ROBOTIC PERSON-FOLLOWING IN CLUTTERED ENVIRONMENTS

WILLIAM R. KULP

Submitted in partial fulfillment of the requirements

for the degree of Master of Science

Department of Electrical Engineering and Computer Science

Case Western Reserve University

June, 2012

Case Western Reserve University

School of Graduate Studies

We hereby approve the thesis/dissertation of

William R. Kulp

candidate for the Master of Science degree *.

(signed)_____
(chair of the committee)

_____

_____

(date)_____

* We also certify that written approval has been obtained for any proprietary material contained therein.

Table of Contents

List of Tables

List of Figures

Acknowledgements

I would like to thank my academic advisor, Dr. Wyatt Newman, for his tireless mentoring since freshman year. Thanks for advising me as an undergraduate, for steering me into the B.S./M.S. program, and of course for all of the help that has gone into this thesis. Thanks to previous graduate students at Case, especially Chad Rockey and Eric Perko, who designed Harlie, wrote many of Harlie's software packages, and provided great help with ROS. Finally, I will always be grateful to my parents for constantly being there for me with encouragement and support for over 21 years.

ROBOTIC PERSON-FOLLOWING IN CLUTTERED ENVIRONMENTS

Abstract

WILLIAM R. Kulp

This thesis shows the development of hardware and software elements to enable a mobile robot to follow a target person through an obstacle-filled room. Person-tracking is performed via fusion of information from the Microsoft Kinect and other sensors. A dynamic replanning system was created to allow the robot to track moving targets.

1 Introduction and Background

Human-machine interaction is a highly active area of research in the field of robotics. A fundamental task is the ability to detect and follow a human. This deceptively simple task could open a huge range of possibilities. A smart wheelchair could automatically follow a nurse through a hallway [1]. A pack robot could follow a soldier into combat. A tourguide robot guiding a group of people through a museum would face similar challenges in maintaining recognition and navigating safely in the presence of obstacles and people.

Although person tracking comes naturally to humans, it is a highly nontrivial job for a machine, requiring the integration of many unreliable sources of information and the creation a model of the environment from changing conditions. Humans have wide variation in size, shape, and colors, and their appearances change over time with changes in posture and lighting. The background of a typical scene contains a great deal of clutter in shape, texture, and color. When the robot is in motion, it becomes difficult to separate the target's motion from background motion. Additionally, a method must be developed to allow the robot to plan to a moving target under changing conditions. Recently, much research has focused on the task of person following from a mobile robot. The remainder of this chapter will provide an overview of the current technologies.

1.1 Computer Vision

Cameras and computer vision algorithms are commonly used on mobile robots for the purpose of person-tracking. High-resolution color cameras are inexpensive and vision-based tracking is intuitive to humans. Computer vision algorithms are very diverse, and this section will touch on a few of the most popular approaches, specifically for people tracking.

Some vision systems require the user to face the camera to provide a consistent view. A frontal view of the user is also necessary if face detection is desired. Face detection is sometimes used to re-initialize a person-tracking system after occlusion or target loss [2] because human faces are highly distinctive and face recognition is a reasonably mature technology [3]. While requiring the target to face the robot may work for some applications, such as a tourguide robot, in general it is an inconvenient burden to place on a person-following system.

Many vision systems rely on color information and occasionally texture information [4]. These properties are readily accessible from cameras and intuitive to humans. Some of the simplest tracking approaches simply look for solid regions of a certain color. Calisi *et al.* used a single-color segmentation algorithm assisted by stereo depth information to track a user wearing a single-colored shirt [5]. While methods that rely on color alone are simple and computationally efficient, they are restricted to cases in which the target is wearing a solid color and that color is uncommon in the environment. More complicated approaches may track areas of multiple colors, or compute a color

histogram for an area of interest [6]. Skin colors are also frequently used to identify regions of interest.

Going beyond color, many authors have taken the approach of identifying certain shapes relating to human bodies. Shape information may be computed for both 2D and 3D images [7], and is often done using cascades of Haar-like features [8]. Some authors use part-based representations that combine multiple classifiers for different parts of the body [9]. Such systems combine many weak classifiers to create a strong classifier, possibly using a training algorithm such as AdaBoost (Adaptive Boosting) [10].

Some other vision-based approaches rely on the detection of keypoints. A keypoint represents a distinctive, salient geometric feature of an object. A number of popular algorithms including SIFT (Scale Invariant Feature Transform) [11], SURF (Speeded Up Robust Features) [12], and HOG (Histogram of Oriented Gradients) [7] detect scale-invariant keypoints, providing the ability to detect objects consistently at varying ranges and orientations. These algorithms are commonly used for object detection and prove useful in real-world scenes where people appear at multiple ranges and orientations. Keypoints may also be related to a higher-order part-based model. Seemann *et al.* evaluated various keypoint detectors trained on images to identify and tag pedestrians in a scene [13].

Binocular, or stereo cameras, are increasingly common on mobile robots. By computing disparity between the left and right images, stereo cameras can estimate the depth of points in 3D space and create a depth image. The process is

similar to how the human brain processes depth. Depth information can be greatly helpful in segmenting targets from the background [7], especially targets that are free-standing. Omnidirectional cameras are sometimes used on mobile robots, as in Kobilarov et al. [6]. This type of camera uses a special optical system that can view targets all around the robot, although omnidirectional cameras often have issues with distortion and limited resolution.

Bajracharya *et al.* [14] segmented pedestrians based on range data from a stereo camera setup. They down-projected 3D range data onto a 2D ground plane, looking for large accumulations of pixels that corresponded to upright objects in 3D space. They used 3D geometric features and color information to classify the resultant blobs as people. Although this method worked well outdoors, methods that rely on down-projection can be confused in indoor environments where ceilings, doorframes, and other upright objects are in the robot's field of view. Miura and Satake [15] took a different approach with a stereo camera system, using template matching on a depth image. They used depth templates with a support vector machine (SVM) classifier to detect the distinctive shape of a person's head and shoulders.

Optical flow is sometimes used for person tracking [16], although it is very difficult to calculate optical flow while compensating for the motion of a mobile robot. Jung and Sukhatme [17] attempted to do so by estimating the egomotion of the robot and compensating for this frame-to-frame by using a projective transform. This method breaks down if the robot moves quickly or if the robot's motion is not free of bumps.

The Microsoft Kinect is a more recent innovation that combines a traditional RGB camera with an IR depth camera. The Kinect provides capabilities similar to a high-end stereo vision system [18] at a fraction of the cost using different technology. Numerous authors have used the Kinect's depth sensing capabilities as a replacement for a stereo camera, for static surveillance [19] or as a depth sensor from a mobile robot.

The Kinect's developers provide an API for the built-in skeleton-tracking software designed to track users from a fixed position, enabling the Kinect's use as a game controller. Some papers have explored these built-in skeleton tracking facilities from a static viewpoint [20], although attempts have not been made in the literature to evaluate the Kinect's person-tracking capabilities from a mobile robot.

1.2 Other sensors

Because the performance of vision systems depends on viewing angle and lighting conditions, they are often supplemented by other sensors. Stereo microphones may be helpful in detecting the location of a speaking person. Sonar sensors can provide range data at a low cost, although their spatial resolution is extremely course. Some systems have used active RFID [2] or IR beacons, although these require the user to wear specialized equipment which is undesirable.

Besides cameras, LIDAR (LIght Detection And Ranging) units are also frequently used for person tracking. LIDAR units work by measuring the time of

flight of a laser beam swept in an arc, producing a precise 2-dimensional polar slice of obstacles around the robot. LIDAR units such as those made by the German company SICK tend to be expensive, on the order of several thousand dollars. Recently, less expensive units have been produced, such as the unit developed for the Neato XV-11 vacuum cleaner [1].

For the purpose of tracking people, LIDAR units may be mounted at hip height, creating a single blob per person, or below knee height [21], creating a blob for each leg. Geometric features can be calculated from the scan data and these features can be run through a classifier to detect people [21]. Using an adaptive algorithm such as AdaBoost [22], such a classifier can automatically be created from scan data. LIDAR units have been used both for person detection from a mobile robot and for static surveillance from a fixed point [6].

Laser rangefinders have a very wide field of view, although they have a limited resolution on the order of one reading per degree. LIDAR units perform best when the person is up close and the unit can record many laser returns per leg [22]. More laser returns result in a more accurate calculation of geometric features. The performance drops off with distance: after several meters, a human leg may only register several laser returns, in which case classification is highly error-prone. Additionally, 2D range sensors have no way of distinguishing one person from another. Therefore 2D range sensors are rarely used on their own, usually augmenting vision-based methods.

1.3 Sensor fusion

When using multiple sensors, a method is needed to combine disparate measurements into a unified estimate of the user's position. Most person tracking systems incorporate a probabilistic model such as a Kalman filter or a particle filter [9] [15]. When the system receives measurements, the measurements are associated with the filter's latest position estimate by distance or other criteria [23]. Successfully associated measurements are used to update the filter's state. In the case of tracking multiple people, joint probabilistic data association filters (JPDAFs) [23] have seen successful, providing a probabilistic framework to associate measurements with multiple targets. These filters are useful for tracking multiple people or planning around the movement of pedestrians.

Even when a person-tracking system only uses one source of information, sensor fusion algorithms are still popular as a way to provide consistency over time. Instead of tracking the person by detection in each frame, a filter can maintain an estimate of the user's position and use this estimate to weed out false positives and protect against momentary target loss. This method proves useful when multiple targets are in the scene with similar appearances. Filters can also be used to reduce a system's computational load. Instead of running a person detector on the entire scene, a filter can focus the detection effort on regions of interest near the last known location of the person [7].

1.4 Planning

Motion planning for mobile robots is a very difficult problem. Computationally, motion planning is NP-hard, and has been the subject of research for over 30 years. Due to the problem's NP-hardness, many heuristic approaches have been developed [24] because an exact, optimal solution may be unfeasible.

The simplest person-following algorithms do not use planning at all, but rely on simple control algorithms. Such algorithms usually command a velocity sufficient to maintain a following distance and to minimize bearing between the robot and target [6]. Control algorithms are not robust enough for use in real-world environments, and are mainly used when the objective is to demonstrate a person-following system rather than spend time implementing true motion planning.

There are a wide range of classical approaches to motion planning that rely on so-called configuration space (C-space) [25]. A map in C-space represents all valid positions, or configurations, of the robot. Obstacles are "inflated," so that given a point on the obstacle map, if that point is free, it represents a valid pose for the robot. Thus, in C-space, planning is greatly simplified because the robot can be treated as a single point and motion planning reduces to a 2-dimensional problem [26]. Configuration space has some drawbacks resulting from the assumption that the robot can be treated as a point. For non-circular robots, the C-space map is dependent on orientation, adding an additional dimension to the

problem. C-space has difficulties in representing non-holonomic robots because the set of actions available to the robot are dependent on the robot's state.

From configuration space, a number of classical algorithms may be applied including roadmaps, wavefront planning, repulsive/attractive forces [27], Voronoi diagrams, visibility graphs, and cell decomposition [26]. While most classical methods are intuitive, they are usually unable to perform complex, multi-stage moves such as three-point-turns. Classical methods may become stuck at local minima. Classical planning methods are especially risky when used on non-holonomic robots.

2 Experimental Mobile-Robot System: Harlie

Harlie, pictured in Figure 1, is a mobile robot designed and constructed at Case Western Reserve University. Harlie is built on an electric wheelchair base which allows for operation on varied terrain, both indoors and outdoors. The robot carries an inverter and several lead-acid batteries internally and is capable of running for several hours on its own power. Harlie has been used for several projects in the past, including an entry in the annual Intelligent Ground Vehicle Competition (IGVC) where the goal was to create an autonomous robot to navigate an outdoor obstacle course.

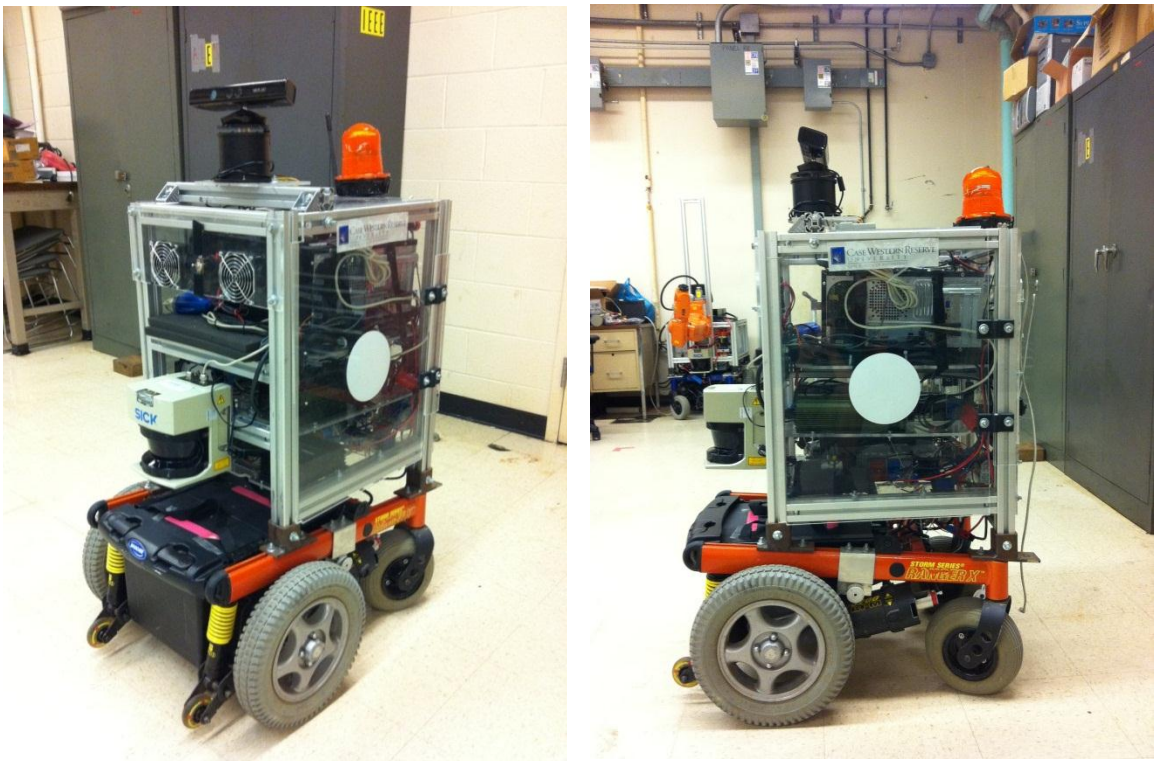

Figure 1: Harlie, the mobile robot

Harlie carries an assortment of sensors. A LIDAR unit mounted on the robot's front is used for obstacle detection and localization within a static map. The LIDAR unit has a 180-degree field of view and a resolution of one reading per degree. Optical encoders are used on each wheel to provide accurate odometery information. A Microsoft Kinect sensor on a rotating mount, added for this project, is used for person-tracking. Harlie additionally carries a GPS unit, a magnetometer, an inertial measurement unit, and several sonar sensors, which were not needed for this project.

Harlie was equipped with a server containing a 2.67 GHz Intel Core i7 920 CPU and 6 GB of RAM. The server communicated with an onboard National Instruments CompactRIO for real-time, low-level control of peripherals including motors and encoders. Additional processing power was provided by a Dell Latitude E6510 laptop which was mounted on Harlie and connected to the server via Ethernet. The laptop contained a 2.67 GHz Intel Core i5 560M CPU and 4GB of RAM.

Processing was shared between the two computers. Harlie's server ran software related to path planning, steering, and localization within a static map. The laptop ran software related to the Microsoft Kinect and person-tracking. Additionally, the laptop was used to interact with the server and monitor the status of the robot.

All software was developed for Ubuntu Linux using the open-source ROS (Robot Operating System) framework hosted by Willow Garage [28]. ROS provides facilities for different software modules, called nodes, to communicate

by sending user-defined messages. ROS is highly flexible and modular, and simplifies tasks such as connecting processes running on separate machines. ROS is designed in such a way that nodes and subsystems can easily be evaluated and reused between projects. ROS has a large, active user community with many open-source, high-quality code modules.

Figure 2 provides a high-level overview of this project's software architecture. Robot localization within a static map was performed by the ROS AMCL (Adaptive Monte-Carlo Localization) package [29]. Steering was performed by the Precision Steering package [30] which was created by Eric Perko, a fellow graduate student at Case Western Reserve University. The person-tracking and planning modules were designed for this project and elaborated upon in their respective chapters.

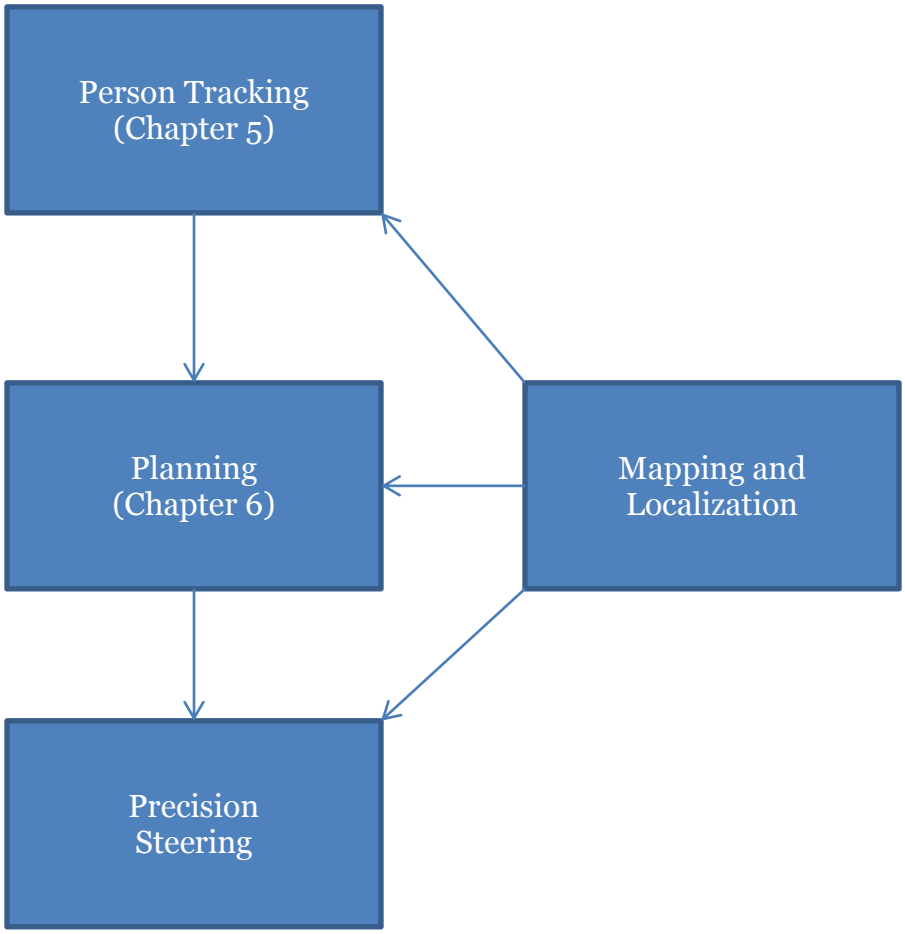

Figure 2: Overall software architecture

3 Evaluation of the Microsoft Kinect

The Microsoft Kinect, released in the year 2010, is a human interface device originally developed for the Microsoft Xbox to facilitate gestural controls and natural user interaction. When used as a game controller, the Kinect is able to track the positions of multiple users in real time. The Kinect provides the Xbox with the users' locations in 3D space as well as the positions and orientations of their limbs. Games may use this data to provide an interactive, intuitive user interface, similar to the concept of the Nintendo Wii remote and the Sony PlayStation Move.

Internally, the Kinect carries a 640x480 RGB camera and a 640x480 infrared camera. An infrared projector shines a known dot pattern through a diffraction grating, and by computing disparity between the known pattern and what is observed from the infrared camera, a depth value can be computed for any given pixel. This gives the Kinect great utility as a 3D sensing system. The retail price of $150 prices the Kinect far below comparable systems on the market. The Kinect provides the capabilities of a high-end stereo vision system at a fraction of the cost [18].

PrimeSense, the Israeli company that develops the Kinect's 3D sensing software, has released an open-source API called OpenNI (Open Natural Interaction) to allow developers to tap into the Kinect's functionality [31]. In addition to accessing the raw depth and RGB camera feeds, PrimeSense provides high-level functionality for tracking user skeletons through a library called NiTE

[32]. With OpenNI and NiTE, the Kinect is able to detect and track multiple human users in its field of view, which is appealing for the application of person tracking.

Livingston *et al.* evaluated the skeleton tracking capabilities of the Kinect from a static viewpoint [20] with the goal of using the Kinect in a virtual reality control interface. They found that the Kinect can track users from 4m to just under 1m distance, although the performance at these extremes was erratic and sensor noise increased as a function of distance. Microsoft recommends an operational range of 1.2-3.5 meters.

No published attempts were found evaluating the Kinect's person-tracking capabilities from a mobile robot. For this project, it was necessary to mount the Kinect on Harlie, a moving platform. The Kinect is remarkably proficient at its intended task, although when mounted on Harlie, the Kinect is operating outside of its design parameters.

Several major challenges were identified that had to be overcome. The Kinect's limited field of view (57 degrees) poses a challenge when following users through an environment. Also, when a new user enters the scene, the user must make a calibration pose before tracking can begin. By default, the Kinect has no means of telling one specific user from another, relying on spatial and temporal continuity to tell users apart. Finally, the Kinect has difficulties when used from a mobile vantage point, being susceptible to bumps and sudden motions.

Possibly, these issues could have been dealt with by patching the skeleton tracking software. Unfortunately, NiTE is distributed as a closed-source binary and there are few options to probe the library's inner workings. Consequently, higher-level software workarounds had to be employed to make up for some shortcomings of NiTE.

3.1 Calibration of Users

By default, whenever the Kinect detects a new user in its field of view it requires the user to stand in a "psi" calibration pose to acquire an accurate measure of the user's limbs. This calibration step takes several seconds and requires both the target and the camera to be still.

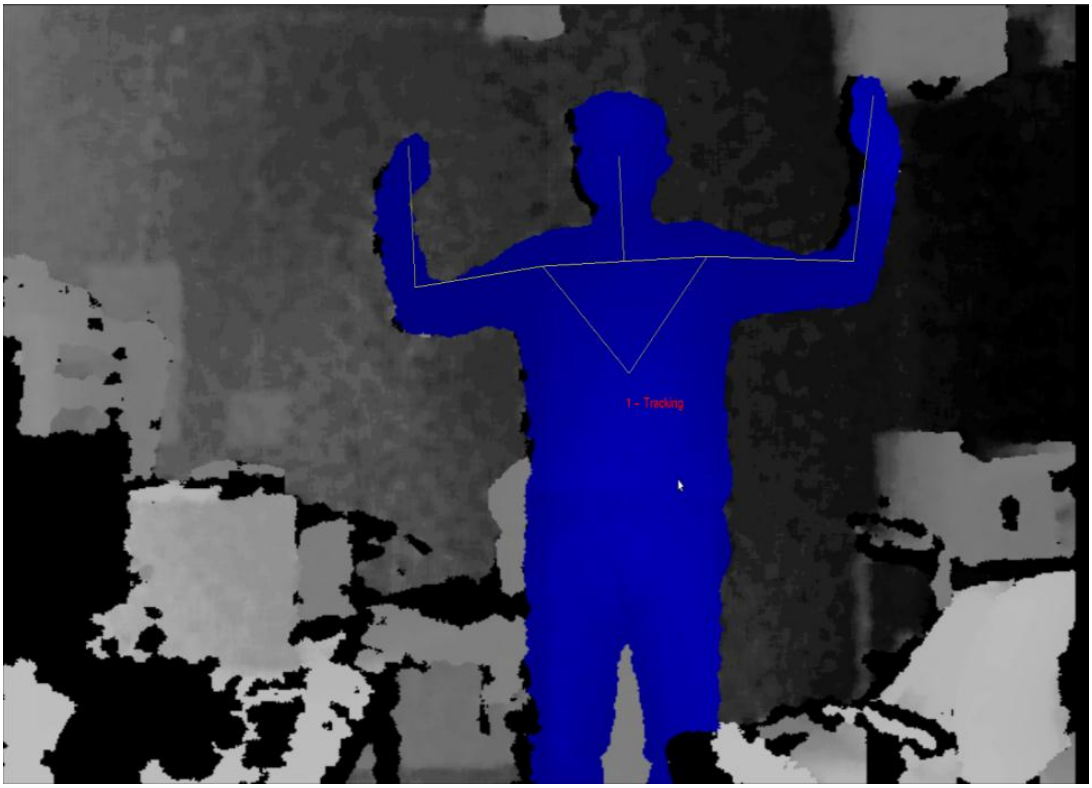

Figure 3: Kinect's distinctive "psi" calibration pose

With the Kinect on a moving base, occasionally the target will be lost due to relative motion or sudden jolts, as discussed later. Upon target reacquisition, frequently the software will not remember the user and will require recalibration. Recalibration would require both Harlie and the target to come to a halt, which is onerous given the goal of smoothly following the target. Luckily, through somewhat of a hack, OpenNI can be instructed to save the calibration of the first detected user. This saved calibration can be applied to all subsequent users.

Skipping the calibration step comes at a cost. The distinctive "psi" pose required for calibration greatly reduces the possibility of the robot following the wrong user. It is highly unlikely that a bystander would make the "psi" pose. Without the calibration step, Harlie no longer has an easy way of telling which user to track. Furthermore, when on a moving base, the Kinect tends to misclassify some inanimate objects, such as chairs, as users. These chairs would never pass the calibration step, although without calibration they may appear as spurious person measurements.

3.2 Discrimination Between Users

A challenge for people-following applications is that the Kinect does not do a good job of discriminating between users. While the Kinect has the potential to store color and texture information to recognize individuals, in practice, once OpenNI calibrates on a user, no information is stored other than limb measurements. As a result, if a user exits the scene, there is no guarantee that when the user is re-detected that OpenNI will assign that user the same ID. The

same is true if a target is momentarily lost due to a sudden bump or relative motion.

For its intended application as a game controller, where players never leave the field of view and the Kinect is stationary, the target lock is rarely broken. However, for applications where the Kinect itself is moving, frequent dropouts must be addressed. This project's solution, as detailed in Chapter 5.3 is to use the Kinect as one of several inputs to a Kalman filter that tracks the overall hypothesized location of a person, as well as to store a unique fingerprint of the tracked user's color information.

3.3 Limited Field of View

The Kinect has a field of view of 57 degrees. While this is sufficient for tracking a target with limited freedom from a fixed vantage point, it shows weaknesses for moving targets. When using the Kinect as the sole source of observation, Harlie must constantly face the user (within ±29 degrees) or lose the target. This puts severe constraints on the ability to maneuver and plan paths while maintaining contact with the target.

Even a task such as following a target down a straight hall can be problematic. If an obstacle appears between the user and the robot, the robot must navigate around the obstacle. As part of the obstacle avoidance, the robot will likely rotate far enough that the user leaves the Kinect's field of view, leading to loss of the target (Figure 4.) When the robot once again faces the user, it will have to re-acquire the target, leading to a delay.

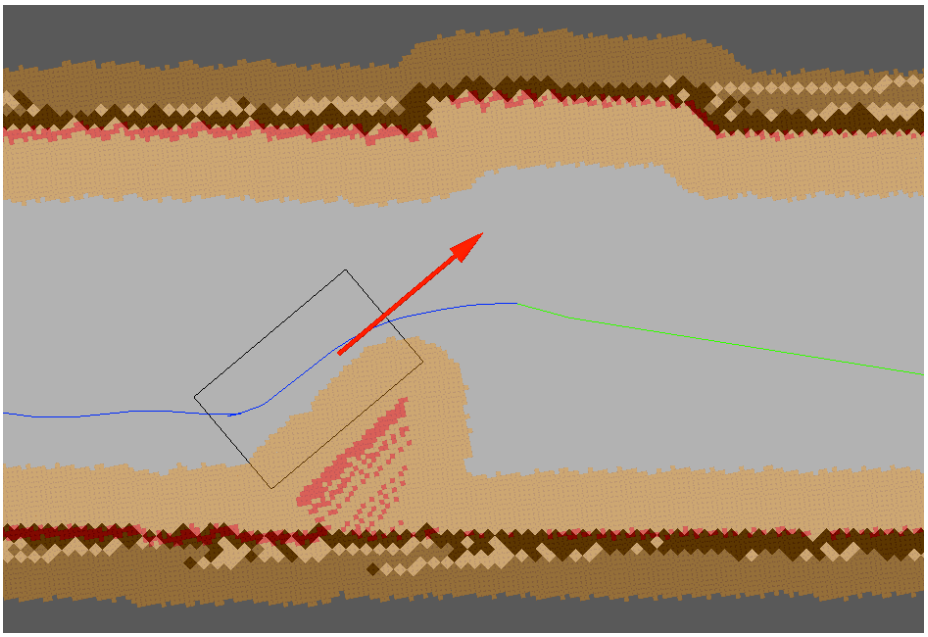

Figure 4: Obstacle avoidance may lead to target loss due to Kinect's limited field of view

The situation becomes even worse if the user doubles back behind the robot. In tight spaces such as hallways, the user must come close to Harlie when moving behind it. The Kinect's depth camera breaks down when targets are closer than 2 feet away, so Harlie effectively has a blind spot for close objects. In a hallway scenario, this can result in Harlie being stuck pointing at close range to a wall within the blind-spot range.

As an additional issue with OpenNI, the default behavior of the software is to track the entire human body (head, arms, torso, and legs). Full-body tracking is desirable for the Kinect's intended application as a game controller, although Harlie's Kinect is mounted in such a way that users' legs are often obscured. Luckily, OpenNI can be instructed to ignore users' legs and just track the target from the waste up. This results in better tracking from Harlie's point of view, but results in an additional tradeoff. Without the shape cues that legs provide, the

tracking software loses an important characteristic that can discriminate people from inanimate objects (Figure 5.)

These issues introduced by skipping calibration are resolved in Chapter 5 by treating the bodies detected with OpenNI as one input to an overall Kalman filter and adding a "fingerprint" to uniquely identify a user.

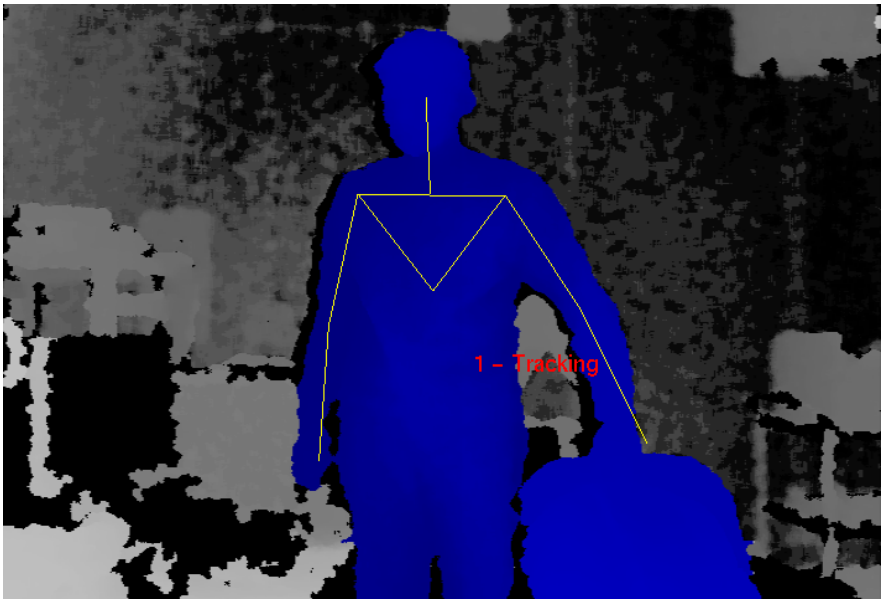

Figure 5: Difficulties arise in tracking a user in contact with a chair

3.4 Moving Base Problem

The Kinect was designed to be placed in front of a television to track users playing a video game. Mounting the Kinect on Harlie's moving base poses challenges outside of the Kinect's design parameters. Tests were therefore performed to ensure that the Kinect was suitable for such use. A walking pace for an average human is around 1 m/s, and for decent maneuverability, Harlie should be able to navigate curves with a radius of 1m. Thus, by informal

calculation, Harlie should be able to handle peak angular speeds of 1 radian/second.

The Kinect is a complicated system and the tracking software is closed-source, so it was difficult to exactly characterize the system's performance. However, some metric of performance was necessary. A test was performed in which Harlie was rotated back and forth through 1 radian of angle (slightly less than the Kinect's FOV) with a sinusoidal velocity profile. The Kinect attempted to track a person standing 2m away shifting his weight from foot to foot (corresponding to 20cm of motion at 1Hz). If the Kinect performed perfectly, it would have maintained a lock on the user 100% of the time. In reality, the Kinect periodically dropped the user due to bumps and high relative motion. The performance of the Kinect (the percentage of the time that it was able to maintain a lock on the user) was gathered as a function of peak angular speed. Figure 6 displays a plot of this data.

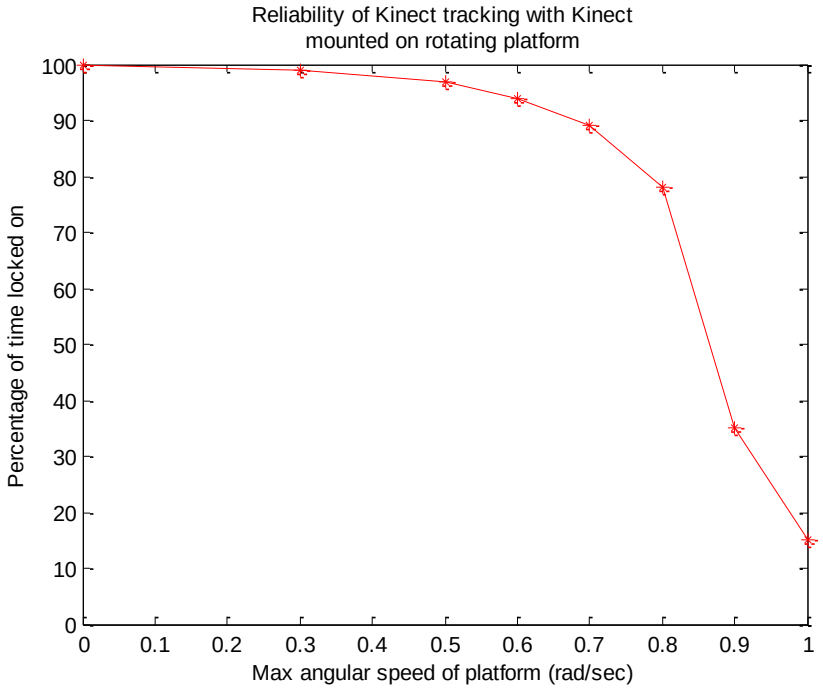

Figure 6: Tracking performance of Kinect under motion

Qualitatively, When the Kinect was still, performance was best. The Kinect could detect users rapidly moving through the scene, and it could easily deal with partial occlusion. The Kinect only lost a lock when a target moved very quickly or exited and reentered the scene. The Kinect could be confused if two users came close together, being unable to tell users apart by means other than their spatial positions.

The Kinect's performance degraded as Harlie's angular velocity increased. When the Kinect lost the target, it usually reacquired the target right away. This resulted in a flickering effect as the Kinect struggled to maintain a lock. With a peak velocity below 0.5 radians/second, the performance was comparable to the case of standing still. The incidence of flickering increased with speed, as well as the chance that the Kinect would lose a target and not quickly reestablish it. At

the maximum tested speed of 1.0 radians/second, the Kinect performed very poorly at tracking, maintaining a lock only around 15% of the time. At these high speeds, target reacquisition was slow and spotty after a dropout.

In general, the Kinect performed well from a slow-moving base. At low speeds, there was not much difference from the Kinect's stationary performance. At higher speeds, the Kinect performed more poorly. It is hypothesized that this was partially due to relative motion between the Kinect and the target, and partially due to bumps resulting from Harlie's dynamics of motion.

4 Pan Mount

To improve the Kinect's performance for a person-following robot, a rotating mount was built to allow the Kinect to pan and face its target. The Kinect had a limited field of view that proved problematic when being used from a mobile base, and the pan mount greatly expanded the effective field of view. The Kinect was most adept at tracking targets with low relative motion, and the pan mount helped by lowering relative lateral motion between the Kinect and the target.

4.1 Hardware

To maximize field of view, the pan mount was placed on top of Harlie and near the center (Figure 1.) This required removal of an aluminum mast and the relocation of some electronics that previously blocked the front of the robot. A mount with both pan and tilt capability was initially considered, although it was determined that the Kinect's vertical field of view was sufficient so tilt capability was eliminated to reduce complexity and cost. If in the future it becomes necessary to track users offset vertically from the robot, tilt capability may be desirable.

The chosen mount was a ServoCity DDP155 Base Pan (Figure 7) [33]. The DP155 is a low-cost, direct-drive pan mount that incorporates a standard hobby servo. The Hitec HS-485B, a mid-range hobby servo, was selected to power the mount. A standard PWM signal is sent to the servo to control the mount's

position. The DP155 has a ball-bearing shaft that makes the pan platform very rigid and reduces axial stresses on the servo.

Figure 7: DP155 Base Pan (left), Phidgets 1066_0 Servo Controller (right)

To drive the servo, several servo controllers were compared and the 1066_0 PhidgetAdvancedServo 1-Motor was selected [34]. The Phidgets 1066_0 enabled precise open-loop control of a hobby servo at 30 Hz, obeying programmed constraints on velocity and acceleration. For this project, a maximum velocity of 40 degrees/sec and acceleration of 90 degrees/sec^2 was chosen. The device was completely powered by a USB port and provided real-time feedback on current consumption as well as open-loop estimates of position and velocity (Figure 8). Phidgets provided a convenient API with bindings in multiple languages to communicate with the device.

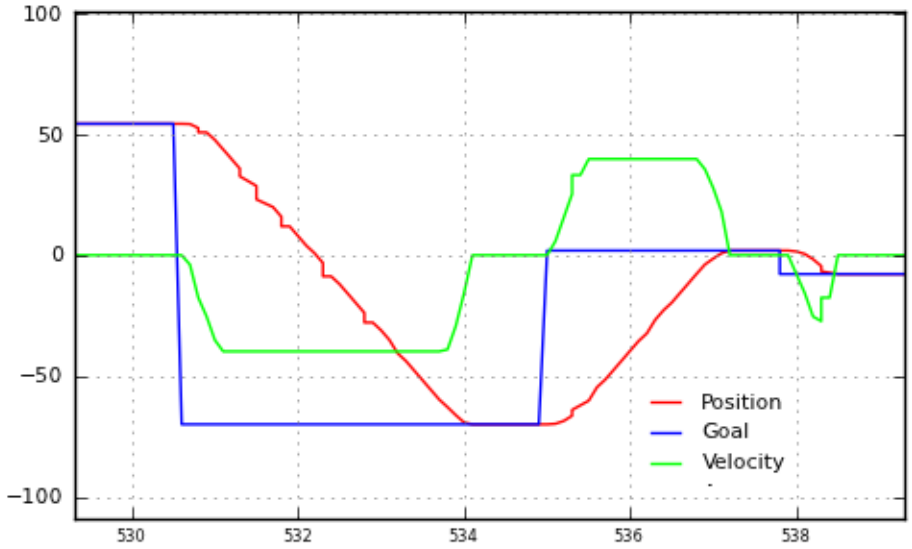

Figure 8: Output from Phidgets 1066_0, showing commanded position and open-loop feedback for position and velocity

Control software was designed to continuously monitor the last known position of the detected person and direct the pan mount to move to that angle. In turn, the control software repeatedly received open-loop feedback from the Phidgets 1066_0 and published a transform incorporating the open-loop feedback. The TF (transform) API of ROS was used to represent the time-varying transform between the Kinect and the rest of the robot. As a result, when other nodes in the system used data from the Kinect, the data could easily be transformed from the rotating reference of the pan mount to an absolute world coordinate frame.

4.2 Performance

The pan mount clearly alleviated one issue with the Kinect, the limited field of view. Without the pan motion, the Kinect had an extremely limited 57 degree field of view. The pan mount provided 180 degrees of rotation, so given

that the pan mount was allowed to track a target, the Kinect's field of view was increased from 57 degrees to an effective 237 degrees. This represents an improvement of over 300%.

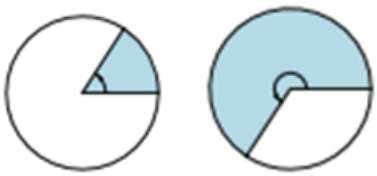

Figure 9: Kinect's effective FOV without (left) and with (right) pan mount

The ability of the pan mount to compensate for angular motion was tested. A subject stood 1.5m away from Harlie while the Kinect's RGB data was fed into a Haar cascade face detector at 2Hz, as explained in chapter 5.1 . The pan mount was ordered to rotate through 50 degrees of angle with a sinusoidal velocity profile and a peak speed of 40 degrees/second. The face detector located the subject's face in Kinect-relative coordinates which were transformed to world coordinates to account for the motion of the pan mount. If the pan mount and its associated transformations were working perfectly, the detected face would always be in the same world-relative position, no matter the position or velocity of the pan mount.

As shown in Figure 10, the pan mount performed fairly well. Most measurements were less than 5cm from the mean (standard deviation = 3.7cm). While an error of 5cm would be troublesome for tasks that require high precision such as mapping, this error does not pose a problem for person tracking. People

are large, distinct objects, and this project could easily tolerate absolute error as high as 50cm of error in positions of reported people.

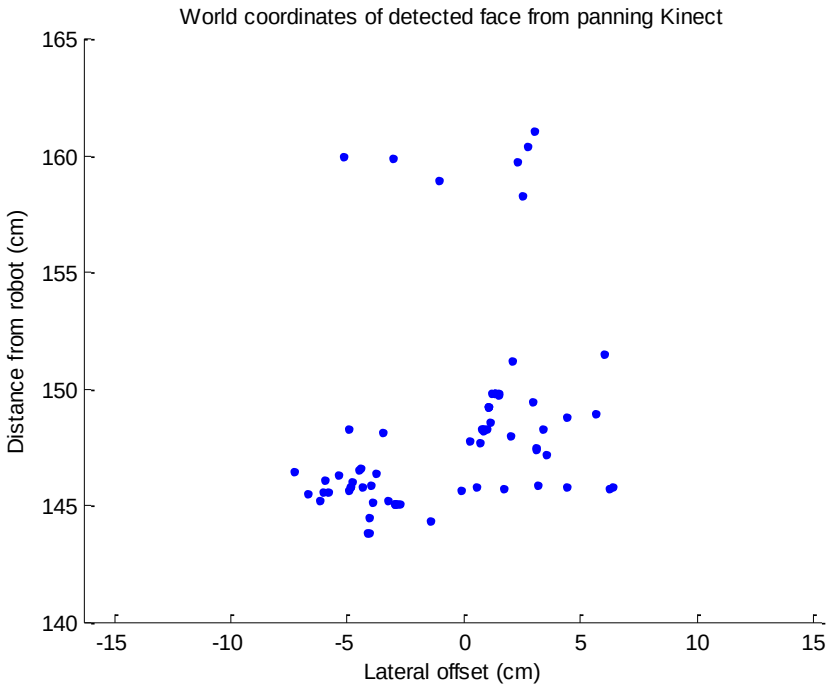

Figure 10: World coordinates of detected face while pan mount is under motion. True position is at (0,145). Discrepancy is due to errors in the pan mount's ability to accurately report its angular position and publish an accurate transform to world coordinates.

To quantify performance relative to Figure 6 (without a pan device,) the tracking performance of the Kinect was again tested, this time with motion compensation from the pan mount. Figure 11 includes the new data. Somewhat surprisingly, the pan compensation resulted in decreased performance under 0.8 radians/second compared to the baseline. Because a standard hobby servo was used in the pan mount, its motion was not entirely smooth. It is hypothesized that the pan mount introduced some jitter that made tracking more difficult at low speeds. At speeds higher than 0.8 m/s, the negative effects of servo jitter were compensated for by positive effects resulting from the reduction in relative

motion. The decrease in performance in low speeds was tolerable, made up for by the increase in performance at high speeds and the increase in effective field of view.

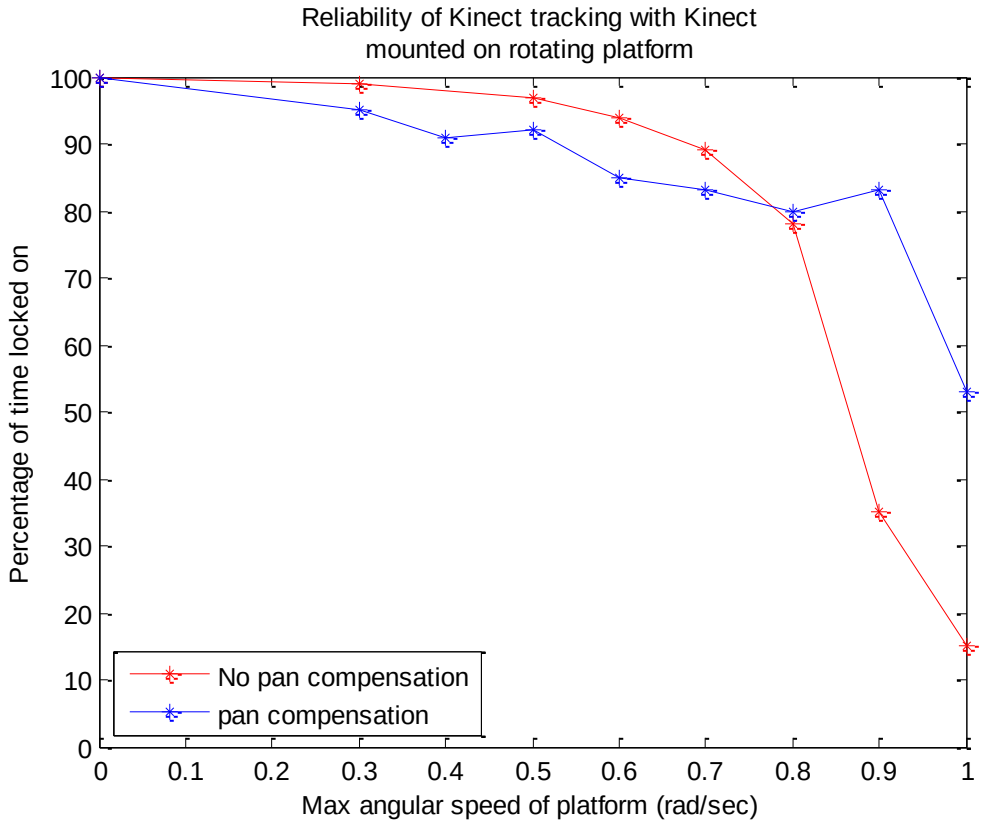

Figure 11: Tracking performance of Kinect with pan compensation

Overall, the pan mount greatly improved the tracking capabilities of the Kinect from a mobile base, by quadrupling the effective field of view and compensating for some relative motion. The greatest problem with the current pan mount is its susceptibility to bumps and vibrations. As evidenced by Figure 11, the mount introduces some vibrations that decrease the Kinect's performance. While the current benefits of the pan mount outweigh the drawbacks, this could be a subject for future work. A higher-grade mount with a geared DC motor and optical encoder could be explored to provide smoother motion. Additionally, a

vibration-isolating mount could be explored to shield the Kinect from bumps arising from Harlie's dynamics. With an improved, vibration-isolating mount, it is hypothesized that pan compensation would result in improvement from the Kinect's baseline performance at all speeds.

5 Person Tracking

As introduced in Chapter 3, the Kinect alone was not sufficient to provide reliable person tracking. Even with pan compensation, the Kinect was subject to bumps and was blind to objects at very close ranges.

To address these issues, a multi-modal approach was adopted (Figure 12) built on the ROS people stack. A main filter node maintained a Kalman filter to track the target person, continuously publishing estimates of the user's position. Measurement nodes communicated with the filter node, attempting to associate their measurements with the filter's estimate by distance and other criteria. When a measurement node successfully associated a measurement with the filter's estimate, it published an observation that the filter node used to update the Kalman filter.

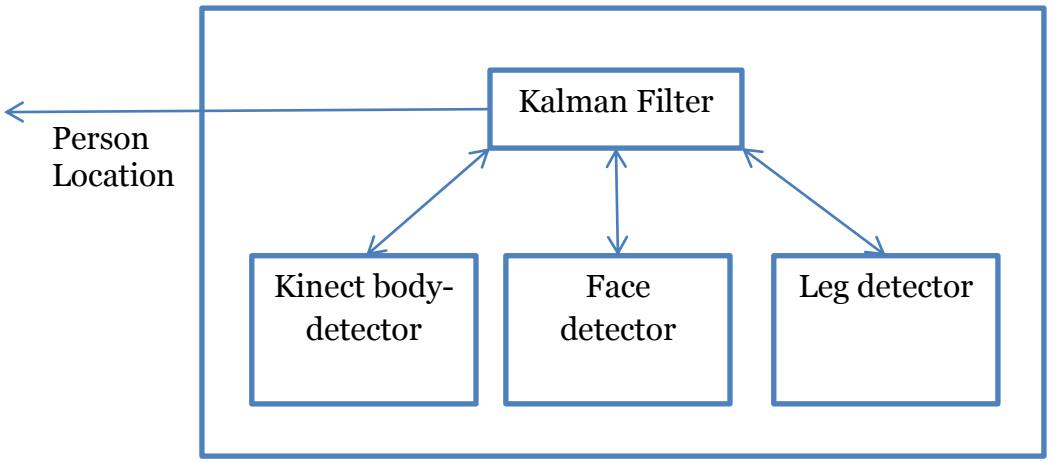

Figure 12: Person-tracking architecture

For this project, three sources of observation were used: a face detector, a leg detector, and a custom body detector based on the Kinect. The architecture made it simple to integrate multiple sources of observation from various sensors, so in the future more sources of observation could easily be added.

5.1 Face Detector Node

The face detector is one of the nodes included in the ROS People stack. The face detector ran an OpenCV cascade of Haar-like features on the Kinect's camera feed to detect faces [3]. It ran at 2Hz, processing the Kinect's full 640x480 video feed converted to monochrome. The face detector combined its matches with depth data from the Kinect, pruning faces based on plausible sizes in three dimensions. This pruning provided resistance against spurious readings from face-like objects.

With a list of plausible faces and their positions in 3D space, the face detector tried to associate these matches with the tracker from the filter node. If a face was close enough to the tracker to make an association, the face detector published a position measurement back to the filter node.

The face detector could reliably detect faces up to 8m away at sizes as small as 20x20 pixels. The face detector did not rely on persistence between frames, so it could reliably detect users when Harlie was moving rapidly. Although the face detector was capable, it was inherently restricted to cases in which the user was staring directly at the robot. It failed to detect faces at angles.

Furthermore, the face detector did not perform recognition. It detected human faces, but could not tell one face from another.

5.2 Leg Detector Node

The leg detector was another node distributed with the ROS People stack. It detected legs using a boosted cascade of features computed from a LIDAR scan [21] [22]. The leg detector performed best at close ranges where a large number of laser returns were recorded per leg. Performance dropped off with distance because at long distances, few laser returns were recorded per leg. Calculating accurate features from small clusters of points is a difficult task. Additionally, the leg detector was able to perform detection but not recognition, because there is not enough information in a 2D scan to uniquely identify a person.

The leg detector proved most useful at close ranges, compensating for some of the shortcomings of the Kinect. At close ranges, the Kinect performed poorly because of its limited field of view and the minimum range of the Kinect's depth sensing. However, when the user was near to Harlie, each leg had a large number of laser returns and tracking via leg detection was very accurate. The SICK scanner had a 180-degree field of view, so the user could be tracked over a wide field of view at close range.

5.3 Kinect Person Detector Node

The final source of observation for the Kalman filter was a person detector node which used the Kinect to track users within its field of view. A reliability layer was added on top of the Kinect's built-in skeleton tracking to store

persistent information about the user, providing a sort of fingerprint to increase accuracy in identifying the tracked user. A normalized, 2D hue-saturation histogram was chosen as the persistent information, similar to [6]. When identifying a person, color information is an obvious first choice because of its salience and the ease of obtaining and processing. The hue-saturation histogram was chosen to represent color information while protecting against changes in lighting intensity. In the future, rather than treating the person as a whole, perhaps the Kinect could be used to segment each individual limb, and a separate color histogram could be computed for each body part.

When the system first started up, the Kinect was calibrated as described in section 3.1 . At the moment that the calibration was complete, a color snapshot of the user was taken (Figure 13). The hue-saturation histogram was then constructed (Figure 14). For this example, one can clearly see three major patches of color: reds for the shirt, blues for the jeans, and beiges for skin tones.

Figure 13: Kinect's RGB image masked for a user, taken after calibration

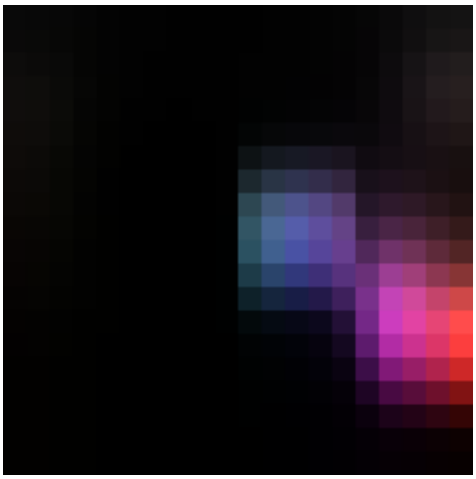

Figure 14: Histogram computed from Figure 13: hue on horizontal axis, saturation on vertical axis, brightness represents to histogram value.

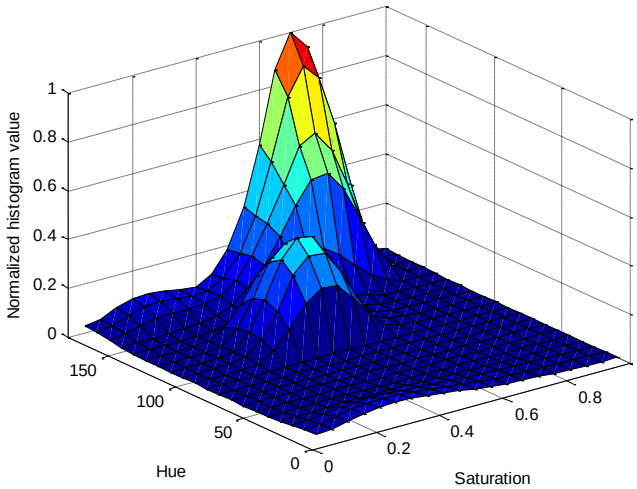

Figure 15: Alternate view of Figure 14 as 3D surface plot

Software was designed to maintain an idea of the user's current histogram, using it to eliminate non-tracked users. In the program's main loop, the body detector received a list of users from the Kinect along with masked images of their respective pixels as in Figure 13. The program computed a histogram for each detected user and tried to make an association with the tracked user. Correlation was chosen as a metric for comparing histograms. For two histograms H_1 and H_2, the correlation equals 1.0 when the histograms are identical, and is near zero for two random histograms.

$$corr(H_1, H_2) = \frac{\sum_i (H_1(i) - \overline{H_1})(H_2(i) - \overline{H_2})}{\sqrt{\sum_i (H_1(i) - \overline{H_1})^2 \ \sum_i (H_2(i) - \overline{H_2})^2}}$$

Still, the user's histogram may change over time due to variations in lighting color and posture. The user's histogram will also change if the user picks up an object or new article of clothing. Therefore, a method was included to account for changes in the user's appearance over time.

The hue-saturation histogram can be represented by a matrix **H**. Let the user's histogram at calibration be $\mathbf{H_{cal}}$, the current idea of the user's histogram $\mathbf{H_{track}}$, and the latest associated measurement of the user $\mathbf{H}_{meas}$. Over time, given new measurements of $\mathbf{H_{meas}}$, $\mathbf{H_{track}}$ is allowed to drift away from $\mathbf{H_{cal}}$. This is accomplished through a low-pass filter:

$$\mathbf{H_{track,new}} = \text{normalize}\left[\ \alpha\ \mathbf{H_{track,old}} + (1 - \alpha)\mathbf{H_{meas}}\ \right]$$

Where $\mathbf{H_{track}}$ is slowly pulled in the direction of $\mathbf{H_{meas}}$. With this method, it is possible that $\mathbf{H_{track}}$ will drift too far away and the user will be lost. Suppose

the user slowly picks up a large object. The program will receive many incremental measurements of $\mathbf{H_{meas}}$, and $\mathbf{H_{track}}$ will have a chance to adjust to the new appearance of the user. If the user suddenly drops the object, $\mathbf{H_{meas}}$ will quickly change and $\mathbf{H_{track}}$ will no longer be valid. To account for such cases, if $\mathbf{H_{meas}}$ is not successfully associated with $\mathbf{H_{track}}$, then $\mathbf{H_{meas}}$ is compared to the original calibration $\mathbf{H_{cal}}$. If $\mathbf{H_{meas}}$ is associated with $\mathbf{H_{cal}}$, then $\mathbf{H_{track}}$ is shifted back toward $\mathbf{H_{cal}}$ with a second low-pass filter and the association with $\mathbf{H_{meas}}$ is attempted again.

$$\mathbf{H_{track,new}} = \text{normalize}\left[\, \beta\, \mathbf{H_{track,old}} + (1-\beta)\mathbf{H_{cal}} \,\right]$$

Figure 16 illustrates the ability of the histogram to adapt to the changing appearance of a tracked user. With the Kinect in a fixed position, the user walked around a room for fifteen seconds. The correlations of the user to both $\mathbf{H_{track}}$ and $\mathbf{H_{cal}}$ were recorded and plotted. The user's correlation to $\mathbf{H_{cal}}$ was inconsistent, dropping below 0.7 at times. This was due to variation in room lighting and the different body silhouettes that the user exposed to the camera over time. However, the user's correlation to $\mathbf{H_{track}}$ remained above 0.9 for the entire duration of the test. Thus, it is concluded that the low-pass filter was helpful in adapting to the changing appearance of the user.

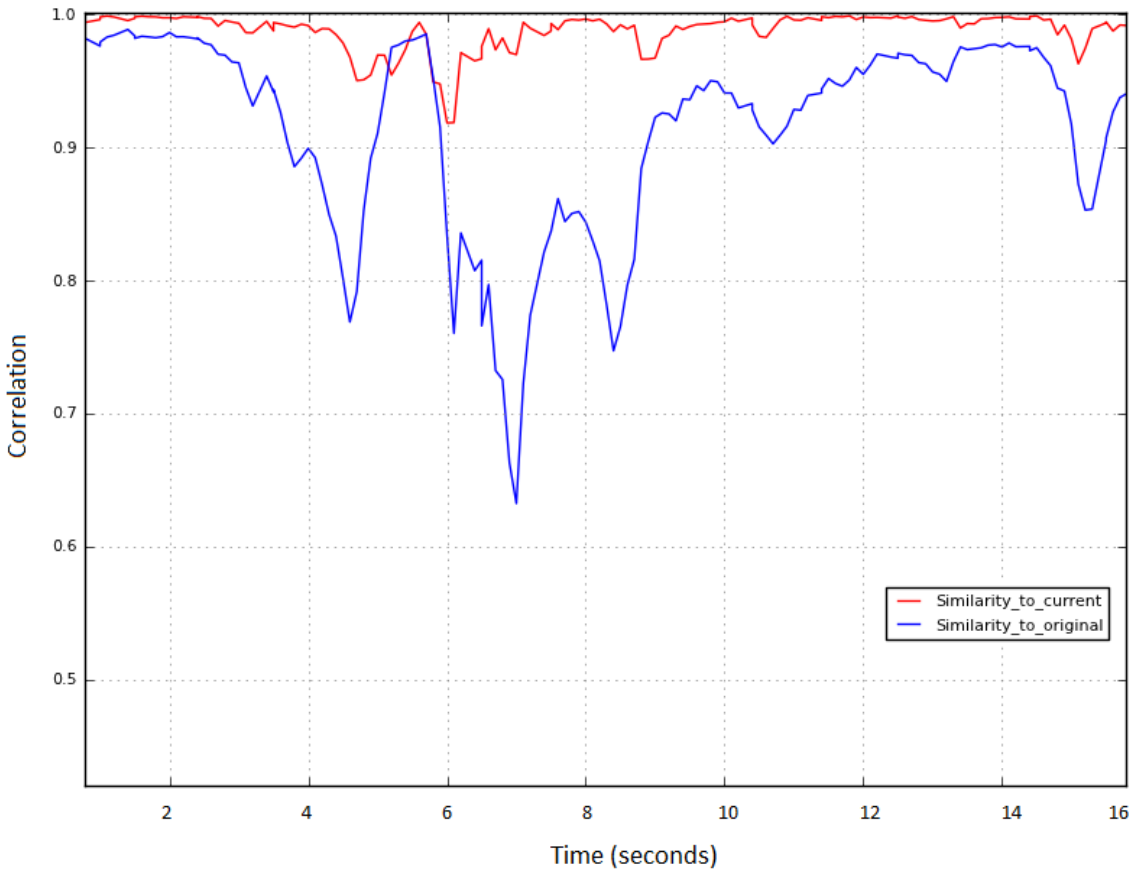

Figure 16: Correlation over time of user's hue-saturation histogram with $\mathbf{H}_{\mathbf{track}}$ (red) and $\mathbf{H}_{\mathbf{cal}}$ (blue)

6 Planning

A major component of this project involved the creation of a system to perform dynamic path replanning. Previous attempts were made to perform person tracking at CWRU, although these attempts were unsuccessful due to the limitations of traditional planning methods. Traditional point-to-point planning performs well for cases in which only static navigation is required, such as the case of a tour-guide robot moving through a fixed series of poses. However, point-to-point planning is not well-suited for following dynamic targets.

When tracking a person, a traditional point-to-point planner would need to create a new plan every time that the person moved. Typically a robot would need to halt every time that a new plan was made, resulting in an unacceptable amount of stuttering. To resolve these issues, this project combined a point-to-point planner with an intelligent rolling-window algorithm. Figure 17 provides an architectural overview of the chosen method.

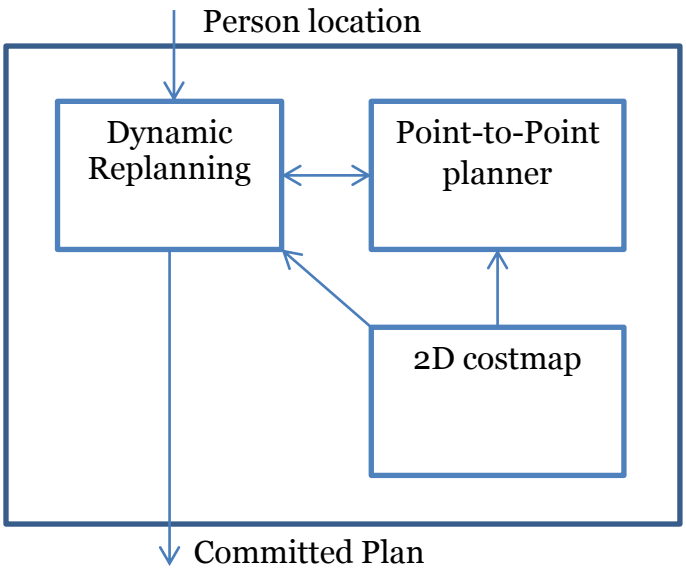

Figure 17: Planning module architecture

6.1 Point-to-point planner

This project's dynamic replanning algorithm employed a point-to-point planner from the ROS SBPL (search-based planning lattice) package. The SBPL software was developed by Maxim Likhachev at the University of Pennsylvania in collaboration with Willow Garage [27] [35].

The SBPL planner implements a search-based, ARA* planning algorithm that supports operation in three-dimensional (x, y, θ) space. The third, angular dimension was essential for representing Harlie's full range of motions because Harlie was not holonomic. The motions available to Harlie were dependent on orientation: due to the mechanical constraints of the wheelchair base, it was impossible for Harlie to move sideways.

The SBPL planner discretized the x-y plane with 2.5cm square resolution and discretized angles with resolution of $\pi/8$ radians. When constructing paths,

the planner selected elements from a pre-defined library of motion primitives. The motion primitives are short path segments that can be combined to make a complete path. Motion primitives can be chosen to correspond to kinematically-plausible motions of the robot, and each motion primitive can be assigned a cost. A typical scenario is to lower the cost of wide arcs and straight lines while penalizing sharp turns and backwards motion. With a wise selection of motion primitives and associated costs, the SBPL planner produces smooth, kinematically-feasible paths (Figure 18.)

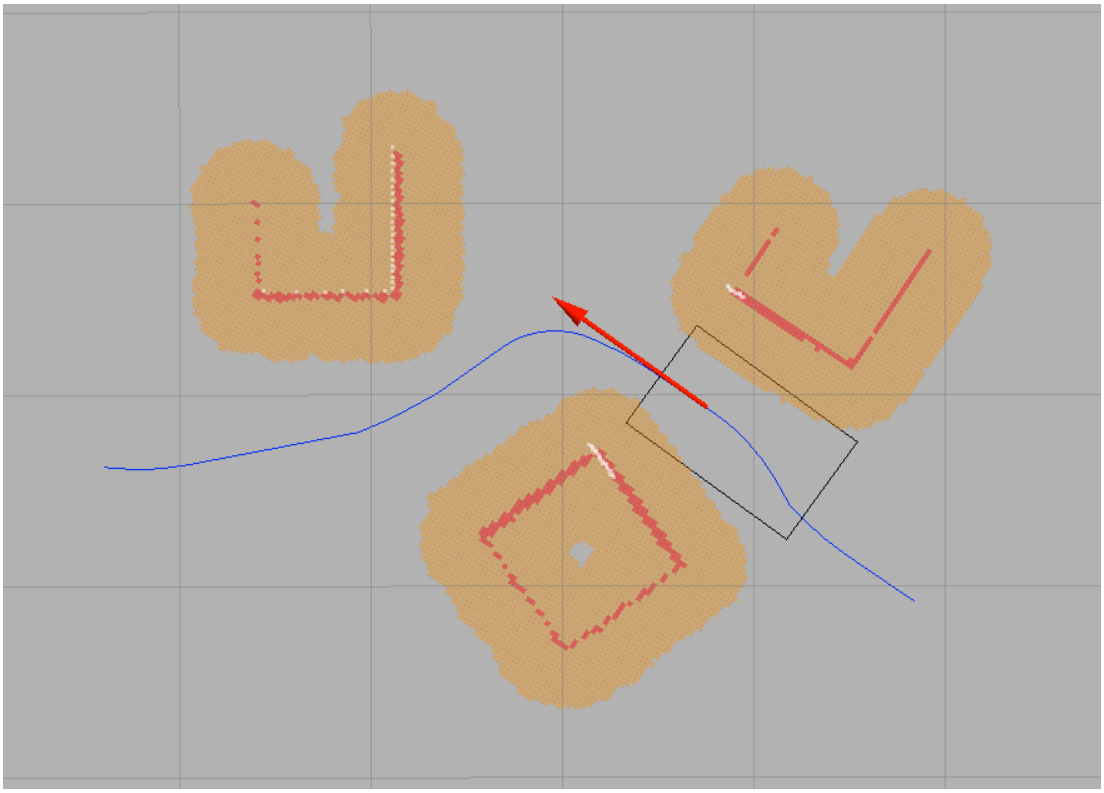

Figure 18: Smooth path produced by SBPL planner in presence of obstacles (grid size 1m)

Previous work at Case Western Reserve University involved navigation using path segments (lines, arcs, and spin-in-place segments), which were a natural fit for the SBPL planner's motion primitives. A set of motion primitives were created for Harlie including forward and reverse line moves, spin-in-place

moves, and arc moves of two different radii. The arc radii (approximately 1m and 2m) were constrained by the grid discretization of 2.5cm along with the angle discretization of $\pi/16$. Figure 19 shows the motion primitives customized for Harlie.

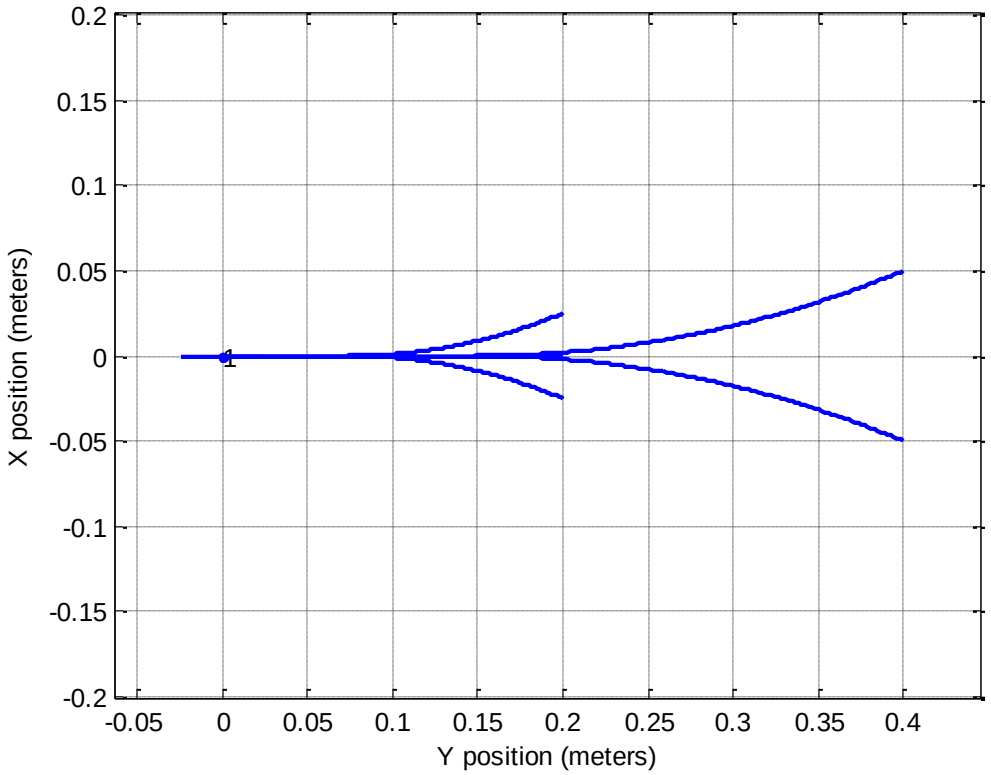

Figure 19: Harlie's motion primitives (spin-in-place moves not shown)

The SBPL package integrated with a ROS costmap [36]. The costmap provided facilities for constructing a dynamic obstacle map on top of the static map used for localization. Observations from Harlie's LIDAR unit were fed into the costmap and used to mark new obstacles and clear free space. The costmap provided a flexible framework designed to integrate with observations from multiple sensors. Eventually, it is planned to use the Kinect to detect drivable surfaces and low obstacles such as curbs. The LIDAR sensor is capable, although

it is restricted to detecting obstacles that fall inside of a single 2-dimensional plane.

At every pose along the path, the SBPL planner checked Harlie's boundary for collision against the costmap with a resolution of 2.5cm. This ensured the safety of Harlie and nearby pedestrians. The resolution of 2.5cm was sufficient to enable Harlie to maneuver through tight spaces such as doorways and to perform complicated maneuvers including multi-point turns.

The SBPL planner was fast in normal operation: a typical runtime for planning several meters in a clear setting was 0.2 seconds. The runtime increased for difficult moves, especially those requiring backward motion or squeezes through tight spaces. Even in the worst cases, the runtime rarely exceeded 1.5 seconds. Thus, the SBPL planner had the speed necessary for dynamic replanning.

Multiple modifications to the base SBPL planner were performed for this project. A custom motion primitive file was created for Harlie as illustrated in Figure 19. The output of the SBPL planner was converted from a series of points to the CWRU path segment standard. Finally, discretization error relating to the planner's 2.5cm grid was corrected in order to ensure continuity when multiple paths were spliced together.

6.2 Dynamic Planning

A major portion of this project involved the creation of a dynamic replanning algorithm to allow Harlie to track a moving target without coming to a halt. At the heart of the algorithm is a rolling window which divides the robot's path into two sections, referred to as the committed path and the uncommitted path. Figure 20 provides a high-level illustration of the algorithm.

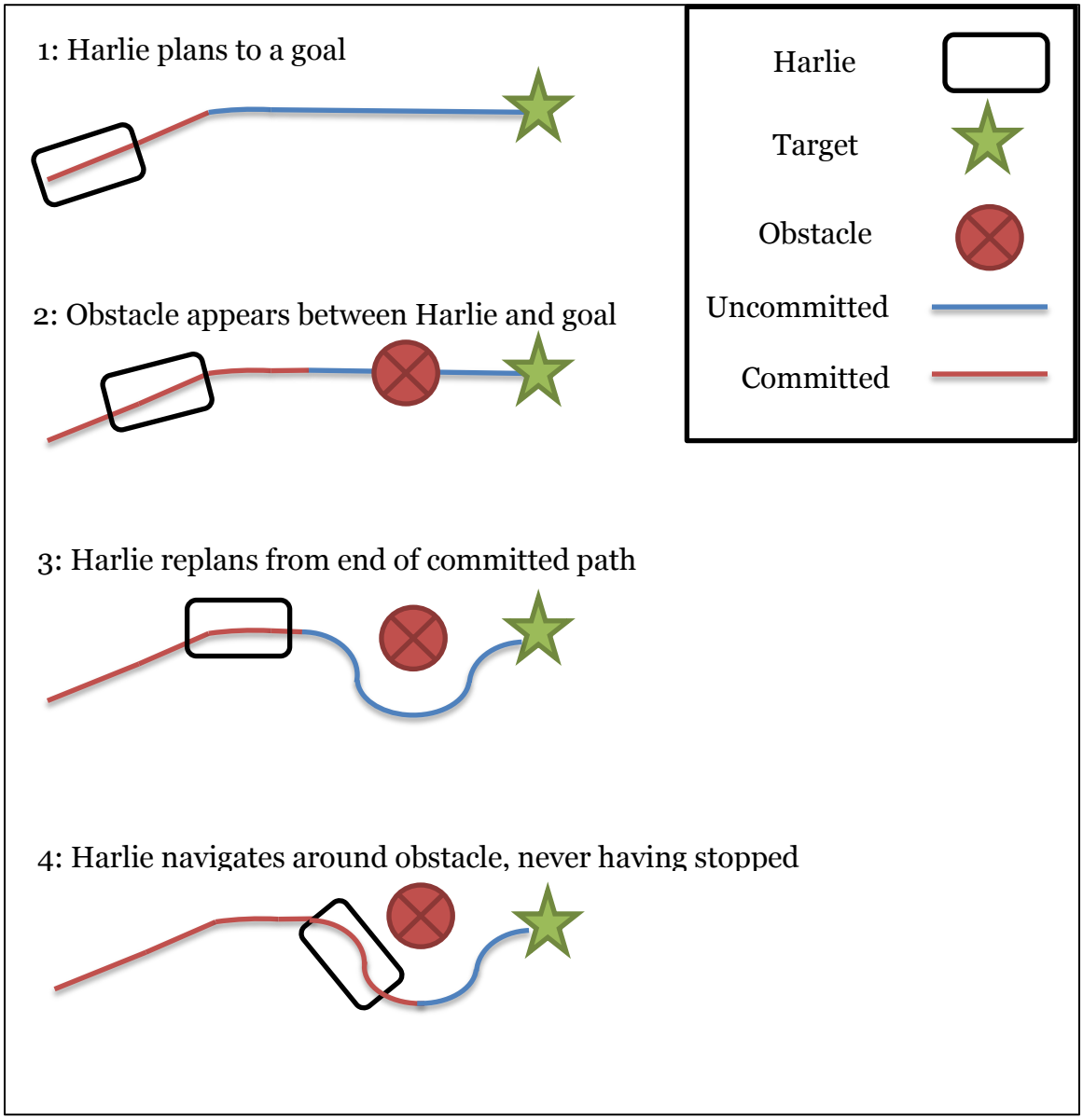

Figure 20: Illustration of rolling-window approach

The committed path represents a short-term plan that Harlie will execute in the next several seconds. The committed path is nominally 1m long, just enough to keep the robot moving for 1-2 seconds. This section of path is referred to as "committed" because it is shared with the steering module. The planner is committing to the steering module that this section of path will not be changed except under extenuating circumstances.

The uncommitted path represents the robot's long-term plan to get to the goal, subject to change if the target moves or obstacles appear. The uncommitted path is held internally by the planner, and it is not shared with the steering module. This section of path is uncommitted, meaning that it can be changed without penalty as long as its starting pose is constrained to match the end of the committed path.

The planner continuously monitors the committed path and tries to maintain a length of approximately 1 meter. If the length of the committed path drops below a threshold, path segments are shifted from uncommitted to committed. If the steering module reaches the end of the committed path (the robot has reached the goal or is taking a long time planning) the robot simply comes to a halt until more uncommitted segments are available to commit.

Setting the desired length of the committed path involves a tradeoff. If the committed path is too long, Harlie will lose flexibility in planning to the target by committing too soon to a path that may be unsuitable in the future. If the committed path is too short, Harlie will likely run out of committed path before it is able to replan to the moving goal, causing the robot to come to an early halt. A

length of 0.8-1.0m for the committed path has been found to produce good results in simulation and practice.

When planning, Harlie has two actions available: a partial replan and a full replan, as illustrated in Figure 21. When performing a partial replan, Harlie discards the uncommitted path and creates a new uncommitted path that is constrained to start at the end of the committed path. The committed path is not modified and Harlie is able to remain in motion. When performing a full replan, the entire path is discarded, Harlie is brought to a halt, and a new plan is made from the halt state. The steering module is reset. A partial replan is always preferred to a full replan as to not force Harlie to stop. Table 1 summarizes the conditions leading to full and partial replans, and these conditions are elaborated upon in the following paragraphs.

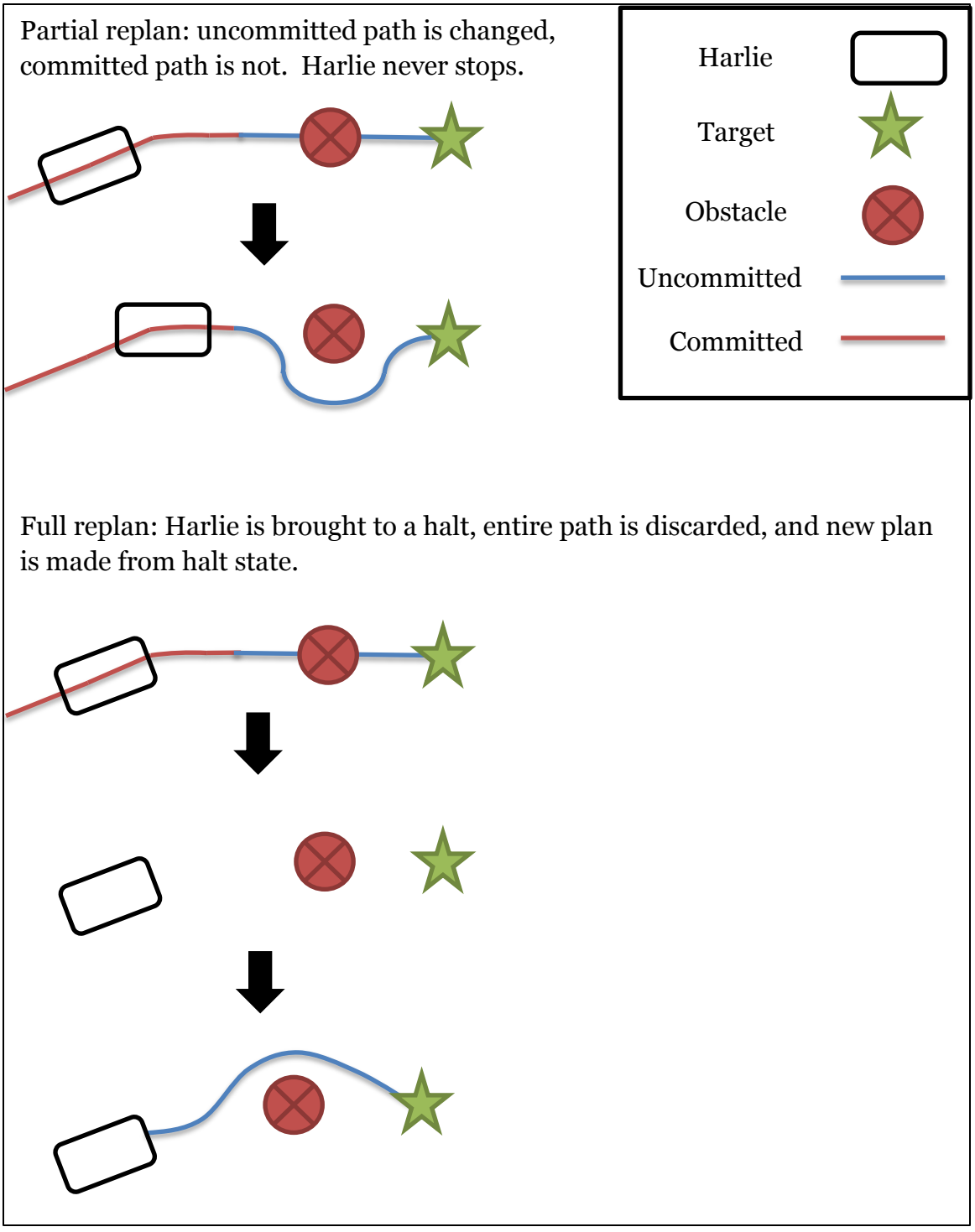

Figure 21: Illustration of partial and full replanning

When the planner receives a new goal from the person-tracking module, it attempts to perform a partial replan. This is part of normal operation and is the mechanism that allows Harlie to remain in motion when following a moving target. As illustrated in Figure 21, a partial replan is also triggered if a potential collision is detected along the uncommitted path. In this way, Harlie can stay in motion even when dynamic obstacles such as other pedestrians are present. If a partial replan fails, a full replan is performed. Also, a full replan is performed by default when the robot is at rest and no committed path exists.

Conditions for partial replan
• New goal • Obstacle in uncommitted path
Conditions for full replan
• Partial replan fails • Robot currently at rest • Obstacle in committed path • Target moves behind robot

Table 1: Conditions for full and partial replanning

If a potential collision is detected along the committed path, the committed path is no longer safe and Harlie is brought to a halt. A full replan is performed from the halt state. Because the committed path is short, this mechanism serves as an emergency reflex to prevent Harlie from hitting pedestrians that stray too close.

Finally, a full replan is performed if it is deemed less expensive to bring Harlie to a halt and create a new plan than to follow the previously committed path. Currently, this is only done if the target moves behind the robot, as illustrated in Figure 22.

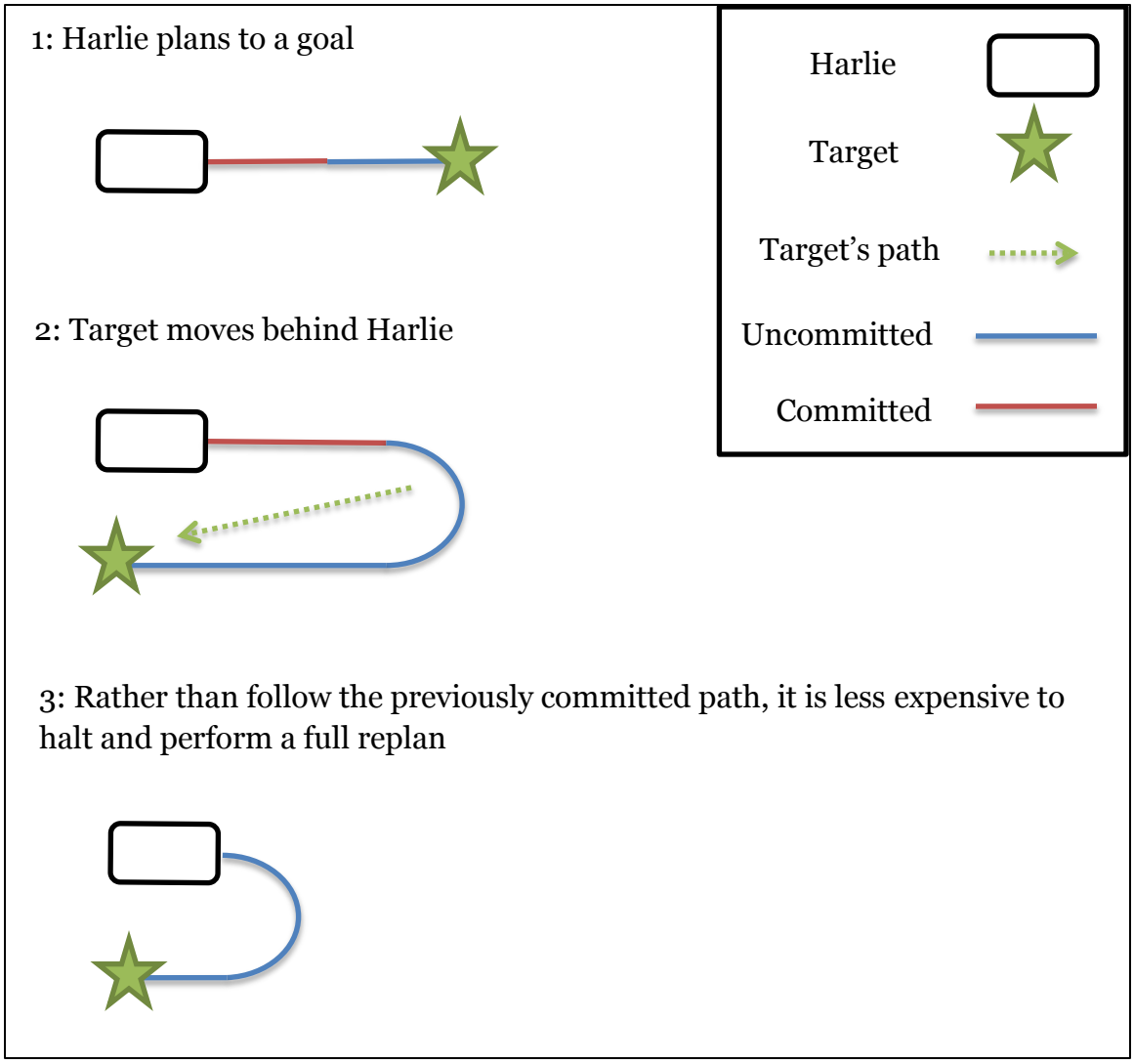

Figure 22: Special condition leading to full replan: target moves behind the robot

6.3 Goal Generation

For the purpose of person tracking, special consideration had to be given to goal generation. Goals were passed from the person-tracking module to the planner as discussed in chapter 5.1 . Without modification, these goals were unsuitable for planning. It would be impolite for the robot to plan directly to the target, because that space was occupied by the person being tracked. Thus, it was necessary to generate goal candidates offset by some distance from the true goal

This project's solution was to generate a "constellation" of goal candidates offset by varying angles and distances. The positions were chosen based on experience and simulation, to give Harlie flexibility in planning to the target (Figure 23). Upon generating the goal constellation, each goal was checked for collision with the robot's footprint against the 2D obstacle map with a resolution of 2.5cm. Goals in collision were removed. To keep planning time reasonable, only the first several cleared goals were passed to planning.

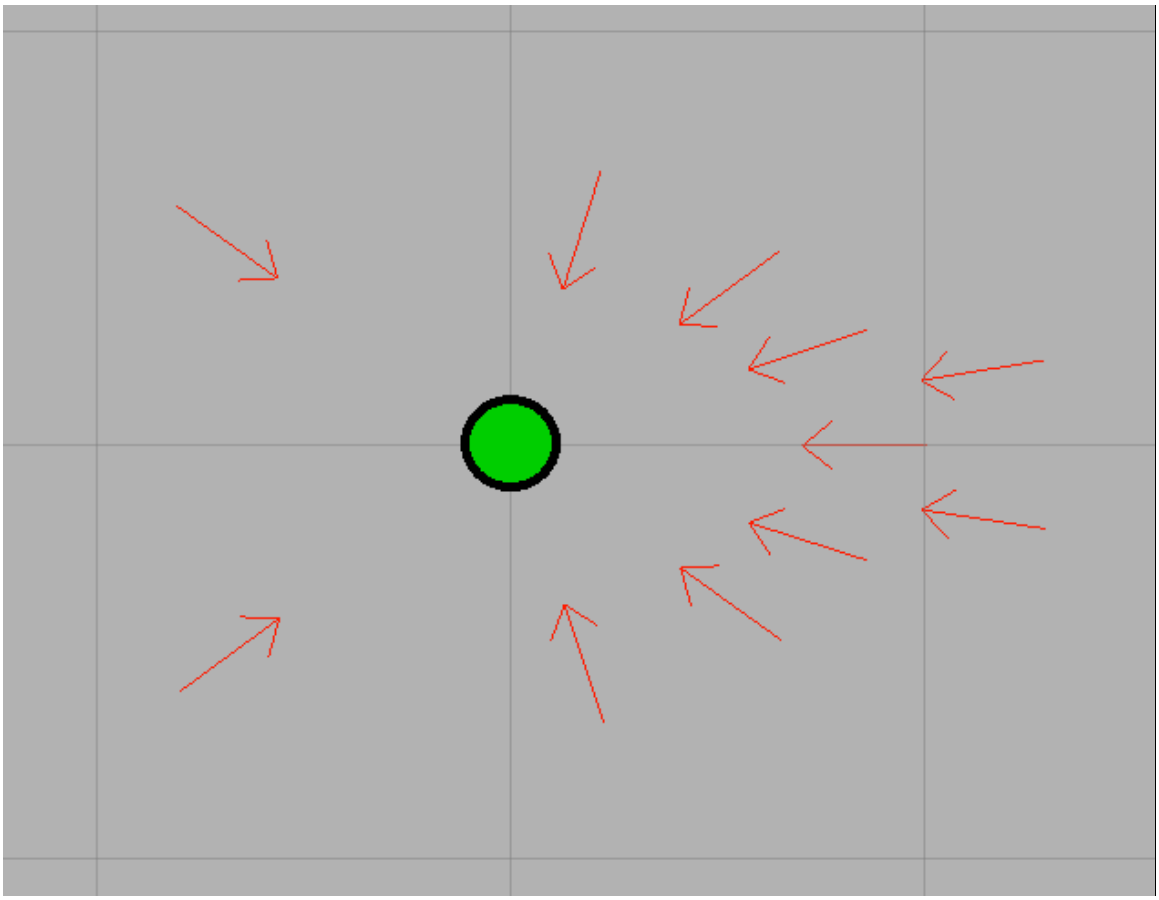

Figure 23: Goal constellation, true goal in green (grid resolution 1m)

6.4 Benchmarks

Several benchmarks were performed to verify that the dynamic planning algorithm was appropriate for person-following. First, tests were done in simulation in order to decouple the effects of the person-tracking module. The ROS Stage software package was used to simulate Harlie's kinematics and to generate obstacles. The simulation was run on the Dell laptop.

First, planning was benchmarked in a clear setting with no obstructions. A series of twenty goals were generated in a line in front of Harlie. The goals were spaced 0.5m apart and fed to Harlie sequentially. The next goal was triggered when the distance to Harlie dropped below 3m. Because Harlie was continuously in motion, the receipt of a new goal triggered a partial replan. The time taken for each partial replan was recorded and a histogram of the times was created (Figure 24.) As shown, planning in an obstruction-free setting was a computationally-simple task with a median runtime of 0.1 seconds.

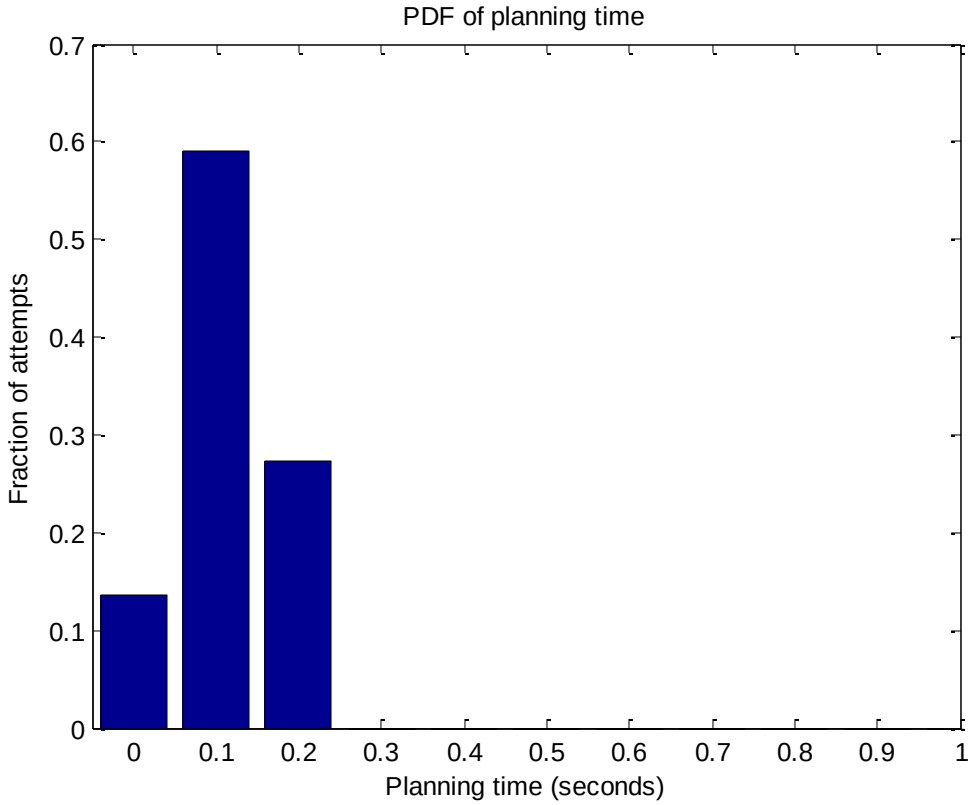

Figure 24: Planning benchmark in obstruction-free setting

Next, Harlie's ability to dynamically replan around an obstacle was tested. In a scenario similar to that illustrated in Figure 20, Harlie was given a static goal 6m to its front. Harlie's original plan represented the path of least resistance, a straight line from the start to the goal. After Harlie created its original plan and was in motion, an obstacle (a 0.8 m^3 box) was dropped 3m in front of the robot. Harlie was forced to perform a partial replan around the obstacle, producing a revised path such as the one in Figure 25. Twenty trials were performed, resulting in a median time of 0.4 seconds (mean 0.5 seconds) to perform a partial replan around the box. The distribution of times is shown in Figure 26. Most importantly, in no cases did the replan take so long that the committed path ran out, forcing Harlie to come to a stop.

As an aside, two partial replans were usually required to navigate around the box (only the first was benchmarked.) As evidenced in Figure 25, at first Harlie could only view the box's front face. When the LIDAR unit cleared the side of the box, more of the obstacle came into view, invalidating the uncommitted path and necessitating a second replan. The second replan was less complex than the first and took less time. In all trials performed, neither of the two replans caused Harlie to come to a halt.

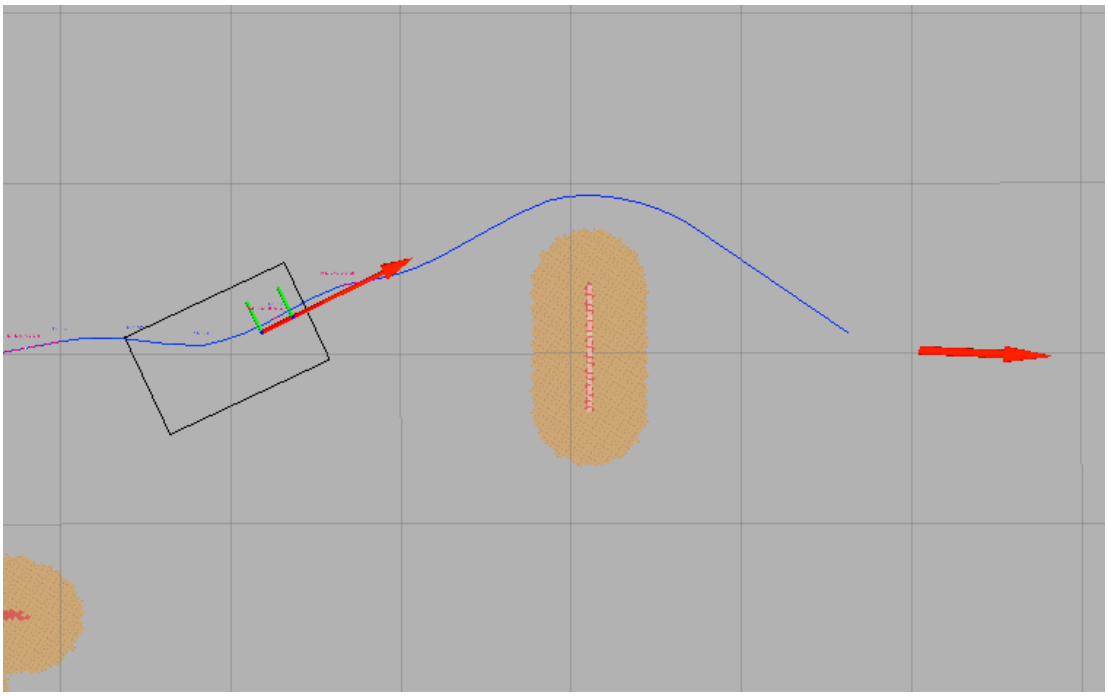

Figure 25: Path taken by Harlie to avoid box (grid size 1m.) Note that only front face of box is visible.

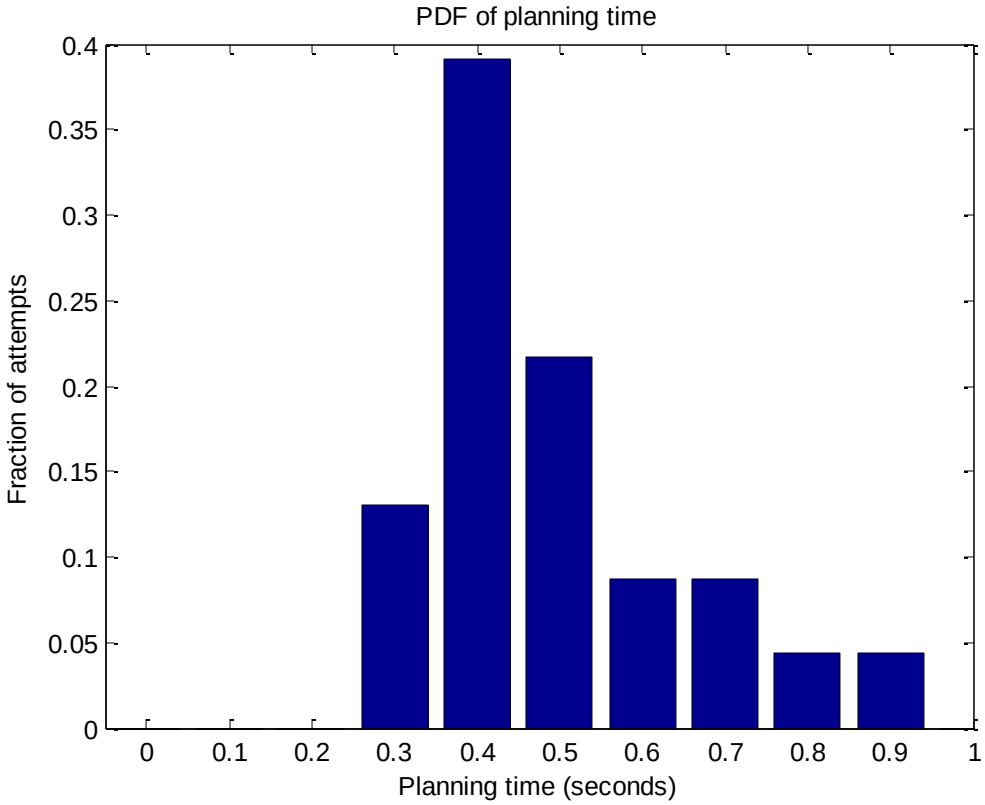

Figure 26: Planning benchmark in dynamic replanning scenario

After benchmarking the planning module in simulation, trials were carried out to test the entire system, including the person-tracking module. These final trials were performed in a lab setting with several chairs and traffic barrels as obstacles. The target person took a circuitous path and was followed by Harlie. Figure 27 illustrates the path of the target person in green and the path of Harlie in blue. Dots represent position measurements. Harlie's position was recorded every 30cm and the person's position was recorded at those same instants. Every fifth measurement was highlighted and numbered for convenience. As shown, Harlie was able to fluidly track the target person while avoiding the obstacles.

Notable is the fact that Harlie did not precisely follow the target person's trajectory. The target person took a roundabout path, nominally staying 2-3

meters in front of Harlie. If Harlie were to follow this trajectory exactly, it would be highly inefficient. Due to the action of Harlie's dynamic replanning, Harlie was not confused by the target person's motion, but took a tight trajectory to follow the leader.

Figure 28 illustrates another trial demonstrating Harlie's person-tracking capabilities. Once again, Harlie took a short, efficient path when following the target person. Also notable, at the point marked as "5," the person went through a space too narrow for Harlie to follow. Harlie did not get confused; the planning algorithm allowed the robot to go around the obstacle to meet up with the target person on the other side.

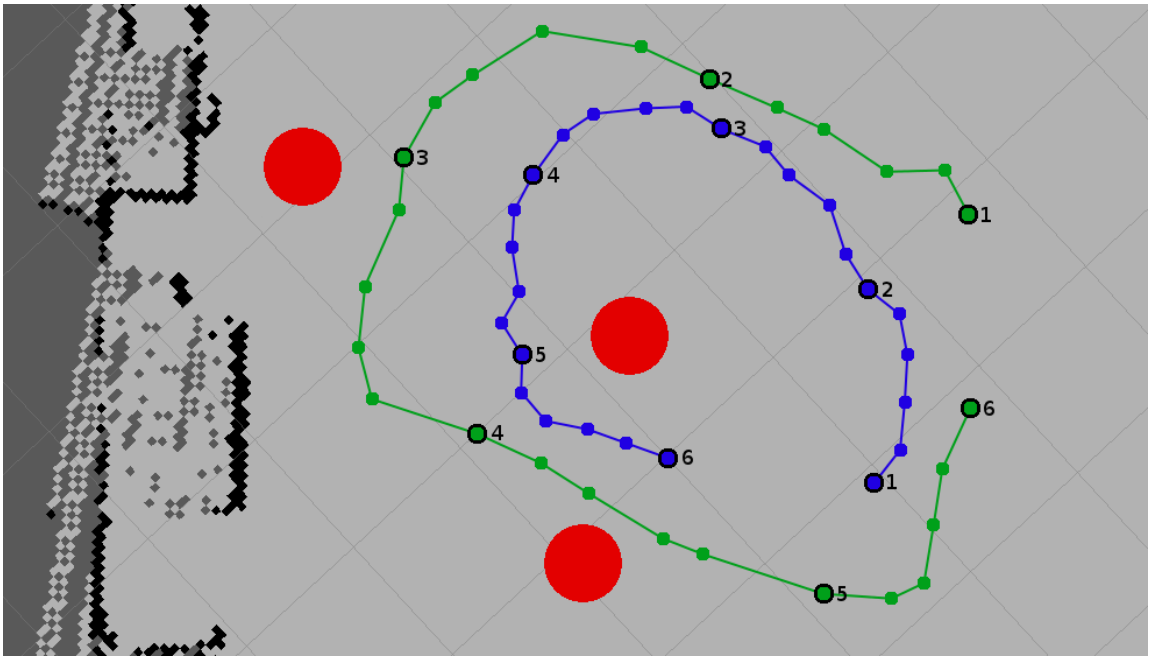

Figure 27: Path taken by Harlie (blue) and target person (green). Dots represent position measurements, every 5th measurement highlighted. Grid size 1m. Red and black indicate obstacles.

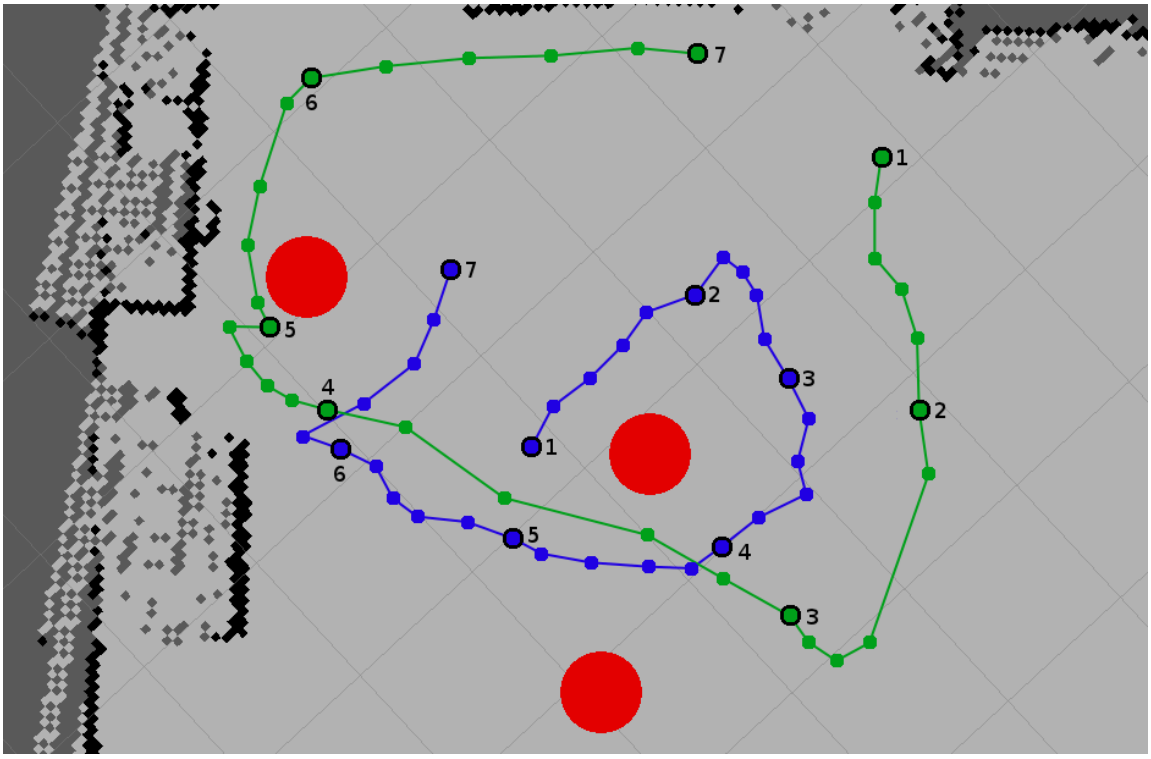

Figure 28: Additional trial, same parameters as Figure 28

7 Conclusions

A person-following system was successfully developed for the mobile robot Harlie. The system used a Kalman filter to integrate the Microsoft Kinect's person-tracking capabilities with a face detector and a LIDAR-based leg detector. The sensor-fusion approach allowed the system to benefit from each detector's strengths while avoiding their individual weaknesses. A dynamic replanning algorithm was developed to allow Harlie to follow a moving target. The system worked well in tests with fairly complex environments, and it was able to smoothly follow a user through a cluttered room.

7.1 Summary of Accomplishments

The performance of the Microsoft Kinect and the OpenNI person-tracking software were tested under conditions that would be encountered on a mobile robot. The Kinect was designed for use from a stationary vantage point, and difficulties were encountered in tracking targets when the Kinect itself was in motion. Additional difficulties arose from the Kinect's limited field of view and inability to distinguish between users.

To address some of the Kinect's limitations, a panning mount was developed to rotate the Kinect and point it at the target. Software was written to assist in transforming data from the Kinect's to world coordinates, accounting for the motion of the pan mount. The pan mount resulted in a 300% increase in the Kinect's effective field of view. Performance data demonstrated that the pan

mount also resulted in improvements in the Kinect's ability to track moving targets.

Person-tracking software was built using elements of the ROS People stack. From the People stack, a Kalman filter node, a face detector, and a LIDAR-based leg detector node were used with modification. A body-detector node was developed using the Kinect and the OpenNI library. Addressing a major limitation of the Kinect, the inability to distinguish between users, a reliability layer was added on top of OpenNI. A fingerprint in the form of a 2D hue-saturation histogram was maintained for the tracked user and used to distinguish the desired user from other targets.

Path-planning software required the majority of this project's programming work. The ROS SBPL (search-based planning lattice) planner was used with modification as a point-to-point planner. Modifications were made for compatibility with the CWRU path segment standard, and the SBPL planner's parameters and motion primitives were customized for Harlie. The SBPL planner was then integrated into a dynamic replanning framework.

A dynamic replanning algorithm was created to provide the flexibility needed to follow a moving target. The algorithm uses a rolling-window approach to allow Harlie to follow a moving target and avoid obstacles without coming to a halt. Tests and benchmarks show that the dynamic replanning algorithm provides the flexibility and computational speed necessary for person-following.

Finally, the complete system was integrated on Harlie. The complete system has been shown to perform well, and it meets the goal of smoothly tracking a person from a mobile robot.

7.2 Future Work

The Kinect with OpenNI and NiTE has been shown to perform well enough from a mobile base that it can be used as part of a person-following system. The largest drawback of the current setup is the Kinect's susceptibility to bumps and vibrations. In the future, this could be resolved with a more robust pan platform to give smoother motion and a vibration-damping mount to isolate the Kinect from the worst bumps.

The current system relies on a LIDAR sensor for obstacle detection. The LIDAR sensor is highly capable, although it is inherently restricted to detecting obstacles that fall within a single 2-dimensionsal plane. Eventually, it is planned to use the Kinect's depth data to detect drivable surfaces and low obstacles such as curbs.

The developed path planning algorithm performs well over medium to long distances, although Harlie shows some weakness when following users at close range (less than 1 meter.) Path planning performs quickly enough that the robot can smoothly follow a moving target, although there is a nontrivial lag between when Harlie gets a new goal and Harlie updates its path. At short distances, the committed path is short or nonexistent so Harlie does not have a

buffer to protect against this lag. As a result, Harlie performs sluggishly when following a user at close range.

Work has been done to improve planning over short ranges, including the special condition for planning illustrated in Figure 22. In the future, perhaps a more responsive, but less intelligent, planning algorithm could be used at short range to increase fluidness. Harlie could switch to search-based planning when the target moves far enough away.

Works Cited

[1] C. Rockey, *Low-Cost Sensor Package for Smart Wheelchair Obstacle Avoidance,* Case Western Reserve University, EECS dept., 2012.

[2] T. Germa, F. Lerasle, N. Ouadah, V. Cadenat and M. Devy, "Vision and RFID-based Person Tracking in Crowds from a Mobile Robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis, 2009.

[3] P. Viola and M. J. Jones, "Robust Real-Time Face Detection," *International Journal of Computer Vision,* vol. 57, no. 2, pp. 137-154, 2004.

[4] A. Shashua, Y. Gdalyahu and G. Hayun, "Pedestrian detection for driving assistance systems: single-frame classification and system level performance," in *Intelligent Vehicles Symposium*, 2004.

[5] D. Calisi, L. Iocchi and R. Leone, "Person Following through Appearance Models and Stereo Vision using a Mobile Robot," in *Proceedings of VISAPP-2007 Workshop on Robot Vision*, 2007.

[6] M. Kobilarov, G. Sukhatme, J. Hyams and P. Batavia, "People tracking and following with mobile robot using an omnidirectional camera and a laser," in *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, Orlando, FL, 2006.

[7] M. Bansal, S.-H. Jung, B. Matei, J. Eledath and H. Sawhney, "A Real-time Pedestrian Detection System based on Structure and Appearance Classification," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, Anchorage, AK, 2010.

[8] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001.

[9] Z. Zivkovic and B. Krose, "Part based people detection using 2D range data and images," in *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, 2007.

[10] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting," *Journal of Computer and*

System Sciences, vol. 55, p. 119.139, 1997.

[11] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision,* vol. 60, no. 2, pp. 91-110, 2004.

[12] H. Bay, A. Ess, T. Tuytelaars and L. Van Gool, "SURF: Speeded Up Robust Features," *Computer Vision and Image Understanding (CVIU),* vol. 110, no. 3, pp. 346-359, 2008.

[13] E. Seemann, B. Leibe, K. Mikolajczyk and B. Schiele, "An Evaluation of Local Shape-Based Features for Pedestrian Detection," in *Proceedings of the British Machine Vision Conference*, 2005.

[14] M. Bajracharya, B. Moghaddam, A. Howard, S. Brennan and L. H. Matthies, "Results from a Real-time Stereo-based Pedestrian Detection System on a Moving Vehicle," in *Proceedings of the IEEE ICRA 2009 Workshop on People Detection and Tracking*, Kobe, Japan, 2009.

[15] J. Satake and J. Miura, "Robust Stereo-Based Person Detection and Tracking for a Person Following Robot," in *Proceedings of the IEEE ICRA 2009 Workshop on People Detection and Tracking*, Kobe, Japan, 2009.

[16] T. Nakada, S. Kagami and H. Mizoguchi, "Pedestrian detection using 3D optical flow sequences for a mobile robot," in *IEEE Sensors*, 2008.

[17] B. Jung and G. S. Sukhatme, "Detecting moving objects using a single camera on a mobile robot in an outdoor environment," in *International Conference on Intelligent Autonomous Systems*, Amsterdam, The Netherlands, 2004.

[18] T. Gill, J. Keller, D. Anderson and R. Luke, "A system for change detection and human recognition in voxel space using the Microsoft Kinect sensor," in *Applied Imagery Pattern Recognition Workshop (AIPR), 2011 IEEE*, Washington, DC, 2011.

[19] V. Gulshan, V. Lempitsky and A. Zisserman, "Humanising GrabCut: Learning to segment humans using the Kinect," in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, Barcelona, Spain, 2011.

[20] M. A. Livingston, J. Sebastian, Z. Ai and J. W. Decker, "Performance measurements for the Microsoft Kinect skeleton," in *Virtual Reality (VR), 2012 IEEE,* Costa Mesa, CA, 2012.

[21] K. Arras, O. Mozos and W. Burgard, "Using Boosted Features for the Detection of People in 2D Range Data," in *IEEE International Conference on Robotics and Automation*, Roma, Italy, 2007.

[22] K. Arras, S. Grzonka, M. Luber and W. Burgard, "Efficient people tracking in laser range data using a multi-hypothesis leg-tracker with adaptive occlusion probabilities," in *IEEE International Conference on Robotics and Automation*, Pasadena, CA, 2008.

[23] D. Schulz, W. Burgard, D. Fox and A. Cremers, "People tracking with a mobile robot using sample-based joint probabilistic data association filters," *International Journal of Robotics Research,* vol. 22, no. 2, pp. 99-116, 2003.

[24] E. Masehian and D. Sedighizadeh, "Classic and Heuristic Approaches in Robot Motion - A Chronological Review," in *World Academy of Science, Engineering and Technology*, 2007.

[25] T. Lozano-Perez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Commun. ACM,* vol. 22, no. 10, pp. 560-570, 1979.

[26] P. Raja and S. Pugazhenthi, "Optimal path planning of mobile robots: A review," *International Journal of Physical Sciences,* vol. 7, no. 9, pp. 1314-1320, 2012.

[27] M. Likhachev and D. Ferguson, "Planning Long Dynamically-Feasible Maneuvers forAutonomous Vehicles," *The International Journal of Robotics Research,* vol. 28, no. 8, pp. 933-945, 2009.

[28] Willow Garage, "ROS," [Online]. Available: http://www.willowgarage.com/pages/software/ros-platform. [Accessed 4 May 2012].

[29] B. P. Gerkey, "AMCL Package Summary," 3 August 2011. [Online]. Available: http://www.ros.org/wiki/amcl. [Accessed 9 May 2012].

[30] E. M. Perko, "Precision Navigation for Indoor Mobile Robots," 2012.

[31] OpenNI, "About OpenNI," DotNetNuke Corporation, 2011. [Online]. Available: http://www.openni.org/About.aspx. [Accessed 9 May 2012].

[32] PrimeSense, Ltd., "PrimeSense Natural Interraction," 2011. [Online].

Available: http://www.primesense.com/nite. [Accessed 9 May 2012].

[33] Robotzone, LLC., "DDP155 Base Pan," 2012. [Online]. Available: http://www.servocity.com/html/ddp155_base_pan.html. [Accessed 9 May 2012].

[34] Phidgets, Inc., "1066_0 PhidgetAdvancedServo 1-Motor," 2011. [Online]. Available: http://www.phidgets.com/products.php?product_id=1066_0. [Accessed 9 May 2012].

[35] M. Likhachev, "Search-based Planning with Motion Primitives," 2009. [Online]. Available: http://www.cs.cmu.edu/~maxim/files/tutorials/robschooltutorial_oct10.pdf. [Accessed 30 4 2012].

[36] E. Marder-Eppstein, "Costmap_2d Package Summary," 11 August 2011. [Online]. Available: http://www.ros.org/wiki/costmap_2d. [Accessed 9 May 2012].