Autonomous Mobile Robot Localization and Navigation Using Hierarchical Map Representation Primarily Guided by Vision

Christian Siagian *
*
Department of Computer Science
University of Southern California
Los Angeles, CA 90089
siagian@usc.edu

Chin Kai Chang *
†
Department of Computer Science
University of Southern California
Los Angeles, CA 90089
chinkaic@usc.edu

Laurent Itti
Department of Computer Science
University of Southern California
Los Angeles, CA 90089
itti@pollux.usc.edu

Abstract

While impressive recent progress has been achieved with autonomous vehicles both indoors and on streets, autonomous localization and navigation in less constrained and more dynamic environments, such as outdoor pedestrian and bicycle-friendly sites, remains a challenging problem. We describe a new approach that utilizes several visual perception modules — place recognition, landmark recognition, and road lane detection — supplemented by proximity cues from a planar Laser Range Finder for obstacle avoidance. At the core of our system is a new hybrid topological/grid-occupancy map which integrates the outputs from all perceptual modules, despite different latencies and timescales. Our approach allows for real-time performance through a combination of fast but shallow processing modules that update the map's state while slower but more discriminating modules are still computing. We validated our system using a ground vehicle that autonomously traversed several times three outdoor routes, each 400m or longer, in a university campus. The routes featured different road types, environmental hazards, moving pedestrians, and service vehicles. In total, the robot logged over 10km of successful recorded experiments, driving within a median of 1.37m laterally of the center of the road, and localizing within 0.97m (median) longitudinally of its true location along the route.

1 Introduction

The ability to localize or recognize one's location, and to navigate or move about one's environment, are critical building blocks in creating truly autonomous robots. In the past decade, there has been significant

*website: ilab.usc.edu/siagian; *equal authorship
†website: ilab.usc.edu/ kai; * equal authorship

progress in this aspect of autonomy, particularly indoors (Thrun et al., 1999; Fox et al., 1999; Marder-Eppstein et al., 2010) and on streets and highways for autonomous cars (Montemerlo et al., 2008; Thrun, 2011). However, despite these successes, ample opportunity remains for substantial progress in less constrained pedestrian environments, such as a university campus or an outdoor shopping center. This type of settings encompasses a major portion of the working environments for human-sized service robots whose tasks are exemplified in figure 1. One major hurdle is that sensors and techniques which are reliable in other areas are not as effective here.

Figure 1: Examples of tasks for service robots illustrated from left to right: patrolling alongside law enforcement personnel, searching for target objects, providing information to visitors, and assessing the situations and conditions of the robot's area of responsibility.

For one, current indoor localization (Thrun et al., 2000; Fox et al., 1999) and navigation (Minguez and Montano, 2004; Marder-Eppstein et al., 2010) systems rely primarily on planar proximity sensors such as the Laser Range Finder (LRF). These proximity sensors exploit the confined nature of indoor space, e.g., narrow hallways, to create a unique location signature for localization, as well as to direct the robot heading. In the open-space outdoors, however, these sensors are less effective because, first, surfaces detected by laser may not be as structurally simple as indoors (e.g., trees, shrubs), and, second, often the nearest solid surfaces may be out of the sensor's range. Thus, it is desirable to find other ways to robustly sense the environment. Two primary sub-problems in navigation are to estimate the road heading, so that the robot stays on its route, and to avoid static and dynamic obstacles. Because obstacle avoidance is readily and identically solved by proximity sensors both indoors and outdoors, road recognition becomes the more pressing navigational concern.

Another commonly-used group of devices are depth sensors, which produce dense three dimensional proximity information by exploiting a number of physical properties. However, many of them, e.g., the Swiss-ranger or Kinect, do not work robustly outdoors, especially in the presence of sunlight. Even for the ones that do, such as stereo cameras, the system still has to perform the difficult task of extracting the shape and appearance of the road, possibly without the help of surrounding walls.

The problem of road recognition can now be robustly solved on urban streets and highways because of distinguishing cues such as clear lane markings and bankings. One key breakthrough of the DARPA Grand Challenge (Montemerlo et al., 2008) is the introduction of a sensor called the Velodyne (Glennie and Lichti, 2010). It provides combined proximity and appearance information, measured densely in all 360°, with 120 meter range. The sensor has played a major role in enabling many teams to finish the race. Furthermore, it has also allowed the Google Car (Thrun, 2011) to travel more than 300,000 miles for many months in the San Francisco area. However, because of its cost, the Velodyne has not been widely utilized. In addition, pedestrian roads, which are less regulated and whose delineations are more subtle and varied, pose a different challenge, one that still has not been generally solved even with a Velodyne.

Another aspect of autonomous cars that does not transfer to human-sized service robots is how localization is performed. Unlike faster moving cars driving on a highway, which can afford to make Global Positioning System (GPS) location queries that are far apart, service robots operate on a smaller environment scale and move at the speed of humans. A GPS reading, with its relatively coarse resolution, would not be able to localize a robot accurately enough to turn correctly at a typical campus road intersection. Furthermore, many pedestrian roads have tall trees or buildings alongside them, which denies satellite visibility that the

GPS relies upon.

Given these factors, vision, the primary sense of humans, is an ideal perceptual modality to localize and navigate in most outdoor places that humans can travel to. In addition, it is also an attractive alternative because of the affordability and availability of cameras. However, one drawback of these versatile devices is the associated processing complexity, further exacerbated by the real-time requirements that robotic systems demand.

In this paper, we present a fully autonomous localization and navigation system for a human-sized service robot operating in unconstrained pedestrian environments, primarily using monocular vision and complemented by LRF-based obstacle avoidance. Our main contributions are four-fold:

- Using vision algorithms (visual attention, landmark recognition, gist classification, localization, road finding) to compute localization and navigation cues that are robust in busy outdoor pedestrian areas;
- Developing a framework for the integration of these cues using a proposed hierarchical hybrid topological/grid occupancy map representation.
- Exploiting the hierarchical map, along with applying techniques such as forward projection and tracking, to resolve differences in algorithmic complexity, latency, and throughput, to create an overall real-time robotics system.
- Testing over more than 10km total traveled distance, with routes 400m or longer, to demonstrate that the approach indeed is viable and robust even in crowded environments like a university campus. As far as we know, this is the most extensive evaluation of a service robot localization and navigation system to date.

The system is described in section 3 and tested in section 4. We then discuss our findings in section 5. First, we describe related vision localization and navigation research in the next section.

2 Related works

We start with similar full systems that perform both navigation and localization in section 2.1. In addition, because many research efforts have focused on specific sub-components of our system, we also separately survey vision localization (section 2.2), road recognition (section 2.3), and path planning (section 2.4). A large portion of these references directly motivate the implementations of our system components, techniques, and overall architecture. Furthermore, we also cite references that contrast our approaches to complete the background literature. These cited works shown that our decisions are well founded by the respective communities.

2.1 Indoor and Outdoor Mobile Robot Localization and Navigation

In recent years, a few systems have ventured out to pedestrian outdoor environments. Systems such as (Trulls et al., 2011) and (Montella et al., 2012) do so by solving localization, road heading estimation, and obstacle avoidance, all by using Laser Range Finders (LRF) only. The key is being able to take advantage of various detectable and distinguishable environment features within the LRF range. In their testing, these static cues are obtained from nearby building surfaces, walls, poles, and trees. In the absence of such features, the technique would have to find other cues to utilize.

Another approach to navigation and localization makes use of visual landmarks through the teach-and-replay paradigm (Chen and Birchfield, 2006; Chang et al., 2010; Furgale and Barfoot, 2010). The robot is first

manually steered through a specific route in the teaching stage, and is then asked to execute the same exact route during autonomous operation. Once a few landmarks are detected, the system can synchronize what it currently sees with the training frames. Localization, like other visual recognition-based systems, is straightforward because training frames are associated with a specific location. Navigation, on the other hand, is achieved by moving the robot to align the image coordinates of the seen landmarks to the appropriate locations in the training images.

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 dynamic obstacles such as pedestrians. One important concern is on how to get back to the set route when the robot has to deviate from the set path momentarily, e.g., to avoid unexpected obstacles. Indeed, if such deviation was not encountered during training, the robot will have no memory of what it sees after deviating. Recent improvements by (Cherubini et al., 2012) have shown promising results in overcoming such difficulty.

Another issue arises when the system drifts and becomes out of synchronization with the reference training image sets. In that case, the system has to reset itself. This has to be done while the robot is stationary because the landmarks are responsible for both localization and navigation. Our system separates these two components. When our localization system is temporarily confused about where it is, the robot can still move forward using road recognition, allowing the localization system to possibly recover later down the route.

2.2 Vision Localization

The key to a successful vision localization system is to be able to match observed visual landmarks despite outside interference or visual variability. Since the recent introduction of visual keypoints, such as SIFT (Lowe, 2004) and SURF (Bay et al., 2006), as well as subsequent techniques such as the visual bag-of-words (Lazebnik et al., 2006), systems can now robustly perform this critical matching step in many more environments, including outdoors.

There are now many systems that are capable of accurate coordinate-level localization (Segvic et al., 2009; Se et al., 2005; Murillo et al., 2007; Royer et al., 2007), as well as Simultaneous Localization And Mapping (SLAM) (Eade and Drummond, 2008; Angeli et al., 2008; Strasdat et al., 2010). In fact, many systems are capable of localizing in large scale areas the size of a few city blocks (Cummins and Newman, 2008; Schindler et al., 2007), as well as across many seasons (Valgren and Lilienthal, 2008). However, many of these systems have thus far only been tested off-line as the computational complexity can be prohibitive.

An interesting recent effort focuses on a lower-resolution localization task called place recognition (Ulrich and Nourbakhsh, 2000; Torralba et al., 2003; Pronobis et al., 2006; Fazl-Ersi and Tsotsos, 2012). Here, the problem is framed as a classification task, to distinguish places in the environment, not the exact coordinate. Because of the coarser resolution, these systems easily run in real time.

A practical way to exploit the speed of place recognition is by combining it with the accuracy of metric localization systems. Many systems (Zhang and Kosecka, 2005; Wang et al., 2006; Weiss et al., 2007; Siagian and Itti, 2009) do so through a coarse-to-fine hierarchical approach, where place recognition primes possible areas in the map before refining the location hypothesis to a coordinate level.

For completeness, it should be noted that many systems combine vision-based localization with other modalities such as proximity (Pronobis et al., 2010; Newman et al., 2009; Ouerhani et al., 2005), GPS (Agrawal and Konolige, 2006; Vargas et al., 2007), or visual odometry (Nistér et al., 2006; Tardif et al., 2008). Each modality provides something that complements vision localization. For example, even though GPS is coarser, has a lower update frequency, and is less reliable in some places, it provides clean global absolute location information that can be used to limit the matching process to a smaller geographic area. Note that we make a distinction between vision localization and visual odometery. The former is a technique that tries to

visually recognize one's global location, while the latter only visually estimates ones motion. However, if an initial global location is available, visual odometry can be critical in maintaining localization.

2.3 Road Lane Recognition

A critical task in navigating in pedestrian environments is to recognize and stay on the road. It is important to distinguish road lane navigation from systems that simply drive through any flat terrain. The former specifically enforces proper and socially acceptable ways to travel in pedestrian environments, e.g., avoiding excessive zigzags. In addition, often it is important for the robot to travel in the middle of the road, because it provides an optimal vantage point to observe signs or landmarks. Also, by trying to recognize roads in general, instead of performing teach-and-replay, the system is not limited to a set path that has previously been manually driven. Instead, the system has the freedom to move about its surroundings and explore unmapped areas.

Aside from using vision, systems have been proposed that use proximity cues from LRF (Hu et al., 2009; Wurm et al., 2009), stereo cameras (Hrabar and Sukhatme, 2009), and Kinect (Cunha et al., 2011). They have shown a measure of success in restricted conditions, without the presence of sunlight or where the surfaces characteristics of the road and its flanking areas are sufficiently dissimilar. Monocular visual cues, on the other hand, readily encode road information in most environments.

One visual approach relies on modeling the road appearance using color histograms (Rasmussen et al., 2009; Kuhnl et al., 2011), which assumes that the road is contiguous, uniform, and visually different than the flanking areas (Santana et al., 2010). In addition, to ease the recognition process, the technique usually simplifies the road shape, as viewed in the image, as a triangle. These assumptions are oftentimes violated in cases where there are complex markings, shadows, or pedestrians on the road. Furthermore, they also do not hold when the road appearance is similar to the flanking areas.

Another way to visually recognize the road direction is by detecting the vanishing point (VP). Most systems (Kong et al., 2010; Moghamadam et al., 2012; Miksik, 2012) use the consensus direction of local textures and 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, many systems also attach a triangle to the VP to idealize the road shape. Our road recognition module (Siagian et al., 2013a) improves upon these limitations by detecting the VP using more robust contour segments and not assuming the triangular shape.

2.4 Path Planning

Generally, a path planning system needs a mapping of the surroundings to generate motion commands. However, a few systems do not explicitly model the nearby obstacles, but simply detect and avoid potentially hazardous conditions, for example, using optical flow (Coombs et al., 1995; Santos-Victor et al., 1993) or other visual features (Michels et al., 2005; i Badia et al., 2007; Nakamura and Asada, 1995). A standard representation allows a system to utilize multiple perceptual modalities and naturally fuse them together.

The mapping is usually in form of a grid occupancy map, whether it be the traditional ground-view two dimensional (Moravec and Elfes, 1985) map, or more recent three dimensional (Marder-Eppstein et al., 2010) version. These maps are primarily filled by planar proximity sensors such as the LRF, which can also be swept to create a three dimensional perception. Another way to create a three dimensional map is by using the aforementioned Velodyne (Glennie and Lichti, 2010). From our experience, because of the openness of the outdoors, a well placed planar LRF is sufficient to avoid most obstacles. For more aggressive maneuvers through narrow openings such as under a table, a three dimensional representation would be needed.

Given an occupancy map, there are two approaches to compute a motor command: directional space control and velocity space control. In the former, the algorithm selects a command that moves the robot to the

best neighbor location in terms of advancing to the goal, while avoiding obstacles. A class of this is called the Vector Field Histogram (Borenstein and Koren, 1991; Ulrich and Borenstein, 1998), which combines an attractive force from the goal with repelling forces from nearby obstacles. However, this technique, as well as related approaches such as the Nearness-Diagram algorithm (Minguez and Montano, 2004), suffers from local minima problems because they do not explicitly search for complete paths that eventually arrive at the goal. On the other hand, shortest path algorithms such as A* (Wilson, 1982) and its derivatives, e.g., D*-lite (Koenig and Likhachev, 2002), ensure path completion to the goal.

One difficulty of using grid map is determining the optimal grid size. An overly large size creates a coarse map that excludes some needed details, for example, in cluttered settings. A grid size that is too small is also not ideal as it renders the system inefficient because the exponential increase of the search space. One way to combat such problem is through the use of non-uniform grid size, such as an octree (Wurm et al., 2010).

Even with an ideal grid size, because of the map discretization, the resulting path may be too rigid and does not maximally avoid obstacles. This is because the path consists of straight segments that connect the centers of the visited grid cells. Techniques such as the elastic band approach (Quinlan and Khatib, 1993) modifies and smooths out the resulting rigid path. Another approach by (Yang and Lavalle, 2004) achieves the same objective by directly sampling the free space in the map without gridding it altogether.

In order for the robot to behave decisively, it is important to create a path that is consistent over time. An extension of the shortest path algorithms called the Incremental Heuristic Search (Koenig et al., 2004) encourages such requirement as well as speeds up the process by incorporating paths from previous time steps as a prior to guide the subsequent search.

A fundamental problem with directional space control is that it does not deal with kinematic and dynamic constraints, for example, whether a robot is holonomic. Velocity space control techniques (Fox et al., 1997; Simmons, 1996; Fiorini and Shiller, 1998) address them directly by modeling factors such as robot geometry, actuator configuration, and the robot's current and achievable velocities.

The technique requires a good motion model to compute precise trajectory estimation. This is in contrast with directional space control which does not take into account how fast the robot will get to a certain spot, as long as it ends up there. In addition, constructing a full path to the goal is time consuming considering the number of options in each step.

One way to alleviate this is using a rapidly exploring random tree (RRT) (Kuffner and LaValle, 2000), which is designed to efficiently search through high-dimensional space. Another way is to compute a full path in the directional space, then refine an immediate portion in the velocity space (Marder-Eppstein et al., 2010). The combination allows a system to gain the benefit of both techniques. Our presented system is an example of such an algorithm.

3 Design and Implementation

The flow of our overall system is illustrated in figure 2. The left side of the image depicts our mobile platform, Beobot 2.0, with the utilized sensors – a forward-pointing camera, a Laser Range Finder (LRF), wheel encoders, an IMU – as well as motors to move the robot. Our IMU (MicroStrain, Inc., 2009) is also equipped with a compass that makes magnetic measurements. It is placed high above the computers and motors to minimize magnetic distortion induced by the robot. Also note that ferrous obstacles in the environment may affect the IMU.

The two major sub-systems, localization and navigation, are displayed in the middle of the figure. The localization system (the top of the column) is described in section 3.1, while the navigation system (bottom)

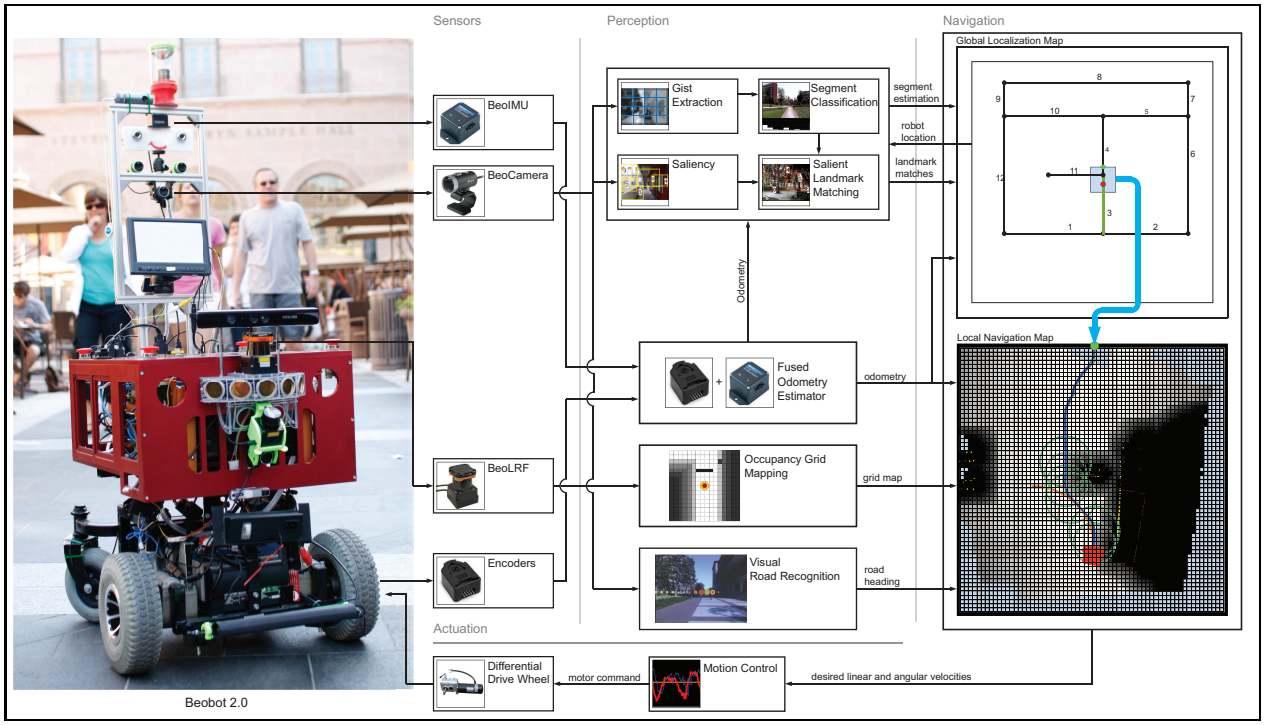

Figure 2: Diagram for the overall localization and navigation system. We run our system on a wheelchair-based robot called the Beobot 2.0. As shown on the sensor column, the system utilizes an IMU, camera, Laser Range Finder (LRF), and wheel encoders. We use the camera to perform vision localization by integrating results from salient landmark recognition and place recognition. For navigation we use visual road recognition for estimating the road heading, and LRF-based grid-occupancy mapping for obstacle avoidance. The system utilizes a topological map for global location representation and a grid-occupancy map for local surrounding mapping. The grid map, which is represented by the blue rectangle on the topological map, is connected with the topological map by the intermediate goal (green dot).

is in section 3.2. Because earlier versions of these sub-systems have been described previously (Siagian and Itti, 2009; Siagian et al., 2013a), here we provide summary descriptions of their technical details, instead focusing on information flow, timing, and interactions between sub-systems, which are key to designing a complete working mobile robot system. Note that both sub-systems utilize the fused odometry.

The rightmost column in the figure is the hierarchical spatial representation of the robot's environment. The first level is a global topological map that compactly represents all the needed metric information for global path planning. The next one is a local grid-occupancy map, which details the robot surroundings. Mechanically, the local map is connected to the global map through an intermediate goal location that is registered in both maps (observe figure 2). The task of the navigation system is simply to safely reach the intermediate goal. The local map, which is robot-centric, is then advanced along the current graph edge in the global map, to always keep the robot at a fixed location in the local map.

For each sub-system, we describe not only the algorithm and its pertinent components, but also the timescale of the implementation. By analyzing the timescale, we demonstrate the advantages afforded by the hierarchical map construction, as well as the mechanisms that are utilized to overcome long latencies of some components. We then continue to the overall system and examine how it accommodates the different component latencies to ensure a timely system execution.

We start the detailed description with the vision localization system.

3.1 Localization System

We summarize the vision localization system, which is inspired by previously published work (Siagian and Itti, 2009). We focus on the mechanisms that allow it to run in real time: multi-level hierarchical and multi-perception localization. As illustrated in figure 3, the system is divided into three stages. The first stage takes an input image and extracts low-level Visual Cortex-like features, consisting of center-surround color, intensity, and orientation channels. These features are used by two parallel processes to extract the "gist" and salient points of the scene, two extensively studied human visual capabilities. We define the gist features as a low-dimensional vector (compared to raw image pixel array) that represents a scene and can be acquired over very short time frames (Siagian and Itti, 2007). Saliency, on the other hand, is computed as a measure of interest at every image pixel, to efficiently direct the time-consuming landmark matching process towards the most likely candidate locations in the image. The gist features and salient regions, along with fused odometry, are input to the back-end Monte-Carlo localization (Thrun et al., 2000; Fox et al., 1999) algorithm to estimate the robot's position.

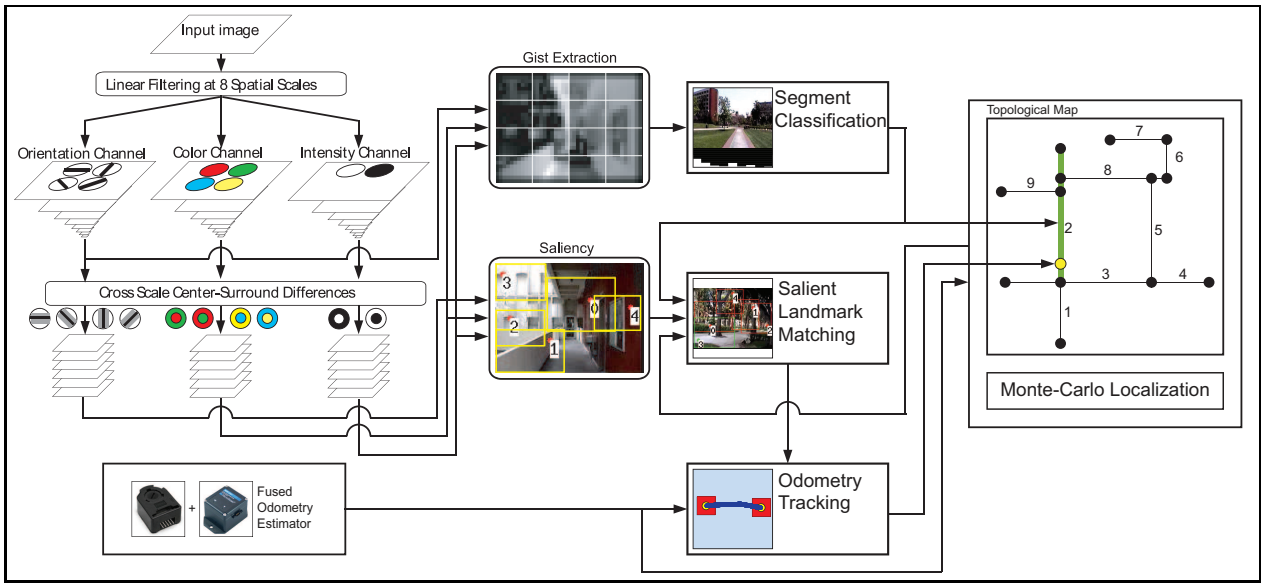

Figure 3: Diagram of the vision localization system. The system takes an input image and extracts low-level features in the color, intensity, and orientation domains. They are further processed to produce gist features and salient regions. The localization system then uses these features for segment classification and salient landmark matching (respectively), along with fused odometry, to estimate the robot's location using a Monte-Carlo framework.

The topological map, which currently is manually constructed, is represented by a graph $G = \{V, E\}$. The nodes $v \in V$ are augmented with metric information specifying their coordinate locations. Accordingly, the edge cost W_e for each edge $e \in E$ is set to the distance between its respective end-nodes. The system also defines segments $s \subseteq E$, each an ordered list of edges, with one edge connected to the next to form a continuous path. This grouping is motivated by the fact that the views/layouts in a segment are similar and can be classified using gist features. An example of a segment is the highlighted three-edge segment 2 (thicker and in green) in the map in figure 3.

The term salient region refers to a conspicuous area in an input image depicting an easily detected part of the environment. An ideal salient region is one that is persistently observed from different points of view and at different times of the day. In addition, the set of salient regions that portray the same point of interest is considered jointly, and that set is called a landmark. To recognize a landmark and to associate it with a location, the system employs a training regimen by manually driving the robot through all the paths in the environment. The produced database is quite large, about one landmark per meter of path. As a

result, unlike the instantaneous segment classification, the landmark recognition process can take seconds. However, as opposed to just localizing to the segments, salient landmarks produce coordinate locations.

The localization system takes advantage of the contrasting segment classification and landmark matching by coarsely but quickly estimating the robot's segment location in every video frame. It then refines the estimation to a more accurate metric location when the salient region matching results are available. In addition, the system also speeds up the database matching process by using contextual information such as current location belief and segment classification output. It orders the database items to be compared from the most likely to be positively matched to the least. This helps because the search ends, for efficiency reasons, after the first match is found. Also, to keep the database match results relevant despite the processing latency, the system stores the accumulated odometry during the landmark matching process. When the results are ready, the landmark recognition module adds the longitudinal movement offset before sending the resulting location to the back-end Monte-Carlo localization.

Within the Monte-Carlo framework, we formulate the location state S_t as a set of weighted particles $S_t = \{x_{t,i},\ w_{t,i}\},\ i = 1\ ...\ N$ at time t, with N being the number of particles. Each particle or possible robot location $x_{t,i} = \{k_{t,i}, l_{t,i}\}$, with k being the segment number and l the length traveled along the segment edges normalized by the segment length (from 0.0 at the start of the segment to 1.0 at the end). Because the localization formulation is over the graph edges but with exact metric location specification, the search space is effectively continuous one dimensional. That is, even though the locus of the considered locations describe a two dimensional geographical space, it is constrained to within the path edges. Thus, a small and computationally less demanding $N = 100$ suffices, even in kidnapped robot instances. Also, with this formulation, we specify a path as the exact location of the end points and a list of edges that connects them.

The system estimates the location belief $Bel(S_t)$ by recursively updating the posterior $p(S_t|z^t, u^t)$ using segment estimation $z_t^{'}$, landmark matching $z_t^{''}$, and motion measurements u_t. In addition, the system invokes importance resampling whenever more than 75% of the particles collapse to within the robot width of 60cm. It chooses 95 percent of the total N particles from the existing population by taking into account their weights $w_{t,i}$, while randomly generates the remaining 5 percent. The latter is done by randomly choosing an integer among the segment numbers for $k_{t,i}$ and decimal value between 0.0 and 1.0 for $l_{t,i}$.

To robustly measure robot motion, we implement a Kalman filter (King, 2008), which fuses the wheel encoders and IMU. The filter's state space is the current absolute heading θ_t and distance traveled d_t. Using these estimated values, we compute the longitudinal distance traveled $u_t = d_t * cos(\theta_t - \theta_{\psi_t})$, with θ_{ψ_t} being the absolute road direction estimate, discussed in section 3.2.1 below.

The system then computes the motion model $p(S_t|S_{t-1}, u_t)$ by advancing the particles by u_t and adding noise to account for uncertainties such as wheel slippage that can produce large odometry errors. This is done by drawing a random location from a Gaussian probability density $p(x_{t,i}|u_t, x_{t-1,i})$, where the mean is the previous location $x_{t-1,i}$ plus the odometry u_t, and the standard deviation is 3cm (about 1/6th of a typical single step). If in the process a particle moves past either end of a segment, we change the segment number appropriately and normalize the excess distance from the end of the original segment. If the original segment ends at an intersection with multiple continuing segments, we randomly select one. If no other segment extends the path, we leave the particle at the end.

For the visual observations, the segment estimator (Siagian and Itti, 2007) uses the gist features to classify the input scene, and outputs the likelihood $z_t^{'} = \{\ \phi_{t,j}\ \},\ j = 1\ ...\ N_{segment}$ with $\phi_{t,j}$ being the likelihood value for segment j at time t. The system then updates the weights $w_{t,i}$ of each particle $x_{t,i}$ to:

$$p(z_t^{'}|x_{t,i}) = \frac{\phi_{t,k_{t,i}}}{\sum_{j=1}^{N_{segment}} \phi_{t,j}} * \phi_{t,k_{t,i}} \tag{1}$$

Here, the likelihood that particle $x_{t,i}$ observes $z_t^{'}$ is proportional to its segment location estimation value $\phi_{t,k_{t,i}}$ over the total sum (first term) times the value itself (second term). The first term accounts for the segment relative strength, while the second preserves the ratio with respect to the absolute 1.0 maximum value.

The landmark matching process, which is illustrated in figure 4, compares incoming salient regions to the landmark database regions using two sets of signatures: SIFT keypoints (Lowe, 2004) within the region bounding box, and salient feature vector. The former denotes the shape of the region, while the latter quantifies its saliency characteristics. We employ a straight-forward SIFT-based recognition system (Lowe, 2004) using the suggested parameters, while the salient feature vector matching is described here.

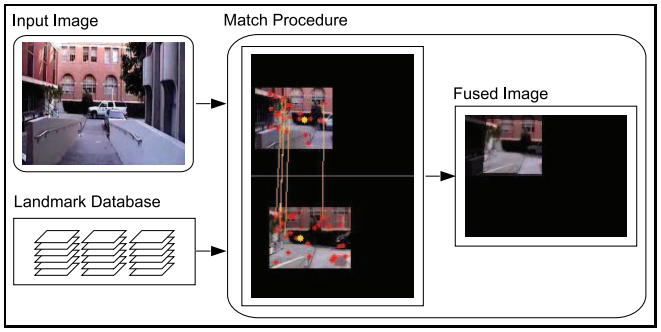

Figure 4: Matching process of two salient regions using SIFT keypoints (drawn as red disks) and salient feature vector, which is a set of feature map values taken at the salient point location (yellow disk). The lines indicate the keypoint correspondences. The fused image is added to show alignment.

A salient feature vector is a set of feature map values (Itti et al., 1998; Itti, 2000) from the color, intensity, and orientation channels. These values are taken from a 5-by-5 window centered at the salient point location of a region. In total, there are 1050 features: 7 sub-channels times 6 feature maps times 5×5 array. To compare salient feature vectors of two regions ρ_1 and ρ_2, we factor in feature similarity $sfSim$ (equation 2) and salient point location proximity $sfProx$ (equation 3) after aligning the regions (observe the fused image in figure 4). The former is based on Euclidian-distance in feature space:

$$sfSim(\rho_1, \rho_2) \;=\; 1 - \frac{\sqrt{\sum_{i=1}^{N_{sf}} (\rho_{1,i} - \rho_{2,i})^2}}{N_{sf}} \tag{2}$$

with N_{sf} being the feature dimension. For a match to be confirmed, the feature similarity has to be above .75 out of the maximal 1.0. The location proximity $sfProx$, on the other hand, is the Euclidian distance in pixel units (denoted by the function $dist$), normalized by the image diagonal length η:

$$sfProx(\rho_1, \rho_2) \;=\; 1 - \frac{dist(\rho_1, \rho_2)}{\eta} \tag{3}$$

The matching threshold for $sfProx$ is 95% or within 5% of η.

The resulting matches are denoted as $z_t^{''} = \{\; o_{t,j} \;\},\; j = 1 \;...\; M_t$ with $o_{t,j}$ being the j-th of the total M_t matched database regions at time t. The system computes the likelihood of simultaneously observing the matched regions at a given particle location $x_{t,i}$ by updating the weight $w_{t,i} = p(z_t^{''}|x_{t,i})$:

$$p(z_t^{''}|x_{t,i}) = \prod_{j=1}^{M_t} p(o_{t,j}|x_{t,i}) \tag{4}$$

Because each salient region match observation is independent, the system can multiply each likelihood to calculate the joint probability. Each $p(o_{t,j}|x_{t,i})$ is modeled by a Gaussian where the likelihood value is set to the probability of drawing a length longer than the distance between the particle and the location where the matched database region is acquired. The standard deviation of the Gaussian is set to 5% of the map diagonal to reflect increased level of uncertainty for larger environments.

The system then estimates the robot current location using the highest weighted particle.

We illustrate the temporal integration of the localization components with a timeline in figure 5. The localization system, along with other sub-systems, uses a subscribe/publish architecture, where each component is an independent process. The outputting localization module, which is highlighted at the last row, processes observations from the components as soon as they are available. This allows the system to not be explicitly tied to a rigid time step synchrony. However, because odometry and segment estimation come in regular intervals, the system appears to produce output regularly from the outside perspective.

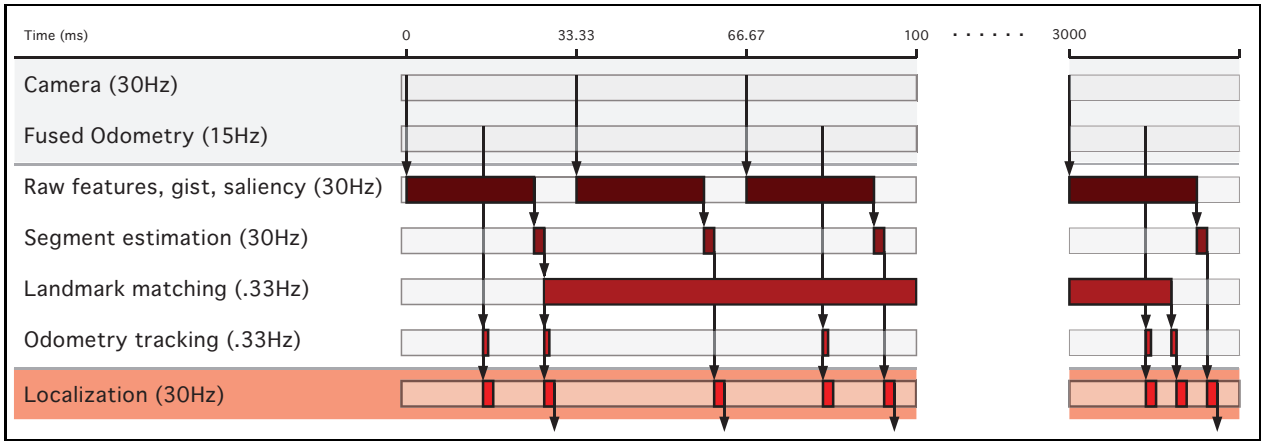

Figure 5: Diagram of the localization sub-system timeline. The figure lists all the localization sub-system components: two sensors, four perceptual processing steps, and a back-end localization estimation. Please see the text for explanation.

The top section of the figure (the first two rows) lists the utilized sensors: a camera, which runs 30Hz, and the 15Hz fused odometry. Note that the fused odometry uses a Kalman Filter to combine information from the 80,000Hz encoders and 50Hz IMU. Even though odometry is reported less frequently, the information latency is closer to that of the encoders. Furthermore, if we consider the robot speed of under 1m/s, a 15Hz or 66.67ms period latency only amounts to 7cm localization error.

Below the sensors are the perceptual components. The raw feature extraction, gist, and saliency computations start after being triggered by the camera output, as indicated by the arrow from the first row. The module, which is implemented in parallel over multiple processing cores, takes about 25ms (noted by the length of the solid maroon block). The process then idles until the next image arrives. For each module, we state its throughput in parentheses, with the output indicated by an arrow that drops down from the right end of the processing block. Note that there are also a number of arrows that seem to do the same from the left side of a block. They are actually pass through arrows from their respective original processes. The first arrow from Fused Odometry, for example, is processed by both Odometry Tracking and Localization. However, it may look as if Localization takes an input from Odometry Tracking.

Segment classification, which takes less than 1ms to compute, passes its result to landmark matching for search priming and localization module for location update. The landmark matching process, on the other hand, takes three seconds on average (.33Hz), despite a multi-threaded landmark search implementation. However, because odometry tracking keeps track of where the original image is being triggered, the long latency is accounted for. The final localization module itself outputs results at 30Hz with the input image

and, subsequently, segment estimation being the trigger.

A critical point that enables localization to effectively operate in real time is that each perceptual input already accounts for its own internal latency. In addition, because the topological map formulation is one dimensional along the graph edges and needs only 100 particles to localize, the localization computation itself takes less than 1ms. Thus the system is not burdened by the frequent observation updates, which allows the location belief to be accurate to the most recent input. Furthermore, the system can still be expanded with new observations and modalities of any timescale as long as their latencies are accounted for. Most importantly, the localization system timely mode of operation allows the full system to integrate it with real-time navigation.

3.2 Navigation System

The components of the navigation system are the visual road recognition (Siagian et al., 2013a) to estimate the road direction (described in section 3.2.1), grid map construction (section 3.2.2), and path planning to generate motor commands (section 3.2.3). In the last part of section, we put the whole system together, integrating both localization and navigation.

The local map represents the robot's surroundings by using a 64 by 64 grid occupancy map (Moravec and Elfes, 1985), with each grid cell spanning 15cm by 15cm spatial area for a 9.6m by 9.6m local map extent. We select 15cm or 1/4 the robot size (our robot is 60cm in length and width) because it is a balance between sufficiently detailed depiction and being manageable for real-time update.

As shown in figure 2, the robot is displayed as a red rectangle, and is placed at the local map horizontal center and three quarters down the vertical length to increase front-of-robot coverage. In addition, there is a layer of grid surrounding the map to represent goal locations that are outside the map. The figure shows the goal to be straight ahead, which, by convention, is the direction of the road. Thus a large portion of motion control is trying to minimize the deviation between the robot and road heading.

Given the robot autonomous speed of 0.66m/s average and 0.89m/s maximum, the 7.2m (0.75 x 64 x 15cm) mapping of the robot front area affords at least 8.1 seconds of forward look-ahead to properly react to potential static obstacles. On the other hand, dynamic obstacles such as people have to be evaluated with respect to the local map update rate of 20Hz. Here, using the average human walking speed of 1.4m/s, the system latency of as much as 100ms translates to 0.14m movement or within one grid size.

3.2.1 Visual Road Recognition

Our visual road recognition system (Siagian et al., 2013a), which is illustrated in figure 6, is a novel vanishing point (VP) -based algorithm that uses long and robust contour segments, instead of the smaller and often noisier edgels. In addition, much like the localization system, it also implements a tracking mechanism to ensure that it runs in real time. The output of the system is the absolute road direction θ_{road}. Note that the road recognition system does not make use of odometry information.

The system takes the input image and performs Canny edge detection to create an edge map. From here it has two ways to recognize the road. One (the top pipeline in the image) is through a full but slower recognition process, which detects segments in the edge map, votes for the most likely VP, and extracts the road lines that are extended from the VP. On the other hand, the bottom pipeline runs in real-time because it only tracks the already available road lines to update the VP location. Because the system has both mechanisms, it balances accuracy and timeliness.

The recognition pipeline uses Hough transform to find straight segments (OpenCV (Bradski, 2001) implementation) in the edgemap. The process then filters out segments that are above the manually calibrated

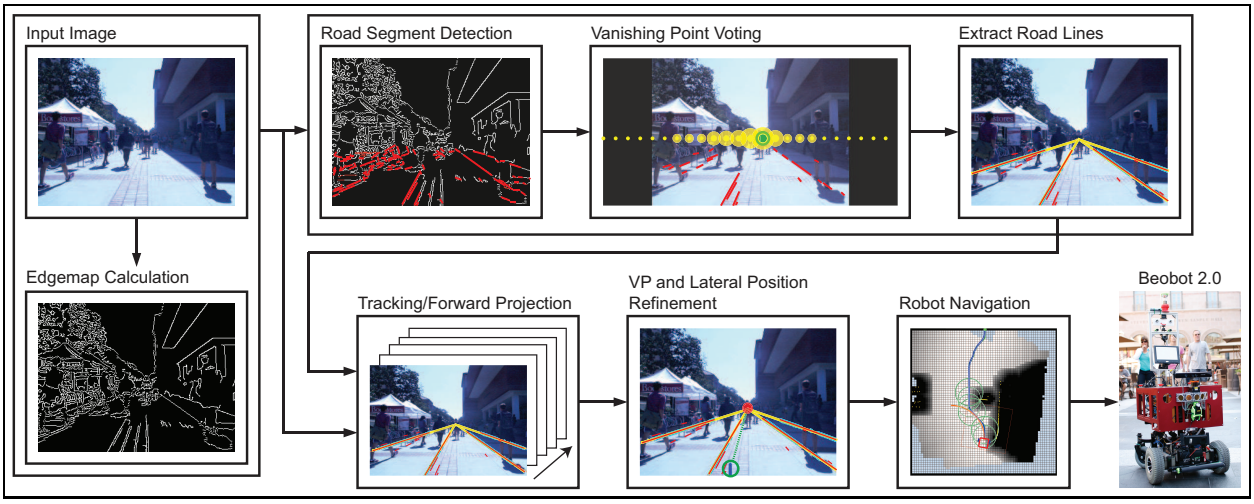

Figure 6: Overall visual road recognition system. The algorithm starts by computing a Canny edgemap from the input image. The system has two ways to estimate the road. One is using the slower full recognition step (the top pipeline), where it performs road segment detection. Here, the segments are used to vote for the vanishing point (VP) as well as to extract road lines. The second (bottom) is by tracking the previously discovered road lines to update the VP location as well as the robot lateral position. The tracker is also used to project forward output road lines from the top pipeline, which may fall behind the camera frame rate. The road recognition system then outputs the VP-based road heading direction as well as lateral position to the navigation system to generate motor commands.

horizon line, near horizontal (usually hover around the horizon line), and near vertical (usually part of close-by objects or buildings). The remaining segments vote for candidate VPs on the horizon line, regularly spaced 1/16th image width apart.

To cast a vote, each segment is extended on a straight-line to intersect the horizon line (observe figure 7). The voting score of segment s for a vanishing point p is the product of segment length $|s|$ and the inverse proportion of the proximity of p to the intersection point of the extended line and the horizon line, denoted by $hintercept(s)$ in the equation below:

$$score(s,p) \;=\; (1.0 - \frac{\overline{|hintercept(s),p|}}{\mu}) * |s| \tag{5}$$

with μ set to 1/8th of image width. μ is also the voting influence limit, with any segment farther not considered. Note that the exact values of the constants in the algorithm do not significantly affect overall system performance.

To improve the robustness of the current VP estimation (denoted as ψ_t), the system multiplies the accumulated scores with the inverse proportion of the proximity to the previous VP location ψ_{t-1}:

$$\psi_t \;=\; \arg\max_p \sum_s score(s,p) * (1.0 - \frac{\overline{|p,\psi_{t-1}|}}{\mu}) \tag{6}$$

We then use the segments that support the winning VP, indicated in red in figure 7, to extract lines for fast road tracking. The system first sorts the supporting segments based on their lengths. It then fits a line equation through the longest segment using least-squares, and iteratively adds the remaining nearby

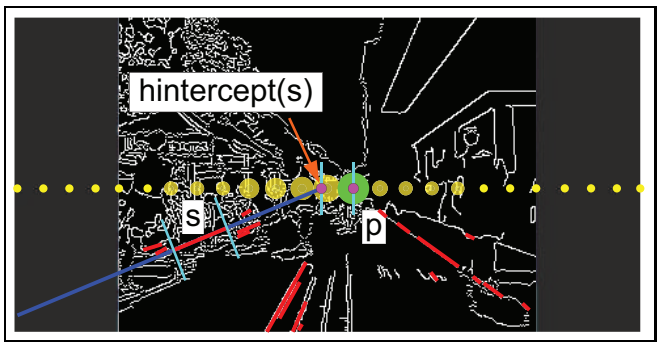

Figure 7: Vanishing point (VP) voting. The VP candidates are illustrated by disks on the calibrated horizon line, with radii proportional to their respective accumulated votes from the detected segments. For clarity, the figure only displays segments that support the winning VP. A segment s contributes to a vanishing point p by the product of its length and distance between p and the intersection point of the horizon line and a line extended from s, labeled as $hintercept(s)$.

segments (if all edgel in the segment are within five pixels) to the line, always re-estimating the equation after each addition. Once all 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.

Given a line equation from the previous frame and the current edge map, the system calculates a new equation by perturbing the line's end-points in the image. The first is the road bottom point, which is the onscreen intersection point with either the bottom or the side of the image. The second is the horizon support point, which is the line's intersection point with a line 20 pixels below the horizon line (called the horizon support line), as illustrated in figure 8.

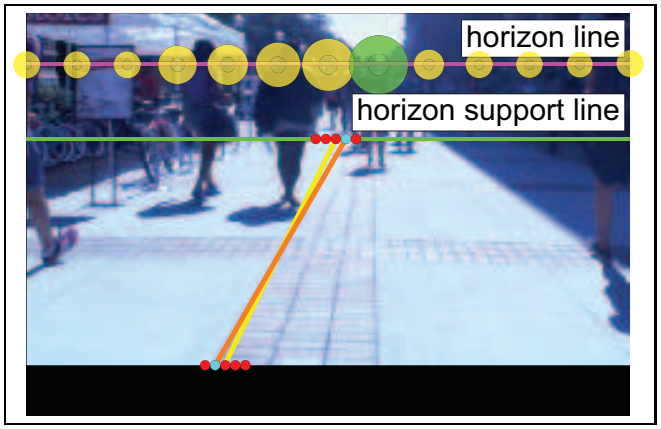

Figure 8: Line tracking process. The system tracks a line equation from the previous frame (denoted in yellow) in the current edge map by finding an optimally fit line (orange) among a set of lines obtained by shifting the horizontal coordinate of the previous line's bottom point (bottom of the image) and horizon support point (intersection point of line and the horizon support line) by +/- 10 pixels with 2 pixel spacing. The set of candidate shifted points is shown in red on the bottom and on the horizon support line.

The system tries to fit a new line by shifting the horizontal coordinate of both end-points 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 top pipeline also uses the same tracking mechanism for other purposes. One is to refine the VP estimate because the voting scheme only selects a location among sparsely spaced hypotheses at the calibrated horizon line. By using the intersection of road lines instead, the estimate is much more accurate. The second use of tracking is to help the recognition pipeline catch up to the current frame by projecting the newly discovered road lines through unprocessed frames accumulated during the recognition computation.

At the end of both pipelines, the system converts the VP pixel location to angle deviation from the road direction by linear approximation, from 0° at the center of the image to 27° at the edge (or .16875° per pixel). Here we assume the camera is fitted with a standard 60° field-of-view lens. The system then calculates the absolute road heading by summing the angular deviation with the current IMU reading. By approximating the road direction in absolute heading, the system does not directly couple the vision-based road recognition with motion generation. Instead it can use IMU readings to maintain robot heading. This, at times, is essential when the road is momentarily undetected visually, e.g., when going under a bridge.

To improve the heading estimation robustness, the system, which runs at 30Hz, considers the last 100 absolute headings using a histogram of 72 bins, each covering 5°. It only updates the road heading estimate if 75% of the values reside within 3 bins, updating it with the average value within those bins. In addition, the system also discards heading inputs that are farther than 30° from the current estimate, which usually occurs during obstacle avoidance. The resulting filtered VP-based absolute road direction is called θ_{ψ_t}.

This angle by itself is actually not accurate enough to properly drive a robot. Even though the general heading is correct, a single degree of bias can slowly drift a robot to one side of the road. To rectify this, the system locks the robot to its initial lateral position by storing the locations where the road lines intersect with the bottom of the image and trying to maintain these coordinates.

The system converts the unit-pixel lateral deviation to metric length *dev* using an empirically calibrated distance of 9.525mm per pixel. The system then corrects for lateral deviation by adding a small proportional bias to θ_{ψ_t}.

$$\theta_{road} \;=\; \theta_{\psi_t} + atan(\delta / long(g)) \tag{7}$$

with θ_{road} being the final absolute road direction to be sent to the navigation system. The second term, which is the lateral deviation correction, is the angle created by the observed deviation δ and the current longitudinal distance from the robot to the goal location $long(g)$, computed using the *atan* function. Observe figure 9 for the illustration. The small correction is sufficient because the robot is already going in the right direction and lateral correction only needs to guard against long-term drift, in the scale of 1m for 30m traversal.

Figure 10 displays the road recognition system timeline. The canny edge map computations run in 3ms and precede both the full recognition (third row) and road line tracking pipelines (fourth). The former runs close to 40ms, with Hough segments lasting for 20ms, VP voting for 3ms, and road line extractions for 15ms. The line tracking process runs in 5ms, depending on the number of lines being tracked. Usually the system tracks no more than six road lines at a time.

The final road heading estimation itself is triggered by the IMU output, which reports the current robot absolute heading. This allows the system to quickly correct deviation between the robot and road heading, for both straight and curved roads. (Siagian et al., 2013a) has shown that the system is able to follow curved roads. In the paper, we experimentally tested the system on four paths, one of which (HWY path) contained curved segments. On average, the system performed only slightly worse (10cm lower) than its average performance. The key to following this type of road is being able to extract as much of the straight image segments on the curved path as possible. In this case, these segments point toward a moving vanishing point, which coincides with the changing direction of the road.

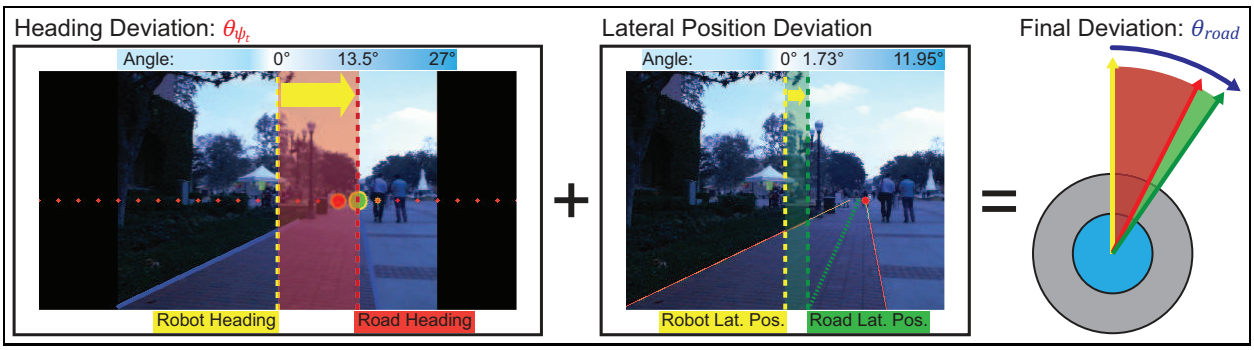

Figure 9: The conversion of the vanishing point (VP) shift from the center of the image and of the lateral deviation from the original location to robot lateral position. In the VP shift, each pixel (in this case 120 pixels) away from the center is multiplied by .16875°, while on lateral deviation, every pixel (in this case 22.8 pixels) away from the original position equals to 9.525mm. Since the lateral deviation δ is of low priority, it only needs to be corrected by the time the robot arrives at the top of the local map or in the goal longitudinal distance $long(g)$, which is 7.2m. The resulting angle computes to $atan(\delta/long(g)) = atan(22.8 * .009525/7.2) = 1.73°$.

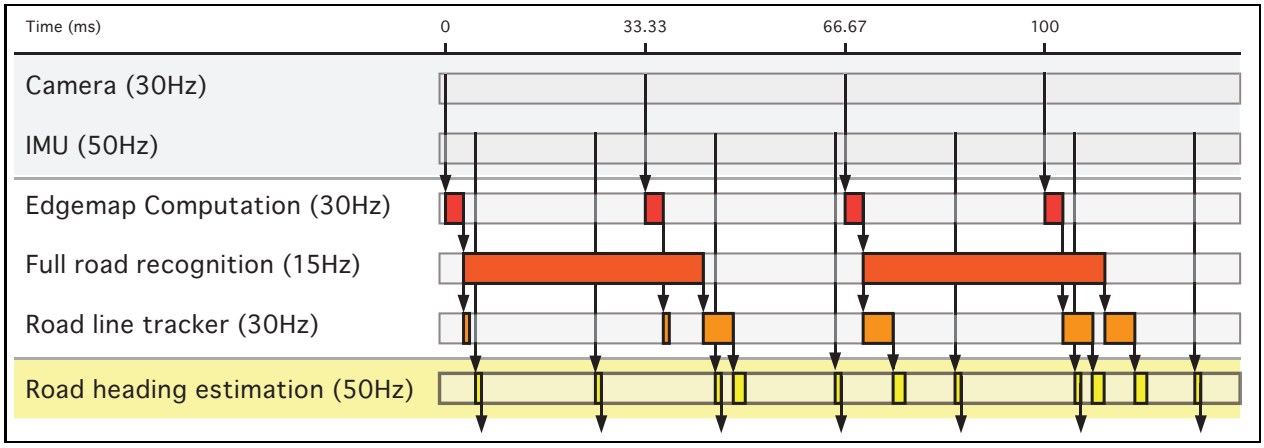

Figure 10: Diagram of road recognition timeline. The figure lists all the components in the system: camera and IMU, edgemap computation, the two visual recognition pipelines, and the final road heading estimation.

3.2.2 Constructing the Grid Occupancy Map

The system updates the grid occupancy values using LRF proximity distances that are already appropriately aligned with respect to the road heading. The grid values are initially set to -1.0 to indicate unknown occupancy. The grid cells are updated only if they are hit by or in line with the LRF rays. For each ray, the system draws a line from the robot location to the endpoint. If the laser line goes through a grid cell, indicating that it is free, the occupancy value is set to 0.0. If the laser line ends on the grid cell, it is set to 1.0. This discrete valued map D is stored and updated at each time step.

To generate a path, the system first blurs map D using the equation below. The resulting map is called a blurred map B.

$$B(i,j) = \max_{-n_{size}<=d_i,d_j<=n_{size}} D(i+d_i, j+d_j) * (1 - \frac{\sqrt{d_i^2 + d_j^2}}{\sqrt{(n_{size}+1)^2 + (n_{size}+1)^2}}) \quad (8)$$

Here the value $B(i, j)$ is the maximum of the weighted occupancy values within an $n_{size} = 4$ neighborhood, where each neighbor's discrete value $D(i+d_i, j+d_j)$ is inversely weighted by its distance to (i, j). By setting the influence neighborhood to 4 or a full robot width, the repelling force of the obstacle starts to influence the robot half of its length away from its surface. Note that in the calculation of $B(i, j)$, $D(i, j)$ is included in the neighborhood with the distance weight of 1.0, while the weight terms for the farthest neighbors evaluate 0.2. In addition, the blurring is not applied to cells with unknown occupancy status, to keep their influence to a minimum. After blurring, these cells are given a conservative value of 0.99 to allow the path planning algorithm to go through them if all other options have been exhausted.

As the robot moves longitudinally toward the goal, the grid values are shifted down one row at a time, keeping the robot location as is. This means that the local map is robot-centric in denoting the location, but allocentric in heading direction, aligning with the road. The robot movement estimation, which is computed using the fused odometry, is sufficiently accurate for grid shifting purposes. Here perfect occupancy alignment is not critical because the robot quickly passes through the local environment and discards any briefly accumulated errors. Unlike in Simultaneous Localization and Mapping (SLAM) formulations, the system does not intend to create a complete global map, which requires reasoning about motion uncertainties. Even when the system encounters an occasional large odometry error, it can quickly recover from grid map misalignment because the occupancy values of every grid visible by the LRF is updated in each frame. This way the misalignment only affects the grids that are in the local map and in robot's current blind spot. This is fine because the robot always turns to where it is moving to, thus exposing as well as correcting the erroneous blind spots.

Observe figure 11 for an illustration of the grid occupancy process. It also includes a path biasing step, described in the next section.

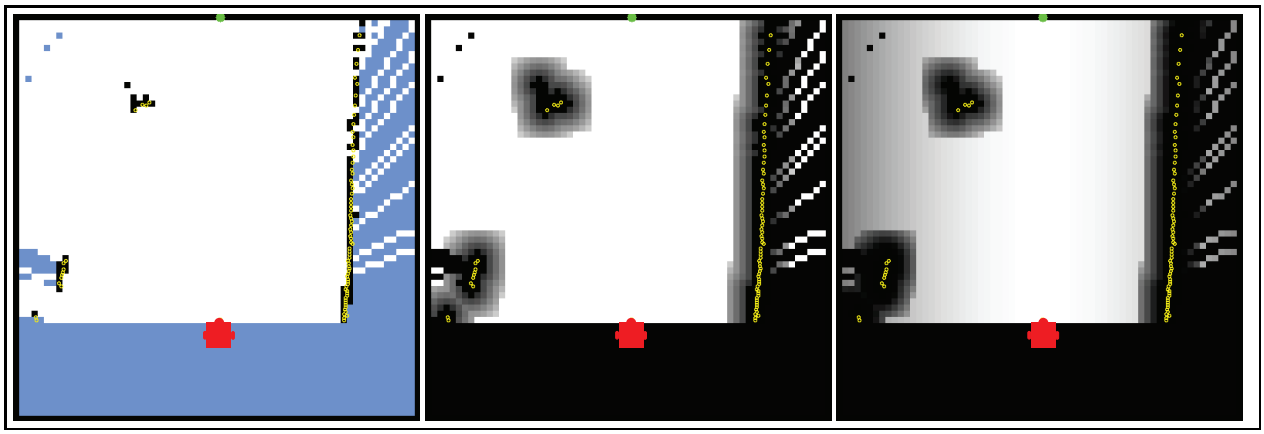

Figure 11: Three stages of the grid occupancy map. The first, leftmost, image is the discrete map D of all the discovered occupancy values, with the white color being unoccupied, black fully occupied, and cyan unknown. The second map is the blurred map B, while the last occupancy map F is biased by most recently produced path to encourage temporal path consistency. Also notice that the unknown grid cells, such as the ones below the robot, do not extend their influence out in both the blurred and biased map.

The grid construction timeline is shown in figure 12. The process, because the map uses a relatively coarse 15cm by 15cm resolution, has a low computational requirement of less than 1ms, and outputs results at 20Hz, after receiving each LRF readings. The path planning step, on the other hand, takes substantially longer.

3.2.3 Path Generation

The system first applies A* search to the local map to create a coarse path trajectory to the goal. For each grid cell, the search process creates eight directed edges to the surrounding nodes. The edge weight

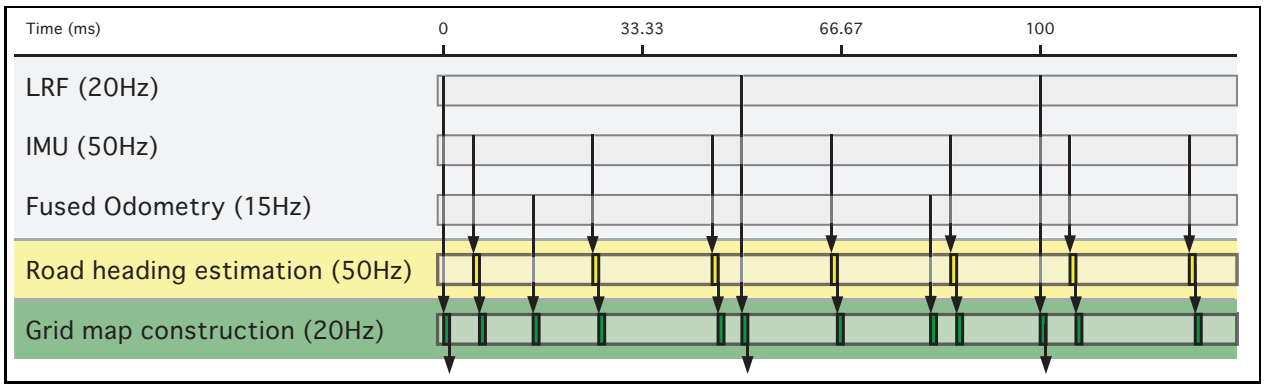

Figure 12: Diagram of grid map construction timeline. The figure lists all the components: the three utilized sensors, the road heading estimation, and the grid map construction process.

is computed by adding the edge length and $F(i,j)$ value of the destination cell, which is the sum of the destination $B(i,j)$ and a bias term to encourage path consistency. The bias term is calculated by adding the proximity of the destination cell to the corresponding closest point in the previous A* path, divided by the local map unit pixel diagonal length λ.

$$F(i,j) = B(i,j) + \min_{k=1 \ to \ n_{path}} \frac{\sqrt{(P_{k,i} - i)^2 + (P_{k,j} - j)^2}}{\lambda} \tag{9}$$

Here the previous path P is defined as k ordered and connected points P_k, with $P_{k,i}$ and $P_{k,j}$ referring to the i and j coordinate locations of the path point. To save time, the algorithm does not add edges to cells going into fully-occupied nodes $D(i,j) = 1.0$. Consequently, to include all non-fully occupied cells, all $F(i,j)$ values are capped at .99 for nodes $D(i,j) \neq 1.0$. In addition, the system zeroes out the bias if distance to the closest obstacle is less than 2 unit pixels, to allow for flexibility of finding an optimum nearby path.

We find that adding a bias and generating a path on every frame, which is computed efficiently because the coarse 15cm grid size, is more practical for encouraging temporal smoothness. The more burdensome alternative would be to advance and modify an old path when ever there are changes to the local map. Here, generating new paths allows for the fastest possible reaction to new changes without a need to explicitly detect them.

There are times when A* cannot return a path to the goal, usually because the robot is moving into a dead-end and the goal is on the other side of the wall. In this case the system creates a straight forward path, to keep the robot moving ahead until it is directly in front of a wall, within two robot lengths. This ensures that there is no way to the goal, and not because of inaccuracies in the local map construction, especially using far away LRF values. In cases where the robot is truly stuck, the system invokes an exploratory procedure by spinning the robot 360°. In the future, we plan to implement a more robust procedure such as moving backwards as well as considering pertinent contextual information.

To allow for maximum obstacle avoidance and to smooth out the discrete path, the system applies a modified elastic band algorithm (Quinlan and Khatib, 1993) using the current LRF laser scans as individual obstacles. Each path point undergoes an iterative process for $t_{max} = 20$, where at each time step the point is moved by an attractive force A to straighten the path, and a repulsive force R to repel it perpendicular to the closest obstacle. The attractive force is defined as:

$$A_{t+1,k} = \frac{P_{t,k+1} + P_{t,k-1}}{2} - P_{t,k} \tag{10}$$

On the other hand, the repulsive force R uses a vector $C_{t,k} = P_{t,k} - closestObs(P_{t,k})$, which extends from the closest obstacle $closestObs(P_{t,k})$ of path point $P_{t,k}$ to that point. The force can then be defined as:

$$R_{t+1,k} = \frac{C_{t,k}}{|C_{t,k}|} * \frac{m - |C_{t,k}|}{|C_{t,k}|} \tag{11}$$

Here the first term normalizes the vector $C_{t,k}$, while the second term creates an exponential magnitude weight, where the closer the obstacle the larger the repulsive force. Note that, when identifying the closest obstacle points, the system models the robot body more accurately as a square. Here the maximum obstacle distance $m = 5$ limits the obstacle consideration to within 5 grid units of the robot, otherwise $R_{t+1,k} = 0$.

Equation 12 below combines the two factors:

$$P_{t+1,k} = P_{t,k} + W_t * (\frac{1}{3} * A_{t+1,k} + \frac{2}{3} * R_{t+1,k}) \tag{12}$$

The weight W_t decays overtime, becoming linearly weaker 5% at a time from .2 to .01, or $W_t = .2*(1.0-\frac{t}{t_{max}})$.

To compute the motor command using the smoothed path, the system takes the path's first step and applies the Dynamic Window Approach (DWA) (Fox et al., 1997). It not only calculates the deviation between the current robot heading and the heading of the initial path step, but also takes into account the robot dynamics. This is done by only considering commands within the allowable velocity space based on the estimated velocity calculated from the previous command, its execution delay, and motor and body dynamics. Furthermore, the approach then accurately simulates each resulting arc trajectory of the allowable commands (translational and rotational velocity), modeling the robot shape to test whether any part of the robot would hit an obstacle. In the end, the system selects a motor command that is expected to best achieve the target heading and velocity, while staying clear of the obstacles. Figure 13 illustrates the three stages of the motion generation.

Note that even though the displayed DWA trajectory deviates from the smoothed path, this is only for one step. When the next time step is executed, the front of the robot is expected to be in the dotted location. At that point, the system is going to recompute the path again. After all the steps are computed and executed, the actual robot trajectory usually ends up similar to the smoothed path.

The path planning timeline, which integrates the sub-systems in our presented system, can be viewed in figure 14. The path planning sub-system creates the occupancy grid map and computes A* in 5ms, path smoothing step (elastic band algorithm) in 1.8ms, DWA step in 5ms, which totals to about 12ms. The output path itself is triggered by the grid-occupancy map output, which is triggered by the LRF output.

While other modules influence the robot motion in almost every time step, localization updates the intermediate goal in the local map quite infrequently. For the most part, the goal location is placed in the canonical straight ahead position (observe figure 13). There are, however, a few occurrences where the goal is moved, such as during a turn procedure, or when the goal is within the local map dimensions. For turns, the localization system sends a command to the local map to initiate a turn. It specifies the turn angle as the angle created by the involved incoming and outgoing edges in the map.

The navigation system then moves the goal to the appropriate location, triggering a path generation that forces the robot to turn immediately. The system then lets the robot move until its angle is aligned to the goal direction, usually in less than two seconds. When the turn is completed, the local map is rotated so that the goal location is at the canonical straight-forward position, and the robot can proceed in a normal fashion.

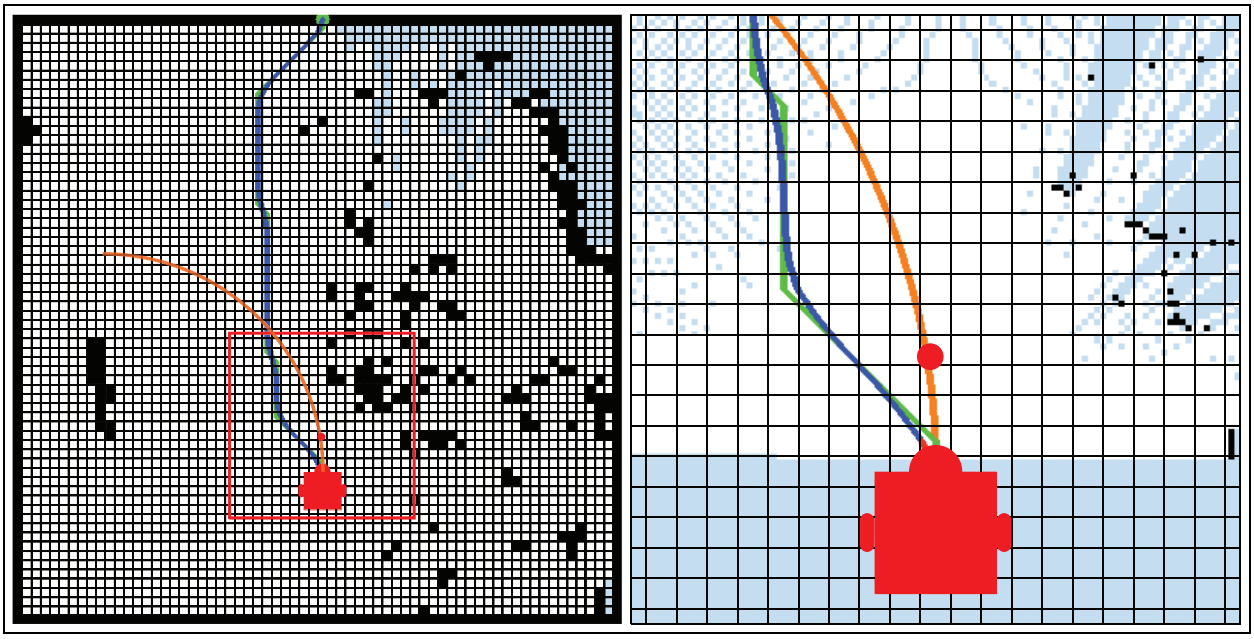

Figure 13: The three stages of the path trajectory and motion command generation. The figure displays both the full local map on the left and a zoomed part within the red rectangle on the right for a clearer picture. The system first runs A* to generate a rigid path shown in green in both images. The system then deforms the rigid path (in blue) to optimally and smoothly avoid obstacles. Finally, the system uses Dynamic Window Approach (DWA) to generate the actual command, one that accounts for the current robot dynamics. That resulting command is modeled accurately as a curve in orange. In addition, the figure also displays the predicted location of the robot, noted by the red disk.

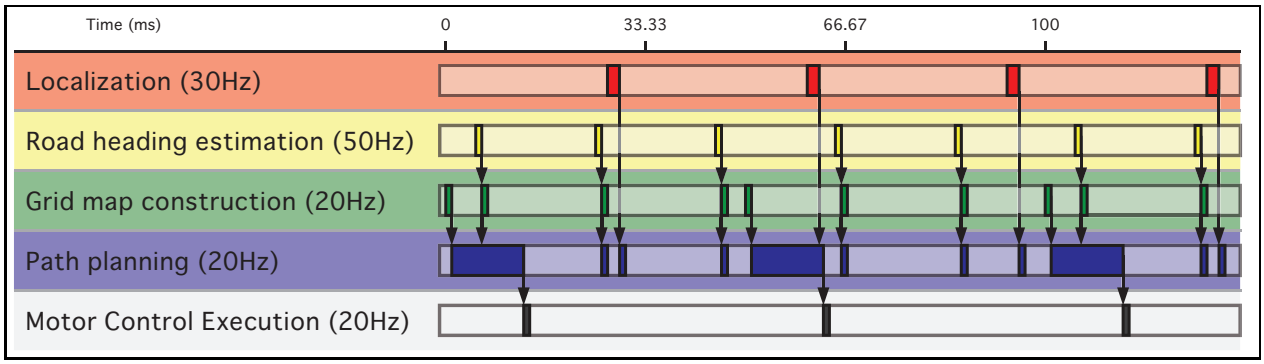

Figure 14: Diagram of full system timeline. The final motor command to be executed by the robot (bottom row) is generated by the path planning sub-system. It utilizes all the previously described sub-system, with grid map construction being the trigger.

The framework has the flexibility to allow the robot to still perform robustly despite not using some of the sub-systems. In the absence of the localization system, the robot can keep itself on the road. If the road recognition is also absent, the robot then simply tries to move in the direction of its heading when the system is turned on, while also avoiding obstacles. In addition, in the case where any of the sensors (camera, LRF, and IMU) is broken, it is easy to set up the corresponding sub-systems (localization, road heading estimation, grid map construction, respectively) to trigger on other events, such as fused odometry.

Table 1: Experiment Sites Information

Site	Route Length (m)	Pedestrian Density (Person/m)	Weather/lighting Condition	Speed (m/s)	Total Training Frames	Total Testing Frames	Total Time (s)	Obstable Avoidance Time (s)
Library	511.25	.19	4 - 6pm, partial sun	.61	16621	13825	838.35	29.99
		.97	12 - 2pm, sunny	.44	16688	34979	1165.88	73.13
Professional	615.48	.06	4 - 6pm, cloudy	.73	21488	20000	840.79	0.00
		.36	12 - 2pm, sunny	.89	9702	20773	692.37	46.10
Athletic	401.16	.04	4 - 6pm, clear sky	.77	19022	11017	522.21	13.67
		.43	11am - 6pm, sunny	.68	9102	17499	583.24	27.10

4 Testing and Results

We rigorously test the system using our robot, Beobot 2.0 (Siagian et al., 2011), pictured in figure 2, at the University of Southern California campus (USC) in Los Angeles. We construct three routes of 511.25m, 615.48m, and 401.16m in length (paths shown in figure 15 in red, blue, and green, respectively) through many parts of the campus. The goal is to cross the finish line at the end of the route, wherever the robot is laterally on the road. In the figure, we also include paths in black, which are roads that have been traversed during system development but are not used in the documented experiment. They show the extent of the network of roads that we have used during testing. In addition, some of these paths are currently too dangerous for the robot to traverse. One is the leftmost portion of the network. This is a narrow sidewalk with steep boundary cliff and people walking behind and toward the robot. Currently, the situation forces the robot either to quickly get off the sidewalk or be stuck thinking that the road is a dead end.

Beobot 2.0 logged over 3 hours 17 minutes or 10.11km of recorded experiment conducted within a three month winter period (December - February). The average robot speed during autonomous driving is 0.66m/s (about half the average walking speed of people), with a maximum of 0.89m/s. The difference is caused by a dropping battery charge as the experiment wears on. A video summary can be found in (Siagian et al., 2013b), while a snapshot is displayed in figure 16.

For the experiments, as seen in figure 2, we use a subset of Beobot 2.0's sensors: a front-facing monocular camera and LRF (117cm and 60cm above the ground, respectively), as well as wheel encoders and IMU. The camera outputs 320×240-pixel images, which are kept at the same dimension by the road recognition module, but scaled to 160×120 for vision localization. We did not apply camera calibration or lens distortion correction, which may help in salient region matching.

Beobot 2.0 has a sixteen-core 2.6 GHz distributed computing platform, which we take full advantage of by employing a subscribe/publish architecture to manage all functionalities, from sensor data acquisition, localization, navigation, to actuation. In addition, the setup simplifies the expansion process if we want to add capabilities such as object recognition and human robot interaction. All the code is available in the INVT toolkit page at (Itti, 2012).

We selected routes that make up a fairly complete representation of the different environments at USC. They are also picked to vary the localization and navigation challenges and conditions, which are listed in table 1. Each route consists of four segments. Between them are intersections that force the system to make decisions of which road to take. Scenes from each route are displayed on separate rows in figure 17. The first route is called the libraries route (LIBR) because the path goes through a cluster of libraries at the heart of the campus. This route includes roads with some of the densest crowds. The second route at the south part of USC goes through the professional schools (PROF) neighborhood, where there are more trees. The third route, going towards the north part of campus, is called the athletic (ATHL) route, which rounds out the variety of scenes in the experiments.

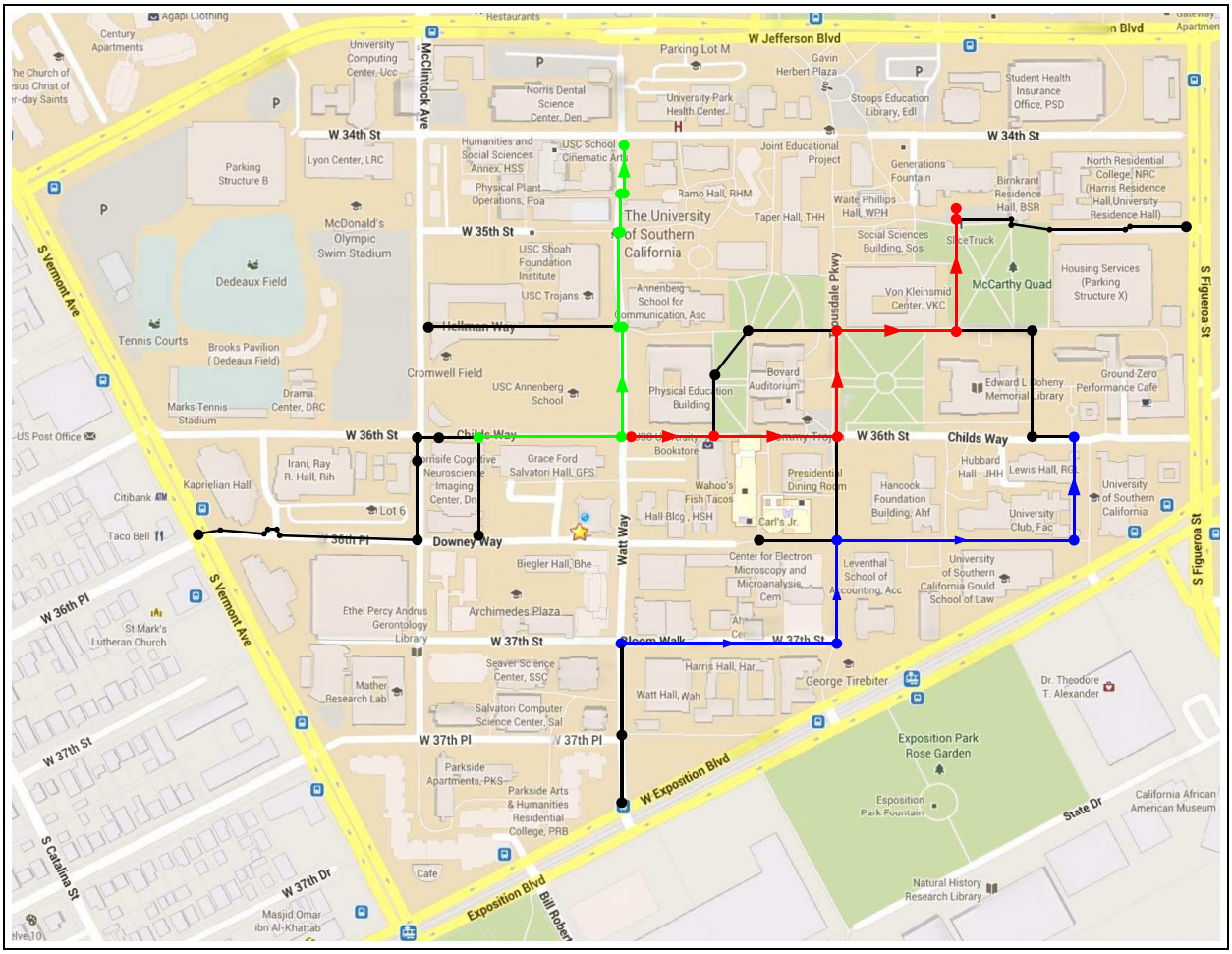

Figure 15: Map of the experiment routes at the University of Southern California. They are the libraries (LIBR) route, indicated in red, the professional schools (PROF) route in blue, and the athletic complex (ATHL) route in green. The remaining routes in black are not used in the documented experiment but have been traversed during system development.

There are parts of the full system that have been previously published. (Siagian and Itti, 2009) shows that the vision localization system can maintain comparable results on a number of lighting conditions at different times of the day, from the brightest (noon time) to the darkest (early evening), and in real time (Siagian et al., 2011). (Siagian et al., 2013a) has also demonstrated that the visual road recognition system performs better than the state of the art (Rasmussen et al., 2009; Kong et al., 2010; Chang et al., 2012) on a standard (offline) dataset (Chang et al., 2012).

In this paper, because the full system connects the navigation and localization modules, we examine the effect of one module upon the other. In addition, since both modules have been shown to be robust in the presence of various lighting conditions, we focus on the effect of crowding on our system. That is, we test the system on two conditions: low and high crowd; the former during off-peak hours, while the latter during peak hours, with many pedestrians on the road. (Siagian et al., 2013a) has shown that the road recognition system is affected by crowding more so than lighting. Here, we complete the analysis by rigorously testing the whole system.

We carry out a number of testing procedures, measuring different metrics. In section 4.1, we focus on the localization system, while section 4.2 is on navigation. In the former, we isolate the localization problem

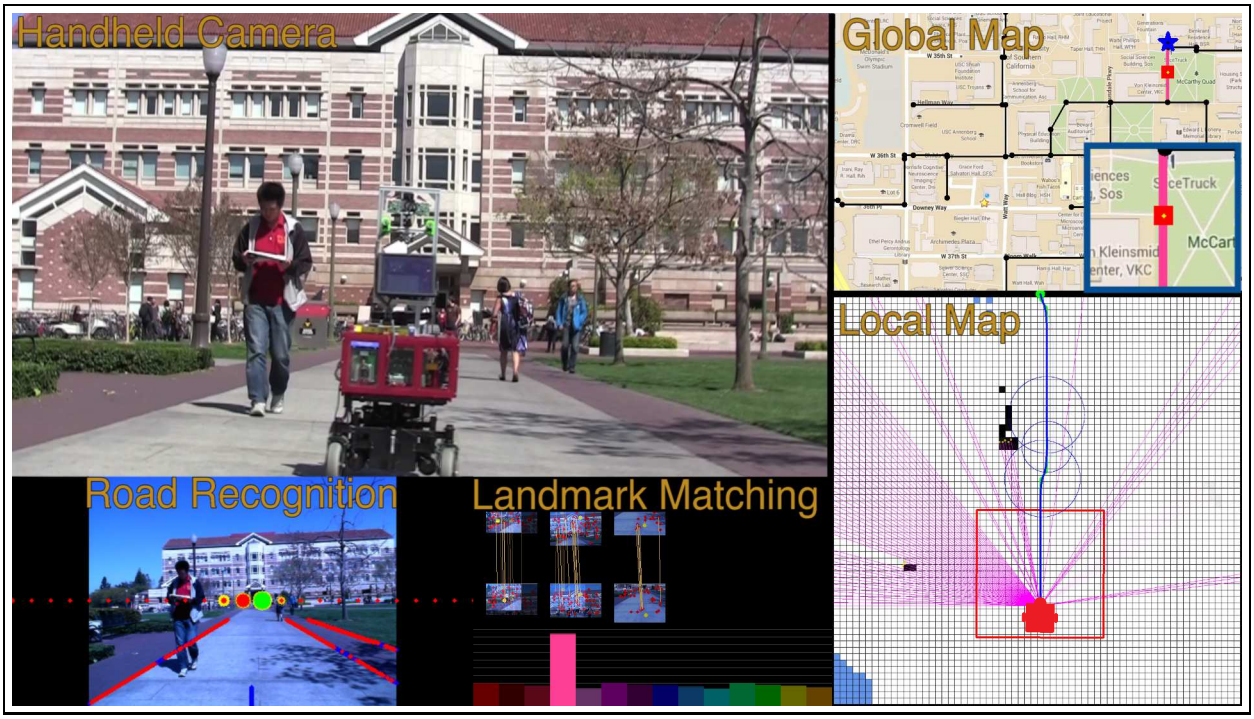

Figure 16: A snapshot of the system test-run. Top-left (main) image displays the robot moving towards the goal. Below is the road recognition system which identifies the road heading (indicated by the green disk) and the lateral position of the robot (blue stub on the bottom of the image). To the right are three salient landmark matches as well as the segment estimation histogram below. The top-right of the figure shows the global topological map of the campus, with the current robot location indicated by a red square, and the goal by a blue star. It also draws the current segment in pink, matching the segment estimation output underneath landmark matching. Finally, the right bottom of the figure displays the local grid occupancy map, which shows a robot path to avoid an approaching pedestrian.

Figure 17: Examples of scenes from the libraries (LIBR), professional schools (PROF), and athletic complex (ATHL) route, each displayed in its respective row. The figure also depicts various challenges that the robot encountered, such as pedestrians, service vehicles, and shadows. These images are adjusted to remove the bluish tint in the original images for better display.

by comparing its performance when the robot is manually driven versus fully autonomous. In addition, we systematically manual drive the robot on a straight-line at different offsets from the center of the road to gauge how navigation error affects localization results. We also isolate localization results at the junctions. Junction is a critical part of a route because it is where a robot has to make a turn. In the navigation section, we evaluate how well the navigation system maintains the robot's lateral position throughout traversal, and how it is affected by obstacle avoidance.

Altogether, we evaluate the system using twelve full path traversals, four for each route, for all combinations of conditions: autonomous mode in low crowd, manual driving in low crowd, autonomous mode in high crowd, and manual driving in high crowd. In addition, we also add four 180m manual runs to test the effect of lateral shift on localization results. Although there are many more environments that we can test the robot in, these selected sites encompass a large part of the challenges that outdoor autonomous mobile systems face today. We believe there are many lessons that can be gained from testing the robot at these challenging sites and under the proposed conditions.

4.1 Localization Testing

We measure the localization system accuracy by comparing its output with manually obtained ground truth. We decided to use this metric because the system never completely failed to localize, and evaluating displacement error can provide a better understanding of how it performed. The error measured here is in the longitudinal direction, along the graph edges of the topological map. We record the ground truth by sending signals to the system, indicating the true current robot location. The signals are sent at specific parts of the path and are spaced appropriately (as small as 3 meters) to accurately record the robot's location throughout the experiment. Assuming the robot is running at approximately constant speed, we use interpolation to fill in the ground truth between signal points. Note that these signals are only processed after testing and are not used during run-time. For our experiments, we manually initialized the system with the starting location. In (Siagian and Itti, 2009) we have shown that the system can re-localize from kidnapped robot as well as random initialization. Here we decided not to do so because the system currently does not converge to the true location in reasonable amount of time with 100% certainty.

To create a landmark database and a segment classifier, we run the robot manually through the routes to take images for our training procedure (Siagian and Itti, 2008b). We have systematically shown in (Siagian and Itti, 2009) for the localization system and (Siagian and Itti, 2007) for the segment classifier that training at different times with three hours intervals endows our system with lighting invariance. In those papers, we test the system from 9 am to late afternoon, after performing multiple the prescribed training runs. We found that the segment classification errors are within 5 percent of the average, while the localization errors are within 60 cm, regardless of lighting condition. Thus, here, we test the system at different times of day, but only with the training and testing taken in similar lighting for each site. Table 1 lists the number of training and testing images involved.

The localization boxplots can be viewed in figure 18 for all three sites as well as the total, in both crowding condition and driving mode. We also show in figure 19 an example run from the PROF route under low crowd condition for more detailed observation. To reduce the effect of dependence in the samples we average the errors every 30 samples or within 1 second (localization output is 30Hz) before creating the box plots. In addition, because our result distributions are not gaussian, we use non-parametric statistical significance tests. Because of this, we do not test for interaction between effects, e.g. whether crowd level causes greater error for autonomous navigation than for manual driving.

The results show that the system is able to satisfactorily localize within 0.97 meters (median) of the true location while driving autonomously and reaching goals of more than four hundred meters away. The farthest localization disparity is 7.15m, which happens in the middle of a long segment, allowing the system to recover. The main source of error, which is prevalent throughout all conditions, can be attributed to the less exact nature of the salient region matching step. To speed up the process, the system uses small salient

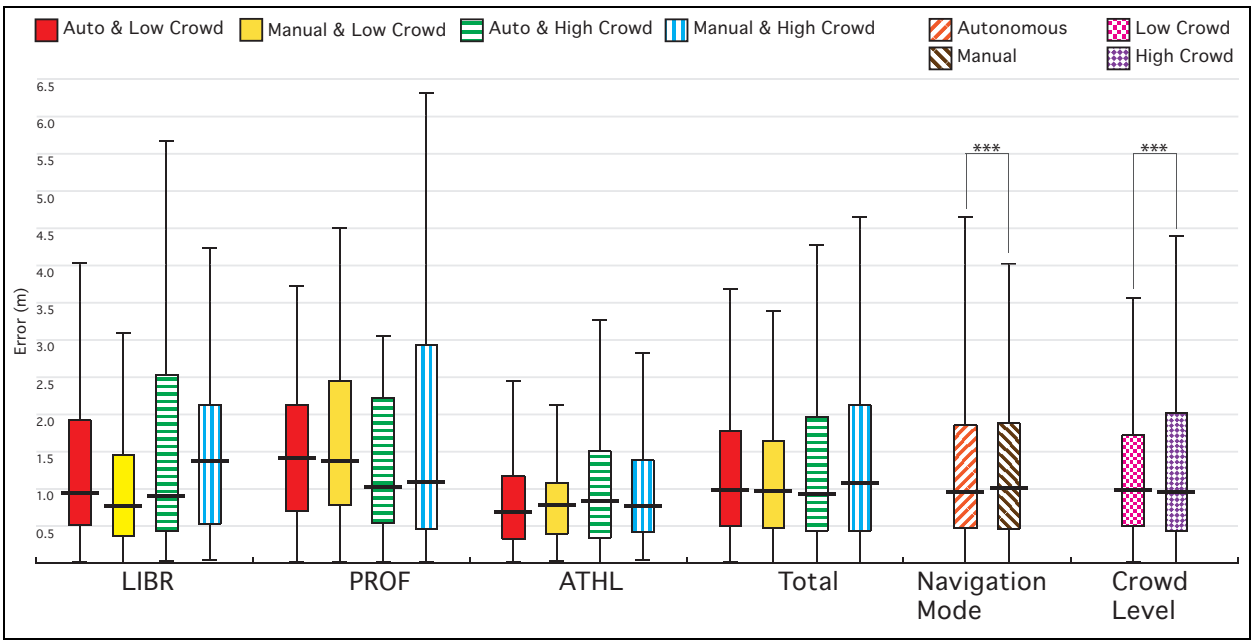

Figure 18: Box plot of metric localization error from different routes, for both manual and autonomous navigation, and in both low and high crowd conditions. In addition, the figure also reports the total performance accuracies for each condition. The figure shows that the overall autonomous mode error median is 0.97m, while medians of the various conditions remain low, from .75 to 1.5. Each box plot notes the median (the thicker middle line), the first and third quartile, and the end whiskers, which are either the minima/maxima or 1.5 * IQR (inter-quartile range). The stars above the lines joining various box plots note the statistical significance with one star indicates $p < 0.05$, two stars $p < 0.01$, and three stars $p < 0.005$.

regions of about one third the downscaled 160 by 120 input image. For landmarks that are near the robot, the small size regions are sufficient to pinpoint locations with high accuracy. In fact, whenever the system observed nearby landmarks, the localization error tends to decrease. However, when the landmarks are far away from the robot, they can appear identical from a large range of viewing distances. Because of the low discrimination power, often times, the system cannot quickly correct a sizeable steady-state error.

The main problem is that there is no way of knowing which landmarks are far from the robot as the system only records the robot location when viewing them. A possible labor-intensive solution without incorporating long range 3D sensors would be to annotate the estimated distances of the landmarks to the robot. This way, the system would know that landmarks in the form of the top of buildings are hundreds of meters away and would not be as discriminating as signs on top of a nearby door.

In terms of the effect of routes on system performance, the data shows that it is statistically significant (Friedman test, $p < 0.005$). As shown in the figure, the Athletic (ATHL) route generally produces lower error distributions than that of the libraries (LIBR) and professional school (PROF). We found that the disparity comes from a sustained spike in error, which often occurs when the robot is driving on a long segment and which the system continually utilizes far away silhouettes at the end of the road as a landmark. An example is taken from the last segment of PROF route, where the system sustains errors above 3m for more than 50m (observe figure 19). As the robot is moving through a corridor (shown in figure 20), the light at the end makes that part of the image very salient, rendering other parts less conspicuous by comparison.

As for the difference between low and high crowd conditions, the difference is statistically significant (Friedman test, $p < 0.005$) but with near equal medians (0.98m low crowd vs. 0.96 high crowd). Most times, the localization system is able to identify enough landmarks among the pedestrians just as well as when the crowd is much sparser because our training procedure rejects moving objects as possible landmarks during

Figure 19: Detailed diagram of localization error while the robot autonomously navigates the professional schools (PROF) route during low crowd condition. The error is indicated by a gradation of red color, with the darker hue being higher error.

Figure 20: Landmark recognition in a building corridor at the last segment of PROF route. The opening at the end is the only salient landmark that is consistently recognized at different distances, with minimally observed changes throughout the traversal.

training (Siagian and Itti, 2008b). However, there are times when pedestrians do affect the system, particularly when they occlude the landmarks that are at ground level. These landmarks tend to be the ones that are closer to the robot. When this happens, the robot has to rely on landmarks that are higher above pedestrians. They are likely to be farther away, thus reducing the accuracy of the localization system.

In addition, we also observe a statistically significant localization performance differences between autonomous and manual-drive navigation (Friedman test, $p < 0.005$) with the autonomous drive median of 0.96m being lower than that of manual driving (1.00m). This is quite surprising given that in autonomous

drive the robot can veer in and out of the middle of the road due to the granularity of the road heading perception. That is, it takes substantial deviation from the road center for the system to react to the error. Furthermore, it can also be attributed to robot response delay, uneven ground surfaces, and motors not having equal efficiency, biasing the robot to drift to one side. In manual drive, on the other hand, the robot moves steadily in the middle of the road, which is the same vantage point as during training.

When the robot is not moving at the center of the road, there are fewer landmarks that are matched with the ones stored in the database, which lowers the system accuracy. We demonstrate this point by manually driving the robot at different lateral positions, 0m, 1.33m, 2.33m, 3.33m away from the road center at the first segment of LIBR route. We selected this segment because it has sections with many nearby salient landmarks as well as sections that primarily use a far away silhouette at the end of the segment.

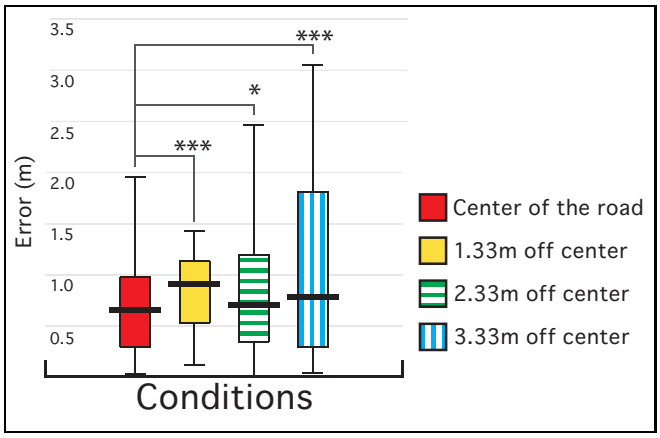

Figure 21: The localization error box plot for lateral deviation of 0m, 1.33m, 2.33m, and 3.33m from the center of the road. All the error differences compared to that of driving in the middle of the road are statistically significant based on Wilcoxon rank sum test ($p = 1.093 * 10^{-7}$ vs. 1.33m, $p = 0.015$ vs. 2.33m, and $p = 2.476 * 10^{-5}$ vs. 3.33m).

As we can see in figure 21, there are statistically significant differences in performance between 0m lateral position and the others, with the system becoming increasingly less accurate as it moves away from the center. The good news is, even at a deviation of 3.33m, the localization system still does not completely misplace the robot. This leads us to believe that, as long as the robot does not permanently travel far away from the center of the road, instead moving in and out of the center, the system can match the localization performance of the manual driving.

Another aspect of the testing that we observe closely is during obstacle avoidance. In total, in our data, there are 25 trajectory-changing obstacles that affect the robot's heading. This definition differentiates a select few from many other nearby pedestrians seen in the camera but not close enough to warrant avoidance. A majority of them only results in slight deflections in the robot's trajectory, and not a clearly visible maneuver. Refer to table 1 for the accumulated times that the robot spent avoiding obstacles, listed on the last column.

As shown in figure 22, we observe statistically significant performance differences between low and high crowd conditions during obstacle avoidance maneuvers (Friedman Test, $p < 0.005$), low and high crowd conditions during non-obstacle avoidance traversals (Friedman Test, $p < 0.005$), as well as between obstacle and non-obstacle avoidance traversals for both crowd levels combined (Friedman Test, $p < 0.005$). It should be pointed out that the median difference for the last comparison is only 1cm, 0.95m during obstacle avoidance versus 0.96m during non-obstacle avoidance. This stability, which is also shown by the respective distribution shapes, demonstrates that the localization system can endure these significant but short perturbations. At times we see that an avoidance maneuver actually allows the robot to go back to the center of the road, improving the localization accuracy. In fact, on average, the localization error actually decreases from 1.39m to 1.09m before and after avoidance.

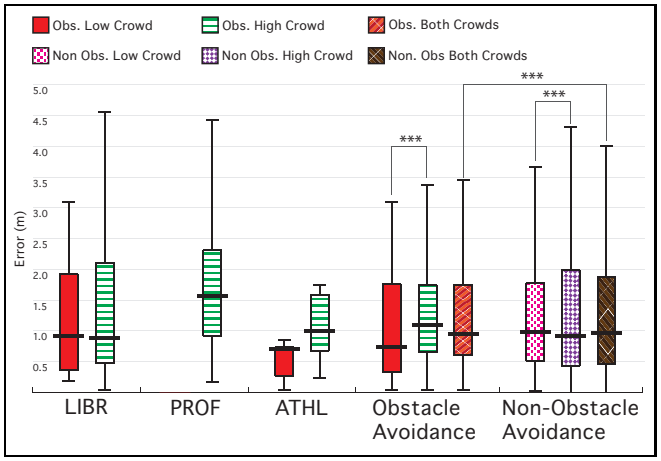

Figure 22: Box plot of localization errors during obstacle avoidance on the different routes, and in both low and high crowd conditions. Note that the result for the PROF route during low crowd is missing because the robot did not execute a single obstacle avoidance maneuver on that run.

For the most part, the amount of localization error carried by the system is manageable, especially in the middle of segments, where the task of the robot is mainly to proceed forward. However, it becomes critical when the robot is trying to turn to the next segment at an intersection. In figure 23, we plot the turn locations with respect to the center of the intersection.

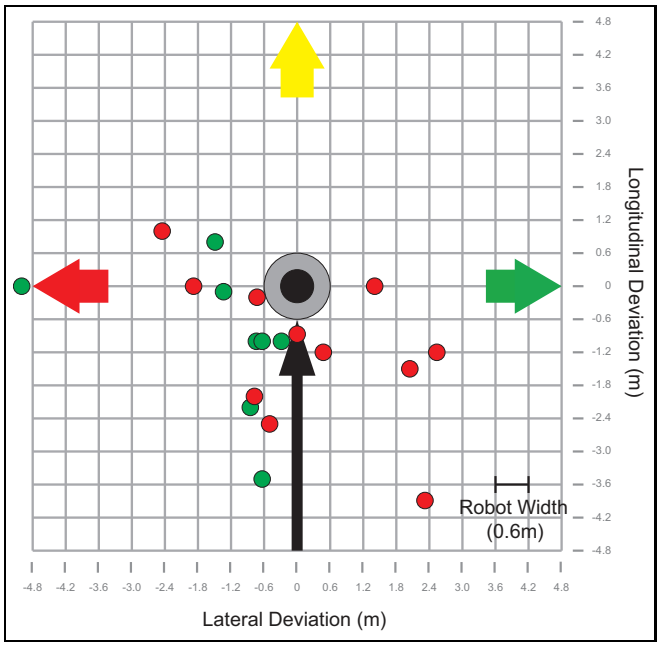

Figure 23: Plot of turning locations throughout the experiment. The black partially-blocked arrow coming from the bottom of the figure indicates the direction of where the robot is coming from. It is pointing toward the gray and black circle representing the center of the intersection, which is the ideal turn location. The color of the dots, which corresponds to the color of the arrows, indicates where the robot is trying to go: red is for turning left, green for right, and yellow for straight. Each red, yellow, and green dot represents where the robot was when turning (the robot turns in place).

Fortunately, considering that the robot is traveling on campus roads of 6 to 8 meters in width, it never misses the road completely. Even when the robot needs to turn to narrower roads, it manages to do so because

these roads tend to be in between buildings. An example is when the robot is entering the corridor at the last segment of PROF route. In this case, the obstacle avoidance, or wall following behavior, kicks in and funnels the robot to the segment. We believe that in order for the robot to properly turn at the center of an intersection, both localization and navigation must work in concert. That is, localization can seed the system that it is nearly time to turn, while navigation should then estimate the configuration of the junction using the involved road boundaries to fine-tune the turn location.

4.2 Navigation Testing

We measure the accuracy of the system in being able to navigate and maintain its position in the middle of the road on many different road types and challenges. They range from clean roads with clear borders, roads with complex markings, to roads without clear boundaries. Navigation error is defined as the physical lateral distance of the robot center from the center of the road. We measure the ground truth by manually annotating the robot's lateral position when crossing certain points of the path. We then interpolate the points in between using odometry.

Figure 24 reports the result, which is divided by sites, and then by crowding conditions. Overall, the system is able to stay within 1.37m median of the center, which is acceptable given the challenges that the robot faced. However, clearly, there are a number of aspects that needs improving.

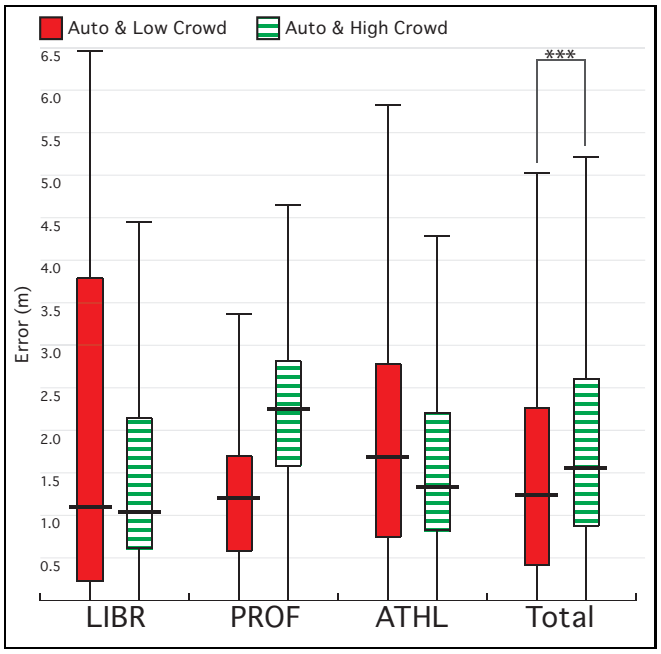

Figure 24: Results of the navigation system in each site and under low and high crowding conditions. We measure the error as lateral deviation of the robot from the road center. The far right bars display the distributions of low vs. high crowd performance, which is statistically significant (Friedman test, $p < 0.005$).

The data shows that there is a statistically significant difference between the low and high crowd conditions (Friedman test, $p < 0.005$), with the median of 1.23m for low crowd vs. 1.55m for high. As mentioned in the previous section, excluding numerous slight changes of direction, there are only a low number (25 in total) of obvious obstacle avoidance situations. One reason for this is because the 7.2m front extent of the local map allows the robot to register obstacles early enough to move away gracefully. In addition, the robot only encounters a small number of static obstacles because it is moving in the middle of the street. Also, because of the robot's appearance and relatively slower moving speed, pedestrians are aware of it approaching them. In some cases, however, the robot does have to break from its path to avoid collision. And, sometimes, the

Figure 25: Trajectory of the robot on the PROF route trial run under low crowd condition. Here the deviation from the ground truth center of road is indicated by the gradation of the red color, with the darker hue being higher error. In addition, the green bar behind the trajectory indicates that the error is within 1.5m of the center of the road, while the wider black bar is for 5m.

navigation performance suffers because of it.

It is important to note that the starting point of an avoidance maneuver's temporal window is when the robot first rapidly changes its direction, and ends when it stabilizes its lateral position. For some situations, however, the system error remains long after the maneuver is deemed over. For example, during the LIBR test run in low crowd condition, the robot has to avoid two people, street signs, and a long bike rack in succession (observe figure 26). Because the compound maneuver keeps the robot on the left flank of the road (3m from the center) for so long, the system resets its current lateral position, thinking that the robot is at a new center of the road. Had the robot moved back to the center, as opposed to staying out in the flank for more than 70 meters, the overall median would have been 0.87m, lower than the actual 1.09m error.

We decided to add a reset option because the current road recognition system does not have the capability to recognize the true center robustly. Resetting the center whenever a new stable lateral position is established allows the system to react to changing boundary positions. If the system is able to estimate the road center, it would help not only for obstacle avoidance recovery, but also in the case where the localization system turns at the wrong point of the intersection at the start of a segment. Currently, the navigation system assumes that the original lateral position is the center of the road, and simply tries to maintain that spot.

We also analyze the navigation error during obstacle avoidance, with the result shown in figure 27. Again,

Figure 26: Successive avoidance maneuver by the robot. It has to avoid two people (first image), street signs (second), and long bike rack (third).

the figure shows statistical significance on all cases: navigation error during obstacle avoidance for low vs. high crowd conditions (Friedman Test, $p < 0.005$), during non-obstacle avoidance traversals for low vs. high crowd conditions (Friedman Test, $p < 0.005$), as well as between obstacle and non-obstacle avoidance traversals for both crowd conditions combined (Friedman Test, $p < 0.005$). Here the difference for the latter comparison is more pronounced, 1.58m vs. 1.36m medians, respectively. In addition, the disparity is also supported by the average error increase between start and end of obstacle maneuver of 32 cm.

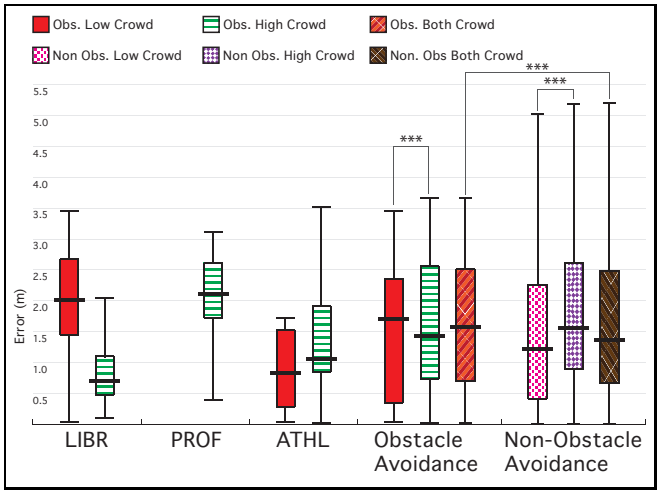

Figure 27: Navigation errors during obstacle avoidance from different routes, and in both low and high crowd conditions. Note that the result for the PROF route during low crowd is missing because the robot did not execute a single obstacle avoidance maneuver during the time.

Even without obstacle avoidance, in the case of the PROF route during the low crowd condition (indicated by missing bar in figure 27), there is still error in the system. A major reason for this is because, in some areas, visual cues for road direction are very sparse. An example is the water fountain area in the PROF route, where the road changes seamlessly to an open space before continuing to another road (observe figure 28). In the absence of these visual cues, and thus of a vanishing point, the robot simply maintains the last road IMU heading estimate while avoiding obstacles whenever needed. Despite this difficult stretch, however, the robot was still able to arrive at the target location, about 200m further.

On a related topic, open areas present a challenge in terms of constructing a topological representation because the robot can move about in any direction. In a truly open area devoid of obstacles, there can be any number of maps that can be created to describe the space, but in the end, these do not matter since the space is open and hence can be traversed in any way. Our strategy with mapping open spaces thus incorporates the robot behavior of keep going straight while avoiding obstacles. So, for example, in the case of a large plaza with 4 roads connecting to it from the North, East, South and West, we just reduce the plaza to its center point and extend the 4 roads to meet at that point. If the open space does contain fixed

obstacles, these will be represented as additional nodes in our topological map.

Figure 28: Views of the environment as the robot is moving through a water fountain area.

Throughout the experiment, there were only a few times when the robot failed to detect obstacles in its path. On these occasions, we stepped in front of the robot to assert the presence of the obstacles, and let the system adjust by itself. One cause of the mistakes is because the planar LRF is placed too high above the ground to detect shorter or protruding object parts such as the seat of a bench. However, we find that there is no correct height because, if it is placed low, the robot may attempt to drive underneath an obstacle. Other reasons to note are the inability of the LRF to consistently detect thin objects (e.g., poles, fences), and objects with holes such as a bicycle approached at certain angles. Furthermore, the navigation system currently is unable to detect negative obstacles such as a ditch or a hole, making it unsafe to navigate on roads with hazardous boundaries such as sidewalks, ramps, and cliffs. In section 5.3, we describe our plan to rectify these problems.

5 Discussion and Conclusion

In this paper we present a real-time visually-guided autonomous localization and navigation system. We demonstrate the system's efficacy using a wheeled robot and driving it through long-range and challenging routes of at least 400m at the University of Southern California (USC) campus, for a total of 10.11km of documented testing. In each route, the robot has to show its robustness in navigating among crowds, turning at several intersections, and stopping at the assigned destination. In section 5.1, we discuss why the system is able to perform as well as it did. We then compare its performance with the available state of the art systems and describe how our work relates to them in section 5.2. We finish by listing the current system failures as well as future improvements in section 5.3.

For our setup, we have clearly framed the objective of our system to within a specific type of environments and application and, at this point, we do not know whether this approach generalizes to a completely different scenario (e.g., autonomous cars). However, the service robot task in unconstrained pedestrian environments has quite a broad potential impact onto society. We believe our extensive work will help be an inspiration for many others in continuing the research in autonomous mobile robotics.

5.1 Performance Evaluation

A key factor for our system's success is the hierarchical representation, which effectively separates the localization and navigation tasks. Here, the former is dealt with at the local map level, while the latter is at the global map level. The separation enables the navigation module to independently control the robot, affording the localization module a longer period to perform its computations. This, in fact, is a necessity because it takes seconds for the localization module to compare input landmarks with its large stored database. However, unlike obstacle avoidance and road heading correction, localization-related tasks such as turning at an intersection or stopping at the goal are infrequently invoked and can be updated at a substantial delay. In addition, the infrequent goal location updates do not affect the navigation system on a frame-to-frame basis because it is assumed that the robot wants to go toward the direction of the road.

On the contrary, by having a mechanism that keeps the robot's viewing perspective from or near the center of the road, we do not have to train the vision localization system on landmarks from other perspectives. Such requirements would almost certainly put our recognition system past its limits because we use a regular camera as opposed to omni-directional camera (Murillo et al., 2007; Valgren and Lilienthal, 2008) or creating 3D landmark representation (Trulls et al., 2011). In addition, staying in the middle of the road also allows the robot to better see other semantically important objects, such as stop or exit signs, building entrances, and doors. In the future we would like to have the system be able to navigate through open areas where the road lines are not readily visible, while still be able to optimally view these important cues.

Another important aspect for the success of the system is the implementation of many techniques to account for different processing delays of the vision modules. For the long latency of the localization module, the system implements a multilevel recognition with gist for segment recognition, and landmark recognition to refine the localization hypothesis. In addition, the system also projects forward the results by adding an odometry difference to the localization estimation between the start and end of the landmark recognition process. Also, in the road recognition system, the module runs two processing pipelines: one for the full visual road recognition, while another tracks previously detected road lines to produce updated estimates while the full recognition pipeline looks for new road lines.

In terms of implementation, the overall system uses a number of parameters. For the most part, we have chosen them experimentally during development. We tried several values for each parameter and used the one that produce the best performance. For example, we tried to incorporate the last 100 and 500 absolute headings in the road direction filtering step. We found that the former, which we decided on using in the end, allows for robust filtering while also being responsive enough to detect a change in road direction. On the other hand, the latter seems to adapt to road changes noticeably more slowly. Furthermore, when possible, we and others have previously conducted extensive parametric testing of system components: for attention (Itti and Koch, 2001), gist (Siagian and Itti, 2007), SIFT (Lowe, 2004), road recognition (Siagian et al., 2013a).

To obtain a truly optimal set of parameters, we have to test all possible combinations of all possible values. Unfortunately, the presented system is tightly coupled, where the number of variants to consider when adjusting possible parameters is subject to combinatorial explosion. Nevertheless, we have shown that the system still performs quite robustly with the parameters chosen. We believe that most of these parameters are not extremely sensitive for performance, and there is flexibility in selecting these numbers. In this paper we tried to give an intuition for what the factors are that should be considered, and we provide these values to gauge what is the expected ideal range. We do not claim that these are the best or only possible values. It is possible that better values will be found in future research. However, we report those numbers to allow the readers to reproduce our exact system if they want to. To that end, we also provide the software source code at (Itti, 2012) to port to other robotic systems, or as a guide to adapt our algorithms for specific usage.

5.2 System Comparison

Our evaluation, in terms of scope, is most similar to the Tsukuba challenge (Yuta et al., 2011), which takes its human-sized service robot participants through 1.1km route of pedestrian road in the city of Tsukuba. This is unlike the Darpa grand challenge (Montemerlo et al., 2008) and European Land-Robot Trial (ELROB) (Schneider and Wildermuth, 2011), which are for autonomous cars, or AUVSI's Intelligent Ground Vehicle Competition (IGVC) (Theisen et al., 2011) and NASA's Centennial Challenge (NASA, 2013), which are conducted on open grass fields.

In terms of goals, our system relates to a number of fully autonomous systems that are operating in outdoor pedestrian environments. The ACE project robot (Lidoris et al., 2009) performs outdoor navigation but relies on passersby to guide it to the final destination 1.5km away. The EUROPA project robot (Stachniss, 2013), on the other hand, is able to navigate and localize in unconstrained urban settings using LRF sensors. To the best of our knowledge, we did not find a fully autonomous visually-guided localization and navigation

system that is tested in the pedestrian environment to compare with.

One that comes closest is RatSLAM (Milford and Wyeth, 2008), which takes inspiration from research done in how rat brains perform navigation tasks. It has a fully autonomous version (Milford and Wyeth, 2010), where it simultaneously localizes and maps (SLAM) the environment using vision, and navigates using a planar LRF indoors. Our system, despite not having SLAM yet, performs both navigation and localization outdoors. Another aspect of RatSLAM to note is its visual features. It uses a column average intensity profile taken at specific windows for three different purposes: scene recognition, rotation detection, and speed estimation. This is a stark difference from what most state-of-the-art vision localization systems, including ours, used: computationally complex keypoint-based recognition. Our perceptual system, which relies on SIFT-based salient landmark database matching, is more inline with that of the current SLAM systems: FAB MAP (Cummins and Newman, 2008) or CAT-SLAM (Maddern et al., 2012).

In terms of results, the best way to benchmark our system is by directly comparing our results with others in the same settings. However, this would be a large undertaking. To have a fair comparison, the systems would have to be run on the same robot that they are originally published in, or at least the same configuration of sensors. In addition, there is also an issue of code availability as well as the care needed to port and have it perform optimally. A way to circumvent this is by requesting the authors to run their systems on a recorded data. However, we are dealing with active systems, where different decisions at each step can lead to very different outcomes. Similarly, simulation exercises would also fall short of real-life testing because the robot operates in a dynamic, unpredictable, and not fully accessible (in the A.I. sense) world, which we believe cannot be well captured by the current world models. Given these difficulties, we decided to compare our results with the results of other systems as reported in their respective papers.

Our localization median error of 0.97m is quite high compared to that of outdoor LRF-based systems such as (Montella et al., 2012), which reports error near 10 cm. This is because proximity data is much more discriminating when there is sufficient amount of valid laser readings, which is not always the case outdoors. Visual features, on the other hand, observe the appearance of the environment. They help the system to rely less on nearby surfaces by extending its perceptual range. Because of this we believe our vision-based navigation and localization system will complement laser-based systems.

The use of geographical distance from ground truth as localization metric in our testing is actually different than how vision-based SLAM systems are currently evaluated. They use precision-recall curve, which focuses more on the recognition step. Here precision is the number of correct landmark matches over the number of matches, while recall is the number of correct matches over the number of possible matches. Our recall rate is close to one landmark per five possible matches in an image (20%). Our precision rate, on the other hand, is close to 100% because of the high threshold on positive matches, at least 3 keypoints in a proper affine configuration. Note that almost perfect precision rate is the norm for keypoint-based systems. Thus, precision is usually set at 99% to 100% before evaluating the recall rate. From (Maddern et al., 2012), we gauge our recall rate is competitive as anything above 10% is respectable.

Recognition rate is also the metric of choice for visual place recognition systems. As (Siagian and Itti, 2008a) and a number of others (Song and Tao, 2010; Fazl-Ersi and Tsotsos, 2012) have shown, the performance of our gist-based system, which is about 80% for the standard USC dataset, is only 10% lower than some of the more recent systems. In a sense, place recognition is a different classification problem than that of visual SLAM systems. The former has to discriminate a smaller number of classes, but with more in-class variance, while the latter has to discriminate many more classes, with fewer number of views within class. We believe that, to build a vision localization system that can perform robustly in many environments, a system has to be able to distinguish places at different levels, from the general place names to the specific locations.

For navigation, there are a number of metrics that are currently used for evaluation. One is the success rate in reaching an assigned goal. Although localization is also involved, it is clear to us that the main difficulty lies in path creation and execution. Even if the path created is perfect, if the robot is not able to properly execute the commands, it can still hit an obstacle. Thus, systems are also evaluated on the

severity of the failure, whether it is a square hit or just a graze (Marder-Eppstein et al., 2010). Our system, aside from a number of isolated incidents, usually provides about 30cm of space between its body and the obstacles. In addition, we can also consider execution speed, which is currently less emphasized. The goal for service robots, which are meant to help people, such as ours is to at least match that of an average person (about 1.4m/s). Our average velocity is currently half (0.66m/s) that and is competitive with most currently available systems. It should be noted that robot speed also depends on computational platform and can be increased by simply upgrading them.

The success of a system is also influenced by the travel distance, the longer the higher the likelihood of failure. For shorter distances of 40 to 50 m, it is not unusual to achieve near 100% success rate (Trulls et al., 2011). Our system is asked to reach a goal of 400m or more in busy outdoor environments. Out of six fully autonomous runs, we have to intervene in two of them, all of which are more than 200m from the start. As a comparison, there are a number of indoor systems that have achieved much longer autonomy: lasting a full 26.2 mile marathon (Marder-Eppstein et al., 2010) and 40km over 37 hours (Milford and Wyeth, 2010).

Because a large part of failure to avoid an obstacle originates in the detection step, another way of evaluating is based on the produced obstacle avoidance map. This way is often seen in systems (Maye et al., 2012) that try to detect negative obstacles (e.g., a dip next to a sidewalk). Given the large area which our system is operating in, it would be time-consuming to create a complete ground truth occupancy map. We are, however, able to qualitatively assess our system and produce a list of events in which it fails, in section 5.3.

Our lateral deviation navigation metric delves deeper into the quality of robot traversal because we believe the middle of the road allows the robot to be in position to perform more tasks, if called upon, along the way. Our road recognition system has been shown to be statistically superior in detecting road centers in images with a number of other state-of-the-art systems (Kong et al., 2010; Rasmussen et al., 2009; Chang et al., 2012) with an accuracy level of near 40cm. However, the system performs worse in our robot long-range testing, with a median of 1.37m. Thus, it is hard to predict how this, or other image-based metrics (Rasmussen et al., 2009; Miksik, 2012) that measure road segmentation overlap, would translate to actual autonomous navigation. As such, it is important to create a testing regimen where all the systems are tested on a robot, under the same conditions.

5.3 Failure Cases and Future Improvements

During testing we observe a number of aspects that we would like to improve upon.

Automatic localization initialization. The current localization system has to be given an initial location to guarantee correct convergence. It uses the current location to order the database landmarks when comparing them with the input regions. In addition, the system also employs a search cutoff in the comparison process for efficiency reasons. Thus, instead of going through the whole database, there are times when the matching landmark may not even get to be compared with the input region because of an incorrect prior. Another related issue is the need to fine tune the matching criteria to maximize matches while keeping false positives to a minimum. In addition, when the robot is lost, it should be more active in searching for better views, where the landmarks are more likely to be matched.

Simultaneous Localization & Mapping. Currently, the localization map is still manually constructed. We always understood that this is inconvenient, although, for many application scenarios (see figure 1), we believe this step is well worth the effort because the robot will be used many times in the same environment. In the next iteration, we plan to create the metric topological map automatically using a Simultaneous Localization and Mapping (SLAM) algorithm. There are a number of SLAM algorithms that aim to create such hierarchical map. A system by (Marder-Eppstein et al., 2011) connects local overlapping LRF-based occupancy maps to a single global topological map. Similarly (Pradeep et al., 2010) constructs a topological map by associating stereo-based local submaps. Note that these systems have only been tested indoors, and it may not be trivial to adapt them to outdoor environments. For our system, we are particularly

interested in creating a map with meaningful nodes that represent important geographical points, similar to (Ranganathan and Dellaert, 2011), where it identifies unique LRF signatures produced by junctions in indoor environments. The key is to robustly do the same in unconstrained pedestrian outdoor environments, where the presence of surrounding walls cannot be assumed. One way is by using an online gist feature grouping technique (Valgren and Lilienthal, 2008), which detects a large change in the feature space, corresponding to moving into a new segment. In addition, related to metric topological SLAM, there is also a research problem (Beeson et al., 2010; Kuipers et al., 2004) of how to topologically represent more complex path or space such as winding roads or open area.

Proper Turning at an Intersection. Although the localization system did not completely fail or misplace the robot during testing, at times, it converged a few meters away from the true location. This becomes an issue when the robot is near an intersection and needs to turn. If the turn occurs far away from the center of the intersection, the robot may end up on the shoulder instead of near the center of the next road. One way to solve this problem is by explicitly recognizing the junction or intersection. Given a prior that the robot is near an intersection, the system can look for road lines perpendicular to the lines that are pointing toward the current robot direction. Another way to avoid traveling on the side of the road is by recovering the true road center established during training. The visual landmarks in the stored database also carry image coordinate locations and can be used to adjust the target lateral position. This approach is similar to teach-and-replay (Chang et al., 2010), but different in that the result is not directly coupled with motor command generation. It merely corrects for the road center estimation whenever the matches are available, preventing the system from having to stop the robot when recognition failed. This technique is also useful for navigating when the vanishing point is undetected, as well as on more extreme curved and winding roads, improving upon the system's current ability to follow the more gradually changing roads (Siagian et al., 2013a).

Robust Obstacle Avoidance Navigation. In terms of navigational shortcomings, there are a few times when the robot failed to detect obstacles in its path, particularly, thin or short objects, as well as objects with holes. In addition, there are also a number of scenarios that we did not include in our test, but are expected to currently lead to system failures and need to be handled in the future: curbs, ramps, off-roads. To improve the system competencies in these situations, we are adding an up-down sweeping LRF, as well as applying techniques by (Klasing et al., 2008; Maye et al., 2012). We believe a moving LRF is the best way to maximize obstacle visibility, while the algorithms enable our system to handle those situations more robustly. In addition, we also plan to incorporate the evidence from the sweeping LRF with our visual road recognition module by identifying which road lines are solid boundaries and should never be crossed. Here, the road boundary can be noted by visual appearance change, change in proximity profile, or a sharp change in elevation. Once we incorporate these improvements, we believe the resulting system will be robust in many different outdoor scenarios, sufficient to travel for even longer distances.

Better Road Detection. The system also needs to improve on detecting roads in more difficult conditions, such as bright sunlight. This can be done in both hardware and software. For the former, a camera with wider dynamic range would be a welcome addition. In addition, having a lens with wider field-of-view is also helpful, especially for the wider uniform roads with no direction cues between the boundaries. For software improvements, a better low-level edge and line segment detector (with local contrast adjustments) can significantly boost performance. A different way to improve the system is by utilizing visual cues other than road lines. We are planning to incorporate the use of histogram differencing to segment out the road from its flanking areas, much like (Chang et al., 2012). By creating a system that runs a number of contrasting approaches in parallel, we hope to enable the system to robustly detect more road types, even in more naturalistic off-road environments. Related to this, the localization system, particularly our biologically-inspired gist classification and saliency, have been shown to perform well on non-urban environments (Siagian and Itti, 2009; Siagian and Itti, 2007; Itti and Baldi, 2009).

Improved Navigation in Open Areas. In situations where there is no road in front of the robot, the system simply maintains its current forward heading set by the last seen road, while trying to avoid obstacles. This goes on until there is a continuing road or the localization module instructs to make a turn. In these

situations, the system performs navigation exclusively using LRF, producing behaviors such as wall-following or centering along a hallway. Our robot in a sense follows a subsumption architecture (Brooks, 1986), i.e., when a road is detected, follow it, otherwise default to driving straight forward. Thus the navigation system is not dependent on the presence of the vanishing point. On the contrary, it only uses visual road recognition when it is advantageous. This way the road recognition system enhances the standard laser-based navigation system largely utilized today. In pedestrian environments, however, there are open areas such as a square or a large hall, where there are no nearby wall surfaces to guide the robot. In the experiment, the system was able to navigate through an open water fountain area (observe figure 28). However, this success, in large part, is because the continuing road is aligned with the previous road. In less ideal configurations, the system can be confused and drive to the wrong road. To rectify this problem, the system must have a richer understanding of the space, knowing all the roads that come in and out of the area.

Navigation in Crowded Areas. As for the effect of crowding to our system, we are encouraged by the results that show that it is able to localize and navigate through those difficult conditions. In the future we would like improve on Beobot's navigation skills in moving more naturally among people (Trautman and Krause, 2010), for example using a fast GPU-based pedestrian recognition system (Benenson et al., 2012) to properly anticipate where they are headed. We can also try to characterize how pedestrian behave as a group (Henry et al., 2010), which would be more feasible for denser crowds.

Acknowledgments

This work was supported by the National Science Foundation (grant numbers CCF-1317433 and CMMI-1235539), the Army Research Office (W911NF-11-1-0046 and W911NF-12-1-0433), 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

Agrawal, M. and Konolige, K. (2006). Real-time localization in outdoor environments using stereo vision and inexpensive gps. In *Proc. International Conference on Pattern Recognition (ICPR)*, volume 3, pages 1063–1068.

Angeli, A., Filliat, D., Doncieux, S., and Meyer, J.-A. (2008). A fast and incremental method for loop-closure detection using bags of visual words. *IEEE Transactions On Robotics, Special Issue on Visual SLAM*, 24(5):1027–1037.

Bay, H., Tuytelaars, T., and Gool, L. V. (2006). Surf: Speeded up robust features. In *Proc. European Conference on Computer Vision (ECCV)*, pages 404–417.

Beeson, P., Modayil, J., and Kuipers, B. (2010). Factoring the mapping problem: Mobile robot map-building in the hybrid spatial semantic hierarchy. 29:428–459.

Benenson, R., Mathias, M., Timofte, R., and Gool, L. V. (2012). Pedestrian detection at 100 frames per second. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 290–2910, Providence, RI, USA. IEEE Computer Society.

Borenstein, J. and Koren, Y. (1991). The vector field histogram- fast obstacle avoidance for mobile robots. *IEEE Journal of Robotics and Automation*, 7(3):278–288.

Bradski, G. (2001). Open source computer vision library.

Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Trans. Robotics and Automation*, 2(1):14–23.

Chang, C.-K., Siagian, C., and Itti, L. (2010). Mobile robot vision navigation & localization using gist and saliency. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4147–4154. Both first authors contributed equally.

Chang, C.-K., Siagian, C., and Itti, L. (2012). Mobile robot monocular vision navigation based on road region and boundary estimation. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1043–1050.

Chen, Z. and Birchfield, S. (2006). Quantitative vision-based mobile robot navigation. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 2686 – 2692.

Cherubini, A., Spindler, F., and Chaumette, F. (2012). A new tentacles-based technique for avoiding obstacles during visual navigation. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 4850–4855.

Coombs, D., Herman, M., Hong, T., and Nashman, M. (1995). Real-time obstacle avoidance using central flow divergence and peripheral flow. In *Proc. International Conference on Computer Vision (ICCV)*, pages 276–283.

Cummins, M. and Newman, P. (2008). Fab-map: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research (IJRR)*, 27(6):647–665.

Cunha, J., Pedrosa, E., Cruz, C., Neves, A., and N.Lau (2011). Using a depth camera for indoor robot localization and navigation. In *RGB-D Robotics: Science and Systems (RSS) Workshop*, Los Angeles, California.

Eade, E. and Drummond, T. (2008). Unified loop closing and recovery for real time monocular slam. In *Proc. British Machine Vision Conference (BMVC)*, pages 6.1 – 6.10.

Fazl-Ersi, E. and Tsotsos, J. (2012). Histogram of oriented uniform patterns for robust place recognition and categorization. *The International Journal of Robotics Research (IJRR)*, 31(2):468–483.

Fiorini, P. and Shiller, Z. (1998). Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research (IJRR)*, 17(7):760–772.

Fox, D., Burgard, W., Dellaert, F., and Thrun, S. (1999). Monte carlo localization: Efficient position estimation for mobile robots. In *Proc. of Sixteenth National Conference on Artificial Intelligence (AAAI'99).*

Fox, D., Burgard, W., and Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23 – 33.

Furgale, P. and Barfoot, T. D. (2010). Visual teach and repeat for long-range rover autonomy. *Journal of Field Robotics*, 27(5):534 – 560.

Glennie, C. and Lichti, D. D. (2010). Static calibration and analysis of the velodyne hdl-64e s2 for high accuracy mobile scanning. *Remote Sensing*, 2(6):1610–1624.

Henry, P., Vollmer, C., Ferris, B., and Fox, D. (2010). Learning to navigate through crowded environments. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 981–986.

Hrabar, S. and Sukhatme, G. (2009). Vision-based navigation through urban canyons. *Journal of Field Robotics*, 26(5):431 – 452.

Hu, Z., Wang, H., Zhang, L., and Yu, H. (2009). Laser based sensor boundary recognition of mobile robot. In *International Conference on Networking, Sensing and Control*, Okayama, Japan.

i Badia, S. B., Pyk, P., and Verschure, P. F. M. J. (2007). International journal of robotics research. *A Fly-Locust Based Neuronal Control System Applied to an Unmanned Aerial Vehicle: The Invertebrate Neuronal Principles for Course Stabilization, Altitude Control and Collision Avoidance*, 26(7):759 772.

Itti, L. (2000). *Models of Bottom-Up and Top-Down Visual Attention*. bu;td;mod;psy;cv, Pasadena, California.

Itti, L. (2012). iLab Neuromorphic Vision C++ Toolkit (iNVT). `http://ilab.usc.edu/toolkit/`. Accessed: December 15, 2012.

Itti, L. and Baldi, P. F. (2009). Bayesian surprise attracts human attention. *Vision Research*, 49(10):1295–1306. Top cited article 2008-2010 award from Vision Research.

Itti, L. and Koch, C. (2001). Feature combination strategies for saliency-based visual attention systems. *Journal of Electronic Imaging*, 10(1):161–169.

Itti, L., Koch, C., and Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259.

King, P. H. (2008). A Low Cost Localization Solution Using a Kalman Filter for Data Fusion. Master's thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA.

Klasing, K., Lidoris, G., Bauer, A., Rohrmuller, F., Wollherr, D., and Buss, M. (2008). The autonomous city explorer: Towards semantic navigation in urban environments. In *Proc. International Workshop on Cognition For Technical Systems (COTESYS)*.

Koenig, S. and Likhachev, M. (2002). D* lite. In *AAAI Conference of Artificial Intelligence (AAAI)*, pages 476–483.

Koenig, S., Likhachev, M., Liu, Y., and Furcy, D. (2004). Incremental heuristic search in artificial intelligence. *Artificial Intelligence Magazine*, 25(2):99 – 112.

Kong, H., Audibert, J.-Y., and Ponce, J. (2010). General road detection from a single image. *IEEE Transactions on Image Processing*, 19(8):2211 – 2220.

Kuffner, J. J. and LaValle, S. M. (2000). Rrt-connect: An efficient approach to single-query path planning. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 995–1001.

Kuhnl, T., Kummert, F., and Fritsch, J. (2011). Monocular road segmentation using slow feature analysis. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 800 – 806.

Kuipers, B., Modayil, J., Beeson, P., Macmahon, M., and Savelli, F. (2004). Local metrical and global topological maps in the hybrid spatial semantic hierarchy. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 4845–4851.

Lazebnik, S., Schmid, C., and Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2169–2178, Washington, DC, USA. IEEE Computer Society.

Lidoris, G., Rohrmullera, F., Wollherr, D., and Buss, M. (2009). he autonomous city explorer (ace) project mobile robot navigation in highly populated urban environments. *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 1416–1422.

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *Intl. Journal of Computer Vision*, 60(2):91–110.

Maddern, W., Milford, M., and Wyeth, G. (2012). Cat-slam: probabilistic localisation and mapping using a continuous appearance-based trajectory. *The International Journal of Robotics Research (IJRR)*, 31:429–451.

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

Marder-Eppstein, E., Berger, E., Foote, T., Gerkey, B., and Konolige, K. (2011). Kurt konolige, eitan marder-eppstein, bhaskara marthi. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 3041 – 3047.

Maye, J., Kaestner, R., and Siegwart, R. (2012). Curb detection for a pedestrian robot in urban environments. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 367–373.

Michels, J., Saxena, A., and Ng, A. Y. (2005). High speed obstacle avoidance using monocular vision and reinforcement learning. In *International Conference on Machine Learning*.

MicroStrain, Inc. (2009). 3DM-GX2:: MicroStrain, AHRS Orientation Sensor. `http://www.microstrain.com/3dm-gx2.aspx`. Accessed: July 15, 2009.

Miksik, O. (2012). Rapid vanishing point estimation for general road detection. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*.

Milford, M. and Wyeth, G. (2008). Mapping a suburb with a single camera using a biologically inspired slam system. *IEEE Transactions on Robotics*, 24(5):1038–1053.

Milford, M. and Wyeth, G. (2010). Persistent navigation and mapping using a biologically inspired slam system. *The International Journal of Robotics Research (IJRR)*, 29:1131–1153.

Minguez, J. and Montano, L. (2004). Nearness diagram (nd) navigation: Collision avoidance in troublesome scenario. *IEEE Transactions on Robotics and Automation*, 20(1):45 – 59.

Moghamadam, P., Starzyk, J. A., and Wijesoma, W. S. (2012). Fast vanishing-point detection in unstructured environments. *IEEE Transactions on Image Processing*, 21(1):425 – 430.

Montella, C., Perkins, T., Spletzer, J., and Sands, M. (2012). To the bookstore! autonomous wheelchair navigation in an urban environment. In *Proc. International Conference on Field and Service Robotics (FSR)*, Matsushima, Japan.

Montemerlo, M., Becker, J., Bhat, S., Dahlkamp, H., Dolgov, D., Ettinger, S., Haehnel, D., Hilden, T., Hoffmann, G., Huhnke, B., Johnston, D., Klumpp, S., Langer, D., , Levandowski, A., Levinson, J., Marcil, J., Orenstein, D., Paefgen, J., Penny, I., Petrovskaya, A., Pflueger, M., Stanek, G., Stavens, D., Vogt, A., and Thrun, S. (2008). Junior: The stanford entry in the urban challenge. *Journal Field Robotics*, 25(9):569–597.

Moravec, H. and Elfes, A. (1985). High resolution maps from wide angle sonar. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 116–121.

Murillo, A., Guerrero, J., and Sagues, C. (2007). Surf features for efficient robot localization with omnidirectional images. In *Proc. International Conference on Robotics and Automation (ICRA)*, pages 3901–3907.

Nakamura, T. and Asada, M. (1995). Motion sketch: Acquisition of visual motion guided behaviors. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 126 – 132.

NASA (2013). Centennial challenges. http://www.nasa.gov/offices/oct/stp/centennial_challenges/index.html. Accessed: January 15, 2013.

Newman, P., Sibley, G., Smith, M., Cummins, M., Harrison, A., Mei, C., Posner, I., Shade, R., Schroeter, D., Murphy, L., Churchill, W., Cole, D., and Reid, I. (2009). Navigating, recognizing and describing urban spaces with vision and lasers. *The International Journal of Robotics Research (IJRR)*, 28(11 - 12):1406–1433.

Nistér, D., Naroditsky, O., and Bergen, J. (2006). Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23(1):3 – 20.

Ouerhani, N., Bur, A., and Hugli, H. (2005). Visual attention-based robot self-localization. In *Proc. European Conference on Mobile Robotics (ECMR)*, pages 8–13, Ancona, Italy.

Pradeep, V., Medioni, G., and Weiland, J. (2010). Robot vision for the visually impaired. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15 – 22. IEEE Computer Society.

Pronobis, A., Caputo, B., Jensfelt, P., and Christensen, H. (2006). A discriminative approach to robust visual place recognition. In *Proc. IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 3829–3836.

Pronobis, A., Mozos, O. M., Caputo, B., and Jensfelt, P. (2010). Multi-modal semantic place classification. *The International Journal of Robotics Research (IJRR), Special Issue on Robotic Vision*, 29(2-3):298–320.

Quinlan, S. and Khatib, O. (1993). Elastic bands: Connecting path planning and control. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 802–807.

Ranganathan, A. and Dellaert, F. (2011). Online probabilistic topological mapping. In *The International Journal of Robotics Research (IJRR)*, volume 30, pages 755 – 771.

Rasmussen, C., Lu, Y., and Kocamaz, M. (2009). Appearance contrast for fast, robust trail-following. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3505 – 3512.

Royer, E., Lhuillier, M., Dhome, M., and Lavest, J.-M. (2007). Monocular vision for mobile robot localization and autonomous navigation. *International Journal of Computer Vision*, 74(3):237 – 260.

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

Santos-Victor, J., Sandini, G., Curotto, F., and Garibaldi, S. (1993). Divergent stereo in autonomous navigation: Learning from bees. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 434 –439.

Schindler, G., Brown, M., and Szeliski, R. (2007). City-scale location recognition. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–7, Los Alamitos, CA, USA. IEEE Computer Society.

Schneider, F. E. and Wildermuth, D. (2011). Results of the european land robot trial and their usability for benchmarking outdoor robot systems. In *Toward Autonomous Robotic Systems*, volume 6856 of *Lecture Notes in Computer Science*, pages 408–409. Springer.

Se, S., Lowe, D. G., and Little, J. J. (2005). Vision-based global localization and mapping for mobile robots. *IEEE Transactions on Robotics*, 21(3):364–375.

Segvic, S., Remazeilles, A., Diosi, A., and Chaumette, F. (2009). A mapping and localization framework for scalable appearance-based navigation. *Computer Vision and Image Understanding*, 113(2):172 – 187.

Siagian, C., Chang, C., and Itti, L. (2013a). Mobile robot navigation system in outdoor pedestrian environment using vision-based road recognition. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*. Both first authors contributed equally.

Siagian, C., Chang, C. K., and Itti, L. (2013b). Beobot 2.0. `http://ilab.usc.edu/beobot2`. Accessed: December 15, 2012.

Siagian, C., Chang, C.-K., Voorhies, R., and Itti, L. (2011). Beobot 2.0: Cluster architecture for mobile robotics. *Journal of Field Robotics*, 28(2):278–302.

Siagian, C. and Itti, L. (2007). Rapid biologically-inspired scene classification using features shared with visual attention. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(2):300–312.

Siagian, C. and Itti, L. (2008a). Comparison of gist models in rapid scene categorization tasks. In *Proc. Vision Science Society Annual Meeting (VSS08)*.

Siagian, C. and Itti, L. (2008b). Storing and recalling information for vision localization. In *IEEE International Conference on Robotics and Automation (ICRA), Pasadena, California*, pages 1848–1855.

Siagian, C. and Itti, L. (2009). Biologically inspired mobile robot vision localization. *IEEE Transactions on Robotics*, 25(4):861–873.

Simmons, R. (1996). The curvature-velocity method for local obstacle avoidance. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 3375–3382.

Song, D. and Tao, D. (2010). Biologically inspired feature manifold for scene classification. *IEEE Transactions on Image Processing*, 19(1):174 – 184.

Stachniss, C. (2013). Europa. `http://europa.informatik.uni-freiburg.de/`. Accessed: June 30, 2013.

Strasdat, H., Montiel, J. M. M., and Davison, A. J. (2010). Scale drift-aware large scale monocular slam. In *Robotics: Science and Systems (RSS)*, volume 2.

Tardif, J.-P., Pavlidis, Y., and Daniilidis, K. (2008). Monocular visual odometry in urban environments using an omnidirectional camera. In *Proc. IEEE/RSJ International Conference on Intelligence Robotics and Systems (IROS)*, pages 2531 – 2538.

Theisen, B. L., Frederick, P., and Smuda, W. (2011). The 18th annual intelligent ground vehicle competition: trends and influences for intelligent ground vehicle control. In *Proc. SPIE 7878, Intelligent Robots and Computer Vision XXVIII: Algorithms and Techniques*.

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

Thrun, S., Bennewitz, M., Burgard, W., Cremers, A., Dellaert, F., Fox, D., Hähnel, D., Rosenberg, C., Roy, N., Schulte, J., and Schulz, D. (1999). MINERVA: A second generation mobile tour-guide robot. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*.

Thrun, S., Fox, D., Burgard, W., and Dellaert, F. (2000). Robust monte-carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2):99–141.

Torralba, A., Murphy, K. P., Freeman, W. T., and Rubin, M. A. (2003). Context-based vision system for place and object recognition. In *Proc. International Conference on Computer Vision (ICCV)*, pages 1023 – 1029, Nice, France.

Trautman, P. and Krause, A. (2010). Unfreezing the robot: Navigation in dense, interacting crowds. In *Proc. IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 797–803.

Trulls, E., Murtra, A. C., Pérez-Ibarz, J., Ferrer, G., Vasquez, D., Mirats-Tur, J. M., and Sanfeliu, A. (2011). Autonomous navigation for mobile service robots in urban pedestrian environments. *Journal of Field Robotics*, 28(3):329 – 354.

Ulrich, I. and Borenstein, J. (1998). Vfh+: Reliable obstacle avoidance for fast mobile robots. *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 1572–1577.

Ulrich, I. and Nourbakhsh, I. (2000). Appearance-based place recognition for topological localization. In *Proc. IEEE International Conference on Robotics and Automation (ICRA*, pages 1023 – 1029.

Valgren, C. and Lilienthal, A. J. (2008). Incremental spectral clustering and seasons: Appearance-based localization in outdoor environments. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 1856–1861, Pasadena, CA.

Vargas, G. R. A., Nagatani, K., and Yoshida, K. (2007). Adaptive kalman filtering for gps-based mobile robot localization. In *IEEE International Workshop on Safety, Security and Rescue Robotics*, Rome, Italy.

Wang, J., Zha, H., and Cipolla, R. (2006). Coarse-to-fine vision-based localization by indexing scale-invariant features. *IEEE Trans. Systems, Man and Cybernetics*, 36(2):413–422.

Weiss, C., Tamimi, H., Masselli, A., and Zell, A. (2007). A hybrid approach for vision-based outdoor robot localization using global and local image features. In *Proc. IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 1047 – 1052.

Wilson, N. J. (1982). In *Principles of Artificial Intelligence*. Springer Verlag, Berlin, Germany.

Wurm, K., Kummerle, R., Stachniss, C., and Burgard, W. (2009). Improving robot navigation in structured outdoor environmets by identifying vegetation from laser data. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1217 –1222.

Wurm, K. M., Hornung, A., Bennewitz, M., Stachniss, C., and Burgard, W. (2010). Octomap: A probabilistic, flexible, and compact 3d map representation for robotic systems. In *ICRA Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, pages 300 – 307.

Yang, L. and Lavalle, S. M. (2004). The sampling-based neighborhood graph: An approach to computing and executing feedback motion strategies. *IEEE Transactions on Robotics and Automation*, 20(3):419 – 432.

Yuta, S., Mizukawa, M., Hashimoto, H., Tashiro, H., and Okubo, T. (2011). Tsukuba challenge 2009 - towards robots working in the real world: Records in 2009. *Journal of Robotics and Mechatronics*, 23(2):201–206.

Zhang, W. and Kosecka, J. (2005). Localization based on building recognition. In *IEEE Workshop on Applications for Visually Impaired*, pages 21 – 28.