The Microsoft Kinect is a human interface device originally developed for the Xbox to facilitate natural user interaction. The Kinect has a 640x480 RGB camera as well as a 640x480 IR camera. An infrared projector shines a known dot pattern on the scene, 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.

The Kinect is remarkably proficient at its intended task, although when mounted on a moving base like Harlie, the Kinect is operating outside of its design parameters. The Kinect has a limited field of view (57 degrees), and normally has to be calibrated on a user before tracking can begin. The Kinect was designed to track users from a fixed position, and has trouble when used from a mobile vantage point. The Kinect is especially sensitive to sudden jolts and vibrations. The Kinect also does not work well outdoors, especially in direct sunlight (which interferes with the projected IR pattern).

The Kinect is accessed through an open-source API called OpenNI (Open Natural Interraction). However, the actual skeleton tracking is done by a closed-source binary (NITE, made by PrimeSense.) NITE provides few options for configuration, so it was not possible to probe the inner workings of the drivers and provide fixes at that level. Higher-level software workarounds had to be employed.

The Kinect has several disadvantages that had to be overcome, largely due to the closed-source nature of the skeleton-tracking software.

# CALIBRATION

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



Figure : Kinect's calibration pose

When 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, recalibration is frequently necessary. Recalibration would require both Harlie and the target to come to a halt, which is unacceptable 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 for all subsequent users to use the saved calibration.

Skipping the calibration step comes at a cost. The distinctive pose required for calibration reduces the possibility of the robot following the wrong user, because it is highly unlikely that a bystander would make the 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 classify some chairs as users. These chairs would never pass the calibration step, although without calibration they appear as spurious measurements. This issue was resolved by treating the bodies detected with OpenNI as one input to an overall Kalman filter as discussed in chapter [WHAT?].

One 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 (INSERT MECHANICAL DRAWING OF KINECT'S FOV). Luckily, OpenNI can be instructed to ignore legs and just track the target's upper torso, head, and arms. 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 : Difficulties arise in tracking a user in contact with a chair

# DISCRIMINTATION 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 my application with a moving base, frequent dropouts must be dealt with. My solution, as explained later, is to use the Kinect as one of several inputs to a Kalman filter that tracks the overall hypothesized location of a person (to be discussed in a later section.)

# 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 a target loss. When the robot once again faces the user, it will have to re-acquire the user, leading to delay.



Figure : Obstacle avoidance may lead to target loss

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 objects. In a hallway scenario, this can result in Harlie being stuck pointing at close range to a wall, within the blind-spot range. (IMAGE OF DESCRIBED SCENARIO WITH TWO-FOOT BLIND SPOT)

# 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. When the Kinect is still, performance is 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, not being able to tell users apart by means other than their spatial positions.

Although mounted on Harlie, relative velocity must be dealt with. 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 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 Kinect’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 (moving about 20cm 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 maximum angular speed.



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.

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.