MATTHEW KLINGENSMITH

THESIS PROPOSAL: ARTICULATED 3D SLAM

Contents

Introduction

To interact with and understand the world, robots must interpret and fuse noisy and ambiguous signals from their sensors. Consider the case of a multi-jointed robot arm with a 2D or 3D camera attached to its end effector (Fig. 1). The robot receives noisy signals from its joint encoders, the hand-mounted camera, and other sensors. How can the robot make sense of the world given these noisy, heterogenous signals? How can it then take action based on this information? To successfully complete a task (such as grasping an object), it is critical that the robot have an accurate geometric representation of the world around it.

Figure 1: The Ada robot with hand-mounted camera manipulating a book in a cluttered bookshelf.

Dense 3D reconstruction is a well-studied problem in computer vision and robotics. Given a series of noisy sensor readings a unknown poses, the task is to infer the true geometric structure and color of the world. This is a variant of the Simultaneous Localization and Mapping (SLAM) problem [1], since it requires simultaneously estimating the state of the sensor and the scene at once. In the context

[1] See page 13

of robotic manipulation, dense 3D reconstruction is useful for scene understanding, planning, grasping, and more.

Typical techniques for solving this problem focus on handheld sensors, and thus assume that little is known about the pose of the sensor aside from weak priors on the magnitude of the motion (though it is sometimes assumed that other sources of motion estimation, such as markers, gyroscopes, or inertial measurement units are available). For this reason, many prior works directly estimate the pose of a handheld sensor using visual (or depth) data alone. Depending on the type of sensor, the lack of additional information can lead to inaccuracies in pose and ultimately in the resulting 3D reconstruction. For instance, using a handheld depth amera, it is impossible to detect motion parallel to a flat plane using the (unchanging) depth data alone [2], without additional sensors or *a. priori* knowledge of how the sensor moves.

[2] See page 14

On the other hand, robot linkages provide extremely strong cues of the sensor's motion, due to their constrained kinematics [3]. As the robot actuates its joints, the sensor moves in a predictable manner. If the joint angles of the robot are perfeclty known, and the robot is perfectly rigid, the pose of the sensor is known with absolute certainty through forward kinematics. In this context, dense 3D reconstruction is a simple mapping problem rather than a full SLAM problem.

[3] See page 9

However, even articulated robotic linkages may have significant noise and systematic inaccuracies [4]. The use of relative joint encoders leads to inaccuracy that compounds over time; and on some systems, joint encoders are seperated from the actuation system by a mechanism with unpredictable dynamics. The *Baxter* (Fig. 2) robot, for instance, has series-elastic actuators and deformable plastic links which significantly increase its kinematic uncertainty. Another robot arm, the *Barret WAM*, is driven by cables. Stretch in the cables results in systematic error that is configuration-dependant. In the extreme case, there may be degrees of freedom that are entirely unknown (such as the pose of the robot's base, or the extrinsic calibration of the sensor). One way of dealing with systematic error is to calibrate the joint encoders to account for unknown dynamics and other sources of systematic error; but if additional unmodeled dynamics are introduced (such as a grasped object, or exterior forces), the calibration will fail to capture them.

[4] See page 21

Figure 2: A camera mounted on the hand of a *Rethink Robotics* Baxter robot.

Uncertainty and unmodeled systematic error can be corrected simultaneously by fusing visual data from the camera with the robot's joint encoder data. If the camera is mounted externally, so that all or part of the robot is visible, the problem becomes the well-studied *articulated tracking* [5] problem. In this case, the robot's known

[5] See page 11

kinematic model can be used in conjunction with visual sensor data to directly estimate the robot's configuration. However, this work is concerned primarily with *hand-mounted* sensors which, for the most part, cannot see the actuator on which they're mounted. In this case, the localization algorithm can't rely on a known model of the world to estimate the robot's configuration. Without this information, is it still possible to estimate the robot's configuration?

This work proposes a method of simultaneously estimating the joint angles of a robot and a dense 3D map of the world around it by solving a constrained visual SLAM problem [6]. Like in the traditional 3D SLAM problem, this method assumes no *a. priori* knowledge of the scene, and will have to take the sensor's pose uncertainty into account. Like the articulated tracking problem, it leverages the known kinematic structure of the robot to strongly inform the sensor's pose.

In some ways, articulated SLAM is easier than the direct 6DOF SLAM problem, because the pose of the sensor is so tightly constrained by the robot's kinematics. However, since the inverse mapping between the sensor's pose and the robot's joint angles is nonlinear and underconstrained, translating 3D SLAM techniques to this domain will require work.

This work includes a brief overview of theory and related works [7], an outline of potential methods of solving the articulated SLAM problem [8], 2D and 3D simulations and experiments [9] involving robots with depth sensors, and an outline of future work [10].

[6] See page 19

[7] See page 9

[8] See page 19

[9] See page 27

[10] See page 31

Background

Consider the use case of a robot arm with an attached 2D or 3D camera. As the robot adctuates its joints, the algorithm must use the (noisy, incomplete) data from the camera in conjunction with the robot's (noisy, biased) joint encoders to simultaneously localize the robot and reconstruct a geometric representation of the robot's workspace. Solving this problem requires an understanding of the robot's kinematics, the mechanics of joint encoders, the characteristics of the sensor, and SLAM techniques.

Robot Kinematics

A *kinematic linkage*[11] 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. For instance, a rotary joint has one degree of freedom that defines a pure rotation around an axis, while a ball joint has two to three degrees of freedom defining a pure rotation around 2 or more axes. The joint which constrains link A to link B, and which has configuration q_i has the transformation:

$$T_B^A(q_i) \in SE(3) \tag{1}$$

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

A kinematic linkage can be represented as an undirected graph where vertexes are links, and the edges are joints between pairs of links. When the graph is acyclic, the linkage is said to be a *kinematic tree*. For instance, a robot with two arms, a head, and a fixed base has a kinematic tree with three branches: one for each arm, and one for the head. Any link may be treated arbitrarily as the root of the tree.

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

[11] Matthew T. Mason . *Mechanics of Robotic Manipulation*. MIT Press, Cambridge, MA, August 2001

the tree (a process called *forward kinematics*):

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

the path from the root to the link in question is called that link's *kinematic chain*. The end of the kinematic chain is called the *end effector*. This work will use the reference frame of the camera as the end of the arm's kinematic chain.

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

$$\mathbf{q} = \begin{bmatrix} q_1 \\ \vdots \\ q_N \end{bmatrix} \tag{3}$$

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

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

is called the link's *kinematic Jacobian*, and for simple kinematic chains it can be computed in closed-form efficiently. The Jacobian is an important object for our problem, because it allows us to map instantaneous changes of the camera's pose to instantaneous changes in the robot's joints.

The robot can also control its joints. Control signals $\mathbf{u} \in \mathbb{R}^N$ are sent to the joints, causing them to move based on the robot's physical dynamics.

Depth Cameras

This work is primarily concerned with two kinds of sensors: 2D cameras and depth cameras. In both cases, the output is a 2D image. Call the 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 [12] model:

$$\text{Proj}(x, y, z) = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{f_x x}{z} + c_x \\ \frac{f_y y}{z} + c_y \end{bmatrix} \tag{5}$$

where u, v 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. The set of all points in 3D space which project onto the camera is called the *field of view* of the camera, and is usually approximated by a truncated pyramid (or *frustum*). We can also define the inverse projection model, which takes a camera coordinate

[12] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004

u, v and depth measurement z, and converts it into a 3D vector relative to the camera's focal point:

$$\text{Proj}^{-1}(u, v, z) = z \begin{bmatrix} \frac{u - c_x}{f_x} \\ \frac{v - c_y}{f_y} \\ 1 \end{bmatrix} \tag{6}$$

Depth cameras measure the range (z) at each image pixel to points in the scene. Projective depth cameras accomplish this by projecting a pattern of infrared light onto the scene and measuring the disparity of this pattern using an offset infrared camera. Time-of-flight cameras measure depth by directly shooting rays of infrared light at the scene and measuring the time it takes for the rays to reflect back to the sensor. In any case, the depth measured by depth cameras is noisy, incomplete, and contains systematic error.

Articulated Tracking

In practice, robots cannot directly measure their configuration, instead they must rely on sensors (called *joint encoders*) which indirectly measure the configuration of their joints. Between the robot's joint encoder and the actual degree of freedom of the joint, there may be mechanisms with unmodeled dynamics (such as non-rigid links, springs, gear trains, or pulley systems) which prevent the robot from knowing its own configuration with certainty. For example, each joint of the *Baxter* robot consists of a motor with plastic gearing, followed by a spring, followed by a rotating mechanism. In the extreme case (*e. g.* human pose tracking or hand tracking), some degrees of freedom may be completely unknown, and can only be inferred through external sensing.

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 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.

When only the kinematic structure of the body is known, but no markers are available, the problem 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, given a series of sensor measurements from timestep 1 to $(t-1)$, and control inputs

$$Z_{1\ldots t} = \{Z_1, \ldots, Z_t\} \tag{7}$$
$$u_{1\ldots t} = \{\mathbf{u}_1, \ldots \mathbf{u}_t\} \tag{8}$$

the task is to estimate the configuration of the body at time t:

$$\mathbf{q}_t = \underset{\mathbf{q}}{\operatorname{argmax}} \, \mathrm{P}\big(\mathbf{q} | Z_{1\ldots t}, \mathbf{q}_{1\ldots(t-1)}, \mathbf{u}_{1\ldots(t-1)}\big) \tag{9}$$

If the system is assumed to have the Markov property (*i. e* the current state is conditionally independent of everything given the prior state), the articulated tracking problem reduces to merely finding the next configuration of the body given its previous configuration and sensor measurement:

$$\mathbf{q}_t = \underset{\mathbf{q}}{\operatorname{argmax}} \, \mathrm{P}\big(\mathbf{q} | Z_{(t-1)}, \mathbf{q}_{(t-1)}, \mathbf{u}_{(t-1)}\big) \tag{10}$$
$$= \underset{\mathbf{q}}{\operatorname{argmax}} \, \frac{\mathrm{P}\big(Z_{(t-1)} | \mathbf{q}, \mathbf{q}_{(t-1)}, \mathbf{u}_{(t-1)}\big) \mathrm{P}\big(\mathbf{q} | \mathbf{q}_{(t-1)}, \mathbf{u}_{(t-1)}\big)}{\mathrm{P}\big(Z_{(t-1)}, \mathbf{q}_{(t-1)}, \mathbf{u}_{(t-1)}\big)} \tag{11}$$
$$= \underset{\mathbf{q}}{\operatorname{argmax}} \left[\log \mathrm{P}\big(Z_{(t-1)} | \mathbf{q}, \mathbf{q}_{(t-1)}, \mathbf{u}_{(t-1)}\big) + \log \mathrm{P}\big(\mathbf{q} | \mathbf{q}_{(t-1)}, \mathbf{u}_{(t-1)}\big) \right] \tag{12}$$

the term $\mathrm{P}\big(Z_{(t-1)} | \mathbf{q}, \mathbf{q}_{(t-1)}, \mathbf{u}_{(t-1)}\big)$ represents the posterior probability that a sensor measurement was observed given some body configuration, while the term $\mathrm{P}\big(\mathbf{q} | \mathbf{q}_{(t-1)}, \mathbf{u}_{(t-1)}\big)$ represents the prior probability of the body's configuration given its previous configuration (called the *motion model*). Articulated tracking approaches differ mainly in how they compute these two quantities, as well as how they maximize them.

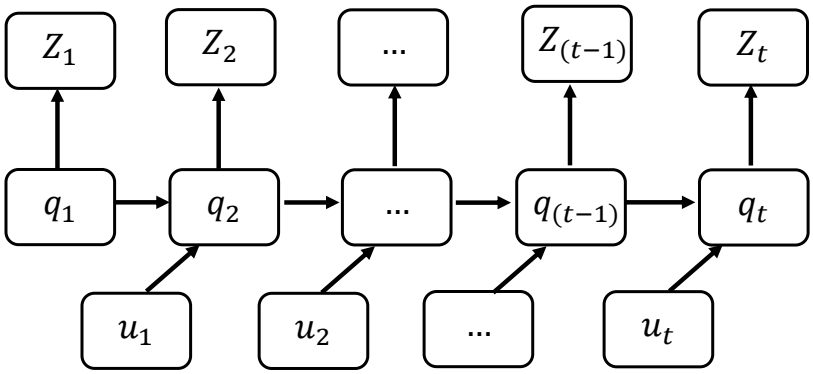


Figure 3: A graphical model of the articulated tracking problem. Notice that the graph has the Markov property.

For instance, one method intended for tracking humans in 2D images, Articulated ICP [13], computes the posterior using image-

[13] L. Mundermann, S. Corazza, and T.P. Andriacchi. Accurately measuring human movement using articulated icp with soft-joint constraints and a repository of articulated models. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–6, June 2007. DOI: 10.1109/CVPR.2007.383302

based edge features, and maximizes the likelihood of a configuration by coordinate descent in configuration space.

One of the author's earlier works, Real-time Markerless Articulated Tracking (RMAT)[14] tracks robots using a depth camera, a simple posterior model that sensor measurements are near the robot's surface as projected onto the depth image, and a motion model which assumes the robot's configuration changes slowly between timesteps. 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.

[14] Matthew Klingensmith, Thomas Galluzzo, Christopher Dellin, Moslem Kazemi, J Andrew Bagnell, and Nancy Pollard. Closed-loop Servoing using Real-time Markerless Arm Tracking. In *International Conference on Robotics And Automation (Humanoids Workshop)*, May 2013

A related but seperate work, Dense Articulated Real-time Tracking (DART) [15], 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.

[15] Tanner Schmidt, Richard Newcombe, and Dieter Fox. DART: Dense Articulated Real-Time Tracking. In *Robotics: Science and Systems*, number 1, 2014. URL http://www.roboticsproceedings.org/rss10/p30.pdf

The articulated tracking problem is a subset of the problem this thesis will address. The requirement that the robot's sensor be able to see its own links greatly reduces the complexity of the problem, turning it into a tracking problem rather than a SLAM problem – but the same techniques used to compute the robot's configuration from sensor measurements can be used even when the scene is unknown.

Simultaneous Localization and Mapping

The Simultaneous Localization and Mapping (SLAM) problem [16] involves concurrently estimating the pose of a moving sensor and the structure of the world around it when both are unknown. That is, given a sequence of noisy sensor readings

[16] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. ISBN 0262201623

$$Z_{1\ldots t} = \{Z_1, Z_2, Z_3, \ldots, Z_t\} \tag{13}$$

estimate the pose of the sensor at all times, $H_{1\ldots t} \in SE(3)$ and the scene M:

$$H_{1..t}, M = \underset{H,M}{\operatorname{argmax}} \operatorname{P}\big(H, M \big| Z_{1\ldots t}\big) \tag{14}$$

Sparse SLAM

Depending on the scene representation and sensor type, different approaches to solving this problem are appropriate. If the measurement model of the sensor consists of a sparse set of features (such as

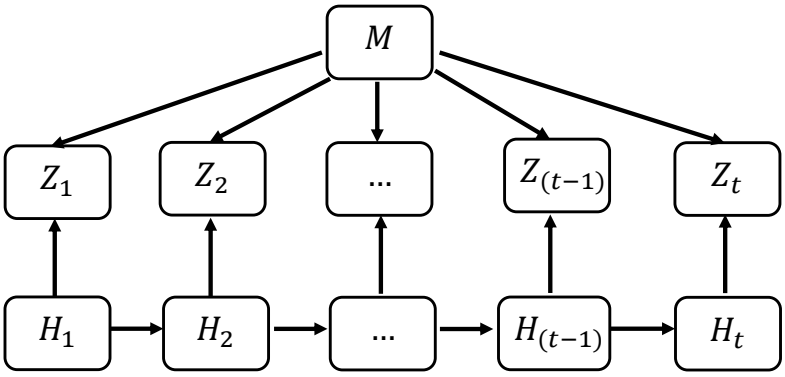


Figure 4: A graphical model of the SLAM problem. Note that given a known map, it has the Markov property, and is just the tracking problem. Without a known map, all measurements are correlated.

SIFT features, corner features, or related), a *sparse* landmark-based scene representation is most appropriate. Sparse SLAM techniques build a globally-consistent graph of tracked landmarks (called a *pose graph*) with pairwise edges on sensor motion between frames; and then optimize this pose graph based on a consistency metric. Since only a very small subset of the sensor data is used, this method can be made to work efficiently over very long datasets featuring loop closures. Specifically, the typical sparse SLAM pose graph has the form:

$$M = \{V, E\} \tag{15}$$

where $V = \{v_1, \ldots, v_N\}$ is a colletion of vertices, with $v_i \in SE(3)$ as either a landmark pose or a sensor pose, and $E = \{e_1, \ldots, e_M\}$ is a collection of directed edges, with each edge defining a transformation $e_i = T_{v_a}^{v_b}$ from one vertex v_a to another vertex v_b. A pose graph is said to be *consistent* whenever each edge is consistent (*i. e*, whenever $v_a e_i = v_b$). A *loop closure* happens whenever two edges share the same outgoing vertex.

This breaks up the graphical model (Fig. 4) so that sensor measurements are only corellated by shared landmarks, rather than all sensor measurements being corellated.

Dense SLAM

However, our use case calls for a robot to localize itself only in a very small workspace, and to use the reconstructed scene for geometric reasoning about objects for the purpose of manipulating them. For this use case, it is much more important to have a detailed geometric representation of the scene than it is to have consistent pose over very long distances.

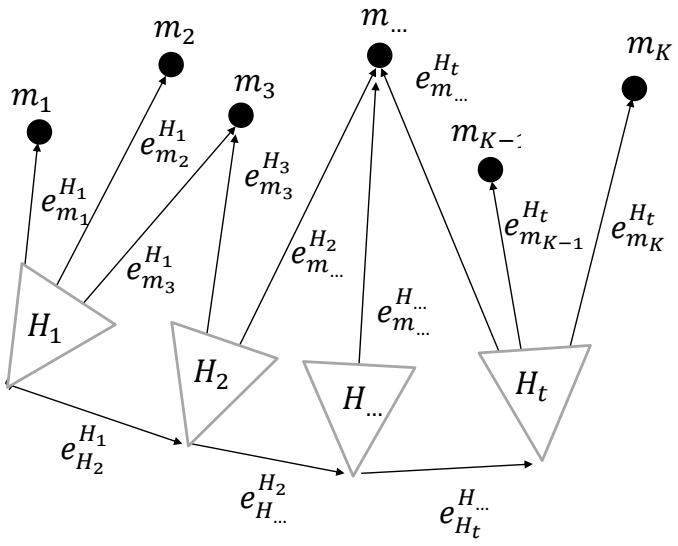


Figure 5: An example of a pose graph.

Dense SLAM techniques, in contrast to sparse SLAM techniques, use all or most of the sensor data to reconstruct a geometric model of the scene. This technique ensures that at all timesteps, a plausible geometric model taking into account all of the sensor data exists – but consequently, the memory required to store the map at all timesteps is much larger than in landmark-based SLAM. Some dense methods (such as *point fusion*[17]) still manage to store a globally consistent pose graph as well as a geometric model by sacrificing some model quality, while most others (such as *kinect fusion*[18] and variants) store the map and pose only for time $t-1$, and then maximize:

$$M_{t-1} = \operatorname*{argmax}_{M} \mathrm{P}\big(M | Z_{t-1}, H_{t-1}\big) \tag{16}$$

$$H_t = \operatorname*{argmax}_{H} \mathrm{P}\big(H | H_{t-1}, M_{t-1}\big) \tag{17}$$

before generating the current model M_t from the previous model M_{t-1} and current pose H_t, and so on. By only storing a single map, these dense techniques avoid having to store dense models for every frame. This simplification ignores the fact that the unknown map correlates all of the sensor readings together (Fig. 4), but results in systemic errors in pose/map estimation building up over time.

A dense map representation of particular interest to our problem is the signed distance field (SDF) [19]. The SDF stores a function that

[17] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb. Real-time 3d reconstruction in dynamic scenes using point-based fusion. In *3D Vision - 3DV 2013, 2013 International Conference on*, pages 1–8, June 2013. DOI: 10.1109/3DV.2013.9

[18] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *UIST 2011*, 2011

[19] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, 1996

gives the distance to the nearest surface in the scene:

$$\Phi(\mathbf{x}) = \min_{\mathbf{s} \in S} \|\mathbf{x} - \mathbf{s}\| \mathbf{I}(\mathbf{s}) \tag{18}$$

where $S \subseteq \mathbb{R}^3$ is the set of all occupied points in the scene, and

$$\mathbf{I}(\mathbf{s}) = \begin{cases} -1 & \text{if } \mathbf{s} \text{ is inside an object} \\ +1 & \text{otherwise} \end{cases} \tag{19}$$

Φ is negative inside objects, positive outside objects, and zero along the surface. The gradient $\nabla\Phi$ can be easily calculated, and the representation very flexible, allowing Φ to represent many kinds of scenes. On the other hand, storing the SDF requires memory on the order of the volume of the scene.

Algorithm 1: FuseTSDF

// Given a depth image, sensor pose, previous SDF, previous voxel weights, a weighting function, a volume of interest, and a truncation distance
Input: $I_D, H_t, \Phi_{t-1}, W_{t-1}, w, V, \tau$
$H' \leftarrow {H_t}^{-1}$
// Initialize the new distance field and voxel weights
$\Phi_t \leftarrow \Phi_{t-1}$
$W_t \leftarrow W_{t-1}$
// For each voxel center in a volume of interest
for $\mathbf{v} \in V$ **do**
    // Transform the voxel to the camera frame and project it on the depth image.
    $\mathbf{v}_h \leftarrow H'\mathbf{v}$
    $d_i \leftarrow I_D\left[\text{Proj}(\mathbf{v}_h)\right]$
    // The geometric depth of the voxel to the camera plane.
    $d_v \leftarrow \mathbf{v}_h(z)$
    // The difference between the geometric depth and the reported depth is a local approximation of the SDF
    $u \leftarrow d_v - d_i$
    // If the local approximation is within the truncation distance...
    if $|u| < \tau$ **then**
        // Compute the weighted average with the previous SDF.
        $\Phi_t(\mathbf{v}) \leftarrow \frac{W_t(\mathbf{v})\Phi_t(\mathbf{v}) + w(u)u}{W_t(\mathbf{v}) + w(u)}$
        $W_t(\mathbf{v}) \leftarrow W_t(\mathbf{v}) + w(u)$
Output: Φ_t, W_t

In practice, the SDF is computed only for a small region around

the surfaces of objects (called the *truncation region*) for which the locally-linear approximation is good. The result is the truncated signed distance field (TSDF[20]) (Alg. 1). TSDF-based dense SLAM techniques (such as *kinect fusion* and variants) build Φ at time t by fusing the depth image produced by the camera at time $t-1$. A depth image can be fused into an SDF by computing a locally-linear approximation of the SDF for the depth image, and averaging it into the SDF computed at $t-1$. Since only one dense model is maintained at each timestep, errors in the model can compound over time.

[20] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, 1996

One of the author's earlier works, Chisel[21], explores techniques for efficiently building and storing the SDF used in dense SLAM, and extends the technique to mobile phones. This thesis will be using Chisel as a backend dense mapping technique.

[21] Matthew Klingensmith, Ivan Dryanovski, Siddhartha Srinivasa, and Jizhong Xiao. Chisel: Real time large scale 3d reconstruction onboard a mobile device. In *RSS*, 2015

Direct Monocular SLAM

For monocular cameras, techniques exist which combine the benefits of sparse and dense SLAM. These techniques (called direct, or direct semi-dense techniques) track the pose of a moving camera by directly minimizing the *photometric error* between subsequent camera frames. This involves directly estimating the depth of pixels in the 2D camera image and reprojecting pixels from one frame onto another, and then maximizing the likelihood of the subsequent camera poses:

$$H_t = \underset{H}{\operatorname{argmin}} \left\| I_{t-1}[Z_{t-1}] - I_t[\text{Proj}(HZ_{t-1})] \right\| \tag{20}$$

where I is the camera image at a given time, and Z_t is the point cloud as inferred by stereo matching between pose H_t and H_{t-1}. Examples of direct techniques include Orb-Slam[22], Lsd-Slam[23], and Dtam [24].

[22] Raúl Mur-Artal and Juan Tardós. Probabilistic semi-dense mapping from highly accurate feature-based monocular slam. In *RSS*, 2015

[23] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *EECV*, 2014

[24] Richard A. Newcombe, Steven J. Lovegrove, and Andrew J. Davison. Dtam: Dense tracking and mapping in real-time. In *ICCV*, 2011

Direct monocular methods, like sparse keypoint methods, are scale ambigous, and cannot distinguish between two scenes that differ only in scale. Like sparse keypoint methods, they generally build a pose graph on top of the dense or semi-dense localization system.

Unlike sparse keyframe-based algorithms, semi-dense direct techniques are capable of creating high-quality, dense 3D reconstructions. Unlike purely dense algorithms, they do not require a depth image to work. These qualities make semi-dense techniques attractive for this thesis.

Framework

Articulated SLAM

When the sensor mounted to the robot is outward-facing, and nothing is known about the world around the robot, how can the robot simultaneously estimate its configuration, and reconstruct the 3D geometry of the world? This is equivalent to the articulated tracking problem with an unknown world model, and also equivalent to the visual SLAM problem with additional constraints on the pose of the sensor.

Given a series of sensor measurements $Z_1, \ldots, Z_t$ and control inputs $\mathbf{u}_1, \ldots, \mathbf{u}_t$, the algorithm must simultaneously estimate the trajectory of the robot's configuration $\mathbf{q}_1, \ldots \mathbf{q}_t$, and a dense model of the world M:

$$\mathbf{q}_1, \ldots, \mathbf{q}_t, M = \underset{\mathbf{q}_\ldots, M}{\operatorname{argmax}} \mathrm{P}\left(\mathbf{q}_\ldots, M | Z_{1\ldots t}, \mathbf{u}_{1\ldots t}\right) \tag{21}$$

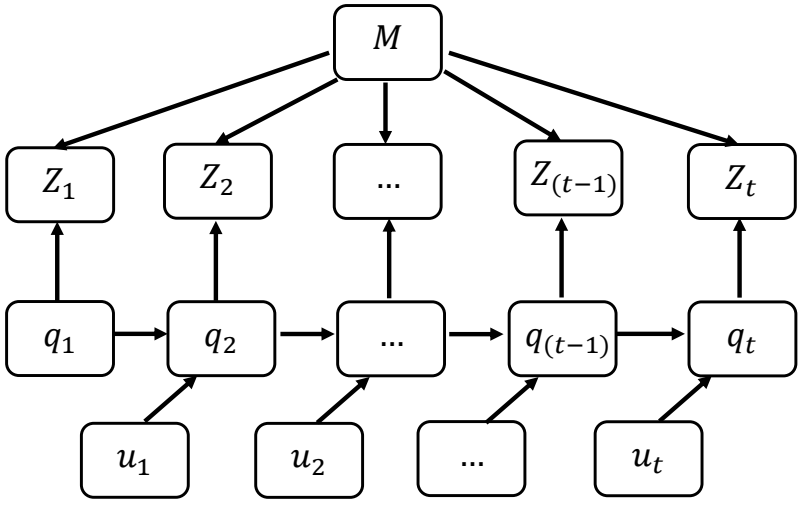


Figure 6: A graphical model of the articulated SLAM problem. It combines aspects of the articulated tracking problem and the SLAM problem..

This problem is made difficult by the fact that the unknown map

M correlates all of the sensor measurements together (Fig. 6). First, consider the articulated tracking problem with a *known* map. In this case, Eq. 21 reduces to

$$\mathbf{q}_1, \ldots, \mathbf{q}_t = \underset{\mathbf{q}_{\ldots}}{\operatorname{argmax}} \mathrm{P}\left(\mathbf{q}_{\ldots} | M, Z_{1\ldots t}, \mathbf{u}_{1\ldots t}\right) \tag{22}$$

This becomes a state estimation problem rather than a full SLAM problem, and therefore acquires the Markov property. So it suffices to estimate the current state given the previous state estimate only:

$$\mathbf{q}_t = \underset{\mathbf{q}}{\operatorname{argmax}} \mathrm{P}\left(\mathbf{q} | \mathbf{q}_{t-1}, \mathbf{u}_{t-1}, M, Z_{1\ldots t}\right) \tag{23}$$

$$= \underset{\mathbf{q}}{\operatorname{argmax}} \frac{\mathrm{P}\left(Z_{1\ldots t} | \mathbf{q}, \mathbf{q}_{t-1}, \mathbf{u}_{t-1}, M\right) \mathrm{P}\left(\mathbf{q} | \mathbf{q}_{t-1}, \mathbf{u}_{t-1}, M\right)}{\mathrm{P}\left(Z_{1\ldots t}, \mathbf{q}_{t-1}, \mathbf{u}_{t-1}, M\right)} \tag{24}$$

$$= \underset{\mathbf{q}}{\operatorname{argmax}} \log \mathrm{P}\left(Z_{1\ldots t} | \mathbf{q}, \mathbf{q}_{t-1}, \mathbf{u}_{t-1}, M\right) + \log \mathrm{P}\left(\mathbf{q} | \mathbf{q}_{t-1}, \mathbf{u}_{t-1}, M\right) \tag{25}$$

$$= \underset{\mathbf{q}}{\operatorname{argmax}} \log \mathrm{P}\left(Z_t | \mathbf{q}, M\right) + \log \mathrm{P}\left(\mathbf{q} | \mathbf{q}_{t-1}, \mathbf{u}_{t-1}\right) \tag{26}$$

yielding two terms which must be maximized: a posterior term $\mathrm{P}\left(Z_t | \mathbf{q}, M\right)$, representing the likelihood of the sensor measurements given a robot configuration and known map, and a prior term $\mathrm{P}\left(\mathbf{q} | \mathbf{q}_{t-1}, \mathbf{u}_{t-1}\right)$, representing the likelihood of a robot configuration given its past configuration and control input.

Prior Motion Model

The prior $\mathrm{P}\left(\mathbf{q} | \mathbf{q}_{t-1}, \mathbf{u}_{t-1}\right)$ depends on the robot's dynamics, and its relationship to gravity and other external forces. In general, the forward dynamics model of a robot arm can be expressed using the well-known differential equation based on its Lagrangian:

$$\tau(\mathbf{u}) = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}, \dot{\mathbf{q}}, F_{\text{ext}}) \tag{27}$$

where τ is the torque the robot produces at its joints given $\mathbf{q}$, $\mathbf{M}$ is a function representing the robot's mass projected into joint space, $\mathbf{C}$ is a function representing the internal corolis forces that the robot experiences, and $\mathbf{G}$ is a function representing how the robot responds to external forces F_{ext} (usually gravity).

If this dynamics model were known perfectly, then the prior motion model would be completely deterministic, and indeed, no external sensors of any kind (not even joint encoders) would be needed to track its state over time. Unfortunately, the dynamics model of a robot manipulator is not easy to calculate, and depends

on having a full account of external and internal forces the robot experiences.

These unmodeled dynamics can be mitigated in two ways: first, by treating them like noise in the motion model, and second, by using external sensors to account for them. At the very least, the dynamic model of the robot provides a narrow region of configuration space that the robot's configuration can be in, given its previous configuration and control input.

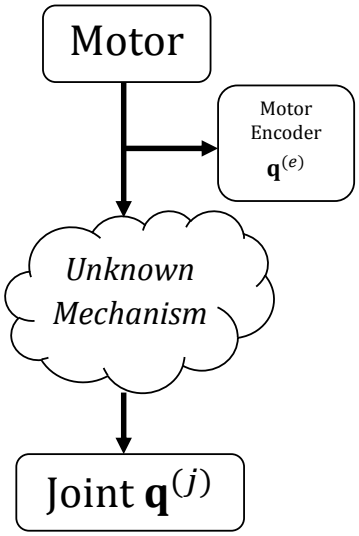


Figure 7: There is always some unknown mechanism between the method of actuation (a motor), and the physical robot degree of freedom.

Motor Encoder Posterior

Robots invariably have sensors (called *encoders*) mounted on their motors and/or joints. These sensors report a value $\mathbf{q}^{(e)}$ that correlates to the actual geometric joint angle $\mathbf{q}^{(j)}$, but, due to the complications of an unknown intervening mechanism (Fig. 7), the encoder's reported value may have noise and systematic error (Fig. 8). For instance, the Barrett *WAM* robot arm is cable-driven. Its' encoders are attached to its' motors, and a complicated system of cables and pulleys transmits torque from the motor to the joint. Unmodeled dynamics from stretch in the cable system leads to a discrepancy between $\mathbf{q}^{(e)}$ and $\mathbf{q}^{(j)}$.

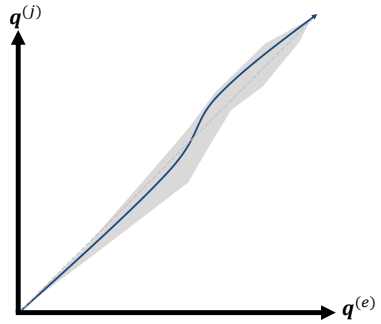


Figure 8: A sketch of a hypothetical relationship between the motor encoders $\mathbf{q}^{(e)}$ and the physical joint angles $\mathbf{q}^{(j)}$, note that the relationship is multivariate, even though it is shown as one-dimensional here.

Define a simple probabalistic model:

$$\mathbf{q}^{(j)} = g(\mathbf{q}^{(e)}) + \epsilon(\mathbf{q}^{(e)}) \tag{28}$$

where $g : \mathbb{R}^N \to \mathbb{R}^N$ is a static calibration function that maps motor encoders to joint angles, and $\epsilon(\mathbf{q}^{(e)})$ is a random variable capturing the variance of the mapping from the motor encoders to joint angles (Fig. 8). This model may be learned by Gaussian Process Regression, or other regression techniques, when ground truth data is known.

Initial tests carried out prior to this proposal on a Barret *WAM* arm using optical joint encoders as ground truth have revealed that the mapping (Eq. 28) between the motor encoders and joint angles depends not only on the configuration of the robot, but also on the applied torque, the path that the robot has taken to get to a particular configuration, and other variables that are not easy to capture using a simple Gaussian model.

Depth Camera Posterior

The sensor posterior term, $\mathrm{P}(Z_t|\mathbf{q}, M)$, represents the likelihood of sensor measurements given a particular robot configuration $\mathbf{q}$ and dense map M. Here, the relevant work from dense 3D SLAM can be taken into account. Consider a depth camera mounted on the

robot's end effector which can see portions of the dense map. The depth data can be modeled as cones, rays, or points. Since each depth pixel measurement is independent given the map and robot configuration, the likelihood of each ray, cone, or point, can be calculated independently.

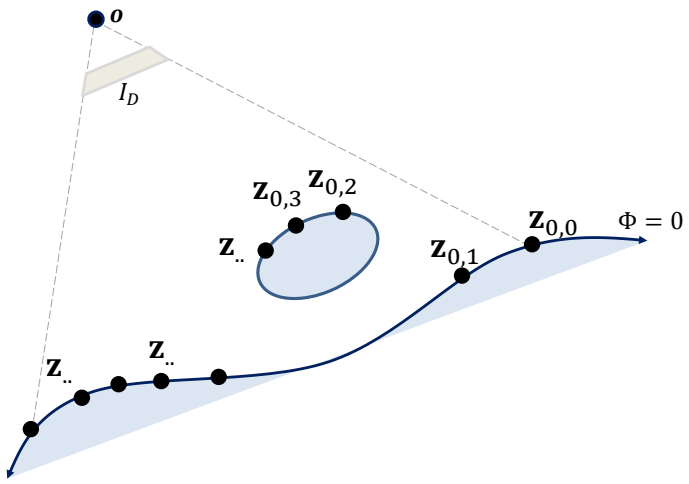


Figure 9: A depth camera with focal point $\mathbf{o}$ and depth image I_D projects points $\mathbf{z}_{u,v}$ into the scene. All the points are expected to lie along surfaces where the signed distance to surfaces in the scene $\Phi = 0$.

Point Cloud Model One of the simplest means of modeling the depth camera is to unproject each depth pixel into the scene as a point using the camera intrinsics. The result is the unstructured *point cloud* of the depth image (Fig. 9). So the sensor measurement is:

$$Z_t = \{H_t \text{ Proj}^{-1}(I_D[u,v]) = \mathbf{z}_{u,v} \; \forall [u,v] \in \Omega\} \tag{29}$$

and due to the conditional independence of the depth image given a robot configuration and map, the posterior is:

$$\text{P}(Z_t|\mathbf{q}, M) = \prod_{u,v \in \Omega} \text{P}(\mathbf{z}_{u,v}|\mathbf{q}, M) \tag{30}$$

Assume that the points in the point cloud are corrupted by simple isotropic Gaussian noise. If this is the case,

$$\text{P}(\mathbf{z}_{u,v}|\mathbf{q}, M) \propto \exp\left(-\Phi(\mathbf{z}_{u,v})^2\right) \tag{31}$$

where Φ is the signed distance field representation of the scene M.

This model, while simple, ignores occlusions. It makes no distinction between points behind obstacles and points in front of obstacles from the perspective of the camera. It also ignores the aniostropic

(direction-dependant) nature of depth sensor noise. A more complicated model may be needed to correctly capture the depth camera posterior.

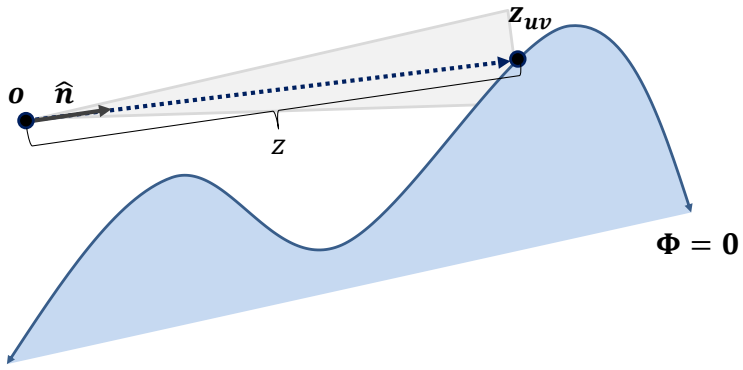


Figure 10: A cone/ray model of the depth camera. The cone emenates from the sensor's focal point $\mathbf{o}$, with direction $\hat{n}$. It hits the surface of the scene at point $\mathbf{z_u v}$, which is distance z along its center.

Ray/Cone Cloud Model The depth camera can be modeled as a set of cones emenating from the camera's focal point – one cone for each depth pixel. The cone is the swept volume of the depth pixel projected into the scene, and has a length equal to the depth pixel's measurement (Fig. 10). The cone may be seen as the union of all rays that pass through the depth pixel from the focal point of the camera.

This is a much more complex model, incorporating information about which points in the scene have been passed through by rays as well as which points lie on surfaces (as well as their surface normal).

Dense Sensor-To-Map Constrained SLAM

Given a model of the sensor posterior and motion model, and a dense signed distance field representation of the map, it is possible to formulate a simple dense [25] sensor-to-map algorithm for articulated SLAM. Using the point cloud posterior model [26], and a simple motion model which just assumes that the difference between the robot's configuration and the reported motor encoder readings [27] $\|\mathbf{q}_t - \mathbf{q}_t^e\|$ is small, the problem can be treated as a simple optimization problem at each timestep.

[25] See page 14

[26] See page 22

[27] See page 20

$$\mathbf{q}_t \leftarrow \underset{\mathbf{q}}{\operatorname{argmax}} \, \mathrm{P}\big(\mathbf{q} | Z_t, M_{t-1}, \mathbf{q}_{t-1}\big) \tag{32}$$

$$M_t \leftarrow \underset{M}{\operatorname{argmax}} \, \mathrm{P}\big(M | \mathbf{q}_t, Z_t, M_{t-1}\big) \tag{33}$$

The configuration update (Eq. 33) can be maximized by breaking it into its component parts

$$\underset{\mathbf{q}}{\operatorname{argmax}} \, \mathrm{P}\big(\mathbf{q} | Z_t, M_{t-1}, \mathbf{q}_{t-1}\big) = \underset{\mathbf{q}}{\operatorname{argmax}} \log \mathrm{P}\big(Z_t | \mathbf{q}, M_{t-1}\big) + \log \mathrm{P}\big(\mathbf{q} | \mathbf{q}_{t-1}\big) \tag{34}$$

and the sensor posterior from the point cloud model can be computed as:

$$\log \mathrm{P}\big(Z_t | \mathbf{q}, M_{t-1}\big) = \log \prod_{[u,v] \in \Omega} \mathrm{P}(\mathbf{z}_{u,v} | \mathbf{q}, M_{t-1}) \tag{35}$$

$$\propto \sum_{[u,v] \in \Omega} \log \exp\left(-\Phi(\mathbf{z}_{u,v})^2\right) \tag{36}$$

$$= \sum_{[u,v] \in \Omega} -\Phi(\mathbf{z}_{u,v})^2 \tag{37}$$

where $\mathbf{z}_{u,v} = H_t \, \mathrm{Proj}^{-1}\left(I_D[u,v]\right)$ is a point from the point cloud of the depth image at time t, and Φ is the signed distance field of the map at time $t-1$.

So, maximizing the sensor posterior is equivalent to minimizing the simple cost function:

$$c(\mathbf{q}) = \frac{1}{2} \sum_{[u,v] \in \Omega} \Phi(\mathbf{z}_{u,v})^2 \tag{38}$$

and c is minimized whenever $\frac{\partial c}{\partial \mathbf{q}} = 0$. That is:

$$\nabla c = \frac{1}{2} \sum_{[u,v] \in \Omega} \left[\frac{\partial}{\partial \mathbf{q}} \Phi(\mathbf{z}_{u,v})^2\right] \tag{39}$$

$$= \sum_{[u,v] \in \Omega} \left[\Phi(\mathbf{z}_{u,v}) \frac{\partial}{\partial \mathbf{q}} \nabla \Phi(\mathbf{z}_{u,v})\right] \tag{40}$$

$$= \sum_{[u,v] \in \Omega} \left[\Phi(\mathbf{z}_{u,v}) \mathbf{J}(\mathbf{q}, \mathbf{z}_{u,v})^T \nabla \Phi(\mathbf{z}_{u,v})\right] \tag{41}$$

where $\mathbf{J}(\mathbf{q}, \mathbf{z}_{u,v}) \in \mathbb{R}^{3 \times N}$ is the manipulator's kinematic jacobian[28] evaluated for the point $\mathbf{z}_{u,v}$ as rigidly attached to the end effector. The gradient of the SDF $\nabla \Phi$ is easy to approximate using finite differencing.

[28] See page 9

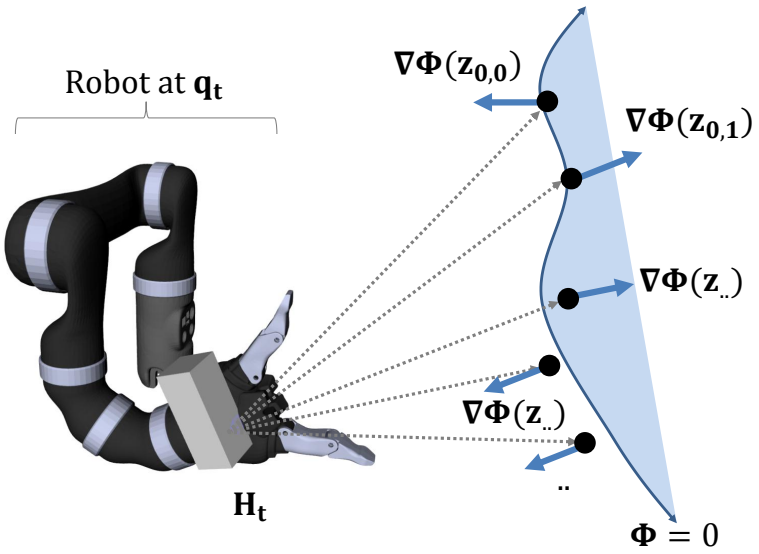


Figure 11: A rendering of a robot manipulator with attached depth sensor. Points $\mathbf{z}_{u,v}$ are projected into the scene, which is compared with the map's distance field from the previous timestep Φ. The gradient of Φ at each point in the point cloud is shown as an arrow. The end effector transform at time t is given by H_t, the robot's configuration at that time is $\mathbf{q}_t$.

This cost function and gradient has an interesting physical interpretation (Fig. 11). If all of the points in the point cloud are interpereted as being rigidly attached to the camera frame, and a force is applied at each of the points $\mathbf{z}_{u,v}$ with a force in the direction $\nabla\Phi(\mathbf{z}_{u,v})$ with magnitude $\Phi(\mathbf{z}_{u,v})$, the gradient ∇c is the same as the instantaneous acceleration the joints would experience from the sum of such forces.

The likelihood of $\mathbf{q}_t$ can be maximized using gradient descent of the cost function c regularized by the motor encoders (Alg. 2). In this way, the robot's configuration stays near what is reported by its motor encoders, but is allowed to change in response to the sensor data.

Algorithm 2: Regularized Dense SDF Gradient Descent.

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

Input: $Z_t, \mathbf{q}_t^{(e)}, \mathbf{q}_{t-1}, \Phi_{t-1}, \lambda, \gamma$

// Initialize configuration offset $\epsilon \in \mathbb{R}^N$ to zero.

$\epsilon \leftarrow \mathbf{0}$

repeat

  // Current estimate for $\mathbf{q}_t$ is the joint encoders plus the offset

  $\mathbf{q}_t \leftarrow \mathbf{q}_t^{(e)} + \epsilon$

  // Compute the gradient of the sensor measurement posterior

  $\nabla c(\mathbf{q}_t) \leftarrow \sum_{[u,v]\in\Omega} \left[\Phi(\mathbf{z}_{u,v})\mathbf{J}(\mathbf{q}_t, \mathbf{z}_{u,v})^T \nabla\Phi(\mathbf{z}_{u,v})\right]$

  // Descend the gradient regularized by γ

  $\epsilon \leftarrow \epsilon - \lambda\nabla c(\mathbf{q}_t) - \gamma\epsilon$

until *convergence*;

// Now that there is a new estimate of $\mathbf{q}_t$, fuze the sensor data into the SDF.

$\Phi_t \leftarrow \textsc{FuseTSDF}(Z_t, F(\mathbf{q}_t), \Phi_{t-1}, \ldots)$

Output: $\mathbf{q}_t, \Phi_t$

Experiments

2D Simulation Experiments

The constrained descent algorithm (Alg. 2) can be illustrated by constructing a simple 2D simulation (Table 13). In the simulation, a 3-link serial robot manipulator with a simulated 1D depth sensor scans a scene. Zero-centered Perlin [29] noise is added to its joint encoder readings. That is,

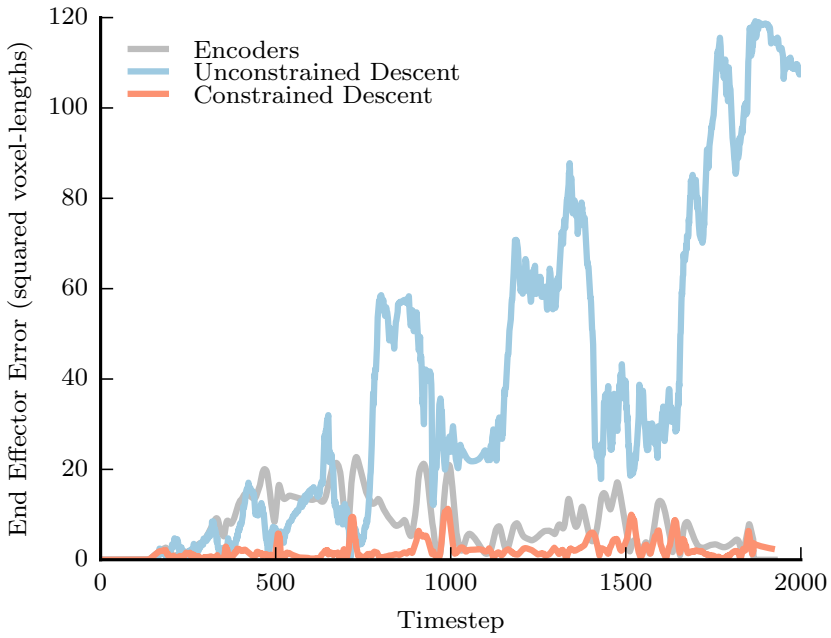
$$\mathbf{q}_t^{(e)} = \beta_n \textsc{Perlin}\left(s_n \mathbf{q}_t^{(j)}\right) \tag{42}$$

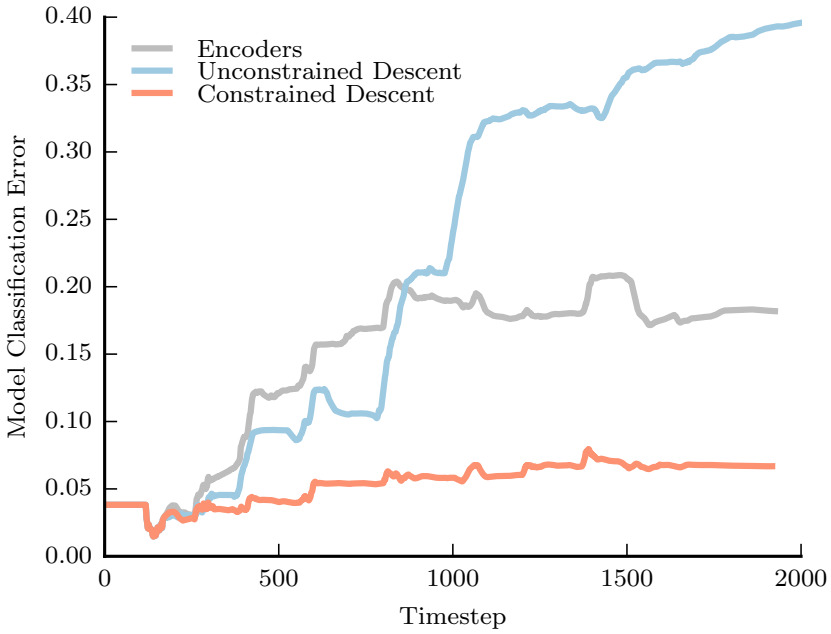
where s_n is a noise scale parameter, and β_n is a noise magnitude parameter.

For the world model, a simple 2D truncated signed distance field (TSDF) is constructed. The performance of the constrained gradient descent algorithm (Alg. 2) is compared against a simple unconstrained descent algorithm which assumes the sensor can move and rotate freely, without considering the robot kinematics.

The robot scans the scene, and makes a single loop closure. When using the (noisy) joint encoders for pose only, much of the detail of the scene is lost, since the TSDF (Φ) smooths out errors through averaging. Constrained gradient descent allows the robot to correct its joint encoder error and recapture fine details in the scene (though drift is still made apparent by the loop closure). Unconstrained gradient descent, on the other hand, causes the robot's estimate of the sensor pose to drift; and the 3D reconstruction is quickly destroyed as error accumulates (Fig. 12).

[29] Ken Perlin. Improving noise. In *SIGGRAPH*, 2002





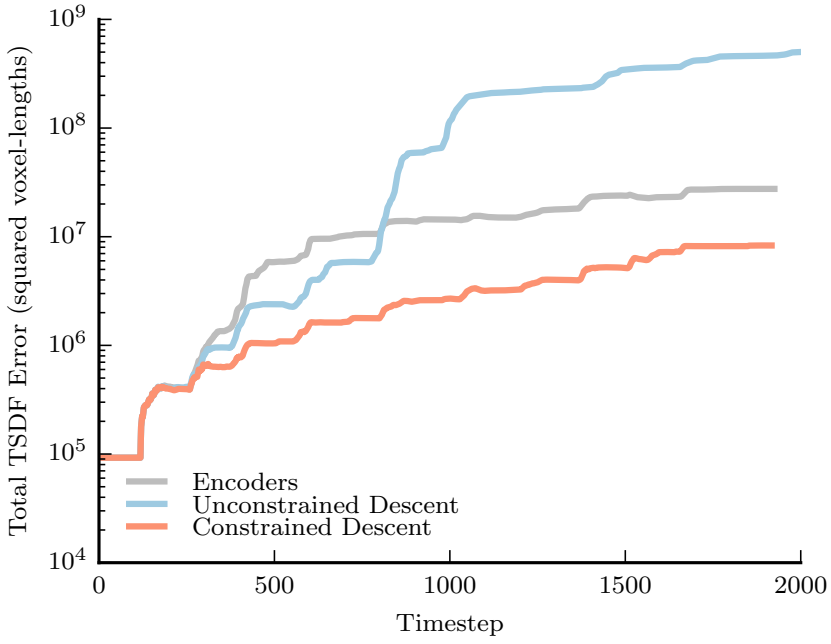


Figure 12: 2D experiment (Table 13) data. Here, end effector error is the squared distance between the reported end effector position and ground truth. The model classification error is the proportion of TSDF voxels that are misclassified (full vs. empty), and the total TSDF error is the total squared difference between the ground truth distances and the reported distances.

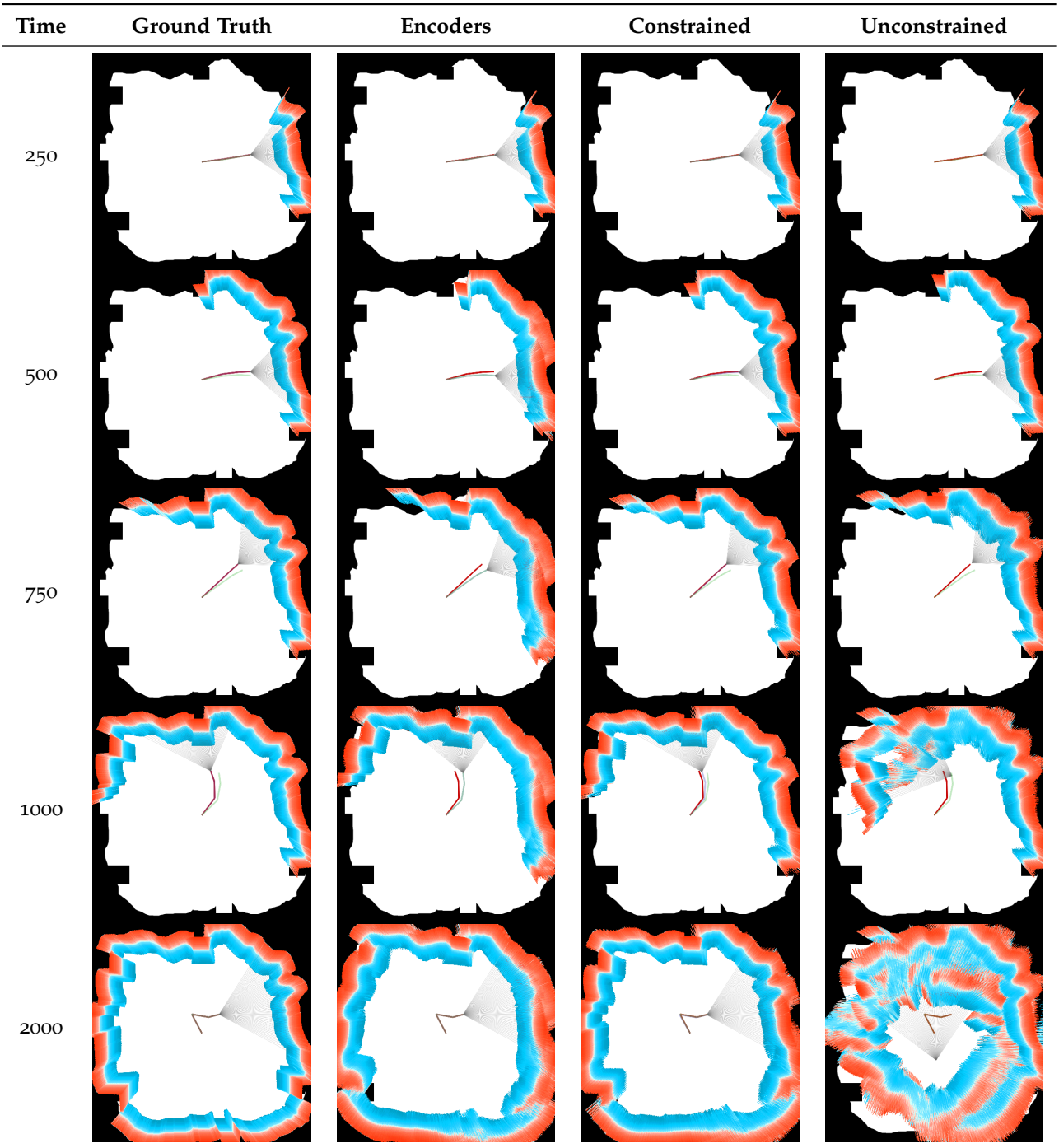


Table 13 : 2D experiments. A 3-joint robot arm (dark red) scans the scene with a simulated depth camera (light grey lines). Its joint encoders are corrupted by noise (green). It builds up a TSDF, Φ (red and blue pixels) of the scene (black pixels). The ground truth is compared to just using the robot encoders, constrained descent (Alg. 2), and unconstrained descent over time.

3D Simulation Experiments

The same concept can be extended to 3D scenes (Fig. 13). We built a framework for doing dense sensor-to-model articulated tracking (Alg. 2) in 3D, as well as a depth sensor simulator. Like in the 2D experiments, the robot follows a fixed trajectory. Perlin noise is added to the joint encoders. The robot builds a 3D TSDF using the simulated depth data (Fig. 15). Fig. 16 shows the result of the same robot trajectory and depth data being used to reconstruct a bookshelf without any joint encoder noise, with noise and dense articulated 3D tracking, and without any tracking. Even under conditions of extreme encoder noise, dense frame-to-model SLAM is able to create a high-qaulity 3D reconstructions of the scene. On the other hand, using the noisy encoders alone, the 3D reconstruction is almost totally unrecognizable.

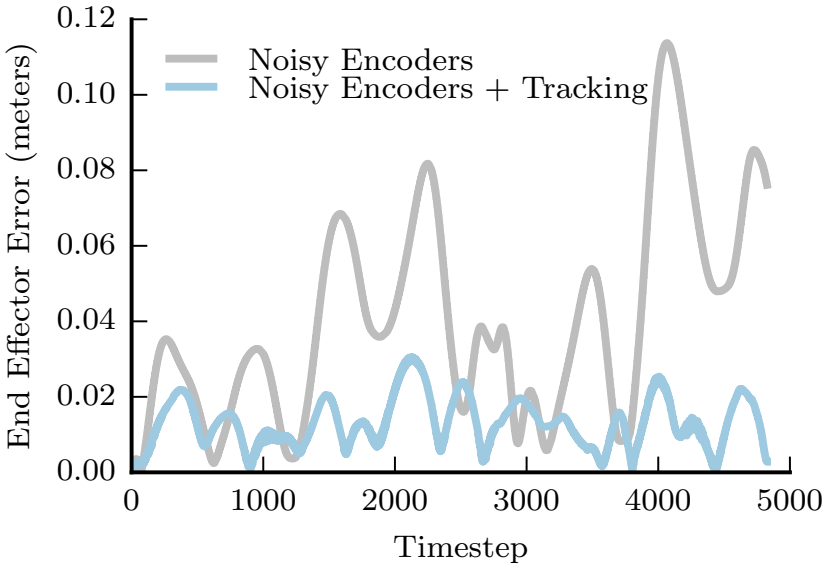


Figure 14: The end effector error in the 3D simulation experiment.

Fig. 14 shows the translation error of the end effector vs. ground truth during tracking. The end effector error consistently converges to be beneath 2 cm even when the error due to noisy joint encoders is more than 11 cm.

The deficiencies of the simple gradient descent algorithm can be seen around timestep 500, 3000, etc. where the noisy encoders have better end effector accuracy than the tracking. This is because the prior motion model is too simple to account for high-frequency noise.

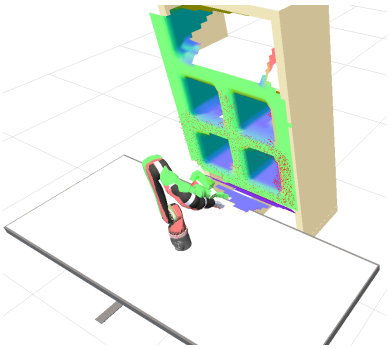


Figure 13: A 3D simulation environment featuring a robot mounted to a table, looking at a bookshelf.

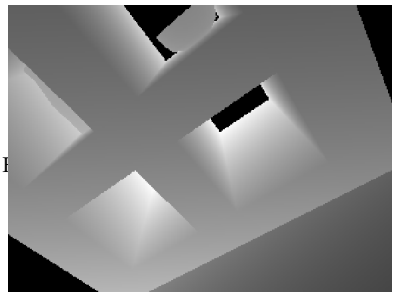
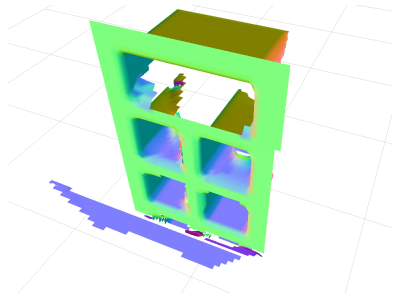
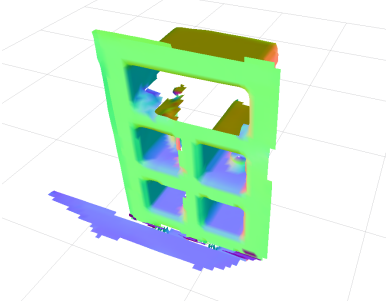


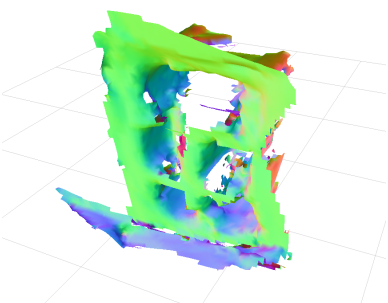
Figure 15: Simulated depth camera.



Ground Truth



Noisy Encoders + Tracking



Noisy Encoders Only

Figure 16: Simulated 3D TSDF reconstructions.

Research Questions

This thesis proposal has introduced a novel framework for doing 3D visual SLAM in articulated systems; but much further research is needed to investigate the theoretical grounding and concrete properties of this framework. Here are a few questions which must be answered by the thesis:

What applications are enabled by articulated SLAM?

Building high quality 3D reconstructions is one thing, but what can be done with the output of the articulated SLAM system? One possibility is to use the model to plan manipulation strategies (such as grasping and reaching into tight spaces), or to control the arm (*i. e* visual servoing). When the robot is no longer passively observing the model of the world, but instead can control where it places the sensor, the whole field of active perception opens up. How should the robot move to best reconstruct the scene?

As interesting as these questions are, they will have to be secondary to the fundamental question of how to get the SLAM system to work in the first place.

How can the encoder noise be modeled on real robot arms?

So far, we have only considered vague properties of encoder noise [30] – that it is really unmodeled dynamics causing the discrepency between joint encoders and the actual joint angles, and that it is likely configuration-dependant. In simulated experiments, we have modeled the noise using a random Perlin function [31]. But what exactly *are* the unmodeled dynamics we are trying to capture? What physical processes cause them, and how can these physical processes be modeled?

[30] See page 21

[31] See page 27

Answering this question will require a literature review of existing methods, data collection on real robots, and a framework for testing different dynamics models.

What are the limitations of dense camera-to-model SLAM using real data/robots?

So far, our experiments have only covered 2D and 3D simulated datasets. Obviously, real robots have unique challenges not addressed by the simulations: extrinsic/intrinsic calibration error, sensor noise, unmodeled dynamics, etc. We are eager to test articulated SLAM on a real robot. Experiments must be devised to test the convergence basin of the SLAM system. One particularly challenging component will be getting ground-truth data for the pose of the robot and geometric structure of the world (which are easy to get in simulation).

How can a pose graph be incorporated into the articulated SLAM framework?

So far, we have only derived dense camera-to-model 3D SLAM [32]. This approach works well for small workspaces when the displacement between camera poses is small, but fails over long trajectories and large workspaces. The camera-to-keyframe or pose graph methods [33] correct for drift by globally optimizing the entire trajectory with respect to every keyframe. Incorporating a pose graph into the articulated SLAM problem would be an ideal way of correcting for drift.

[32] See page 23

[33] See page 13

Answering this question will require re-deriving pose graph optimization in terms of *robot configurations* rather than *camera poses*.

How can direct monocular articulated SLAM be accomplished?

So far, we have only discussed solutions for dense camera-to-model 3D SLAM, which requires knowledge of a depth image at all time steps. How can we reconstruct a scene and estimate the robot's configuration when only a *monocular* camera is available? Direct or semi-direct [34] SLAM is an attractive set of solutions for 3D reconstruction using monocular cameras. Such methods rely on direct optimization in $SE(3)$ of camera poses given reprojected pixel intensities.

[34] See page 17

Answering this question will require constructing a direct image residual optimization technique in the configuration space of the robot, instead of in $SE(3)$. That is, we will have to derive the direct relationship between *pixel intensities* and *joint angle displacements*.

Why articulated SLAM?

This is perhaps the biggest research question. What advantages does an articulated SLAM system have over a traditional unconstrained

6DOF SLAM system? We believe that doing SLAM directly in the natural space of the robot arm (its configuration space) is a more accurate and theoretically grounded way of solving the problem – but data will have to be collected to verify this, comparing existing SLAM systems with their articulated counterparts.

Timeline

I want to complete this thesis work in one year. I will be focusing my efforts on two major conference papers: one for ICRA 2016, and one for IROS 2016. Other conferences may be selected depending on how my research is going.

Date		Conference
July 11, 2015	Present CHISEL at RSS.	RSS 2015
August 2016	Real experiments with dense frame-to-model tracking.	
Sep 15, 2015	Write paper on dense frame-to-model tracking.	ICRA 2016
Fall 2015	Begin implementing direct monocular system.	
Mar 1, 2016	Write paper on direct monocular system.	IROS 2016
Spring 2016	Implement pose graph.	
June 8, 2016	Finish and defend this thesis	

The first paper will focus on dense camera-to-model SLAM [35], and will theoretical ground it, and provide results against other (non-articulated) SLAM algorithms. The second paper will focus on a direct monocular method [36]. Finally, I will look at incorporating a pose graph [37]. Work from these conference papers will be incorporated into the thesis.

[35] See page 23

[36] See page 17

[37] See page 13

Bibliography

Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, 1996.

J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *EECV*, 2014.

R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *UIST 2011*, 2011.

M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb. Real-time 3d reconstruction in dynamic scenes using point-based fusion. In *3D Vision - 3DV 2013, 2013 International Conference on*, pages 1–8, June 2013. DOI: 10.1109/3DV.2013.9.

Matthew Klingensmith, Thomas Galluzzo, Christopher Dellin, Moslem Kazemi, J Andrew Bagnell, and Nancy Pollard. Closed-loop Servoing using Real-time Markerless Arm Tracking. In *International Conference on Robotics And Automation (Humanoids Workshop)*, May 2013.

Matthew Klingensmith, Ivan Dryanovski, Siddhartha Srinivasa, and Jizhong Xiao. Chisel: Real time large scale 3d reconstruction onboard a mobile device. In *RSS*, 2015.

Matthew T. Mason . *Mechanics of Robotic Manipulation*. MIT Press, Cambridge, MA, August 2001.

L. Mundermann, S. Corazza, and T.P. Andriacchi. Accurately measuring human movement using articulated icp with soft-joint

constraints and a repository of articulated models. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–6, June 2007. DOI: 10.1109/CVPR.2007.383302.

Raúl Mur-Artal and Juan Tardós. Probabilistic semi-dense mapping from highly accurate feature-based monocular slam. In *RSS*, 2015.

Richard A. Newcombe, Steven J. Lovegrove, and Andrew J. Davison. Dtam: Dense tracking and mapping in real-time. In *ICCV*, 2011.

Ken Perlin. Improving noise. In *SIGGRAPH*, 2002.

Tanner Schmidt, Richard Newcombe, and Dieter Fox. DART: Dense Articulated Real-Time Tracking. In *Robotics: Science and Systems*, number 1, 2014. URL http://www.roboticsproceedings.org/rss10/p30.pdf.

Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. ISBN 0262201623.