Scalable Cell Based Image Localization

Andrew Zhai, Matthew Clements, and Avideh Zakhor
Signetron Inc. and University of California, Berkeley
Berkeley, CA 94720
{azhai,clements,avz}@eecs.berkeley.edu

Abstract

We present a 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 rerank 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.

1. 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, image localization has the potential to localize any image 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 localization task.

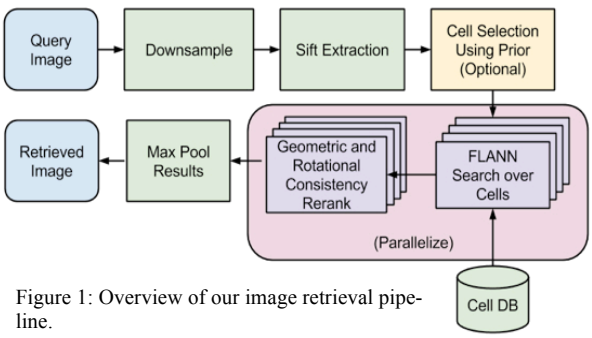
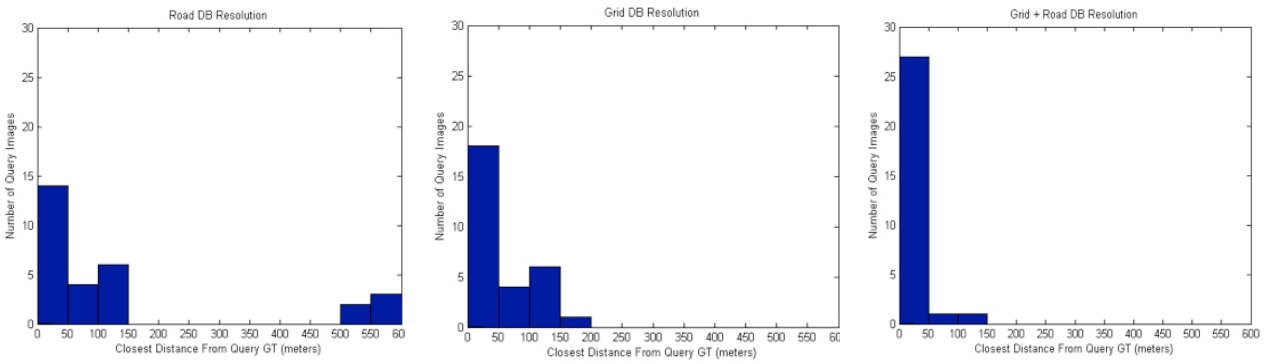


Figure 1: Overview of our image retrieval pipeline.



(a) (b) (c)

Figure 2: Distribution of closest database image distance in meters for the 29 TW queries. (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

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 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.

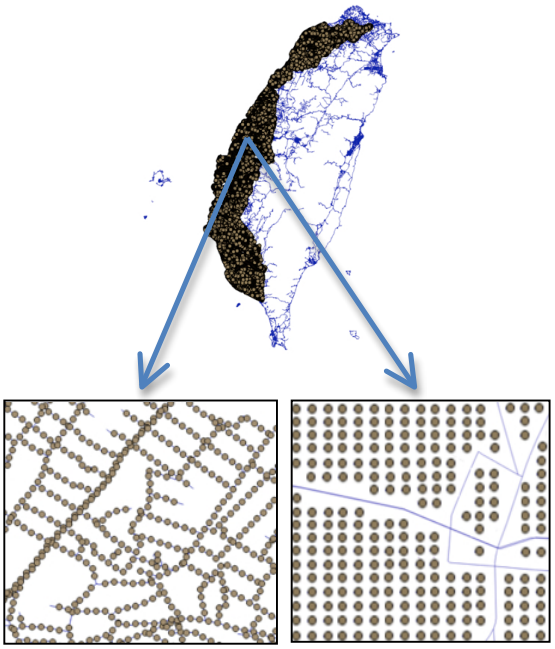
2. System Overview

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 Figure 1, is similar to [1] except for the cell generation step. Specifically our system consists of the following components:

1) 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.

2) 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].

3) 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



(a) (b)

Figure 3: Visualization of the points used for scraping. (a) Points along OpenStreetMap roads. (b) Points used for grid scrape.

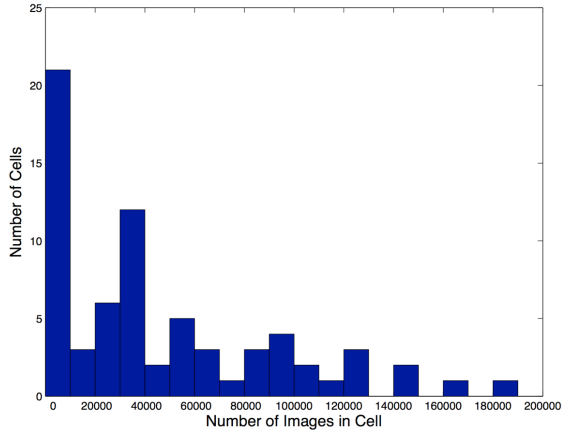
the matching database images and rank them by number of point correspondences. We take the maximum number of point correspondences for database images that are duplicated in multiple cells. The top matching database image is the one with the most number of correspondences.

3. Image Database Generation

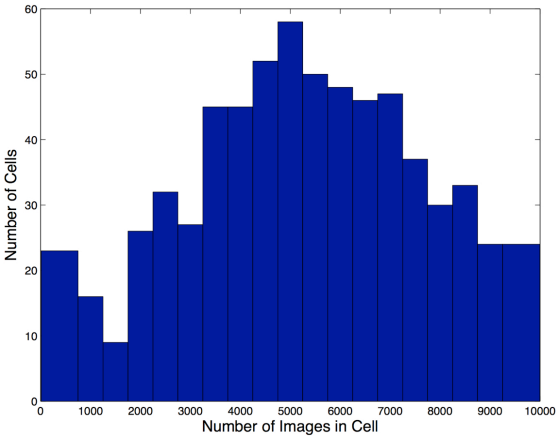
The 10 million Google Taiwan Street View images have been scrapped through the combination of two methods in order to provide adequate resolution for our geo-tagged 29 Taiwan query images, which are different from Google Street View images.

Since Google Street View images are only taken along roads, we initially sample 80 meter separated points along OpenStreetMap's roads for Taiwan [16] as shown in Figure 3(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 Figure 2(a). Specifically, only 14 of 29 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



(a)



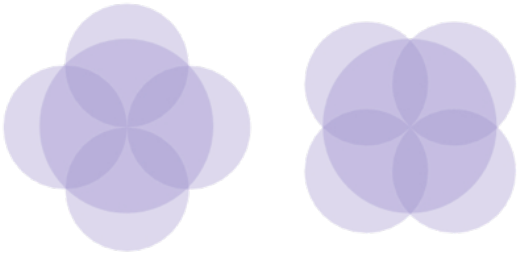
(b)

Figure 4: (a) 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.

redundant work by removing points near OpenStreetMap roads as shown in Figure 3(b). This results in an additional 5 million images after duplicates are removed for a total of 10 million images. As shown in Figure 2(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.

4. Cell Generation

Initial runs of image localization using the fixed radius cell division scheme described in [1] result in poor performance due to the non-uniform geographic distribution of database features. This non-uniformity is due to the span of our region of interest covering both urban and rural areas in Taiwan as high-density areas such as cities have a higher concentration of Street



(a) (b)

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

View images per square meter than low-density ones such as villages. Since the scheme in [1] relies on fixed radius geographic cells, the number of images per cell, or cell size, varies dramatically as shown in Figure 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 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 ~24 GB of RAM. Thus, to achieve scalability, it is desirable to devise a scheme that allows for non-uniform geographic cell size while imposing some uniformity on the feature counts within cells.

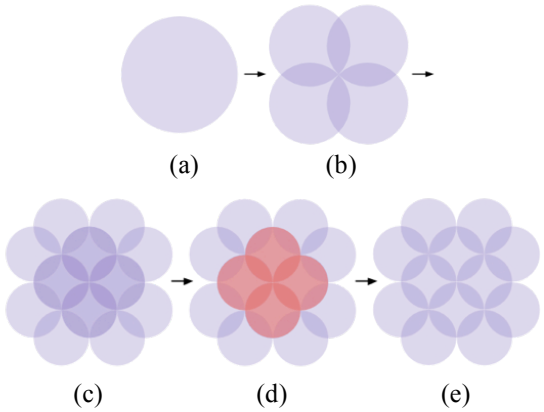
In an abstract sense, the division of our database into cells is a form of clustering, where we assign elements to clusters based on geographic proximity. This is especially useful for our system as it is designed to run not only on full ROI but also on smaller context regions that represent a priori 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 specific 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 features 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 Figure 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 *features* 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 feature in our database. The features in a cell c are referred to by *c.features*. The other attributes of a cell are denoted by a similar



(a) (b) (c) (d) (e)

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).

notation. As long as there are any cells c in *cell_list* with more than n_{max} features, 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 features lying within its geographic extent. We then add that child to the *cell_list* provided it has non-zero number of features and an equivalent cell does not already exist in the *cell_list*. The reason for the latter is pictorially explained in Figure 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 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 shown in Appendix A to be bounded by:

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

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. In Appendix B, we provide bounds tighter than those shown in Equation 1.

The size distribution of the dynamically sized cells for n_{max} = 10,000 is shown in Figure 4(b). 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.

A visualization of the dynamic radius geographic cells on our region of interest is shown in Figure 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.

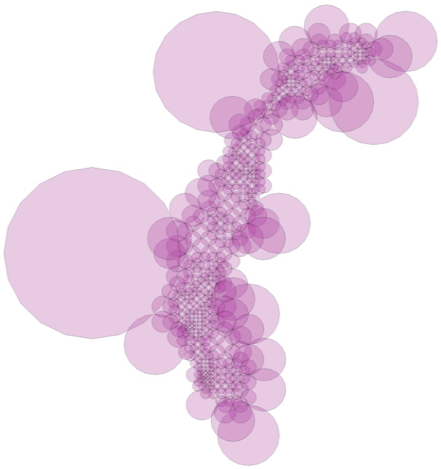


Figure 7: Dynamic radius cells visualized on top of Taiwan Region of Interest for 1 million image subset db.

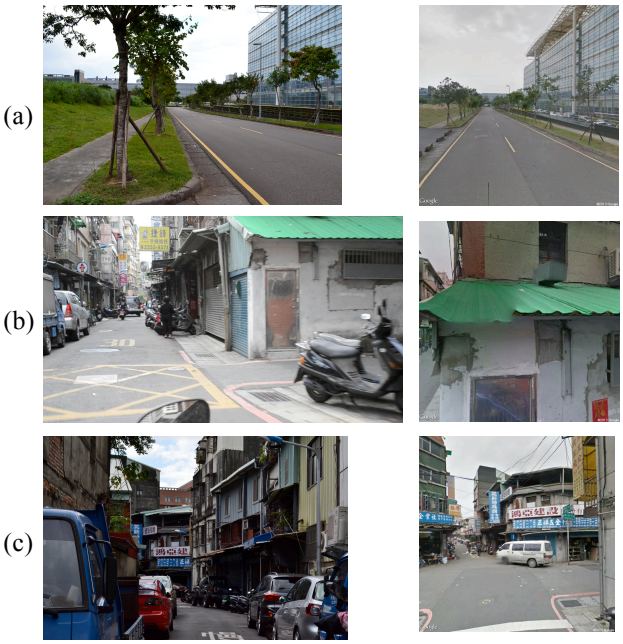


Figure 8: Successful results with query on the left and the retrieved Google Street View image on the right. Image stitching artifacts can be seen on the right image of (b).

5. Evaluation

We evaluate our retrieval system using the 10 million image Taiwan Street View database described in Section 3. Since the fixed radius cells approach in [1] results in cells with kd-trees that are too large to fit in memory, we have opted not to compare our proposed dynamic subdivision cells approach to [1] for our 10 million image database.

We evaluate our system on a set of 29 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 Figures 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, Figures 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 Figures 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.

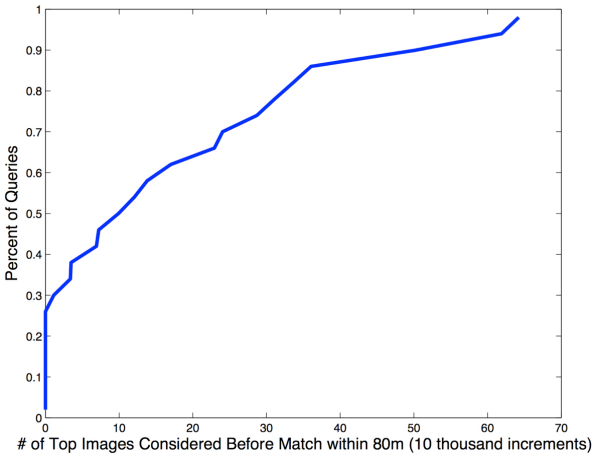


Figure 10: Distribution of the fraction of queries having matches within 80m of ground truth while varying the top number of images considered.

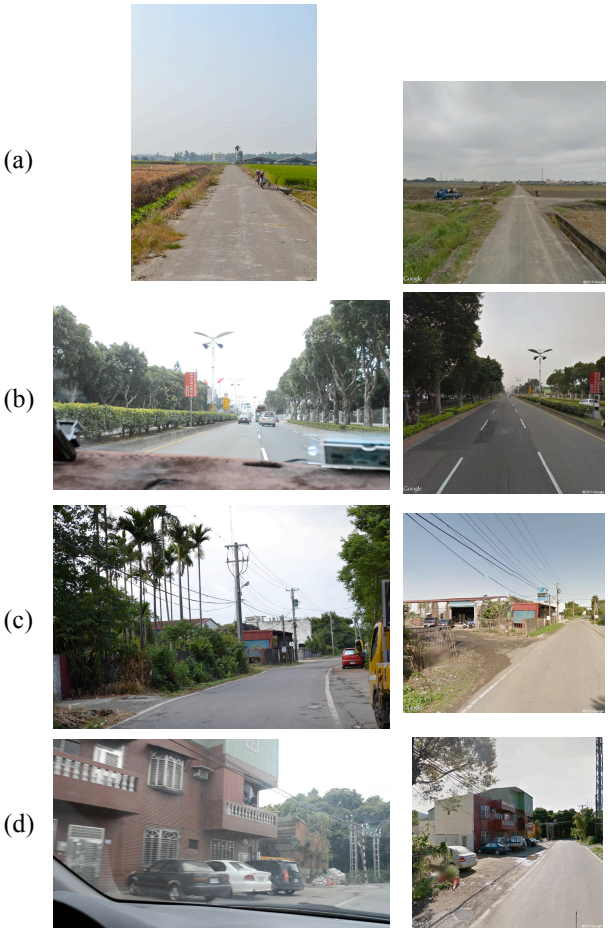


Figure 9: Failed results with query on the left and the closest matching Google Street View image on the right. We can see that test images taken in (b) and (d) are from within a car and exhibit motion blur distorting the distinctive features between query and database match. Images (a) and (c) also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

To measure retrieval performance, in Figure 10 we plot the percent of queries having matches within 80 meters of the ground truth location against the number of top ranked images considered. As seen, around 25% of our queries can be matched by only considering the top 5 images. These test images all correspond to urban regions. Figures 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 label result in excellent localization due to the distinct features resulting from these texts. This is also seen in Figure 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 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 Figure 9(a) where the color and shape of the grass fields between the query and database images differs. Though the database image in Figure 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.

6. Conclusion

In this paper we have presented a 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.

Appendix A: Loose Bounds

This appendix characterizes the level of overlap between cells for a database with no alternation of split orientation and deletion of duplicate cells.

Let d denote the uniform depth of the dendrogram of cell divisions within the database, with a single cell undivided corresponding to $d = 1$, and let database overlap be characterized by

$$\rho = \frac{\sum_{c \in cells} area(c)}{area(\cup_{c \in cells} c)}$$

where c denotes a cell and *cells* denotes the set of all cells in the database. We will show:

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

These loose bounds arise from a pair of simplifying approximations, one of which is guaranteed not to overestimate ρ, resulting in a lower bound, the other of which is guaranteed to overestimate, resulting in an upper bound. Note that for $d = 1$, a single cell, $\rho = 1$.

For the lower bound, the simplifying assumption is to assume that the descendants of one cell might overlap the descendants of another, such that we only need to consider the overlap between siblings when computing ρ. Supposing that we start with a single cell of radius r_0, its four children each have area $\frac{\pi r_0^2}{2}$ for a summed area of $2\pi r_0^2$. The area of the intersection between two adjacent children, one "petal" such as is visible in Figure 5, is half the difference between the area of one child cell and the square inscribed within it: $\frac{1}{2}\left(\frac{\pi r_0^2}{2} - r_0^2\right)$. As there are four such "petals", and no other intersections between the children, inclusion-exclusion gives an area of the union of the four cells of $\pi r_0^2 + 2r_0^2$. Thus, each increment of the depth d results in a multiplicative $\frac{2\pi}{\pi+2}$ increase in ρ, making $\rho \geq \left(\frac{2\pi}{\pi+2}\right)^{d-1}$.

For the upper bound, the simplifying approximation is to ignore the fact that the descendants of a given cell partially occupy regions that the original cell did not, i.e. we treat the denominator of ρ as constant. As shown above, the numerator increases by a factor of 2 after each level of division, resulting in an upper bound of $\rho < 2^{d-1}$.

Appendix B: Tight Bounds

This appendix provides a tighter bound than that of Appendix A. Again, letting d refer to the depth of division and ρ to the level of database overlap, tighter bounds on ρ will be shown to be:

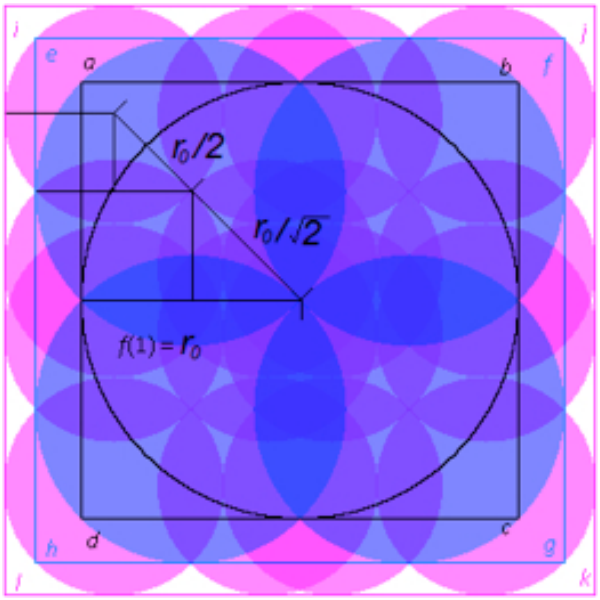


Figure 11: A diagram of uniformly oriented splits and their bounding boxes. The black line segments, along with some simple geometry, allow us to compute the sizes of the bounding boxes.

$$\frac{\pi 2^{d-1}}{4\left(\frac{3}{2}+\sqrt{2}-\frac{1}{\sqrt{2}^{d-2}}-\frac{1}{\sqrt{2}^{d-1}}+\frac{1}{2^{d}}\right)} < \rho < \frac{\pi 2^{d-1}}{4\left(\frac{3}{2}+\sqrt{2}-\frac{1}{\sqrt{2}^{d-3}}-\frac{1}{\sqrt{2}^{d-2}}+\frac{1}{2^{d-1}}\right)} \quad (2)$$

Both of these bounds are based on the observation that when all splits are in the same orientation, be it cardinal or diagonal, the circular cells occupy an area that approximates a square as d increases. The diagonal case is shown in Figure 11.

The minimal bounding box around the union of the cells – for example, the square *ijkl* in Figure 11, around the union of the magenta cells or square *efgh* around the blue – provides an overestimate of the area of that union, ρ's denominator. Substituting the area of that bounding box for the area of the union thus provides an underestimate of ρ itself.

For a database of depth d, we define $f(d)$ to be the perpendicular distance from the center of the database to one side of the square bounding box around it; this value can be thought of as the "radius" of the square, and is equivalent to one half the length of a side of that bounding box. Clearly, for an initial cell radius of r_0, we have $f(1) = r_0$. As shown in Figure 11, we have $f(2) = \frac{r_0}{2} + \frac{r_0}{\sqrt{2}}$ and the difference between $f(2)$ and $f(1)$ is $r_0\left(\frac{\sqrt{2}-1}{2}\right)$. The difference between $f(3)$ and $f(2)$ is $\frac{1}{\sqrt{2}}$ times the preceding difference, a pattern that continues as d increases, which allows f to be expressed in terms of a geometric sum:

$$\begin{aligned} f(d) &= r_0\left(\frac{1}{\sqrt{2}} + \sum_{i=1}^{d}(\sqrt{2}-1)\frac{1}{\sqrt{2}^{i}}\right) \\ &= r_0\left(\frac{1}{\sqrt{2}} + \sum_{i=0}^{d}(\sqrt{2}-1)\frac{1}{\sqrt{2}^{i}} - (\sqrt{2}-1)\right) \\ &= r_0\left(\frac{1}{\sqrt{2}} + (\sqrt{2}-1)\left(\frac{1-\frac{1}{\sqrt{2}^{d+1}}}{1-\frac{1}{\sqrt{2}}}\right) - (\sqrt{2}-1)\right) \\ &= r_0\left(\frac{1}{\sqrt{2}} + \sqrt{2}\left(1-\frac{1}{\sqrt{2}^{d+1}}\right) - (\sqrt{2}-1)\right) \\ f(d) &= r_0\left(\frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}^{d}} + 1\right) \end{aligned}$$

In terms of f, the area of the bounding square at depth d is $4f^2(d)$, or $4\left(\frac{3}{2}+\sqrt{2}-\frac{1}{\sqrt{2}^{d-3}}-\frac{1}{\sqrt{2}^{d-2}}+\frac{1}{2^{d-1}}\right)$. The sum of the areas of the cells within that square is once again $2^{d-1}\pi r_0^2$, leading to our lower bound on ρ.

For the upper bound, we substitute the area of the maximum square entirely contained within the union of the cells. Conveniently, this maximum interior square – square *efgh* for the magenta cells in the figure, or *abcd* for the blue – turns out to be the minimum bounding box of the cells in the previous level of division so its area is $4f^2(d-1)$. This gives us our upper bound.

Noting that $f(d) \to r_0\left(1+\frac{1}{\sqrt{2}}\right)$ as $d \to \infty$, we can declare ρ to be $\Theta(2^d)$.

Acknowledgement

Supported by the Intelligence Advanced Research Projects Activity (IARPA) via Air Force Research Laboratory, contract FA8650-12-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," *Image Process. (ICIP), ...*, vol. 50, no. 150, pp. 1–4, 2011.

[2] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," in International Journal of Computer Vision, 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 Computer Vision and Pattern Recognition, 2008. *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,” *Proc. Ninth IEEE 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/