Wall Following for Autonomous Navigation

Karl Bayer

Mechanical Engineering, Columbia University

SUNFEST, University of Pennsylvania, Summer 2012

Mentor: Avik De
PI: D. E. Koditschek

ABSTRACT:

Self-sufficiency in robotics motivates the need for continual improvement in autonomous navigation for mobile robots. Wall-following provides a simplification of 2-D navigation problems by providing a reference by which these problems are reduced to 1-D. A laser scanner for wall detection (abstracted as an active tangent/curvature sensor) is used on a kinematic unicycle robot, together with a controller inspired by [1] with an explicit representation of the wall as a planar curve. Reported experiments (performed using the X-RHex [2] robot) include straight wall trials to demonstrate the efficacy of the controller for a variety of initial conditions, as well as longer trials in more complicated environments including concave and convex corners up to and exceeding 90º.

Contents

INTRODUCTION 4
IMPLEMENTATION 5
CONTROLLER 6
SENSOR 6
RANSAC 7
Wall Loss 9
Curvature 9
ROBOT SPEED 10
DISCUSSION 11
CONCLUSION 17
REFERENCES 18

Table of Figures

Figure 1: X-Rhex with a laser scanner payload. 4
Figure 2: The world model is described by a Cartesian coordinate system with the wall located on the x-axis. The robot's offset distance x should approach some target offset distance T. 5
Figure 3: Unrestricted data cloud returned by the laser scanner. Points in red represent the RANSAC wall fit. 8
Figure 4: Relationship between set-point speed and actual speed on a smooth vynal tile floor using GaitRunner's default parameters. 10
Figure 5: Experimental Setup with April Tags 11
Figure 6: Three initial starting points for the straight-wall experiments. .05 meter offset and 80 degrees from the wall, 1.5 meter offset and 20 degrees from the wall, and 1.5 meter offset and 110 degrees into the wall. 11
Figure 7: Config. 1 ground truth robot path 12
Figure 8: Config. 1 Robot controller parameters 12
Figure 9: Config. 2 ground truth robot path 12
Figure 10: Config. 2 robot control parameters 12
Figure 11: Config. 3 ground truth path 12
Figure 12: Config. 3. Robot controller parameters 12
Figure 13: Laser scanner view of corner experiment with wall loss detection disabled. 13

Figure 14: Laser scanner view of corner experiment with wall loss detection enabled..... 14
Figure 15: Control parameters for corner experiment..... 15
Figure 16: Laser Scanner view of random wall experiment sequence..... 16
Figure 17: Control parameters for the extended test..... 16

INTRODUCTION:

In a world of increasingly advanced machines gaining uses in our everyday lives, we see a continuing trend driving these machines away from direct human control. Robots are continuing to take over many monotonous jobs which require little human intuition with a greater degree of autonomy. However, this trend is pushing past the easy applications like labeling a bottle in a bottling plant or applying a spot weld in a vehicle production line. As a result, there is an increasing need for an autonomy in robotics of greater intelligence and versatility. For example, the searching of a burning building, one suspected of explosive rigging, or another indoor application could be given to a robot with corresponding capabilities. Additionally, it is useful for a robot to retain the ability to perform these tasks with little or no human intervention or control. Therein we discover an application of autonomy which requires a robot to self-navigate indoor environments.

It has therefore been deemed appropriate for the RHex platform robots [2] to obtain algorithms giving them the ability to autonomously negotiate an environment by tracking a vertical surface (a navigation capability hereafter referred to as wall-following). RHex robots are hexapedal robots designed to act as platforms for sensors or other laboratory equipment to be mounted on their backs for use in extremely diverse environments and terrains. While they are intrinsically capable locomotion platforms, additional sensory capabilities are needed to solve the problems of navigation, localization, and mapping which are required to make it operable without constant human intervention. The wall-following capability aids navigation by essentially reducing a two-dimensional navigation problem to a one-dimensional problem by adding a motion constraint such that the robot must maintain a desired stand-off distance from a wall in an indoor environment.

The project seeks to achieve the fastest possible robust wall following behavior in typical indoor environments with corners and large obstacles. We seek a method which does not rely on tactile sensing with strict offset limitations. Rather, the detection of the wall will be done by a planar laser scanner which has a viewing radius of less than four meters. Hence, the robot has a larger “look-ahead” distance on the surface of the wall when it is closer to the wall.

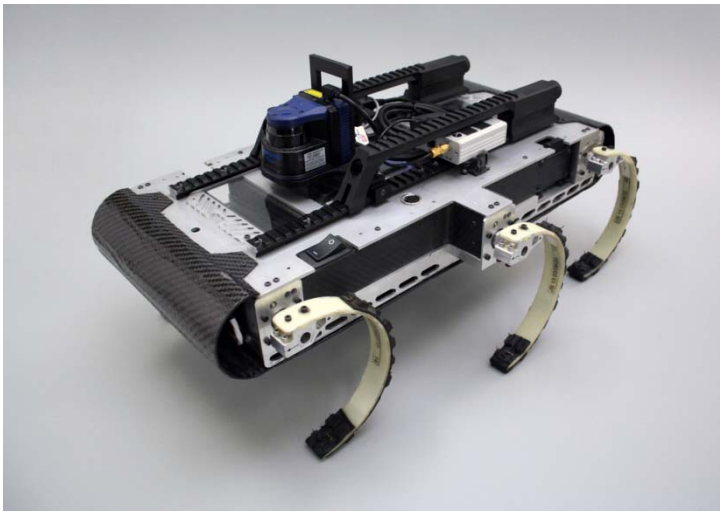


Figure 1: X-Rhex with a laser scanner payload.

Tasks relating to robot interfacing and control are already completed and are regarded as previous work done. This includes GaitRunner, a predefined Buehler clock which coordinates the legs' movement for robot locomotion, established on the software packages with RHex robots. Gait parameters [3] are left at their default values optimized for power efficiency in robot speeds around one half body length per second.

IMPLEMENTATION:

The world model assumes that there is always a wall somewhere within the view of the laser scanner, and makes no reservations for the absence thereof.

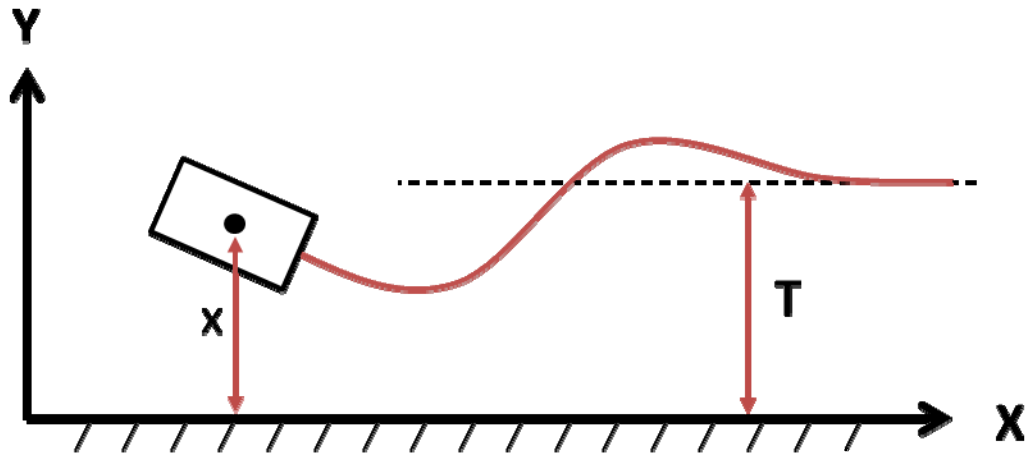


Figure 2: The world model is described by a Cartesian coordinate system with the wall located on the x-axis. The robot's offset distance x should approach some target offset distance T.

A wall is defined as a planar curve. Allowances are made for extreme variations in this definition corresponding to the huge variation in testable (structured) real-world environments including gaps, corners, and obstacles. We model our robot as a simple kinematic unicycle with state:

$$\boldsymbol{q} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \quad \textbf{(1)}$$

where (x, y) is the Cartesian location of the robot's center within the world model and θ is the robot's yaw angle as measured from the wall, or the world model's X-axis.

There are two main abstractions in the wall following algorithm: the Sensor and the Controller. The sensor refers to all methods required to pull data from the robot's environment, interpret a wall model, and develop a state estimate $\boldsymbol{q}$ of the robot's position relative to the wall. The controller dictates the movement of the robot based on the given wall model and an the robot's forward speed.

CONTROLLER:

The Controller abstraction assumes that a wall state estimate is always available, even if the sensor must fudge the data to fit the world model. Given the wall offset $\boldsymbol{q_2}$ and angle $\boldsymbol{q_3}$, the controller gives the desired turn value ω (desired rate of change of the wall angle) as:

$$\omega = (-k(sin(\boldsymbol{q}_3) - (\boldsymbol{q_2} - \boldsymbol{T}) + \boldsymbol{K})\boldsymbol{v} \quad \textbf{(2)}$$

where $\boldsymbol{v}$ is the robot's forward speed, k is a tunable control parameter and $\boldsymbol{K}$ is the curvature estimate. The default value for k is set to 3. This is an intuitive decision based on a number of experimental observations (see the Experimental section). Additionally, this report is intended to state the functionality of the controller, but does not attempt to provide analytical proof of its stability. A publication is currently in progress which will provide a proof of global Lyapunov stability given several conditions including bounded curvature.

SENSOR:

The sensor abstraction includes one physical component, a Hokuyo Scanning Laser Range Finder [4] also referred to as LIDAR (Light Detection And Ranging) mounted to the back of the robot. The laser scanner returns an array spanning 240º every 28 ms with a distance detection error of approximately 1%. All other components of the sensor are virtual implementations.

This project avoided the implementation of a very specific 'wall model' since time limitations on the project preclude us from creating a model accurate enough to describe the wide variety of wall situations we will attempt to navigate. We postulate that by limiting the sensor's view to a specific area, we increase the likelihood that only one wall lies within that view. A tunable linear model coupled with the 'limited-view assumption' yields an acceptable model versatile enough for the purposes of the wall-following project. This assumption does not account well for extreme changes in the wall (sharp corners), so additional control parameters (**$\boldsymbol{K}$**) were implemented to account for this (see Wall Loss).

We define a view path as the robot's isle of focus given by angle and width. The laser scanner data from all other paths is removed. The view path is given by a view angle from the robot's x-axis in degrees, and a path width given in meters. The data is initially filtered to remove data points smaller than .01 meters which are assumed to be errors within the laser scanner itself. By our previous 'wall' definition, all data points outside the specified view path are rejected by a set of comparators and then converted into the robot's Cartesian coordinate system.

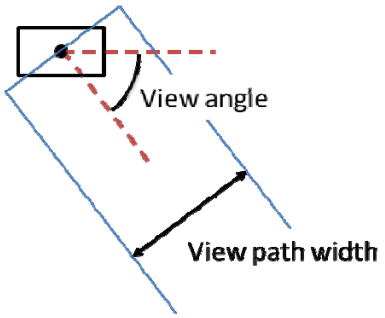


Figure 3: View path defined by angle and width

RANSAC:

RANSAC (Random Sample Consensus) is used to discover a 'wall' within the point cloud returned by the laser scanner. RANSAC uses an iterative method in an attempt to chance upon a subset of points with good characteristics for a wall:

1. Two seed points are picked at random and a line is fit to connect the two seed points using the least squares solution. Assuming **a** is an n x 2 (n is the number of data points, two in this case) matrix where each row represents an x-y data point in the robot's frame and **b** is an n x 1 matrix of ones, the least squares method solves the equation by finding a 2 x 1 vector **x** that minimizes the squared Euclidian norm.
2. All points within an orthogonal distance to the seed line are labeled as potential inliers. A linear least squares best fit line is fit to the potential inliers in the same manner as described above (except that n is the number of potential inliers) and a residual error is calculated according to: where N is the number of potential inliers.
3. If the set of potential inliers is greater than 15% of the total number of data points returned by the laser scanner, then the potential inliers' fit line and its residual error are saved as a potential wall fit.

There are two possible paths of termination of the RANSAC iteration loop. First, if the model's residual error is below .01, the set of points is very straight, and thus likely represents a wall. The second case for RANSAC termination occurs when the total number of iterations reaches 50. Experience has shown 50 to be a reasonable termination point, above which a better wall fit will not be found. This is partially due to the continuous case, described below. In either path, RANSAC will return the set of determined 'inliers', the estimated wall fit, and the residual error.

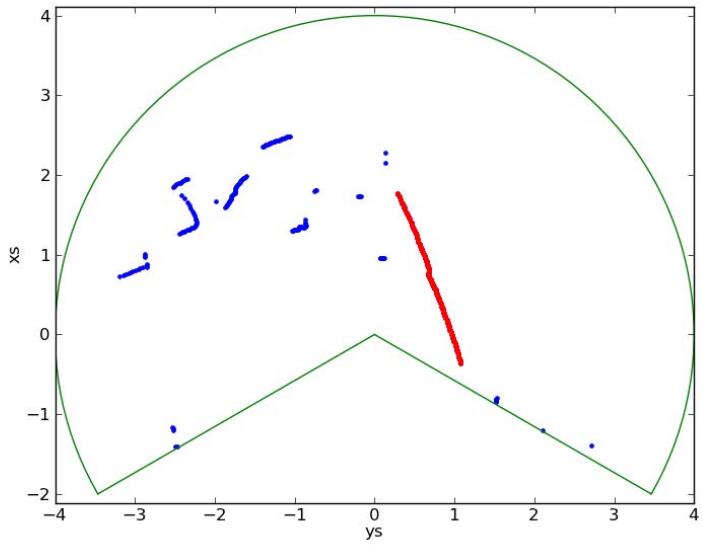


Figure 4: Unrestricted data cloud returned by the laser scanner. Points in red represent the RANSAC wall fit.

A wall model must be found and passed to the controller with every set of data points returned by the laser scanner. A full RANSAC analysis of each scan is deemed too costly in computer resources and is used only for a base case to find the initial wall. A continuous, noniterative case is convenient for every consecutive set following the initial base case. Such a case is realized by the same method as RANSAC, except for a few minor changes. Assuming the base case RANCAC performed as expected, there is a good wall fit within the view path. As a result, even if the robot is moving at high speed, the latency is so small between sequential laser scanner data sets that the 'wall' will have changed very little in the time between sequential scans. The previous wall fit is used to seed the new data set and designate the inliers for the least squares fit. Steps two and three from the iterative base case are then performed to find the new wall fit.

Should the size of the inlier sets ever drop below a ‘fit threshold’ usually fixed at 20 points, the algorithm terminates. This set point was determined by intuition, considering that slight variations in the laser scanners readings for very small inlier sets can cause drastic fluctuations in the wall fit. Experimentally, this error only ever occurred when the robot approached a situation where the laser scanner could not see around a corner. Even a limited view of the wall consistently provided an inlier fit numbering well above 20 points.

The wall fit is then translated into the robot's frame of reference giving $\boldsymbol{q_2}$ and $\boldsymbol{q_3}$, the wall offset and angle respectively. The robot’s state variables are then filtered by a Kalman Filter to smooth out the noise.

WALL DISAPPEARANCE:

The continuous case ‘seeded’ RANSAC does not produce a wall model providing accurate state estimates for the desired variety of situations. Simply averaging the points within a static, predefined view of the wall lacks the versatility to operate in greatly varied environments. Imagine the robot driving parallel to a wall, approaching a sharp corner. The view path is set at -

60º and a width of 1.5 meters. No requisites have been set to account for the disappearance of a wall behind a corner, other than to terminate the run when the set of inliers drops below the 'inlier threshold'.

One method to increase the versatility of the wall model is to modify the method by which the limited view is set. A publication is currently in progress which will propose algorithms and proofs for an active sensing model, where a second controller determines the robot's view path. In cases of wall loss, the robot would automatically look in places where it knows a wall to exist until it is in a position to sense the next wall (around a corner). It is useful for the robot to account for a loss-of-wall situation by positioning itself such that it can easily sense the next wall. In order to do this, the robot must have an additional sensor component which activates when the wall is disappearing.

Wall Loss Compensation:

A 'wall-loss' detection algorithm is implemented to serve the case described above. The purpose of the algorithm is to prevent failure of the wall-following behavior in cases where obstacles or corners inhibit the laser scanner's sight. Our world model asserts that it is impossible for a wall to simply disappear. A wall must always go somewhere, even if it turns upon itself in a full 180º. We then choose to assume that the 'unseen' wall is just outside of the view of the laser scanner and invent a cluster of data points to realize the assumption. Points are added along the line given by the robot's position and the last point in the set of inliers such that the points are added at the limit of the robot's view. The number of points added is proportional to the distance between the last point in the set of inliers and the edge of the view path. The new cluster is used to introduce a second controller parameter which represents the added points.

Curvature:

We introduce a new control parameter to provide an estimate of the wall's curvature. Note that we choose not to focus on accurate modeling of the wall. Rather, we focus on the resulting behavior. It is therefore unnecessary to obtain a high degree of accuracy for curvature. We seek a number somewhat proportional to the actual curvature and assume that parameter tuning within the controller will account for the discrepancy. Additionally, curvature for a corner is infinite, a number which is not useful for the stated purpose. Curvature $\boldsymbol{K}$ is then calculated by the following:

$$\boldsymbol{K} = \frac{2}{N}\sum_{a=.3N}^{N/2}\left(\frac{P_{f,a,y}-P_{c,a,y}}{P_{f,a,x}-P_{c,a,x}} - \frac{P_{c,a,y}-P_{b,a,y}}{P_{c,a,x}-P_{b,a,x}}\right) \quad \textbf{(3)}$$

where N is the length of the data cluster, P_c is the center point in the cluster, and P_b and P_f are points at opposite ends of the cluster, 'a' from the center point.

ROBOT SPEED:

It is nearly impossible to accurately determine the robot's forward speed in real time due to surface variance, leg compliance, and a host of other variables. The scope of this project prevents a thorough examination of the robot's actual speed and this is left as future work. It is possible to easily develop a rough estimate of the robot's forward speed using a visual ground truth mechanism. April-Tags [6] are used as this mechanism. A brief test was performed in an attempt to discover a proportional relationship between GaitRunner's set-point speed and the robot's world-speed:

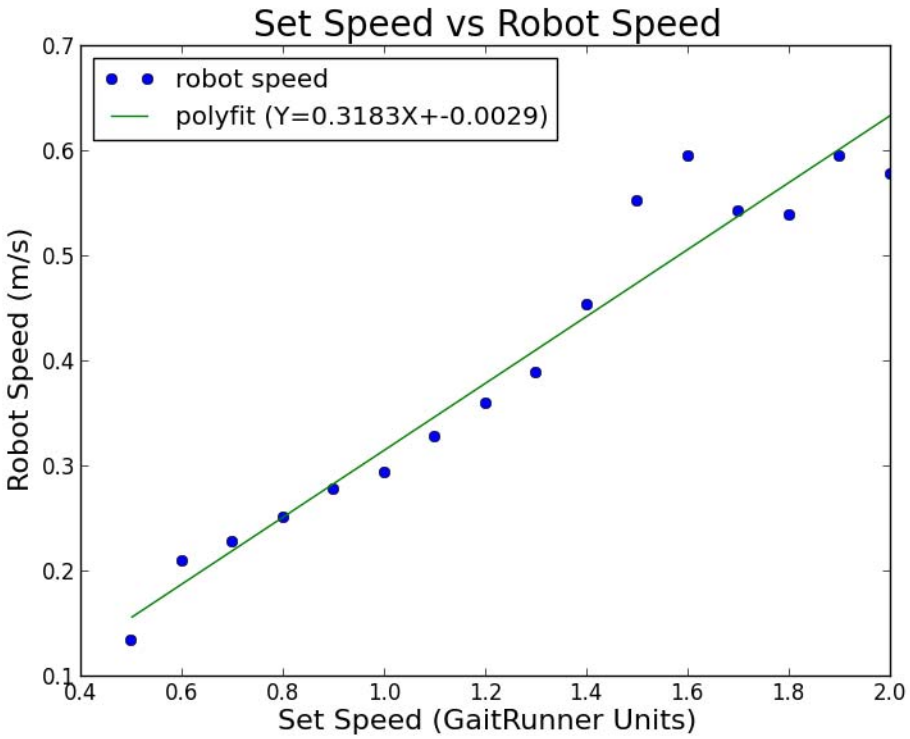


Figure 5: Relationship between set-point speed and actual speed on a smooth vynal tile floor using GaitRunner's default parameters.

The time taken to commute between two points was recorded using April-Tags for set-point speed values ranging from .6 to 2.0 in increments of 0.2. The robot's world speed is estimated by:

$$V = \frac{(P_s - P_e)}{t} \quad \textbf{(4)}$$

where t is the run duration, and Ps and Pf are two world location points given by April-Tags. A simple linear fit was set to the data, giving a constant relationship of approximately .32. The parameter $\boldsymbol{v}$ in equation (2) is then given by:

$$v = .32 * setspeed \quad \textbf{(5)}$$

Figure 5 displays an example of an early experimental setup with April-Tags. Later experiments were performed in engineered environments with a greater degree of precision.

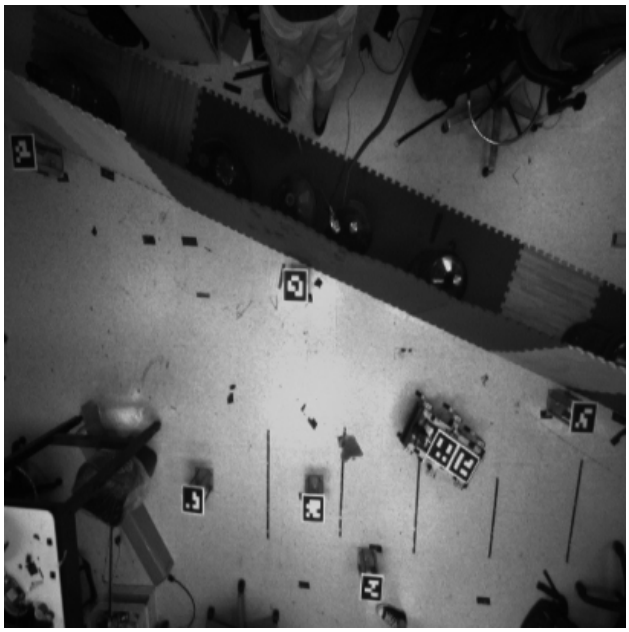


Figure 6: Experimental Setup with April Tags

DISCUSSION:

A series of straight wall experiments were performed to demonstrate the functionality of the controller in a structured environment with April-Tags. The robot was placed at three different initial conditions:

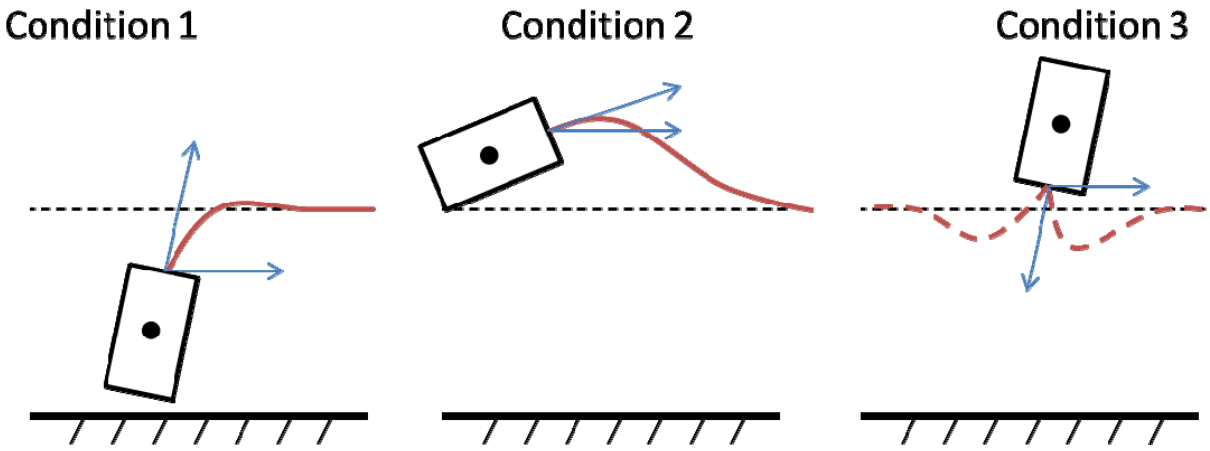


Figure 7: Three initial starting points for the straight-wall experiments. .05 meter offset at 80 degrees from the wall, 1.5 meter offset at 20 degrees from the wall, and 1.5 meter offset at 110 degrees into the wall.

Figures 7 and 8 describe a configuration 1 test result set with k=3:

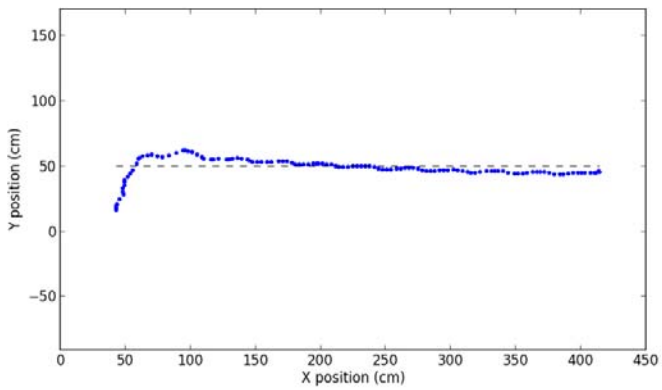


Figure 8: Config. 1 ground truth robot path

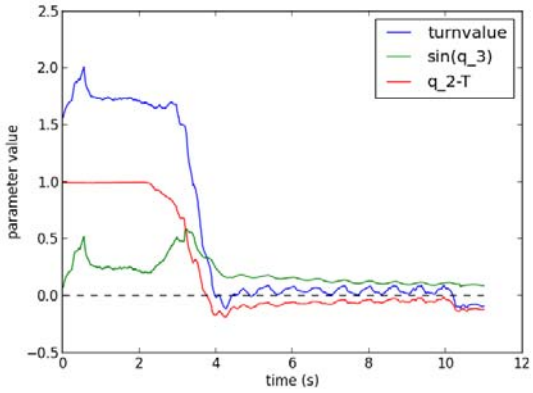


Figure 9: Config. 1 Robot controller parameters

Figures 9 and 10 describe a configuration 2 test result set with k=3:

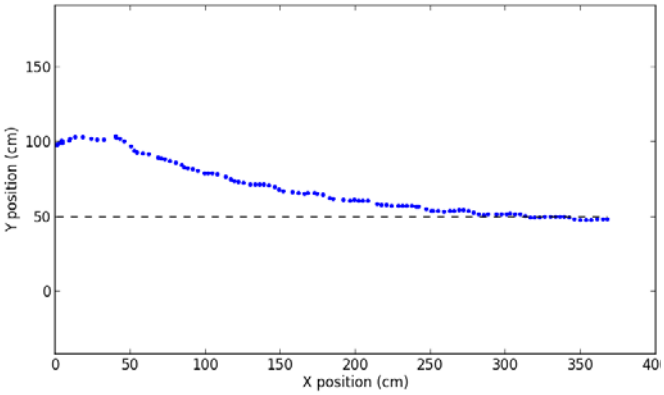


Figure 10: Config. 2 ground truth robot path

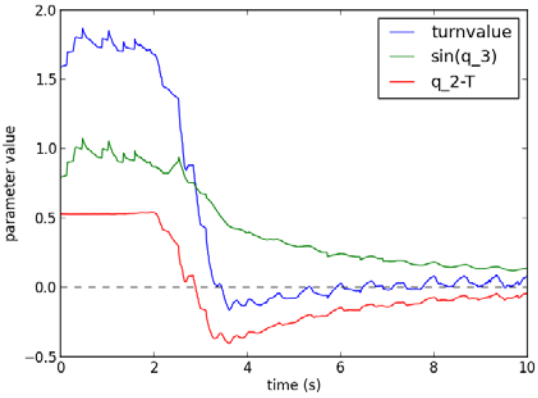


Figure 11: Config. 2 robot control parameters

Figures 11 and 12 describe a configuration 2 test result set with k=3:

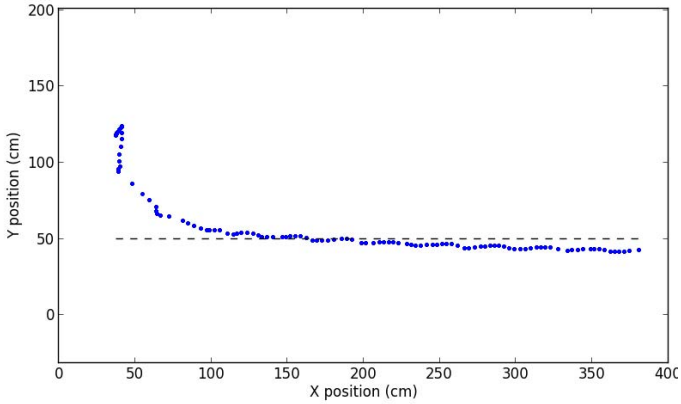
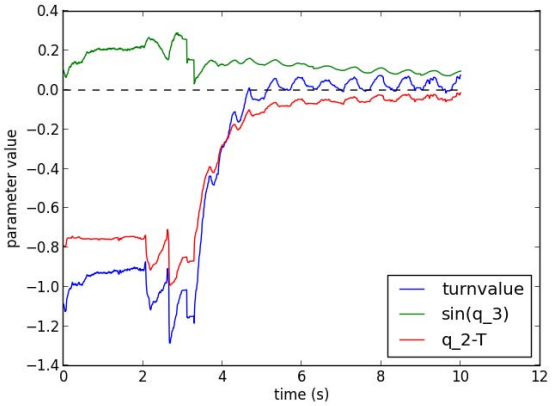


Figure 12: Config. 3 ground truth path

Figure 13: Config. 3. Robot controller parameters



In all three configurations illustrated above, it is obvious that the controller has not been fully tuned as there is some drifting in the robot's heading after it reaches the target. The robot appears to overshoot its target in each test. This problem can be easily illustrated by running the same test over a much longer distance using a non-stationary camera to follow the robot along a continuous path of April-Tags. Such a test was not attempted since the accuracy of April-Tags has not yet been determined when the camera is not stationary. However, the control parameters do appear to approach the zero value. It should also be noted that the curvature ***K*** is not relevant and was not used in the three experiments described above.

Two 90 degree corner experiments were performed to illustrate both the need for and the effectiveness of the wall-loss and curvature algorithms. The first is a 90 degree corner attempt without the wall loss detection. Shown below is a sequence of laser scanner cloud images from the robot's view of the 90 degree corner. Each dot represents a data point returned by the laser scanner. Red dots represent the RANSAC inlier set and the dotted line represents the wall fit line.

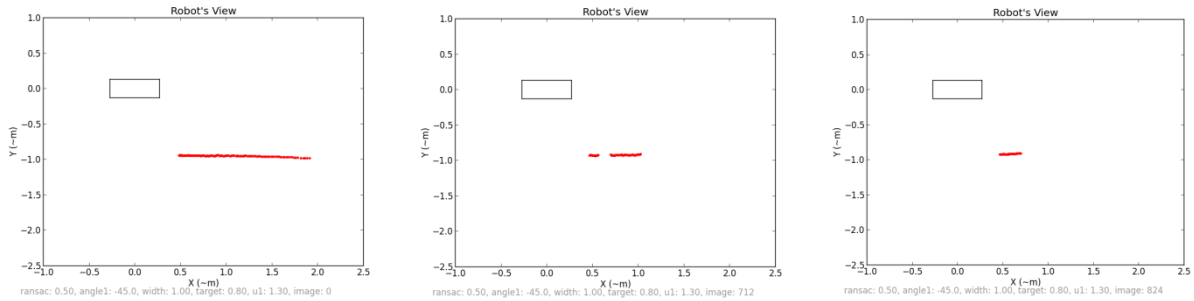


Figure 14: Laser scanner view of corner experiment with wall loss detection disabled. Each image represents one snapshot of the robot's view. Three snapshots are given at the beginning, middle, and end of the experiment.

Shown below is a similar example, but with the wall loss detection enabled. Each X in the third, fourth, and fifth images represents a point added by the wall loss algorithm.

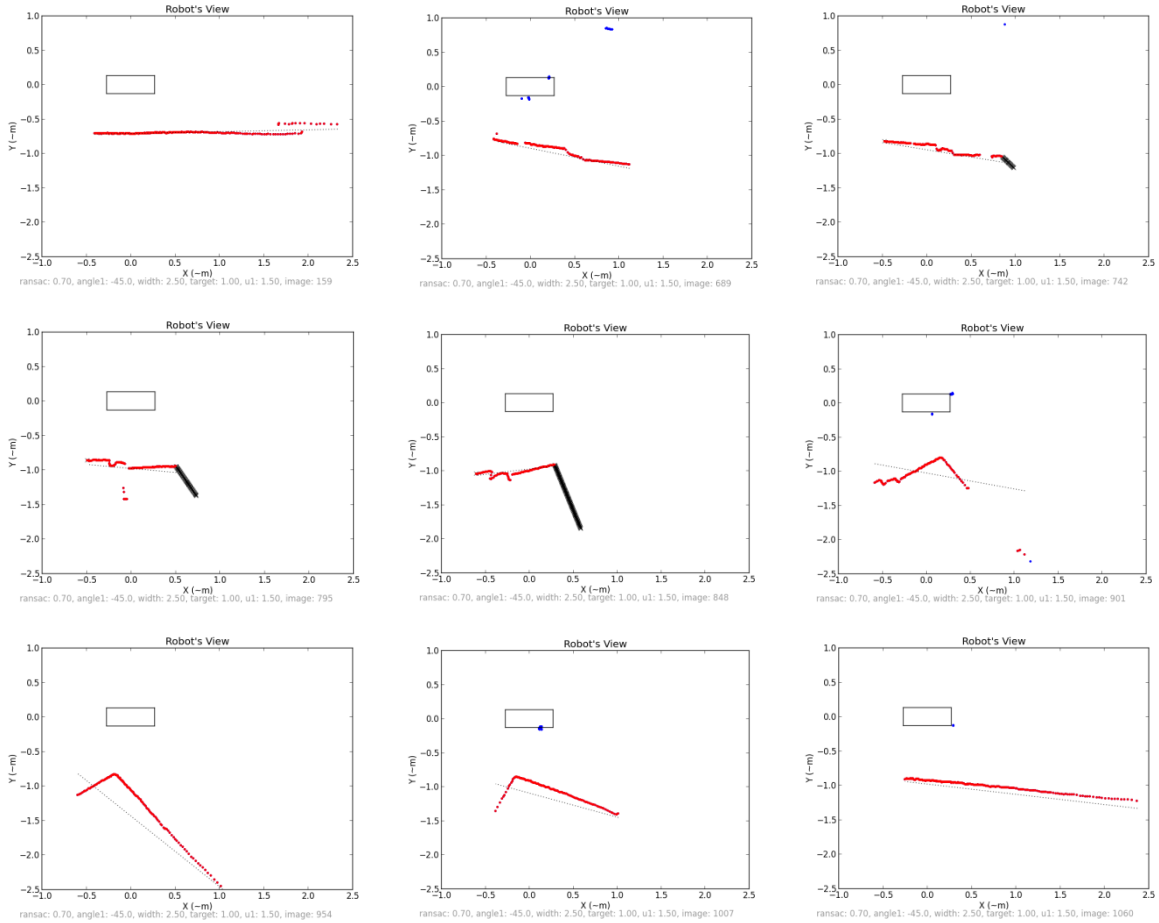


Figure 15: Laser scanner view of corner experiment with wall loss detection enabled

As depicted in the sequence above, when the wall loss algorithm detects a gap between the end of the line of points and the edge of the view path, it adds a cluster of points at the end of the existing cloud. The curvature of the new cluster is then calculated and passed to the controller.

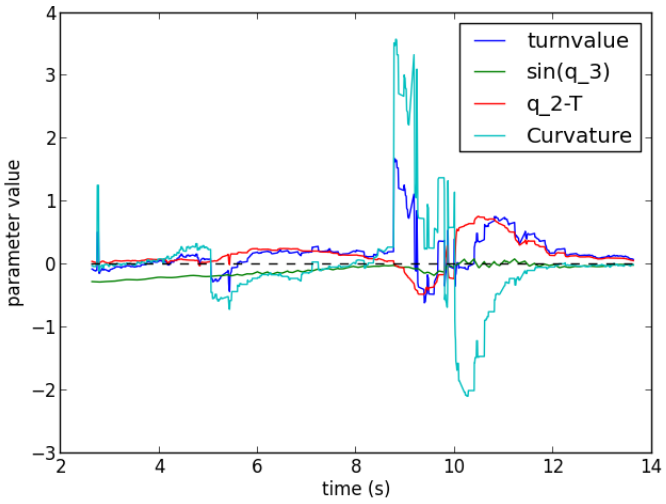
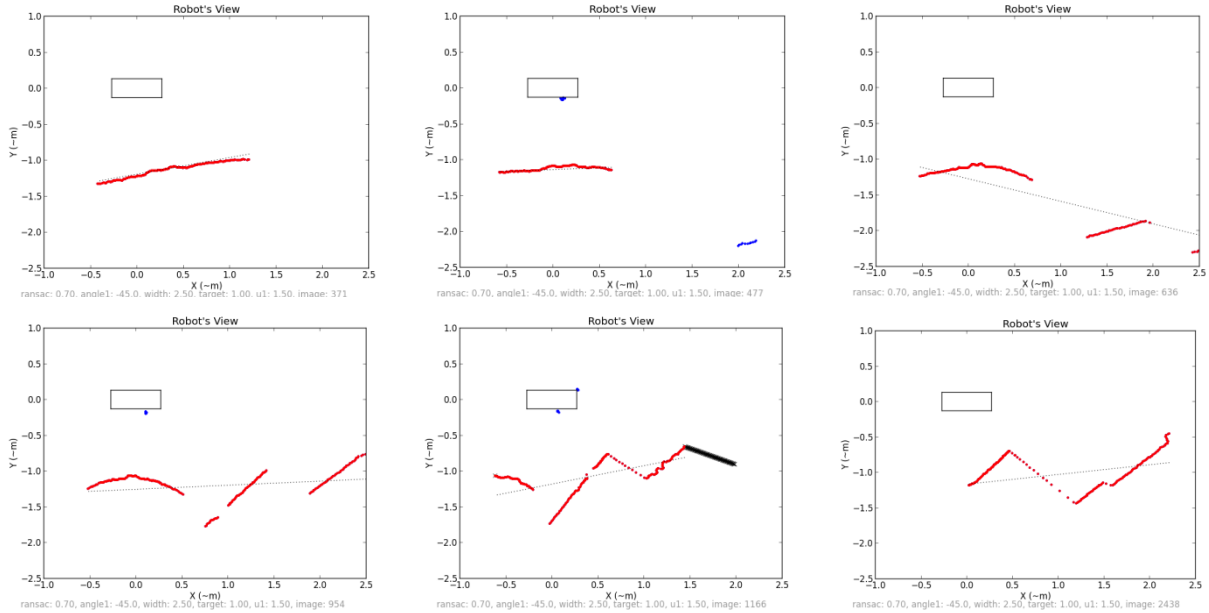


Figure 16: Control parameters for corner experiment.

Figure 14 depicts the controller parameters of the wall following experiment as a function of time. Note the spike in the curvature value at approximately nine seconds. The modification provided by the curvature value placed the robot in a position to view the next wall which the robot would not have otherwise achieved.

An experiment was performed using a wall with a high degree of variance. The wall had several corners in both directions and a small gap as well. The robot's view path was set to 45 degrees with a width of 2.5 meters. The control parameter k was set to 3, the target offset was 1 meter, and the GateRunner's speed parameter was set to 1.5. Snapshots of the robots view are shown below:



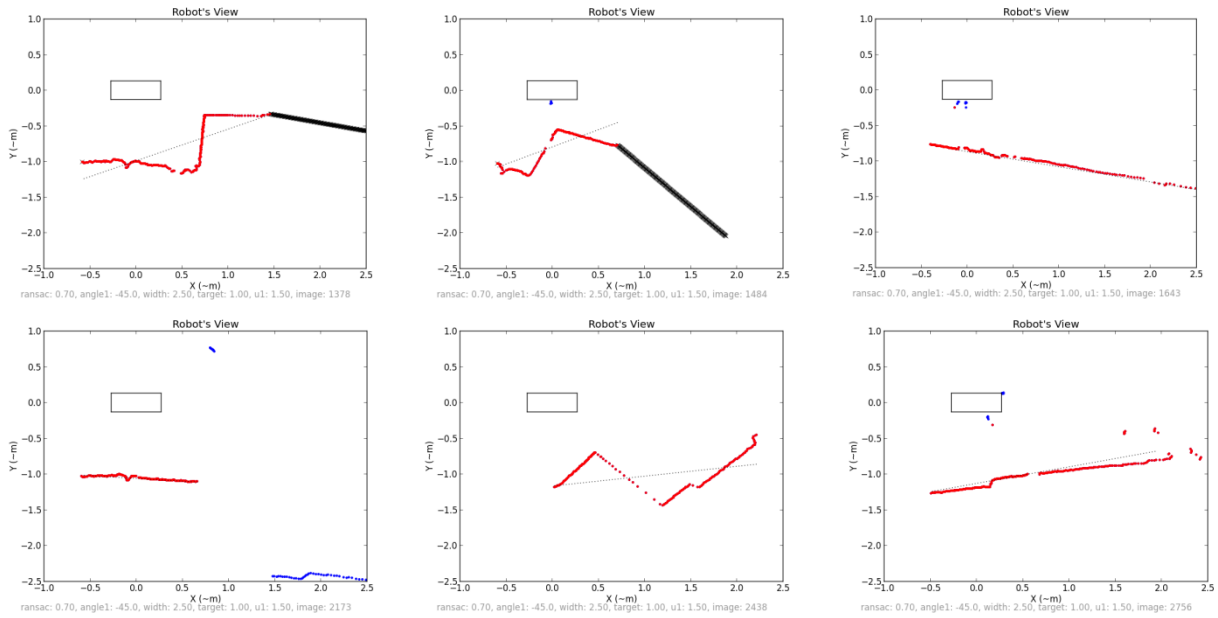


Figure 17: Laser Scanner view of random wall experiment sequence

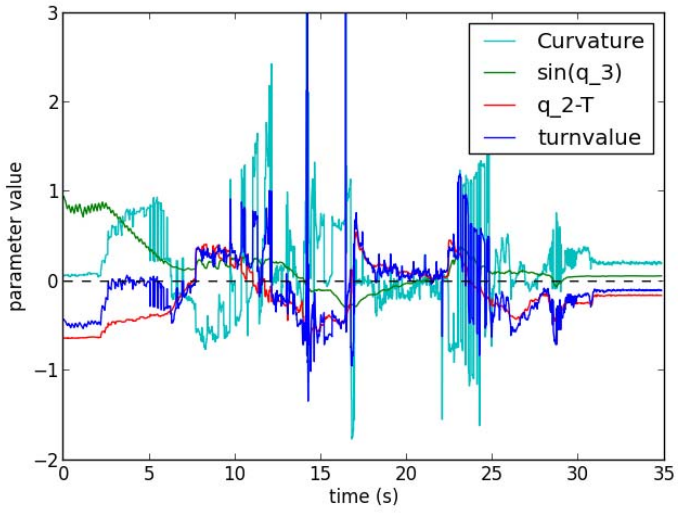


Figure 18: Control parameters for the extended test

The wall disappearance was not severe enough in the experiment above to justify the use of the wall loss compensation algorithm. However, this experiment illustrates that the wall loss algorithm does not render the controller useless in environments which are less structured than straight walls, even with strong fluctuations in the curvature as shown in the figure above.

CONCLUSIONS:

The robot must view far enough ahead so as to be able to react to significant positive changes in the wall. However, the robot must also be able to react to similar negative changes in the wall. Testing of the current algorithm illustrated that without definitions for active sensing (i.e. actively choosing the view path in real time) the set width of the view path must be very large (greater than 1.5 meters) in order to react to significant changes in the wall. Imagine for a moment that the view has been set to 45 degrees from forward. With a narrow view, it is impossible for the wall-loss detection to position the robot such that it can find a wall around a corner while retaining the current wall in its narrow field of view without a significantly increased risk of collision.

Successful indoor navigation in widely varied scenarios was obtained with the above algorithm. There are a great many possible improvements to be made which will greatly advance the algorithms behavior. Continuing work on the project will produce a robust wall following behavior which will operate without failure in a large range of environments. In addition, the behavior will be applied to augment existing autonomous behaviors such as stair climbing and landing exploration.

REFERENCES

[1] V. Kallem and N. J. Cowan, “Task-Induced Symmetry and Reduction With Application to Needle Steering” IEEE Trans. Autom. Control., 2010.

[2] K. C. Galloway, G. C. Haynes, B. D. Ilhan, A. M. Johnson, R. Knopf, G. Lynch, B. Plotnick, M. White and D. E. Koditschek, "X-RHex: A Highly Mobile Hexapedal Robot for Sensorimotor Tasks," University of Pennsylvania, 2010 [Online]. Available: http://kodlab.seas.upenn.edu/Kod/Xrhextech

[3] J.D. Weingarten, R.E. Groff, and D.E. Koditschek. “A framework for the coordination of legged robot gaits,” IEEE Conference on Robotics, Automation and Mechatronics, December 2004.

[4] Hokuyo Automatic. (2008) Scanning Laser Range Finder, UBG-04LX-F01 Specifications [Online]. Available: http://www.hokuyo-aut.jp/02sensor/07scanner/download/products/ubg-04lx-f01/data/UBG-04LX-F01_spec1.pdf

[5] E. Olson, “AprilTag: A robust and flexible visual fiducial system.” in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). May, 2011. [Online]. Available: http://april.eecs.umich.edu/papers/details.php?name=olson2010tags