2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Uncertainty Estimation of AR-Marker Poses for Graph-SLAM Optimization in 3D Object Model Generation with RGBD Data

Razvan-George Mihalyi, Kaustubh Pathak, Narunas Vaskevicius, Andreas Birk

Abstract— **This paper presents an approach to acquire textured 3D models of objects without the need for sophisticated hardware infrastructures. The approach is inexpensive, using a low-cost Microsoft Kinect RGB-D sensor and Augmented Reality (AR) markers printed on paper sheets. The AR-markers can be freely placed in the scene, allowing the modeling of objects of various sizes, and the sensor can be moved by the hand of an untrained person. To generate usable models with this very inexpensive and simple set-up, the sequence of RGB-D scans is embedded in a graph-based optimizer for automatic post-refinement. The main novelty of this contribution is the development of an uncertainty model for an AR-marker. The AR-marker uncertainty models are used as constraints in an optimization problem to better estimate the object pose. The models are in the end further fine-tuned by a standard point-based registration algorithm. The results section presents realistic models of various objects generated using this system, e.g., parcels, sport balls, human dolls etc. Additionally, a quantitative analysis is presented using objects of known dimensions.**

I. INTRODUCTION

There is a significant recent interest in textured 3D models of objects in the robotics community. Textured 3D information – also known as RGB-D for color-depth – can be useful in a wide range of advanced robotics applications that deal with object recognition, semantic mapping, or manipulation tasks, to name a few examples. There are three main approaches for generating textured 3D object models: 1) hand-crafted virtual models, 2) high-end models generated with sophisticated hardware infrastructure, and 3) coarse models generated with low-cost, infrastructure-less methods.

Examples of databases of virtual models of textured 3D objects are the Princeton Shape Benchmark [1] and the Google warehouse [2]. The Princeton Shape Benchmark consists of polygonal models that were collected from the World Wide Web. The common trait of the models in the database is that they have been handcrafted. The same holds for the Google warehouse where the models are based on different user contributions. Generally speaking, virtual models have the advantage of being easily generated with appropriate design tools. However, as they are not constructed from real objects, they consist of simplified shapes and textures and are influenced by the modeling skills of the contributing users.

Option two, in form of high-end models generated with sophisticated hardware infrastructures, is at the opposite end of the scale. These textured 3D models are highly realistic but their generation requires complex hardware set-ups. An example for this kind of approach is the KIT Object Model Database [3]. The high-end models contained therewith were generated using a sophisticated infrastructure consisting of a 3D digitizer, a turn-table and a pair of 2D cameras mounted on a sled moving along a curved rail [3]. Commercial solutions for high-end 3D object model acquisition are also available and are typically targeted at reverse engineering or quality assurance [4]. While a database like [3] is very useful for the community, as basis for a benchmark, it is very costly to procure a similar infrastructure to generate own models of objects that do not already exist in such databases.

The authors are with the Dept. of EECS, Jacobs University Bremen, 28759 Bremen, Germany. {r.mihalyi, k.pathak, n.vaskevicius, a.birk} @jacobs-university.de

The research leading to the results presented here has received funding from the European Community's Seventh Framework Programme (EU FP7 ICT-2) within the project "Cognitive Robot for Automation Logistics Processes (RobLog)".

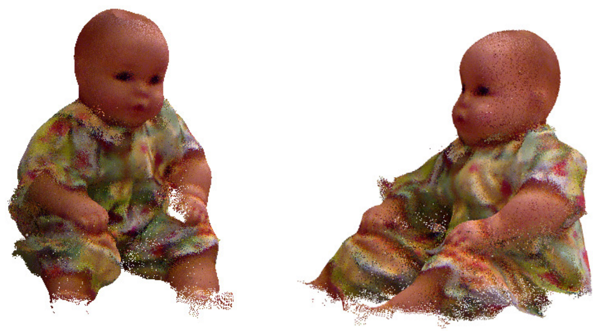


Fig. 1. Baby doll model. Fig. (a), (b) show different views of the model.

The third option is somewhere in the middle between the first two approaches, namely to obtain realistic models with little or no infrastructure needs. The work presented here falls into this category. A core motivation is the availability of low-cost RGB-D sensors, e.g., Microsoft Kinect, Asus Xtion Pro, which allow for object model generation without large additional investments. The University of Washington RGB-D Object Dataset [5] is an example along those lines, as it is based on a Kinect-like sensor. But this dataset is generated using some infrastructure, namely a turn-table and a fixed frame for mounting the sensor at specific angles. In contrast, the object modeler [6] from the RoboEarth project [7] only uses a Kinect sensor and simple Augmented Reality (AR) markers that can be printed on paper sheets. Concretely, the RoboEarth modeler employs the ARToolkit [8], [9] library for a coarse extraction of marker poses, which is followed by a least-squares fit of points on two lines passing through the center of the markers. The 6 AR-markers in the RoboEarth set-up are printed in a fixed orthogonal pattern on two A4-sheets that are glued together. The known fixed pattern among the six AR-markers is used for a plausibility check based on known distances between the markers. A Gram-Schmidt process determines a coordinate system that is centered in the fixed AR pattern to assemble the RGB-D scans into a model. While the approach proposed in this paper also uses a Microsoft Kinect RGB-D sensor and AR-markers, it differs from [6] in two aspects. Firstly, in the approach presented here the AR-markers can be placed freely in scene, hence allowing to model objects of a wide range of sizes (volumes ranging from $10^{-4}m^3$ to $3.5m^3$). Secondly, a non-trivial post-processing of the registered RGB-D scans is used to clearly improve the model quality, as shown in the results section.

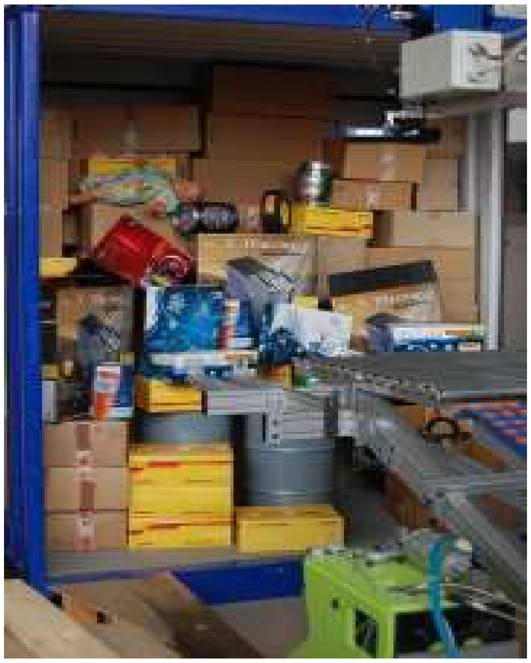


Fig. 2. Close-up of an example container content that is to be unloaded in the context of the RobLog project.

Part of the method presented here relies on an uncertainty model for the AR-markers poses (cf. Sec. II-B). AR-markers have been used in the context of pose estimation in robotics before, e.g., in [10], [11], [12], [13]. The authors are not aware of a proper uncertainty treatment of AR-markers poses, which includes both translational and rotational aspects. The discussion in Sec. II-B is therefore of interest for general AR-marker applications beyond the application of object modeling presented here.

The authors' interest in object modeling stems from the EU-project "Cognitive Robot for Automation of Logistic Processes (RobLog)". RobLog (`www.roblog.eu`) is an Integrated Project (IP) funded by the European Commission in the area "Cognitive Systems and Robotics" under grant agreement 270350. The objective of this project is to automate logistic processes, such as the unloading of containers. Examples of container content are shown in Fig. 2 and the current hardware set-up is shown in Fig. 3. In this context it is of interest to generate an object model on the spot with a low-cost device like the Kinect sensor and without any complex infrastructure. Therefore, a simple modeling set-up using only AR-markers, as in the RoboEarth modeler, is desirable. An important feature in the context of the RobLog project is the ability to model a wide range of object sizes. Consequently, the AR-markers cannot be arranged in a fixed geometric pattern as in the RoboEarth modeler [6]. Instead, the markers can be literally thrown into the scene to be placed at arbitrary positions. The RGB-D sensor can then be moved, with a sufficiently low speed so as not to induce motion blur, around the object by the hand of an untrained person.

The initial step is the estimation of the object pose in each frame of the RGB-D data stream, based on the AR-markers distributed in the scene (Sec. II-B). Once the object pose is approximately known in one scan, the object is segmented from the scene, the resulting point cloud constituting one of the object views. The core idea of this approach is to accumulate point clouds from all object views into a consistent frame. This is done via a graph-based optimization of both the sensor poses, which capture the object views, and the AR-markers in the scene (Sec. II-C, Sec. II-D). Furthermore, each point cloud view, corresponding to an optimized sensor pose, is aligned with the model point cloud, using the standard choice for the registration of 3D point clouds, namely the Iterative Closest Point (ICP) algorithm [14]. Additionally, the models are further fine-tuned by using a voxelgrid filter to remove outlying points. The complete workflow is detailed in Sec. II.

Using this approach, one can generate 3D models of various objects, like barrels, boxes, sports balls, etc. These are presented in Sec. III. The ground-truth models, such as sport balls and cuboidal boxes, are assessed quantitatively by comparing the radii or, respectively, the volumes of the model and the physical object. The errors thus obtained are in the order of 3% for both spherical and cuboidal objects. Other models, such as human baby dolls, car tires, etc., as seen in Fig. 1 and Fig. 7, are visually inspected and deemed satisfactory when not exhibiting registration errors.

Notation

The notation used in this paper is summarized below. In general, scalars are in normal lowercase letters, vectors in bold small letters, and matrices in bold capitals. For quantities resolved in different frames, the left superscript/subscript notation of [15] is used. Right subscripts are used for indexing or for denoting vector components.

Using this notation, a position vector of a spatial point resolved in the reference frame $\mathcal{F}_i$ is denoted by ${}^{i}\mathbf{p} \in \mathbb{R}^3$ and its homogeneous coordinates representation by ${}^{i}\underline{\mathbf{p}}$. Two position vectors of the same physical point observed from two different frames $\mathcal{F}_i$, $\mathcal{F}_j$ with their respective origins at $\mathcal{O}_i$, $\mathcal{O}_j$, are related by:

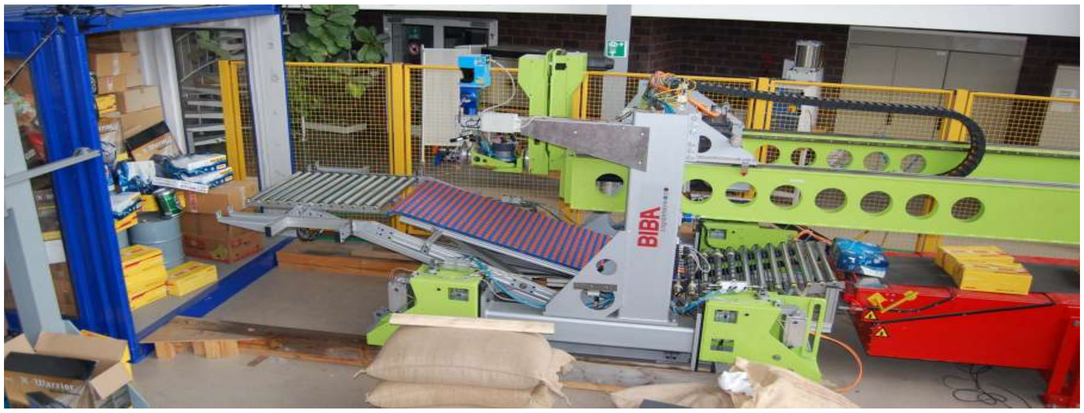


Fig. 3. The set-up of the RobLog scenario for container unloading.

$$ {}^{i}\underline{\mathbf{p}} = {}^{i}_{j}\mathbf{T}\;{}^{j}\underline{\mathbf{p}}, \text{ where,} \tag{1a} $$

$$ {}^{i}_{j}\mathbf{T} \triangleq \begin{bmatrix} {}^{i}_{j}\mathbf{R} & {}^{i}_{j}\mathbf{t} \\ \mathbf{0}_3^{\mathsf{T}} & 1 \end{bmatrix}, \qquad {}^{i}_{j}\mathbf{t} \triangleq \overrightarrow{\mathcal{O}_i\mathcal{O}_j} \text{ resolved in } \mathcal{F}_i. \tag{1b} $$

II. METHOD

A. Overview

Object modeling using just point clouds has been widely studied in the past two decades. An early attempt, described in [16], uses range images and a variant of ICP to construct a 3D model. In contrast to ICP, an iterative algorithm that minimizes the error between pairs of points [14], the approach from [16] consists of an iterative algorithm that minimizes point-to-plane distances. The most advanced system for range-only based registration in terms of performance is the KinectFusion system [17]. While targeted at the mapping and reconstruction of "complex room-sized scenes", this system could, in principle, also be used for modeling of individual objects. The main concepts behind KinectFusion are the continuous update of the surface representations by fusing already registered views and the accurate tracking of the camera pose by depth disparity measurements. KinectFusion is designed as a real-time system, i.e., the algorithms for modeling and tracking are optimized for GPU computation.

Recent work on modeling 3D surfaces also uses a graph-based optimization [18], where the standard point based registration is supplemented by an additional heuristic, namely that the points are sampled from a regular surface. The method in [18] is a variant of the bundle adjustment problem [19] where the surface points in the scans and the poses of the sensor are globally and jointly optimized.

The approach proposed here differs from the above methods in that it does not rely solely on registration algorithms to align the multiple views of the object to obtain the final model. Instead, the modeling problem is formulated as a graph-based optimization problem, where the error of the pose estimates of the RGB-D sensor and the AR-marker landmarks is minimized in a nonlinear least squares sense. Furthermore, an element of novelty in this contribution is the derivation of an AR-marker uncertainty model, which is used as a constraint in the optimization graph.

For the purpose of minimizing the estimation error of the RGB-D sensor and the AR-marker poses, the g2o framework [20] is used. The approach presented here builds on three major components, described in the following subsections: the derivation of an AR-marker uncertainty model, the estimation of the transformations between the markers (here denoted as the *calibration* step) and the final part of accumulating point clouds in a single, consistent frame (the *modeling* step).

B. AR-marker uncertainty model

The procedure for determining the uncertainty of an AR-marker from a 3D point cloud relies on fitting a plane, in a least-squares sense, to the marker. The plane is parametrized by $(\hat{\mathbf{n}}, d)$ and the uncertainty of the plane fitting is given by covariance matrix $\mathbf{C}_{\hat{\mathbf{n}}d}$.

1) Translation Covariance: Let $\mathbf{m} = \begin{bmatrix} u, v \end{bmatrix}^{\mathsf{T}}$ be the 2D pixel coordinates of a marker corner point; this is returned by ARToolkit [9], [21]. Let $\mathbf{p} = \begin{bmatrix} x, y, z \end{bmatrix}^{\mathsf{T}}$ be a corner point obtained by the intersection of the ray at the marker pixel $\mathbf{m}$. Given the camera matrix, $\mathbf{A}$, it holds that:

$$ \mathbf{A} \cdot \mathbf{p} \simeq \begin{pmatrix} \mathbf{m} \\ 1 \end{pmatrix} \tag{2} $$

Writing the first two rows, normalized by z:

$$ \mathbf{A}_{00} \cdot \frac{x}{z} + \mathbf{A}_{01} \cdot \frac{y}{z} + \mathbf{A}_{02} = u \tag{3} $$

$$ \mathbf{A}_{10} \cdot \frac{x}{z} + \mathbf{A}_{11} \cdot \frac{y}{z} + \mathbf{A}_{12} = v \tag{4} $$

Furthermore, the corner point $\mathbf{p}$ lies on the detected plane:

$$ \hat{\mathbf{n}}^{\mathsf{T}}\mathbf{p} = d \tag{5} $$

From Eq. 3, 4, by perturbing the x, y, z and the u, v components:

$$ \underbrace{\begin{bmatrix} \frac{\mathbf{A}_{00}}{z} & \frac{\mathbf{A}_{01}}{z} & \frac{-\mathbf{A}_{00}\cdot x - \mathbf{A}_{01}\cdot y}{z^2} \\ \frac{\mathbf{A}_{10}}{z} & \frac{\mathbf{A}_{11}}{z} & \frac{-\mathbf{A}_{10}\cdot x - \mathbf{A}_{11}\cdot y}{z^2} \end{bmatrix}}_{\triangleq \mathbf{J}_1} \cdot \begin{bmatrix} \delta x \\ \delta y \\ \delta z \end{bmatrix} = \begin{bmatrix} \delta u \\ \delta v \end{bmatrix} \tag{6} $$

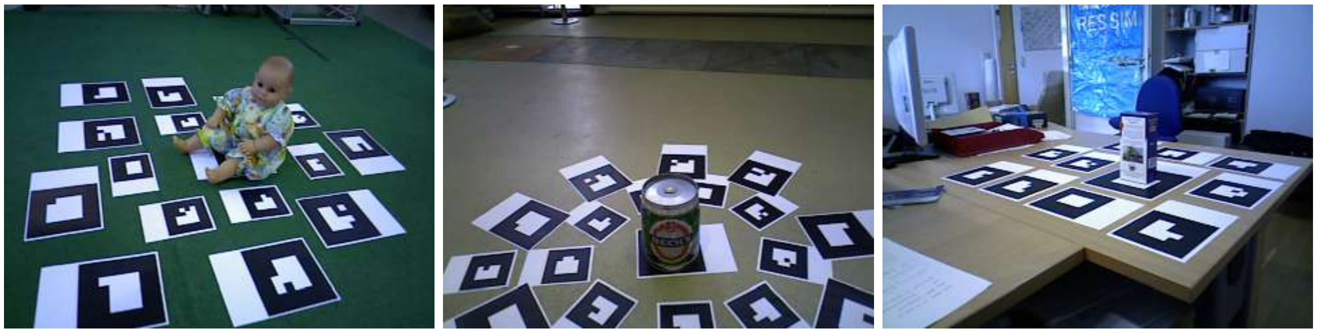


Fig. 4. The Augmented Reality (AR) markers that provide coarse localization information can be freely placed in the scene where the object is recorded. This allows the modeling of objects of very different sizes.

From Eq. 5 by perturbing $\mathbf{p}$, $\hat{\mathbf{n}}$ and d:

$$\hat{\mathbf{n}}^{\mathsf{T}} \cdot \begin{bmatrix} \delta x \\ \delta y \\ \delta z \end{bmatrix} = \underbrace{\begin{bmatrix} -\mathbf{p}^{\mathsf{T}} \; 1 \end{bmatrix}}_{\triangleq \mathbf{J}_2} \cdot \begin{bmatrix} \delta \hat{\mathbf{n}} \\ \delta d \end{bmatrix} \tag{7}$$

Concatenating Eq. 6, 7 yields:

$$\underbrace{\begin{bmatrix} \mathbf{J}_1 \\ \hat{\mathbf{n}}^{\mathsf{T}} \end{bmatrix}}_{\triangleq \mathbf{B}} \cdot \begin{bmatrix} \delta x \\ \delta y \\ \delta z \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{I}_{2\times 2} & \mathbf{O}_{2\times 4} \\ \mathbf{O}_{1\times 2} & \mathbf{J}_2 \end{bmatrix}}_{\triangleq \mathbf{J}_3} \cdot \begin{bmatrix} \delta u \\ \delta v \\ \delta \hat{\mathbf{n}} \\ \delta d \end{bmatrix} \tag{8}$$

Rewriting Eq. 8:

$$\begin{bmatrix} \delta x \\ \delta y \\ \delta z \end{bmatrix} = \underbrace{\mathbf{B}^{-1} \cdot \mathbf{J}_3}_{\triangleq \mathbf{J}_4} \cdot \begin{bmatrix} \delta u \\ \delta v \\ \delta \hat{\mathbf{n}} \\ \delta d \end{bmatrix} \tag{9}$$

Finally, the covariance of a corner point, $\mathbf{C}_{\mathbf{p}}$, is obtained as:

$$\mathbf{C}_{\mathbf{p}} = \mathbf{J}_4 \cdot \begin{bmatrix} \mathbf{C}_{uv} & \mathbf{0}_{2\times 4} \\ \mathbf{0}_{4\times 2} & \mathbf{C}_{\hat{\mathbf{n}}d} \end{bmatrix} \cdot \mathbf{J}_4^{\mathsf{T}} \tag{10}$$

where $\mathbf{C}_{uv}$ is the uncertainty of a pixel, defined as: $\mathbf{C}_{uv} = \sigma^2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $\sigma^2 = 1$ pixel^2, where $\mathbf{C}_{\hat{\mathbf{n}}d}$ is the plane uncertainty obtained from fitting a plane to the marker.

The translation covariance of the marker is obtained from the covariance of the four corner points, $\mathbf{p}_i$, of the AR-marker as:

$$\mathbf{C}_{translation} = \frac{1}{16} \cdot \sum_{i=0}^{3} \mathbf{C}_{\mathbf{p}_i} \tag{11}$$

2) Rotation Covariance: The computation of the rotation covariance assumes that the rotation of the marker with respect to the camera frame, ${}^{C}_{M}\mathbf{R}$, parametrized by roll, pitch, yaw angles, is available. Additionally, it is assumed that the corner points, $\mathbf{p}_i$, are numbered as shown in Fig. 5.

The $\mathbf{x}$ and $\mathbf{y}$ axes of the marker with respect to the camera frame, ${}^{C}\mathbf{x}$, ${}^{C}\mathbf{y}$, respectively, are defined as:

$${}^{C}\mathbf{x_1} = \frac{{}^{C}\mathbf{p_1} - {}^{C}\mathbf{p_0}}{\| {}^{C}\mathbf{p_1} - {}^{C}\mathbf{p_0} \|}; \; {}^{C}\mathbf{x_2} = \frac{{}^{C}\mathbf{p_2} - {}^{C}\mathbf{p_3}}{\| {}^{C}\mathbf{p_2} - {}^{C}\mathbf{p_3} \|}$$

$${}^{C}\mathbf{y_1} = \frac{{}^{C}\mathbf{p_1} - {}^{C}\mathbf{p_2}}{\| {}^{C}\mathbf{p_1} - {}^{C}\mathbf{p_2} \|}; \; {}^{C}\mathbf{y_2} = \frac{{}^{C}\mathbf{p_0} - {}^{C}\mathbf{p_3}}{\| {}^{C}\mathbf{p_0} - {}^{C}\mathbf{p_3} \|}$$

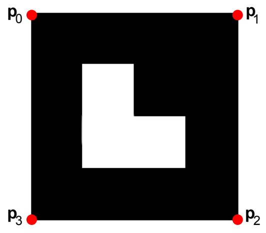


Fig. 5. AR-Marker Corner Points $\mathbf{p}_i$

Using the above, let

$$\eta = \begin{bmatrix} {}^{C}\hat{\mathbf{n}} \\ {}^{C}\mathbf{x_1} \\ {}^{C}\mathbf{y_1} \\ {}^{C}\mathbf{x_2} \\ {}^{C}\mathbf{y_2} \end{bmatrix} = \left.\begin{bmatrix} {}^{C}_{M}\mathbf{R} \cdot {}^{M}\mathbf{z} \\ {}^{C}_{M}\mathbf{R} \cdot {}^{M}\mathbf{x} \\ {}^{C}_{M}\mathbf{R} \cdot {}^{M}\mathbf{y} \\ {}^{C}_{M}\mathbf{R} \cdot {}^{M}\mathbf{x} \\ {}^{C}_{M}\mathbf{R} \cdot {}^{M}\mathbf{y} \end{bmatrix}\right\} \triangleq \mathbf{q}(r,p,y) \tag{12}$$

where ${}^{M}\mathbf{x} = [1\ 0\ 0]^{\mathsf{T}}$, ${}^{M}\mathbf{y} = [0\ 1\ 0]^{\mathsf{T}}$, ${}^{M}\mathbf{z} = [0\ 0\ 1]^{\mathsf{T}}$

$$\mathbf{C}_{\eta\eta} = \mathbf{J} \cdot \mathbf{C}_{rpy} \cdot \mathbf{J}^{\mathsf{T}}, \text{ where } \mathbf{J} = \frac{\partial \mathbf{q}(r,p,y)}{\partial (r,p,y)} \tag{13}$$

Solving for $\mathbf{C}_{rpy}$ yields:

$$\mathbf{C}_{rpy} = (\mathbf{J}^{\mathsf{T}}\mathbf{J})^{-1}\mathbf{J}^{\mathsf{T}} \cdot \mathbf{C}_{\eta\eta} \cdot \mathbf{J}(\mathbf{J}^{\mathsf{T}}\mathbf{J})^{-1} \tag{14}$$

$$\text{where } \mathbf{C}_{\eta\eta} = \begin{bmatrix} \mathbf{C}_{\hat{\mathbf{n}}\hat{\mathbf{n}}} & \mathbf{0}_{3\times 12} & \\ \mathbf{0}_{3\times 3} & \mathbf{C}_{\mathbf{x_1x_1}} & \mathbf{0}_{3\times 9} \\ \mathbf{0}_{3\times 6} & \mathbf{C}_{\mathbf{y_1y_1}} & \mathbf{0}_{3\times 6} \\ \mathbf{0}_{3\times 9} & \mathbf{C}_{\mathbf{x_2x_2}} & \mathbf{0}_{3\times 3} \\ & \mathbf{0}_{3\times 12} & \mathbf{C}_{\mathbf{y_2y_2}} \end{bmatrix},$$

$\mathbf{C}_{\hat{\mathbf{n}}\hat{\mathbf{n}}}$ is obtained from $\mathbf{C}_{\hat{\mathbf{n}}d}$

$$\mathbf{C}_{\mathbf{x_1x_1}} = \mathbf{J}_1 \cdot \mathbf{C}_{\mathbf{p}_1} \cdot \mathbf{J}_1^{\mathsf{T}} + \mathbf{J}_0 \cdot \mathbf{C}_{\mathbf{p}_0} \cdot \mathbf{J}_0^{\mathsf{T}},$$
$$\mathbf{C}_{\mathbf{y_1y_1}} = \mathbf{J}_3 \cdot \mathbf{C}_{\mathbf{p}_1} \cdot \mathbf{J}_3^{\mathsf{T}} + \mathbf{J}_2 \cdot \mathbf{C}_{\mathbf{p}_2} \cdot \mathbf{J}_2^{\mathsf{T}},$$
$$\mathbf{C}_{\mathbf{x_2x_2}} = \mathbf{J}_5 \cdot \mathbf{C}_{\mathbf{p}_2} \cdot \mathbf{J}_5^{\mathsf{T}} + \mathbf{J}_4 \cdot \mathbf{C}_{\mathbf{p}_3} \cdot \mathbf{J}_4^{\mathsf{T}},$$
$$\mathbf{C}_{\mathbf{y_2y_2}} = \mathbf{J}_7 \cdot \mathbf{C}_{\mathbf{p}_0} \cdot \mathbf{J}_7^{\mathsf{T}} + \mathbf{J}_6 \cdot \mathbf{C}_{\mathbf{p}_3} \cdot \mathbf{J}_6^{\mathsf{T}},$$

$$\mathbf{J}_1 = \tfrac{\partial \mathbf{x_1}}{\partial \mathbf{p_1}}, \quad \mathbf{J}_0 = -\mathbf{J}_1, \quad \mathbf{J}_3 = \tfrac{\partial \mathbf{y_1}}{\partial \mathbf{p_1}}, \quad \mathbf{J}_2 = -\mathbf{J}_3,$$
$$\mathbf{J}_5 = \tfrac{\partial \mathbf{x_2}}{\partial \mathbf{p_2}}, \quad \mathbf{J}_4 = -\mathbf{J}_5, \quad \mathbf{J}_7 = \tfrac{\partial \mathbf{y_2}}{\partial \mathbf{p_0}}, \quad \mathbf{J}_7 = -\mathbf{J}_6$$

Finally, from Eq. 11, 14, the covariance of the AR-marker is written as:

$$\mathbf{C}_{marker} = \begin{bmatrix} \mathbf{C}_{translation} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \mathbf{C}_{rpy} \end{bmatrix} \tag{15}$$

C. Calibration Step

This step assumes an experiment set-up where markers are randomly distributed across the scene, as shown in Fig. 4. The purpose of the calibration step is to estimate ${}^{O}_{m_i}\mathbf{T}\ \forall i$, the transformations between the marker on which the object is going to be placed, henceforth object marker, and the rest of the markers.

For estimating the marker poses the g2o graph optimization framework [20] is used. g2o is a framework for optimizing nonlinear least squares problems that can be represented as graphs [20]. The problems of interest, i.e., estimating the marker poses (translation and orientation) during *calibration* and estimating the camera poses during *modeling*, can be represented as non-linear least squares optimization problems:

$$\mathbf{F}(x) = \sum_{i,j} e(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij})^T \boldsymbol{\Omega}_{ij} e(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij}) \quad (16)$$

$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x}} \mathbf{F}(\mathbf{x}) \quad (17)$$

where $\mathbf{x} = (\underbrace{t_x, t_y, t_z}_{\text{translation}}, \underbrace{r_w, r_x, r_y, r_z}_{\text{quaternion rotation}})^T$ represents a vertex in the optimization graph and $\mathbf{z}_{ij}$ and $\boldsymbol{\Omega}_{ij}$ represent the mean and the information matrix of an edge in the graph, i.e., a constraint between the parameters $\mathbf{x}_i$ and $\mathbf{x}_j$.

In the calibration step, the g2o graph consists of the following vertex types:

- sensor poses expressed in the object marker frame $\mathcal{F}_O$ and encoded in a homogeneous transformation matrix, ${}^{O}_{C}\mathbf{T}$
- marker poses ${}^{O}_{m_i}\mathbf{T}$, also expressed in the object marker frame $\mathcal{F}_O$

The edges in the graph are directed edges from camera poses to marker pose and the edge constraint is ${}^{C}_{m_i}\mathbf{T}$:

$${}^{O}_{C}\mathbf{T} \cdot {}^{C}_{m_i}\mathbf{T} = {}^{O}_{m_i}\mathbf{T} \quad (18)$$

The information matrix $\boldsymbol{\Omega}_{ij}$ between camera vertex i and marker vertex j is obtained from the covariance $\mathbf{C}_{marker_j}$ of the marker at vertex j, as follows:

$$\boldsymbol{\Omega}_{ij} = c_j \cdot \mathbf{C}^{+}{}_{marker_j} \quad (19)$$

where the generalized inverse of the marker covariance matrix is scaled with the confidence, c_j, with which marker j was detected by ARToolkit [9], [21].

The optimized g2o graph is shown in Fig. 6, where the sensor poses are indicated by the triangular vertices in the upper part of the figure and the marker poses are indicated by the vertices in the lower part of the figure.

D. Modeling

The modeling step is a repetition of the calibration step wherein the physical object is placed on the object marker.

After the g2o optimization, the 3D model is built by accumulating surface points from the object, ${}^{C}\underline{\mathbf{p}}$, into a single consistent frame $\mathcal{F}_O$, i.e., the object frame. More specifically, the optimized sensor poses are used to transform the points on the object's surface into $\mathcal{F}_O$:

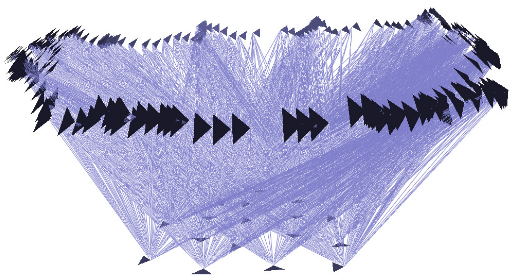


Fig. 6. The g2o graph after *calibration*, i.e., after the RGB-D sensor has been moved over the freely placed markers without an object in the scene. The upper triangles show the estimated sensor poses, the lower ones are the estimated marker poses.

$${}^{O}\underline{\mathbf{p}} = {}^{O}_{C}\mathbf{T} \cdot {}^{C}\underline{\mathbf{p}} \quad (20)$$

The final refinement steps applied to the 3D model consist of a standard ICP algorithm, used to register the current frame with the previously registered frames, and a voxelgrid filter to remove any outlying points.

III. EXPERIMENTS AND RESULTS

This section introduces the experiment set-up and presents the modeling results of ground truth objects, with simple geometries, and objects with more complex geometries.

A. Experiment Set-up

In these experiments, two types of AR-markers were used: 10 large markers, with a single marker printed on a DIN A4 paper and 10 small markers, printed on a DIN A5 paper. The markers were distributed on the ground, as presented in Fig. 4. The Kinect sensor was then moved by hand smoothly, so as not to induce motion blur, and circularly around the markers, at a distance of $\approx 1m$ from the AR-markers.

B. Ground Truth Objects

Models of ground truth objects, such as sports balls and cuboidal boxes, are used to quantify the accuracy of the proposed approach. The error measurements differ for the two objects:

- For spherical objects, the estimate of R, the radius of the best fitting sphere is compared with the true object radius. R and $\mathbf{r}_o$, the 3D centroid of the estimated sphere, are found by minimizing the cost function e, where $\mathbf{r}_i = \begin{bmatrix} x_i & y_i & z_i \end{bmatrix}^\mathsf{T}$ are N points on the object:

$$e = \sum_{i=1}^{N} (\|\mathbf{r}_o - \mathbf{r}_i\| - R)^2 \quad (21)$$

Eq. 21 is minimized by setting $\frac{\partial e}{\partial \mathbf{r}_o} = \mathbf{0}$ and $\frac{\partial e}{\partial R} = 0$:

$$\frac{\partial e}{\partial \mathbf{r}_o} = \sum_{i=1}^{N}(\mathbf{r}_o - \mathbf{r}_i) - R \cdot \frac{\mathbf{r}_o - \mathbf{r}_i}{\|\mathbf{r}_o - \mathbf{r}_i\|} = \mathbf{0} \quad (22)$$

$$\Rightarrow \quad \mathbf{r}_o = \frac{1}{N} \cdot \left[\sum_{i=1}^{N}\mathbf{r}_i + R \cdot \frac{\mathbf{r}_o - r_i}{\|\mathbf{r}_o - r_i\|}\right] \quad (23)$$

$$R = \frac{1}{N} \cdot \sum_{i=1}^{N}\|\mathbf{r}_o - \mathbf{r}_i\| \quad (24)$$

Eq. 23 and 24, form the basis of an iterative procedure to determine $\mathbf{r}_o$ and R.

- Measuring the error for cuboidal objects is done by obtaining the axes-aligned bounding box of the point cloud and comparing the volume defined by the bounding box to the volume defined by the true cuboidal box.

The errors listed in Table I are reported for two modeling scenarios for each object: using the AR-marker uncertainty model and not using the uncertainty model. In the latter case, the information matrix is obtained by scaling the identity matrix with the confidence, c_j of the marker detection: $\mathbf{\Omega_{ij}} = c_j \cdot \mathbf{I}_{6\times 6}$. The results show smaller errors when using the AR-marker uncertainty model.

The results in Table I were generated on an Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz, 16GB RAM machine, running 64-bit Ubuntu 12.04. The results show errors of $\approx 3\,mm\,(3\%)$ for sports balls models (cf. Fig. 7(a), Fig. 7(b)). For cuboidal boxes models (cf. Fig. 7(c), Fig. 7(d), Fig. 7(e), Fig. 7(f)), the errors are $\leq 2\,mm^3$ (3%).

The above evaluation shows that the approach presented in this paper is fairly accurate and able to model objects of different sizes, from small sports balls ($0.1m$ radius) to medium-sized boxes ($0.08m^3 = 80l$ volume).

C. Other Objects

In addition to ground truth objects, models of objects that can be found in container unloading scenarios are shown in Fig. 7. These are objects of various sizes and categories, e.g., parcels, barrels, car tires, etc.

IV. CONCLUSIONS

In this paper an approach for modeling physical objects as 3D point clouds was presented. The method relies on data from an RGB-D sensor, such as Microsoft Kinect, and uses a type of fiducial markers to accurately estimate the pose of the object in the scene. After the object is segmented from the scene, the surface points, obtained from multiple object views, are accumulated into a single consistent frame which forms the initial model. The model becomes final by sequential refinements using standard registration algorithms and outlier removal methods. Evaluations of the final object models indicate good levels of accuracy.

This paper introduced a convenient method for modeling objects of different sizes, by easily changing the experimental set-up, i.e., the size and distribution of the AR-markers. Furthermore, an uncertainty model for an AR-marker was developed in the context of object modeling. However, it may be relevant for other applications involving AR-marker pose detection.

REFERENCES

[1] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser, "The princeton shape benchmark," in *Shape Modeling Applications, 2004. Proceedings*, 2004, pp. 167–178.

[2] Google, "3d warehouse," http://sketchup.google.com/3dwarehouse/, 2012.

[3] 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," *The international Journal of Robotics Research*, vol. 31, no. 8, pp. 927–934, 2012.

[4] W. Associates, "3d scanning and reverse engineering," http://www.wohlersassociates.com/scanning.html, 2012.

[5] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view rgb-d object dataset," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011, pp. 1817–1824.

[6] D. Di Marco, A. Koch, O. Zweigle, K. Haussermann, B. Schiessle, P. Levi, D. Galvez-Lopez, L. Riazuelo, J. Civera, J. M. M. Montiel, M. Tenorth, A. Perzylo, M. Waibel, and R. van de Molengraft, "Creating and using roboearth object models," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2012, pp. 3549–3550.

[7] M. Waibel, M. Beetz, J. Civera, R. D'Andrea, J. Elfring, D. Galvez-Lopez, K. Haussermann, R. Janssen, J. M. M. Montiel, A. Perzylo, B. Schiessle, M. Tenorth, O. Zweigle, and R. van de Molengraft, "Roboearth," *Robotics and Automation Magazine, IEEE*, vol. 18, no. 2, pp. 69–82, 2011.

[8] H. Kato and M. Billinghurst, "Marker tracking and hmd calibration for a video-based augmented reality conferencing system," in *Augmented Reality, 1999. (IWAR '99) Proceedings. 2nd IEEE and ACM International Workshop on*, 1999, pp. 85–94.

[9] M. Billinghurst and H. Kato, *Proceedings of the First IEEE International Augmented Reality Toolkit Workshop*, ser. Augmented Reality Toolkit, The First IEEE International Workshop, 2002.

[10] K. Jongbae and J. Heesung, "Vision-based location positioning using augmented reality for indoor navigation," *Consumer Electronics, IEEE Transactions on*, vol. 54, no. 3, pp. 954–962, 2008.

[11] O. Mohareri and A. B. Rad, "Autonomous humanoid robot navigation using augmented reality technique," in *Mechatronics (ICM), 2011 IEEE International Conference on*, 2011, pp. 463–468.

[12] M. Ishida and K. Shimonomura, "Marker based camera pose estimation for underwater robots," in *System Integration (SII), 2012 IEEE/SICE International Symposium on*, 2012, pp. 629–634.

[13] W. Huiyao, C. Zhihao, and W. Yingxun, "Vison-based auxiliary navigation method using augmented reality for unmanned aerial vehicles," in *Industrial Informatics (INDIN), 2012 10th IEEE International Conference on*, 2012, pp. 520–525.

[14] P. Besl and N. McKay, "A Method for Registration of 3-D Shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, pp. 239–256, 1992.

[15] Craig, J. J., *Introduction to robotics – Mechanics and control.* Prentice Hall, 2005.

[16] Y. Chen and G. Medioni, "Object modeling by registration of multiple range images," in *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, 1991, pp. 2724–2729 vol.3.

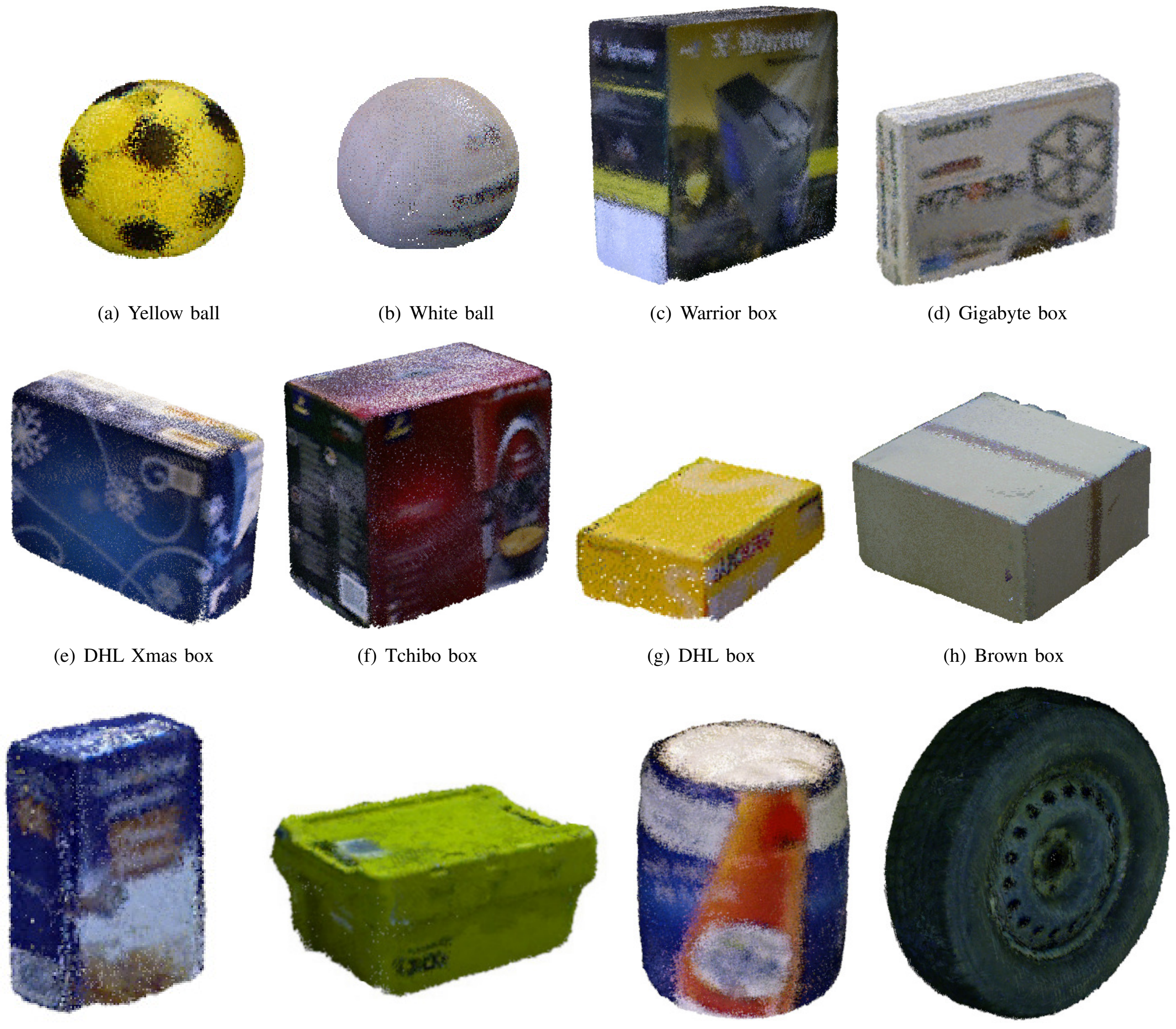
[17] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. W. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *ISMAR*, 2011, pp. 127–136.

[18] Ruhnke, M. and Kümmerle, R. and Grisetti, G. and Burgard, W., "Highly Accurate 3D Surface Models by Sparse Surface Adjustment," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012.

[19] B. Triggs, P. Mclauchlan, R. Hartley, and A. Fitzgibbon, "Bundle adjustment – a modern synthesis," in *Vision Algorithms: Theory and Practice, LNCS*. Springer Verlag, 2000, pp. 298–375.

[20] Kuemmerle, R. and Grisetti, G. and Strasdat, H. and Konolige, K. and Burgard, W., "g2o: A general framework for graph optimization," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

[21] M. Ferguson, "ROS package for AR marker detection and pose estimation, based on ARToolkit." [Online]. Available: http://ros.org/wiki/ar_kinect



(a) Yellow ball (b) White ball (c) Warrior box (d) Gigabyte box

(e) DHL Xmas box (f) Tchibo box (g) DHL box (h) Brown box

(i) Sugar bag (j) Post box (k) Beer barrel (l) Car tire

Fig. 7. Collection of object models

TABLE I

ERROR MEASUREMENT FOR GROUND TRUTH OBJECTS

Object	Description	Absolute error for complete model $[m]$	Relative error for complete model	Modeling duration $[s]$
Yellow ball	w/ uncertainty model	1.989e-03	1.989%	69s
Yellow ball	w/o uncertainty model	1.61e-02	16.1%	80s
White ball	w/ uncertainty model	3.567e-03	3.567%	60s
White ball	w/o uncertainty model	2.12e-02	21.2%	77s

Object	Description	Absolute error for complete model $[m^3]$	Relative error for complete model	Modeling duration $[s]$
Warrior box	w/ uncertainty model	1.961e-03	2.485%	756s
Warrior box	w/o uncertainty model	6.584e-03	8.34%	870s
Gigabyte box	w/ uncertainty model	1.843e-04	3.07%	112s
Gigabyte box	w/o uncertainty model	9.109e-04	15.16%	193s
DHL Xmas box	w/ uncertainty model	2.893e-04	1.95%	218s
DHL Xmas box	w/o uncertainty model	1.21e-03	8.107%	240s
Tchibo box	w/ uncertainty model	9.77e-04	2.584%	427s
Tchibo box	w/o uncertainty model	3.1147e-03	8.24%	521s