PROGRESSIVE SPATIAL CLUSTERING OF CONTENT-BASED SATELLITE IMAGERY RETRIEVAL RESULTS

Matt Klaric and Chi-Ren Shyu
Center for Geospatial Intelligence
University of Missouri-Columbia

1 PROBLEM INTRODUCTION

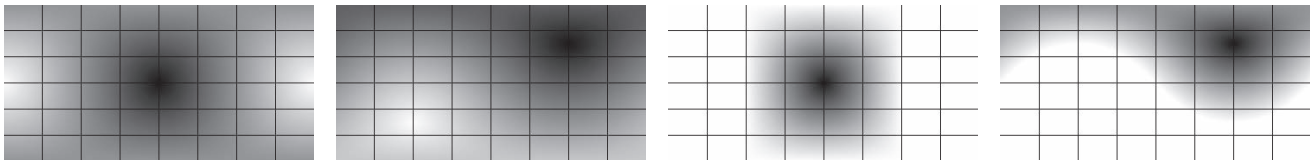
Several systems exist for performing search operations on large databases of satellite imagery. These content-based image retrieval (CBIR) systems retrieve imagery via a query-by-example mechanism [1] or by computationally analyzing pairs of images to detect change that has occurred between images [2, 3]. The results returned by these systems have both spatial and a non-spatial components–each of which are independently useful to users. However, when performing queries with these systems users may not only be interested in the content of the results returned but also how these results relate to one another spatially.

Existing clustering algorithms are capable of clustering the results based on their spatial dimensions; DBSCAN [4] is one of many algorithms that could be applied to this problem. Moreover, a few papers have addressed the problem of attempting to simultaneously cluster data in its spatial and non-spatial domains [5, 6]. However, these approaches fail to consider the unique aspects of clustering results in an image retrieval system. The results returned by these systems are presented sequentially or progressively. For example, an imagery retrieval system does not rank the entire contents of the database, but instead only returns the top-k results. The user is frequently given the ability to request additional sets of k results. Accordingly, the algorithm that we present performs an on-demand clustering of all of the returned search results as additional sets of k results are retrieved.

2 TECHNICAL METHODOLOGY

The ProgressiveDBSCAN algorithm is an adaptation of the original DBSCAN algorithm [4] that is capable of progressively clustering data as additional points are added to the dataset. This algorithm clusters data based on similarity in both a spatial and non-spatial domain. Assuming that the non-spatial measure is a distance value from a CBIR query, a similarity measure that can be used to compare the two distances is shown in Eq. 1. A similar function could be defined to compare change scores from a change detection system.

$$f(\vec{x}, \vec{y}) = \frac{1}{|\vec{x_c} - \vec{y_c}| + 1} \tag{1}$$



(a) 0°N 0°E (b) 45°N 90°E (c) 0°N 0°E (d) 45°N 90°E

Figure 1: Examples that show great circle distances from two given points. Figures 1a and 1b use a localizer value of 0 and Figures 1c and 1d use a localizer value of 0.5. Darker colors indicate smaller distances while lighter colors indicate larger distances.

The distance between any two points on Earth can be found using the equation shown in Eq. 2 where r is, 6371.01 km, the mean radius of Earth, ϕ_a and ϕ_b are the latitude of the first and second point respectively, and $\Delta\lambda$ is the difference between the longitudes of the two points. This equation calculates what is known as the Great Circle Distance.

$$c(\vec{x},\vec{y}) = r \cdot \arctan\left(\frac{\sqrt{(\cos\phi_b \sin\Delta\lambda)^2 + (\cos\phi_a \sin\phi_b - \sin\phi_a \cos\phi_b \cos\Delta\lambda)^2}}{\sin\phi_a \sin\phi_b + \cos\phi_a \cos\phi_b \cos\Delta\lambda}\right) \tag{2}$$

Equation 2 calculates a distance in km, but the algorithm only needs a relative measure of distance. Thus, in order to normalize the distance, Eq. 3 will be used. In this equation h is 20015.11 km, the mean semicircumference of Earth, and l is a localizer in the range $[0, 1]$. The effect of the localizer is that larger values emphasize results that are closer to a given point; this effect can be seen in Figure 1. It should also be noted that the value of Eq. 3 is clamped to the range $[0, 1]$.

$$g(\vec{x},\vec{y}) = \frac{c(\vec{x},\vec{y})}{h \cdot (1-l)} \tag{3}$$

Finally, an aggregate similarity score can be defined according to Eq. 4. This similarity score allows the parameter b to be tuned in order to balance the effect of spatial and non-spatial domains.

$$s(\vec{x},\vec{y}) = b \cdot f(\vec{x},\vec{y}) + (1-b) \cdot g(\vec{x},\vec{y}) \tag{4}$$

Using the definition of the similarity score in Eq. 4 the ProgressiveDBSCAN algorithm can now be defined. The basic outline of our algorithm can be found in Alg. 1. For each point that is presented to the algorithm a list of neighboring points is collected. If the number of points found is greater than a threshold, then a new cluster is created. Lastly, a function to merge adjacent clusters should be run occasionally; this is needed during the progressive nature of the algorithm. It does not need to be executed during every iteration; for example, if results are returned in groups of size k, then it is likely sufficient to execute this function every k iterations. (Due to space limitations this algorithm does not appear in the extended abstract, but will appear in the final paper.)

The *createCluster* function shown in Alg. 2 is the process that actually creates a new cluster and identifies the neighboring points that should make up that cluster. Based on the DBSCAN algorithm, the *createCluster* function evaluates all points in a neighborhood around the point in question. For each of

Algorithm 1 ProgressiveDBSCAN (p, eps, minpts)

Require: $eps \geq 0 \vee minpts > 0$
  $N = getNeighbors(p, eps)$
  if $|N| > minpts$ **then**
    $createCluster(p, N, eps, minpts)$
    if it's been a while **then**
      mergeClusters(C, eps, minpts)
    end if
  end if

these points, their neighborhood will be searched to find additional candidate points. If the neighborhood around each point is sufficiently dense–has more points than the value of the parameter *minpts*–then those points are added to the cluster if they do not already belong to another cluster.

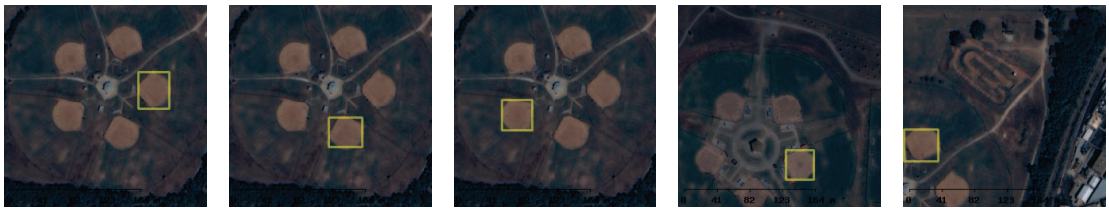
Algorithm 2 createCluster(p, N, eps, minpts)

Require: $eps \geq 0 \vee minpts > 0$
  $V = \emptyset$
  $i = getNextClusterNumber()$
  $C_i = \{p\}$
  for all p' in N **do**
    if $p \notin V$ **then**
      $V = V \cup \{p'\}$
      $N' = getNeighbors(P', eps)$
      if $|N'| \geq minpts$ **then**
        $N = N \cup N'$
      end if
    end if
    if $p' \notin C_i, \forall i$ **then**
      $C_i = C_i \cup \{p\}$
    end if
  end for

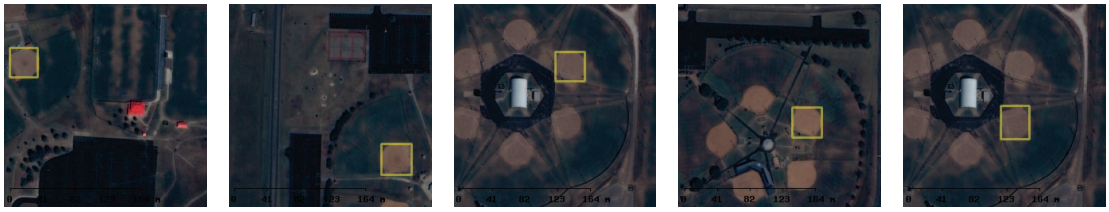
To illustrate the effect of the algorithm, Figure 2 shows the top 10 results from a CBIR query of satellite imagery containing a baseball diamond. By applying the above algorithm we are able to cluster our results into spatially proximate groups of results. An example of clustering results can be seen in Figure 3.

3 CONCLUSIONS

The ProgressiveDBSCAN algorithm allows for the progressive clustering of results from a geospatial information retrieval system. Results can be clustered by a combination of both their spatial and non-spatial attributes. The clustering performed is incremental in nature; as more results are generated and returned to the user the algorithm either incorporates them into existing clusters or assigned them to new clusters. The benefit of this clustering is that users are able to sort through the results returned from a geospatial information retrieval system in a spatial context. No longer are results from disparate locations presented to



(a) Result 1 (b) Result 2 (c) Result 3 (d) Result 4 (e) Result 5



(f) Result 6 (g) Result 7 (h) Result 8 (i) Result 9 (j) Result 10

Figure 2: The top-10 CIBR results from a search using the baseball diamond identified in Figure 2a as the query object. The query object is the top-ranked result found in the database and other baseball diamonds are returned based on their similarity to the query object.

the user, but instead compact spatial clusters are displayed. This improvement should reduce the analysis time spent interrogating the results.

References

[1] C. R. Shyu *et al.*, "GeoIRIS: Geospatial Information Retrieval and Indexing System-Content Mining, Semantics Modeling, and Complex Queries," *IEEE Trans. Geosci. Remote Sens.*, vol. 45, no. 4, pp. 839-852, April 2007.

[2] O. Sjahputera *et al.*, "GeoCDX: An Automated Change Detection & Exploitation System for High Resolution Satelite Imagery," *Proc. IEEE IGARSS* 2008, pp. 467-470.

[3] L. Gueguen and M. Datcu, "A Similarity Metric for Retrieval of Compressed Objects: Application for Mining Satellite Image Time Series," *IEEE Trans. Knowledge Data Eng.*, vol. 20, no. 4, pp.562-575, April 2008

[4] M. Ester *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," *Proc. Second Int. Conf. on Knowledge Disc. Data Mining*, pp. 226-231.

[5] C. R. Lin, K. H. Liu and M. S. Chen, "Dual clustering: integrating data clustering over optimization and constraint domains," *IEEE Trans. Knowledge Data Eng.*, vol. 17, no. 5, pp. 628-637, May 2005.

[6] J. Zhou *et al.* "Fast Implementation of Dual Clustering Algorithm for Spatial Data Mining," *Fourth Int. Conf. Fuzzy Sys. Knowledge Disc.*, vol. 3, pp. 568-572, Aug. 2007.

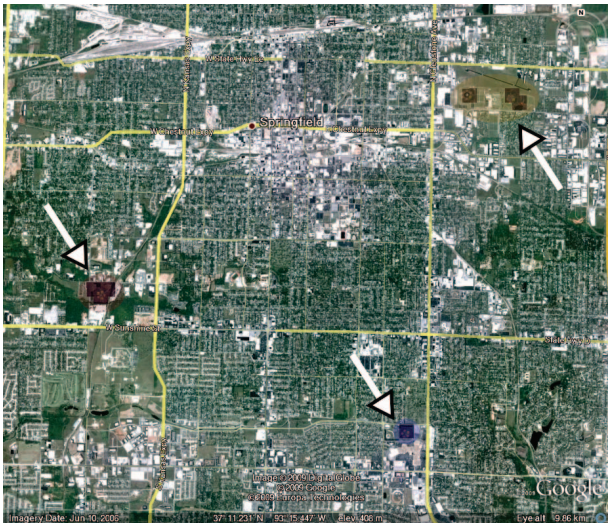


Figure 3: The results from Figure 2 clustered using Algorithm 1. The results are grouped into three clusters; in the picture each cluster is identified by a large arrow.