HHS Public Access

Author manuscript

IEEE Robot Autom Lett. Author manuscript; available in PMC 2019 January 11.

Published in final edited form as:

IEEE Robot Autom Lett. 2016 July ; 1(2): 1156–1163. doi:10.1109/LRA.2016.2518242.

Articulated Robot Motion for Simultaneous Localization and Mapping (ARM-SLAM)

Matthew Klingensmith[1], **Siddartha S. Sirinivasa**[2], and **Michael Kaess**[3]

[1,2,3]Carnegie Mellon Robotics Institute, 5000 Forbes Avenue, Pittsburgh PA 15213

Abstract

A robot with a hand-mounted depth sensor scans a scene. When the robot's joint angles are not known with certainty, how can it best reconstruct the scene? In this work, we simultaneously estimate the joint angles of the robot and reconstruct a dense volumetric model of the scene. In this way, we perform simultaneous localization and mapping in the configuration space of the robot, rather than in the pose space of the camera. We show using simulations and robot experiments that our approach greatly reduces both 3D reconstruction error and joint angle error over simply using the forward kinematics. Unlike other approaches, ours directly reasons about robot joint angles, and can use these to constrain the pose of the sensor. Because of this, it is more robust to missing or ambiguous depth data than approaches that are unconstrained by the robot's kinematics.

I. INTRODUCTION

Uncertainty is a central problem in robotics. In order to understand and interact with the world, robots need to interpret signals from noisy sensors to reconstruct clear models not only of the world around them, but also their own internal state. For example, a mobile robot navigating an unknown space must simultaneously reconstruct a model of the world around it, and localize itself against that model using noisy sensor data from wheel odometry, lasers, cameras, or other sensors. This problem (called the Simultaneous Localization and Mapping (SLAM) problem) is very well-studied in the mobile robotics community.

Less well-studied is the equivalent problem for robot manipulators. That is, given a multi-jointed robot arm with a noisy hand-mounted sensor, how can the robot simultaneously estimate its state and generate a coherent 3D model of the world? We call this the articulated SLAM problem. Solving it would allow the robot manipulator to plan around obstacles and locate objects of interest. If done online, the SLAM system would enable the robot to do eye-in-hand 3D visual servoing against the map.

At first glance, this problem appears trivial; because typically the joint angles of the robot are directly measurable from joint encoders, and the forward kinematic equations of the robot are known with certainty. Therefore, the pose of the sensor is known with certainty, and so mapping can be accomplished without simultaneously localizing the sensor. However, in practice, all robots are subject to some amount of *actuator uncertainty*. Their joint encoders do not perfectly capture the true geometric angles of the robot's joints

because of gear thrash, cable stretch, nonrigid deformities, and other unknown dynamics (see section III-B).

Given actuator uncertainty and sensor uncertainty, what is the correct way of simultaneously constructing a model of the world and estimating the robot's state? In this work, we show that certain contemporary visual SLAM techniques can be mapped to the articulated SLAM problem by using the robot's joint configuration space as the state space for localization, rather than the typical $SE(3)$. We map one kind of visual SLAM technique, *Kinect Fusion*[9] to the robot's configuration space, and show how the robot's joint encoders can be used appropriately to inform the pose of the camera.

The idea that the configuration of the robot is not merely a *sensor* which informs the pose of the camera, but rather it is the underlying *latent state* of the system is critical to our analysis. Tracking the configuration of the robot directly allows us to build algorithms on top of the SLAM system which depend on knowledge of the joint angles (such as motion planners and control systems).

II. RELATED WORK

Our work combines ideas from two other related fields: visual SLAM, and articulated tracking. Visual SLAM is concerned with tracking the pose of a camera as it moves through an unknown scene. Articulated tracking (a subset of *motion capture*) is concerned with finding the joint angles of a robot or actor by use of an externally mounted camera. There is also some relation to robot arm state estimation in control theory, and to visual servoing.

A. Robot Arm State Estimation

In control theory, often the state of the system being controlled cannot be directly observed, but instead must be inferred from sensor measurements. Generalized filtering techniques (such as Kalman filters, and Particle filters) have long been applied to robot arms which have only partially observable state. For instance, recent works have tracked the state of flexible robot arms using inertial sensors on the end effector [1] and from motor voltage inputs alone using a learned dynamic model and particle filter [29].

State estimation literature from control theory provides a backdrop for our work, which is concerned with estimating the state of a robot arm. However, unlike these other works we wish to also simultaneously estimate a model of the scene using a visual sensor, and use this model to further inform the state of the robot.

B. Articulated Tracking

Tracking articulated bodies with known kinematic structure using externally-mounted visual sensors is a well-studied topic in computer vision. For instance, commercial motion capture systems [19] often use markers (such as fiducials or reflective spheres) attached to articulated bodies along with an external camera to track the pose of human actors and objects in the scene. The kinds of motion capture systems most related to our purposes are those which use actor-mounted sensor systems (most commonly inertial sensors [26]) to measure the actor's joint angles.

When only the kinematic structure of the body is known, but no markers are available, the problem is more difficult due to the unknown correspondences between sensor measurements and the body itself (*i. e* the *segmentation problem*). But even in this case, efficient solutions for tracking articulated bodies exist.

In general, the approach is to model the articulated body so as to simulate sensor measurements at a particular configuration. Then, a maximum likelihood estimate is obtained which causes the sensor measurements from the external camera to agree with the model of the articulated body.

For instance, one method intended for tracking humans in 2D images, Articulated Iterative Closest Point (ICP) [20], computes the posterior using image-based edge features, and maximizes the likelihood of a configuration by coordinate descent in configuration space.

Another algorithm, Real-time Markerless Articulated Tracking (RMAT) [15] tracks robots using a depth camera. It uses a simple posterior model that sensor measurements are near the robot's surface as projected onto the depth image, and a simple motion model which assumes the robot's configuration changes slowly between time-steps. Sensor measurements are matched to corresponding points on the robot's body using an octree, and gradient descent is used to maximize the likelihood of the robot's configuration given its previous configuration and depth image.

A related work, Dense Articulated Real Time Tracking (DART) [27], improves upon RMAT by using a signed distance field representation of the robot's body rather than an octree, a more complex motion model that considers joint velocity, and an extended Kalman filter for tracking. DART has been shown to effectively track robots, human hands, and rigid bodies in real-time with commercial depth cameras.

Our work builds on some of the mathematical foundations of these approaches. The key difference is that we are concerned with *eye-in-hand* sensors which cannot see any part of the robot. This means we must simultaneously estimate the robot configuration *and* the structure of the scene; whereas in articulated tracking, only the robot configuration must be estimated.

C. Visual Servoing

When the camera is not mounted externally, but instead is mounted *in-hand*, it can be used to control the robot to achieve visual objectives (such as aligning points or shapes in an image). Visual servoing and control [4] expresses the robot's objective in terms of positions and velocities in the camera image frame, and system state as the robot's joint angles.

Visual servoing works are related to ours in the sense that they use a hand-mounted camera to inform the joint angles of a robot, but they typically rely on known models of the scene (in terms of image points or features), and do not explicitly consider actuator uncertainty; but in our case, we cannot assume any prior knowledge about the composition of the scene, and must consider actuator uncertainty.

However, the underlying mathematics of visual servoing relating motions in the image space to joint angle displacements are used extensively in our work. Further, our work enables a kind of model-based 3D visual servoing by way of creating a 3D model of the scene while the robot localizes itself.

D. Visual SLAM

Visual SLAM involves determining the full 6 Degree of Freedom (DOF) trajectory of a camera as it moves through a scene as well as a geometric model of the (unknown) world online [8]. A broad range of techniques have been used in the literature depending on the type of the sensor and desired world model.

When only a single (monocular) camera is available, *sparse* feature-based techniques can be used to determine the camera pose. Feature-based techniques typically minimize the reprojection error of a global set of 3D feature landmarks shared between camera frames. Examples of this kind of approach include Parallel Tracking and Mapping (PTAM)[13], and ORB-SLAM [21]. Feature-based methods have very good performance due to their sparsity, but the reconstruction quality is limited to a sparse set of 3D points.

Dense or semi-dense monocular approaches to visual SLAM, in contrast, compute image intensity error for all (or most) pixels in each camera frame. Examples include LSD-SLAM [6] and DTAM [22]. These approaches are more memory intensive and computationally expensive than their sparse counterparts, but provide much more detailed world models that are suitable for robotics.

When a depth sensor is available, dense visual SLAM is made easier because the need to estimate the depth of visual features is eliminated. Fully dense geometric methods, such as Kinect Fusion [9], Point Fusion [10] and Elastic Fusion [30] generate a full geometric 3D model of the world, which is in turn used to estimate the pose of the depth sensor using mainly geometric techniques, such as point-to-plane ICP. Fully dense methods enable very high quality pose estimation and scene reconstruction within a small area, but they tend to drift over time, and are unable to track the sensor against scenes without much geometric structure.

Our work makes use of SLAM techniques and terminology. But, unlike the pure visual SLAM problem, we are not concerned with a free-floating camera, but rather a camera attached to an articulated robot arm. Because of this, we have an extremely strong prior on the allowable motion of the camera from the robot's kinematics, and have a very strong indication of the sensor's pose from the joint encoders.

In this sense, our work is related to other SLAM works which fuse visual SLAM together with other sensors, such as inertial sensors. State of the art examples include Li *et. al* [17], Leutteneger *et. al* [16] and Forster *et. al* [3]. These techniques harness the advantages of both visual and inertial sensors to provide much more robust pose estimation in the presence of missing or ambiguous visual data. However, unlike these works, we do not treat the robot's kinematics as a mere *sensor* to inform the camera pose, but rather treat it as the true latent state of the system, and directly estimate it.

In this work, we present a fully-dense geometric visual SLAM technique which estimates the robot configuration (rather than camera pose) from an online 3D model produced with depth images. As in Kinect Fusion [9], we construct a volumetric model of the scene, and localize against it using a geometric objective function.

III. BACKGROUND

A. Robot Kinematics

A *kinematic linkage* [18] consists of a series of rigid bodies (called *links*) attached to one another through mechanisms (called *joints*) that constrain their motion. A joint has between 1 and 6 degrees of freedom (DOF) which define how it constrains the motion of its attached links. The joint which constrains link A to link B, and which has configuration q_i has the transformation:

$$T_B^A(q_i) \in SE(3) \quad (1)$$

and as q_i changes, so does the transformation between links A and B.

The transformation of any link L_i with respect to a fixed reference frame W can be calculated by traversing the kinematic tree and appending the transformations implied by each joint from the root of the tree (a process called *forward kinematics*):

$$T_{L_i}^W = T_{L_i}^{L_{i-1}}(q_{i-1})\ldots T_{L_2}^{L_1}(q_1)T_{L_1}^W \quad (2)$$

A robot's configuration $\mathbf{q} \in \mathbb{R}^{\mathbb{N}}$ is a vector which concatenates all of its joints' degrees of freedom:

$$\mathbf{q} = [q_1 \ldots q_N]^T \quad (3)$$

The partial derivative of link i's reference frame with respect to $\mathbf{q}$:

$$\mathbf{J}_i(\mathbf{q}) = \frac{\partial}{\partial \mathbf{q}} T_{L_i}^W \quad (4)$$

is called the link's *kinematic Jacobian*, and for simple kinematic chains it can be computed in closed-form efficiently.

B. Actuator Uncertainty

Robots usually have motor encoders which measure the number of rotations that each of their motors make. Motor encoders can be used to indirectly infer the angle of the robot's joints. However, intervening mechanisms (such as gear trains, non rigid links, elastic bands,

cables, etc.) make the mapping between the joint encoder reading and the robot's true joint angles unclear.

For instance, in the case of the Barrett WAM robot arm, motors drive a series of pulleys and cables which in turn rotate the joints. Depending on the amount of *torque* applied to the cables, they can stretch and deform, introducing hysteresis into the system. Boots *et. al* [2] found the end effector error on this robot due to cable stretch to be over 8 cm, and Klingensmith *et. al* [15] found it to be over 10 cm. Worse, the error is nonlinear, and depends mostly on the torque applied to the robot along its trajectory.

We sent our robot, a Kinova *Mico* [12] to ten inverse kinematics solutions, and measured the resulting end effector pose with an *Optitrack* [24] motion capture system Fig. 2. Even though mathematically, the end effector should be in exactly the same place each time, we found the end effector pose to differ by as much as 5cm. According to the manufacturer specifications [11], the robot's actuators have a resolution of 0.055°; and we found the numerical error from our inverse kinematics solver to account for less than a millimeter of end effector error. These factors are thus too small to account for the error we see. Instead, non-rigid deformation of the plastic links under gravity, and off-axis motion of the joint seems to account for this error.

In our work, we do not attempt to model the actuator uncertainty directly, and instead simply assume it follows a simple Gaussian Process model. Define the joint encoder readings as a random variable drawn from the distribution

$$\mathbf{q}_e \sim \mathbf{q} + \mathcal{N}(\mu_{\mathbf{q}}, \Sigma_{\mathbf{q}}) \quad (5)$$

where $\mu_{\mathbf{q}}$ is the mean of the distribution at $\mathbf{q}$, and $\Sigma_{\mathbf{q}}$ is its covariance. Since we anticipate the uncertainty to be more like an unmodeled dynamic effect and less like a random process, $\Sigma_{\mathbf{q}}$ is likely to be small, while $\mu_{\mathbf{q}}$ is a function representing an offset in configuration space due to the dynamic effect.

C. Depth Sensors

In this work, we will assume access to a depth camera mounted to the robot's hand. Depth cameras work by either projecting a pattern onto a scene and reading with an infrared camera, or with active time-of-flight pulses. Call the depth image I_D, it is a function with domain $\Omega \in \mathbb{R}^2$. The relationship between 3D points in the scene and 2D points on a camera image can be modeled using the simple pinhole camera intrinsic [8] model:

$$\text{Proj}(x, y, z) = [u, v] = \left[\frac{f_x x}{z} + c_x, \frac{f_y y}{z} + c_y\right] \quad (6)$$

where u, υ are the 2D images coordinates, x, y, z are the 3D point's coordinates in the camera's frame of reference (with x to the right, y down, and z forward), and f_x, f_y, c_x, c_y are the intrinsic parameters of the camera. We can also define the inverse projection model,

which takes a camera coordinate u, v and depth measurement z, and converts it into a 3D vector relative to the camera's focal point. The resulting 3D points are called the *point cloud* of the depth image. For a particular pixel u, v with depth z, its point in the point cloud is given by:

$$\text{Proj}^{-1}(u, v, z) = z\left[\frac{u - c_x}{f_x}, \frac{v - c_y}{f_y}, 1\right] \quad (7)$$

Depth cameras measure the range (z) at each image pixel to points in the scene. The depth measured by depth cameras is noisy, incomplete, and contains systematic error.

D. Dense Fusion

When multiple depth images are given at known camera poses, it is possible to reconstruct an estimate of the 3D geometry of the scene. This process is called dense fusion. In this work, we use a method of dense fusion from Curless and Levoy [5], also used in Kinect Fusion [9] and variants, called Truncated Signed Distance Field (TSDF) fusion. The TSDF ($\Phi : \mathbb{R}^3 \rightarrow [-\tau, \tau]$) stores a voxelized representation of the scene where each voxel encodes the distance to the nearest surface in meters, and has a weight. Positively signed distances correspond to points outside of surfaces, and negatively signed distances correspond to points inside of surfaces. Voxels with a distance of zero correspond implicitly to the surfaces of objects.

To fuse multiple depth images into a TSDF, we can simply compute a local linearization of the distance field around each depth pixel as projected into the scene. Overlapping linearizations are simply averaged. Curless and Levoy [5] provide an efficient means of doing this (Alg. 1), and show that the resulting implicit surface is a least squared minimizer of the point clouds from each depth image.

Kinect Fusion [9] and variants use the TSDF both for mapping and localization. New camera poses are computed by aligning the point cloud of the depth image to the previously constructed map using the gradients of the TSDF. Our work uses a similar approach to estimate the robot's configuration.

IV. Articulated Robot Manipulator SLAM

The task is, given a series of joint encoder readings $Q_e = \mathbf{q}_e^{(1)}, \ldots, \mathbf{q}_e^{(t)}$ received online along with sensor measurements from a hand-mounted depth camera $Z = z^{(1)} \ldots z^{(t)}$, simultaneously reconstruct a TSDF of the scene Φ, and estimate the true joint angles $Q = \mathbf{q}^{(1)}, \ldots \mathbf{q}^{(t)}$.

Formally, this can be expressed as a maximum likelihood estimate problem, first for localization:

Algorithm 1

FuseTSDF

// Given a depth image, sensor pose, previous TSDF, previous voxel weights, a weighting function, a volume of interest, and a truncation distance

Input: I_D, T, $\Phi^{(t-1)}$, $W^{(t-1)}$, w, V, τ

$T' \leftarrow T^{-1}$

$\Phi_t \leftarrow \Phi_{t-1}$

$W_t \leftarrow W_{t-1}$

for $\mathbf{v} \in V$ **do**

  // Depth of voxel $\mathbf{v}$

  $\mathbf{v}_c \leftarrow T'\mathbf{v}$

  $d_i \leftarrow I_D[\text{Proj}(\mathbf{v}_c)]$

  // Dist to camera plane.

  $d_v \leftarrow \mathbf{v}_c(z)$

  // Locally linear approximation.

  $u \leftarrow d_v - d_i$

  // If dist within τ

  if $|u| < \tau$ **then**

    // Weighted average

    $$\Phi_t(\mathbf{v}) \leftarrow \frac{W_t(\mathbf{v})\Phi_t(\mathbf{v}) + w(u)u}{W_t(\mathbf{v}) + w(u)}$$

    $W_t(\mathbf{v}) \leftarrow W_t(\mathbf{v}) + w(u)$

  end

end

Output: Φ_t, W_t

$$Q^* = \underset{Q}{\text{argmax}}\, \mathbf{P}(Q \mid Q_e, Z) \quad (8)$$

$$= \underset{Q}{\text{argmax}} \frac{\mathbf{P}(Z \mid Q_e, Q)\mathbf{P}(Q)}{\mathbf{P}(Q_e, Z)} \quad (9)$$

$$= \underset{Q}{\text{argmax}} \log \mathbf{P}(Z \mid Q_e, Q) + \log \mathbf{P}(Q \mid Q_e) \quad (10)$$

and for mapping

$$\Phi^* = \underset{\Phi}{\operatorname{argmax}}\, \mathbf{P}(\Phi | Q^*, Z) \quad (11)$$

As in Kinect Fusion [9], we neglect optimizing the entire trajectory and map simultaneously, and instead focus on alternatively optimizing the current pose estimate $\mathbf{q}^{(t)}$ and the current map $\Phi^{(t)}$. This implicitly assumes the problem has a Markov property, and makes the problem tractable at the expense of allowing drift over time. First, the localization step:

$$\mathbf{q}^{(t)} \leftarrow \underset{\mathbf{q}}{\operatorname{argmax}} \log \mathbf{P}\left(z^{(t)} | \mathbf{q}, \Phi^{(t-1)}\right) \quad (12)$$

$$+ \log \mathbf{P}\left(\mathbf{q} | \mathbf{q}^{(t-1)}, \mathbf{q}_e^{(t)}\right) \quad (13)$$

and the mapping step:

$$\Phi^{(t)} \leftarrow \underset{\Phi}{\operatorname{argmax}} \left(\Phi | \Phi^{(t-1)}, z^{(t)}, \mathbf{q}^{(t)}\right) \quad (14)$$

Eq. 13 has two components: a sensor posterior, and a joint encoder prior. We will now analyze each term in detail.

A. Sensor Posterior

Kinect Fusion and variants [9] treat the sensor posterior geometrically, and align sensor points to the TSDF surface using point-to-plane ICP. This geometric argument can also be derived probabilistically, as in generalized ICP [28]. If we assume that the probability of some world-projected sensor point $\mathbf{z} \in \mathbf{R}^3$ has probability proportional to its distance to the nearest surface,

$$\mathbf{P}(\mathbf{z} | \phi) \propto \exp\left(-\Phi[\mathbf{z}]^2\right) \quad (15)$$

implying in a sense that all surfaces in the scene are "blurred" with uniform Gaussian noise, it becomes straightforward to derive the sensor posterior, assuming independence between all the points in the point cloud.

$$\mathbf{P}\left(z^{(t)} | \mathbf{q}, \Phi^{(t-1)}\right) = \prod_{\mathbf{z} \in z^{(t)}} \mathbf{P}(T_{\mathbf{q}}\mathbf{z} | \Phi^{(t-1)}) \quad (16)$$

$$\propto \prod_{\mathbf{z} \in z^{(t)}} \exp\left(-\Phi[T_{\mathbf{q}}\mathbf{z}]^2 \right) \quad (17)$$

where $T_{\mathbf{q}} = T_{L_k}^{W}(\mathbf{q})$ is the pose of the sensor implied by configuration $\mathbf{q}$ in the world frame.

Admittedly this straightforward model is too simple to capture some important aspects of the depth sensor, such as the presence of occlusions and aniostropy in the sensor. More complex posteriors, like those used in DART [27], could be used in its place.

B. Joint Encoder Prior

The other term in Eq. 13 relates the probability of a robot's configuration given its joint encoders. As discussed in Section III-B, we can model this as a Gaussian process so that the prior is given by

$$\mathbf{P}(\mathbf{q}|\mathbf{q}_e) \propto \exp[(\mathbf{q} - \mu_{\mathbf{q}} - \mathbf{q}_e)^T \Sigma_{\mathbf{q}} (\mathbf{q} - \mu_{\mathbf{q}} - \mathbf{q}_e)] \quad (18)$$

in our experiments, we simply use a prior that has a mean centered on zero, with uniform noise $\Sigma_{\mathbf{q}} = \gamma \mathbf{I}$; but a more complicated prior learned from data could be used.

C. Algorithm

The sensor posterior and joint encoder prior together imply a cost function that can be minimized to localize the robot

$$C(\mathbf{q}) = \gamma \varepsilon^T \varepsilon + \frac{1}{2} \sum_{\mathbf{z} \in Z^{(t)}} \Phi^{(t-1)}[T_{\mathbf{q}}\mathbf{z}]^2 \quad (19)$$

where $\varepsilon = \mathbf{q} - \mathbf{q}_e^{(t)}$, and γ is a regularization term. The gradient with respect to $\mathbf{q}$ can be obtained using the chain rule:

$$\nabla C = \gamma \varepsilon + \sum_{\mathbf{z} \in Z^{(t)}} \nabla \Phi^{(t-1)}[T_{\mathbf{q}}\mathbf{z}] \mathbf{J}_{\mathbf{q}}^T \nabla \Phi^{(t-1)}[T_{\mathbf{q}}\mathbf{z}] \quad (20)$$

where $\mathbf{J}_{\mathbf{q}} = \frac{\partial T_{\mathbf{q}}\mathbf{z}}{\partial \mathbf{q}}$ is the manipulator Jacobian, as described in Section III-A. This cost function can be minimized by simple gradient descent. This leads to a filtering approach, wherein an offset between the joint encoders and true joint angles is tracked over time, subject to kinematic constraints such as joint limits.

$$\varepsilon^{(t)} \leftarrow \varepsilon^{(t-1)} - \lambda \nabla C\left(\mathbf{q}_e^{(t)} + \varepsilon^{(t-1)}\right) - \gamma \varepsilon^{(t-1)} \quad (21)$$

Algorithm 2

ARM-SLAM

// Where $\mathbf{q}_e^{(t)}$ are the motor encoders at time t, λ is a learning rate, and γ is a regularization parameter.

Input: Z_t, $\mathbf{q}_e^{(t)}$, $\Phi^{(t-1)}$, λ, γ, $\varepsilon^{(t-1)}$

$\varepsilon^{(t)} \leftarrow \varepsilon^{(t-1)}$

repeat

  $\mathbf{q}^{(t)} \leftarrow \mathbf{q}_e^{(t)} + \varepsilon$

  // The camera transform.

  $T_{\mathbf{q}} \leftarrow T_{L_k}^{W}(\mathbf{q}^{(t)})$

  // Gradient of the sensor measurement posterior.

  $\nabla C \leftarrow \sum_{z \in Z^{(t)}} \left[\Phi^{(t-1)}[T_{\mathbf{q}} z] \mathbf{J}_{\mathbf{q}}^{T} \nabla \Phi^{(t-1)}[T_{\mathbf{q}} z] \right]$

  // Descend the gradient.

  $\varepsilon^{(t)} \leftarrow \varepsilon^{(t)} - \lambda \nabla C - \gamma \varepsilon^{(t)}$

until *convergence*;

// Mapping step.

$\Phi^{(t)} \leftarrow$ FuseTSDF ($\Phi^{(t-1)}$, $Z^{(t)}$, $\mathbf{q}^{(t)}$)

Output: $\varepsilon^{(t)}$, $\Phi^{(t)}$

For mapping, we can simply take the tracked joint encoder position as ground truth for fusing the depth image into the TSDF as in Curless and Levoy [5] (Alg. 1). Tracking and mapping are repeated in alternation.

V. EXPERIMENTS

We conducted three types of experiments to observe the behavior of this algorithm; 2D simulation experiments, 3D simulation experiments, and a real robot experiment.[1]

[1]Videos of these experiments are attached. High resolution videos are available at http://youtu.be/QrFyaxFUs9w

A. 2D Simulation

In the simple 2D simulation experiment, a 3-link serial robot manipulator with a simulated 1D depth sensor scans a scene. We added zero-centered Perlin [25] noise to its joint encoder readings. That is,

$$\mathbf{q}_e^{(t)} = \mathbf{q}^{(t)} + \beta_n \text{PERLIN}\left(s_n \mathbf{q}^{(t)}\right) \quad (22)$$

where s_n, β_n are parameters which control noise frequency and magnitude, respectively. In our experiments, $s_n = 1.0$, $\beta_n = 0.2$. The simulated depth image is noiseless.

For the world model, we constructed a simple 2D TSDF. We compare the performance of ARM-SLAM (Alg. 2) against a simple unconstrained descent algorithm which assumes the sensor can move and rotate freely, without considering the robot kinematics (Fig. 3). We found that ARM-SLAM managed to both reduce end effector error and dramatically reduce model error (Table I), whereas just using a 2D dense fusion technique without constraining using the robot's kinematics resulted in severe, unrecoverable drift because of the scene's self-similarity and the robot's fast motion. Note that in the real experiments, there is comparatively much less actuator noise, and a much smaller scene than in the 2D experiments.

B. 3D Simulation

We developed a 3D simulation of a Kinova *Mico* robot with a hand-mounted Occipital *Structure* [23] depth sensor. In the simulation, the robot scans a simulated bookshelf. As in the 2D experiments, Perlin noise is added to the ground truth joint angles to simulate actuator uncertainty. We use the *Open Chisel* [14] chunked TSDF library for mapping. The simulated depth image is noiseless. Reconstructions were done at a resolution of 1.5 cm per voxel.

We found that ARM-SLAM was able to correct for very large actuator error (see Fig. 5), resulting in a final reconstruction near the ground truth (Fig. 4). By artificially increasing the actuator noise, we found that ARM-SLAM significantly reduced the end effector error even when the uncertainty in the camera's pose was up to 12 cm (Fig. 5a), we also found ARM-SLAM to be more robust to tracking failure from lost data than unconstrained Kinect Fusion (Fig. 5b) due to the very strong motion prior from the robot kinematics.

C. Bookshelf Scanning

Using the same framework as in the 3D simulation, we reconstructed a bookshelf with a Kinova *Mico* robot with a hand-mounted Occipital *Structure* sensor (Fig. 1). The robot was teleoperated using a joystick. Beforehand, the Structure sensor was extrinsically calibrated to the robot's hand using the Tsai[7] method and a fiducial, though extrinsic calibration error cannot be ruled out. The end effector deviation was measured using an *Optitrack* motion tracking system. One challenge of working with the real robot data is that the joint encoders and depth sensor are not synchronized. The joint encoder data is emitted at ~ 500 Hz, whereas the camera data is produced at 30 Hz. To compensate for this, we store the robot's

configuration space trajectory as a series of linearly interpolated, timestamped waypoints. Using this, we can infer the joint encoder readings at the time when the depth image was received.

The 3D reconstructions (Fig. 1) show that our method is able to recover 3D structure in the scene that is lost when only the (noisy) forward kinematics are used. This is especially apparent around the edges of the bookshelf and its adjacent walls. Our reconstructions are comparable to Kinect Fusion run at the same voxel resolution (1.5 cm). We measured end-effector motion with an optical motion capture system (Fig. 5c) and found that Kinect fusion occasionally lost (and regained) tracking due to self-similar surfaces in the bookshelf and surrounding walls. Because of the strong motion prior from the robot's joints ARM-SLAM did not have this issue. However, our data from the motion capture system is too noisy to conclude ARM-SLAM performed any better than forward kinematics at reporting the true pose of the end effector (ARM-SLAM had an end effector deviation of 1.2 ± 0.9cm while forward kinematics had a deviation of 1.4 ± 1.0cm). It may be that extrinsic calibration error between the sensor and rigid hand mount is dominating any error produced at the robot's joints.

VI. DISCUSSION AND FUTURE WORK

In this work, we have introduced a framework for visual SLAM in a robot's configuration space. We have shown that our approach is capable of reconstructing scenes and reducing actuator uncertainty simultaneously. Many questions remain to be answered about this problem domain, and it is clear that our work does not yet address some of its key components.

First, since it is a pure model-based dense SLAM approach (like *Kinect Fusion*[9]), it suffers from many of the problems that plague these approaches. The system requires clear geometric structure and a large field of view to localize correctly, and since it uses no global *pose graph*, it is susceptible to drift over longer trajectories. Further, we are only able to track the configuration of the robot when a depth image is available. Also like those approaches, the underlying tracking and mapping techniques are largely based on geometric arguments, making it difficult to incorporate probabilistic models. As a consequence, we don't have a way of tracking the uncertainty in the predicted joint angles.

By committing to localization in the configuration space of the robot, rather than $SE(3)$, we gain the benefit of only predicting physically plausible camera poses. We are also able to express costs and priors (such as joint limit and self-collision costs) on robot configuration trivially. On the other hand, error that can't be expressed in the configuration space (such as error in the extrinsic calibration, or motion of the robot base) cannot be corrected for using our technique. Also, the more joints a robot has in comparison to $SE(3)$, the more work our technique has to do to compute Jacobian terms, and the larger the camera motion null-space is (worsening susceptibility to local minima). For instance, a 2-jointed robot pan-tilt head would be comparatively easy to localize vs. a highly redundant 50-jointed snake robot.

In spite of these limitations, our approach provides a good baseline for conducting further research. We are eager to re-express other visual SLAM techniques in the configuration space of the robot, and to explore other ways of correcting actuator noise through visual sensors.

Acknowledgments

This work was supported in part by Toyota USA grant No. 1011344 the U.S Office of Naval Research grant No. N000141210613 and NSF grant No. IIS-1426703

References

1. Axelsson P, Karlsson R, Norrlöf M. Bayesian state estimation of a flexible industrial robot. Control Engineering Practice. 2012:1220–1228.
2. Boots B, Arunkumar B, Fox D. Learning predictive models of a depth camera and manipulator from raw execution traces. International Conferene on Robotics and Automation. 2014:4021–4028.
3. Dellaert F, Forester C, Carlone L, Scaramuzza D. Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. Robotics Science and Systems. 2015
4. Corke, PI. Visual Servoing. World Scientific; 1994. Visual control of robot manipulators – a review; 1–31.
5. Curless B, Levoy M. A volumetric method for building complex models from range images. ACM Special Interest Group on Graphics and Interactive Techniques. 1996:303–312.
6. Engel J, Schöps T, Cremers D. LSD-SLAM: Large-scale direct monocular SLAM. European Conference on Computer Vision. 2014:834–849.
7. Gupta M, Upadhyay S, Nagawat AK. Camera calibration technique using tsais algorithm. International Journal of Enterprise Computing and Business Systems. 2011
8. Hartley, RI, Zisserman, A. Multiple View Geometry in Computer Vision. second. Cambridge University Press; 2004.
9. Izadi S, Kim D, Hilliges O, Molyneaux D, Newcombe R, Kohli P, Shotton J, Hodges S, Freeman D, Davison A, Fitzgibbon A. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. ACM Symposium on User Interface and Software Technology. 2011:559–568.
10. Keller M, Lefloch D, Lambers M, Izadi S, Weyrich T, Kolb A. Real-time 3d reconstruction in dynamic scenes using point-based fusion. International Conference on 3D Vision. 2013:1–8.
11. Kinova. K-Series Technical Specifications. 2015
12. Kinova. Kinova mico arm. 2015
13. Klein G, Murray D. Parallel tracking and mapping for small AR workspaces. International Symposium on Mixed and Augmented Reality. 2007:225–234.
14. Klingensmith M, Dryanovski I, Srinivasa S, Xiao J. Chisel: Real time large scale 3d reconstruction onboard a mobile device. Robotics Science and Systems. 2015
15. Klingensmith M, Galluzzo T, Dellin C, Kazemi M, Bagnell JA, Pollard N. Closed-loop Servoing using Real-time Markerless Arm Tracking. International Conferene on Robotics and Automation (Humanoids Workshop). 2013
16. Leutenegger S, Lynen S, Bosse M, Siegwart R, Furgale PT. Keyframe-based visual-inertial odometry using nonlinear optimization. International Journal of Robotics Research. 2015:314–334.
17. Li M, Mourikis AI. High-precision, consistent ekf-based visual-inertial odometry. International Journal of Robotics Research. 2013:690–711.
18. Mason, M. Mechanics of Robotic Manipulation. Cambridge, MA: MIT Press; 2001.
19. Moeslund TB, Hilton A, Krüger V. A survey of advances in vision-based human motion capture and analysis. Computer Vision and Image Understanding. 2006:90–126.

20. Mundermann L, Corazza S, Andriacchi TP. Accurately measuring human movement using articulated icp with soft-joint constraints and a repository of articulated models. Computer Vision and Pattern Recognition. 2007:1–6.
21. Mur-Artal R, Tardós JD. Probabilistic semi-dense mapping from highly accurate feature-based monocular slam. Robotics: Science and Systems. 2015
22. Newcombe RA, Lovegrove SJ, Davison AJ. Dtam: Dense tracking and mapping in real-time. International Conference on Computer Vision. 2011:2320–2327.
23. Occipital. Occipital structure sensor. 2015
24. Optitrack. Optitrack motion capture. 2015
25. Perlin K. Improving noise. ACM Special Interest Group on Graphics and Interactive Techniques. 2002:681–682.
26. Roetenberg D, Luinge H, Slycke P. Xsens mvn: full 6dof human motion tracking using miniature inertial sensors. xsens motion technologies bv. Technical report, XSens. 2009
27. Schmidt T, Newcombe R, Fox D. DART: Dense Articulated Real-Time Tracking. Robotics Science and Systems. 2014; 2
28. Segal A, Haehnel D, Thrun S. Generalized-icp. Robotics Science and Systems. 2009; 2
29. Wang Y, Chaib-Draa B. An adaptive nonparametric particle filter for state estimation. International Conferene on Robotics and Automation. 2012:4355–4360.
30. Whelan T, Leutenegger S, Moreno RS, Glocker B, Davison A. Elasticfusion: Dense slam without a pose graph. Robotics Science and Systems. 2015

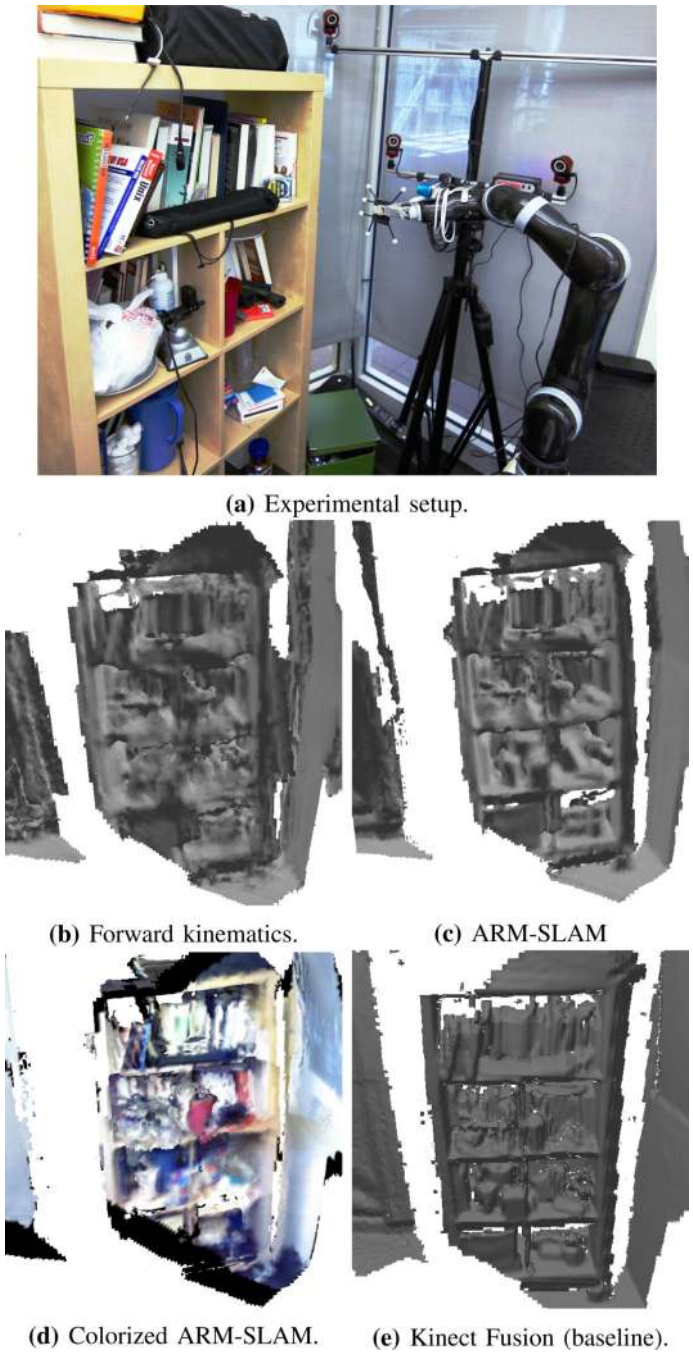


Fig. 1.

Robot scans and reconstructions of a bookshelf at 1.5cm resolution using real depth and encoder data (Section V-C). Our approach (which estimates robot configurations rather than camera poses) results in preservation of fine detail and edges that are lost when only using the robot's forward kinematics, with comparable reconstruction quality to Kinect Fusion.

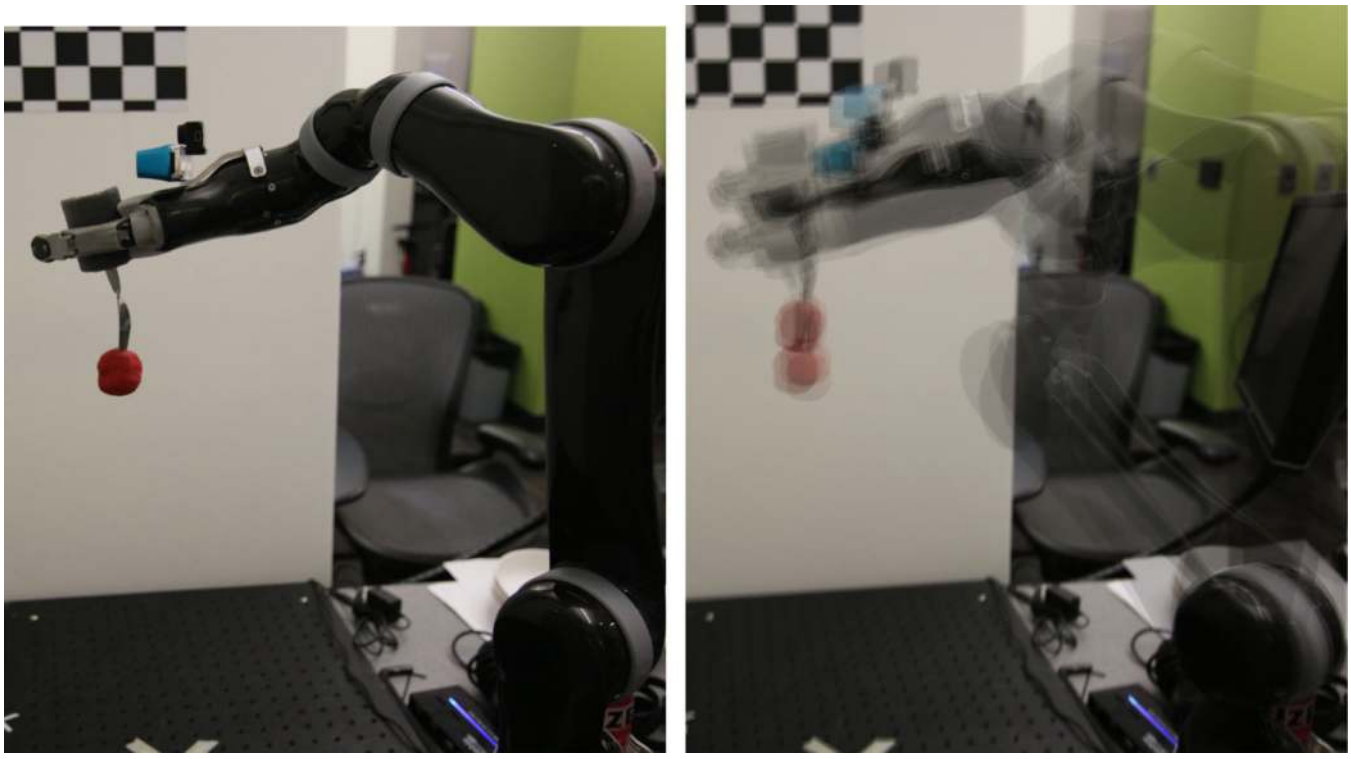


Fig. 2.

The Kinova *Mico* robot was sent to 10 inverse kinematics solutions for a fixed pose (left). The solutions are overlaid on the right. Error at the end effector exceeds 5 cm.

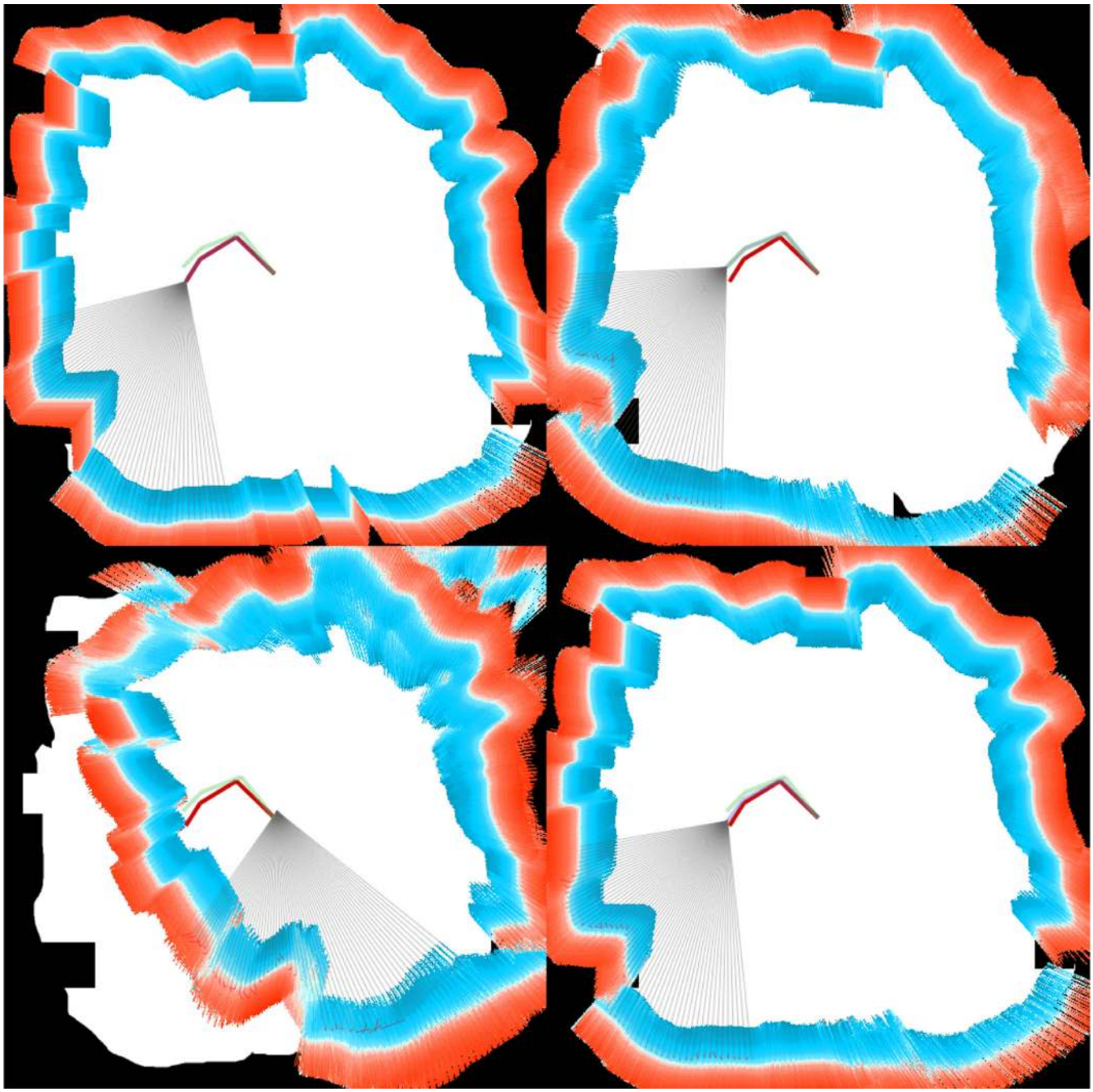


Fig. 3.

2D simulation experiment (Section V-A). The robot is shown in red. The simulated depth image is shown as grey rays. The TSDf is shown as orange or blue pixels. Top left shows the ground truth TSDF, top right is with forward kinematics only (with actuator uncertainty). Bottom left corrects actuator noise using unconstrained dense fusion. Bottom right corrects using ARM-SLAM.

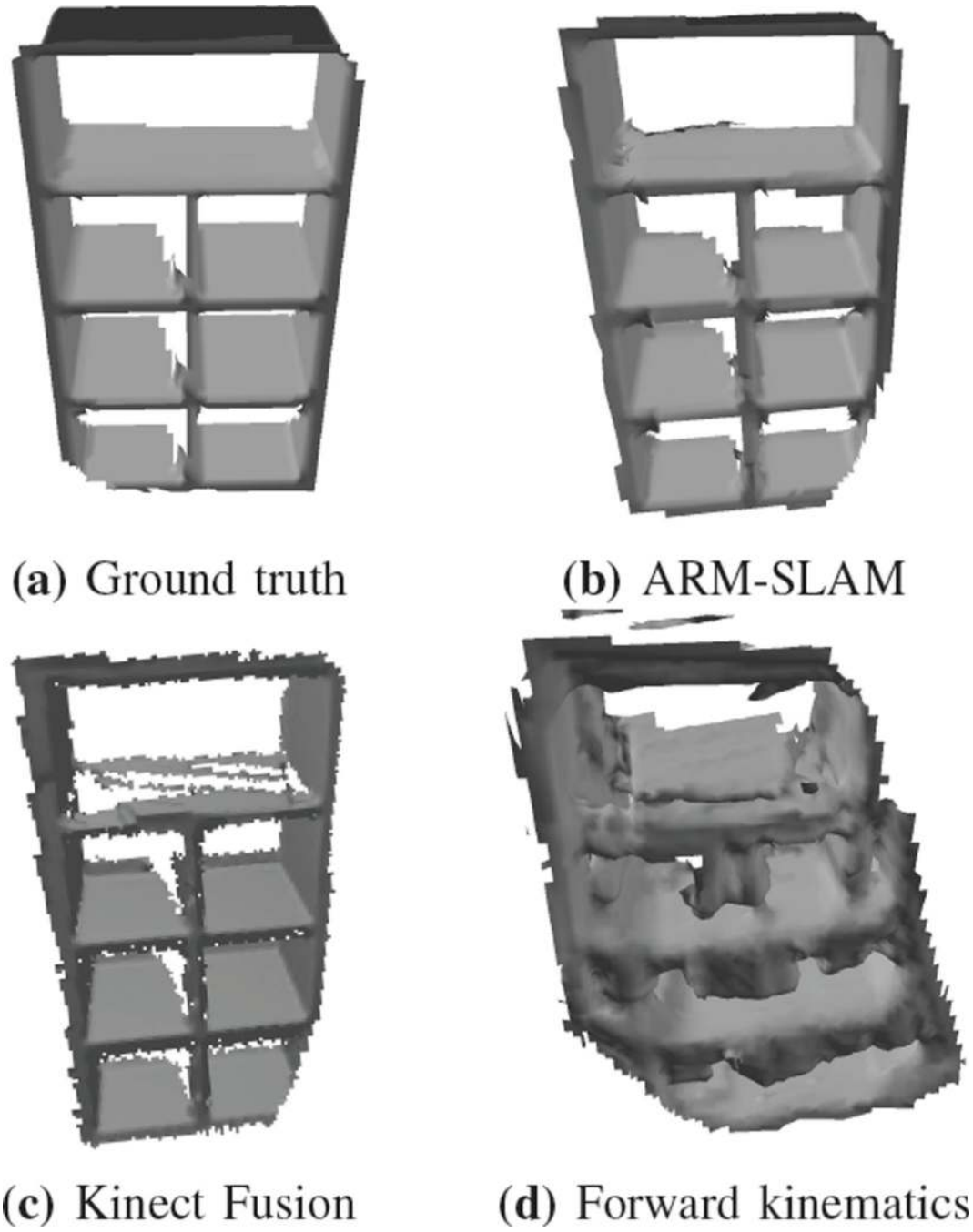
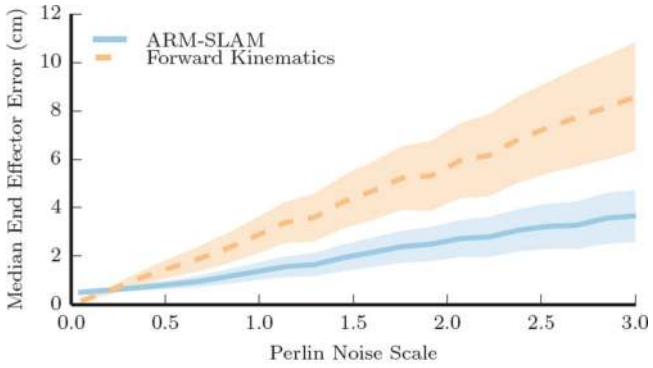
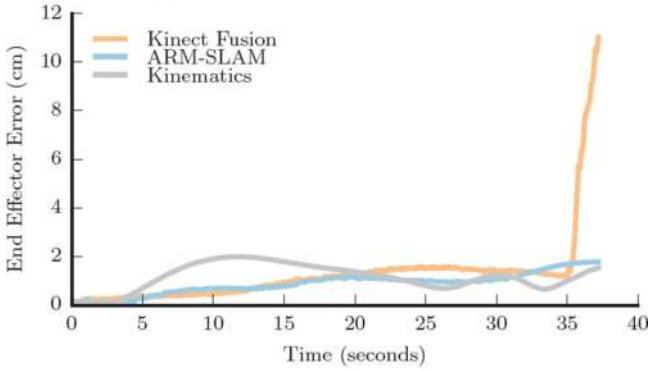


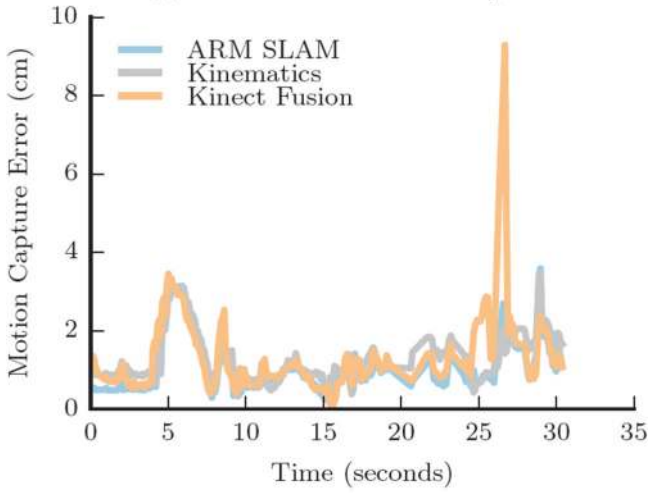
Fig. 4.
Results of the 3D simulation (Section V-B) with up to 0.8 radians of added noise per joint.



(a) 3D simulation with added noise.



(b) 3D simulation with lost tracking.



(c) Real data validated with motion capture.

Fig. 5.

End effector error observed in the 3D simulation (Section V-B) experiments. Fig. 5a: 100 trials with different noise seeds are run with increasing noise magnitude. Each trial lasts 60 seconds. The median deviation of the end effector from ground truth is recorded. Fig. 5b: in a different simulation, the robot briefly looks away from the scene and then looks back. Kinect Fusion loses tracking. Fig. 5c: end-effector deviation in the real dataset as measured by an optical motion capture system, Kinect Fusion briefly loses and then regains tracking.

TABLE I

Results for the 2D simulation experiments (Section V-A). The end effector error in pixels, joint angle error in radians, distance field error in 10^6 pixels, and occupancy classification error (the proportion of pixels misclassified as containing an obstacle) is shown for forward kinematics, unconstrained dense fusion, and ARM-SLAM for a dataset with 500 and 999 time-steps. Our approach (ARM-SLAM) reduces all three error terms.

t		Fwd. Kin.	Dense Fusion	ARM-SLAM
500	EE Err. (pix.)	5.2 ± 5.9	3.4 ± 4.2	**0.8 ± 0.7**
	Jnt. Err (rad.)	0.08 ± 0.06	–	**0.06 ± 0.05**
	SDF Err (pix.)	1.4 ± 1.7	0.8 ± 0.8	**0.5 ± 0.3**
	Class Err (%)	5.7 ± 3.2	4.7 ± 2.3	**3.5 ± 0.6**
999	EE Err. (pix.)	9.2 ± 6.7	14.7 ± 17.8	**1.4 ± 1.9**
	Jnt. Err (rad.)	0.17 ± 0.07	–	**0.08 ± 0.05**
	SDF Err. (pix.)	6.1 ± 5.3	12.2 ± 22.2	**1.2 ± 0.8**
	Class Err. (%)	11.3 ± 6.2	9.5 ± 6.1	**4.4 ± 1.1**