2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

Storing and Recalling Information for Vision Localization

Christian Siagian and Laurent Itti

Abstract— **In implementing a vision localization system, a crucial issue to consider is how to efficiently store and recall the necessary information so that the robot is not only able to accurately localize itself, but does so in a timely manner. In the presented system, we discuss a strategy to minimize the amount of stored data by analyzing the strengths and weaknesses of several cooperating recognition modules, and by using them through a prioritization scheme, which orders the data entries from the most likely to match to the least. We validate the system is a series of experiments at three large scale outdoor environments: a building complex (126x180ft. area, 3583 testing images), a vegetation-filled park (270x360ft. area, 6006 testing images), and an open-field area (450x585ft. area, 8823 testing images) - each with its own set of challenges. Not only is the system able to localize in these environments (on average 3.46ft., 6.55ft. 12.96ft. of error, respectively), it does so while searching through only 7.35%, 3.50%, and 6.12% of all the stored information, respectively.**

I. INTRODUCTION

Vision localization has been an active research branch for the past few dacades [1]. In general, a localization system has the following parts: image acquisition and pre-processing, database matching, pose estimation, and localization. In image acquisition and pre-processing, the system computes the necessary discriminating visual cues from a raw image. Database matching is a process of comparing those cues to stored information, usually obtained during training. After a match is found, the system still has to estimate the current pose of the robot's camera with respect to the database (reference) entry before an actual attempt at localization is performed. In recent years, probabilistic localization has matured and its standard techniques (largely utilize other sensors such are range sensors [2], [3], GPS [4], and odometry) are now well understood. However, the use of vision sensors (cameras) has not been as extensively developed. In order to implement a successful vision localization system, everything hinges on achieving a reliable database matching process, one that leads to satisfactory pose estimation. And because robots are real-time systems, it is not enough just to have an accurate recognition system; it is also important to consider the amount of time needed to find a match, especially when exploring large environments.

A way to solve this problem is by storing as little as possible while still able to cover all pertinent aspects of the environment. The decision of what to store is tightly coupled with the capability of the system's recognition module, the part most responsible for its accuracy. If the module is able to robustly match an object in any condition, then one photograph would suffice. However, such capability is not yet achievable, for example, in invariance with respect to lighting (particularly outdoors) and viewpoint. What is the alternative? We are forced to consider objects under different conditions as different objects.

This work was supported by NSF, NGA, DARPA and HFSP.

C. Siagian is with Department of Computer Science, University of Southern California, Hedco Neuroscience Building - Room 10, 3641 Watt Way, Los Angeles, California, 90089-2520, USA. 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

One may want to deduce that the better the recognition system, the smaller the database size. However, sometimes, the reason that a system is better than another is because the number of descriptors associated is higher than its competition. Consequently, even though we lower the number of entries in the database, we also increase the number of features per entry. And thus, when looking into the amount of information stored by a system, we have to note the numbers of features it extracts as well.

Broadly speaking, there are two types of visual features: local features and global features. Local features are computed over a limited area of the image, as opposed to global features which may pool information over the entire image into, e.g., histograms.

In recent years we have seen a number of systems utilizing local features called the SIFT keypoints [5], which have been proven to be quite accurate for this purpose [6], [7] because they are invariant to scale, with some viewpoint and lighting invariance. Other local-feature such as Kernel PCA features [8] and Harris corners [9] are also used with varying degrees of success. One disadvantage of using local features is the number of features needed to be stored for each image. In addition, in general, local features are less stable than global features. For example, in a park where vegetation dominates (observe second row of figure 5), the majority of the local features is probably only be found in a single image.

Global features, on the other hand, never have to perform matching to such level of detail as histograms average out the local activities to form more robust values. However, these holistic approaches, which utilize color [10], [11], textures [12], or a combination of [13], [14], are limited, for the most part, to classifying places. This is because the end correspondences are not at the coordinate level, which are needed for accurate pose estimation. Nevertheless, with lower localization resolution, global features gain sizable advantage in speed as classifiers usually output their results almost instantaneously. And yet, in the end, it would be hard to perform metric localization using just global features.

So where do we go from here? Up to this point we are only working on the hypothesis that the smaller the database

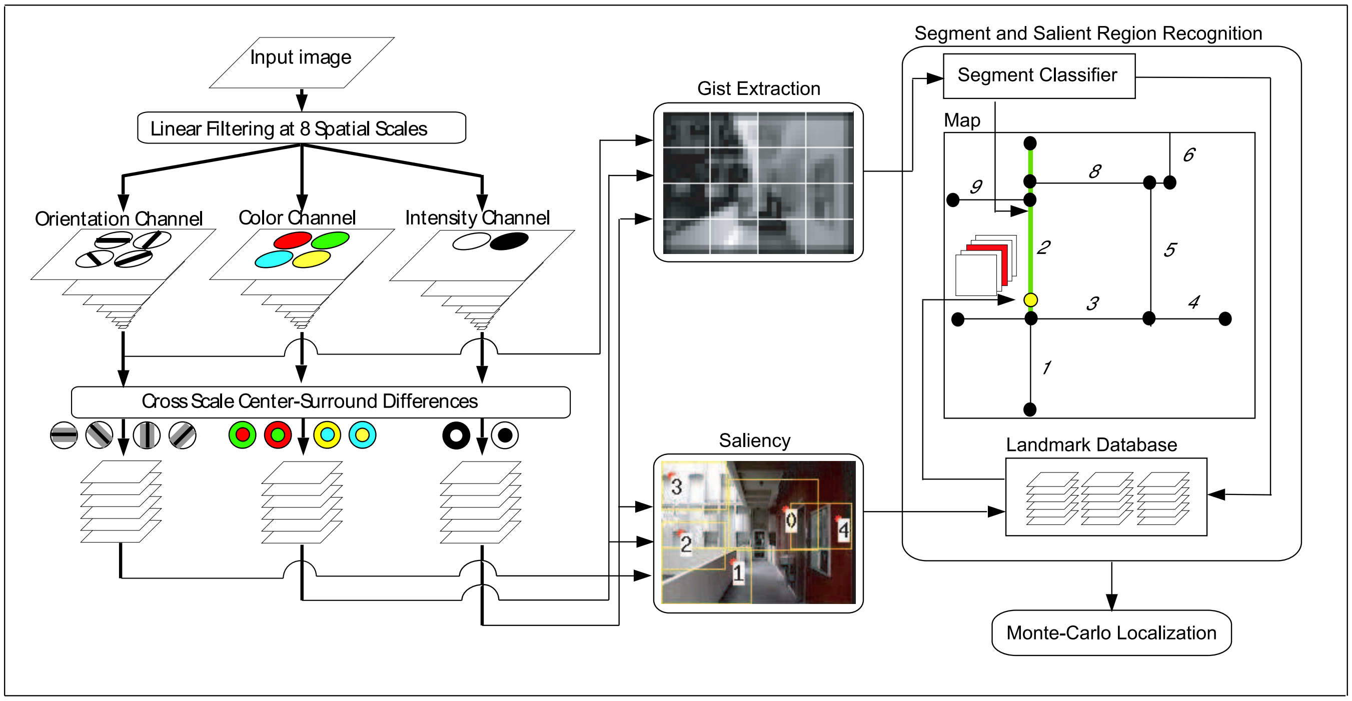

Fig. 1. Diagram of the Vision Localization System. The system extract various features from several domains (color, orientation, and intensity) and computes gist features and salient regions. At the next stage the system estimate the segment using gist and tries to match the regions with the ones previously seen in the environment. These matches are then used as an input to the localization stage to make a decision of where the robot might be.

or the lower the number of features per image, the faster the matching procedure is. The problem is, it implicitly assumes the need for a best match, which requires comparisons with the whole database. However, for real-time systems such as robot localization, a positive match is all that we need. And once it is found, the search process can be stopped. To increase the likelihood of this event to occur early, our system utilizes a prioritization step to compare database entries in the order of the most likely to the least. Thus, in practice, it never has to look at all the entries because (through experiments) it has a good idea when to stop given the unlikelihood that a match will be found thereafter. In previous work [15], we presented a biologically plausible vision localization system. Here we improve it by prioritizing the search through the stored information that is already minimized both in the number of entries as well as the number of features per entry. We then test the system in visually contrasting and challenging large-scale environments, which validate the accuracy and scalability of the approach.

II. DESIGN AND IMPLEMENTATIONS

At the core of the system [15] illustrated in figure 1, is the utilization of saliency [16], [17] and gist [13], which complement each other to form a multi-expert recognition system that localizes at two levels. That is, gist, which is a global feature, tries to recognize places called segments and saliency combined with SIFT (both are local features) form a salient region to further refine the result to the coordinate location using a back-end Monte Carlo Localization.

A segment is an ordered list of edges to form a continuous pathway in an environment. It can be a portion of a hallway or a road interrupted by physical barriers (crossing, intersection, etc.) at both ends for natural delineations. This grouping is motivated by the fact that the views in a segment are coarsely similar, which allows the segment estimator to classify them using gist features. The selected three-edge segment (highlighted in green) in the map of figure 1 is an example. Because the segment classifier (a back-propagation neural networks [13]) runs trivially fast, our effort to speed up the system mostly focuses on the salient region matching.

A salient region, which can be viewed in figure 1 (there are 5 of them in the frame), refers to a conspicuous area of an input image that is easily detected in the environment, making it a good candidate for a localization cue. An ideal salient region is one that is persistently observed from different points of view and at different times of the day. A salient region does not have to depict an individual object (many times it is a small part of an object or a set of objects), it just has to be a snapshot of a point of interest situated in the real world that, as time goes on, is proven to be consistently detectable. To this end, a set of salient regions that portrays the same point of interest is grouped together and the set is called a landmark. And so, a salient region can be considered as an evidence of a landmark. One important consequence of using salient regions, which has been demonstrated in previous works [18], [19], [9], is that it relieves the system from matching whole scenes. By extracting features within a small window, the number of SIFT keypoints can be drastically reduced. This is substantial because SIFT matching is the slowest part of the system.

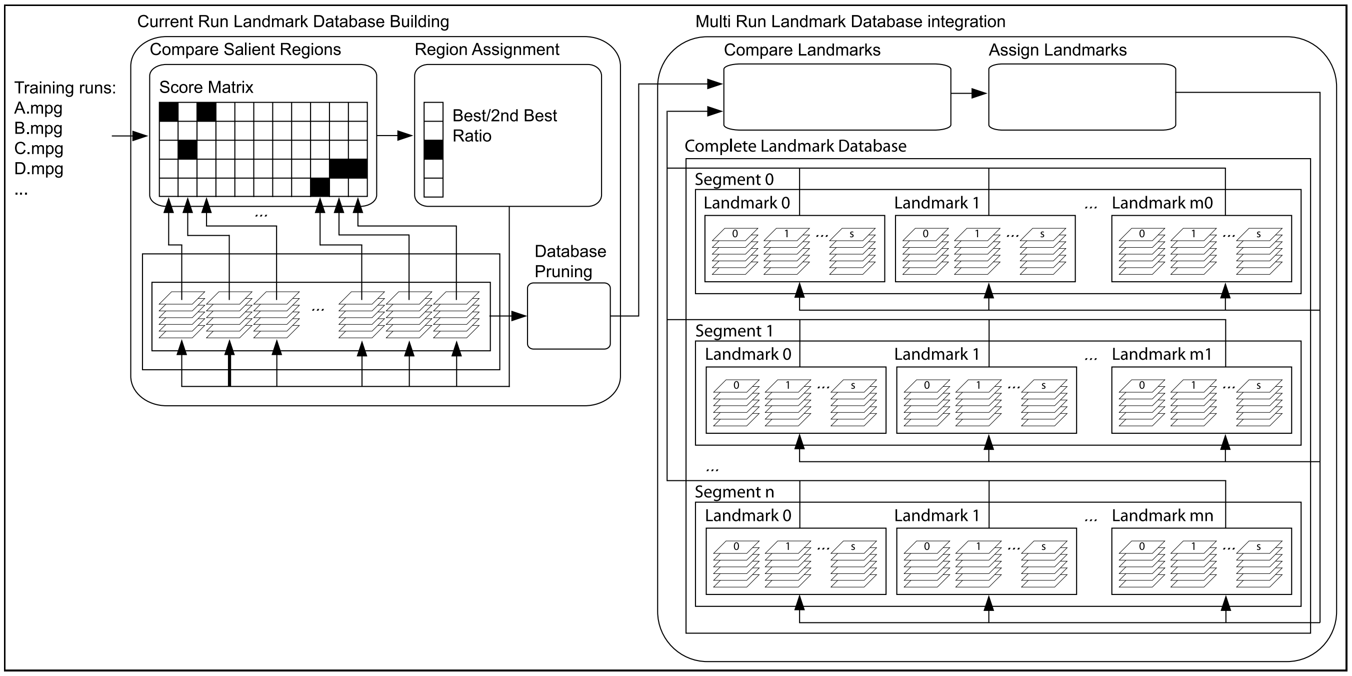

Fig. 2. The landmark database building procedure is done in two steps: create a current-episode landmark database for each training session, then iteratively (not all at once) integrate them together to create one complete landmark database.

Therefore, in constructing an optimal sized landmark database, we have to pay close attention to the shortcomings of SIFT keypoints, which have less invariance to lighting and out-of-plane viewpoint change. That is, multiple entries of a landmark should directly increase the robustness of the recognition step to those changes. For improving viewpoint invariance, when building a landmark, we keep its snapshots from all the viewing angles spaced as far as SIFT allow us to. This strategy actually produces low salient region counts because a lot of the regions isolate parts of images that are physically far away from the robot (signs, buildings), which means that angle changes induced by its movement do not affect their viewing as much. As for lighting, as with other textures, SIFT keypoints are usually very different in wider disparity cases. Our solution is to survey and select training sessions to include all distinct conditions but with enough overlap for the same landmarks from each session to be considered similar.

The landmark database building procedure is illustrated in figure 2 and it goes as follows: create a landmark database for each training session (section II-A), then combine them to create one complete landmark database (section II-B). We can then discuss the run-time prioritization step in section II-C. In these sections, we describe procedures that need a few decision thresholds, which may be viewed as making the approach weaker. However, they are quite intuitive and we try to characterize what overall impact each of them has.

A. Building a Database Within an Episode

Given a series of frames of robot traversal from a training session, we create a database that stores all of its persistent landmarks. Training is done with a person controlling the robot's movement, running straight through all the paths in the map, noting which segment the current frame belongs to (the landmarks are compartmentalized in segments of origin).

From the first frame we obtain a set of salient regions to create initial landmarks. When the next set of regions arrives (from the subsequent frame), the system tries to concurrently match them with the ones in existing landmarks . We first create a two-dimensional match score matrix between all combinations of the incoming regions and current landmarks. This is done through a matching process is illustrated in figure 3. In the figure, location of the salient points are drawn as yellow points. These points are where we obtain a set of values called the salient feature vector [15], which are normalized values from the six center-surround (feature) maps [16], [17] of each of the sub-channels of the color, intensity, and orientation channels. For each of the maps we store values from a 5-by-5 window centered at the salient point.

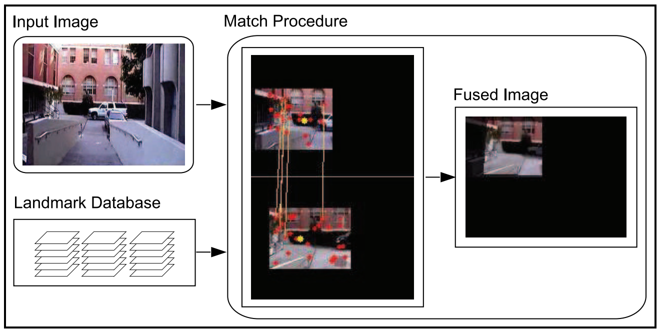

Fig. 3. Matching process of two salient regions using SIFT keypoints and salient feature vector

There are two matching steps: SIFT and saliency feature vector matching. If either step turned up negative, the score entry in the match matrix is set to zero. For SIFT matching, the [5] procedure is directly applied, and we use a generalized Hough transform to estimate the affine motion. For the latter step, we use equation 1, which factors in both salient feature similarity s_{diff} (euclidian distance) and salient point location proximity s_{dist} (observe the fused image in figure 3 where the two regions are aligned together) normalized by the image diagonal length l_{Diag}. The second term is reversed to allow for increase in value the closer the points are. The threshold score is .75 out of the maximal 1.0. We find that, in experimentation, any values above the threshold are most likely a match (it is a conservative cut-off).

$$score = s_{diff} * (1 - \frac{s_{dist}}{l_{Diag}}) \tag{1}$$

Once all the comparisons are performed, we start the insertion process by calculating the best/2nd best salient feature similarity score ratio for each region-landmark pair. The region with the highest remaining ratio at the current iteration is inserted to the corresponding landmark. We keep doing this until there are no more matches, in which we can then create new landmarks for the remaining regions. We only take the saliency score into account (and not the SIFT score) because we want to cluster regions based on just the salient landmark they depict, not on the overall region similarity. Adding the SIFT score can allow overlapping regions that depict different landmarks to be clustered.

After all the frames are processed, we prune out landmarks using two criteria: number of salient regions and range of frame numbers, both of which indicate persistance in the environment. The first one is for landmarks that are very salient but are viewed briefly (20 frames or less): their total counts have to be larger than 7. The second one is for landmarks that are less salient but are detected for a long period (frame number range larger than 20): at least 5 regions. These set of thresholds control how many landmarks we want to keep. We find that these values allow enough of the smaller but useful landmarks (fewer number of regions registered) to be kept by the database. Adding even smaller ones would just increase the size of the database without getting much in return.

In the region assigment step, for regions that are positively matched with multiple landmarks, the situation becomes complicated. First, we have to find the best landmark to insert to; adding a salient region to multiple landmarks would unnecessarily increase the size of the database. Here, we select the one with the highest number of regions to create a momentum towards the larger landmarks. Multiple landmark matches occur because matches that were supposed to happen in the previous frames did not go through, and we are left with more than one landmark depicting the same point of interest. The reason we use landmark size and not the score is because it is not unusual to have two landmark matches where one has a large number of salient regions and the other has one that comes from the previous frame, and thus have a higher score. It is obvious that the smaller landmark is supposed to be part of the larger one, and the new region makes the connection evident. With the policy, we keep landmarks from spliting to smaller ones.

For this reason, the system also consolidates all the regions involved in the multiple match to the largest landmark. It does not combine the landmarks together because the other regions in those landmarks may legitimately describe other points of interest. For example, there is a possibility that two landmarks somehow move closer together and create an ambiguity. What is achieved by moving these regions is that the landmarks become less similar, which is a reasonable compromise.

Before explaining the transfer of salient regions, we would like to describe the actual inner working of a landmark. In training, when a landmark is being built, it actually consists of two lists: a main list and a temporary list. The main list has regions that are saved at the end. For the purpose of pruning, however, the total number of salient regions is taken as the sum of the two lists. When an incoming salient region is compared to a landmark, it is first compared with the main list (actually, just the last 10 in reverse order) and, if needed, the temp list as well (also the last 10 in reverse order). If there is a match in the main list (and the landmark is also selected through the ratio test), the new region goes to the temporary list. If we find no match in the main list but one is found in the temporary list (and also passes the ratio test) the corresponding matched region in the temporary list is put to the main list and the new one is put to the temporary list. What is essentially accomplished is that one region is a representative for as many regions as possible until its appearance becomes so different compared to an incoming region that we are forced to store an additional one.

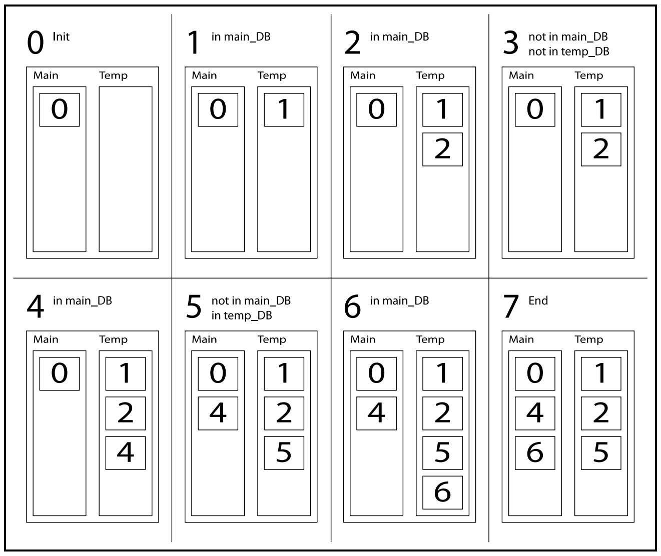

Fig. 4. An example of how a series of 8 frames affects the number of salient regions that are stored in a landmark during training/building. The number in the top left corner of each grid is the frame number. The label next to it is the matching condition or command that comes the system.

Figure 4 shows how the algorithm works in a series of 8 frames. In the first frame, an initial salient region is used to create a landmark and it is automatically put to the main list. In frames 1 and 2, the new regions are sufficiently similar to region 0 that they are put to the temp list (this is what happens the majority of the time). Frame 3 is an example where the landmark is not salient enough in the frame and thus is not detected. In frame 4, the new region is again still similar to frame 0, so it goes to temp. Frame 5 is where the algorithm provides a benefit, the incoming region is not similar to region 0 but close enough to the one from frame 4 (obviously because they are from back-to-back frames). And thus, region 4 is moved to the main list and region 5 is inserted to the temp list. After another uneventful frame 6, the end signal is received and the last entry in the temp list is moved to the main list to produce a complete list that will be saved. We find that the save:discard ratio obtained is (on average) about 1:5.

In transfer of evidence for multiple landmark matches, we have to be careful in how to re-link the lists in landmarks that lose a salient region. There are two different cases, the region to be moved is either in a main or a temp list. If the latter is the case, it can simply be moved because there is a similar region in the main list. If, on the other hand, the region is in the main list we have to replace it with one most similar in the temp list: the one from the closest but higher frame number.

B. Building a Database Across Episodes

The procedure of combining databases from individual training episodes to a complete database is done iteratively; we match and add one episode at a time. The matches are done at landmark-to-landmark level for all incoming-and-stored landmark combinations. That is, when deciding if two landmarks depict the same real-world point-of-interest, the system counts how many regions from one landmark matches the ones from the other. In addition, it also looks at the percentage of regions matched in the landmark to be added. For two landmarks to be combined, they have to pass any of the following thresholds:

- 2 to 5 match count and $>= 50\%$ of matches
- 6 to 10 match count and $>= 25\%$ of matches
- above 10 match count

We find that these values to be fairly safe, that the combined landmarks are almost always the same ones found at different sessions. In the overall scheme, knowledge that certain landmarks are found at multiple occasions would be helpful in gauging its recall reliability, which we have not fully exploit. For one, we can add a filtering step in the end that prunes landmarks across sessions, only keeping the ones that occur in more than one episodes.

When combining two landmarks, no salient regions are deleted (even if there are identical ones taken from different sessions). We simply append one list to the back of the other to keep the episodic progression (for priming where to expect to landmark at later frames) in tact. Also, if a landmark from the currently processed database matches with more than one landmark in the accumulated database, these stored landmarks are first combined before appending the incoming landmark. To keep the salient regions properly stored, they are sorted based on the session names first (alpha-numeric order), and frame numbers second.

C. Landmark Database Search Prioritization

The database search prioritization module is a fast run-time procedure that puts the landmarks to be compared in an order from the most likely to be positively matched to the least. This procedure speeds up the database search because once a match is found, the search is called off.

We formulate a priority value for comparison between landmark $lmk_{i,j}$ (from segment i of index j) and incoming salient region $sReg_k$ denoted in equation 2 which weighs the following factors: segment estimation using gist features, salient feature similarity, and current location belief.

$$\begin{aligned} priority(lmk_{i,j}, sReg_k) = \\ & W_{gist} * sval_i + \\ & W_{sal} * salDiff(lmk_{i,j}, sReg_k) + \\ & W_{loc} * dist(lmk_{i,j}, loc(S_t)) \end{aligned} \tag{2}$$

The weights used are $W_{gist} = .5$, $W_{sal} = .2$, and $W_{loc} = .3$, which we found through experimentation.

Because the landmarks are arranged by segment of origin, the segment estimator values $sval_i$ can be used to prioritize search order by most likely segment first. For salient feature similarity (second term), we pre-compute the average salient feature values for each stored landmark $lmk_{i,j}$ and, during run time, we compute $salDiff(lmk_{i,j}, sReg_k)$ (the euclidian distance between the salient features of the incoming region $sReg_k$ and the landmark $lmk_{i,j}$ average feature vector). Because this priority computation is done on the landmark level (a landmark, on average, have a little less than 20 regions), the procedure is still fast. The third term, $dist(lmk_{i,j}, loc(S_t))$, orders the landmarks by its proximity to the current belief location S_t (formulation convention in [15]) for the state of the robot at time t, which adds a temporal aspect to the priority value.

The system then creates a job item for all incoming salient-region-and-landmark combinations for multi-threaded search. These job items are put to a priority queue accessible by all the processors in the robot, so that each can start the slow region recognition process. We noticed that most regions that are found are discovered early in the search. Equipped with this knowledge we employ a number of exit conditions that calls off the searches if:

- 3 regions are matched.
- 2 regions matched and 10% of queue has been processed since last match.
- 1 region is matched and 20% of queue has been processed since last match.
- no regions are matched and 33% of queue has been processed.

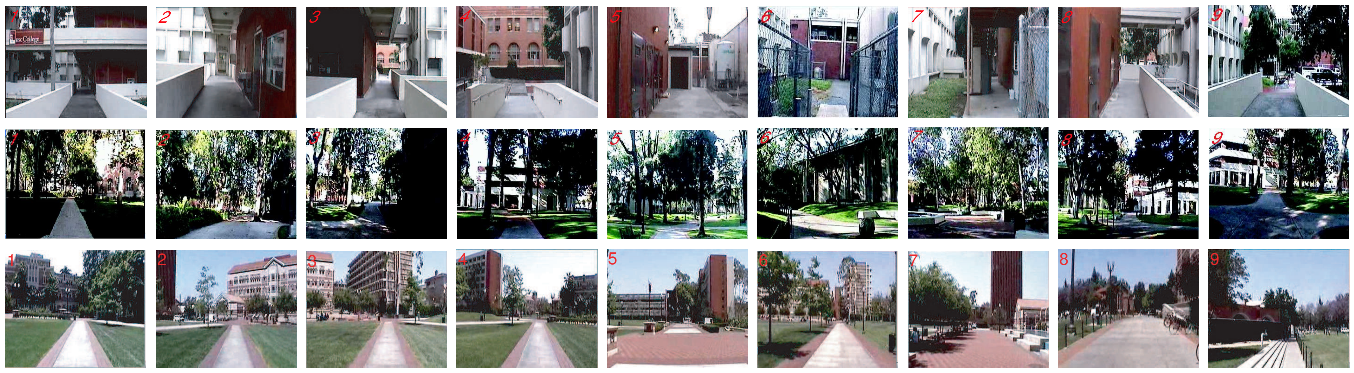

Fig. 5. Examples of images in each of the nine segments (with corresponding label) of ACB (first row), AnFpark (second row), and FDFpark (third row)

These thresholds are very conservative, as shown in the following testing section III.

III. TESTING AND RESULTS

We test the system at three sites on campus with example scenes of each site occupying a row of figure 5. Each of the sites has nine segments and each image is a sample for a segment. The first site (first row of figure 5) is the 126x180ft. Ahmanson Center for Biological Research (ACB) building complex. The second site (second row) is a region comprised of two adjoining parks: Associate and Founders park (AnF), which are dominated by vegetation and make up 270x360ft. area. The third site (third row) is an open area in the 450x585ft. Frederick. D. Fagg park where a large portion of the scenes is the sky. The same environments are used to test the gist model in [13] (maps of individual sites and the campus and available there) to perform segment classification, and in [15] for localization. Here we test and report not only the accuracy of the system, but also its efficiency in the number of comparisons with the salient regions in the landmark database.

To collect visual data we use an 8mm handheld camcorder carried by a person walking while filming. Moreover, because data is taken at approximately constant speed, we use interpolation to come up with the ground-truth location of the person for both training and testing. The main issue in collecting training samples is the selection of filming times that include all lighting conditions. We perform trial-and-error to come up with times of day (morning, noon, afternoon, early evening) and other natural events (overcast, after raining) that cover the whole lighting space. In each site we have between 9 and 11 training runs.

For each site we run the system several times with different prioritization strategies to show their impact. As a baseline localization accuracy and efficiency, we assign random priorities for each landmark in the first run. We then run the system using individual context cues (segment estimation, salient feature vector proximity, and current location belief) exclusively by zeroing out the weight of each but one (the desired) of the terms in equation 2. And lastly, we report the results using weights that we find optimum using trial and error, both with and without the early exit policy. We compare speeds of the different runs by using the total percentage of regions searched (a platform independent measure) and because the salient region recognition process is, by far, the slowest part of the system.

In our platform (a 16-core 2.6GHz machine, operating on input image size of 160x120), the Visual Cortex, gist, and saliency computations, which are also implemented in parallel (each sub-channel has its own thread), takes about 50ms/frame, while the segment estimation takes less than 1 ms. The search process, on the other hand, usually takes a few seconds to finish (note the stored number of salient regions for the large environments are between 29710 and 90660), even with the parallel implementation (we dispatch 16 threads to compare input with different parts of the landmark database). Therefore, system at its current state is not yet real time. However, we are developing a framework where the system can use just the segment estimation to provides a coarse localization hypothesis on every frame, while salient region recognition, which takes multiple time-steps, can refine the location belief whenever it is available.

Tables I, II, and III report the results. The first part of each table reports the statistics of the testing condition and the size of the database. The second part shows the performance of each run (with different prioritization parameters), which consists of the percentage of the searched salient regions in the database and number of input regions per frame for the ones that are found, not found, and the total. The numbers of input salient regions are capped at 5 per frame [15] and the totals differ slightly because of the small amount of noise added in the saliency model.

Within each environment, the errors are approximately the same, with the combination priority with the early exit policy being barely better, although not statistically significant given the standard deviation. The position disparities are mostly along the path where the ground truth is (the belief is either a bit behind or ahead), but not completely off. In FDF site in particular, a substantial number of the salient regions are far away from the robot which makes deducing very accurate (within a foot) localization visually difficult. Here, localization using regions that are closer to the robot is actually essential for convergence to the correct location.

We attribute the small difference of errors between the

TABLE I

AHMANSON CENTER FOR BIOLOGY EXPERIMENTAL RESULTS

Number of Segments: 9	Number of Landmarks in Database: 1501
Number of Training Sessions: 9	Number of Salient Regions/landmark: 19.79
Number of testing frames: 3583	Number of Salient Regions: 29710

Search Order Policy	found		not found		total		error
	% search	# of sreg./fr.	% search	# of sreg./fr.	% search	# of sreg./fr.	(ft.)
random priority	27.64%	2.77 ± 1.14	100.00%	2.13 ± 1.14	59.06%	4.89 ± 0.40	3.46 ± 4.84
segment priority	6.17%	2.77 ± 1.14	100.00%	2.13 ± 1.14	46.91%	4.89 ± 0.40	3.57 ± 3.61
saliency priority	6.24%	2.77 ± 1.14	100.00%	2.13 ± 1.14	46.95%	4.89 ± 0.40	3.62 ± 4.98
location priority	3.38%	2.77 ± 1.14	100.00%	2.13 ± 1.14	45.34%	4.89 ± 0.40	3.67 ± 3.58
combination	1.03%	2.77 ± 1.14	100.00%	2.13 ± 1.13	44.00%	4.89 ± 0.40	3.63 ± 3.83
combination + early exit	0.85%	2.50 ± 0.85	14.16%	2.39 ± 0.88	7.35%	4.89 ± 0.40	3.46 ± 3.08

NOTE (applies to all 3 tables): The first part of the table reports the environment parameters and training results. The second part shows the performance of each run (with different prioritization parameters), which consists of the percentage of the compared salient regions in the database and the average number of regions/frame for input regions that are found, not found, and the total.

TABLE II

ASSOCIATE AND FOUNDERS PARK EXPERIMENTAL RESULTS

Number of Segments: 9	Number of Landmarks in Database: 4664
Number of Training Sessions: 10	Number of Salient Regions/landmark: 17.69
Number of testing frames: 6006	Number of Salient Regions: 82502

Search Order Policy	found		not found		total		error
	% search	# of sreg./fr.	% search	# of sreg./fr.	% search	# of sreg./fr.	(ft.)
random priority	24.87%	3.52 ± 1.14	100.00%	1.47 ± 1.14	46.96%	4.98 ± 0.14	7.48 ± 10.33
segment priority	5.77%	3.52 ± 1.14	100.00%	1.47 ± 1.14	33.47%	4.98 ± 0.14	7.60 ± 10.00
saliency priority	5.24%	3.52 ± 1.14	100.00%	1.47 ± 1.14	33.09%	4.98 ± 0.14	7.25 ± 9.92
location priority	2.36%	3.52 ± 1.14	100.00%	1.47 ± 1.14	31.06%	4.98 ± 0.14	7.35 ± 9.45
combination	0.86%	3.52 ± 1.14	100.00%	1.47 ± 1.14	30.00%	4.98 ± 0.14	7.07 ± 9.29
combination + early exit	0.54%	2.84 ± 0.68	7.41%	2.14 ± 0.68	3.50%	4.98 ± 0.14	6.55 ± 5.20

TABLE III

FREDERICK D. FAGG PARK EXPERIMENTAL RESULTS

Number of Segments: 9	Number of Landmarks in Database: 4808
Number of Training Sessions: 11	Number of Salient Regions/landmark: 18.86
Number of testing frames: 8823	Number of Salient Regions: 90660

Search Order Policy	found		not found		total		error
	% search	# of sreg./fr.	% search	# of sreg./fr.	% search	# of sreg./fr.	(ft.)
random priority	29.33%	3.02 ± 1.24	100.00%	1.75 ± 1.25	55.30%	4.77 ± 0.72	15.21 ± 19.37
segment priority	8.85%	3.05 ± 1.24	100.00%	1.73 ± 1.25	41.85%	4.78 ± 0.71	14.56 ± 17.44
saliency priority	8.18%	3.05 ± 1.24	100.00%	1.73 ± 1.25	41.41%	4.78 ± 0.71	14.00 ± 15.89
location priority	3.50%	3.05 ± 1.24	100.00%	1.73 ± 1.25	38.45%	4.78 ± 0.71	14.44 ± 15.23
combination	1.28%	3.05 ± 1.24	100.00%	1.73 ± 1.25	37.02%	4.78 ± 0.71	13.95 ± 16.14
combination + early exit	0.82%	2.57 ± 0.81	12.31%	2.21 ± 0.92	6.12%	4.78 ± 0.71	12.96 ± 11.40

optimum priority and the rest to the fact that prioritization indirectly influences the salient region recognition step, given that the database search ends after the first match is found. This is especially true because the number of SIFT keypoints in salient region windows is lower than if we were to use the whole scene (as low as single digits). Although this speeds up the process by having to compare less data, it may have a downside of turning up incorrect matches (false positives). However, this is where additional context cues that can be matched at a faster rate (for example, gist vector, which encode the coarse layout of the image) can minimize the errors. If there is a coincidental match, a number of independent factors would also have to be in agreement. And, in order to significantly perturb the system, because of the use of Monte Carlo localization, that match would have to persist for some time.

It should be noted, however, that from the random prioritization (or context information being taken out) results, it appears that salient features plus the neighborhood SIFT keypoints are enough to find correct region matching. Most of the times, in our testing, when the system returns a positive region match, it is a correct assessment. The problem with the matching method is that it has its fair number of false negatives.

On each site, using the optimum priority with early exit strategy, the system only needs to make a small number of comparisons (0.85%, 0.54%, and 0.82% of the database, respectively) to obtain salient regions matches, which is better than just using individual priority terms (segment, saliency, or location) because it has both the instantaneous appearance and temporal factor. In addition, it is encouraging to see that a high percentage of the regions is found early when compared to the eventual total found (2.50/2.77, 2.84/3.52 and 2.57/3.05 per image, respectively). This indicates that the early exit policy works, here only performing 7.35%, 3.50%, and 6.12% of the total possible comparisons; a significant speed up, when compared with the random (no context) priority: 8.04 (59.06%/7.35%), 13.42 (46.96%/3.50%), and 9.04 (55.30%/6.12%) times, respectively.

IV. DISCUSSIONS AND CONCLUSIONS

At the start we set out to optimize the speed-accuracy tradeoffs in the recognition process for vision localization. In this work, we achieved a workable compromise and done so with minimization in three different levels: the number of features associated for each entry in the database, the number of entries in the database, and the number of run-time comparisons needed to determine if there is a match.

With the use of the saliency model we can automatically identify a visually distinct part of an input image which allows the system to crop out a small window called the salient region. The effect of this operation is that we only have to compare a small subset of the SIFT keypoints for faster matching time. And because the system provides context from gist and saliency features, possible increase in false positives are minimized, as reflected in the end localization result.

To lower the number of salient regions stored in the database while still keeping all the necessary information, we play to the strength of individual entries (scale and in-plane rotation invariance) and only add new instances when they reduce the weaknesses (out-of-plane view-point and lighting changes). During individual database construction, viewpoint invariance is the main reason why we add a salient region to a landmark. Lighting invariance, on the other hand, is achieved by training the system on multiple lighting conditions. However, because there is enough lighting overlap, we are able to make connections between landmarks that depict the same point of interest and but are created in different training sessions.

The on-line landmark database search prioritization is the culmination of the benefit of using a multi-expert, multi-level approach. By prioritizing the order of salient region matching, we are able to cut the comparison percentage down to single digits. The saved computation time can be used to perform more robust and sophisticated recognition. For example, if there is an improvement needed, it would be to add visual cues that work across wider range of lightings. Also, in addition to the presented prioritization factors it would be easy to add temporal shortcuts such as always compare the previous 10 matched landmarks first. In the same spirit, we can also add recently matched session priority, which, in effect, provide lighting condition priming.

V. ACKNOWLEDGMENTS

The authors gratefully acknowledge the contribution of NSF, HFSP, NGA, and DARPA.

REFERENCES

[1] G. N. DeSouza and A. C. Kak, "Vision for mobile robot navigation: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 2, pp. 237–267, 2002.

[2] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte carlo localization: Efficient position estimation for mobile robots," in *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI'99).*, July 1999.

[3] S. Thrun, D. Fox, and W. Burgard, "A probabilistic approach to concurrent mapping and localization for mobile robots," *Machine Learning*, vol. 31, pp. 29–53, 1998.

[4] M. Agrawal and K. Konolige, "Real-time localization in outdoor environments using stereo vision and inexpensive gps," in *ICPR06*, 2006, pp. III: 1063–1068.

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

[6] S. Se, D. G. Lowe, and J. J. Little, "Vision-based global localization and mapping for mobile robots," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 364–375, 2005.

[7] L. Goncalves, E. D. Bernardo, D. Benson, M. Svedman, J. Ostrowski, N. Karlssona, and P. Pirjanian, "A visual front-end for simultaneous localization and mapping," in *ICRA*, April 18 - 22 2005, pp. 44–49.

[8] H. Tamimi and A. Zell, "Vision based localization of mobile robots using kernel approaches," in *IROS*, 2004.

[9] S. Frintrop, P. Jensfelt, and H. Christensen, "Attentional robot localization and mapping," in *"ICVS Workshop on Computational Attention & Applications (WCAA)"*, Bielefeld, Germany, March 2007.

[10] I. Ulrich and I. Nourbakhsh, "Appearance-based place rcognition for topological localization," in *IEEE-ICRA*, April 2000, pp. 1023 – 1029.

[11] P. Blaer and P. Allen, "Topological mobile robot localization using fast vision techniques," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2002.

[12] A. Torralba, K. P. Murphy, W. T. Freeman, and M. A. Rubin, "Context-based vision system for place and object recognition," in *IEEE Intl. Conference on Computer Vision (ICCV)*, Nice, France, October 2003, pp. 1023 – 1029.

[13] 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.

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

[15] C. Siagian and L. Itti, "Biologically-inspired robotics vision monte-carlo localization in the outdoor environment," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2007, td;bu;sc;mod.

[16] 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.

[17] L. Itti, "Models of bottom-up and top-down visual attention," bu;td;mod;psy;cv, Pasadena, California, Jan 2000.

[18] N. Ouerhani, A. Bur, and H. Hgli, "Visual attention-based robot self-localization," in *Proc. of European Conference on Mobile Robotics (ECMR)*, Ancona, Italy, September 2005, pp. 8–13.

[19] P. Newman and K. Ho, "Slam-loop closing with visually salient features," in *International Conference on Robotics and Automation (ICRA)*, Barcelona, Spain, 2005.