of a set of likelihoods of points $\mathbf{p}_i$ to the corresponding grid $\mathbf{g}_i$, which can be found by the grid index matched to the point $\mathbf{p}_i$:

$$p(\mathbf{G}_f | \theta_t^{(n)}, \mathbf{Z}_f) = \prod_i p(\mathbf{g}_i | \theta_t^{(n)}, \mathbf{p}_i). \tag{14}$$

The likelihood evaluation for each point in (14) is defined as the distance from a point to a grid cell. Here, it can be simply calculated as an average distance of all vertices in



(a) $\pi^{(1)} = 0.1964$ (b) $\pi^{(2)} = 0.0003776$ (c) $\pi^{(3)} = 0.00002274$

(d) $\pi^{(4)} = 0.0001030$ (e) $\pi^{(5)} = 0.1020$ (f) $\pi^{(6)} = 0.7010$

Fig. 7: Examples of weight evaluations given six virtual coordinate hypotheses. They were generated with a translational variance of 0.01m and a rotational variance of 5 degrees. The normalized weight values are proportional to the likelihood of the transformed scene to the mesh model (Green).

the grid. There are two ways to define the distance:

$$d_{point-to-point}(\mathbf{p}, \mathbf{g}) = \frac{1}{n} \sum_{\mathbf{v}_k \in \mathbf{g}} ||\mathbf{p} - \mathbf{v}_k||^2, \quad (15)$$

$$d_{point-to-plane}(\mathbf{p}, \mathbf{g}) = \frac{1}{n} \sum_{\mathbf{v}_k \in \mathbf{g}} ((\mathbf{p} - \mathbf{v}_k) \cdot \mathbf{n}(\mathbf{v}_k))^2, \quad (16)$$

where $\mathbf{n}(\mathbf{v})$ is the normal vector of a vertex $\mathbf{v}$, which can be easily obtained by the mesh model. We used both distances and compared them in the experiments. The likelihood of each point and the corresponding face is defined as

$$p(\mathbf{g}_i | \theta_t^{(n)}, \mathbf{p}_i) = \exp^{-d(T(\mathbf{p}_i, \theta_t^{(n)}), \mathbf{g}_i)}. \quad (17)$$

Fig. 7 shows a series of examples of the evaluated weights. The different camera pose hypotheses are evaluated according to the similarity between the mesh and the point cloud data. Weight values are proportional to the closeness of two point clouds.

E. Visual Calibration

After the virtual coordinate $\overset{*}{\theta}{}_f^v$ is estimated by (12), the original scene measured from camera $\mathbf{P}_c$ is transformed by (3)-(8), producing the calibrated scene relative to the origin.

$$\overset{*}{\mathbf{P}}_o = T(\mathbf{P}_c, \theta_o^f \cdot \overset{*}{\theta}{}_f^v \cdot \theta_{v,t-1}^f \cdot \theta_{f,t-1}^o \cdot \theta_{o,t-1}^c), \quad (18)$$

where $T(\bullet, \theta_1 \cdot \theta_2) \coloneqq T(T(\bullet, \theta_2), \theta_1)$.

The online visual-motor coordination algorithm running at each time step t was implemented with a Graphics Processing Unit (GPU). Algorithm 1 gives details of the implementation and consideration points. First, given the initialized parameters, $\lambda = \{\delta, \lambda_g, (\lambda_{varp}, \lambda_{varr}), \lambda_p\}$, and the focus frame, the visual-motor coordination algorithm executes each time a new point cloud measurement $\mathbf{P}_c$ is received, and produces the calibrated scenes $\overset{*}{\mathbf{P}}_o$, and the updated virtual coordinate $\overset{*}{\theta}{}_f^v$.

Algorithm 1 Online Visual-Motor Coordination

Initialization $\theta_o^c, \mathcal{S}_0, focus$
procedure VISMOT($\theta_o^c, \theta_{f,t-1}^v, \mathcal{S}_{t-1}, \mathbf{P}_c, \mathcal{R}_t, \lambda$)
  function BODYEXPECT($\mathcal{R}_t$)
    $\mathbf{M}_t^i \leftarrow T(\mathbf{M}_i, \theta_o^i), \forall i$
    return $\widehat{\mathcal{M}}_t = \{\mathbf{M}_t^i\}_{i=1}^n$
  function BODYOBSERVE($\theta^c, \mathbf{P}_c, \widehat{\mathcal{M}}_t, \delta$)
    $\mathbf{P}_o \leftarrow T(\mathbf{P}_c, \theta_o^c)$
    execute (1) and (2)
    return $\mathbf{Z}_t$
  $\mathbf{Z}_f \leftarrow T(\mathbf{Z}_o, \theta_{v,t-1}^f \cdot \theta_f^o)$ ▷ visual focus
  $\mathbf{G}_f \leftarrow grid(\widehat{\mathcal{M}}_t, \theta_{v,t-1}^f \cdot \theta_f^o, \lambda_g)$ ▷ body focus
  function VISCALB($\mathbf{P}_c, \mathcal{S}_{t-1}, \mathbf{Z}_f, \mathbf{G}_f, \lambda$)
    $\theta_t^{(n)} \sim SIS(\mathcal{S}_{t-1}, \lambda_{varp}, \lambda_{varr}), \forall n$
    $\pi_t^{(n)} \leftarrow weight(\theta_t^{(n)}, \mathbf{Z}_f, \mathbf{G}_f)$ ▷ GPU
    $\overset{*}{\theta}{}_f^v$ from $\mathcal{S}_t = \{\theta_t^{(n)}, \pi_t^{(n)}\}_{n=1}^{\lambda_p}$ in (12)
    $\overset{*}{\mathbf{P}}_o \leftarrow transform(\mathbf{P}_c)$ in (18)
  return $\overset{*}{\mathbf{P}}_o, \overset{*}{\theta}{}_f^v, \mathcal{S}_t$

IV. Experiments and Results

In order to verify the performance of the proposed calibration method, the accuracy, robustness, and computation time were evaluated through a *recalibration task* with three different robot configurations. For reproducibility, we make our software available open-source[1].

A. Recalibration Task by Changing Visual Focus to Another Arm

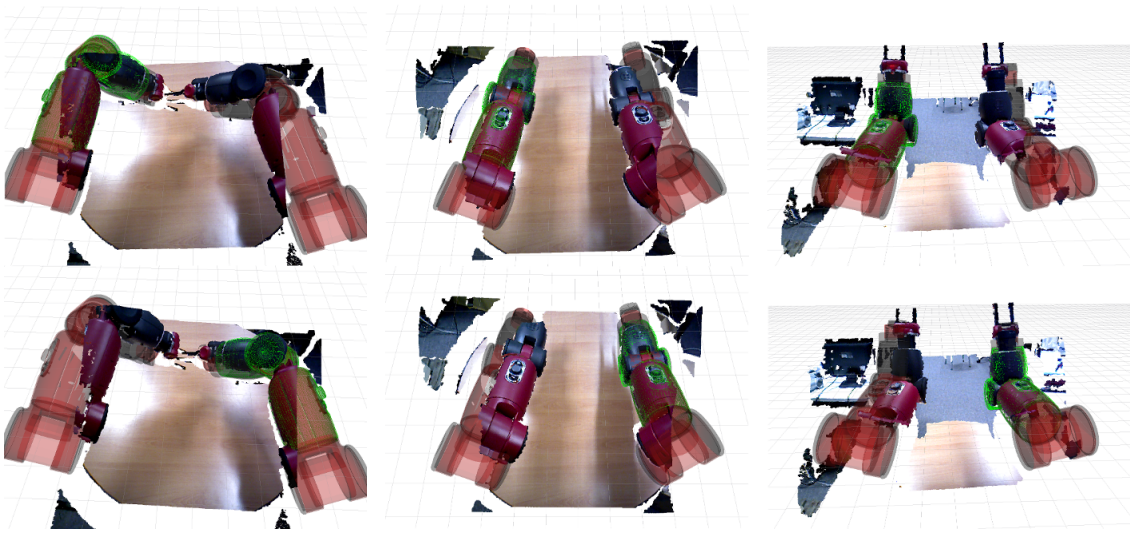
When a dual-arm robot changes from a manipulation task with one hand to the next task with another hand, visual measurements should be recalibrated with focus on the new target hand, due to the errors from camera and robot kinematic models. While in related work the recalibration task was performed with specially designed markers on both robot wrists [7], our method can be applied to calibrate any robot arms only using their 3D mesh data, anytime, even if the wrists are not visible.

To show the validity of the method, the recalibration task was designed to change the robot's visual focus from its well-calibrated left hand to its right hand. Fig. 8 shows the calibrated visual data, $\overset{*}{\mathbf{P}}_o$, in three test configurations. As shown in the results of the recalibration task, the calibration is sufficiently adjusted to reduce the errors between the 3D model and the visual measurements. Quantitative results are analyzed in the following two sections.

B. Performance Analysis

Online calibration requires high performance in terms of both accuracy and computation time to support real-time manipulation tasks. Here, we conducted comparative

[1]The software, which is available from https://github.com/AIS-Bonn/vismotcoord, was implemented based on C++ using ROS, PCL, and CUDA libraries.



(a) *Cross* task (b) *Stretch* task (c) *Upward* task

Fig. 8: Three recalibration tasks and calibration results by Online$^{\text{p}}$. The three top figures show the initial state of the tasks, where the visual measurements are well calibrated on the *left lower forearm* as the focus frame. The three bottom figures depict calibrated visual data focused on the *right lower forearm*.

assessments of the proposed method to another markerless calibration method that works by robust Iterative Closest Point (ICP) registration [15].

The metric to measure the errors between the calibrated visual measurements and the motor coordinates is defined as averaged *point-to-point* distance between the points in the calibrated scene $\mathbf{p}_i \in \overset{*}{\mathbf{P}}_o$ and the vertices of the focused expected body $\mathbf{v}_i \in \mathbf{M}_o^f$ at each time step:

$$E = \frac{1}{N} \sum_{k=1}^{N} ||\mathbf{p}_k - \mathbf{v}_k||, \tag{19}$$

where each $(\mathbf{p}_k, \mathbf{v}_k)$ is a k-th pair of the selected N correspondences by using closest points correspondence estimation, rejecting pairs with duplicate target matches (`CorrespondenceRejectorOneToOne`), implemented in [15].

We tested our two online methods with or without face normals (Online$^{\text{p}}$, Online$^{\text{n}}$), compared with two offline methods with or without face normals (Offline$^{\text{p}}$, Offline$^{\text{n}}$). The two online methods were implemented as described in Algorithm 1 with parameters (δ = 0.05 m, λ_g = 0.005 m, λ_p = 1024, λ_{varp} = 0.005 m, λ_{varr} = 1.8°) which were determined by the sensitivity analysis described in the next section. Online$^{\text{p}}$ takes only the first part of (17) for computational efficiency, while Online$^{\text{n}}$ takes the full (17) to test its robustness. The two offline methods were realized by *point-to-point* ICP and *point-to-plane* ICP [15] with the same rejection method of the error metric (19).

For fair comparison, all methods used the same termination criteria, a relative mean square error (MSE) of 0.0005^2, and computation time was measured through all pipelines, from obtaining raw sensor images to finding the converged calibrated point cloud scene. We used a PrimeSense VGA (640×480) RGB-D camera running at 30 Hz and an Intel 2.5 GHz i7-4710HQ CPU with a NVidia GeForce GTX 960M GPU.

TABLE I: Calibration Performance

Metric	Method	Cross	Stretch	Upward
Error [m]	Online$^{\text{p}}$	0.003915	0.004258	**0.002792**
	Online$^{\text{n}}$	0.003892	0.004042	0.002823
	Offline$^{\text{p}}$	0.003195	0.004108	0.003088
	Offline$^{\text{n}}$	**0.003090**	**0.003826**	0.003134
Computation time [s]	Online$^{\text{p}}$	2.19972	**0.76239**	1.35464
	Online$^{\text{n}}$	**2.17244**	0.80218	**1.14884**
	Offline$^{\text{p}}$	6.95031	5.55796	2.08794
	Offline$^{\text{n}}$	8.14364	6.03880	3.84839

Table I shows the converged *point-to-point* error and computation time of the four methods in the three tasks. The error differences between the online and offline methods are not significant. However, in terms of computation time, the two online methods are always faster than the offline methods. Especially in the *stretch* case, the online methods converged in the minimum time while offline methods took longer than in the *upward* case. This effect is due to the size of the point cloud measurements. The speed of the offline methods is determined by the input size, whereas the proposed online methods with the help of GPU are able to process multiple points in parallel. For online methods, more points actually improve the convergence time, because additional evidences increase the chance of finding high-likelihood particles.

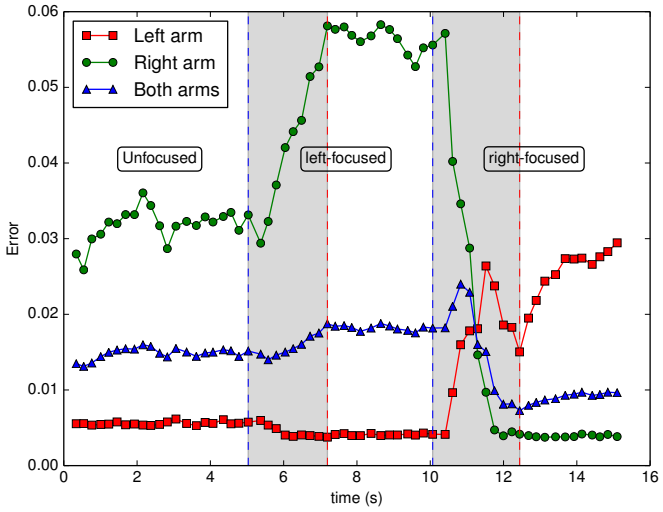


Fig. 9: Three calibration errors of two arms in the *cross* posture. Two blue-dashed vertical lines indicate the moments the focus phase changed. The online calibrations with new focus frames were converged at the time on the red-dashed vertical lines.

In the offline methods, using face normals reduces errors but sacrifices computation time. In the online methods, however, it was hard to find the differences between the two. Because the grid size λ_g restricts the maximum distance of pairs, there is not much difference between *point-to-point* and *point-to-plane* distances between two close points.

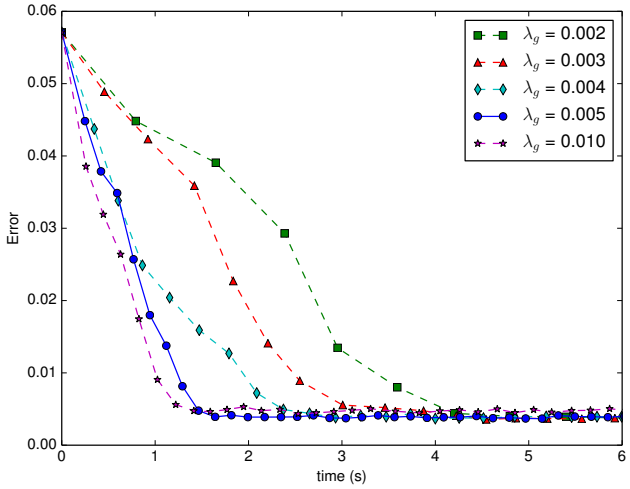
C. Calibration Errors of the Focused and Unfocused Frames

Fig. 9 shows how calibration errors of the two arms fluctuates with changes in the focus frame. Initially, the camera was globally calibrated (unfocused) with an external marker. After five seconds, the focus frame was assigned to the left arm (left-focused) and then to the right arm (right-focused) for five seconds each. For the three phases, three calibration errors (two errors with the 3D model of each arm, and one error with those of both arms) were continuously measured by (19).

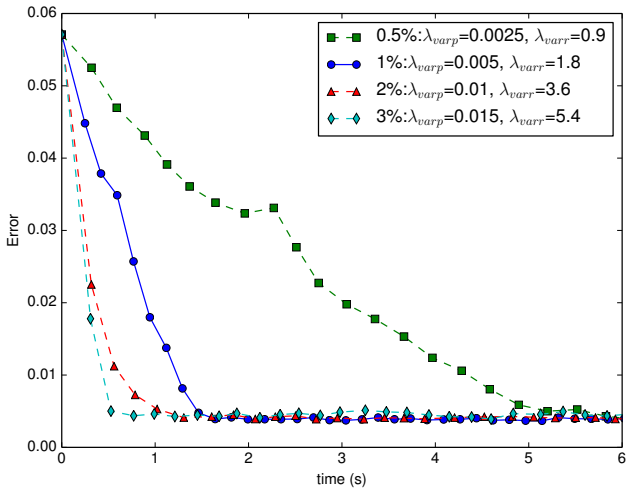
When the online calibration started in the left-focused phase, the measurement errors of the left arm decreased and converged to the minimum observed calibration error while those of the right arm increased. On the other hand, when the focus frame is changed in the right-focused phase, the two errors became opposite. Even though the error of both arms, which is the averaged error sum of two arms, is possible to increase in the focused phase (left-focused case in Fig. 9), the visual-motor discrepancy near to the focus frame is always smaller than any errors in the unfocused phase. This helps to perform more accurate manipulation with the focused arm.

D. Sensitivity Analysis

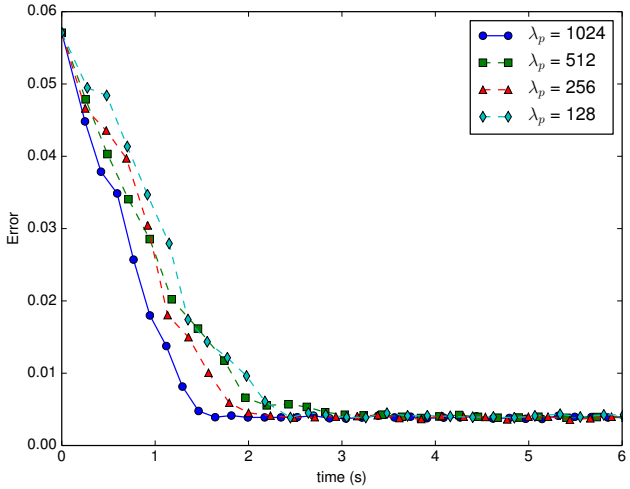
The performance of the proposed method is subject to the priors, $\lambda = \{\delta, \lambda_g, (\lambda_{varp}, \lambda_{varr}), \lambda_p\}$. We found that δ was not critical and used 0.005 m for all other experiments. To investigate the sensitivity of the performance to the other parameters, the two metrics were measured exhaustively by



(a) Varying grid size λ_g. (λ_{varp}=0.005, λ_{varr}=1.8, λ_p=1024)



(b) Varying resampling variance λ_{varp}, λ_{varr}. (λ_g = 0.005, λ_p=1024)



(c) Varying particle number λ_p. (λ_g=0.005, λ_{varp}=0.005, λ_{varr}=1.8)

Fig. 10: Systematic exploration of the three parameters in the *cross* task to find optimal performance of convergence error and computation time.

all possible sets of the parameters in the *cross* task. With the goal of finding empirically optimal parameters, three experiments were performed to find each optimal parameter. Here, each pair of (λ_{varp}, λ_{varr}) was determined by a certain percentage to the maximum permitted variance of transition and rotation, which were set to 0.5 m and 180°. Each experiment also takes the optimal value found from other experiments.

As shown in Fig. 10a, the grid size is an important factor to control the trade-off between the accuracy and computation time. If the grid size is relatively big, such as 0.010 m, the error cannot converge to the minimum value (0.003915 in Table I). The second experiment in Fig. 10b illustrates the trade-off relation of the variance of particle samples. Similar to the first experiment, where the variances are large, the error cannot reach the minimum value. The third experimental result in Fig. 10c indicates that increasing the number of particles improves both accuracy and speed of convergence. These results support our optimal parameter choice, which is identical for the three tasks, which produced the minimum error and convergence time in Table I. With the selected parameters, the online algorithm runs an average at 5 Hz.

V. Conclusion

Camera calibration is necessary to operate robot manipulators. Instead of struggling with estimation of many unknown and possibly changing parameters, we proposed a fast and easy-to-use auto-calibration method by making use of the 3D robot model. The online and focusing features of the algorithm are expected to facilitate various dynamic manipulation tasks without the burden of camera calibration. It is a matter of future work to apply this capability to a dual-arm robot manipulator in such a way that it enables dexterous object manipulation tasks with both hands.

ACKNOWLEDGMENT

This work was supported by the German Research Foundation (DFG) under the grant "ALROMA - Autonomous active object learning through robot manipulation" in the priority programme SPP 1527 Autonomous Learning.

References

[1] A. Shumway-Cook and M. H. Woollacott, *Motor control: translating research into clinical practice*. Lippincott Williams & Wilkins, 2007.

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

[3] G.-Q. Wei and S. De Ma, "Implicit and explicit camera calibration: Theory and experiments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 5, pp. 469–480, 1994.

[4] A. Elatta, L. P. Gen, F. L. Zhi, Y. Daoyuan, and L. Fei, "An overview of robot calibration," *Information Technology Journal*, vol. 3, no. 1, pp. 74–78, 2004.

[5] T. Asfour, K. Welke, P. Azad, A. Ude, and R. Dillmann, "The karlsruhe humanoid head," in *8th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. IEEE, 2008, pp. 447–453.

[6] U. Hubert, J. Stückler, and S. Behnke, "Bayesian calibration of the hand-eye kinematics of an anthropomorphic robot," in *12th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. IEEE, 2012, pp. 618–624.

[7] O. Birbach, U. Frese, and B. Bäuml, "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, 2015.

[8] D. Maier, S. Wrobel, and M. Bennewitz, "Whole-body self-calibration via graph-optimization and automatic configuration selection," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 5662–5668.

[9] I.-W. Park, B.-J. Lee, S.-H. Cho, Y.-D. Hong, and J.-H. Kim, "Laser-based kinematic calibration of robot manipulator using differential kinematics," *IEEE/ASME Transactions on Mechatronics*, vol. 17, no. 6, pp. 1059–1067, 2012.

[10] N. Moutinho, M. Brandao, R. Ferreira, J. A. Gaspar, A. Bernardino, A. Takanishi, and J. Santos-Victor, "Online calibration of a humanoid robot head from relative encoders, imu readings and visual data." in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Citeseer, 2012, pp. 2070–2075.

[11] P. Vicente, R. Ferreira, L. Jamone, and A. Bernardino, "Gpu-enabled particle based optimization for robotic-hand pose estimation and self-calibration," in *2015 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE, 2015, pp. 3–8.

[12] S. Li, S. Koo, and D. Lee, "Real-time and model-free object tracking using particle filter with joint color-spatial descriptor," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 1084–1091.

[13] C. Choi and H. I. Christensen, "Rgb-d object tracking: A particle filter approach on gpu," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2013, pp. 1084–1091.

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

[15] D. Holz, A.-E. Ichim, F. Tombari, R. B. Rusu, and S. Behnke, "A modular framework for aligning 3D point clouds - registration with the point cloud library," *Robotics & Automation Magazine, IEEE*, vol. 22, no. 4, pp. 110–124, 2015.