Mobile Robot Vision Navigation & Localization Using Gist and Saliency

Chin-Kai Chang* Christian Siagian* Laurent Itti

***Abstract*— We present a vision-based navigation and localization system using two biologically-inspired scene understanding models which are studied from human visual capabilities: (1) Gist model which captures the holistic characteristics and layout of an image and (2) Saliency model which emulates the visual attention of primates to identify conspicuous regions in the image. Here the localization system utilizes the gist features and salient regions to accurately localize the robot, while the navigation system uses the salient regions to perform visual feedback control to direct its heading and go to a user-provided goal location. We tested the system on our robot, Beobot2.0, in an indoor and outdoor environment with a route length of 36.67m (10,890 video frames) and 138.27m (28,971 frames), respectively. On average, the robot is able to drive within 3.68cm and 8.78cm (respectively) of the center of the lane.**

I. INTRODUCTION

Ability to go to specific locations in one's environment in a timely manner is fundamental in creating a fully autonomous mobile robotic system. To do so, a robot not only has to be able to move about its environment, but it also has to identify its location. To solve this problem, available approaches use proximity sensors such as Laser Range Finder (LRF), cameras, encoders, and Global Positioning System (GPS). While robot navigation and localization have made strides indoors (using LRF [1], [2]) and on the road (like in DARPA Grand Challenge, relying on a combination of LRF, GPS, and visual road recognition [3], [4]), systems that can perform well in the in-between settings such as a college campus, where a robot needs to travel both inside buildings (through hallways) and outside (between large structures), are still lagging behind. For these scenarios, vision, human primary perception, is ideal because of its versatility in recognizing a variety of environments. However, because of the complexity of a camera image, vision also comes with many challenges such as lighting conditions, perspective, and occlusions. Observe figure 1 for an example of a cluttered outdoor environment that our robot Beobot2.0 is operating in.

In the current state of robotic research, vision-based navigation and localization are usually dealt with separately. For the most part, available literature concentrates more on one but not the other. Researchers who work on vision localization manually control their robots to move about the environment. On the other hand, visual navigation is usually performed for a single predefined route, where robots do not have to decide on where to turn when encountering an intersection. This split in effort, is done for good reasons, in that each individual problem is difficult by itself.

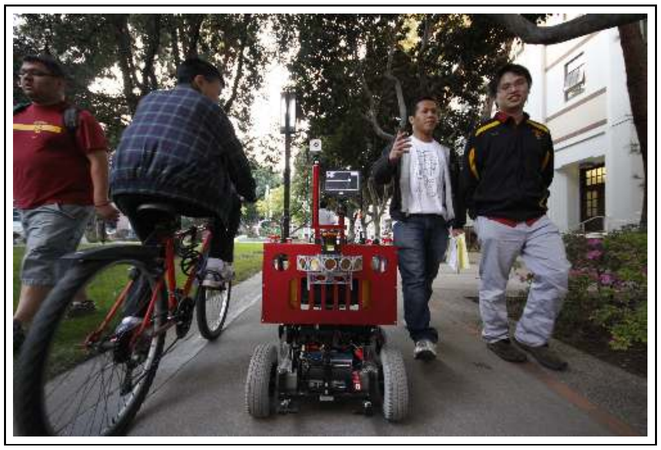

Fig. 1. Beobot2.0 navigating in a cluttered environment.

Although robust vision localization that can operate in all environment are not yet available, state of the art systems can now localize in large areas such as city streets [5], [6], across multiple seasons [7], and in many different indoor [8], [9], [10] and outdoor environments [11], [12], [13]. However, most localization systems are trained and tested while the robot's movement is controlled by a user. The problem with manual driving is that the robot's point of view comes from a perfect control. When one wants to integrate the localization module with the rest of the system, including navigation, the robot may need to recognize views it has not seen. For example, the views from the same location with different robot headings, can be strikingly dissimilar. One way to alleviate this issue is through the use of omni-directional cameras (as opposed to a regular straight-forward cameras) [14] as the images produced can be realigned.

On the navigation side, there are two basic types of systems. One uses a ”teach-and-replay” paradigm, and the other tries to recognize the direction of the road. In the former [15], [16], [17], the robot first is manually driven through a pre-specified path, while recording all the pertinent information necessary to replay the route being taught. Then, in the replay phase, the robot compares the stored visual features with what is currently seen to compute the positional difference, which is used as a control feedback to stay on the path. This type of system, which utilizes features such as KLT [18] or SIFT [19], needs to recognize surrounding landmarks to navigate. In other words, localization within

* equal authorship.

C. Chang and C. Siagian are with Department of Computer Science, University of Southern California, Hedco Neuroscience Building - Room 10, 3641 Watt Way, Los Angeles, California, 90089-2520, USA. (chinkaic,siagian)@usc.edu

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

the path (not throughout the environment) is required. This is done by retracing a sequence of keyframes, keeping track of which frame the robot currently is at. When matching between the current scene and keyframe fails, the system has to re-synchronize itself. This can take extra time as rematching from the start of the series is required.

On the other hand, road recognition systems, instead of identifying landmarks, try to recognize the road's appearance as well as boundaries. Some systems rely on recognizing road edges (based on the intensity of images, for example) that separate the road from its surroundings [20]. Others do so by performing color and texture segmentation [21], [22]. In addition, there are systems that combine both techniques [23], [24], [25] to extract the road region in the image. However, for these type of systems, the robot merely follows the road and keeps itself within its boundaries, not knowing where it will ultimately end up.

If a system has both localization and navigation, we can command it to go to any goal location in the environment by using the former to compute a route and the latter to execute movements. In addition, such an integrated system can internally share resources (visual features, for one) to make the overall system run more effectively. However, this also poses a challenging problem of integrating complex subsystems. In spite of the difficulties, many vision localization [26] and navigation [15] systems have discussed that it is advantageous to combine localization and navigation. Ours is such an implementation.

We present a mobile robot vision system that both localizes and navigates by extending our previous work of Gist-Saliency localization [11]. Instead of just navigating through a pre-specified path (like teach-and-replay), the robot can execute movements through any desired routes in the environment (e.g., routes generated by a planner). On the other hand, unlike teach-and-replay, a stand-alone localization system allows the navigation system to not be overly sensitive to the temporal order of input images. We tested the system (reported in section III) in both indoor and outdoor environments, each recorded multiple times to validate our approach.

II. DESIGN AND IMPLEMENTATIONS

Our presented mobile robot vision navigation and localization system (illustrated in figure 2) utilizes biologically inspired features: gist [27] and salient regions [11] of an image, obtained from a single straight-forward view camera (as opposed to omni-directional). Both features are computed in parallel, utilizing shared raw Visual Cortex visual features from the color, intensity and orientation domain [28], [29]. Here we define gist as a holistic and concise feature vector representation of a scene, acquired over very short amount of time. Salient regions, on the other hand, are obtained using the saliency model [28], [29] which identifies parts of an image that readily capture the viewer's attention. As demonstrated by previous testings [11], the likelihood that the same regions will pop out again when the robot revisits particular locations is quite high. The actual matching is done through SIFT keypoints extracted within each salient region. It is important to note that matching keypoints within small region windows instead of a whole image makes the process much more efficient.

Our localization system [11], [30] estimates the robot's location in an environment represented by a topological map. The map denotes paths as edges, and intersections as nodes. We also define the term segment as an ordered list of edges with one edge connected to the next to form a continuous path (an example is the selected three-edge segment, highlighted as a thicker green line, in the map in figure 2). Geographically speaking, a segment is usually a portion of a hallway, path, or road interrupted by a crossing or a physical barrier at both ends for a natural delineation. This grouping is motivated by the fact that views/layouts in one path segment are coarsely similar. Because of this, the system can use the gist vector to classify an input image to a segment.

After the system estimates the robot's segment location, we then try to refine the coarse localization estimation by matching the salient regions found in the input image with the stored regions from our landmark database (obtained during training). Note that this process is much slower than segment estimation but produces a higher localization resolution. At the end, both gist and salient region observations are incorporated in a back-end probabilistic Monte Carlo Localization to produce the most likely current location.

In our training procedure, we run the robot through the environment once. There is no camera calibration nor manual selection of the landmarks. In addition, the training procedure [30] automatically clusters salient regions that portray the same landmark. And, thus, a landmark, a real entity in the environment, is established by a set of salient regions. In a sense, the regions become evidences that the landmark exists. Furthermore, these regions are stored in the order of discovery. That is, we keep a temporal order of the regions as seen when the robot traverses an environment. This becomes important when we try to navigate using these landmarks.

As a result of the compact topological map representation (figure 2), we do not localize the robot laterally on the lane nor do we have an accurate knowledge of the robot's heading with respect to the road. However, for each database salient region, we do not only record its geographical location, but also register its image coordinates to perform navigation. That is, we minimize the difference in image coordinates between the input region and its corresponding database region to bring and keep the robot at the "correct" position on the lane (same position as during training). In the following sub-section II-A, we describe the main contribution of this paper, the navigation portion of the system and its interaction with the localization component.

A. Navigation System

As shown in figure 2, localization and navigation run in parallel in the system. When a new set of input salient regions arrive, the localization module starts its landmark database search (denoted as $1a$ in the figure). At the same time, the

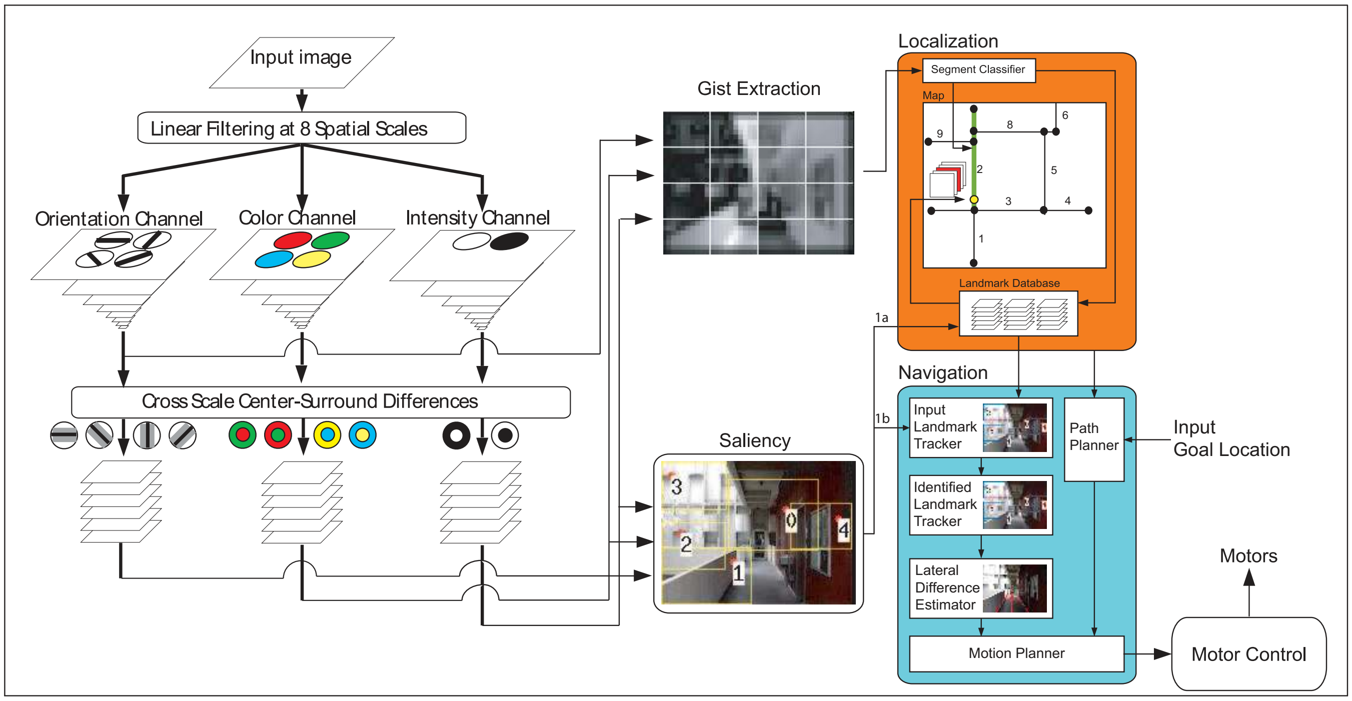

Fig. 2. Diagram of the overall vision navigation and localization system. From an input image the system extracts low-level features consisting of center-surround color, intensity, and orientation that are computed in separate channels. They are then further processed to produce gist features and salient regions. We then compare them with previously obtained environment visual information to estimate the robots location. Furthermore the identified salient regions are used to direct the robot's heading, to navigate itself to its destination.

navigation module also tracks these regions ($1b$). By tracking them, when the regions are positively identified at a later time (usually database search finishes in the order of seconds), even if the robot has moved during the search process, we can still locate the salient regions in the image. When a tracked region is lost or moves away from the robot's field of view, we simply tell the localizer to stop searching for a match. Note that, while the search is in progress, the gist-based segment estimation result, which is available instantaneously on every frame, is used to maintain the location belief.

When the database search concludes, if there is at least one matched region, the localization system informs the navigation system that these regions can be used for navigation. The navigation system first transfers the regions tracked by the input landmark tracker to the identified landmark tracker. A second tracker is needed to free up the first one for newer, yet to be identified landmarks from the current frame. The identified landmark tracker keep track of the transferred regions so that we can utilize them for visual navigation cues for a period of time (until the newer landmarks are recognized).

The navigation system calculates the image coordinate difference (we call this lateral difference) between the tracked region and the matched database region (section II-A.2) as a feedback to the motor controller. In addition, we also bias the salient region matching (explained in section II-A.3) when the robot is at the intersections so that it can correctly direct itself to the goal location.

We start by explaining in details the salient region tracking process in the following sub-section II-A.1.

1) Salient Region Tracker: Figure 3 illustrates the salient region tracking procedure. The tracker utilizes a template matching algorithm (OpenCV implementation with squared-error distance) on conspicuity maps [28], [29] from the 7 Visual Cortex sub-channels used in saliency and gist computation (see Figure 2; the sub-channels correspond to 4 edge orientations, 2 color contrasts, and luminance contrast). For each region, we perform template matching on each map (7 associated templates) before summing the resulting distance map. The templates are initialized using 7x7 windows from each conspicuity map, centered around the salient point of the region. The conspicuity maps are 40x30 pixels in size, downsampled from the original 160x120 input image size, which is acceptable given that we are mostly tracking large salient objects. We then weigh the summation based on the distance from the previous time step for temporal filtering. The coordinate of the minimum value is the predicted new location.

In addition, in each frame, we update the templates for robustness (lighting change, etc) using the following adaptive equation:

$$T_{t,i} \quad = \quad .9 * T_{t-1,i} + .1 * N_{t-1,i} \qquad (1)$$

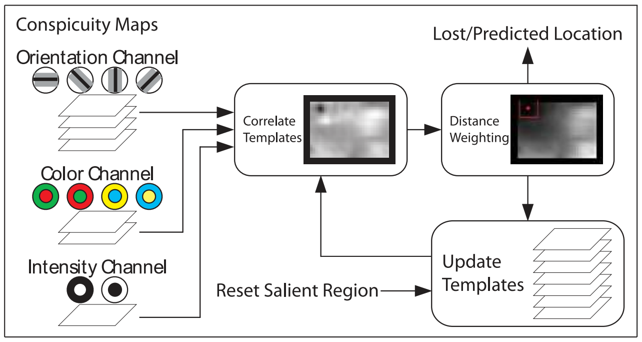

Fig. 3. Diagram for salient region tracking. The system uses conspicuity maps from 7 sub-channels to perform template matching tracking. Before selecting the predicted new location, the system weighs the result with respect to the proximity of the point in the previous time step. In addition, we also adapt the templates overtime for tracking robustness.

Here, $T_{t-1,i}$ is a region's template for sub-channel i at time $t-1$, while $N_{t,i}$ is a new region template around the predicted location. Before the actual update, we added a threshold to check if the resulting template would change drastically; a sign that the tracking may be failing. If this is the case, we do not update, hoping that the error is just coincidental. However, if this occurs three times in a row, we report that the tracking is lost.

This process has proven to be quite robust because, aside from following regions that are already salient, the variety of domains means that for a failure to occur, noise has to corrupt many sub-channels. In addition, the noise is also minimized because the saliency algorithm uses a biologically inspired spatial competition for salience (winner-take-all) [28], [29], where noisy sub-channels with too many peaks are subdued. Also, the advantage of re-using conspicuity maps is that the process is faster. But, more importantly, tracking requires no recall of stored information, which allows the procedure to be scalable in the size of the environment. In the current implementation, the system is usually able to track a salient region for up to 100 frames and the process only takes 2ms per frame for each salient region. On each frame, we track up to 10 regions: 5 for the ones already recognized (in the identified landmark tracker), and 5 more that are currently being matched (in the input landmark tracker).

Before we calculate the lateral difference from the database match result, we perform a forward projection to the current frame to take into account the duration of the matching process. The search outputs a match between an input region from many time steps ago and a region in a database which most closely resembles it. However, since we started the search, that region has undergone transformation as the robot has moved and changed its pose. What we now actually want is a database region that most closely resembles the input region as it appears in the current time step.

Fortunately, in the database, each landmark is represented by a set of views (regions), sorted temporally during training as the robot moves through a path. Thus, the forward projection process is basically a re-matching between the tracked region at the current time frame and a region from the matched database landmark that is usually only a few indexes away from the original match. This procedure takes about 10ms and is illustrated in figure 4.

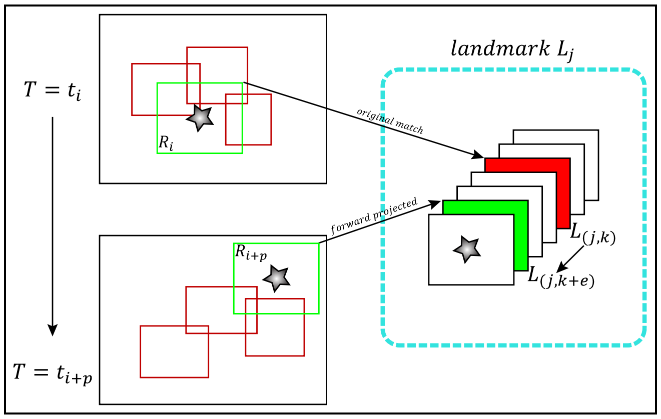

Fig. 4. Forward projection procedure. A detected region R_i from time $T = t_i$ (top left of the figure) is successfully matched to salient region $L_{j,k}$, p frames later. This is the region that belongs to Landmark L_j with index k (denoted in solid red at the right side of the figure), where k is one of the multiple views of the landmark recorded during training. At this current time $T = t_{i+p}$, the region R_i has changed its appearance as the robot moves (now denoted as R_{i+p} on the bottom left side of the figure). We project forward the matched region R_i by finding a corresponding match for R_{i+p} in landmark L_j, in this case with region $L_{j,k+l}$ (denoted in solid green at the right side of the green) , which was found further in the path during training.

2) Lateral Difference Estimation: To estimate the lateral difference between image coordinates of the database and input region we utilize the SIFT keypoints match-pairs from all forward-projected matched salient regions. Here we adapt an algorithm by [15] onto the salient regions. That is, instead of using a single whole image from a training run, we use matched database regions that may come from different time steps. Each keypoint pair (from each region) votes for candidate heading directions. We discard the vertical coordinate difference, and are only concerned with the horizontal component as we assume that the robot stays on the ground at all times.

The algorithm is stated below and is illustrated in figure 5. We consider three scenarios for three resulting headings (left, straight, right). We define X_t as the horizontal coordinate of the SIFT keypoint in a matched input region and X_d as the corresponding point of the landmark database region. In addition, we also set the origin to be at the center of the image, denoted as broken lines in the middle of each case in the figure (there are two cases for each of the three scenarios). This way, $X_d < 0$ means the keypoint is to the left of the midline:

- if($X_t > X_d$ and $X_t > 0$), then turn right
- else if($X_t < X_d$ and $X_t < 0$), then turn left
- else, go straight

Our robot, Beobot 2.0 [31], uses an electric wheelchair base and has a two-wheel differential drive. It is controlled by commanding a translational and rotational speed between -1.0 and 1.0 (full speed backward to full speed forward, and full speed clockwise turn to full speed counter-clockwise

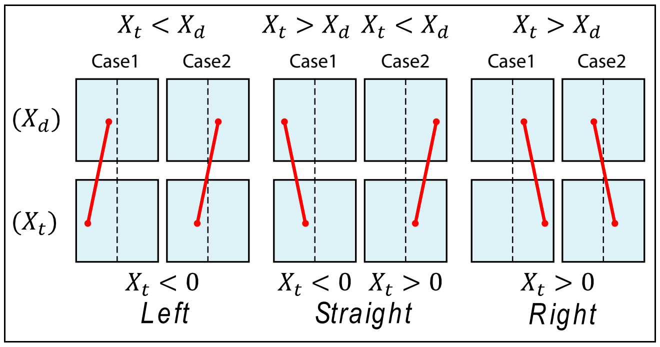

Fig. 5. Diagram for the voting scheme of navigation algorithm. There are three scenarios where the robot either moves left, straight, or right. For each scenario, there are two cases depending on the sign of X_d, the corresponding point of the landmark database region. If X_d is to the left of the center of the image (the dashed line that goes through the origin of the image coordinate system), then $X_d < 0$. The same applies for X_t, the horizontal coordinate of the keypoint in a matched input region. If $X_t < X_d$ and $X_t < 0$, then turn left, else if $X_t > X_d$ and $X_t > 0$, then turn right. If neither, then go straight.

turn, respectively). We keep the translation at constant speed, while the turn speed is proportional to the average horizontal pixel difference. Note that the direction of turn is obtained through the voting process above.

We find that our well-isolated salient regions make for clean correspondences that, for the most part, overwhelmingly agree with one of the three candidate directions. This is because, by using a compact region (as opposed to entire image like the original [15] algorithm), we avoid matching other distracting portions of the image, which may be occupied by dynamic obstacles, such as people.

3) Path Planning Biasing: What we have implemented so far is to a policy that allows the robot to follow a continuous portion of the route, along the edges of the topological map. However, when the robot arrives at an intersection, there may be multiple paths that stem out from it. The system may encounter a situation where it simultaneously recognize regions that are going to lead the robot to two or more different directions. For example, regions from the segment the robot is currently on may suggest it to continue to go straight, while regions on an incident segment are going to suggest it to make a turn. Here, we consider the task, the assigned goal location, by consulting with the path planner.

The system first checks with the localization sub-system to see whether it is approaching an intersection. If so, the path planner then bias the localization system to first compare input regions with database regions along the path to the goal, before expanding the search. In addition, for subsequent matches, we also prioritize comparison with database regions that are further along the intersection turn. This way, the robot is going to follow the same sequence of region matches that are established during training, which is critical for successful turning.

At the end, for failure recovery, whenever the robot cannot recognize any salient region for a period of time (we set for five seconds), it stops and spins in place to scan its surrounding. Note that the path planner re-plans every time the localizer sends out a new current location to make correct adjustments even when the robot loses its way.

In summary, the procedure goes as follows: after the go-to-goal command is received from a user interface, the system starts by recognizing at least one salient region to localize. Once the robot is successful in doing so, it plans its path to the goal location. The navigation system then uses the initially recognized salient regions to properly move closer to its goal. During this movement, the robot continuously try to identify newer regions to follow, to advance its mission. This procedure repeats until the robot arrives at its destination.

A snapshot of how the overall system generates robot motion from a frame can be observed in figure 6.

III. TESTING AND RESULTS

We test the system using our robot, Beobot 2.0 [31] (observe figure 1), which is 57cm in length and 60cm in width. The camera itself is about 117cm above the ground. Beobot 2.0 has a computing cluster of eight 2.2GHz Core2 Duo processors. We use four computers for localization, one for navigation, three for future use.

We select two different sites, an indoor and an outdoor environment, for testing. The first is the hallways of 27.13x27.13m HNB building, with a corridor width of 1.83m. The path forms a square with 90 degree angles when turning from one segment to another (observe the map in figure 6). The second one is the 69.49x18.29m outdoor engineering quad (Equad), where there are buildings as well as trees, with a pathway width of 3m. Snapshots of each environment are shown in figure 7.

Despite the fact that, in our testing environments, we only map one option on each junction, the path planner still perform its job of selecting that option. Note that picking the right junction to proceed to the goal trivial using shortest-path in the graph-based topological map. On the other hand, the actual turning execution on the chosen junction is much more difficult.

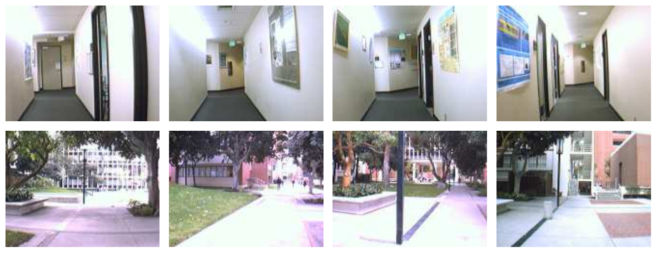

Fig. 7. Scene examples of the HNB and Equad environment.

For each environment, we run the robot through all paths once for training. We then run the robot five times indoors and three times outdoors for the testing phase. During testing, we record the trajectory of the robot odometry and its localization belief. We set the ground truth of the robot location by using manually calibrated odometry reading. The baseline for an ideal navigation itself is the center of the path.

The results for both sites are shown in table I, which specifies the sites' dimensions and length of traversal. In

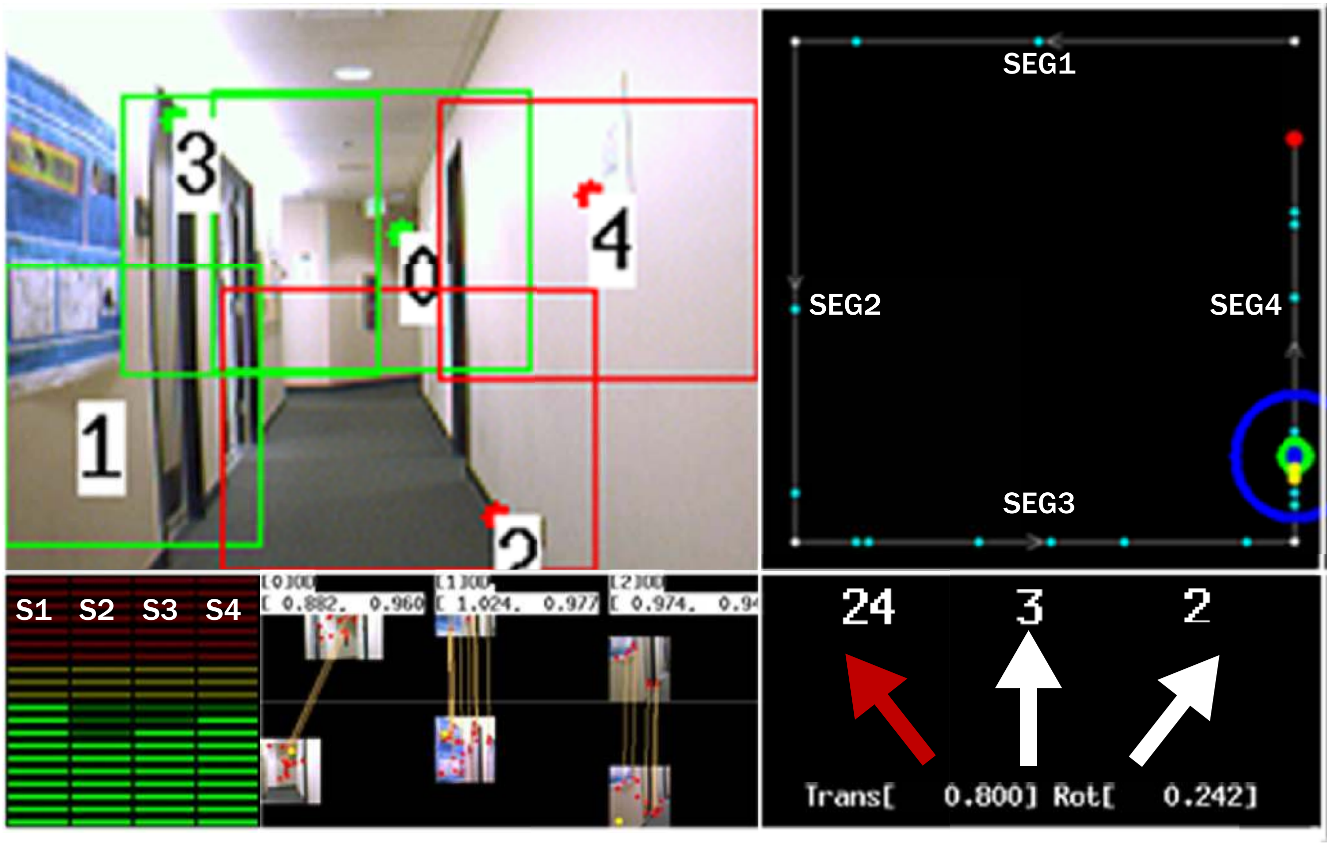

Fig. 6. A snapshot of the localization and navigation system. The input (top left) image contains the salient region windows. Green windows mean a database match, while reds are not found. Below the input image, to the left, is the segment estimation computed from gist features. There are four segments in the environment. Here the actual segment 4 is classified as second most likely (to segment 1) by the system. To the right of the segment estimation result are the matched salient regions drawn at the original detected image coordinate. Next to the input image is the robot state projected onto the map: cyan points are the location particles (hypotheses) generated by the Monte Carlo Localization algorithm, the yellow points are the locations of the matched database salient region, the green circle (the center of the blue circle) is the most likely location. The radius of the blue circle is equivalent to 3m. The robot believes that it is 4 meters from the intersection, which is correct within 20cm. On the navigation side (underneath the map), the robot is turning to the left as decided by the keypoint votes.

TABLE I
NAVIGATION AND LOCALIZATION SYSTEM PERFORMANCE RESULTS

Site	Site Dimensions	Traversal Length	Nav. Error	Loc. Error
HNB	27.13 x 27.13m	36.67m	3.68cm	1.15m
Equad	69.49 x 18.29m	138.27m	8.78cm	5.31m

addition, *nav. error* and *loc. error* refer to the navigational error (deviation from center of the path) and localization error (global robot location discrepancy), respectively.

For the first site, HNB, the result is graphed in figure 8. For both the top and bottom plots, the x-axis denotes the ground truth location of the robot, 0 being the starting location. Here, the robot runs for 36.67m. The y-axis for the top plot indicates the localization error (in meter, longitudinally along the path) as the robot moves forward, while the y-axis for the bottom plot reports the lateral deviation from the center of the hallway (also in meter, transversally to the path). We divide (using vertical lines) and label the route into segments. Note that during the segment transition, the navigation error shoots up as makes its turn. In addition, when the robot arrives at the next segment, it then proceed to recenter itself.

As shown in the graph as well as by observation, Beobot2.0 is able to traverse the route, on average, within 3.68cm of lateral deviation from the center of the hallway, without hitting the wall, which we believe is the more important fact. The longitudinal localization error itself averages to 1.15m. Fortunately the longitudinal error, in most all occasions, do not misplace the robot to a different segment. It merely places the robot ahead or behind the actual location in the route. The reason for why there is so much difference (a factor of 30) between the localization and the lateral deviation error, is because the way the landmarks are viewed with a forward-facing camera. That is, the system is very sensitive to transverse translation, but not as sensitive to longitudinal translation.

The results for testing in the outdoor site, Equad, is shown in figure 9, using the same format as the previous graph for HNB. Beobot2.0 is able to navigate with 8.78cm of lateral deviation from the center of path, on average. The

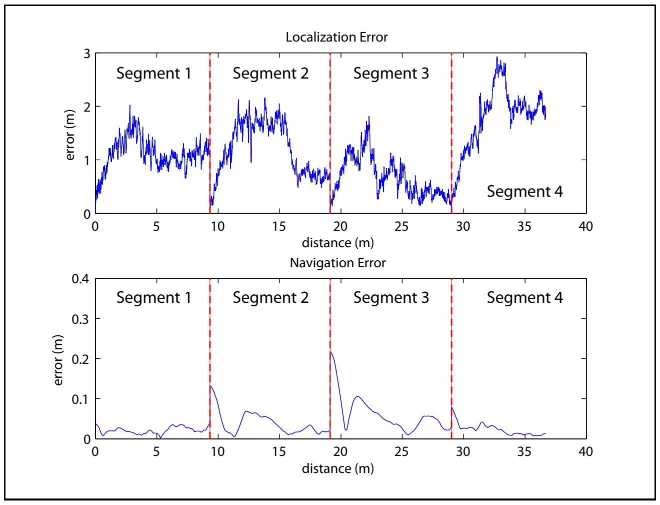

Fig. 8. Result for the indoor HNB testing. Graph labels are described in the text.

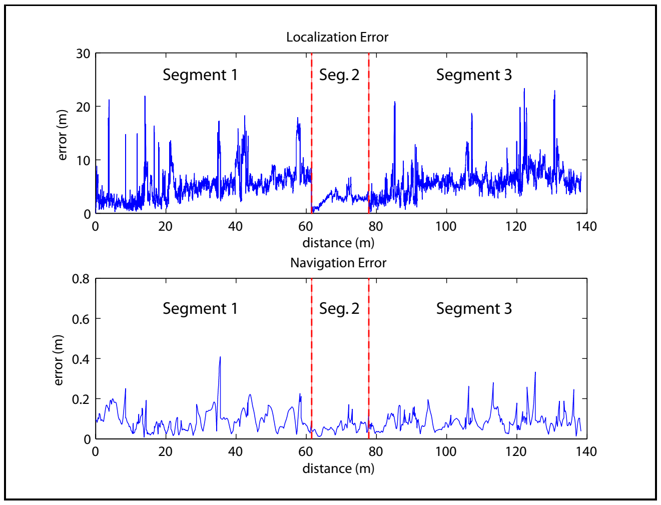

Fig. 9. Result for the outdoor Equad environment. Graph labels are described in the text.

localization error itself averages to 5.31m over a route with a total length of 138.27m. Given that the length of the route is 3.77 times that of the indoor data, both localization as well as the lateral deviation error do not increase uncontrollably. For outdoor navigation, the lateral deviation errors obtained here are acceptable as the lanes (3m wide) in the environment are about two times wider.

The lateral deviation error increases mainly because the localization module cannot find salient regions for a longer period of time, which makes the robot lose its navigation cues. This usually occurred when localization was failing (note the spikes in the localization error graph). We attribute this to the added difficulties in recognizing landmarks outdoors, where lighting conditions are not constant. Note that we train and test the system on separate days. On the other hand, our system has a very low likelihood of false positive identification as it has a high threshold for matching the landmarks. That is, in the event where the robot's location is misplaced, the navigation system will not produce false cues to follow. This allows the system to stop, recover, and proceed forward. There are, however, times where we have to stop the robot because it has swerved off the pathway considerably and gets dangerously close to a ditch or about to hit a pole or a bench.

IV. DISCUSSION AND CONCLUSIONS

In this paper we present a robot vision navigation and localization system. It uses salient regions to both localize the robot as well as to direct its heading. By having both mobile capabilities, our system can perform a user-specified command to go to a goal location. This is an extension to most available systems, which are either just a vision localization system (with the movement controlled by a user) or a vision navigation system (which can only perform navigation from pre-specified starting and ending locations).

For the most part, the robot is able to navigate using landmarks that are identified by the localizer. However, this is also a source of problem in that if the robot fails to recognize a landmark, it cannot navigate. The next step to improve the system is to have robot visually navigate itself without localization during those times. We suggest two different vision techniques. One is road recognition to identify the direction of the path, or whether there is an intersection. The second is a form of 3D structure understanding (using structure from motion [32], or single image [33], [34]). This is especially useful to detect walls and obstacles.

V. ACKNOWLEDGMENTS

The authors gratefully acknowledge the support of NSF, General Motors, ARO, DARPA, and the Okawa Foundation.

The authors affirm that the views expressed herein are solely their own, and do not represent the views of the United States government or any agency thereof.

References

[1] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte-carlo localization for mobile robots," *Artificial Intelligence*, vol. 128, no. 1-2, pp. 99–141, 2000.

[2] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," in *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*. Acapulco, Mexico: IJCAI, 2003.

[3] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Stanley, the robot that won the darpa grand challenge." *Journal of Field Robotics*, vol. 36, pp. 1 – 43, 2007.

[4] M. Montemerlo, S. Thrun, H. Dahlkamp, D. Stavens, and S. Strohband, "Winning the darpa grand challenge with an ai robot," in *AAAI Conference On Artificial Intelligence*, vol. 1, 2006, pp. 982 – 987.

[5] M. Cummins and P. Newman, "Fab-map: Probabilistic localization and mapping in the space of appearance," *Int. J. Rob. Res.*, vol. 27, no. 6, pp. 647–665, 2008.

[6] G. Schindler, M. Brown, and R. Szeliski, "City-scale location recognition," in *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, vol. 0. Los Alamitos, CA, USA: IEEE Computer Society, 2007, pp. 1–7.

[7] C. Valgren and A. J. Lilienthal, "Incremental spectral clustering and seasons: Appearance-based localization in outdoor environments," in *ICRA*, Pasadena, CA, 2008.

[8] A. Pronobis, B. Caputo, P. Jensfelt, and H. Christensen, "A discriminative approach to robust visual place recognition," in *IROS*, 2006.

[9] A. Ramisa, A. Tapus, R. L. de Mantaras, and R. Toledo, "Mobile robot localization using panoramic vision and combination of local feature region detectors," in *ICRA*, Pasadena, CA, May 2008, pp. 538–543.

[10] S. Frintrop, P. Jensfelt, and H. Christensen, "Attention landmark selection for visual slam," in *IROS*, Beijing, October 2006.

[11] C. Siagian and L. Itti, "Biologically inspired mobile robot vision localization," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 861–873, July 2009.

[12] P. Newman, G. Sibley, M. Smith, M. Cummins, A. Harrison, C. Mei, I. Posner, R. Shade, D. Schroeter, L. Murphy, W. Churchill, D. Cole, and I. Reid, "Navigating, recognizing and describing urban spaces with vision and lasers," *International Journal of Robotics Research*, vol. 28, pp. 1406 – 1433, November 2009.

[13] A. Angeli, D. Filliat, S. Doncieux, and J.-A. Meyer, "A fast and incremental method for loop-closure detection using bags of visual words," 2008.

[14] A. Murillo, J. Guerrero, and C. Sagues, "Surf features for efficient robot localization with omnidirectional images," in *Robotics and Automation, 2007 IEEE International Conference on*, April 2007, pp. 3901–3907.

[15] Z. Chen and S. T. Birchfield, "Qualitative vision-based path following," *IEEE Transactions on Robotics*, vol. 25, no. 3, pp. 749 – 754, June 2009.

[16] S. Segvic, A. Remazeilles, A. Diosi, and F. Chaumette, "A mapping and localization framework for scalable appearance-based navigation," *Computer Vision and Image Understanding*, vol. 113, no. 2, pp. 172–187, February 2009.

[17] M. D. E. Royer, M. Lhuillier and J. M. Lavest, "Monocular vision for mobile robot localization and autonomous navigation," *International Journal of Computer Vision*, vol. 74, no. 3, pp. 237–260, January 2007.

[18] J. Shi and C. Tomasi, "Good features to track," in *IEEE Conference on Computer Vision and Pattern Recognition*, 1994, pp. 593–600.

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

[20] F. Paetzold and U. Franke, "Road recognition in urban environment," in *IEEE Conference on Intelligent Transportation Systems*, Stuttgart, 1998.

[21] C. Rasmussen, Y. Lu, and M. Kocamaz, "Appearance contrast for fast, robust trail-following," in *IROS*, St. Louis, Missouri, USA, 2009.

[22] M. Blas, M. Agrawal, A. Sundaresan, and K. Konolige, "Fast color/texture segmentation for outdoor robots," in *IROS*, Nice, France, 2008.

[23] C. Guo and S. Mita, "Stereovision-based road boundary detection for intelligent vehicles in challenging scenarios," in *IROS*, St. Louis, Missouri, USA, 2009.

[24] Y. He, H. Wang, and B. Zhang, "Color-based road detection in urban traffic scenes," *IEEE Transactions on Intelligent Transport System*, vol. 5, no. 4, pp. 309 – 318, 2004.

[25] M. Lutzeler and S. Baten, "Road recognition for a tracked vehicle," *Proc. SPIE Enhanced and Synthetic Vision*, pp. 171 – 180, 2000.

[26] A. Murillo, J. Guerrero, and C. Sagues, "Improving topological maps for safer and robust navigation," in *IROS*, St. Louis, MO, USA, October 2009, pp. 3609–3614.

[27] C. Siagian and L. Itti, "Rapid biologically-inspired scene classification using features shared with visual attention," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 2, pp. 300–312, Feb 2007.

[28] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 11, pp. 1254–1259, Nov 1998.

[29] L. Itti, "Models of bottom-up and top-down visual attention," Ph.D. dissertation, Pasadena, California, Jan 2000.

[30] C. Siagian and L. Itti, "Storing and recalling information for vision localization," in *IEEE International Conference on Robotics and Automation (ICRA), Pasadena, California*, May 2008.

[31] C. Siagian, C. K. Chang, R. Voorhies, and L. Itti, "Beobot 2.0: Cluster architecture for mobile robotics," *Journal of Field Robotics*, 2010, in submission.

[32] N. Snavely, S. M. Seitz, and R. Szeliski, "Skeletal sets for efficient structure from motion," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[33] A. Saxena, M. Sun, and A. Y. Ng, "Make3d: Learning 3-d scene structure from a single still image," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 31, no. 5, pp. 824 – 840, 2008.

[34] D. Hoiem, A. Stein, A. Efros, and M. Hebert, "Recovering occlusion boundaries from a single image," in *ICCV*, 2007.