Mobile Robot Navigation System in Outdoor Pedestrian Environment Using Vision-Based Road Recognition

Christian Siagian* Chin-Kai Chang* Laurent Itti

***Abstract*— We present a mobile robot navigation system guided by a novel vision-based road recognition approach. The system represents the road as a set of lines extrapolated from the detected image contour segments. These lines enable the robot to maintain its heading by centering the vanishing point in its field of view, and to correct the long term drift from its original lateral position. We integrate odometry and our visual road recognition system into a grid-based local map that estimates the robot pose as well as its surroundings to generate a movement path. Our road recognition system is able to estimate the road center on a standard dataset with 25,076 images to within 11.42 cm (with respect to roads at least 3 m wide). It outperforms three other state-of-the-art systems. In addition, we extensively test our navigation system in four busy college campus environments using a wheeled robot. Our tests cover more than 5 km of autonomous driving without failure. This demonstrates robustness of the proposed approach against challenges that include occlusion by pedestrians, non-standard complex road markings and shapes, shadows, and miscellaneous obstacle objects.**

I. INTRODUCTION

Mobile robot navigation is a critical component in creating truly autonomous systems. In the past decade, there has been tremendous progress, particularly indoors [1], as well as on the street for autonomous cars [2], [3]. Because of the confined nature of indoor environments, proximity sensors such as the Laser Range Finder (LRF) play a large role in estimating robot heading. Also, with the introduction of the Velodyne [4], which provides dense and extended proximity and appearance information, robust and long-range travel on the road [3] is now possible. However, such is not the case for autonomous navigation in unconstrained pedestrian environments for applications such as service robots (observe figure 1).

Pedestrian environments pose a different challenge than indoors because they are more open with far fewer surrounding walls, which drastically reduces the effectiveness of proximity sensors to direct the robot. At the same time, pedestrian roads are much less regulated than the ones driven on by cars, which provide well specified markings and boundaries. Furthermore, because the Velodyne is still prohibitively expensive for mass production, more widespread and affordable cameras become an attractive alternative.

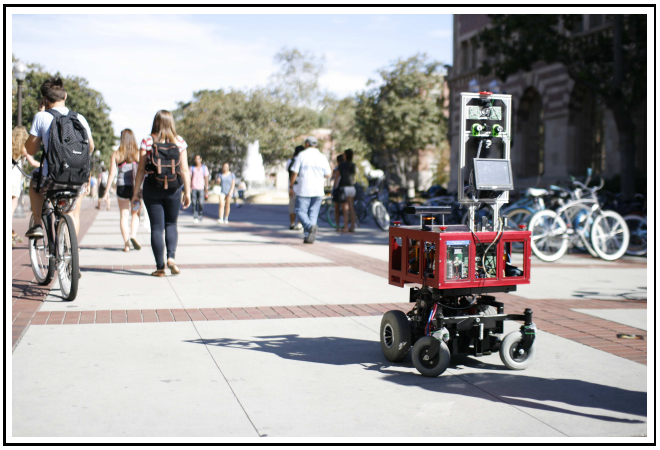

Fig. 1. Beobot 2.0 performing autonomous navigation in an unconstrained outdoor environment (college campus) and among people. The robot has to solve two sub-problems: road heading estimation and obstacle avoidance. Beobot 2.0 estimates the road heading visually, which is more difficult in this type of environment than indoors or on highways, because of complex road markings, surface textures, and shapes, shadows, and pedestrians.

One approach to heading estimation is through the use of teach-and-replay paradigm [5], [6]. The robot is first manually steered through a specific route during the teaching stage, and is then required to execute the same exact route during autonomous operation. The success of the technique depends on being able to quickly and robustly match the input visual features despite changes in lighting conditions or in the presence of dynamic obstacles. Another concern is how to get back to the set route when the robot has to deviate from it momentarily while avoiding novel obstacles (pedestrians), although recent improvements [7] have shown promising results.

Road recognition, on the other hand, not limited to a set path, is designed to readily work on any road without the requirement of prior manual driving. One approach relies on modeling the road appearance using color histograms [8], [9]. This approach assumes the road is contiguous, reasonably uniform and different than the flanking areas [10]. In addition, to ease the recognition process, the technique usually simplifies the road shape (as viewed from the robot) as a triangle. These assumptions are oftentimes violated in cases where there are complex markings, shadows, mixtures of roads and plazas, or pedestrians on the road (observe figure 1). Furthermore, they also do not hold when the road

* equal authorship.

C. Siagian is with the Division of Biology, California Institute of Technology, Division of Biology 216-76, Caltech, Pasadena, California, 91125, USA. siagian@caltech.edu

C.-K. Chang is with Department of Computer Science, University of Southern California, Hedco Neuroscience Building - Room 3641, 10 Watt Way, Los Angeles, California, 90089-2520, USA. chinkaic@usc.edu

L. Itti is with the Faculty of Computer Science, Psychology, and Neuroscience, Univesity of Southern California, Hedco Neuroscience Building - Room 30A, 3641 Watt Way, Los Angeles, California, 90089-2520, USA. itti@pollux.usc.edu

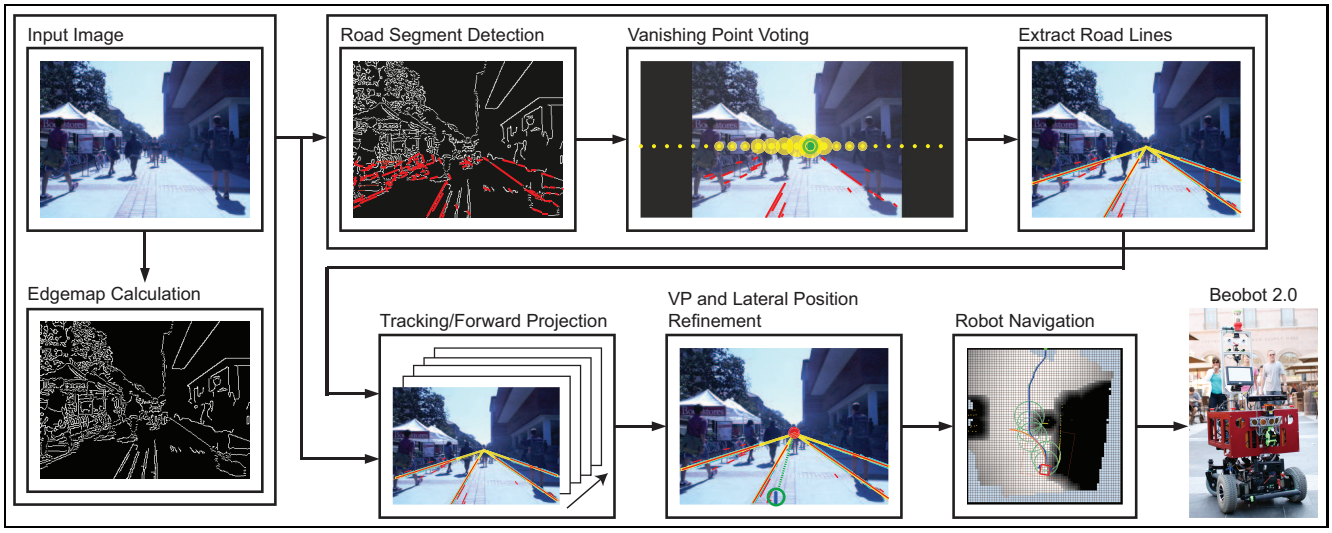

Fig. 2. Overall visual road recognition system. The algorithm starts by creating a Canny edgemap from the input image. The system has two ways to estimate the road, one is using the slower full recognition step (the top pipeline), where it performs road segment detection, which are used for vanishing point (VP) voting as well as road line extraction. The second (bottom) is by tracking the previously discovered road lines. The same tracking mechanism is also used to project forward new road lines from the top pipeline, through the incoming unprocessed frames accumulated while it is computing. The road recognition system then outputs the road direction as well as the robot lateral position to the navigation system. It then proceeds to compute a motor command, to be executed by our robot, Beobot 2.0.

appearance is similar to the flanking areas.

Another way to recognize the road is by using the vanishing point (VP) in the image. Most systems [11], [12], [13] use the consensus direction of local textures or image edgels (edge pixels) to vote for the most likely VP. However, edgels, because of their limited scope of support, can lead to an unstable result. Furthermore, these systems also attach a triangle to the VP to idealize the road shape.

Our contributions start by presenting a novel VP detection algorithm that uses long and robust contour segments. We show that this approach is more robust than previous algorithms that relied on smaller and often noisier edgels. We then flexibly model the road using a group of lines, instead of the rigid triangular shape. We demonstrate how this yields fewer mistakes when the road shape is non-trivial (e.g., a plaza on a university campus). In addition, we design and implement an autonomous navigation framework that fuses the visual road recognition system with odometry information to refine the estimated road heading. Taken together, we find that these new components produce a system that outperforms the state of the art. First, we are able to produce more accurate estimates of the road center than three benchmark algorithms on a standard dataset (25,076 images). Second, implementing the complete system in real-time on a mobile robot, we demonstrate fully autonomous navigation over more than 5 km of different routes on a busy college campus. We believe that our study is to date the largest-scale successful demonstration of an autonomous road finding algorithm in a complex campus environment.

We describe our model in section II and validate it in section III using Beobot 2.0 [14] in multiple outdoor environments. We test the different components of the system, and environmental conditions including shadows, crowding, and robot speed. We discuss the main findings in section IV.

II. DESIGN AND IMPLEMENTATIONS

We first describe the visual road recognition system, illustrated in figure 2, before combining its result with other sensory data in sub-section II-D.

The vision system first takes the input image and performs Canny edge detection to create an edge map. From here there are two ways to recognize the road. One is through a full recognition process (the top pipeline in figure 2), where the system uses detected segments in the edgemap to vote for the most likely VP and to extract the road lines. This process can take a sizable time, exceeding the input frame period. The bottom pipeline, on the other hand, is much faster because it uses the available road lines and tracks them. In both pipelines, the system then utilizes the updated lines to produce an estimated road heading and lateral position of the robot. The latter measures how far the robot has deviated from the original lateral position, which is important, e.g., if one wants the robot to stay in the middle of the road.

In the system, tracking accomplishes two purposes. One is to update previously discovered road lines. The second is to project forward the resulting new road lines through the incoming unprocessed frames accumulated while the recognition process is computing.

We describe VP detection in section II-A, the road line extraction and tracking in section II-B, and the robot lateral position derivation and estimation in section II-C.

A. Heading Estimation using Vanishing Point Detection

The recognition pipeline finds straight segments in the edgemap using Hough transform, available in OpenCV [15]. It filters out segments that are above the manually calibrated horizon line, and near horizontal (usually hover around the

horizon line) and vertical (usually part of objects or buildings). The elimination of vertical lines from consideration takes out the bias of closeby objects from the road estimation process. Horizon line calibration is done by simply denoting the line's pixel coordinate, and can be corrected online when the road is not flat using an IMU. Another way to perform online adjustment is to run a VP estimation for the whole image intermittently in the background.

As illustrated in figure 3, the remaining segments vote for candidate vanishing points (VP) on the horizon line, spaced 20 pixels apart and up to 80 pixels to the left and right of a 320 by 240 image. By considering VP's outside the field of view, the system can deal with robot angles that are far from the road direction, although not near or perpendicular to the road.

To cast a vote, each segment is extended on a straight-line to intersect the horizon line. The voting score of segment s for a vanishing point p (observe figure 3) is the product of segment length $|s|$ and the inverse proportion of the proximity of p to the intersection point of the extended line with the horizon line, denoted by the function $hintercept$ in the equation below:

$$score(s,p) \;=\; (1.0 - \frac{|\overline{hintercept(s),p}|}{\mu}) * |s| \qquad (1)$$

Note that μ is set to $1/8$th of image width or 40 pixels, which is the voting influence limit, with any segment farther not considered.

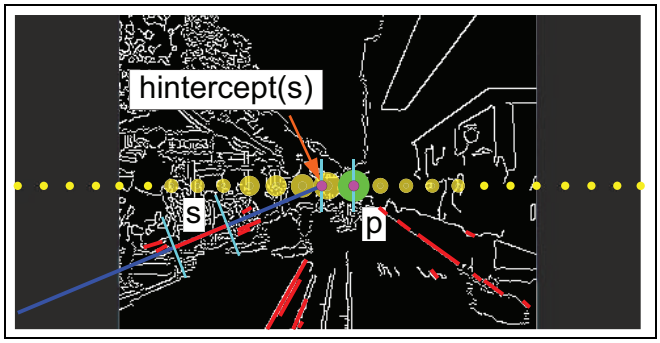

Fig. 3. Vanishing point (VP) voting. The VP candidates, indicated as disks on the calibrated horizon line with radii proportional to their respective accumulated votes from the detected segments. For clarity, the figure only displays segments that supports the winning VP. A segment s contributes to a vanishing point p by the product of its length and distance of p to the intersection point between the horizon line and a line extended from s labeled as $hintercept(s)$.

To increase the VP estimation robustness, the system multiplies the accumulated scores with the inverse proportion of the proximity to the VP location from the previous time step. Note that the system replaces values in the second term below 0.1 with 0.1 to allow a small chance for a substantial jump in the VP estimate.

$$vp_t \;=\; \arg\max_p \sum_s score(s,p) * (1.0 - \frac{|\overline{p, vp_{t-1}}|}{\mu}) \qquad (2)$$

We then use the segments that support the winning VP, indicated in red in figure 3, to extract lines for fast road tracking.

B. Road Line Extraction and Tracking

We chose a line representation, instead of storing individual segments, because it is more robust for tracking. The system first sorts the supporting segments based on their lengths. It then fits a line through the longest segment using least-squares. The system then adds any of the remaining segments that are close enough (if all the edgels in the segment are within 5 pixels) to the line, always re-estimating the line equation after each addition. Once all the segments within close proximity are incorporated, the step is repeated to create more lines using unclaimed segments, processed in length order. To discard weakly supported lines, the system throws away lines that are represented by less than 50 edgels in the map. We call this condition the support criterion.

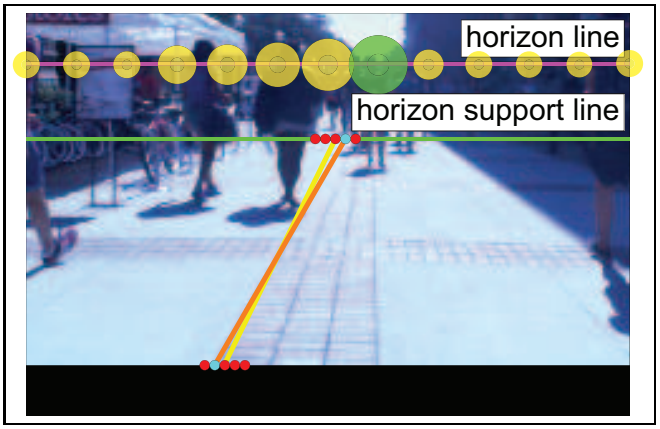

Fig. 4. Line tracking process. The system tracks a line equation from the previous frame (denoted in yellow) in the current edgemap by finding an optimally fit line (orange) among a set of lines obtained by shifting the horizontal coordinate of the previous line's bottom point (bottom of the image) and horizon support point (intersection point of line and the horizon support line) by +/- 10 pixels with 2 pixel spacing. The fitness is based on the score in equation 3. The set of candidate shifted points is shown in red on the bottom and on the horizon support line.

Given a line equation from the previous frame, and the current edgemap, the system calculates the new equation by perturbing the line's horizon support point and road bottom point, as illustrated in figure 4. The former is the line's intersection point with a line 20 pixels below the horizon line (called the horizon support line), while the latter is an onscreen intersection point with either the bottom or the side of the image. The system searches through the surrounding spatial area of the two end points to find an optimally fit line by shifting the horizontal coordinate of each point by +/- 10 pixels with 2 pixel spacing. The reason for using the horizon support line is because we want each new candidate line, when extended, to intersect the horizon line on the same side of where it came from. That is, if the candidate line intersects the bottom of the image on the left side of the VP, it should intersect the horizon line on the left as well. We find that true road lines almost never do otherwise.

The highest scoring candidate is calculated by dividing the total number of segment edgels that coincide with the line equation over total number edgels possible that are below the horizon line and in the image:

$$fitness(l) = \frac{\sum_{s<inline(l)} |s|}{totalPossible(l)} \quad (3)$$

that also passes the support criteria is the new tracked line. If none exists, the tracked line is assigned a zero score and kept as is. After each track the system assesses the condition of each line. Any newly spawned lines will be kept for observation for their first seven frames, regardless of the fitness score. A line that scores above 0.5 in at least five frames is kept and remains active until at least five of the last seven scores fall below 0.3 (hysteresis). We label the lines that have no scores below 0.3 the past 7 frames as *healthy*. In the event a new road line nearly overlaps with a tracked line, if their horizon support and road bottom points are within 7 pixels total, we keep the line with the higher fitness score.

The resulting lines can then be used to refine the VP by performing a weighted average (based on the fitness scores) of the horizon intercept points:

$$vp = \frac{1}{\sum_l fitness(l)} \sum_l fitness(l) * hintercept(l) \quad (4)$$

Note that this equation differs than equation 2 in that the latter is a voting process, where the VP candidates are spaced far apart, while here the VP is computed from road lines and can be in any coordinate in the image.

If there are no lines at the current time step, the system does not output a vanishing point. The robot will try to maintain its heading using other means such as encoders and IMU, explained in section II-D below. To integrate the VP-based heading estimation with these sensors, the system converts the VP pixel location to angle deviation from the road heading. Assuming the camera is fitted with a standard 60 degree field-of-view lens, the conversion is linearly approximated from 0 degree deviation at the center of the image to 27 degrees at the edge.

C. Robot Lateral Position Estimation

The estimated VP by itself is not accurate enough to properly drive the robot. Even though the general heading is correct, the robot slowly drifts to one side of the road, which may disturb pedestrians. To rectify this, our system locks in on the robot's initial lateral position and tries to maintain it. We decided to do this, instead of estimating the true road center, because, often times, road boundaries [11], [8], [13] cannot be visually ascertained, without additional contextual and semantic information.

When the first *healthy* lines are detected, we set the on-screen road bottom points as the canonical servo points. When the robot moves, each road bottom point moves to a new position. The lateral deviation is the horizontal difference between the canonical and new coordinates on the same vertical coordinate. Observe figure 5 for an illustration. We take the weighted average of the deviation to calculate the final estimate.

$$dev = \frac{1}{\sum_l fitness(l)} \sum_l fitness(l) * dev(l) \quad (5)$$

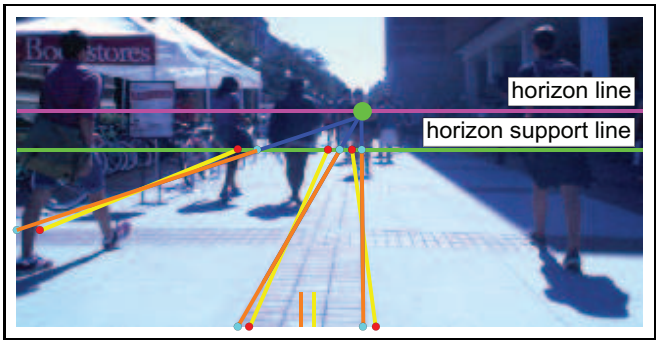

Fig. 5. Lateral deviation estimation. The system estimates the lateral deviation by averaging the road bottom shifts of all the tracked lines (denoted in orange), weighted based on the equation 3 scores. Note that this is the shift from the original line equation (denoted in yellow), when the line is first spawned. The estimated lateral difference is the distance between the orange and yellow stubs in the middle bottom of the image.

If at the time step a line is first used, the current estimated lateral deviation is not 0, the resulting difference based on it has to be offset by the deviation. When there are no tracked lines, the system sets the lateral deviation to zero and assumes the robot is near its original position. When a usable set of lines becomes available, the current lateral location is the new position to maintain. We use an empirically calibrated bottom of the image single pixel distance of 9.525mm to produce a metric lateral deviation.

The road recognition system then feeds the heading and lateral position deviation to the local map navigation system.

D. Local Map Navigation

Beobot 2.0's overall navigation system, as illustrated in figure 6, takes in the visual road recognition output, as well as Laser Range Finder (LRF) and odometry (from IMU and encoders) values. The LRF is particularly useful for obstacle detection and avoidance, while the IMU and wheel encoder data supplement the visual road finding algorithm in difficult conditions (e.g, sun shining into the camera). The local map represents the robot's local surroundings using a 64 by 64 grid occupancy map, where each grid cell spans a 15cm by 15cm spatial area. The robot location in the map is displayed as a red rectangle and is located at the horizontal center and three quarters down vertically to increase front-of-robot coverage. In addition, there is a layer of grid surrounding the map to represent goal locations outside of it. The figure shows the goal to be straight ahead, which, by convention, is the direction of the road.

To standardize the road direction estimation throughout the system, the back-end navigation module (rightmost column in figure 6) calculates the absolute IMU road heading by summing the estimated angular heading deviation from

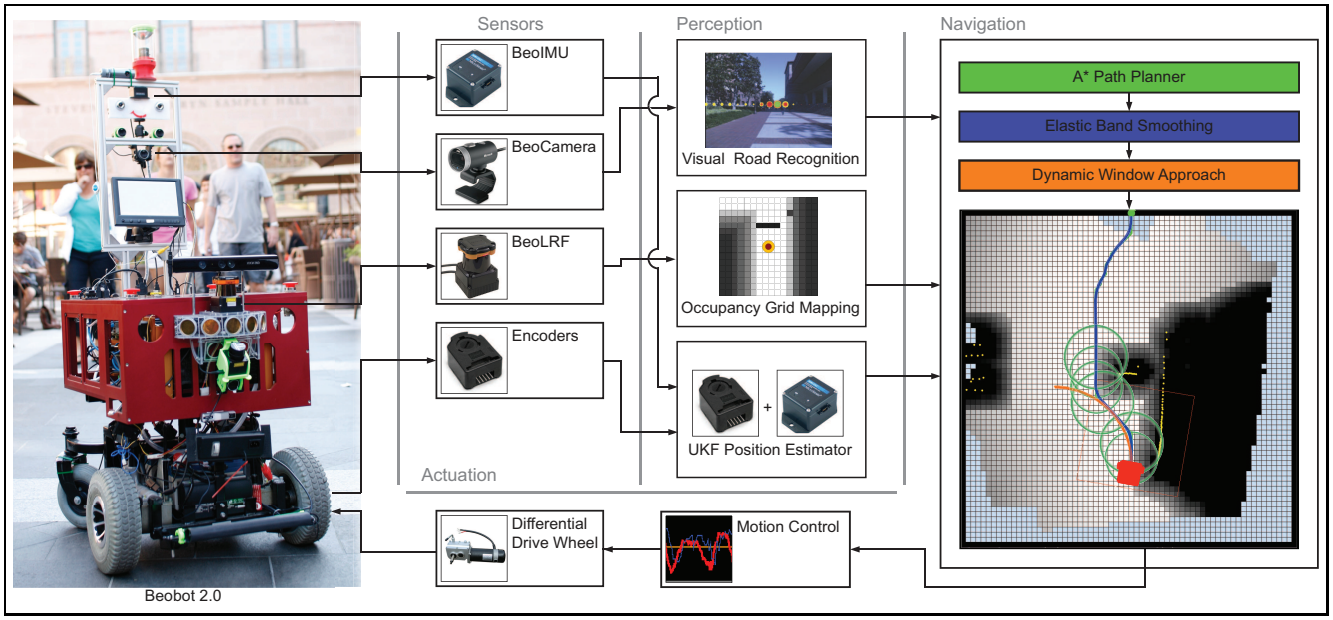

Fig. 6. Beobot 2.0 autonomous navigation system. The system utilizes sensors such as IMU and encoders to estimate robot odometry, LRF to create a grid occupancy map, and camera to recognize the road. The information is then fused to create the robot's local surrounding map, which estimates the road and robot heading, and the surrounding obstacles. To reach the goal the system computes a path using the A*, deforms it maximally avoid obstacles using the elastic band algorithm, and generates motion command that account for robot shape and dynamics using Dynamic Window Approach (DWA).

visual road recognition and the current IMU reading. In the event IMU readings are not available, the heading values become relative to the initial heading recorded by the wheel encoders. To make the estimation more robust, the module considers the last 100 absolute headings in a histogram of 72 bins, each covering 5 degrees. It updates the road heading estimate if 75% of the values reside within 3 bins by averaging the values within those bins. In addition, it also discards heading inputs that are farther than 30 degrees from the current estimate, which usually occur during obstacle avoidance by an adjacent or perpendicular road in the field of view.

By approximating the road direction in absolute IMU heading, the module does not directly couple vision-based road recognition with motion generation. Instead it can also use IMU readings to compute the robot's heading either in between recognition outputs or when the road is undetectable visually.

To correct for lateral position deviation, the module adds a small proportional bias to the heading which is calculated using the deviation over the longitudinal distance to goal:

$$ang \;=\; ang_{robot} + atan(dev/long(goal)) \qquad (6)$$

This is done because lateral correction is not an urgent priority and at times more accurate approximation can only be achieved if deviation is sufficiently large.

The grid occupancy map itself is filled by the LRF on the robot. In this setup, unlike a Simultaneous Localization and Mapping (SLAM) formulation, the odometry uncertainty is not modeled. It is not critical to do so because the robot quickly passes through the local environment and discards any briefly accumulated errors.

The module then applies A* search to the map to find a path to the goal. For each grid, the search process creates eight directed edges to the surrounding nodes weighted by the length of the edge plus the occupancy likelihood of the destination grid. To encourage path consistency, we bias the weight by adding the proximity of the destination grid to the closest point in the previous A* path divided by the map diagonal. If the distance is less than 2 pixels we zero out the bias to allow for flexibility of finding a nearby optimum path. To smooth out the path and allow for maximum avoidance from the obstacle we apply the elastic band algorithm [16].

To compute the motor command from the deformed path, the module utilizes the Dynamic Window Approach (DWA) [17]. It not only calculates the deviation between the current robot heading and the heading of the path's first step, but also takes into account the dynamics of the robot by only considering commands within the allowable space set by the previous command. Furthermore the approach then accurately simulates each resulting arc trajectory of the allowable commands, modeling the robot shape to test whether any part of the robot hits or come close to hitting an obstacle.

III. TESTING AND RESULTS

We carry out three different testing procedures. In section III-A, we test the performance of the vision road recognition sub-system against other road recognition systems [8], [11], [13] on a standard dataset [13]. We then demonstrate in section III-B how each perceptual input that contributes to the navigation system (wheel encoders, IMU, VP detection, lateral deviation estimation) affects performance. In section III-C, we evaluate the full system in real-time on a wheeled robot operating under challenging conditions (variable road

Fig. 7. Example images from the four testing sites, each of which accentuate different road characteristics. The road in Frederick D. Fagg park (FDF) is uniform with well defined red boundaries. The road in front of the Bookstore (BKS) has many complex markings, while Hellman Way (HWY) is more uniform, but curved. The road in front of the Leavey Library (LEA), on the other hand, is not explicitly delineated, and is wider than the rest. For each site we display the road in various conditions: ideal, shadowed, and crowded.

appearance, the existence of shadows, robot speed, and crowding).

For this full test, we select four key road types that encompass different visual challenges, described in figure 7. We test using Beobot 2.0 (pictured in figure 6), which is equipped with a camera and an LRF in front, 117cm and 60cm above the ground, respectively, as well as encoders and IMU to calculate odometry. Beobot 2.0 computes the canny edgemap in 3ms, hough segments in 20ms, VP voting in 1ms, line extractions in 5ms, and line tracking in 3ms on average.

A. Road Recognition System Comparison

We first test the system accuracy in recognizing road center using an image dataset of robot navigation runs by [13]. For our system, we provide the initial road center location of each run as an offset to convert the relative lateral deviation to the absolute road center.

The dataset consists of four sites – HNB, AnF, Equad, SSL – with example images shown in figure 8. The dataset provides manually annotated left and right road boundaries, which are averaged to calculate the road center. We compare the result with road recognition outputs to measure the error in pixel unit, which is equivalent to one centimeter with the dataset camera setup.

We compare our system with another VP based system [11], a color histogram contrast system [8], and one that combines both aproaches [13]. Note that all three idealize the road using a triangular shape template. Figure 8 displays the results.

Our system produces a better road center estimate on all environments, on average, within 11.42cm of the ground truth. It is generally more robust in instances where the road shape are not triangular or when there is only one visible road boundary, when the robot swerves away from the road center, as shown in the second sample image in figure 8.

In addition, the system also benefits from the decoupling of VP and road center estimation by directly and separately computing them from the tracked road lines. In other systems [11], [13], road center is estimated by extending a pair of road boundaries (left and right side) from the VP. If the

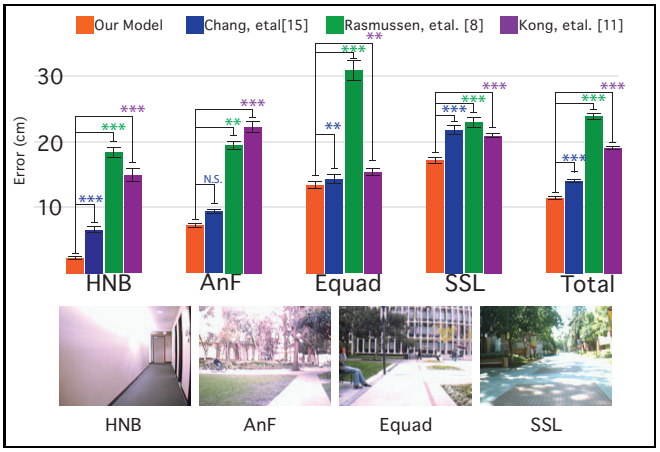

Fig. 8. System comparison bar chart with example images of the four environments. The bar graphs illustrate the performance of our algorithm as well as [11], [8], [13] in estimating the road center. Performance is measured by the difference between the ground truth and estimated road center in cm. Two stars indicates statistical significance p-value of $p < 0.01$ and three stars $p < 0.005$.

VP estimate is incorrect, the road center usually falls on a random point in the image.

In some instances, such as in over-saturated or blurry images, there may not be clear lines in the image because the detected segments are not sufficiently aligned to form even one line. In these cases, our system only provides a vanishing point from the segment VP voting instead of producing erroneous road shape when there is none visually.

B. Individual Part Contribution Testing

We assess the contribution of the different perceptual inputs in the navigation system by systematically adding each of them, from the simplest to the most computationally complex. We first rely on encoder-only odometry, then IMU and encoder-fused odometry, then adding VP, and finally adding lateral position estimation. We start the robot in the middle of the road and measure its lateral deviation as it navigates through the road. We measure the ground truth by manually recording the robot location at certain interval and interpolating the positions in between. Testing is done five times for each setup to obtain an average performance. If at

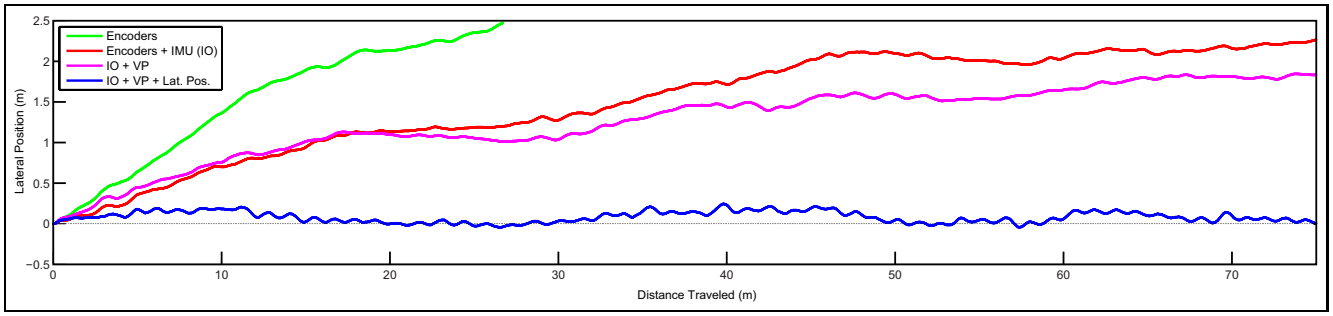

Fig. 9. Example trajectory of autonomous navigation using various components of the presented visual road recognition system. Here Beobot 2.0 has to traverse through a 75m route on a 6.09m wide road, running from left to right. The encoder only odometry trajectory is denoted in green, while the encoder and IMU odometry is red, IMU and VP magenta, and the full system with IMU, VP and lateral position estimation blue. The dashed line is the center of the road and the figure only shows one side of the road

TABLE I
PARTS CONTRIBUTION EXPERIMENT

	Length	Avg	Max.	Stdev.
Encoders	34.68m	1.47m	2.70m	0.73m
Encoders + IMU (IO)	75.00m	1.30m	2.26m	0.52m
IO + VP	75.00m	1.31m	2.77m	0.93m
IO + VP + Lat. Pos.	75.00m	0.20m	0.70m	0.14m

TABLE II
EXPERIMENT SITES INFORMATION

	BKS	FDF	HWY	LEA
Route Length	100.0m	75.0m	70.0m	50.0m
Road Width	7.92m	6.09m	5.20m	8.00m

any time the robot is in danger of falling off the road, we stop it and only record the results up to that point.

We test the system on site FDF, where the 75 meter long road is uniform gray in color with sizable red-colored shoulders, seen in the first set of images in figure 7. The full road width, including the shoulder, is 6.09m. Also, in this experiment as well as the next, we do not turn on the LRF for obstacle avoidance so that the robot can keep moving straight. However, the people around the robot can still occlude its view. Table I reports the experimental results: average length traveled, average deviation from the road center, as well as the deviation's maximum value and standard deviation.

As we can see, encoders by themselves perform the worst, only able to complete an average of 34.68 m of the total road length. To make a robot run straight through a road using blind encoder odometry, the floor has to be perfectly level, which is almost never the case. As for the IMU and encoder infused navigation, the robot has to be aimed almost perfectly to stay centered. Otherwise, as we observe the typical trajectory in figure 9, the robot deviates from the center of the road on a straight line. In addition, there are some places where ambient IMU interference is quite strong, particularly indoors, near buildings, or if there are structures underneath the road.

The VP-only navigation system eventually also gets off-course because our VP estimation is not accurate to the pixel value. Thus, although it allows the robot to maintain a fairly specific heading, there is usually a slight bias that slowly drifts the robot to one side of the road. The full system, with a far better 0.20m average error, is able to stay in the middle of the road because the lateral deviation estimator locks the robot on the spacings of the lines in front of it, rather than solely on a far-away point such as the VP.

C. Environment and Robot Condition Testing

We systematically test the autonomous navigation system on different days, at different times of the day, and in four selected outdoor sites on the USC campus: Bookstore (BKS), Frederick D. Fagg Park (FDF), Hellman Way (HWY), and Leavey Library (LEA). These sites encompass different road appearance characteristics that are explained in figure 7. In addition, their respective route length and road width are available in table II.

We first run the robot in an ideal condition: no crowd, no shadows, and ideal speed 0.5m/s (equivalent to slow walking). For an even lighting with no shadow, we run the robot between 5 and 7pm where the sun is about to set, whereas, to obtain shadow patterns, the robot is run in the afternoon. For a faster speed, the robot is run at 1.0m/s, which is very close to regular walking speed. We display the average deviation from the road center in a series of bar graphs in figure 10, with the different conditions labeled appropriately.

The results between the sites for the ideal testing cases are similar, between 0.12m and 0.29m, with an average of 0.21m lateral deviation error. The HWY site is slightly more difficult because the roads are curved. However, we find that, even on a curved road, the system can still utilize the straight portions of the road boundaries to direct itself.

In addition, the system appears to be able to cope with the crowded and shadow conditions, on average, measuring 0.37m and 0.32m, respectively. There are different types of shadows, which are displayed in figure 7. One that is without sharp straight edges such as the ones underneath bushes (observe the second example of FDF site), while the other has sharp edges, usually shadows of buildings or tree trunks (first BKS and forth LEA example). The latter has

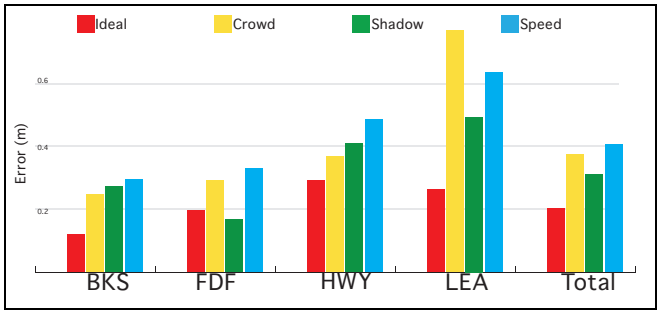

Fig. 10. Comparison of the navigation system performance under various conditions. We measure the average deviation (in m) of the robot from the road center and display the results of the different sites separately. Within the sites we separate the conditions to ideal, shadow, crowded and speed.

the potential to affect the road recognition system if it is not pointing to the VP. However, these shadows would not interfere too severely if the true road lines are also detected.

As for the crowd comparison, the results are close to the ideal condition, except for the LEA site. In the other sites, such as the BKS (second BKS example in figure 7) there are many road cues spread throughout the image. This makes it almost impossible for even the densest crowd to cover all the lines. However, in LEA there are only two wide boundary lines, spaced 8m apart. If one of them is crowded such as the third LEA example image, the robot has to swerve toward the visible boundary.

The difference in speed also shows a notable difference in error, 0.20m farther than in the ideal condition. In the higher speed of 1m/s, the robot visibly oscillates and strays away from the center of the road, accumulating a large amount of error. This is not what is observed with the regular speed of 0.5m/s, where the robot smoothly corrects its heading. Despite the jagged trajectory with the higher speed, the robot still maintains its heading all the way to the end of the route. Because of this encouraging result, we believe that the visual road recognition system can still perform just as well at higher speeds, with improved implementation of forward projection that better incorporates instantaneous robot odometry.

IV. DISCUSSION AND CONCLUSIONS

In this paper we present a novel approach to monocular vision road recognition by using road line segments representation which do not assume rigid road shape such as a triangular template. This flexibility is highlighted in the testing section where the system is able to cope various challenges such as roads with complex shapes, surface textures, or markings, shadows, and crowded situations.

In addition, when we then fuse the visual road recognition system with odometry information within an autonomous navigation framework. The full system is able to autonomously drive the robot over more than 5 km of different routes on a busy college campus. We believe that our study is to date the largest-scale successful demonstration of an autonomous road finding algorithm in a complex campus environment.

V. ACKNOWLEDGMENTS

The authors gratefully acknowledge the support by the National Science Foundation (grant number BCS-0827764), the Army Research Office (W911NF-11-1-0046), and U.S. Army (W81XWH-10-2-0076). The authors affirm that the views expressed herein are solely their own, and do not represent the views of the United States government or any agency thereof.

REFERENCES

[1] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige, "The office marathon: Robust navigation in an indoor office environment," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, May 2010, pp. 300 – 307.

[2] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, G. H. M. Halpenny, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerka, E. Jensen, P. Alessandrini, G. Bradski, B. Davie, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney., "Stanley, the robot that won the darpa grand challenge," *Journal Field Robotics*, vol. 23, no. 9, pp. 661–692, 2005.

[3] S. Thrun, "Google's driverless car," 2011, "Talk was viewed at http://www.ted.com/talks/sebastian_thrun_google_s_driverless_car.html on September 1, 2012".

[4] "Static calibration and analysis of the velodyne hdl-64e s2 for high accuracy mobile scanning," *Remote Sensing*, vol. 2, no. 6, pp. 1610–1624, 2010.

[5] Z. Chen and S. Birhfield, "Quantitative vision-based mobile robot navigation," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, May 2006, pp. 2686 – 2692.

[6] C.-K. Chang, C. Siagian, and L. Itti, "Mobile robot vision navigation & localization using gist and saliency," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2010.

[7] A. Cherubini, F. Spindler, and F. Chaumette, "A new tentacles-based technique for avoiding obstacles during visual navigation," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[8] C. Rasmussen, Y. Lu, and M. Kocamaz, "Appearance contrast for fast, robust trail-following," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2009, pp. 3505 – 3512.

[9] T. Kuhnl, F. Kummert, and J. Fritsch, "Monocular road segmentation using slow feature analysis," in *IEEE Intelligent Vehicles Symposium (IV)*, june 2011, pp. 800 – 806.

[10] P. Santana, N. Alves, L. Correia, and J. Barata, "Swarm-based visual saliency for trail detection," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, oct. 2010, pp. 759 –765.

[11] H. Kong, J.-Y. Audibert, and J. Ponce, "General road detection from a single image," *IEEE Transactions on Image Processing*, vol. 19, no. 8, pp. 2211 – 2220, August 2010.

[12] O. Miksik, "Rapid vanishing point estimation for general road detection," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, May 2012.

[13] C.-K. Chang, C. Siagian, and L. Itti, "Mobile robot monocular vision navigation based on road region and boundary estimation," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2012, bb, pp. 1043–1050.

[14] C. Siagian, C.-K. Chang, R. Voorhies, and L. Itti, "Beobot 2.0: Cluster architecture for mobile robotics," *Journal of Field Robotics*, vol. 28, no. 2, pp. 278–302, March/April 2011.

[15] G. Bradski, "Open source computer vision library," 2001. [Online]. Available: http://opencv.willowgarage.com

[16] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 1993, pp. 802–807.

[17] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23 – 33, 1997.