Large Area Cell Based Image Localization

Andrew Zhai
Signetron Inc. and UC Berkeley
Berkeley, CA. USA
azhai@eecs.berkeley.edu

Matthew Clements
Signetron Inc. and UC Berkeley
Berkeley, CA. USA
clements@eecs.berkeley.edu

Avideh Zakhor
Signetron Inc. and UC Berkeley
Berkeley, CA. USA
avz@eecs.berkeley.edu

***Abstract*— We present a memory scalable image localization system that uses distributed kd-trees created on overlapping geographic cells using a database of 10 million Google Street View images for an area of approximately 10,000 square kilometers in Taiwan. Given a collection of images over a region of interest (ROI), we generate a database by dynamically creating geographic cells that are optimized so that each cell contains roughly the same number of images. We then create kd-trees for each cell from SIFT features extracted from the images in that cell. When querying the system, we run traditional feature matching on each cell and pool the results for each cell to re-rank with a geometric constraint. The key idea is the subdivisions of the ROI into overlapping geographic cells, allowing our system to scale to 10 million images and to efficiently utilize prior query location information when available. We evaluate our system on a test set of 29 geo-tagged images, not from Google Street View, taken throughout Taiwan with various resolutions, aspect ratios, and qualities. We also evaluate our system on a set of 97 images without geo-tag data.**

Keywords-image matching; image retrieval; visual landmark recognition; image localization;

I. INTRODUCTION

In automatic image localization, the location of a given query image is determined by querying a pre-generated database of geo-located images to retrieve the closest visual matches. This problem has many applications such as providing location on smartphones in areas with weak GPS signal and increasing coverage for commercial products such as location based ad targeting. With the increasing availability of geo-tagged images from sources such as Google Street View and Flickr, images have the potential to localize cameras anywhere in the world. To achieve such global localization however, image localization systems must be able to scale to millions or even billions of images.

Generally image localization is done by extracting some type of local feature descriptor for all images and then obtaining feature correspondences between the query and multiple database images through an approximate nearest neighbor algorithm such as the randomized kd-tree method [3]. These feature correspondences are then pruned to resolve erroneous correspondences through a geometric consistency step.

One of the main factors contributing to the non-scalability of existing image localization systems is the large memory footprint of loading the local descriptors of all the database images, a necessary step in kd-tree based approximate nearest neighbor algorithms. In the case of loading in SIFT features [2], only around 100,000,000 features can be loaded into a 12 GB RAM as noted by [10]. Traditionally, scalable methods avoid this large memory footprint by using a vocabulary tree to quantize the local descriptors [8, 9, 10, 11], saving memory by only needing the descriptors at the inner nodes of the vocabulary tree to be loaded into memory during the approximate nearest neighbor search. Though vocabulary trees result in a smaller memory footprint, their performance have been shown to be inferior to kd-trees [7].

To circumvent the above problem, it is possible to use features that require less memory. Rather than using local features, one can use global features to create systems scalable to millions of images [4, 12]. In particular, [4] utilizes an ensemble of global features such as miniaturized raw image pixels, global color histogram, global texton histograms, etc while [12] uses only the miniaturized raw image pixels and weak annotations. While capable of returning images that globally look similar to the query image, these highly scalable global feature systems cannot match fine-grained details, resulting in poor performance in the image retrieval and localization tasks.

Our approach to a scalable image recognition system is most related to the methods proposed in [1, 6]. In [6], rather than creating one kd-tree over all images in our database, the images are subdivided with a kd-tree created for each subdivision. In [1], a geographic region is subdivided into evenly sized overlapping cells and a kd-tree is constructed over each cell. In both [1] and [6], the generic image localization algorithm is then run for each cell independently and finally the top results are aggregated amongst the various cells. The most significant difference between our proposed

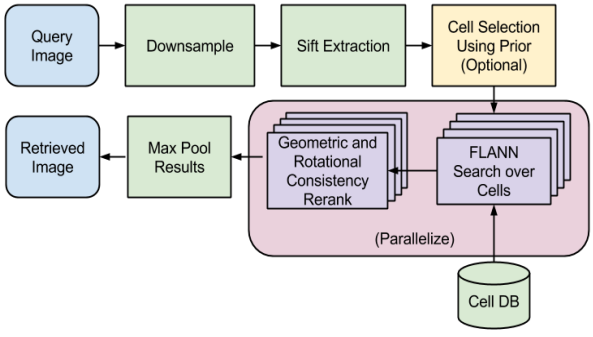
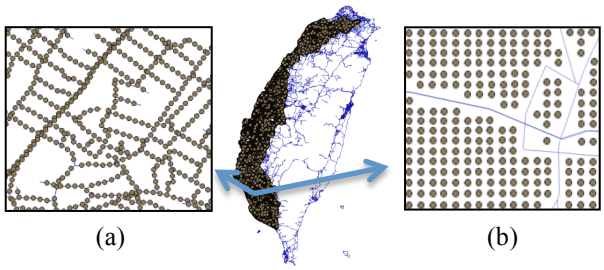


Figure 1. Overview of our image retrieval pipeline.



(a) (b)

Figure 2. Visualization of the points used for scraping. (a) Points along OpenStreetMap roads. (b) Points used for grid scrape.

method and that of [1, 6] is our novel cell generation step. Rather than randomly subdividing images into cells of equal geographic size as in [1], our subdivisions are optimized to create cells with roughly equal number of images. This results in cells with geographically non-uniform size but approximately uniform number of images, improving retrieval performance. By limiting the maximum number of images per cell in the subdivision process, we also mitigate the memory footprint problem mentioned earlier. Finally, similar to [1], this allows for efficient incorporation of prior query location information.

The remainder of the paper is outlined as follows. We provide an overview of our system in Section 2 and explain the image database generation in Section 3. In Section 4, we describe our cell generation algorithm and rationale. We evaluate our results in Section 5 and conclude in Section 6.

II. System Overview

While the motivation for our proposed geographic cell based approach comes from [1], straightforward application of the approach in [1] is infeasible due to its poor retrieval performance and memory scalability issues resulting from our database being around 1000 times larger in size than in [1]. The block diagram for our current system, shown in Fig. 1, is similar to [1] except for the cell generation step. Specifically our system consists of the following components:

A. Database Generation (Offline)

After extracting SIFT features for each image in the 10 million Taiwan image database using [14], we subdivide the images along with their features into geographic cells. For each of these geographic cells, we create a kd-tree using FLANN [3] and save the index for later use. We describe creation of these geographic cells in Section 4.

B. Feature Correspondence Voting

After extracting SIFT features from a query, we obtain the top N approximate nearest neighbors for each feature in the query, resulting in a maximum of N feature correspondences per query feature. The approximate nearest neighbor match is only accepted if the L_2 distance between the query and database feature are within a certain distance threshold. Furthermore, we discard correspondences where the difference in rotation between the query and database feature is greater than 0.2 radians. This process is repeated independently for each cell. We choose $N = 4$ based on previous benchmarks in [15].

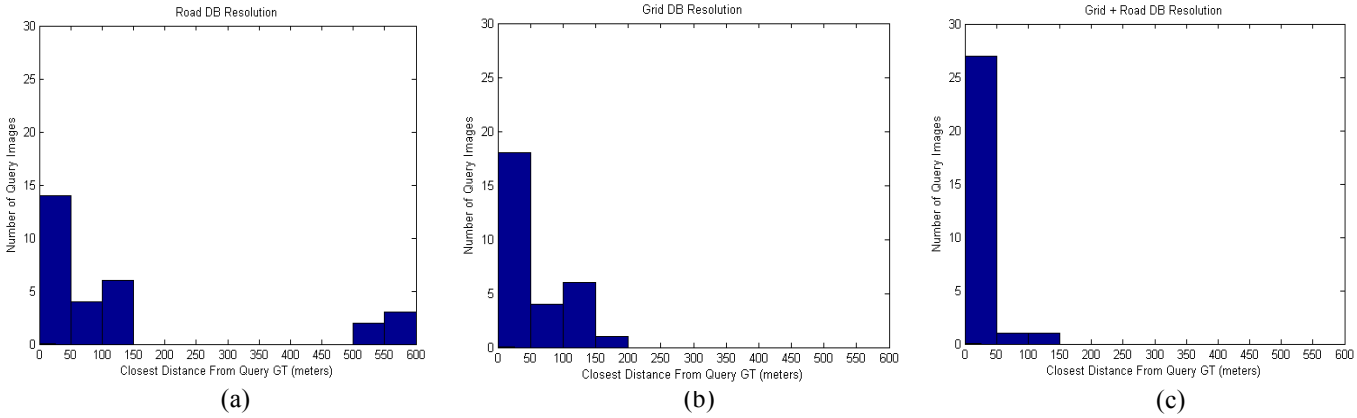
C. Filter and Pooling

After the voting step for each cell, there exist point correspondences between the query and multiple database images. Mismatched point correspondences are then pruned through a geometric consistency step where we solve for the fundamental matrix between the query and each matching database image, removing point correspondences that do not satisfy epipolar constraints. After this step is repeated for all cells, we pool the matching database images and rank them by number of point correspondences. We take the maximum number of point correspondences for database images that are duplicated in multiple cells. The top matching database image is the one with the most number of correspondences.

III. Image Database Generation

The 10 million Google Taiwan Street View images have been scrapped through the combination of two methods in order to provide adequate resolution for our geo-tagged 29 Taiwan query images, which are different from Google Street View images.

Since Google Street View images are only taken along roads, we initially sample 80 meter separated points along



(a) (b) (c)

Figure 3. Distribution of closest database image distance in meters for the 29 TW query images. (a) Distribution with only images scraped from OpenStreetMap road data. (b) Distribution with only images scraped from grid approach. (c) Distribution from the combination of (a) and (b).

OpenStreetMap's roads for Taiwan [16] as shown in Fig. 2(a). Each point results in twelve 640x640 images, evenly spanning 360° heading with 67° field of view. Even though this results in around 5 million images, the resolution is inadequate as shown in Fig. 3(a). Specifically, only 14 of 29 geo-tagged test images had an existing Google Street View image within 50 meters of the image's ground truth. This is due to the insufficient coverage of OpenStreetMap roads.

To boost the resolution, we have augmented our existing database by scraping along an 80 meter separated point grid over our region of interest, avoiding redundant work by removing points near OpenStreetMap roads as shown in Fig. 2(b). This results in an additional 5 million images after duplicates are removed for a total of 10 million images. As shown in Fig. 3(c), by combining the road and grid point scrape methods, 27 out of the 29 Taiwan test images have an existing Google Street View image within 50 meters of the image's ground truth.

IV. Cell Generation

One characteristic of our 10 million image database is that the region of interest covers both urban and rural areas in Taiwan. This results in a non-uniform geographic distribution of our database images as high-density areas such as cities have a higher concentration of Street View images per square meter than low-density ones such as villages. To handle the non-uniform geographic distribution and the database size, we propose our dynamically sized cell method.

Initial runs of image localization using the fixed radius cell division scheme described in [1] result in poor performance as the scheme in [1] relies on fixed radius geographic cells and assumes uniform geographic distribution of the database images. The number of images per cell, or cell size, varies dramatically for the fixed radius method as shown in Fig. 4(a) ranging from less than 100 images to around 200,000 images for a random one million subset of the Taiwan database. As a result, images in these small cells end up with too many votes as each cell has an equal amount of votes to distribute i.e. *# of votes = # of query features × # of nearest neighbors requested*. Furthermore, because of the degradation of kd-tree performance with database size as benchmarked in [15], correct feature correspondence matches occur less in these large ~200,000 image cells as the approximate nearest neighbor search loses precision.

As explained in [5], it is also likely for multiple visually similar features to exist in a database as the database size increases, making even the true nearest neighbor match to result in an incorrect correspondence. Furthermore, by not controlling the maximum image size of a cell, cell sizes grow, such that a single cell might not fit into memory. Specifically, assuming an average of 1000 SIFT features per image, the 200,000 image cell would require ~26 GB of RAM. Thus, to achieve memory scalability, it is desirable to devise a scheme that allows for non-uniform geographic cell size while imposing some uniformity on the number of images within cells.

In an abstract sense, the division of our database into cells is a form of clustering, where we assign geo-tagged images

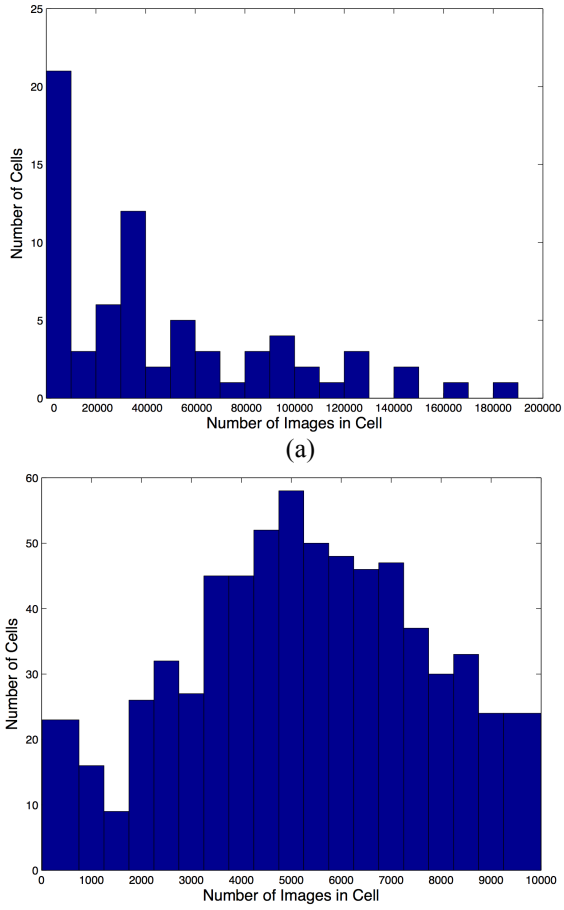


Figure 4. Fixed radius cell size distribution. (b) Dynamic radius cell size distribution showing the max at n_{max} = 10000. Limiting cell size guarantees that all cells fit in memory. Both were generated from the 1 million Taiwan subset database.

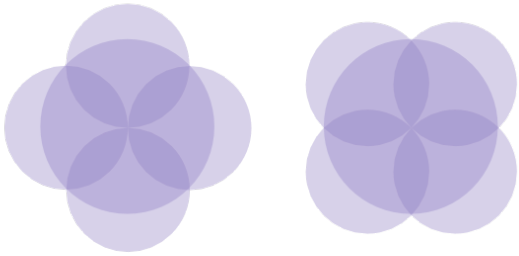


Figure 5. Visualization of (a) cardinal split and (b) diagonal split.

to clusters based on geographic proximity. This is especially useful for our system as it is designed to run not only on full ROI but also on smaller context regions that represent a priori knowledge of the query's position. This prior knowledge can be used to prune the number of cells to be queried, as there is no point to query any circular cells not intersecting with the region defined by a priori location knowledge. The prior knowledge can again be applied after computing the top result list for fine-grained pruning of spe-

cific latitude/longitude points as the context region usually only intersects with a portion of a given cell.

In choosing between divisive and agglomerative clustering approaches, we have determined that agglomerative clustering would result in cells of unusual shapes, making it more difficult to incorporate prior knowledge since such cell shapes would be difficult to characterize and analyze. Therefore, we have opted for a divisive approach, described as follows.

The basic idea behind our approach is to split each cell into four children cells per iteration until the number of images in every cell falls below a given threshold. In doing so, we alternate between cardinal and diagonal splits in order to avoid unbounded growth in cell overlap as shown in Fig. 5. In cardinal splits, children are offset either horizontally or vertically from their parents. In diagonal splits, they are offset diagonally. The main motivation for the above approach is that excessive cell overlap has been shown not to improve or even degrade retrieval performance, while increasing computation and memory requirements [15].

We describe a cell c by its *center*, its *radius*, the *orientation* in which splits, and the *images* it contains. A list of cells is referred to as *cell_list*. Intuitively orientation of a cell encodes whether it is split cardinally or diagonally. Specifically the orientation of a cell is either 0^o or 45^o depending upon whether it is cardinal or diagonal respectively. Mathematically, orientation of a given cell is the angle from that cell's center to the center of its first child with respect to the horizontal axis.

We initialize *cell_list* to contain the smallest single circular cell of radius r_0, center $[x_0, y_0]^T$, and orientation 0^o, such that it contains every geo-tagged image in our database. The images in a cell c are referred to by *c.images*. The other attributes of a cell are denoted by a similar notation. As long as there are any cells c in *cell_list* with more than n_{max} images, we select one and split it into four children cells $c_0, c_1, c_2,$ *and* c_3 with radii equal to $c.radius / \sqrt{2}$. The center of each child cell is a distance $c.radius / \sqrt{2}$ from *c.center*. In a cardinal (diagonal) split the angle between the line connecting the center of the first child to its parent and the horizontal axis is 0^o (45^o). The angle measured at the center of the parent cell between two successive children in both cardinal and diagonal is 90^o.

After each parent cell is decomposed into four child cells, we assign to each child all database images lying within its geographic extent using the database images' geo-tagged location. We then add that child to the *cell_list* provided it has non-zero number of images and an equivalent cell does not already exist in the *cell_list*. The reason for the latter is pictorially explained in Fig. 6 which shows that of any cell's 16 grandchildren, four are duplicates, resulting in only 12 unique grandchildren.

The main motivation for the alternation between cardinal and diagonal in successive splits is to control the amount of overlap in children cells. Specifically, it can be shown that ignoring edge effects, the alternation procedure results in the overlap to remain constant as the iterations proceed as

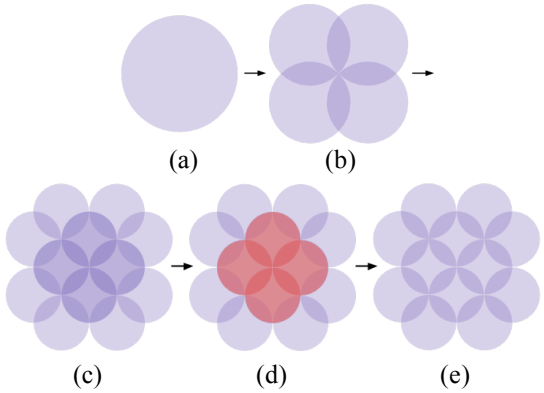


Figure 6. Division of a cell into 4 cells via a diagonal split followed by four additional cardinal splits; (a) parent cell; (b) four children cells resulting from diagonal split; (c) 16 grandchildren cells resulting from four cardinal splits of children cells in (b); (d) Four pairs of duplicate cells in (c) highlighted in red. (e) remaining grandchildren after removal of duplicates in (d).

duplicates are deleted. Without the alternation though, these duplicates do not occur and hence, the amount of overlap in the center region of the original cell continues to increase with the number of iterations. This behavior is undesirable as images in these highly overlapping regions have an unfair and artificial advantage when tallying votes since they occur in a multitude of cells.

Quantitatively, it can be shown that splitting a cell of radius r_0 and area πr_0^2 into 4 overlapping cells of radius $r_0 / \sqrt{2}$ in the manner described above, results in an overlap area of $r_0^2(\pi - 2)$. If we define ρ to be the ratio between sum of the areas of the cells to the area of the union of the cells, it can be shown that for the alternation approach, ρ asymptotically approaches π/2 from below as depth of the tree, d, increases. In contrast without alternations, ρ is bounded by [17]:

$$\left(\frac{2\pi}{\pi+2}\right)^{d-1} \approx 1.2220^{d-1} < \rho < 2^{d-1} \qquad (1)$$

Therefore, as long as $d > 2$, ρ is guaranteed to be smaller for the alternating method than even the lower bound for the non-alternating method.

The size distribution of the dynamically sized cells for n_{max} = 10,000 is shown in Fig. 4(b), resulting in a soft maximum on memory usage. In particular, assuming 1000 SIFT features per image, the maximum dynamic cell size needs ~1.3 GB of memory, ensuring that such cells can be loaded in RAM unlike the ~26 GB cell from the fixed radius method. We see in the histogram that there are cells with fewer than 500 images. Though this indeed does create an unfair voting environment as described earlier, the improved approximate nearest neighbor performance from the larger cells with the maximum cell size results in more matching features in these cells, mitigating the effects of the unfair voting. Furthermore, even though cells with few images can have an inflated vote tally as compared to other cells, usually the geometric constraint step eliminates many false matches. With the combination of these two, even though smaller cells do have an advantage, they do not overwhelm the top results.

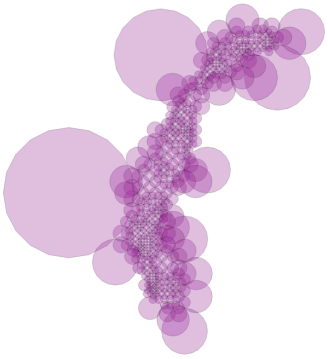


Figure 7. Dynamic radius cells visualized on top of Taiwan Region of Interest for 1 million image subset db.

A visualization of the dynamic radius geographic cells on our region of interest is shown in Fig. 7 on a 1 million image subset of our 10 million image database. The maximum and minimum cell radii in Figure 7 are 70 km and 1 km respectively.

V. EVALUATION

We evaluate our retrieval system using the 10 million image Taiwan Street View database described in Section 3 on a set of 29 geo-tagged Taiwan ground-level images that do not originate from Google Street View with 18 urban images and 11 rural images. Examples of urban and rural images are shown in Figs. 8(c) and 9(a) respectively. Our test images also have a variety of resolutions ranging from 5 to 20 times the size of the Street View database images. Furthermore, the quality of the images varies significantly. As seen, Figs. 8(a) and 8(c) are high quality DSLR images while test image 8(b) exhibits much lower quality with both decreased resolution and motion blur. Some test images are taken through car windows as shown in Figs. 9(b) and 9(d). Though the quality and environment of our test images vary significantly, they are all possible queries to a real world image localization system and as such allow us to estimate our system's practical real world performance. To measure image retrieval performance, in Table I we show the percent of queries that are localized within 80 meters of the ground truth in top N retrievals for N = 1, 5, 10, and 50.

To augment our test set, we combine our 29 geo-tagged Taiwan ground level images with 68 additional Taiwan ground level images of similar origins. Since these 68 images are not geo-tagged, to measure retrieval performance on this combined 97 image test set, we manually check the contents of the retrieved images for a visual match with the query images. The percentage of queries with correct matches in the top N retrievals is shown in Table I for N = 1, 5, 10, and 50.

For both the 29 and 97 test sets, the successfully retrieved queries all correspond to urban regions. Figs. 8 and 9 show successful and failed retrievals respectively depending on whether or not the matching database image to a query is within the top 5 images returned by our system. We have found that images with obviously distinct features existing in both the query and database images such as a brand or text

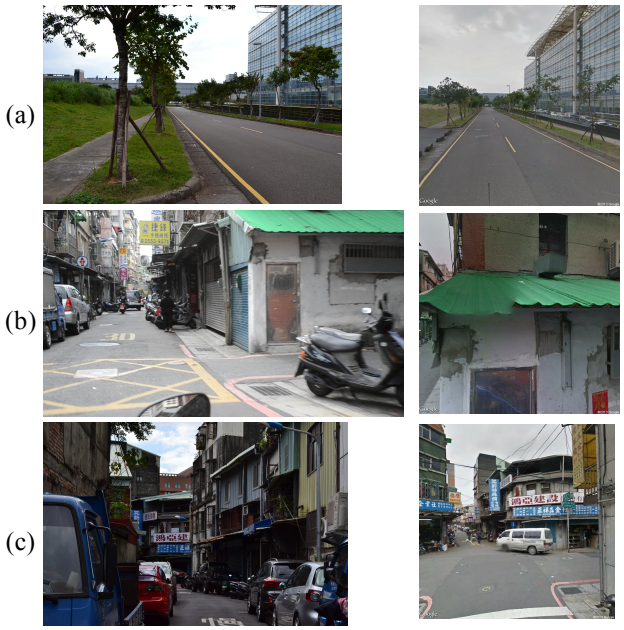


Figure 8. Successful results with query on the left and the retrieved Google Street View image on the right. Image stitching artifacts can be seen on the right image of (b).

label result in excellent localization due to the distinct features resulting from these texts. This is also seen in Fig. 8(b) with the huge crack on the wall underneath the green roof. The lack of distinct matching features is the most significant reason for failures in localizing test images.

Eleven of our test images in the 29 image test set from rural environments are filled with non-rigid vegetation resulting in local descriptors that are likely to not match between query and the correct matching database image as the features detected over vegetation vary with environmental factors such as weather conditions. Furthermore, vegetation is likely to change over time as shown in Fig. 9(a) where the color and shape of the grass fields between the query and database images differs. Though the database image in Fig. 9(d) does seem likely to have discriminative features, the motion blur and low quality of the query image due to being taken through a car window hinders such features from being generated and robustly matched.

Finally we compare the performance of our proposed method to that of the fixed radius method in [1] on our 29 geo-tagged image test set. Unlike the proposed dynamic subdivision cell approach, the fixed radius cells method in [1] results in cells with kd-trees that are too large to fit in memory for a 10 million image database. As such, we opt to compare the two methods on a 1 million image database created by first taking a random subset of the 10 million image database, and then adding database images corresponding to the 29 image test sets in order to ensure existence of a match for all test images. We show performance results on the test set while considering top 1 and 5 retrieved images for both methods in Table II. As seen, our proposed method outperforms the fixed radius method even for a smaller database of 1 million images.

Figure 9. Failed results with query on the left and the closest matching Google Street View image on the right. We can see that test images taken in (b) and (d) are from within a car and exhibit motion blur distorting the distinctive features between query and database match. Images (a) and (c) also show the varying nature of vegetation between the query image and the database image.

TABLE I. PERCENTAGE OF QUERIES WITH CORRECT MATCH AMONG TOP *N* RETRIEVED IMAGES FOR THE PROPOSED METHOD FOR A 10 MILLION IMAGE DATABASE

Test set	**Top *N* Retrieved Images**			
	N = 1	***N = 5***	***N = 10***	***N = 50***
29 images	10.3%	20.7%	24.1%	24.1%
97 Images	6.19%	17.5%	19.6%	21.6%

TABLE II. RETRIEVAL PERFORMANCE COMPARISON OF FIXED RADIUS CELL AND PROPOSED DYNAMIC CELL METHOD FOR A 1 MILLION IMAGE DATABASE

Method on 29 image test set	**Top *N* Retrieved Images**	
	N = 1	***N = 5***
Fixed Radius [1]	13.8%	13.8%
Dynamic Radius	**20.7%**	**24.1%**

VI. CONCLUSION

In this paper we have presented a memory scalable image localization system using dynamically generated geographic cells over a region of interest. By using a novel cell division algorithm, we are able to control the maximum number of images per cell, reducing the memory footprint of each cell so that each cell can be loaded into memory. This allows us to use the robust but memory intensive SIFT features. Furthermore, by directly controlling the size of each cell, we avoid the degradation of performance with increasing database size of the approximate nearest neighbor algorithm using kd-trees from FLANN [3, 15]. Along with allowing our system to scale to 10 million images, these geometric cells have also allowed us to efficiently utilize prior query location knowledge.

ACKNOWLEDGMENT

Supported by the Intelligence Advanced Research Projects Activity (IARPA) via Air Force Research Laboratory, contractFA8650-12-C-7211. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, AFRL, or the U.S. Government.

REFERENCES

[1] J. Zhang and A. Hallquist, “Location-based image retrieval for urban environments,” *ICIP,* vol. 50, no. 150, pp. 1–4, 2011.

[2] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” in *IJCV*, 60, 2, pp. 91-110, 2004.

[3] M. Muja and D. Lowe, “Fast approximate nearest neighbors with automatic algorithm configuration.,” *VISAPP (1)*, 2009.

[4] J. Hays and A. Efros. “Im2gps: estimating geographic information from a single image,” In *CVPR 2008,* 2008.

[5] Y. Li, N. Snavely, D. Huttenlocher, and P. Fua, “Worldwide pose estimation using 3d point clouds,” *Comput. Vision–ECCV 2012*, 2012.

[6] M. Aly, M. Munich, and P. Perona, “Distributed kd-trees for retrieval from very large image collections,” *Br. Mach. Vis. Conf. ...*, pp. 1–11, 2011.

[7] M. Aly, M. Munich, and P. Perona, “Indexing in large scale image collections: Scaling Properties, Parameter Tuning, and Benchmark,” pp. 1–35, 2010.

[8] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, “Object retrieval with large vocabularies and fast spatial matching,” *2007 CVPR*, 2007.

[9] D. Nister and H. Stewenius, “Scalable recognition with a vocabulary tree,” *2006 CVPR*, 2006.

[10] G. Schindler, M. Brown, and R. Szeliski, “City-Scale Location Recognition,” *2007 IEEE CVPR*, 2007.

[11] J. Sivic and A. Zisserman, “Video Google: a text retrieval approach to object matching in videos,” *Int. Conf. Comput. Vis.*, 2003.

[12] A. Torralba, R. Fergus, and W. T. Freeman. “Tiny images.” Technical Report MIT-CSAIL-TR-2007-024, 2007.

[13] C. Valgren, “SIFT , SURF and Seasons : Long-term Outdoor Localization Using Local Features,” vol. 128, pp. 1–6.

[14] http://cs.unc.edu/~ccwu/siftgpu/

[15] E. Liang and A. Zakhor, “Structuring a Sharded Image Retrieval Database,” *IS&T/SPIE Electron. Imaging*, pp. 4–7, 2013.

[16] http://www.openstreetmap.org

[17] A. Zhai, M. Clements, and A. Zakhor, "Scalable cell based image localization", Technical report, September 2014, U.C. Berkeley. http://www-video.eecs.berkeley.edu/papers/azhai/andrew2014miprfinal.pdf, unpublished