Uncertainty Adaptation in Robot Perception and Learning

Pengju Jin

CMU-CS-18-103

December 2017

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

Thesis Committee:
Kris Kitani, Co-Chair
Siddhartha Srinivasa, Co-Chair

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 experiences and your gauidance every step of the way. Thank you for being understanding and always pushing me to be a more independent thinker.

While my time was split between two projects, it gave me the prvilage to meet many awesome labmates / peers. They continuously motivated and helped me through the inevitable difficult moments at CMU. I thank every member in Personal Robotics Lab for all the inspiration: Jen, Mike, Shervin, Laura, Pyry, Stefanos, Rachel, Clint, Shushman, Gilwoo, Henny, Rosario, Shen, JS, Hanjun, Pras, Ariana, Oren, Daqing, Aditya, Brian 0, Brian 1, and Vinitha. I thank the CaBot team for all the fun and energy they bring to the window-less lab every single day: Eshed, Edward, Chris, Jen, Andrew, Yanda, and Divya. Furthermore I would also like to give a special shoutout to my undergrad mentors Aaron and Shushman for tirelessly teaching me from ground zero.

Finally, I would like to thank my parents, Dejiang and Xiaoying, for encouraging, guiding, and mostly importantly, sacrificing. Without you giving up everything to move to a foreign country so I could have a better oppertunity, I would have never been able to accomplish my achievements today.

Contents

List of Figures

Chapter 1

Introduction

Uncertainty 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

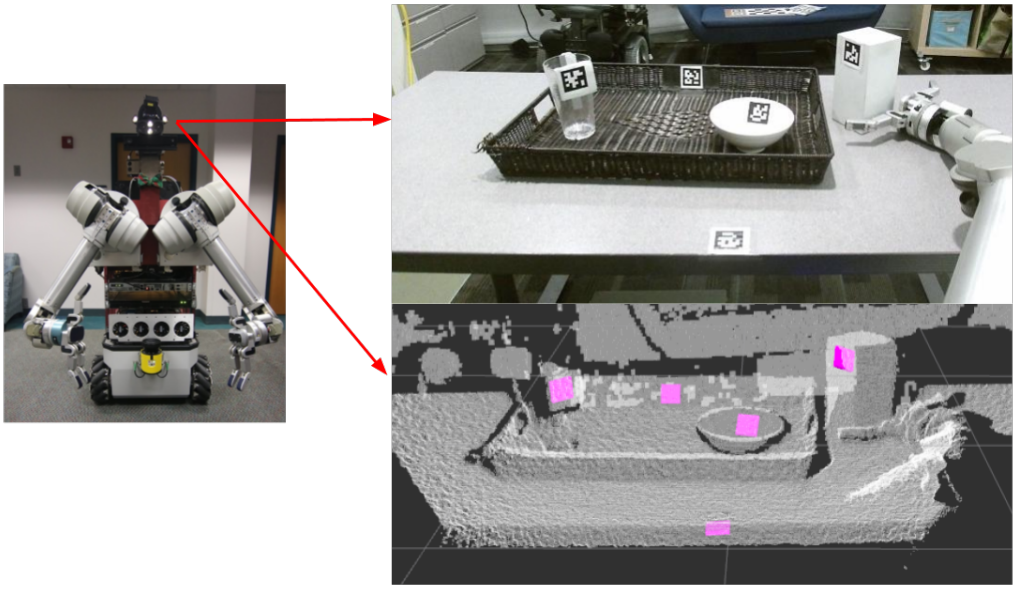


Figure 1.2: Robot about to execute a manipulation task and rearrange the objects on the table. Apriltags are used to find the poses of targeted objects in the scene but the robot ultimately fails to gasp the rectangular prism because the orientation of its pose is wrong.

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 [19] Apriltags [42], and Rune Tags [7], 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 Assistive Navigation for Blinds

The second task we will address involves learning navigation strategies for the blind assistive robot Cabot. Many assistive navigational robotic systems for guiding people with visual impairments are introspective in that they only give navigational instructions to the user and often do not consider the effect the robotic system can have on the environment to clear the path for the user. Without the ability to change the environment, the robotic system must solve a challenging dynamic planning problem where the system must predict the behavior of nearby pedestrians and plan its path accordingly. Unfortunately, in some scenarios it may not be possible to plan a path due to the complexity of the state space or physical constraints imposed by the environment. While dynamic planning is certainly an important component of robotic navigation, we explore the possibility of endowing the robotic system with the ability to manipulate its surroundings in order to clear a path for the blind user.

While there are many possible modes of interaction with the environment, in this work we propose the use of strategically planned audio signals broadcast from our robotic platform to effectively alert pedestrians of potential collision. Modeling the effect of audio interactions is a challenging task, as it is not always clear when and what audio signal should be broadcast to cause pedestrian behavior to change. One of the key reasons for why the task is challenging is that the reaction of nearby pedestrians is not consistent and can change over time based on the situational context. That is, given the same audio signal, the behavior of nearby pedestrians can change due to changes in the environment, changes in the configuration of pedestrians or changes in the unobserved internal state of the pedestrian. This means that the effective state space of the problem is indeed very large and that the distribution over the state space can change over time.

To deal with a dynamic state distribution, we propose to learn an effective strategy to trig-

Figure 1.3: Cognitive Assistance Robot (CaBot): Our blind navigation robot leads a blind user while detecting other pedestrians and making decision of using sound alert to change pedestrians' walking direction.

ger audio signals to clear the path for the blind user by implementing a reinforcement learning framework that continually updates it's audio interaction policy to adapt to the changing nature of the environment. In particular, we propose a adaptive contextual bandit algorithm, where the number of arms (*i.e.,* policies) change over time. The set of policies is repeatedly updated using a Dirichlet Process mixture model to maintain a diverse set of high performing polices. The performance of each policy is based on a reward function that measures collision avoidance and social disturbance. Using our approach, we are able to continuously explore and learn new audio interaction policies for changing situational context.

While the main analysis is performed using simulated pedestrian data, we also evaluate the real-world performance of our algorithm using a suitcase-shaped robot as our assistive navigation system. The robot assumes the form of a suitcase because we envision that such a system will be used in airports and travel scenarios. The suitcase is equipped with a Kinect 2 camera, small computer and a speaker for audio interaction. The system is named the Cognitive Assistant Robot or CaBot for short and is depicted in Figure 1.3.

We make two main contributions in this project: (1) We introduce a new algorithm that has the ability to search and generate new expert policies by maintaining a distribution over a large number of policies. Our algorithm is called Adaptive EXP4, and is an extension to the well-known EXP4 algorithm. (2) We develop a technique for maintaining a potentially infinite number of bandit arms (policies) by utilizing a Dirichlet Process Gaussian mixture model over the space of policies. We show empirically that we are able to minimize the regret of the online algorithm despite having an infinite number of policies.

Chapter 2

Pose Estimation Background

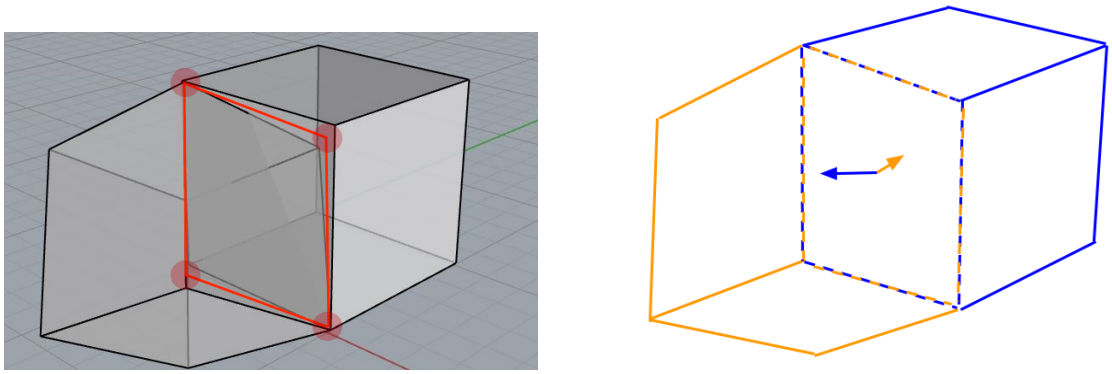
Obtaining highly accurate pose estimation has been an important research area in robotics. Numerous algorithms rely only on RGB or gray scale images. Solving the projection geometry of some detected features and then minimize the reprojection error of the features in the image space [23]. Similarly, methods such as Iterative Closest Point [8] 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 [5, 21]. 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 [38] and Rune tags [7]. The perspective transformation of a circle is an ellipse, which can be used to directly compute the pose 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 [47]. However, small

(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



(a) Perspective 1 (b) Perspective 2

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.

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 [19], ARToolkit [28], ArUco [20], AprilTag [42] and AprilTag 2 [53] 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 [26, 57]. There are numerous opti-

mization methods such as the ones proposed in [16] and [25] to solve this problem. In particular, Horaud et al. [27] 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.

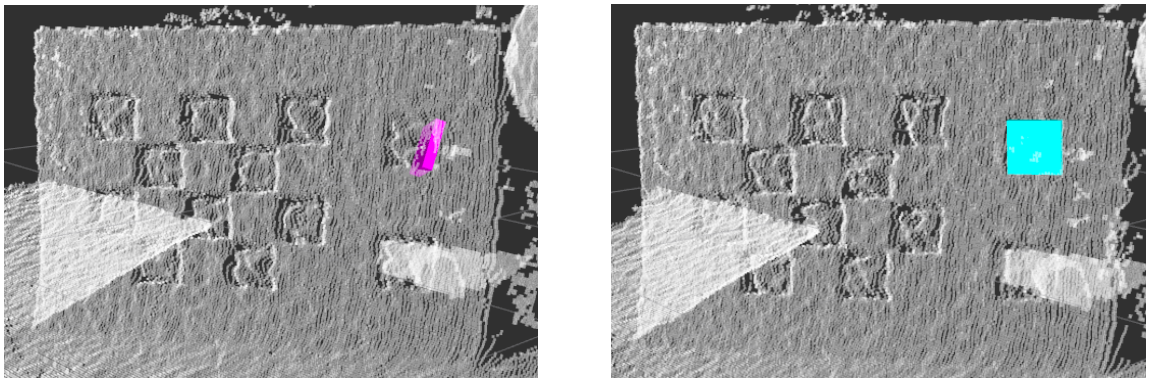


Figure 3.1: The pose of the Apriltag visualized in RViz computed using the original library VS our RGBD fused method.

In implementation, we used a Bayesian plane fitting algorithm described in [44] 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 [40].

An important result we used from [44] 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 [18]. Since we have very few correspondences and they are assumed to be correct, it can be computed efficiently using SVD:

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

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

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

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 function

$$[R^*, \boldsymbol{t}^*] = \quad argmin_{R^*, \boldsymbol{t}^*} \sum_{i}^{n} \|(K[R^*|\boldsymbol{t}^*])\boldsymbol{p_i} - \boldsymbol{y_i}\|^2$$

$$R^* = \quad R(\Delta R)$$

$$\boldsymbol{t}^* = \quad \boldsymbol{t} + R(\Delta \boldsymbol{t})$$

subject to:

$$\Delta R < \quad \Gamma_R,\ \Delta \boldsymbol{t} < \Gamma_t$$

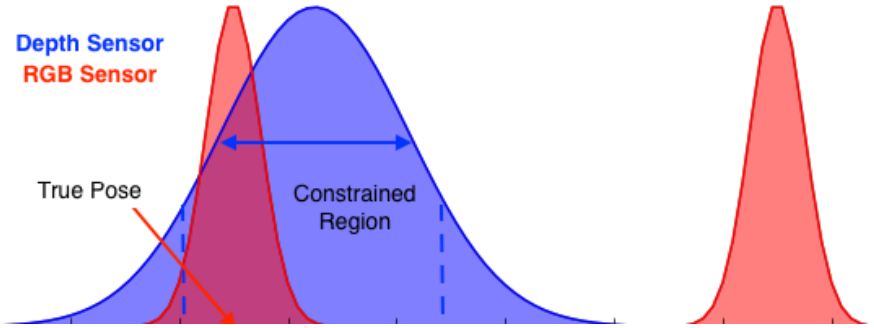


Figure 3.2: An abstract visualization of the optimization constraints. The blue curve is the initial pose estimation obtained from the depth plane. The red curves are the ambiguous poses from the RGB image. We constrained the region of optimization based on how well we fit the depth plane.

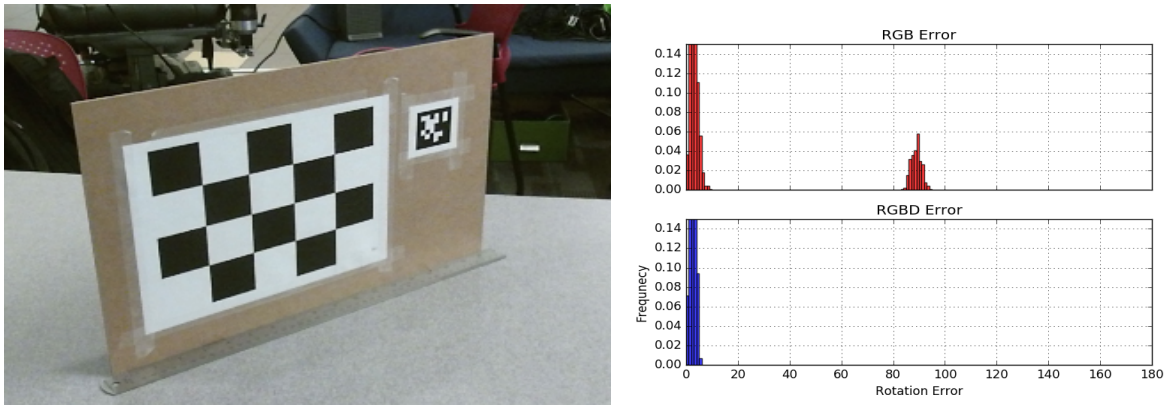
Intuitively, the optimal pose is the one with minimal reprojection error in the RGB space and aligned with the plane in the depth space. Therefore, the goal of the optimization is to find the local minimum closest to the initial estimation within allowable region Γ as illustrated with Figure 3.2. The 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

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



(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.

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 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

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.

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 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.

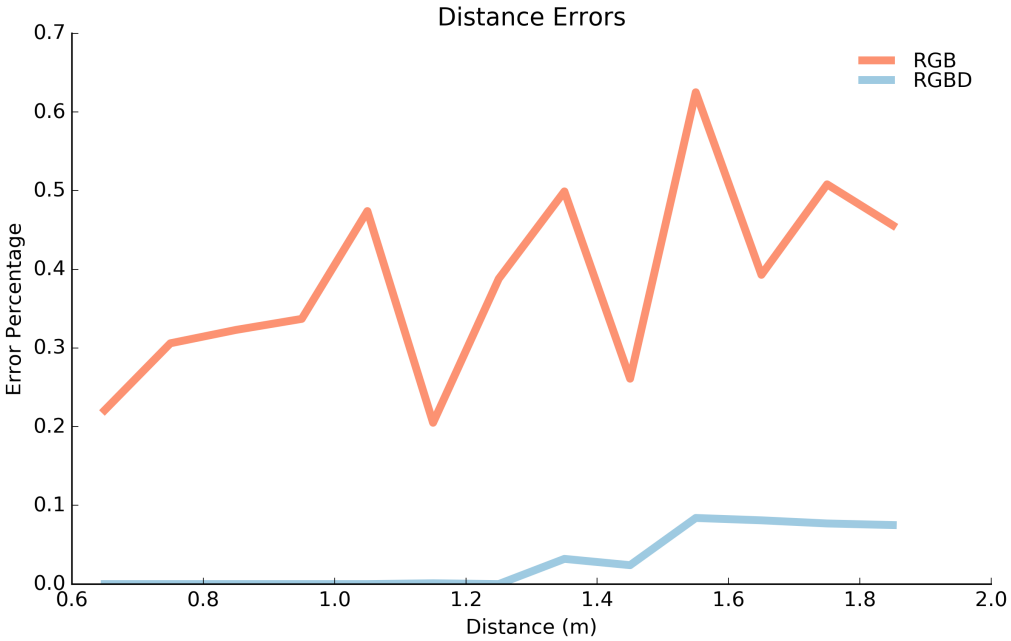


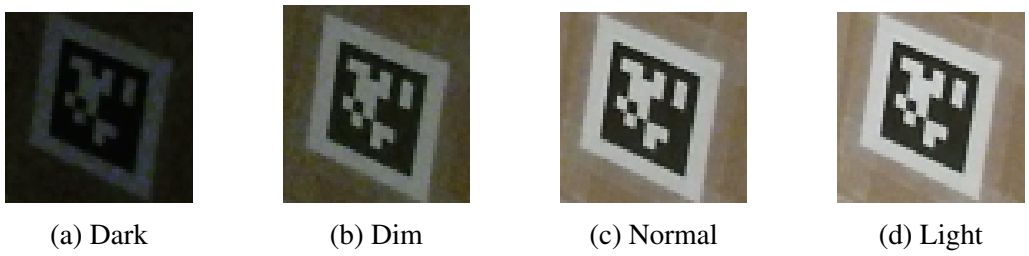
Figure 3.5: Distance vs Error Percentage $(0.1 = 10\%)$. Data are captured at a 10 cm increment from 65 cm to 185 cm.

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.

3.2.4 Benchmark Against ar_track_alvar

ar_track_alvar is a ROS wrapper package for Alvar [41], 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 **??**. 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.



(a) Dark (b) Dim (c) Normal (d) Light

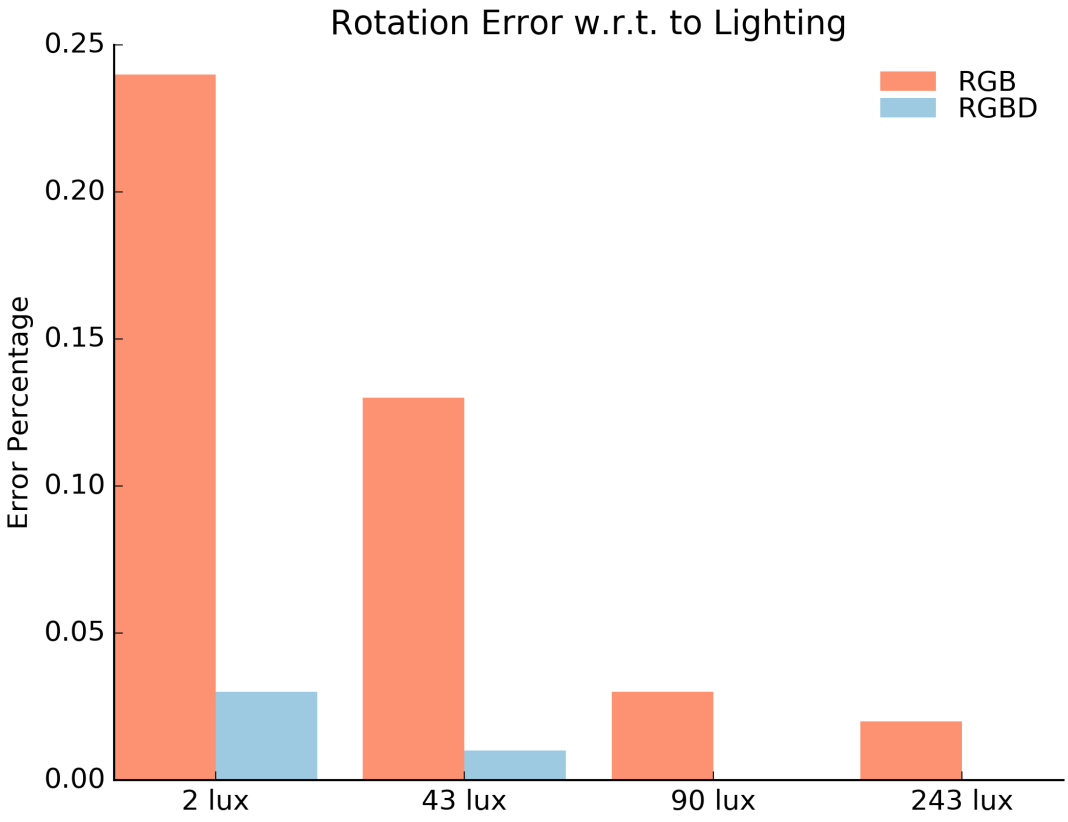
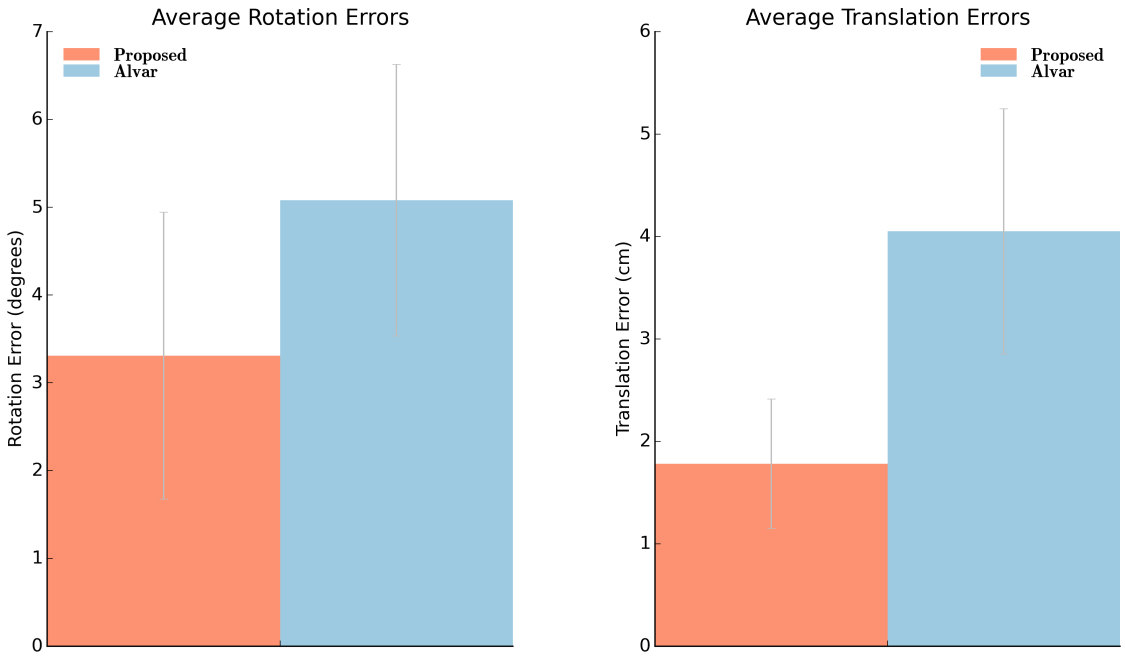


Figure 3.6: Apriltags captured by Kinect V2 under different levels of illumination. The RGB sensor dynamically adjust the exposure time to compensate for low lighting. In 3.6a, the image is captured outside of Kinect's adjustable range and the pixels are underexposed. In 3.6b, the long exposure time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



(a) Rotation Error

(b) Translation Error

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.

3.3 Conclusion

In this paper, we did a in depth analysis of the localization problem with Apriltags. We proposed a novel algorithm of using RGBD sensors to accurately compute the pose of Apriltags robust to noise. It is particularly suitable for robotic applications which require precise poses such as manipulation, SLAM, and others. Furthermore, this technique can be easily generalized to other types of planar fiducial tags. Our implementation is fully open sourced and will be available at: `https://github.com/personalrobotics/`

Chapter 4

Assistive Blind Navigation Background

Navigation Assistance Recent advancement in micro electronics have enabled the development of assistive technologies for the visually impaired to a new level. There is a abundance of systems designed for helping blind navigation through audio feedback [35]. Some rely on GPS for global localization to help with outdoor navigation [4, 46]. Others utilize phone-based technology along with Bluetooth Low Energy beacons or RFID [1, 14] to help blind individuals navigate through indoor scenes. These systems have reliable localization and provide step by step navigation instructions to their users. While these systems are small and easily portable, they often are not intelligent enough to account for dynamically changing environments. Larger robotic systems have also been proposed in [32, 55]. The advantages of larger robotic systems are that they are often equipped with better sensors and are able to make more intelligent decisions based on their surroundings. In our work, we use a similar platform where we embed a semi autonomous navigation system inside a suitcase.

Pedestrian Avoidance One particular problem we are tackling in our work with is pedestrian avoidance. As we observed with passive navigation devices in [1], inattentive pedestrians are difficult dynamic obstacles. Being able to avoid them would drastically improve the user experience, making users more comfortable with the device. This is a common problem in mobile robots. For instance much work has been focusing on building accurate models to predict the pedestrian trajectories in a dynamic setting. Activity forecasting has been studied using semantic scene understanding combined with optimal control theory [29]. Inverse reinforcement learning has proven useful in predicting the trajectories of pedestrians as well [58]. Similarly, active learning approaches to learn pedestrian reward functions and human internal state has proven successful [17, 48]. Building on this work the concept of social LSTMs has been developed, designed to model pedestrian behavior [2]. More recently, concepts from game theory has been used to predict human interactions [36]. In most mobile robot applications [3, 24, 56], prediction models are applied within the planning loop in order to yield to the pedestrians. While this is reasonable in applications such as autonomous cars since the robot has a dedicated operation space, this is not always appropriate with assistive robots which are intended to operate in the same area as the pedestrians. In our work, we wish to learn a set of appropriate actions that allow the pedestrian to move away from the robot instead of yielding all the time.

Policy Learning We will frame our task in the Reinforcement Learning framework where we are going to directly learn a control policy from a set of input observations about the pedestrians. This has been a popular approach in the robotics and AI community. Specifically, Deep Neural Networks (DNN) has been utilized to learn and approximate the optimal control policy directly from sensory inputs and rewards. Recently, policy learning has been shown to be faster and more scalable to high dimensions [15, 33, 34] than value-based methods. Model-free policy learning methods are particularly fitting for robotic controls because it is often difficult to capture an accurate model and robot states are very high dimensional. Most of the work done in this field [45, 49, 54] are variations of computing the policy gradient. In this work, we present an alternative methods of policy learning based on sampling and statistical inference. In these approaches, inference techniques such as Expectation Maximization have been adapted to perform model-free policy search as well [30, 31, 52]. This approach is more suitable for our problem because it allows the system to continuously sample and learn new policies online as our users interact with the world.

Contextual Bandits and Online Learning A fundamental assumption in standard reinforcement learning approach is that the underlying Markov Decision Process (MDP) remains constant over time. However, this is rarely the case in assistant robotics where the robots are often designed to be used in varying scenes. Therefore, we will also formulate the task as an online learning problem since our system must continuously learn appropriate policies as part of life-long learning process. In particular, we will leverage ideas from contextual bandits problems under adversarial settings. The primary bandit algorithm for this case is EXP4, a no-regret algorithm proven to perform well under adversarial circumstances [6]. Various follow-up algorithms that modify EXP4 have been developed that work to improve the regret bounds for EXP4 [9, 37, 39, 51]. We will present a new variation of the EXP4 algorithm and combine it to policy search, similar to [11, 12] to continuously adapt for better policies.

Chapter 5

Adaptive EXP4

5.1 Approach

The goal of this work is to develop an algorithmic framework that allows an assistive navigational robot to naturally signal intent to pass, to a wide range of pedestrians along a navigational path, in order to avoid possible collisions. We model the interaction of the assistive navigational robot (agent) with nearby pedestrians (environment) as a Markov Decision Process (MDP) to learn the best policy for signalling intent. Formally, we would like to learn a policy $\pi(a|s)$ which map an observation $\mathbf{s}$, *i.e.*, observation of nearby pedestrian from on-board sensors, to a distribution over a set of actions $a \in A = \{a_1, a_2, \dots a_M\}$, *i.e.*, a library of warning sounds, such that total reward obtained by following the policy is maximized. To deal with changes in pedestrian behavior over time, we take an online learning approach that dynamically adjusts the weights over a large set of policies such that the most successful policies have the greatest weight. In the following section we describe our MDP formulation and our proposed online learning algorithm called Adaptive EXP4.

5.1.1 Robot-Pedestrian Interaction Model

In our MDP, the state space S consists of a set of observations of visible pedestrians and obstacles in the field of view of the assistive navigation system, where each state is defined as

$$s = [p_1, v_1, \theta_1, \dots, p_L, v_L, \theta_L, o_1, \dots o_N].$$

For each pedestrian l, p_l is a triplet encoding the 3D position, v_l is a triplet encoding the 3D velocity, and double θ_l is a double encoding the 2D bearing. In our implementation, we set $L = 4$ using the four closest pedestrians to the assistive navigation system. We also encode N closest 3D obstacle points in front of the robot, which results in a vector of $3 \times N$ points. Our action space consists of

$$A = \{a_1, \cdots, a_N\},$$

where N is the total number of audio signals, including the no sound action, that the assistive navigation system can broadcast. In our experiments $N = 2$, where we either broadcast the sound or not, but this can be generalized to a longer list of audio signal types.

The reward function $r(s, a)$ is composed of two components: $r(s, a) = r_{ca}(s, a) + r_{sd}(s, a)$. The first component r_{ca} is the *collision avoidance* term, which is zero when no pedestrians are within some collision radius (*e.g.,* 1.5 meters) and a very large negative value otherwise. The second component is the *social disruption* term r_{sd} which is zero when the sound is turned off and a small negative value when the sound is turned on. The collision avoidance reward term encourages the robot to alarm pedestrians who are too close to the system and the second social disruption reward term penalizes the robot for being overly disruptive.

Algorithm 1 Adaptive EXP4

Π is the set of all expert policies
function ADAPTIVE EXP4($\Pi = \{\pi_1, \dots\}$)
    $N = |\Pi|$
    $\mathbf{w} = \{w_i = 1\}$ for $i = 1 \dots N$
    for $t = 1, ..., T$ **do**
        $s_t \leftarrow$ `observeStates()`
        $W \leftarrow \sum_{i=1}^{N} \mathbf{w}_i$
        $P_w \leftarrow \{p_j(t) = \mathbf{w}_j / W\}$
        $\pi_i^t \sim$ `Multinomial`$(\Pi; P_w)$
        explore $\sim$ `Bernoulli`(ϵ)
        if explore **then**
            ${\pi_i}'^t \sim$ `PolicySampler`$(\Pi, \mathbf{w})$
        end if
        $l^t \leftarrow$ `getLoss()`
        if explore **then**
            $\Pi, \mathbf{w} \leftarrow$ `PolicyUpdate`$({\pi_i}'^t, l^t, \Pi, \mathbf{w})$
        else
            $\mathbf{w}_i^{t+1} \leftarrow \mathbf{w}_i^t e^{-\eta^t l^t}$
        end if
    end for
end function

5.1.2 Adaptive EXP4

Since the reactive behavior of neighboring pedestrians can vary greatly over time, the robot-pedestrian interaction policy needs to able to quickly adapt to such changes. To address the dynamic nature of pedestrian behavior, we incrementally learn a large set of robot-pedestrian interaction policies to cover a wide range of pedestrian behavior. To this end, we develop an online algorithm which is able to select the most appropriate policy by maintaining and adapting weights over all polices. In particular, we formulate the temporal adaptation problem using the framework of a contextual bandit algorithm.

In contrast to classical bandit problems, we do not assume that the set of 'arms' (policies) is static but instead attempt to learn many new policies over time. In the classical case, each bandit produces some reward from an underlying reward distribution. In the adversarial case, the

reward distribution changes over time. At each time step, the agent receives a set of contextual information about all arms and it is allowed to choose one arm and claim a reward. The goal is to maximize the reward over a time horizon T. EXP4 is a standard algorithm for solving the contextual adversarial bandit problem through the usage of expert advice. The algorithm uses a set of pre-trained experts to map the contextual information to an arm selection. With each interaction, a set of weights is maintained for the experts and constantly adjusted depending on the result of the trial. The algorithm has been shown to be no-regret with respect to the best pre-trained expert when the number of arms is known. In our scenario, the number of arms grows over time and we develop an algorithm called Adaptive EXP4 (A-EXP4) to select the best arm. A-EXP4 is outlined in Algorithm 1.

Similar to EXP4, A-EXP4 maintains a set of expert policies Π and a vector of weights $\mathbf{w}$ over each policy. In our setting, the policies are represented using linear policy approximator: $\pi(s, a; \theta) = e^{\theta^\top \phi(s,a)}$ where θ is a vector of learned policy parameters and $\phi(s, a)$ is a vector of state features. At each iteration, a policy π is sampled from the multinomial distribution according to the normalized weights. Instead of exclusively applying the policy π as in the classical contextual bandit algorithm, another policy, an *exploration* policy π', is sampled with a small probability ϵ. This probability ϵ is set manually and determines the amount of exploration the algorithm performs. The agent then applies the resulting policy and observes its loss. The loss function we use in our experiments is simply the one step reward described above. Specifically, after taking an action based on the current policy π, we observe the reward by measuring the distance of the closest pedestrians using our vision pipeline. We compute the loss by $l = -r \ / \ $ `|Min Reward|` where `|Min Reward|` is the magnitude of the smallest possible reward (highest penalty). If the selected policy is an exploration policy, it is passed to an online policy update algorithm described in Algorithm 3. Otherwise, the weights are updated according to the received loss using the traditional exponential gradient update. The policy sampling and learning process will be described in detail below.

Algorithm 2 Policy Sampler

1: **function** POLICYSAMPLER($\Pi, \mathbf{w}$)
2: $\quad$ G $\leftarrow$ `GenerateGMM`($\Pi, \mathbf{w}$)
3: $\quad$ $\pi_i'^t \leftarrow$ `randomSampling`(G)
4: $\quad$ Return $\pi_i'^t$
5: **end function**

5.1.3 Policy Sampler: Sampling Exploration Policies

To continually learn new policies, we sample new exploration policies by calling `PolicySampler` (algorithm 2 in step 12 of algorithm 1). The role of `PolicySampler` is to first estimate the distribution over the policy parameters space Θ induced by the current set of policies Π using a Gaussian Mixture Model, and then the GMM is used to sample a new policy. The number of Gaussians in the mixture model is equivalent to $|\Pi|$. The Gaussian mixture distribution induced by a set of two and three policies are visualized in Figure 5.1. The variance of each Gaussian mixture is set using the weight vector $\mathbf{w}$. For each mixture component i, $\sigma_i = L\mathbf{w}_i$) where L is

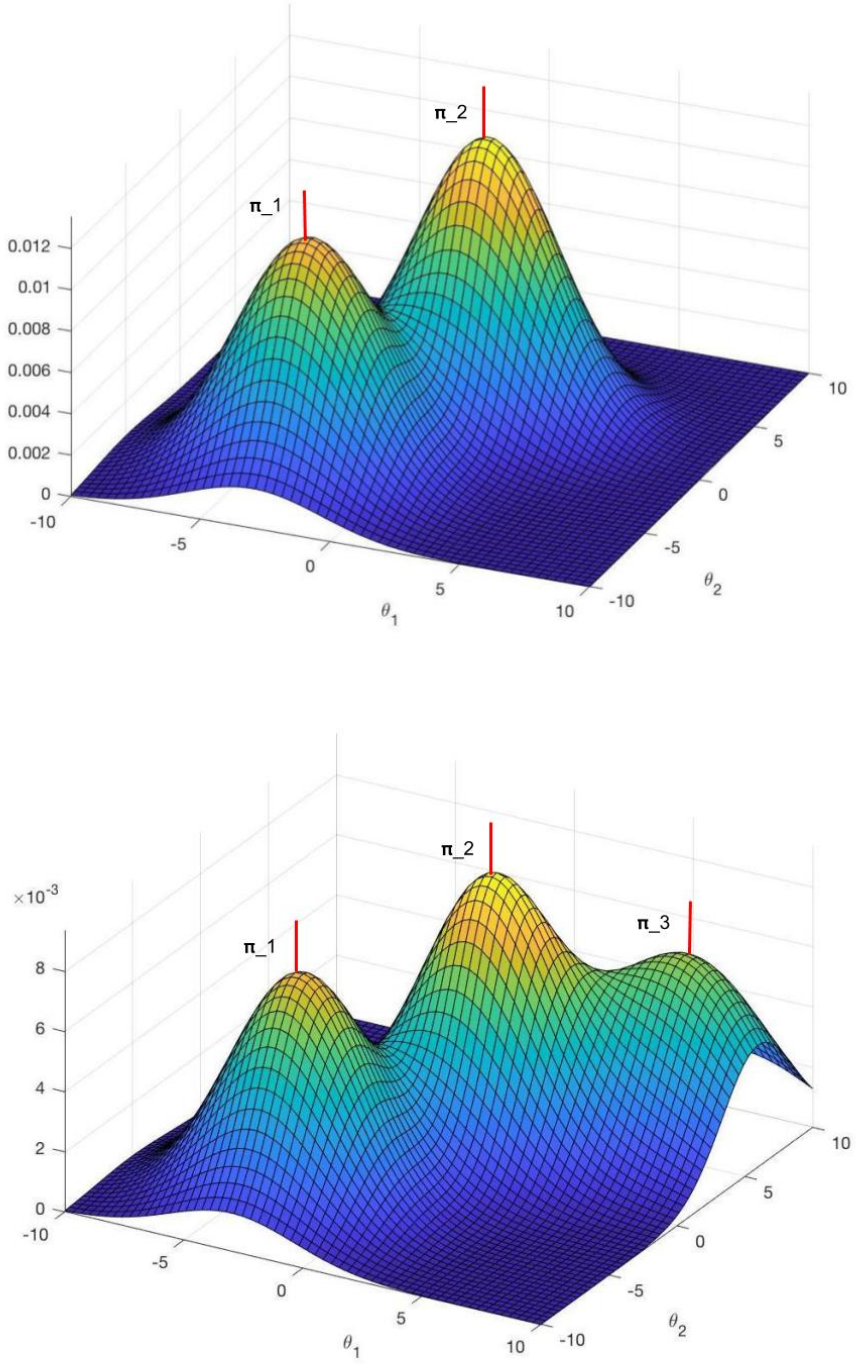


Figure 5.1: Demonstration of an sampled expert policy encoded in GMM. The mixture model is spanned over the parameter space of the policy approximation function. Each component of the GMM represents a single expert policy. In the top figure, there are two discovered expert policies. This representation naturally encodes an policy exploration strategy. As the environment changes over time, we will sample policies according to the GMM and the expert policy weights. With enough samples, we add the new locally optimal policy π_3 to the model as shown on the bottom figure.

a tunable scalar. A high L value encourages more exploration and in our experiments the value is set to 1.5. In this way, we are able to sample new exploration policies which are close to the highest reward yielding policies.

Algorithm 3 Incremental Policy Update

function POLICYUPDATE($\theta, l, \Pi, \mathbf{w}$)
  $r \leftarrow \texttt{exp}(-l)$
  Add policy-reward pair: $\mathcal{D} = \mathcal{D} \cup (\theta, r)$
  Update policy: $\theta^* = \theta^* + \frac{\sum_{d=1}^{|\mathcal{D}|} r_d(\theta_d - \theta^*)}{\sum_{d=1}^{|\mathcal{D}|} r_d}$
  if $|\mathcal{D}| > M$ **then**
    Add policy: $\Pi = \Pi \cup \pi(\theta_i^*)$
    $\Pi, \mathbf{w} \leftarrow \texttt{DirichletProcessMixture}(\Pi, \mathbf{w})$
    $\mathcal{D} = \{\emptyset\}$
  end if
  Return $\Pi, \mathbf{w}$
end function

5.1.4 Policy Update: Incremental Learning from Exploration Policies

In order to continually learn new policies, we implement an *incremental* version of the PoWER algorithm [30] which can be used to search for new locally optimal policies using kernel density estimate over a set of sampled parameter-reward pairs S. As shown by Kober *et al.* in [30], the original PoWER algorithm relies on the idea that a way to safely learn new policies is to look at the convex combination of the sampled policies using importance sampling. In its simplest form, a new policy can be estimated using the following update:

$$\theta^* = \theta^* + \frac{\sum_{d=1}^{|\mathcal{D}|} r_d[\theta_d - \theta^*]}{\sum_{d=1}^{|\mathcal{D}|} r(\theta_d)} \tag{5.1}$$

where θ_d and r_d represents the parameters and reward of a sampled policy π_d, $\mathcal{D}$ is the set of sampled policy-reward pairs and θ^* is the parameters of a mean policy.

As described in Algorithm 3, in our online implementation of PoWER `PolicyUpdate`, each newly sampled exploration policy π' and it's resulting reward is added to a buffer of recent policies $\mathcal{D}$. The current buffered policy $\pi(\theta^*)$ is updated according equation 5.1. Once the buffer reaches a specified size M, we add the current buffered policy $\pi(\theta^*)$ into the set of all policies Π and clear the buffer. In this way, our incremental policy learning algorithm is able to constantly add new policies to the master set of policies Π.

As new policies are added to Π, it is possible that Π will contain policies which are very similar. To address this issue, we estimate a Dirichlet Process mixture model that describes the current set of policies Π. In this way, we are able to effectively reshuffle the policies and indirectly bound the size of Π using the Dirichlet process concentration parameter α_0, similar to [11].

In the Bayesian Dirichlet Process (DP) mixture model, we use DP as a prior for the mixture model distribution. Specifically, we add a probability distribution F_θ to the model, whose parameters θ are drawn from the DP with a base prior distribution G_0:

$$
\begin{aligned}
G &\sim DP(\alpha, G_0) \\
\theta_i &\sim G \\
x_i &\sim F_{\theta_i}
\end{aligned}
$$

The result is an infinite model which can be generated as the limit of a finite process. When a set of n points $\{y_1, y_2, \ldots, y_n\}$ are given and assumed to be distributed according to a DPMM, the posterior probability of any given point y_i belongs to a cluster X_i can be described as:

$$
\begin{aligned}
P(X_i = x | X_{-i}, y_i, \theta^*) &= \frac{c_{-i}(x)}{N - 1 + \alpha} F(y_i, \theta^*_x) \\
P(X_i \neq x_j | X_{-i}, y_i, \theta^*) &= \frac{\alpha}{N - 1 + \alpha} \int F(y_i, \theta^*_x) dG_0(\theta^*)
\end{aligned}
$$

where x is current existing clusters, X_{-i} are the previous assignment, and θ^* being the parameter vector associated with the particular cluster.

Estimating the exact posterior of the DPMM requires computing complex integrals over the infinite DP. In practice, many works have been done to speed up the process by using approximation inference algorithms such as Gibbs sampling [22], variational inference [10] and other methods. In our algorithm, we set F to be a Gaussian distribution and the resulting model is an Infinite Gaussian Mixture Model (IGMM). We the efficient algorithm implemented in [50] so the inference step can be done quickly online.

In our approach, we desire to use DPMM to cluster similar and redundant policy parameters into a single component to preserve the diversity of Π. Furthermore, we would like to eliminate policies with low weights (low returns) to discourage the algorithm from sampling low performance policies. To this end we augment the posterior probability of the DPMM in the following way:

$$
\begin{aligned}
P(C_i = c | C_{-i}, y_i, \theta^*) &= r_i \frac{N_{-i,c}}{N - 1 + \alpha} F(y_i, \theta^*_c) \\
P(C_i \neq c_j | C_{-i}, y_i, \theta^*) &= r_i \frac{\alpha}{N - 1 + \alpha} \int F(y_i, \theta^*_c) dG_0(\theta^*)
\end{aligned}
$$

. This ensures that the probability of each cluster is down scaled by the weights of the expert policies belonging to that cluster. In practice, this is able to limit the number of expert components because most expert policies will have low weights after sufficient iterations of online evaluation.

5.2 Experiments

We will present two sets of experimental results captured in simulation and on our Cognitive Assistance Robot (CaBot) respectively. For our main experiment in the simulation, we will compare the performances of our A-EXP4 algorithm versus the standard EXP4 algorithm (baseline).

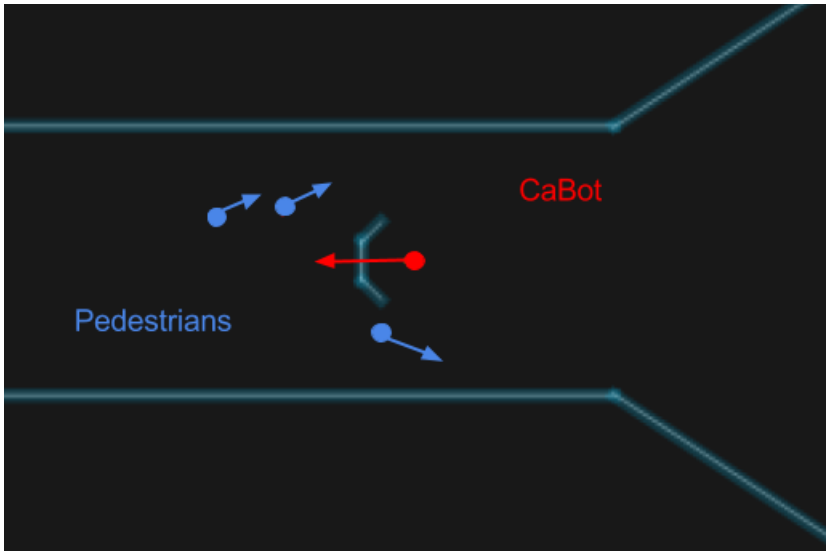
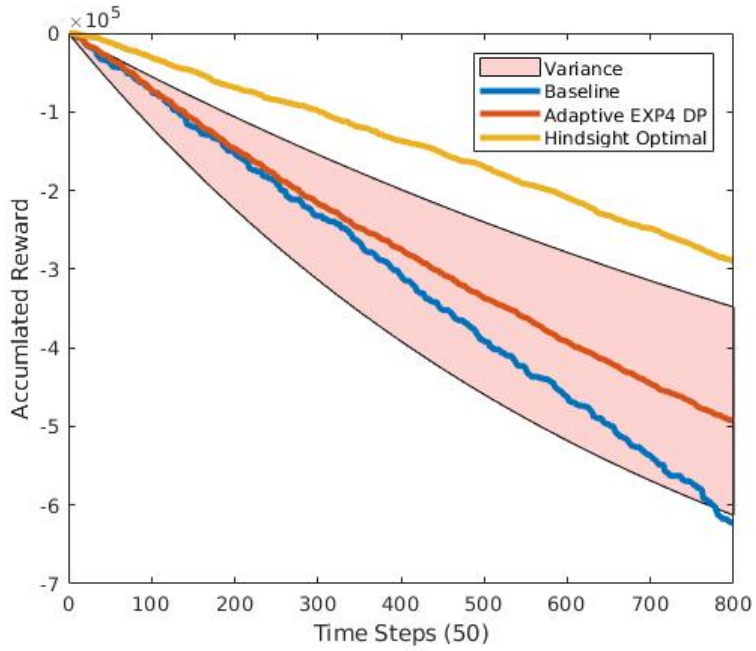


Figure 5.2: Modified PedSim simulation. The pedestrians are represented as blue particles. The robot agent is in red and it toggles a sound force barrier to force the pedestrians to move away.

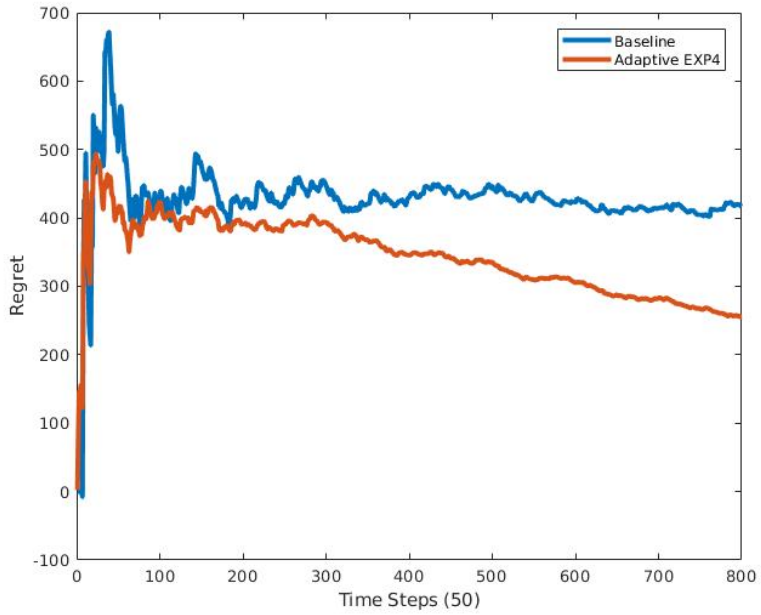
We demonstrate that our adaptive algorithm has superior performances and empirically low regret. Particularly, we show that by reshuffling expert policies with DPMM, we can significantly reduce performance variance and lower the regret. Finally, we will show the overall effect of our approach on incoming pedestrians by running the system on a real robot. We illustrate a shift in pedestrian trajectories distribution around the robot as the result of the warning sound strategy generated by our online learner.

5.2.1 Simulation Environment

We will first describe our custom pedestrian simulation environment. We have constructed a simulation based on the open sourced PedSim package [13], a flexible, light-weight pedestrian simulation engine. In PedSim, the pedestrians are simulated as particles and their movements are computed based on social forces (e.g. the distance from obstacles, other pedestrians). In order to make the simulation more suitable for our problem setup, we extended the base `Tagent` class with two new classes: `RobotAgent`, `Pedestrian`. The `RobotAgent` class overwrites the social force model and always follow its trajectory without yielding. In addition, the robot agent is able to toggle on a semi-circular obstacle in front of itself as an action, as shown in Figure 5.2. The semi-circular obstacle is a simplified simulation of sound waves. Functionally, we use the obstacle to force incoming pedestrians to yield to the robot just as a warning sound would. On the other hand, `Pedestrian` class extends the social force dynamic models with additional attributes such as velocity, avoidance radius, awareness radius, and awareness level. All the attributes are encoded as a floating point value between (0, 1) and they influence how the pedestrians behave around the robot. For example, a high level of awareness makes the pedestrian more likely to move away from the robot when it is far away. In contrast, a low level of awareness makes the pedestrian much more likely to collide with other pedestrians and the

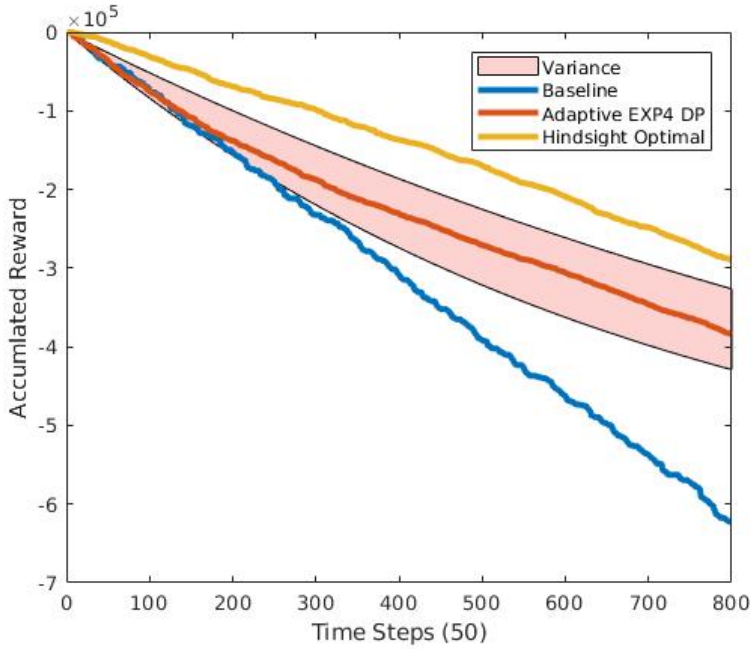


(a) Average accumulated reward of each algorithm over a 40,000 steps testing the AEXP4 algorithm without using DPMM to reshuffle the experts.

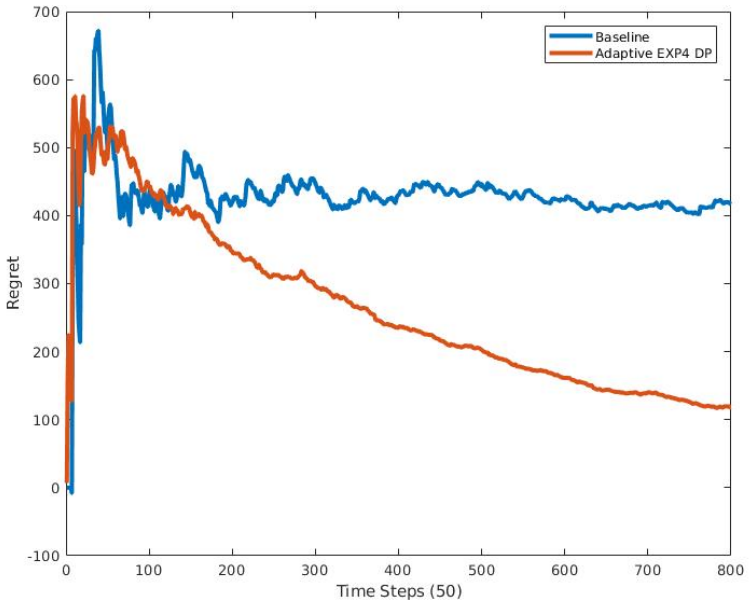


(b) Average regret of the AEXP4 algorithm on a stochastic scene where pedestrian types are randomly generated. The average regret is slowly decreasing over time.

Figure 5.3: Adaptive EXP4 without DPMM

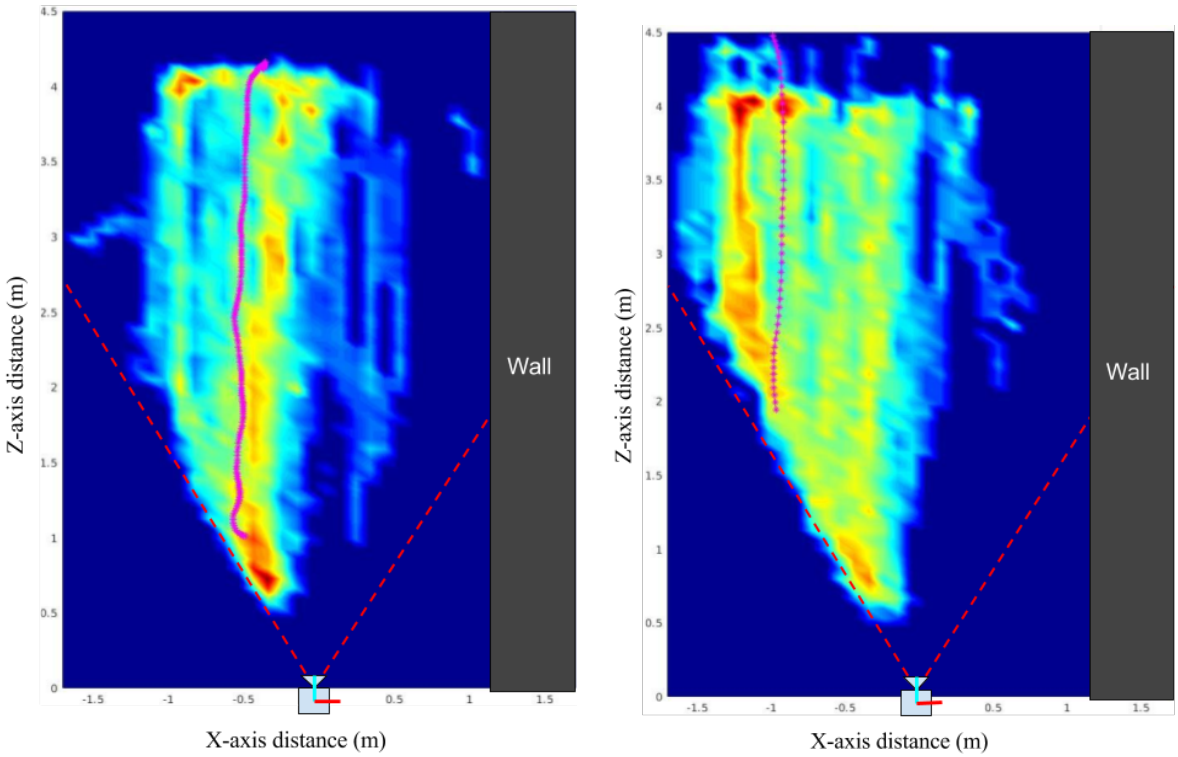


(a) Average accumulated loss of each algorithm over a 40,000 steps testing the AEXP4 algorithm along with DP to reshuffle and bound the expert set.



(b) Average regret of the AEXP4 algorithm on a stochastic scene where pedestrian types are randomly generated. The average regret is decreasing very fast initially and slowly approaches zero over time.

Figure 5.4: Full Adaptive EXP4



(a) Baseline distribution where the robot deploys a default policy and never alarms the pedestrians.

(b) Result distribution of the robot deploy our learning algorithm on the alarm system.

Figure 5.5: Top-down view heatmaps of the pedestrian trajectory distribution. The robot moves on the right side of the hall near a wall with incoming traffic on the left. A sampled trajectory is overlayed over the heatmap for clarity. (An important note: the heatmap is cone shaped due to Kinect's field of view and thus the captured trajectories on the outside are shorter than those in the center.)

robot. We use these attributes to simulate variations in pedestrian behavior.

5.2.2 Simulation Results

To reiterate, the goal of our algorithm is to adapt to variations in pedestrian's reactive behavior. Ideally, our online learner can perform as well on a new pedestrian type in the long run as applying an optimal policy learned offline where the optimal policy is trained with the new pedestrian type beforehand. Therefore, the performance of our algorithm is measured with the notion of *regret* as it is commonly used in online learning literatures. Formally, *regret* is define as:

$$R_t = \sum_{i=0}^{t} l_t(a_t; \theta_t) - _{\theta^*} \sum_{i=0}^{t} l(a_t; \theta^*).$$

In other words, *regret* is the difference of accumulated loss between the performance of the online algorithm (*e.g.* A-EXP4) and the hindsight optimal model (*e.g.* a policy trained with the new

pedestrian type) over time.

For the simulation experiments, we trained two expert policies $\{\pi_1, \pi_2\}$ with two different sets of pedestrians attributes separately offline. Specifically, π_1 is trained with pedestrians who are slow and have a high value for awareness level. π_2 is trained with pedestrians who are fast and have a medium value for awareness level. Both policies are trained using an offline policy gradient algorithm.

During testing, we stochastically generate a new pedestrian behavior type from the start by randomly sampling values for each pedestrian attribute (*e.g.* pedestrians with medium speed and low values for awareness). To analyze our online algorithm, we trained an optimal policy π^* on the new pedestrian type using the same offline policy gradient algorithm to convergence. For the baseline, we will evaluate against the standard EXP4 algorithm which maintains the expert policies statically. Each experiment is repeated 50 trails and the overall average accumulated reward and regret are recorded.

Performance Against EXP4

As shown by our plot in Figure 5.4, our full Adaptive EXP4 algorithm significantly outperforms the standard EXP4 algorithm in average accumulated reward over the span of 40,000 iterations. During the first 5000 iterations of the experiments, our algorithm performs nearly identically to the standard EXP4 algorithm. This is expected because the two algorithms behave similarly during early iterations as they both evaluate the existing expert policies and shift the weights to the better performing one. However, as more exploration policies are sampled over time, AEXP4's performance improves as new policies are inserted into the expert set. In fact, our online learner was able to generate policies competitive with the optimal policy after 10,000 iterations. As the result, average regret of AEXP4 decreases steadily and approaches zero. In contrast, the standard EXP4 algorithm maintains a large average regret because neither pre-trained expert policies were effective against the new pedestrian type.

Performance Without Policy Reshuffling

In addition to comparing AEXP4 with the baseline EXP4 algorithm, we want to demonstrate the importance of using DPMM to reshuffle the expert policies. As illustrated by the Figure 5.3, while the average performance of AEXP4 without policy reshuffling is still marginally better than the standard EXP4 in the long run, the performance variance is significantly higher than the full AEXP4 algorithm.

Without reshuffling the expert policies, the set of expert policies becomes unbounded. This hampers the performance because poorly performing policies will never be removed from the set. Therefore, as the expert policies accumulate, it become increasingly more difficult to sample better ones. We observe that the performance of this AEXP4 algorithm largely depends on the quality of first few sampled policies. By clustering similar policies and removing ones with low weights, we can ensure that the algorithm only samples near policies with high expected reward.

5.2.3 Experiment with CaBot

Platform

We built and implemented the online learning algorithm onto our blind navigation assistant robot CaBot. CaBot is a suitcase robot designed to help blind individuals navigate through crowded scenes such as airports. CaBot is equipped with a Microsfot Kinect sensor for as its primary sensor. We use the Kinect to extract various scene information including obstacles positions, poses, and velocity of incoming pedestrians. Moreover, we utilized the skeleton tracker and combine it with a standard Kalman filter to estimate the full pedestrian trajectory information. The output of our computer vision pipeline are aggregated into a state vector as described in the Approach section above.

Pedestrian Trajectory

To demonstrate the overall effect of our approach on the incoming pedestrians with respect to the navigation task, we tested the CaBot on a simple, straight forward route through a long hallway. For the baseline, the CaBot passively navigates through defined route without any audio warnings. The experiment is then repeated with our adaptive learning system enabled. We initialized our learner with a trivial default policy (0 for all policy parameter values) which never plays any warning sounds. We then allowed the learner to iterate over 150 pedestrian trajectories.

In each experiment, all 150 trajectories of the surrounding pedestrians are recorded. The result is illustrated as a heatmap in Figure 5.5. In the baseline trial 5.5a, a large percentage of the incoming pedestrians persisted on their route and walk closely by the robot without trying to actively avoid it. Although the robot rarely directly collides with anyone during our experiment, it must stayed on course accurately leaving a low margin for controller and user error. In contrast, in our experimental trial 5.5b, we observed that our learner was able to generate policies based on pedestrian distance and velocity. As the result, fast moving pedestrians avoided the robot much earlier and actively stay away from its course.

5.3 Conclusion

In this project, we presented a method for dealing with pedestrian behavior variations in blind navigation robots using a novel online learning algorithm. This method relies on maintaining and adjusting weights over a set of expert policies. Furthermore, we dynamically search new policies online and group them into the expert set using a Bayesian mixture model. Our experimental results shows that it has better performance than using a static set of expert policies. We believe that our proposed method can be utilized to other transfer learning or lifelong learning tasks.

In the future, we would like to expand the state and reward spaces to include feedback from the blind user which is being guided by CaBot. This way, CaBot's behavior can be modified in accordance to the user preferences as well, and not just social acceptability of surrounding pedestrians. In terms of algorithmic future directions, we would like to study the role of the proposed algorithm in adaptive to other types of policies (Neural Networks). However, there are still many limitations to the algorithm hindering it from generalizing to more complex control

policies. First, the `PolicyUpdate` algorithm has a high variance. In the algorithm, we obtain a rough reward estimate of the exploration policies used in `PolicyUpdate` from a single roll out. While this was compensated by sampling more exploration policies in our experiments, it would become more unreliable with complex policies. Secondly, additional policy parameters significantly expand the search space and thus require more iterations to sample exploration policies. As the result, it will become increasingly more difficult to both sample and generate good optimal policies in high dimensional policy parameter space. One interesting area for improvement would be exploring Thompson sampling [43] or other statistical sampling methods to improve the policy search efficiency in higher dimensions.

Bibliography

[1] Dragan Ahmetovic, Cole Gleason, Chengxiong Ruan, Kris M Kitani, Hironobu Takagi, and Chieko Asakawa. Navcog: a navigational cognitive assistant for the blind. In *MobileHCI*, pages 90–99, 2016. 4

[2] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–971, 2016. 4

[3] Marwah M Almasri, Abrar M Alajlan, and Khaled M Elleithy. Trajectory planning and collision avoidance algorithm for mobile robotics system. *IEEE Sensors Journal*, 16(12): 5021–5028, 2016. 4

[4] Masatoshi Arikawa, Shin'ichi Konomi, and Keisuke Ohnishi. Navitime: Supporting pedestrian navigation in the real world. *IEEE Pervasive Computing*, 6(3), 2007. 4

[5] Akbar Assa and Farrokh Janabi-Sharifi. A robust vision-based sensor fusion approach for real-time pose estimation. *IEEE transactions on cybernetics*, 44(2):217–227, 2014. 2

[6] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002. 4

[7] Filippo Bergamasco, Andrea Albarelli, Emanuele Rodola, and Andrea Torsello. Rune-tag: A high accuracy fiducial marker with strong occlusion resilience. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 113–120. IEEE, 2011. 1.1, 2

[8] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Robotics-DL tentative*, pages 586–606. International Society for Optics and Photonics, 1992. 2

[9] Alina Beygelzimer, John Langford, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual bandit algorithms with supervised learning guarantees. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 19–26, 2011. 4

[10] David M Blei, Michael I Jordan, et al. Variational inference for dirichlet process mixtures. *Bayesian analysis*, 1(1):121–143, 2006. 5.1.4

[11] Danilo Bruno, Sylvain Calinon, and Darwin G Caldwell. Bayesian nonparametric multi-optima policy search in reinforcement learning. In *AAAI*, 2013. 4, 5.1.4

[12] Sylvain Calinon, Affan Pervez, and Darwin G Caldwell. Multi-optima exploration with

adaptive gaussian mixture model. In *Development and Learning and Epigenetic Robotics (ICDL), 2012 IEEE International Conference on*, pages 1–6. IEEE, 2012. 4

[13] Christian Gloor. Pedsim: Pedestrian crowd simuulation. URL `http://pedsim.silmaril.org/`. 5.2.1

[14] Sakmongkon Chumkamon, Peranitti Tuvaphanthaphiphat, and Phongsak Keeratiwintakorn. A blind navigation system using rfid for indoor environments. In *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2008. ECTICON 2008. 5th International Conference on*, volume 2, pages 765–768. IEEE, 2008. 4

[15] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011. 4

[16] Daniel DeMenthon and Larry S Davis. Exact and approximate solutions of the perspective-three-point problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14 (11):1100–1105, 1992. 2.1

[17] Anca D Dragan Dorsa Sadigh, Shankar Sastry, and Sanjit A Seshia. Active preference-based learning of reward functions. In *Robotics: Science and Systems (RSS)*, 2017. 4

[18] David W Eggert, Adele Lorusso, and Robert B Fisher. Estimating 3-d rigid body transformations: a comparison of four major algorithms. *Machine vision and applications*, 9(5-6): 272–290, 1997. 3.1.2

[19] Mark Fiala. Artag, an improved marker system based on artoolkit. *National Research Council Canada, Publication Number: NRC*, 47419:2004, 2004. 1.1, 2

[20] Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco José Madrid-Cuevas, and Manuel Jesús Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014. 2

[21] O Serdar Gedik and A Aydin Alatan. Rgbd data based pose estimation: Why sensor fusion? In *Information Fusion (Fusion), 2015 18th International Conference on*, pages 2129–2136. IEEE, 2015. 2

[22] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6):721–741, 1984. 5.1.4

[23] Daniel Grest, Thomas Petersen, and Volker Krüger. A comparison of iterative 2d-3d pose estimation methods for real-time applications. In *Scandinavian Conference on Image Analysis*, pages 706–715. Springer, 2009. 2

[24] Shunsuke Hamasaki, Yusuke Tamura, Atsushi Yamashita, and Hajime Asama. Prediction of human's movement for collision avoidance of mobile robot. In *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, pages 1633–1638. IEEE, 2011. 4

[25] Bert M Haralick, Chung-Nan Lee, Karsten Ottenberg, and Michael Nölle. Review and analysis of solutions of the three point perspective pose estimation problem. *International journal of computer vision*, 13(3):331–356, 1994. 2.1

[26] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cam-

bridge university press, 2003. 2.1

[27] Radu Horaud, Bernard Conio, Olivier Leboulleux, and Bernard Lacolle. An analytic solution for the perspective 4-point problem. *Computer Vision, Graphics, and Image Processing*, 47(1):33–44, 1989. 2.1

[28] Hirokazu Kato. Artoolkit: library for vision-based augmented reality. *IEICE, PRMU*, 6: 79–86, 2002. 2

[29] Kris M Kitani, Brian D Ziebart, James Andrew Bagnell, and Martial Hebert. Activity forecasting. In *European Conference on Computer Vision*, pages 201–214. Springer, 2012. 4

[30] Jens Kober and Jan R Peters. Policy search for motor primitives in robotics. In *Advances in neural information processing systems*, pages 849–856, 2009. 4, 5.1.4

[31] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013. 4

[32] Aditi Kulkarni, Allan Wang, Lynn Urbina, Aaron Steinfeld, and Bernardine Dias. Robotic assistance in indoor navigation for people who are blind. In *The Eleventh ACM/IEEE International Conference on Human Robot Interaction*, pages 461–462. IEEE Press, 2016. 4

[33] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016. 4

[34] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015. 4

[35] Jack M Loomis, Reginald G Golledge, and Roberta L Klatzky. Navigation system for the blind: Auditory display modes and guidance. *Presence*, 7(2):193–203, 1998. 4

[36] Wei-Chiu Ma, De-An Huang, Namhoon Lee, and Kris M Kitani. Forecasting interactive dynamics of pedestrians with fictitious play. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 774–782, 2017. 4

[37] H Brendan McMahan and Matthew J Streeter. Tighter bounds for multi-armed bandits with expert advice. In *COLT*, 2009. 4

[38] Leonid Naimark and Eric Foxlin. Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker. In *Mixed and Augmented Reality, 2002. ISMAR 2002. Proceedings. International Symposium on*, pages 27–36. IEEE, 2002. 2

[39] Gergely Neu. Explore no more: Improved high-probability regret bounds for non-stochastic bandits. In *Advances in Neural Information Processing Systems*, pages 3168–3176, 2015. 4

[40] Chuong V Nguyen, Shahram Izadi, and David Lovell. Modeling kinect sensor noise for improved 3d reconstruction and tracking. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*, pages 524–530.

IEEE, 2012. 3.1.1

[41] Scott Niekum. Ar track alvar ros package. URL `http://wiki.ros.org/ar_track_alvar`. 3.2.4

[42] Edwin Olson. Apriltag: A robust and flexible visual fiducial system. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3400–3407. IEEE, 2011. 1.1, 2

[43] Ian Osband and Benjamin Van Roy. Bootstrapped thompson sampling and deep exploration. *arXiv preprint arXiv:1507.00300*, 2015. 5.3

[44] Kaustubh Pathak, Narunas Vaskevicius, and Andreas Birk. Uncertainty analysis for optimum plane extraction from noisy 3d range-sensor point-clouds. *Intelligent Service Robotics*, 3(1):37–48, 2010. 3.1.1

[45] Jan Peters and Stefan Schaal. Policy gradient methods for robotics. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 2219–2225. IEEE, 2006. 4

[46] Lisa Ran, Sumi Helal, and Steve Moore. Drishti: an integrated indoor/outdoor blind navigation system and service. In *Pervasive Computing and Communications, 2004. PerCom 2004. Proceedings of the Second IEEE Annual Conference on*, pages 23–30. IEEE, 2004. 4

[47] Andrew C Rice, Robert K Harle, and Alastair R Beresford. Analysing fundamental properties of marker-based vision system designs. *Pervasive and Mobile Computing*, 2(4):453–471, 2006. 2

[48] Dorsa Sadigh, S Shankar Sastry, Sanjit A Seshia, and Anca Dragan. Information gathering actions over human internal state. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 66–73. IEEE, 2016. 4

[49] Frank Sehnke, Christian Osendorfer, Thomas Rückstieß, Alex Graves, Jan Peters, and Jürgen Schmidhuber. Parameter-exploring policy gradients. *Neural Networks*, 23(4):551–559, 2010. 4

[50] Daniel Steinberg. An unsupervised approach to modelling visual data. 2013. 5.1.4

[51] Vasilis Syrgkanis, Akshay Krishnamurthy, and Robert Schapire. Efficient algorithms for adversarial contextual learning. In *International Conference on Machine Learning*, pages 2159–2168, 2016. 4

[52] Nikos Vlassis, Marc Toussaint, Georgios Kontes, and Savas Piperidis. Learning model-free robot control by a monte carlo em algorithm. *Autonomous Robots*, 27(2):123–130, 2009. 4

[53] John Wang and Edwin Olson. Apriltag 2: Efficient and robust fiducial detection. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 4193–4198. IEEE, 2016. 2

[54] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992. 4

[55] Cang Ye, Soonhac Hong, and Xiangfei Qian. A co-robotic cane for blind navigation. In *Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on*, pages 1082–

1087. IEEE, 2014. 4

[56] Lingqi Zeng and Gary M Bone. Mobile robot collision avoidance in human environments. *International Journal of Advanced Robotic Systems*, 10(1):41, 2013. 4

[57] Cai-Xia Zhang and Zhan-Yi Hu. A general sufficient condition of four positive solutions of the p3p problem. *Journal of Computer Science and Technology*, 20(6):836–842, 2005. 2.1

[58] Brian D Ziebart, Nathan Ratliff, Garratt Gallagher, Christoph Mertz, Kevin Peterson, J Andrew Bagnell, Martial Hebert, Anind K Dey, and Siddhartha Srinivasa. Planning-based prediction for pedestrians. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 3931–3936. IEEE, 2009. 4