Uncertainity Adapatation in Robot Perception and Learning

Jimmy Jin

December 2017

School of Computer Science
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Master of Science.

Keywords: Robotics, Computer Vision, Reinforcement Learning

I would like to dedicate this thesis to my loving parents Dejiang Jin and Xiaoying Zhu.

Abstract

Dealing with uncertainty is a fundamental challenge for building any practical robot platform. In fact, the ability to adapt and react to uncertain scenarios is an essential sign of an intelligent agent. Furthermore, uncertainty can arise from every component of a robotic system. Inaccurate motion models, sensory noises, and even human factors are all common sources of the unexpected. From an algorithmic perspective, handling uncertainty in robotics introduces a new layer of difficulty because the algorithm not only needs to be accurate in a single scenario but also need to adapt to the changes in uncertainties as the environment shifts. This thesis presents methods for adapting to uncertainties in two tasks: object pose estimation and assistive navigation.

For object pose estimation, we present a sensor fusion method that is highly robust in estimating the pose of fiducial tags. The method leverages the different structural and sensory advantages of RGB and Depth sensors to joint-optimize the Perspective-N-Point problem and obtains the pose. The key insight being adaptively bounding the optimization region by testing the pose solution uncertainty.

For assistive navigation, we wish to tackle the problem of using active signaling to avoid pedestrians while it is minimally invasive to other people. We formulate the problem as a bandit with expert advice problem with reinforcement learning policies as the experts. We present an online learning algorithm which can continuously adapt to new and uncertain pedestrian types by using an online policy search technique and the Dirichlet Process.

Acknowledgments

First and foremost, I would like to thank my advisors Kris Kitani and Siddhartha Srinivasa. You are fantastic mentors, teachers, and sources of inspiration. I cannot express my gratitude enough for giving me the oppertunities to learn from the best over my years at CMU. From being a sophomore four years ago to today, I would not have gained the insights and passions for robotics that I have without these valueable oppertunities and your gauidance every step of the way. Thank you for being understanding and always pushing me to be a more independent thinker.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Uncertainity is a fundemental problem for any robots that intend to perform intellgently in the real world. At its core, uncertainity captures the essence of our ever-changing world and its underlying latent states. In practice, uncertainity arises from almost every part of the robotic system such as noisy sensors, poor localization, and even inputs from surrounding human users. Many of these challgnes have been well studied in different areas of robotics including manipulation, mobile robots, aerial robots, and human-robot interactions.

From an algorithmitic point of view, the challenge of designing algorithms dealing with uncertainity is that we cannot make strong assumptions about the uniformity of its inputs. With the case of classical deterministic algorithm, there is a deterministic mapping from inputs to correct outputs. The mapping can be arbitrarly complicated or difficult to compute but it remains static over time. In other words, all the necessary information are provided as inputs to the algorithm. The accuracy of the algorithm can be objectively measured by verifying against the groundtruth. However, we have to relax this assumption for the inputs under uncertainity. In facts uncertain inputs can have multiple correct answers based on some latent state of the world which can't be captured as part of the input. Furthermore, uncertain inputs are everywhere in robotics. For instances, consectuive images taken from the same camera in a static scene are often not the same due to randomness in lighting variations and the amount of photons captured by each pixel

Figure 1.1: Robot uncertainity

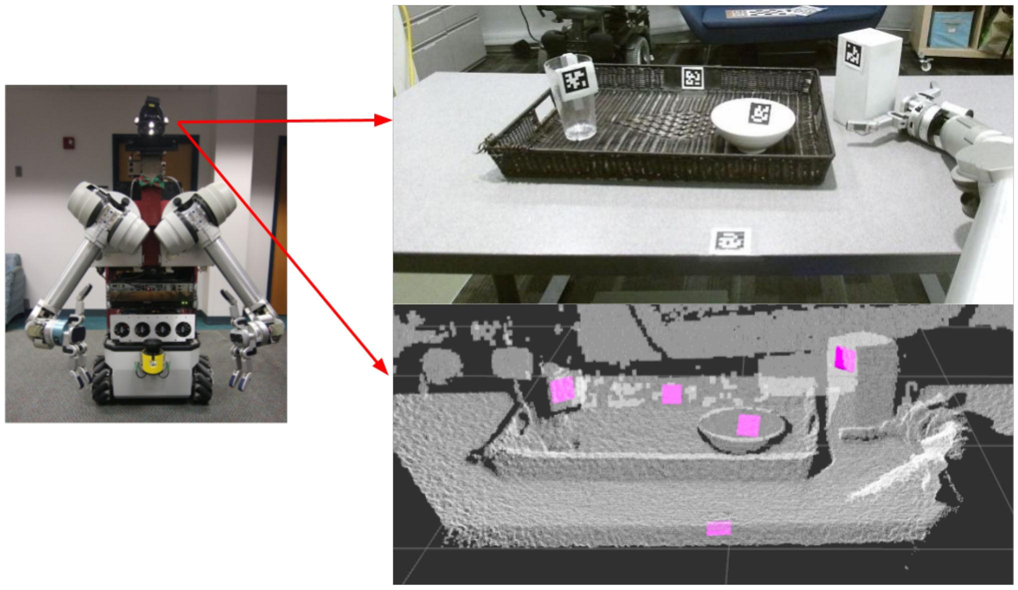


Figure 1.2: Robot about to execute a manipulation task and rearrange the objects on the table. Apriltags are used to find the poses of targeted objects in the scene but the robot ultimately fails to gasp the rectangular prism because the orientation of its pose is wrong.

during the camera exposure. The same person might react differently to the same set of actions depending on his or her mood. Therefore, uncertain inputs are often thought of as samples from a probabilistic distribution and the quality of the algorithm is measured by repeating the algorithms over many trials.

We will address two specific task common in robotic applications and show that by leverging the idea of adpative weighting, we can imporve the performance of these tasks even under uncertainity.

1.1 Object Pose Estimation

The first task we will address is robust pose estimation for table top objects. This has been a diffcult problem due to the size of the objects and precision requirements in large robotic systems such as HERB as shown in Fig 1.2. In particular we will use fiducial markers. Detection and identification using artificial landmarks, known as fiducial markers, has long been used in augmented reality (AR) and computer vision (CV) applications. Over the last decade, there have been numerous marker systems, such as ARTags [**?**] Apriltags [**?**], and Rune Tags [**?**], designed to improve detection encoding precision. In contrast to AR systems, robots often operate in suboptimal conditions where, for instance, camera resolution and illumination are constrained and cause the data to be noisy. In order for fiducial-marker systems to be effective in these settings, they must be robustness to scenery and sensory noises.

There are two qualities of fiducial-marker systems that are especially important to robotic applications: detection rate, the ability to find the tag in the image, and pose accuracy, the accuracy of the estimated 6 DOF pose of the tag. Compared to markerless detection algorithms, fiducial-marker methods are simpler. They yield great results in augmented reality tasks that require high detection speed. Furthermore, the fiducial tags are popular in the robotic community due to their high detection rates and numerous encoding schemes. For example, Apriltags are commonly used to test SLAM systems, or finding ground truth for objects in manipulation and motion planning tasks.

However, obtaining highly accurate pose estimations using fiducial tags from noisy data remains a challenge. This is important for robotic applications because small errors can cause large system failures as the errors propagate and amplify through the system as shown in Figure 1.2. Currently, the fiducial tag systems yield promising results under well conditioned or rendered environments, but this does not translate to ill-conditioned settings. For instance, when AprilTags, a state of the art fiducial marker, are used with low resolution cameras or harsh lighting conditions, the system often produces poses with tremendous rotational errors. We observe that the AprilTag's localization accuracy performs significantly worse when there is noise in the captured image. This is a difficult problem because RGB sensors are often sensitive to lighting, and most popular fiducial systems are not designed to take advantage of other sensors commonly available on robots.

1.2 Blind Navigation

The second task we will address involves learning navigation strategies for the blind assistive robot Cabot. Recently, much work has been done in observing human intent in the area of assistive robotics. This is an interesting field of study because these systems are generally passive - robots are often designed to observe other human behaviors and adjust its actions according to its predictions for better avoidance. In this paper, we instead present a way for robots to actively change other people's course of actions appropriately in order to help the user better. Specifically, we want to come up with a way to actively manipulate the trajectories of pedestrians so that pedestrians do not unintentionally run into mobile robots that operate near people from inattentiveness.

The motivation behind developing such a strategy is clear for robots that interact extensively with humans. In our case, we have a robot designed to navigate the visually impaired in open areas like hotels, malls or airports. This robot must have a strategy to avoid colliding with pedestrians for both its safety and its user's safety. However, avoiding collisions is sometimes impossible when moving around tight spaces or in crowded environments without actively changing pedestrian behavior. A viable solution for this robot can be to play a sound. While the playing this sound continuously would guarantee no collisions, this method is invasive to the environment and energy inefficient. Instead, we want to learn a way to play the sound only when absolutely necessary, that will hopefully be robust enough to take into account the wide variety of possible pedestrian behaviors.

We can represent this robot as a robotic guide dog. Our objective is for the robotic guide dog to learn when it is appropriate to bark. Similarly, police officers inherently use a learned heuristic when deciding whether to simply flash their police cars' emergency lights, or to sound its sirens.

Police cars can't always use its sirens since sirens are too loud and will disrupt the public, but sometimes it is necessary when chasing a dangerous suspect. We want to formally develop a way for robots to learn ways to avoid pedestrians, just like how police officers learn to use their car's sirens strategically.

We view this problem from a reinforcement learning perspective, where we want to learn the sequential decision making actions that maximize our goal reward. Furthermore, this is also an online learning problem for we want to continuously adapt and improve our pedestrian manipulation strategies over time and unseen environment types. Again, since pedestrians are all different in the way they interact with other pedestrians, we want to come up with a strategy that is robust to variations in human behavior. This is especially suitable for assistive technologies because their targeted tasks are often changing and involving over the life time of the user.

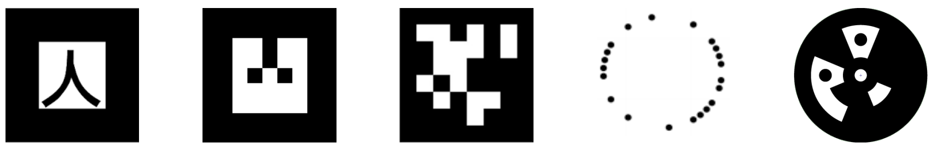
Chapter 2

Pose Estimation Background

Obtaining highly accurate pose estimation has been an important research area in robotics. Numerous algorithms rely only on RGB or gray scale images. Solving the projection geometry of some detected features and then minimize the reprojection error of the features in the image space [**?**]. Similarly, methods such as Iterative Closest Point [**?**] were developed to solve the pose estimation problem using range data by minimizing the Euclidean distance between the model and the depth data. Recently, some approaches in the SLAM community propose to enhance the accuracy of traditional tracking algorithms by fusing RGB with depth data or inertial data in various problems using extended Kalman filters [**? ?**]. Compared to the single-sensor approaches, algorithms utilizing RGBD data are more accurate and perform well in noisy situations where other approaches fail. However, such approaches are often costly in terms of physical hardware as well as computation overhead. It is difficult to apply them in time sensitive applications.

Fiducial markers solve pose estimation by exploiting easily detectable features in the RGB space. There is an abundance of unique tag designs, most of them carry easily recognizable yet precise binary patterns in the inner region to encode information. There are two types of common tags: circular tags and square tags (see Figure 2.1).

Circular tags are created to encode the payload using small circular patterns arranged in various shapes. Examples of circular tags include Intersense [**?**] and Rune tags [**?**]. The perspective transformation of a circle is an ellipse, which can be used to directly compute the pose



(a) ARToolkit (b) ARTag (c) AprilTag (d) RUNE-Tag (e) Intersense

Figure 2.1: Different types of popular fiducial tags. ARToolkit, ARTags, and AprilTags are square tags with black borders. RUNE-tags and Intersense use different circle features as landmarks

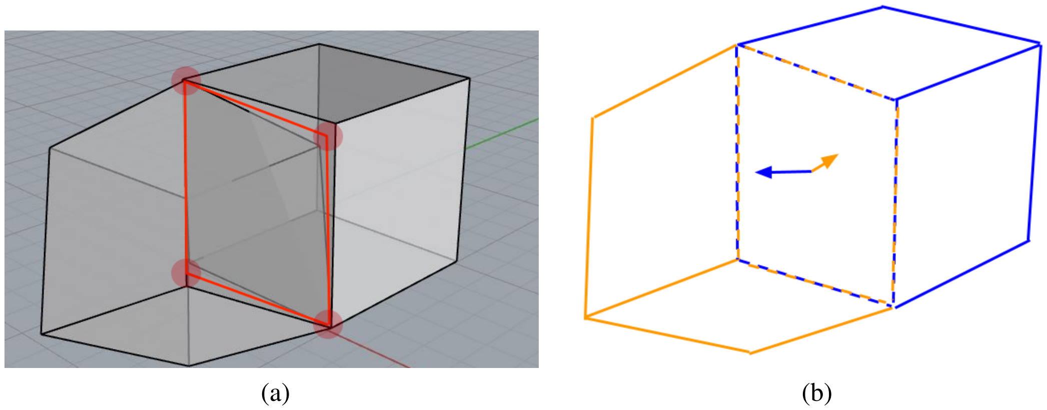


Figure 2.2: The ambiguity effect can be demonstrated with two rendered cubes in the perspective view. The two cubes are rotated such that two faces are interlaced. The red square in 2.2a is a simulated projection of a square tag. The red circular regions denote the region of potential corner detection in a noisy scene. 2.2b is a sketch of the potential resulting 2D projection. The pose can converge to either one of the two faces.

using back projection methods. Localization of circular features is generally more accurate, and thus generates better pose estimation at the cost of higher computation time [**?**]. However, small circular features become hard to detect when they are far away from the camera or prospectively rotated, and thus their effective range is much smaller than that of square tags.

ARTags [**?**], ARToolkit [**?**], ArUco [**?**], AprilTag [**?**] and AprilTag 2 [**?**] are examples of squared-based fiducial tags. The perspective projection of a square becomes a general quadrilateral. Given the scale of a single marker, the full 6-DOF pose can then be estimated using the corners of the quadrilateral. However, since the tags are detected using rectangles and lines, the accuracy of their corner point sub-pixel locations is limited. Among the square tags, ARToolkit is one of the earliest detection systems, and it is mainly used for Augmented reality applications. Built on top of ARToolkit, ARTags and Apriltag reduced the computation time by using a 2D binary pattern as the payload. Both systems use the image gradient to compute the tag border making it robust to lighting changes and partial occlusions. Relative to ARTags, Apriltags have a lower false positive rate, as they use a lexicode-based system that is invariant to rotation. In addition, Apriltags have higher detection rates at further distances and at more difficult viewing angles. Recently AprilTag 2 improved upon the original Apriltag. It implements a new boundary segmentation algorithm which further reduces the computing time for detection and increases the detection rate. Compared to circular tags, the advantages of square tags are that they can be located very efficiently and they have reliable decoding schemes. Therefore, lts they are more suitable for robotic applications that require a robust system.

2.1 Pose Ambiguity

In square fiducial marker detection, the pose is computed using the four corners of the tag. Since the tags are planar, it is easy to compute perspective point correspondences from the corners.

This can be formalized as a specific case of pose estimation from Perspective-N-Point and it has been well studied in geometry-based Computer Vision literatures [**? ?**]. There are numerous optimization methods such as the ones proposed in [**?**] and [**?**] to solve this problem. In particular, Horaud et al. [**?**] show that there is a deterministic analytical solution to the Perspective-4-Point (P4P) problem when the points are coplanar as they are on the tag. In practice, however, these methods are very sensitive to noise in the scene. When ARTags, Apriltags and ARToolkit systems are used in scenarios shown in Figure 1.2, the poses of the tags are unstable even when the scene is static. Since the minimal number of perspective points are used to estimate the pose, a small variance in the corner detection process will yield estimations far from the true pose.

We will illustrate an ambiguity effect caused by noise by using two overlapping cubes, shown in Figure 2.2. The overlapping face of the two cubes are interlaced but rotated by 120 degrees. However, due to perspective projection, the squares appear to be on the same plane. With low camera resolution, the overlapping squares become virtually indistinguishable. The red circular regions are the detected corners under some sensory noise. Even though the reprojection error is minimized in the 2D space using P4P optimization methods, the 3D pose can still be far off. The result of the optimization can be characterized as a bimodal distribution and a function of the the viewing angle and distance. Depending on the noise level in the scene, the optimization might converge to either one of the local minima causing the pose estimation to be unreliable.

Chapter 3

Robust Fiducial Tag via Sensor Fusion

3.1 Approach

This section describes a method for accurately estimating poses for square fiducial tags in noisy settings by fusing RGBD data. The process of detecting and decoding the tag is identical to previous fiducial tag systems. After the tag corners are detected, they are treated as approximated locations of the true corners. Using the corners, the method implicitly evaluates the depth data and RGB data as two separate observations and fuse them to minimize the error in 2D and 3D space.

There are three distinct components to this method. First, we find the plane in $SO(3)$ containing the tag using depth data and detected corners. Secondly, an approximate initial pose is computed using the depth plane. Finally, the method refines the initial pose using the RGB data by minimizing the reprojection error within a constrained space. Each component is described in detail in the following subsections.

3.1.1 Depth Plane Fitting

The first step is to extract the plane which the tag is laying on. We assume that the RGBD sensor is calibrated such that depth and RGB streams are registered to the same frame. The rectangular patch of points in the depth image bounded by the approximated corner pixels $\boldsymbol{y} = [y_1, y_2, y_3, y_4]$ contains the range information of all the points on the tag. Here we take advantage of the planar characteristic of the tag. By fitting a plane over the range data, we can constrain the pose of the tag to be on the plane.

The raw range data retrieved from the depth sensors are generally noisy. The borders and dark regions of the tag produce unreliable range data and artifacts due to a weakness of our depth sensor (time of flight sensor from Kinect V2). Therefore, we first filter the data by removing points too far from the median before fitting the plane. Nevertheless, the remaining points could have a large variance depending on the lighting condition and the magnitude of the in-plane rotation. The accuracy of the plane fit and initial pose estimation is directly affected by the noise level of data. We will characterize the uncertainty of the plane fit and adjust the weight of the depth pose estimation accordingly during the fusing stage.

In implementation, we used a Bayesian plane fitting algorithm described in [**?**] which computes the Hessian Normal parameters $[\hat{\boldsymbol{n}}, d]$ of a plane for noisy range data through optimizing

$$\min_{\hat{\boldsymbol{n}},d} \sum_{j=1}^{N} \frac{(p_j(\hat{\boldsymbol{n}} \cdot \hat{\boldsymbol{m}}_{\boldsymbol{j}}) - d)^2}{(\hat{\boldsymbol{n}} \cdot \hat{\boldsymbol{m}}_{\boldsymbol{j}})^2 \sigma^2 \{\bar{p}_j\}}$$

where $\hat{\boldsymbol{n}}$ is the local normal to the planar surface of the depth point and $\hat{\boldsymbol{m}}_{\boldsymbol{j}}$ is the measurement direction for the sensor for point p_j. The algorithm in the paper assumes a radial Gaussian noise in the range data p_j with the standard deviation modeled by a function in the form

$$\sigma\{\bar{p_j}\} = \frac{kd^2}{\|\hat{\boldsymbol{n}} \cdot \hat{\boldsymbol{m}}_{\boldsymbol{j}}\|}$$

The coefficient $k > 0$ is an estimated value obtained from sensor calibration. In our implementation, we obtained k by using the Kinect V2 model obtained from [**?**].

An important result we used from [**?**] is the covariance matrix for the plane-parameters. The covariance is obtained by taking the *Moore-Penrose generalized inverse* of Hessian matrix computed from the Lagrangian. It characterizes the uncertainty of the plane fit and implicitly measures the relative accuracy of the depth data.

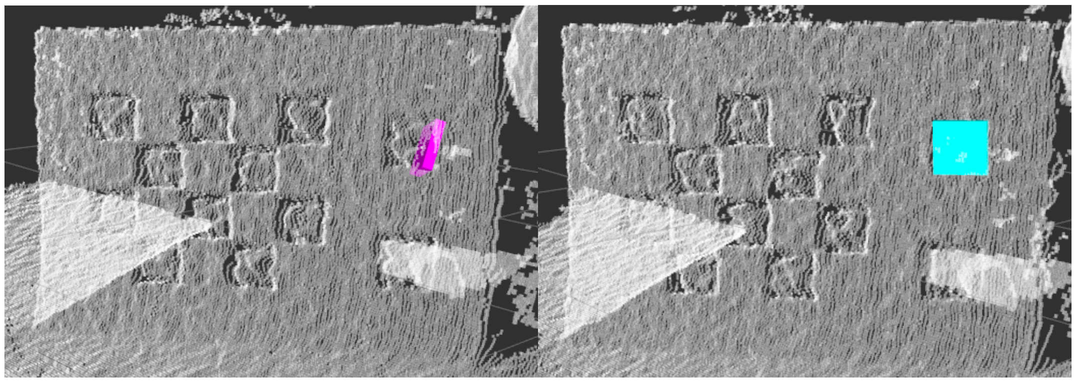
3.1.2 Initial Pose Estimation

The 6 DOF pose of the tag can be described as the transformation $[R, \boldsymbol{t}]$ aligning the tag frame's coordinate system and the sensory frame's coordinate system of the robot. The depth plane $D[\hat{\boldsymbol{n}}, d]$ alone is insufficient to determine the transformation as it only defines 3 DOF. Since the depth plane was computed based on the approximate center of the tag, we can use the center of the tag and center of the plane as a pair point correspondence. However, there are still infinite number of valid poses rotating about the normal $\hat{\boldsymbol{n}}$. One solution is to constrain the pose by using a corner as an extra point correspondence to solve for the optimal rotation. In practice, the accuracy of this method largely depends on the depth accuracy of the chosen corner point.

An alternative is to use all 4 detected corners as 4 pairs of point correspondences for the optimization. We projected the detected corners onto $D[\hat{\boldsymbol{n}}, d]$ to get the coordinates $\boldsymbol{p} = [p_1, p_2, p_3, p_4]$ in the robot sensory frame. The corner coordinates $\boldsymbol{q} = [q_1, q_2, q_3, q_4]$ in the tag frame can be easily calculated since the tag is a square plane. We define the center of the tag as the origin, and the coordinates are simply the location of the corners on a Cartesian plane. Given these two sets of 3D point correspondences, the pose can be computed as a rigid body transformation estimation. Solving for the optimal transformation $[R, \boldsymbol{t}]$ requires minimizing a least squares error objective function given by:

$$[R, \boldsymbol{t}] = argminR \in SO(3), \boldsymbol{t} \in \mathbb{R}^3 \sum_{i=1}^{n} w_i |R\boldsymbol{q_i} + \boldsymbol{t} - \boldsymbol{p_i}\|^2$$

There are numerous approaches to solve Eq. **??** described in [**?**]. Since we have very few



(a) RGB (b) RGBD

Figure 3.1: The pose of the Apriltag visualized in RViz computed using the original library VS our RGBD fused method.

correspondences and they are assumed to be correct, it can be computed efficiently using SVD:

$$\bar{p} = \frac{1}{N}\sum_{i=1}^{N} p_i \qquad p_{ci} = p_i - \bar{p}$$
$$\bar{q} = \frac{1}{N}\sum_{i=1}^{N} q_i \qquad q_{ci} = q_i - \bar{q}$$

$$p_c^\top q_c = U\Sigma V^\top$$
$$R = VU^\top$$
$$\boldsymbol{t} = \bar{q} - R\bar{p}$$

Here, R and $\boldsymbol{t}$ are the corresponding rotation and translation components of the the transformation. The above approach minimizes the least square error of the transformation and it is robust to small errors in the correspondences. The resulting pose obtained from the range data, although not accurate, provides a good approximation for the true pose.

3.1.3 Pose Refinement

Lastly, the pose is refined by minimizing the reprojection error in Eq.**??** using the initial pose estimated from the previous step. The camera is assumed to be calibrated and the camera projection model K is known. Here, R^* and $\boldsymbol{t}^*$ are the optimal pose in the constrained optimization

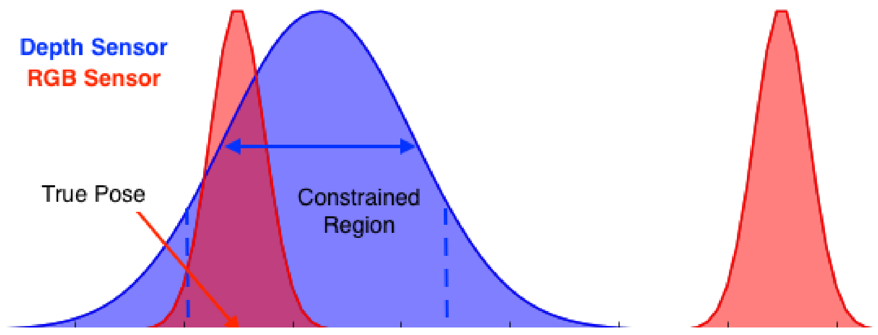


Figure 3.2: An abstract visualization of the optimization constraints. The blue curve is the initial pose estimation obtained from the depth plane. The red curves are the ambiguous poses from the RGB image. We constrained the region of optimization based on how well we fit the depth plane.

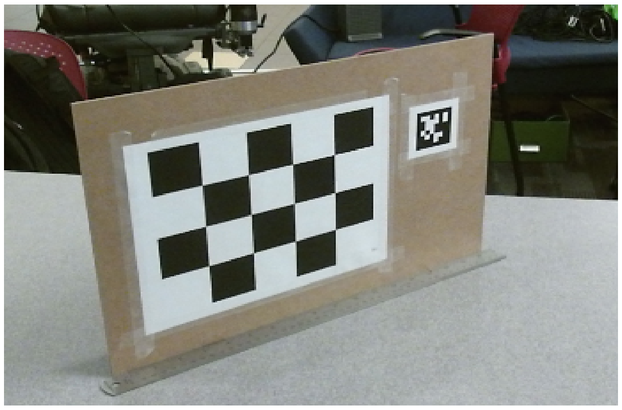
function

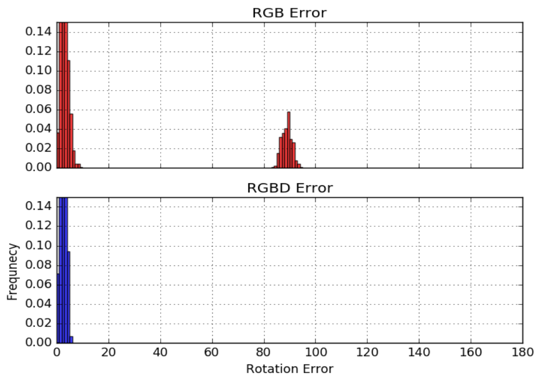
$$\begin{aligned} [R^*, \boldsymbol{t^*}] = \quad & argmin_{R^*,\boldsymbol{t^*}} \sum_i^n \|(K[R^*|\boldsymbol{t^*}])\boldsymbol{p_i} - \boldsymbol{y_i}\|^2 \\ R^* = \quad & R(\Delta R) \\ \boldsymbol{t^*} = \quad & \boldsymbol{t} + R(\Delta \boldsymbol{t}) \\ \texttt{subject to:} \quad & \\ \Delta R < \quad & \Gamma_R,\ \Delta \boldsymbol{t} < \Gamma_t \end{aligned}$$

Intuitively, the optimal pose is the one with minimal reprojection error in the RGB space and aligned with the plane in the depth space. Therefore, the goal of the optimization is to find the local minimum closest to the initial estimation within allowable region Γ as illustrated with Figure 3.2. The key challenge is to determine the constrained region Γ_R and Γ_t such that it include a locally optimal pose and exclude the ambiguous pose. In most cases where the depth plane yields a good fit, this region should be small because the optimal pose is close to the initial estimate. When the depth sensor is noisy, the Γ increases since the initial estimate might be far off. Thus, the constrained region Γ is defined by the uncertainty in the initial estimate and it is characterized by the covariance of the plane parameters. In implementation, we used a trust-region optimization algorithm to bound the constraints. The scaling parameters for the covariance is empirically tested to obtain the best results for our robot.

The strength of this method is that it harness the benefits of RGB and depth information without explicitly assuming their relative accuracy. One advantage of RGBD sensors is that the camera and the depth sensor often work optimally with different constraints. In the example of Kinect, the RGB camera is sensitive to lighting and works poorly in scenes with low illumination. However, the time of flight depth sensor is unaffected by such a problem. On the hand, the time of flight sensor yields poor range results on surface edges, but the RGB camera works exceptionally well with edges where there is a high color contrast.

3.2 Experimental Results


(a) RGB image at 60°



(b) Rotation errors across 1000 trials

Figure 3.3: An example of the experimental setup in 3.3a. Groundtruth is computed from a large chessboard where the relative transformation to the tag is known. Each data collection, shown in 3.3b, is ran through 1000 trials and pose errors are measured. Since a 7 cm tag only occupies 15 pixels, the system has a signficant failure rate even at 65 cm.

The key problem we are trying to resolve is the localization accuracy of Apriltags in noisy situations. Therefore, we want to test the resilience of our algorithm and show that it can obtain reasonable pose estimations under high level of noise. Figure 3.1 demonstrates an example visualization of the result. We also compare our method against *ar_track_alvar*, a popular ARTag detection package that incorporated depth information. Finally, we briefly tested the runtime of the algorithm to show that it remains capable of real time detection.

In our experiments, we measured the rotational and translation accuracy of the detection algorithms with respect to three different independent variables: viewing angles, distances, and lighting conditions. We placed a standard camera calibration chessboard and a 7 cm Apriltag on a solid planar board. The Apriltag has a fixed distance from the chessboard. This is used to compute the ground-truth pose for the tag. By using a large chessboard, we can detect the corners to a sub-pixel accuracy and compute accurate ground-truth poses unsusceptible to lighting and sensory noises.

Since our algorithm aims to solve the pose ambiguity problem, we evaluated all the results based on an adaptive threshold separating the bimodal distribution. This is a reasonable evaluation criteria because easily detectable ambiguous poses are often close to the true pose, making the average of absolute errors small even though the poses might be wrong most of the time.

3.2.1 Viewing Angle

Due to the perspective ambiguity effect, the localization accuracy of the Apriltags is heavily affected by the viewing angle of the tag. To characterize the effect, we placed the testing board with a tag in front of the robot as shown in 3.3a. The testing board is 0.65 meters away from the

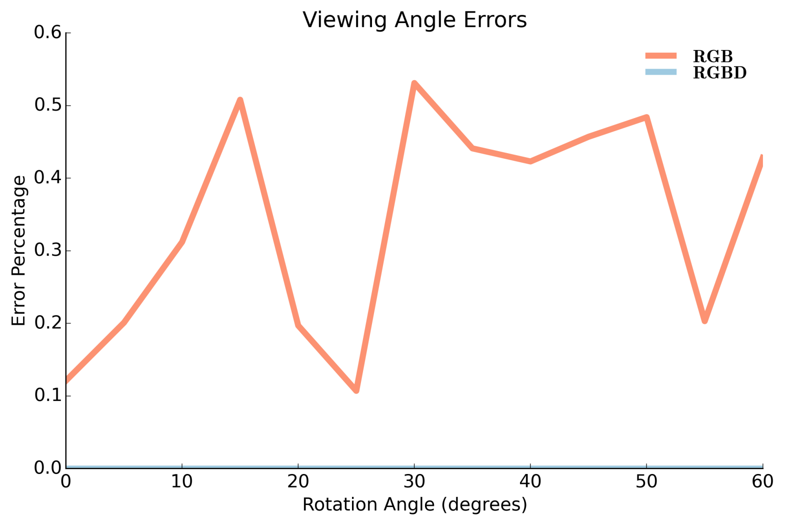


Figure 3.4: Viewing Angle vs Error Percentage $(0.1 = 10\%)$ under different simulated noise level. The new RGBD based algorithm can resist noise in the RGB image and it vastly outperforms the original algorithm.

sensor and rotated it at a increment of 5 degrees from 0 degrees to 60. The angles are measured from the axis parallel to the sensor. This is about the range which the tag can be detected reliably given the camera resolution and the distance. At each angle, we captured the RGB image, depth image, and detection outputs from the Apriltag library.

For each captured data bundle, we introduced 3 levels of Gaussian noise of $\sigma = 0.2$, $\sigma = 0.5$, $\sigma = 1$ to the RGB image and computed the resulting tag pose. This is repeated for 1000 trails for each data bundle per noise level and the errors are computed for each trial.

The empirical result in Figure 3.3b show a very clear bimodal distribution, as we expected, for the detected poses for a given data bundle over 1000 trials. In Figure 3.4, we threshold all the poses based on their rotational errors and plotted the percentage of unacceptable poses at each viewing angle. The proposed RGBD fused algorithm vastly outperforms the original algorithm as it has better localization accuracy at all viewing angles and noise levels.

3.2.2 Distance

The relationship between the distance and localization accuracy is much more apparent. As the tag moves further away from the sensor, the number of pixels on the tag decreases. The perspective ambiguity effect becomes more apparent when there is only a small patch of pixels on the tag. We show the results of both RGB and RGBD methods in Figure 3.5. During the experiment, it is difficult to keep the viewing angle precisely consistent at each trail. Therefore, the pose error percentage using RGB is not increasing smoothly as they are in the simulation results.

We see a clear increase in error percentage in the proposed method when the tag is far away from the camera. This is contributed both by a smaller tag patch size in the depth image and an increase in noise with the Kinect sensor at a further distance. In these cases, the variance of the

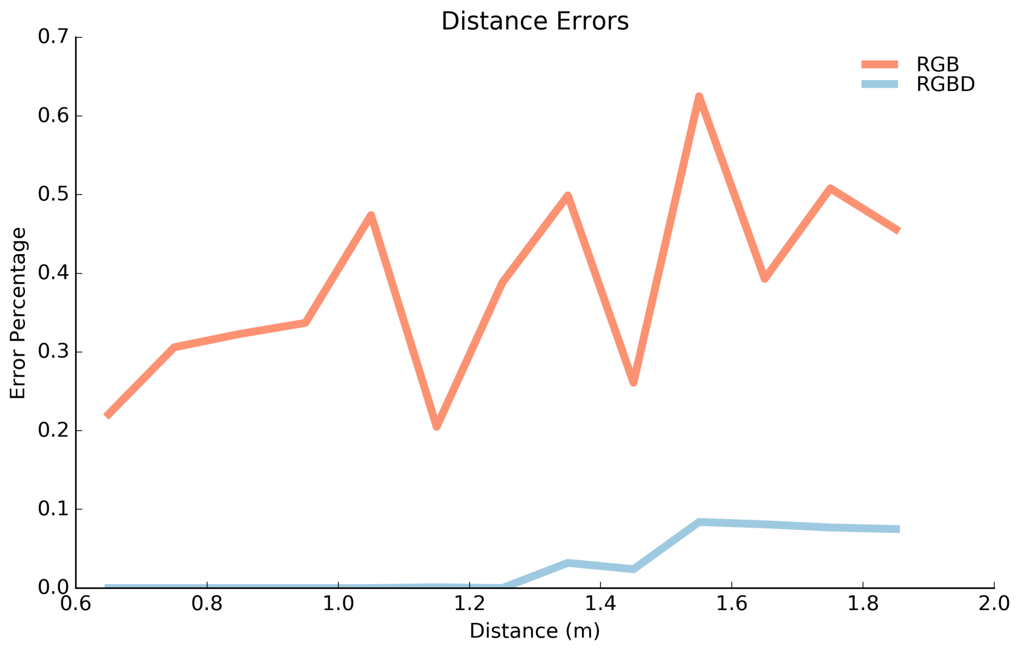


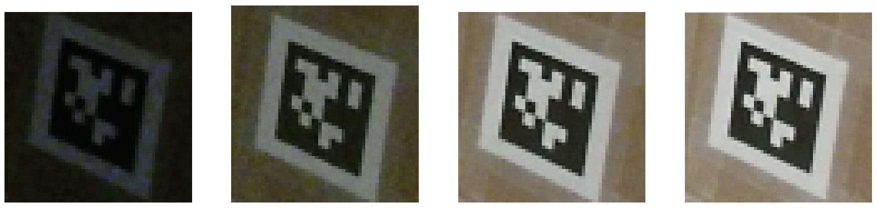
Figure 3.5: Distance vs Error Percentage $(0.1 = 10\%)$. Data are captured at a 10 cm increment from 65 cm to 185 cm.

depth plane estimation becomes very wide and the algorithm is unable to converge to the correct pose. Nevertheless, our method shows a significant gain in accuracy at every distance.

3.2.3 Lighting

From our past observations, poor lighting condition is the most significant contributing factor to noise and it results in low localization accuracy. The Kinect V2 sensor used in our experiments dynamically adjust the exposure time under low lighting conditions. When pictures are taken below or near the adjustable range of the sensor, they contain very noticeable noise as shown in Figure 3.6.

We also tested the algorithm under harsh lighting conditions in a real world setting. The data were captured under 4 different lighting conditions: 20 lux (dark), 43 lux (dim), and 90 lux (normal), 243 lux (bright). We recorded a static scene over 5 seconds and randomly sampled 100 frames to run the test. In Figure **??**, we demonstrate the particular result collected where the board is 0.65 m away and angled at 40 degrees. Other data captures reflect similar results. The localization accuracy significantly improves with better illumination. At the lowest illumination, nearly 25% of the poses were unacceptable. By using depth sensor which is unaffected by poor source radiance, there are only 3% of unacceptable poses.



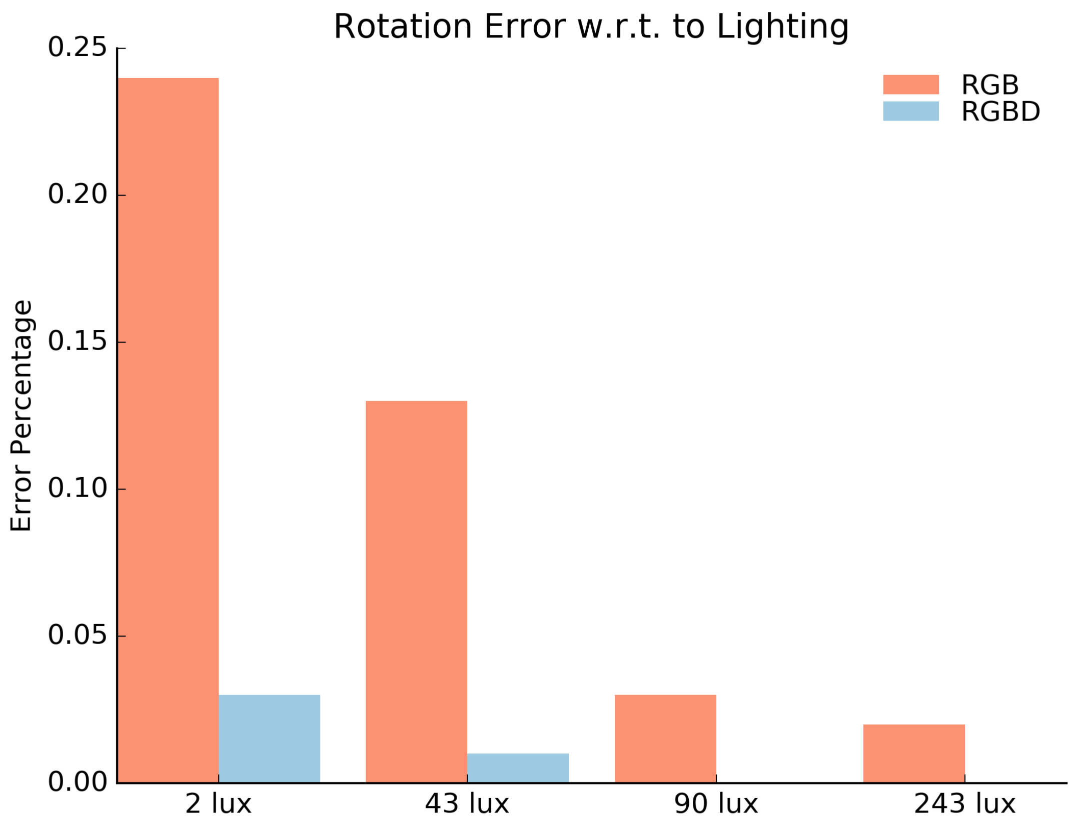
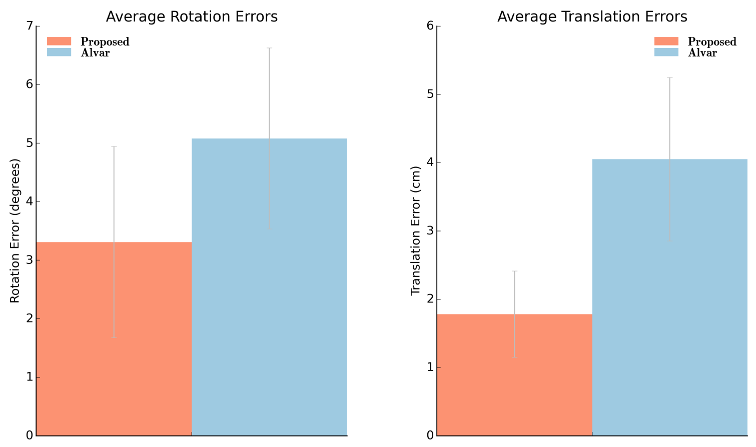


Figure 3.6: Apriltags captured by Kinect V2 under different levels of illumination. The RGB sensor dynamically adjust the exposure time to compensate for low lighting. In 3.6a, the image is captured outside of Kinect's adjustable range and the pixels are underexposed. In 3.6b, the long exposure time introduced noticeable noise to the image.



(a) Rotation Error (b) Translation Error

Figure 3.7: Average pose errors compared with *ar_track_alvar* package.

3.2.4 Benchmark Against ar_track_alvar

ar_track_alvar is a ROS wrapper package for Alvar [**?**], an open source AR tag tracking library. The package is capable of pose estimation for robots similar to Apriltags. In particular, it implements a module where depth sensor is integrated to improve the pose estimation. The package uses the detected corner points to extract a patch of point clouds containing the tag then compute its centroid. The pose is then computed by aligning the centroid with the center of the tag.

We implemented a similar module for the Apriltag and compared the pose accuracy between our proposed method and the module using all the collected data. The results are shown in Figure 3.7. The two algorithms performed similarly in rotation error, but the proposed method was on average 2 cm better with the position component. The spread of error is also much smaller for the position component indicating that our proposed method is more consistent.

3.2.5 Computation Time

With our current implementation in Python, the additional computation time for the sensor fusing process is 11 ms. Therefore the entire detection pipeline can process a 960 x 540 image within 35 ms. All tag detectors and the fusing process were running in a single-threaded mode of an Intel core. Since our sensory updates at roughly $35Hz$, the entire pipeline can process the tags and estimate the pose in near real time.

3.2.6 Discussion

TBD

Chapter 4

Pedestrain Manipulation Background

Pedestrian Intent Prediction To manipulate the trajectories of pedestrians, we must know how the predict the original trajectories of these pedestrians. Activity forecasting has been studied using semantic scene understanding combined with optimal control theory [**?**]. Inverse reinforcement learning has proven useful in predicting the trajectories of pedestrians as well [**?**]. Similarly, active learning approaches to learn pedestrian reward functions and human internal state has proven successful [**? ?**]. Building on this work the concept of social LSTMs has been developed, designed to model pedestrian behavior [**?**]. More recently, concepts from game theory has been used to predict human interactions [**?**].

Human-Robot Interaction The relationship between pedestrians and robots has been studied previously. Studies that explore the relationship between the motion characteristics of a robot and its perceived affect on people is well researched [**?**]. However, in our research we study the motion characteristics of pedestrians and how our robot's actions can change them.

Blind Assistance Robots like our robot for navigating the blind through pedestrian environments is not new. Somewhat similar is NavCog, an application designed to guide the visually impaired through open spaces using Bluetooth Low Energy beacons [**?**]. NavCog, however, does not make an effort to help its user navigate through crowded areas.

Contextual Bandits and Online Learning Since pedestrian distributions can rapidly change as an agent moves through a changing environment, work for bandit algorithms under adversarial settings is relevant. The primary bandit algorithm for this case is EXP4, a no-regret algorithm proven to perform well under adversarial circumstances [**?**]. Various follow-up algorithms that improve EXP4 have been developed that work improve the regret bounds for EXP4 [**? ? ? ?**]. In our case, we want to modify EXP4 to actively generate new expert policies.

Dirichlet Process The Dirichlet Process has been a proven way to compute nonparametric clustering in a wide variety of scenarios. For example, the Dirichlet Process was utilized successfully to learn ego-action categories for first-person sports videos [CITE3], or to learn spatial activation patterns in fMRI data [CITE4]. More interestingly, Gaussian mixture models with the Expectation Maximization have been used to guide the exploration of finding multi-optima policy solutions, instead of developing algorithms to find a single policy solution [CITE1]. Related to this is to find a variable number of policies using a Bayesian non-parametric approach using infinite Gaussian mixture models [CITE2].

Policy Gradients Because value-based reinforcement learning methods are less scalable to high dimesionality environments and greedy policy updates may be unstable, we use policy search methods for computing optimal policies for our robot. Specifically, we desire to use model-free policy search methods using policy gradients. Likelihood policy gradients are traditionally used as policy search methods [**?** **?**], but more recently natural policy gradients have been used successfully [**?**]. Other algorithms like Expectation Maximization have been adapted to perform model-free policy search as well [**?**].

To conduct theoretical experiments before moving to the physical robot, we have constructed a simulation based on the popular PedSim package [] as shown in **??**. In the original PedSim package, the pedestrians are simulated as particles and their movements are computed based on a social force dynamics model. In order to make the simulation more suitable for our problem setup, we modified the pedestrian class and introduced a new robot agent class for controlling our reinforcement learning agent. In particular, we added a new set of attributes and an awareness factor to the pedestrian class. The awareness level is generated stochastically based on the attributes. A high level of awareness makes the pedestrian more likely to move away from the robot agent as it get closer. On the other hand, a low level of awareness makes the pedestrian much more likely to collide with other pedestrians and the robot. The robot agent is a special agent which has no awareness attribute but it is able to toggle on a force field represented by a circular obstacle. The force field can be viewed as a simplified representation of playing a sound which alert the surrounding pedestrians to walk around.

Chapter 5

Adaptive EXP4

Chapter 6

Conclusion