Cover page

School of Graduate Studies page

Table of Contents

List of Tables

List of Figures

Acknowledgements

stuff

Title page

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 [site Chad's thesis]. A pack robot could follow a soldier into combat, or a tourguide robot could intelligently lead a group of people through a museum.

Comment [W1]: Lead is different from follow

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 real-world scene contains a great deal of clutterring shapes, textures, and colors. 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

Of all the various sensors available to mobile robots, cameras are most often used for person tracking. High-resolution color cameras are inexpensive and vision-based tracking is intuitive to us as humans. Computer vision algorithms are very diverse, and this section will touch on the most popular approaches.

Some vision systems require the user to face the camera to provide a consistent view or to allow face detection. Face detection is sometimes used to re-initialize a tracker 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 applications such as a tourguide, 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 us as humans. Some of the simplest tracking approaches simply look for solid regions of a certain color. Calisi et. al. used a single-color segmentation 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 not common in the environment. More complicated approaches may track areas of multiple colors, or compute a color histogram for an area of interest. Skin colors are also frequently tracked.

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 [6]. Some authors use part-based representations that combine multiple classifiers for different parts of the body [8]. Such systems combine many weak classifiers to create a strong classifier, using a training algorithm such as AdaBoost.

Comment [W2]: Cite AdaBoost

Another vision-based approach is to detect keypoints. A keypoint represents a distinctive, salient, geometric feature. A number of popular algorithms including SIFT [9], SURF, and HOG [7] detect scale invariant keypoints, providing consistency 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 can also be related to a higher-order part-based model. Seemann et. al. evaluated various keypoint detectors trained on images to identify pedestrians in a scene [10].

Comment [W3]: Spell out acronyms before using.

Comment [W4]: Cite

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. Such depth information is greatly helpful in segmenting targets from the background [7]. Omnidirectional cameras are sometimes used, as in Kobilarov et. al. with the advantage of being aware of targets all around the robot, although omnidirectional cameras often have issues with distortion and limited resolution [1].

Comment [W5]: Cite?

Bajracharya et. al. 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 ~~which~~ that corresponded to

Comment [W6]: Cite?

upright objects in 3D space. They used 3D geometric features and color information to classify the resultant blobs as people [11]. Although they work 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 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 [12].

Comment [W7]: Cite earlier—as soon as you mention names, so reference is clear

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, although this method breaks down if the robot moves quickly or if the robot's motion is not bump-free.

The Microsoft Kinect is a more recent innovation that combines a traditional RGB camera with an IR depth camera. The Kinect provides the capabilities of a high-end stereo vision system [13] at a fraction of the cost. Numerous authors have used the Kinect's depth sensing capabilities as a replacement for a stereo camera, for static surveillance [13] [14] or as a depth sensor from a mobile robot.

Comment [W8]: Which cite goes with which?

The Kinect also comes with built-in ~~skeleton~~ 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 abilities from a static viewpoint [15]. Few attempts have been made in the literature to evaluate the Kinect's ~~person~~ person-tracking capabilities from a mobile robot.

Comment [W9]: Any? If so, cite,; if not, say so

1.2 Other sensors

Because the performance of vision systems may depend 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, 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 the sensors most often 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, efforts have been made to reduce price, such as reverse engineering the LIDAR used on the Neato XV-11 vacuum cleaner [xxxchad].

Comment [W10]: Can you back up this claim w/ cites?

For the purpose of tracking people, LIDAR units may be mounted at hip height, creating a single blob per person, or more often below knee height, 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 [18]. Using an adaptive algorithm such as AdaBoost, such a classifier can automatically

Comment [W11]: Can you cite instan ces?

be created from scan data [19]. LIDAR units have been used both for person detection from a mobile robot and for static surveillance from a fixed point.

Laser rangefinders have a very wide field of view, although they have a limited resolution on the order of one raytrace per degree. LIDAR units perform best when the person is up close and the unit can record many laser returns per leg. 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, and usually augment vision-based methods.

Comment [W12]: Your own data? If not, cite.

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 [8] [12]. When the system receives measurements, the measurements are associated with the filter's latest position estimate by distance or other criteria [20]. Successfully associated measurements are used to update the filter's state. In the case of tracking multiple people, joint probabilistic data association filters (JPDAFs) have seen success, providing a probabilistic framework to associate measurements with multiple targets [20]. They are useful in the case of tracking multiple people, or planning around the movement of pedestrians.

Even when the system only uses one source of information, sensor fusion algorithms are widely used 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 is useful when multiple targets are in the scene with similar appearances. Filters can also reduce a system's computational load. Instead of running detectors 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

Many person-following algorithms do not use planning at all, but use simple controllers based on minimizing bearing between the robot and the target, and maintaining a following distance [1].

Comment [W13]: ?

The simplest planning algorithms involve gradient descent such as the wavefront algorithm, and force-based techniques where obstacles exert simulated repulsive forces [21]. Rolling-window approaches examine potential trajectories for obstacles. These simple planners are unable to perform complex, multi-stage moves such as three-point-turns, or backing up to get out of tight corners. They especially fail for non-holonomic robots such as Harlie.

Comment [W14]: Did you introduce "Harlie"?

Work has been done in supplementing these local planning methods with global planning~~,~~.

Comment [W15]: Do you mean you? Or are you referring to more cites?

Many recent attempts have focused on search-based planning.

2 Harlie

Harlie is a mobile robot built on an electric wheelchair base. Harlie is equipped with a server [SPECS] and a SICK LIDAR unit used for obstacle detection and localization.

Comment [W16]: Could use cites here for Harlie (theses); include a photo of Harlie

All software was developed for Ubuntu Linux using the ROS (Robot Operating System) framework provided by Willow Garage. Processing was split between two computers. The main computer was Harlie's server, running software related to planning, steering, and localization. Kinect and person-tracking software was run on a laptop connected to Harlie via Ethernet. The laptop was a Dell Latitude E6510 with a 2.67GHz Intel Core i5 CPU and 4GB of RAM.

3 Evaluation of the Microsoft Kinect

The Microsoft Kinect, released in the year 2010, is a human interface device originally developed for the 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, providing the Xbox with their locations in 3D space as well as the position and orientation of their limbs.

The Kinect has a 640x480 RGB camera as well as a 640x480 IR camera. An infrared projector shines a known pattern through a diffraction grating, and by computing disparity between the known pattern and what is observed from the IR camera, a depth value can be computed for any given pixel. This gives the Kinect great potential as a 3D sensing system. Its retail price of $150 prices it 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 [13].

PrimeSense, the makers of the Kinect's software, has released an open-source API called OpenNI (Open Natural Interaction) to allow developers to tap into the Kinect's functionality. In addition to accessing the depth and RGB camera feeds, Primesense provides high-level functionality for tracking user skeletons through a library called NiTE. With OpenNI and NiTE, the Kinect is able to seamlessly detect and track multiple human users in its field of view. This is obviously very appealing for the application of person tracking.

Comment [W17]: Can you cite these?

Livingston et. al. evaluated the skeleton tracking capabilities of the Kinect from a static viewpoint [15]. They found that the Kinect can track users from 4m to just under 1m distance, although the performance at these extremes was erratic. Microsoft recommends an optimal range of 1.2-3.5 meters. They also found that sensor noise increased as a function of distance.

Few attempts have been made in the literature to evaluate 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 a real-world 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.

Comment [W18]: Any?

Most of 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. Higher-level software workarounds had to be employed to make up for some shortcomings of NiTE, mostly due to its closed-source nature.

3.1 Discrimination Between Users

A major issue with the Kinect is the lack of built-in facilities for discriminating between different users. While in theory 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.

The Kinect relies on continuity between frames to maintain a lock on a target, which is perfectly fine for its intended application as a game controller where players never leave the field of view and the Kinect is stationary so the target lock is rarely broken. However, for applications with a moving base, frequent dropouts must be dealt with. My solution as explained 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.

3.2 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.

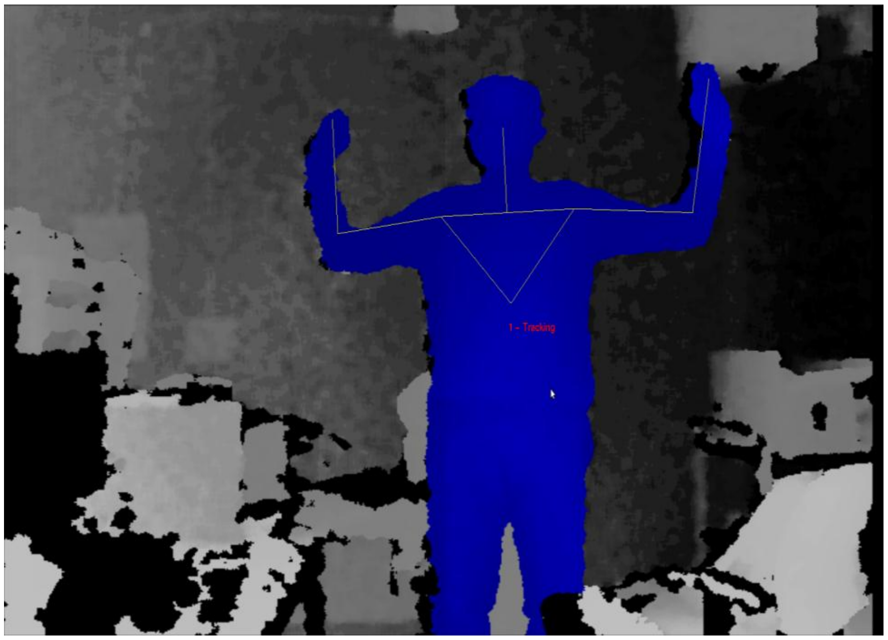


Figure 1: Kinect's distinctive “psi” calibration pose

With the Kinect is on a moving base, occasionally the target will be lost due to relative motion or 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 and to apply that saved calibration 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 measurements.

This raises a larger issue: the Kinect has no built-in facilities to discriminate between users. The tracking software seems to rely on spatial continuity between frames, and it stores no information that could uniquely identify a user (colors, textures, etc.) 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. This is perfectly fine for the intended application as a game controller where players never leave the field of view and the Kinect is stationary so the target lock is rarely broken. However, for applications with a moving base, frequent dropouts must be dealt with. My solution as explained 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 ~~of course~~ 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 a target loss. When the robot once again faces the user, it will have to re-acquire the user, leading to a delay.

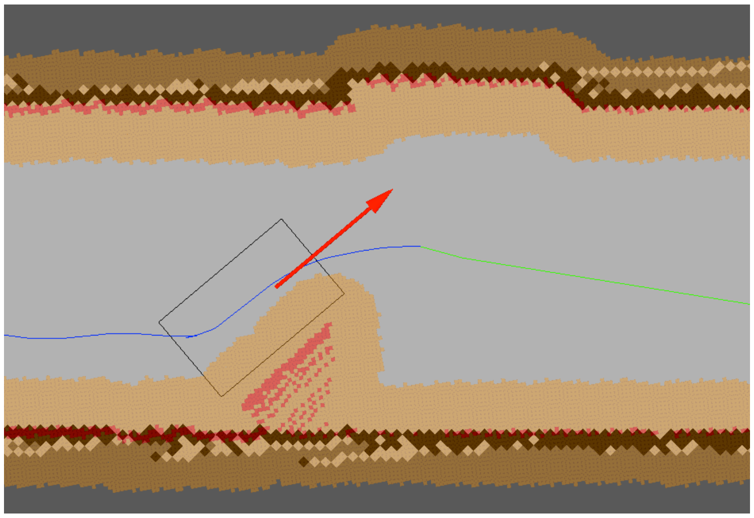


Figure 2: Obstacle avoidance may lead to target loss due to Kinect's limited field of view

The situation becomes even worse if the user doubles back behind the robot. In tight spaces such as hallways, the user will must come close to Harlie when moving behind it. The Kinect's depth camera breaks down when targets are closer than 2 feet away. Thus, Harlie's Kinect has a blind spot for close

Comment [W19]: ?

objects. In a hallway scenario, this can result in Harlie being stuck pointing at close range to a wall within the blind-spot range.

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 (xxxINSERT MECHANICAL DRAWING OF KINECT'S FOV). 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.

These issues introduced by skipping calibration are resolved ~~chapter~~ 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.

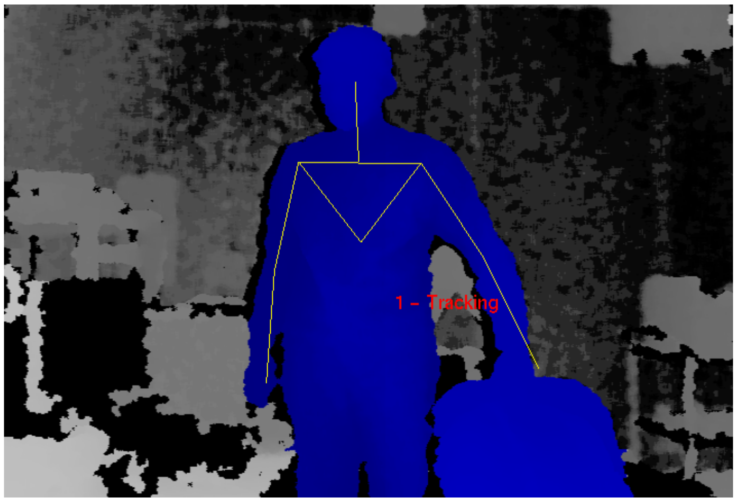


Figure 3: Difficulties arise in tracking a user in contact with a chair

3.4 Moving Base Problem

The Kinect was designed to be placed in front of a television to track users playing a game. Mounting the Kinect on Harlie's moving base poses challenges outside of the Kinect's design parameters. A walking pace for an average human is around 1 m/s. 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 is difficult to exactly characterize the system's performance. However, some metric of performance is 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 maintain a lock on the user 100% of the time. In reality, the Kinect periodically drops the user due to bumps and 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.

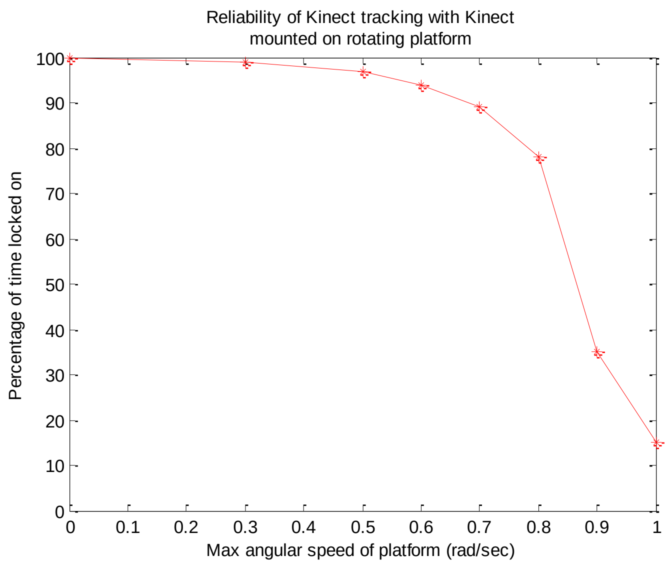


Figure 4: Tracking performance of Kinect under motion

Qualitatively, When the Kinect was still, performance was ~~obviously~~ best. The Kinect can detect users rapidly moving through the scene, and it can easily deal with partial occlusion. The Kinect only loses a lock when a target moves very quickly or exits and reenters the scene. The Kinect can be confused if two users come close together, being unable to tell users apart by means other than their spatial positions.

The Kinect's performance degrades as Harlie's angular velocity increases. When the Kinect loses the target, it usually reacquires the target right away, resulting in a flickering effect as the Kinect tries to maintain a lock. With a peak velocity below 0.5 radians/second, the performance is comparable to the case of standing still. The incidence of flickering increases with speed, as well as the chance that the Kinect will lose a target and not quickly reestablish it. At the maximum tested speed of 1.0 radians/second, the Kinect performs very poorly at tracking, maintaining a lock only around 15% of the time. At these high speeds, target reacquisition is slow and spotty after a dropout.

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

3.5 Summary

Comment [W20]: This is not really a summary

The major issues with the Kinect xxx

As will be discussed in ~~chapter~~ Chapter 5, this xxx issue was resolved by treating the bodies detected with OpenNI as one input to an overall Kalman filter. ~~as discussed in chapter~~

4 Pan Mount

To alleviate some issues inherent with the Kinect, a rotating mount was built to allow the Kinect to pan and face its target. The Kinect has a limited field of view that is problematic when it is being used from a mobile base, and the pan mount greatly expands the effective field of view. The Kinect is most adept at tracking targets with low relative motion, so the pan mount helps by lowering side-side relative motion between the Kinect and the target.

To maximize field of view, the pan mount was placed on top of Harlie and near the center. [xxx INSERT DIAGRAM]. This required removal of an aluminum mast that previously blocked the front of the robot and the relocation of some electronics. 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 cut down on complexity and cost.

The chosen mount is a ServoCity DDP155 Base Pan (Figure 5). The DP155 is a low-cost, direct-drive pan mount that incorporates a standard hobby servo. The DP155 has a ball-bearing shaft that makes the pan platform very rigid and reduces axial stresses on the servo. The Hitec HS-485B, a mid-range hobby servo, was selected to power the mount.

Comment [W21]: Cite

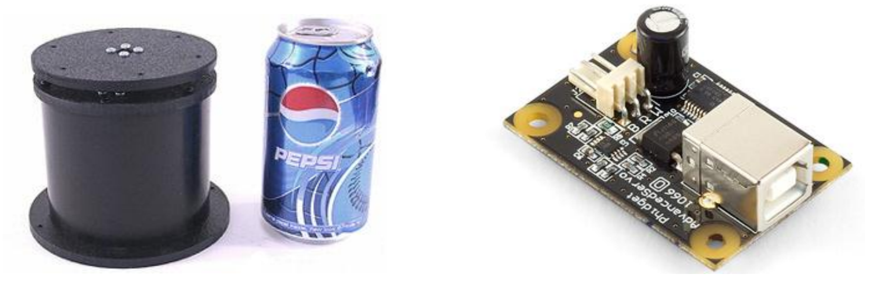


Figure 5: DP155 Base Pan (left), Phidgets 1066_0 Servo Controller (right)

To drive the servo, several servo controllers were compared and the 1066_0 PhidgetAdvancedServo 1-Motor was selected. The Phidgets 1066_0 enables 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 is completely powered by a USB port and provides real-time feedback on current consumption as well as open-loop estimates of position and velocity (Figure 6). Phidgets provides a convenient API with bindings in multiple languages to communicate with the device.

Comment [W22]: Cite

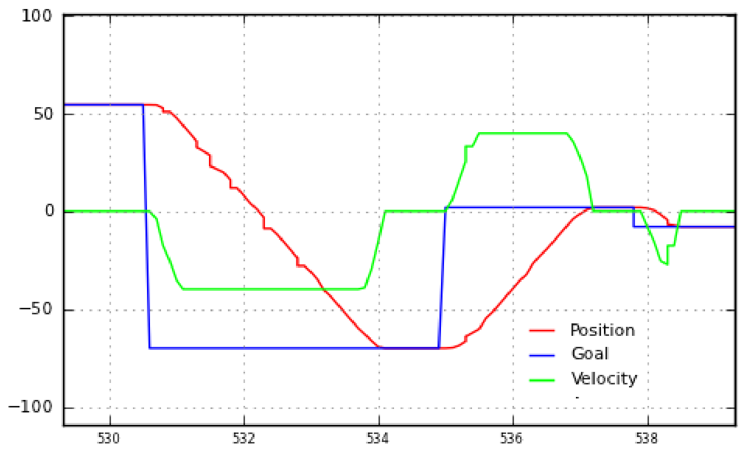


Figure 6: Output from Phidgets 1066_0, showing position command and open-loop feedback for position and velocity

The TF (transform) API of ROS was used to represent the time-varying transform between the Kinect and the rest of the robot. The head controller software continuously monitors the last known position of the detected person, and directs the pan mount to move to that angle. The head controller repeatedly receives open-loop feedback from the Phidgets 1066_0 and publishes a transform incorporating the open-loop feedback.

Comment [W23]: New terminology. Define. Your work? Say so.

4.1 Performance

The pan mount clearly alleviates one issue with the Kinect: the limited field of view. Without the pan motion, the Kinect has an extremely limited 57 degree field of view. The pan mount provides 180 degrees of rotation, so if the pan mount is allowed to track a target, the Kinect's field of view is increased from 57 degrees to an effective 237 degrees. This represents an improvement of over 300%.

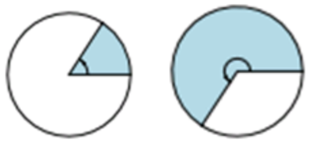


Figure 7: Kinect's effective FOV without (left) and with (right) pan mount

The performance of the pan mount was also tested under dynamic conditions. A subject stood 1.5m away from Harlie while the Kinect's RGB data was fed into a Haar cascade face detector at 2Hz. 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.

Comment [W24]: What was dynamic here? Explain.

As shown in Figure 8, 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.

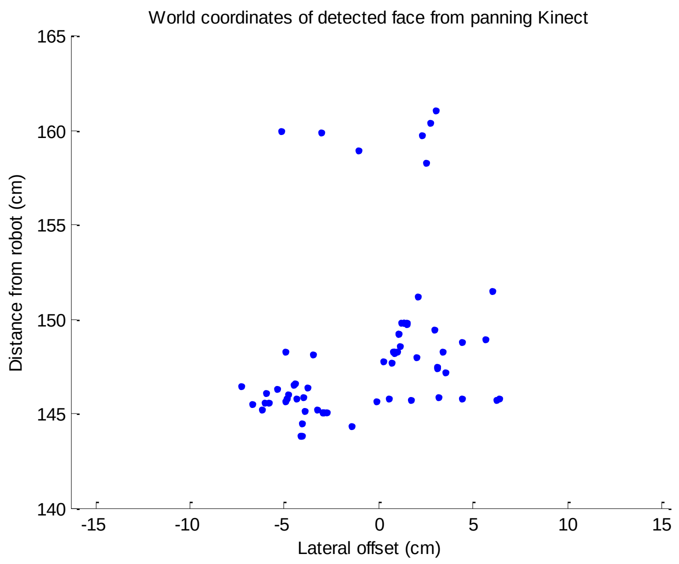


Figure 8: Performance of pan mount in detecting a stationary face

Comment [W25]: Reader cannot interpret this graph. What is "good" performance? Is this good?

To provide an update for Figure 4, the tracking performance of the Kinect was again tested, this time with motion compensation from the pan mount. Figure 9 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 is not entirely smooth. It is hypothesized that the pan mount introduces some jitter that makes tracking more difficult at low speeds. At speeds higher than 0.8 m/s, the negative effects of servo jitter are more than compensated for by positive effects in reducing relative motion. The decrease in performance in low speeds is tolerable, made up for by the increase in performance at high speeds.

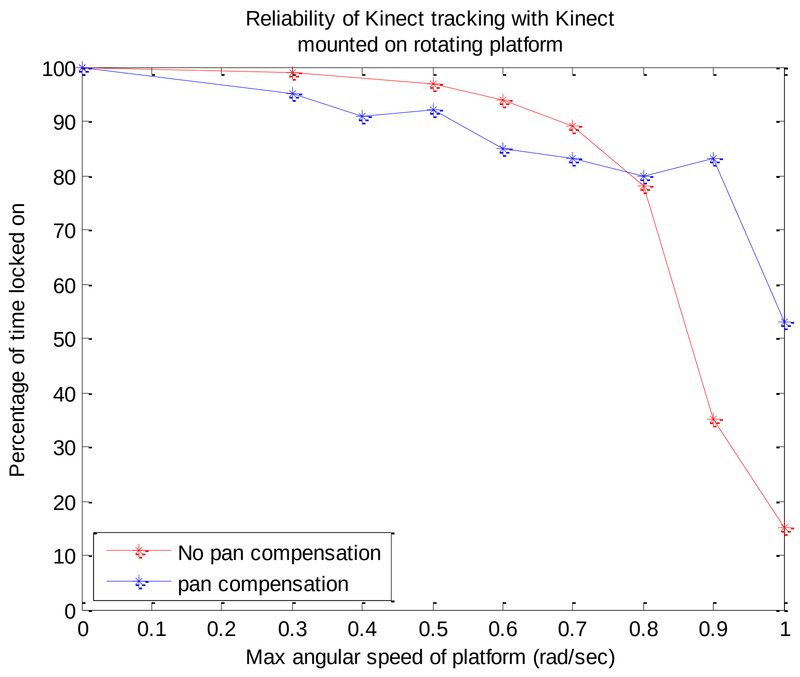


Figure 9: Tracking performance of Kinect with pan compensation

4.2 Summary

The pan mount greatly improves 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 9, the mount introduces some vibrations that decrease the Kinect's performance. While the benefits of the pan mount far outweigh the drawbacks, this could be a subject for future work. A higher-grade pan 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 vibrations arising from Harlie's dynamics. With an improved, vibration-isolating mount, I hypothesize that Figure 9 would shift, and the pan compensation would result in improvement from the baseline at all speeds.

Comment [W26]: Weren't you using passive voice?

5 Person Tracking

For reasons discussed in ~~chapter~~ Chapter 3, the Kinect alone is not sufficient to provide reliable person tracking. Even with pan compensation, the Kinect is subject to bumps and has a blind spot at close range.

To address these issues, a multi-modal approach was adopted built on the ROS people stack. A main filter node maintains a Kalman filter to track the target person, continuously publishing estimates of the user's position. Measurement nodes communicate with the filter node, attempting to associate their measurements with the filter's estimate by distance and other criteria. If a measurement node successfully associates a measurement with the filter's estimate, it publishes an observation which the filter node uses to update the Kalman filter.

Comment [W27]: Give a URL pointer

This architecture makes it easy to integrate multiple sources of observation from various sensors. For this project, three sources of observation were used: a face detector, a leg detector, and a custom body detector based on the Kinect.

Comment [W28]: You need some software architecture figures…intro something early on, and drill down in separate chapters.

5.1 Face Detector Node

The face detector is one of the nodes included in the People stack. The face detector runs an OpenCV cascade of Haar-like features on the Kinect's camera feed to detect faces [3]. It uses the full 640x480 video feed converted to monochrome, running around 2Hz. The face detector correlates its matches with depth data from the Kinect, pruning faces based on plausible sizes in 3D. Once the face detector has a list of plausible faces, it tries to associate these with the tracker from the filter node. If a face is close enough to the tracker to make an association, the face detector publishes a position measurement.

The face detector can reliably detect faces up to 8m away at sizes as small as 20x20 pixels. The face detector does not rely on persistence between frames, so it can reliably detect users when Harlie is rapidly moving. Although the face detector is very capable, it is inherently restricted to cases in which the user is staring directly at the robot. It fails to detect faces at angles. Furthermore, the face detector does not perform recognition. It detects human faces, but cannot tell one face from another.

5.2 Leg Detector Node

The leg detector is another node distributed with the ROS People stack. It detects legs using a boosted cascade of features computed from a LIDAR scan [18] [19]. The leg detector performs best at close ranges where a large number of laser returns are recorded per leg. Its performance drops off with distance. ~~(~~xxxgo into boosting from these papers)

The leg detector is especially useful at close ranges, making up for some of the shortcomings of the Kinect. At close ranges, the Kinect performs poorly because of its limited field of view and the minimum range of the Kinect's depth camera. If the user walks very near to Harlie, the Kinect cannot maintain a lock.

On the other hand, when the user is near to Harlie, each leg will have a large number of laser returns, so tracking via leg detection will be accurate. The SICK LIDAR scanner has a 180-degree field of view, so the user can 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 is a person detector that uses 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 [1]. When identifying a person, color information is an obvious first choice because of its salience and the ease of obtaining and processing it. The hue-saturation histogram was chosen to represent color information while protecting against changes in lighting intensity. In the future, perhaps the Kinect could be used to segment each individual limb, and a separate histogram could be computed for each body part.

When the system first starts up, the Kinect must be calibrated as described in section 3.2 . At the moment that the calibration is complete, a color snapshot of the user is taken (Figure 10). The hue-saturation histogram is then constructed (**Error! Reference source not found.**~~Figure 11~~). For this example, one can clearly see three major patches of color: reds and maroons for the shirt, blues for the jeans, and beiges for skin tones.

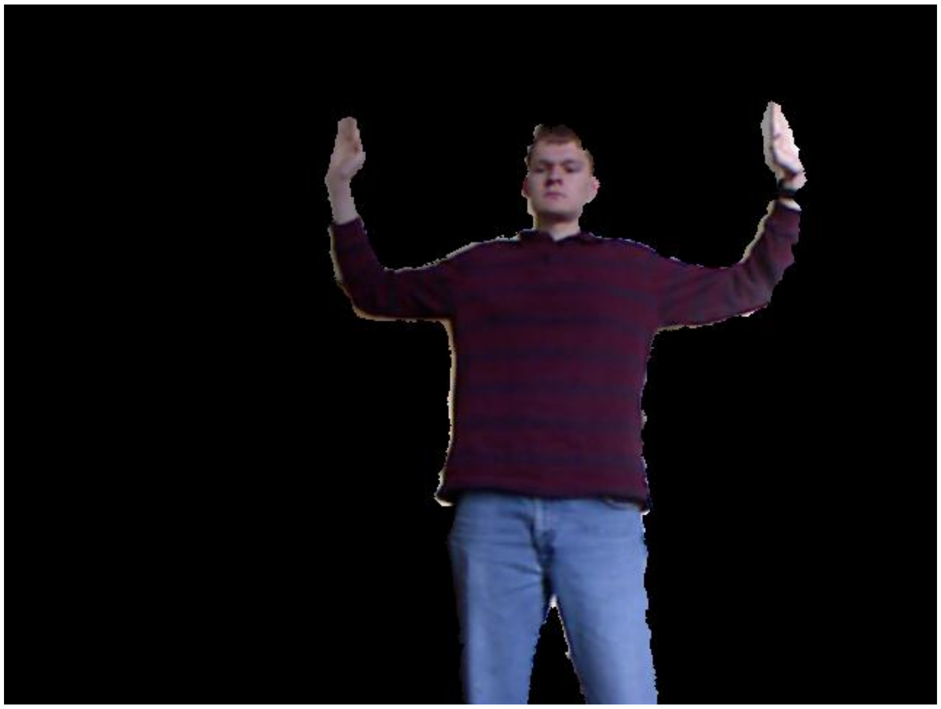
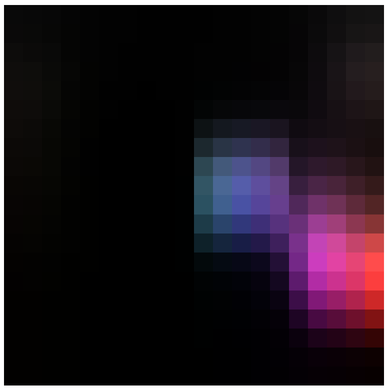
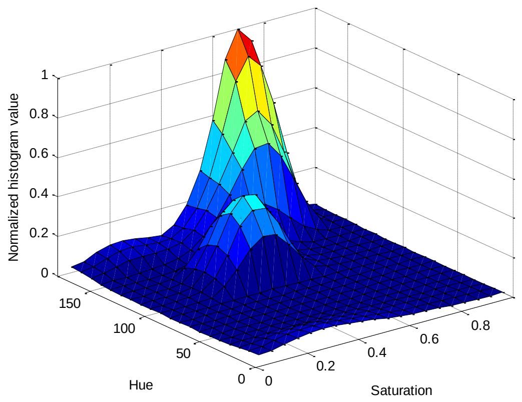


Figure 10: Kinect's RGB image masked for a user, right after calibration

 Figure 11: Histogram of Figure 10: hue on horizontal axis, saturation on vertical axis, brightness represents to histogram value.	 Figure 12: Alternate view of Error! Reference source not found.~~Figure 11~~ as 3D surface plot

The program maintains an idea of the user's current histogram, and uses it to weed out non-tracked users. In the program's main loop, the body detector receives a list of users from the Kinect [xxxIMAGE] along with a masked image of their respective pixels, as in Figure 10. The program computes a histogram for each user, and tries 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 will equal 1.0 when the histograms are identical, and will be near ~~equal~~ zero for two random histograms.

Comment [W29]: Throughout, need to clarify what is your work vs what is ROS re-use

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

Still, the user's histogram may change over time because of varying lighting colors or differences in 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 the user's appearance changing over time.

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

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

Where $\mathbf{H}_{\mathbf{track}}$ is slowly pulled in the direction of $\mathbf{H}_{\mathbf{meas}}$. With this method, however, it is possible that $\mathbf{H}_{\mathbf{track}}$ 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 $\mathbf{H}_{\mathbf{meas}}$, and $\mathbf{H}_{\mathbf{track}}$ will have a chance to adjust to the new appearance of the user. If the user suddenly drops the object, $\mathbf{H}_{\mathbf{meas}}$ will quickly change and $\mathbf{H}_{\mathbf{track}}$ will no longer be valid. To account for cases such as this, if $\mathbf{H}_{\mathbf{meas}}$ is not successfully associated with $\mathbf{H}_{\mathbf{track}}$, then $\mathbf{H}_{\mathbf{meas}}$ is compared to the original calibration $\mathbf{H}_{\mathbf{cal}}$. If $\mathbf{H}_{\mathbf{meas}}$ is associated with $\mathbf{H}_{\mathbf{cal}}$, then $\mathbf{H}_{\mathbf{track}}$ is shifted back toward $\mathbf{H}_{\mathbf{cal}}$ with a second low-pass filter and the association with $\mathbf{H}_{\mathbf{meas}}$ is attempted again.

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

Figure 14~~Figure 13~~ 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 the room for fifteen seconds. The correlation of the user to $\mathbf{H}_{\mathbf{track}}$ and $\mathbf{H}_{\mathbf{cal}}$ was recorded and plotted. The user's correlation to $\mathbf{H}_{\mathbf{cal}}$ is inconsistent and it drops below 0.7 in places. This is due to variations in room lighting and the different body silhouettes that the user exposed to the camera over time. However, the user's correlation to $\mathbf{H}_{\mathbf{track}}$ remains above 0.9 for the entire duration of the test. Thus, it is concluded that the low-pass filter is helpful in adapting to the changing appearance of the user.

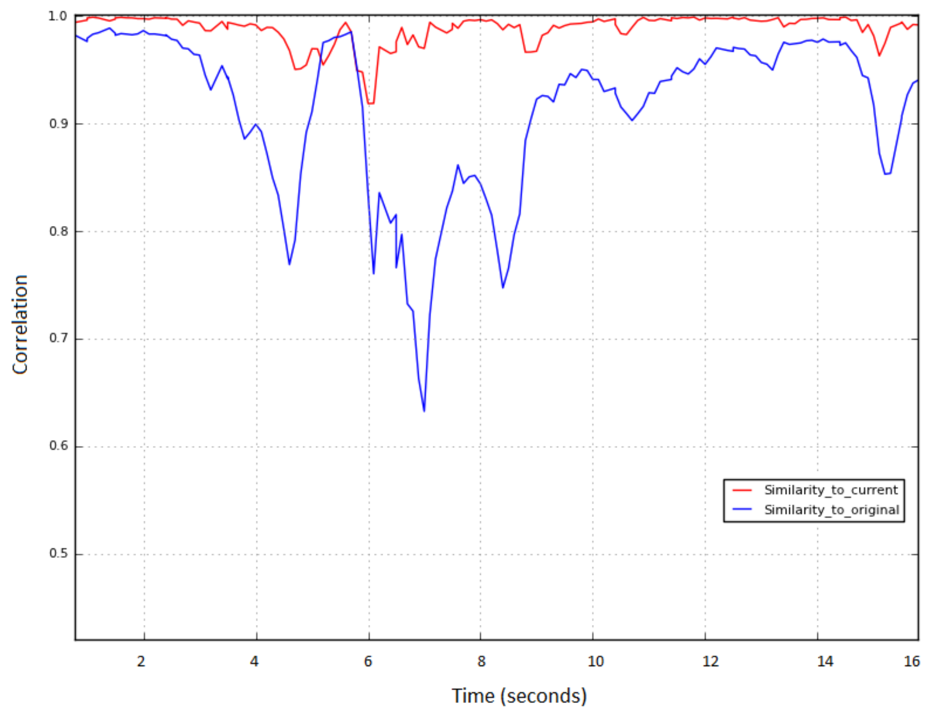


Figure 13: Correlation over time of user's histogram with H_{track} (red) and H_{cal} (blue)

6 Planning

A major component of this project involved dynamic path replanning. Previous attempts at person tracking at CWRU have used the ROS navigation stack, although these attempts failed due to the limitations of traditional planning methods. While traditional point-point planning is fine for static navigation, such as a tour-guide robot moving through a fixed series of poses, point-point planning is not suited for following dynamic targets. When tracking a person, a traditional point-point planner would need to replan every time that the person moves. This would require the robot to halt every time that the target moves, resulting in unacceptable stuttering. This project combined a point-point planner with an intelligent rolling-window approach that successfully addresses these issues.

6.1 Point-point planner

This project's dynamic replanning is built on top of a point-point planning algorithm 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 [21] [22].

Comment [W30]: Spend some time explaining SBPL; include illustrative figs; discuss parameters—max curvature, how many radii considered, ... explain your parameter choices

The SBPL planner is a search-based, ARA* planner that operates in 3D (x, y, θ) space. The x-y plane is discretized with 2.5cm square resolution, and angles are discretized with resolution $\pi/8$. The planner constructs paths from a pre-defined library of motion primitives that may be chosen to correspond to plausible motions of the robot. A cost can be separately assigned to each motion primitive, possibly to prefer wide arcs and straight paths and to penalize backing up. As a result, the SBPL planner produces nice, kinematically feasible paths (Figure 14~~Figure 12~~). Previous work at CWRU involved planning using path segments (lines, arcs, spin-in-place), which were a natural fit for the SBPL planner's motion primitives. Figure 15~~Figure 13~~ shows motion the primitives customized for Harlie, including forward and reverse line moves and arc moves of two different radii. Spin-in-place moves are not shown.

At every pose along the path, the robot's boundary is checked for collision against a 2D obstacle map of 2.5cm resolution. The SBPL planner is fast in normal operation; a typical runtime for planning several meters in a relatively clear setting is 0.1-0.2 seconds. The runtime increases for difficult moves, especially those requiring backward motion or squeezes for tight spaces, although the runtime rarely exceeds 1.5 seconds. Thus, the SBPL planner has the speed necessary for dynamic replanning.

Comment [W31]: Can you handle reverse motion?

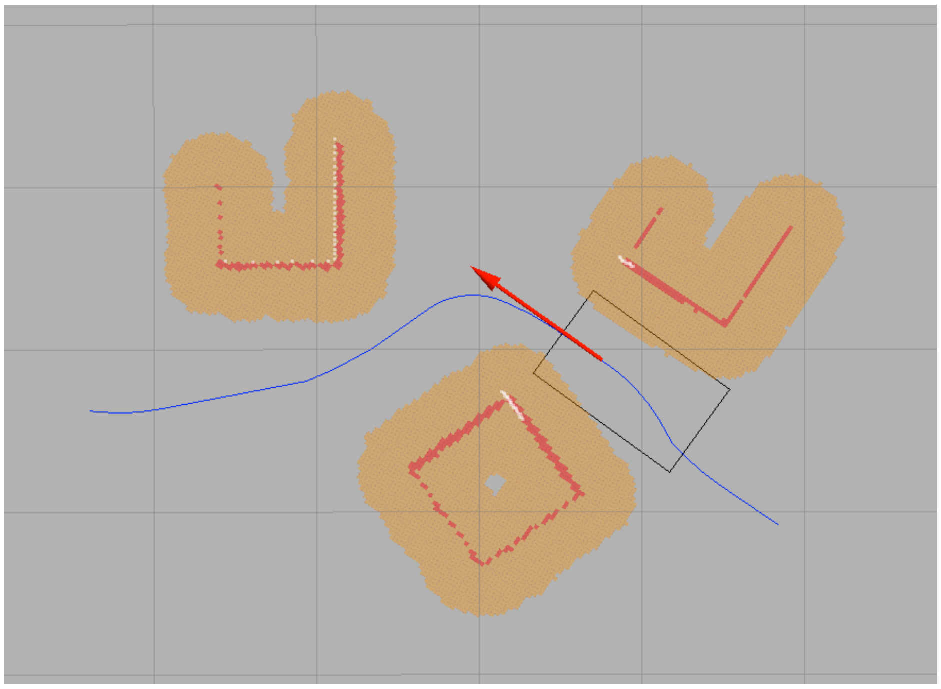


Figure 14~~13~~: Smooth path produced by SBPL planner in presence of obstacles (grid size 1m)

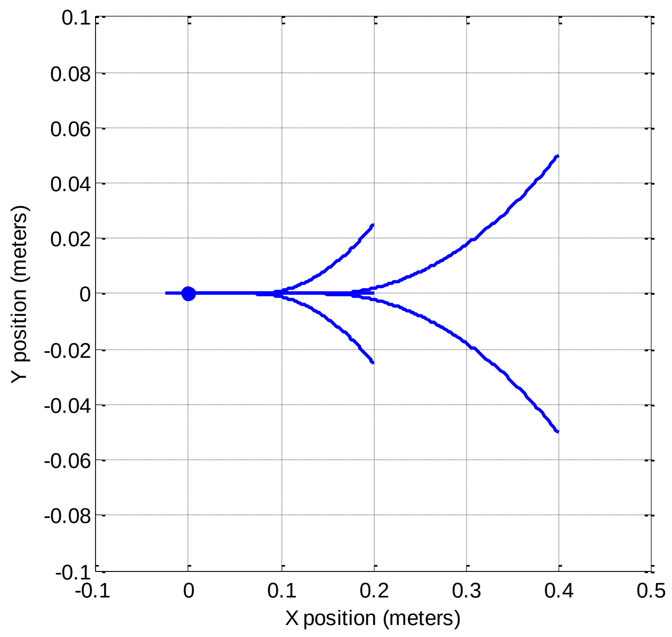


Figure 15~~14~~: Harlie's motion primitives

Modifications for this project included a motion primitive file customized for Harlie. The output of the SBPL planner was converted from a series of points to the CWRU path segment standard. Discretization error relating to the planner's 2.5cm grid was also corrected.

6.2 Dynamic Planning

A major portion of this project involved the creation of a dynamic replanning algorithm to track ~~This project created a planning algorithm to enable dynamic replanning and the tracking of~~ a moving target without the robot coming to a halt. At the heart of the algorithm is a rolling window which divides the ~~A~~

~~rolling window approach splits the~~ robot's path into two sections, referred to as the committed path and the uncommitted path.

Comment [W32]: You need to have a figure to help explain this

The committed path represents a short-term plan that is passed off to Harlie's steering~~,~~ which cannot be modified without bringing the robot to a halt. The committed path is nominally 1m long, just enough to keep the robot moving for 1-2 seconds.

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 can be changed without penalty as long as its starting pose is constrained to the end of the committed path.

The planner continuously monitors the committed path, trying to keep its length to around 1 meter. If the length of the committed path drops below a threshold, path segments are shifted from uncommitted to committed. If the committed path runs out (possibly the robot is taking a long time planning) the robot simply comes to a halt. Setting the nominal length of the committed path involves a tradeoff. If the committed path is too long, the robot will lose flexibility in planning to the target by committing to a path that may be unsuitable in the future. If the committed path is too short, the robot will run out

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

Table 1: Conditions for Replanning

of path before it is able to replan to the moving goal, causing the robot to come to an early halt.

When the planner gets a new goal from the person tracking module, it attempts to perform ~~perform~~ a partial replan from the end of the committed path. A partial replan is also triggered if a potential collision is detected along the uncommitted path.

If a partial replan fails, Harlie ~~the robot~~ is brought from a halt and a full replan is performed, planning from the halt pose. A full replan is also triggered as an emergency reflex when a potential collision is detected along the committed path. A full replan is also done when the robot is at rest and there is no committed path. ~~-~~Finally, to improve the performance of planning when the target is near to the robot, a full replan is performed if the target to be tracked moves behind the robot. In this case, it is less painful to bring robot to a complete halt than to follow the previously committed path.

Comment [W33]: This is all good, but I would expand on this and explain each of these features in more detail

Formatted: Indent: First line: 0"

6.3 Goal Generation

For the purpose of person tracking, special consideration ~~must~~ must be given to goal generation. Goals are received from the ~~person~~ person-tracking ~~module's Kalman filter (~~module as discussed in chapter 5.1 ~~)~~. As-is, these goals are unsuitable for planning. It would be impolite for the robot ~~to attempt~~ to plan directly to the ~~goal~~target, because that space is occupied by the person being tracked. Thus, goals must be generated that are offset by some distance from the true goal

This project's solution was to generate a "constellation" of goals offset by varying angles and distances ~~from the target~~. The positions were chosen based on experience and simulation, to give Harlie flexibility in planning to the target (Figure 16~~Figure 14~~).

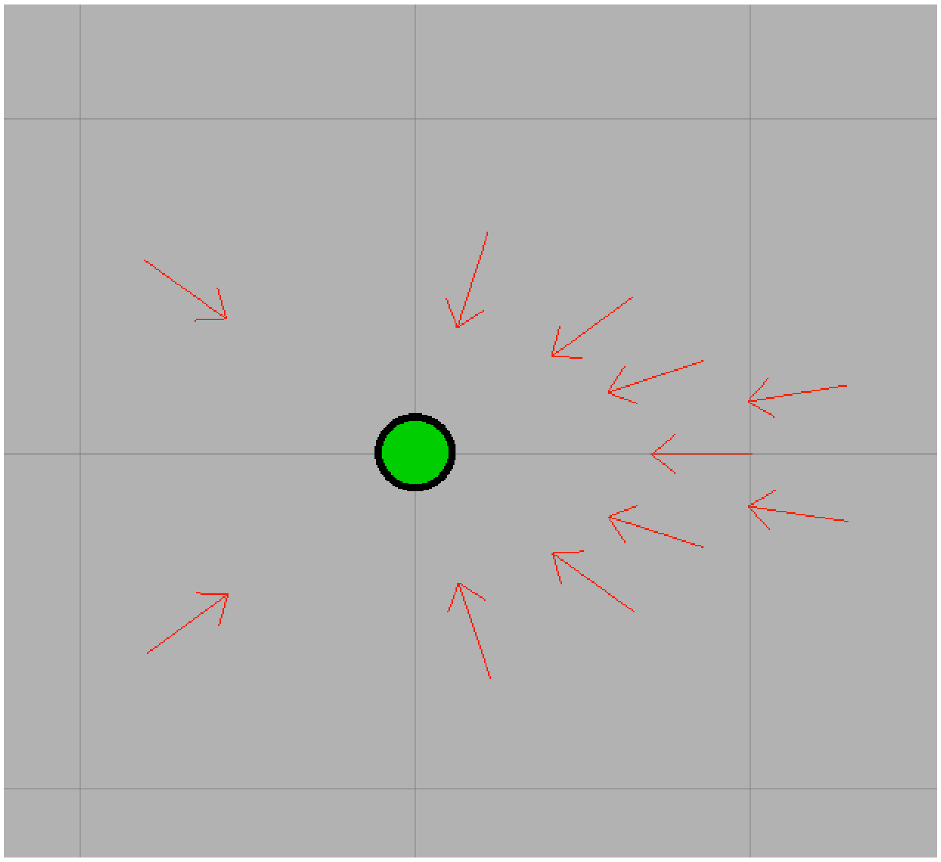


Figure 16~~15~~: Goal constellation, ~~actual~~ true goal in green (grid resolution 1m)

Upon generating the goal constellation, each goal is checked for collision~~s~~ with the robot's footprint against a 2D obstacle map of 2.5cm. Goals in collision are removed. To keep planning time reasonable, only the first several cleared goals are passed to planning. If a full replan is being performed, all goals are kept. As a special case when the target is close, less than 1m away, the robot bypasses planning altogether, and simply generates a turn-in-place path segment to rotate and face the target.~~As a special case when the target is close, less than 1m away, the robot bypasses planning altogether, and simply generates a turn-in-place path segment to rotate and face the target.~~

6.4 Future Work

Although the developed path planning algorithm performs well over medium to long distances, the robot shows some weakness when tracking users at close range. Although the SBPL algorithm produces nice, kinematically-feasible paths, it has trouble planning at extremely close distances. An example would be when the user brushes up against the robot and moves behind it. Perhaps an alternate planning algorithm could be employed at short ranges to make the response more fluid.

7 Conclusions

A person-following system was successfully developed for the mobile robot Harlie. The system uses a Kalman filter to integrate the Microsoft Kinect's person tracking capabilities with a face detector and a LIDAR-based leg detector. A dynamic replanning algorithm was developed to allow the Harlie to follow a moving target. The system works well in a real-world environment, and is able to seamlessly follow a user through a cluttered room.

Comment [W34]: Need to make this much fuller. Recap your accomplishments. Be quantitative as much as possible. You want to remind and convince your committee of your accomplishments

Works Cited

Comment [W35]: Are more coming? This is a bit light

[1] 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.

[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] 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.

[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] 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.

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[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] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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].