Finding Hidden Shrines using AR and Clustering Techniques

Liu Jihao Bryan*, Owen Noel Newton Fernando*, Sujatha Arundathi Meegama†,
Hedren Sum Wai Yuan#, Muhammad Faisal Bin Husni†
*School of Computer Science and Engineering, Nanyang Technological University, Singapore
†School of Art, Design and Media, Nanyang Technological University, Singapore
#Institute of Science and Technology for Humanity, Nanyang Technological University, Singapore

***Abstract*—This paper describes the use of augmented reality technology in an offline mobile application to recognize and find shrines. This application uses clustering algorithms to cluster shrines and then a convex hull is used to combine the geographical coordinates of multiple shrines. The photographs and videos of shrines can be downloaded by tapping on any clusters. Images downloaded will have their features extracted and stored in an augmented database which then will be used by ARCore, which is an augmented reality API, to identify and recognize all shrines in that cluster.**

Keywords–Augmented reality; clustering techniques; cultural computing; mobile application; mobile navigation

I. Introduction

The proliferation of augmented reality ready phones, tablets and any form of mobile devices have sparked a hype in the creation of many augmented reality application. Many developers are banking on augmented reality being the next big thing in the mobile market [1]. Even with the hype, there has been a lack of augmented reality application that aims to educate users on cultural sites particularly shrines and sacred places.

To address the lack of such application, we have implemented an augmented reality mobile application, to cater to users who are especially interested in shrines and would love to learn and find out more about these shrines. This application can work offline and allows users to discover and learn more about shrines.

Users will first have to download all image and videos about the shrines in the area and once it is downloaded, users can use the augmented reality feature of the app powered by ARCore to recognized shrines and get its relevant information. The shrine images and videos will be downloaded from Hidden Shrine, a shrine crowdsourcing website which crowdsources images and videos of shrines from users.

II. Related Works

Mobile application and augmented reality have been used before in similar fashion to educate users on cultural sites and exhibits.

One such example would be the application proposed by Vassilios Vlahakis [2] which utilizes augmented reality to conduct personalized tours of cultural sites and to do a 3D reconstruction of cultural sites that have been previously destroyed in the past in order to provide users with the full experience of what it looks like in the past in its full glory. This application aims to not only provides users with an understanding and knowledge of how ruined cultural sites look like in the past but also aim to make cultural sites more accessible to the public.

Another example would be the CHESS project proposed by Jens Keil [3] which aims to deliver personalized storytelling experience to visitors of cultural sites using augmented reality. The storytelling is tailored to each visitor predefined profile initially and adapts according to visitor's inputs, positions, and behaviors.

Another application, SkyLineDroid proposed by Jae-Beom Kim [4] is an augmented reality mobile application that provides an augmented reality tour of the National Palace Museum of Korea. This application allows visitors of the National Palace Museum of Korea to use the augmented reality feature on exhibits to capture the relevant information of the exhibits such as its name and history.

A research conducted by Newton Fernando [5] present the use of augmented reality for navigating and visualizing cemetery. The research proposed the use of natural feature tracking to recognize tombstone eliminating the need for engraving. In addition, clustering techniques are used to search feature dataset which helps to reduce computation time.

III. System Design and Implementation

A. System Architecture

The entire system architecture (Figure 1) follows the client-server implementation and it consists of a website client, mobile client and the PostgreSQL database server hosted in Heroku. The website allows users to upload shrine information and images. All this uploaded data will be stored in our database server. The database server is exposed to the mobile client via a restful service. The mobile client contains a map with the shrine geographical points group according to their clusters. Once users tap on any clusters, it will retrieve and download the latest shrine images and videos of all the shrines in the cluster from the server. Once the images and videos are downloaded, users can utilize the augmented reality feature by pointing the camera at a shrine and it will retrieve information about the shrine.

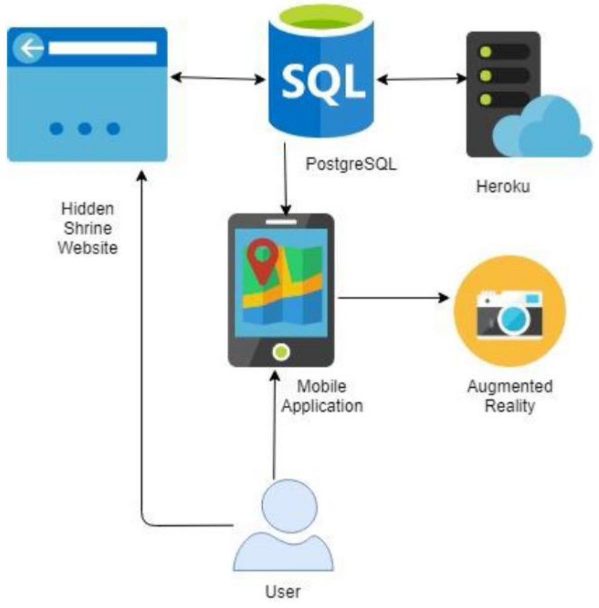

Figure 1. Overview of our System Architecture

B. *Clustering Algorithm*

To perform image recognition to recognize shrines, requires images to be downloaded on the phone. However, downloading all these images requires a lot of data. To reduce the amount of data required to download the image of shrines in an area, we will use a clustering algorithm to divide the shrines into multiple clusters, and then users can just download shrines by clusters instead of downloading all the shrines. This is useful because if users are near a cluster, they can just choose the nearest cluster to their location instead of having to download all the images.

To choose the appropriate clustering algorithm, it must specify the following criteria:

- Must be scalable to large n samples
- Non-flat geometry (longitude, latitude)
- Cannot detect outlier

Firstly, the clustering algorithm must be scalable to large n samples as there will be many shrines that will require clustering. Secondly, it also must accommodate non-flat geometry as we are dealing with longitude and latitude and lastly, it cannot treat points as outliers as all shrines have to be in a cluster regardless of its distances to each other. The table (Figure 2) shows the comparison between different clustering algorithm [6].

Method Name	Scalability	Use Case
K-Means	Very large n samples	Not many cluster, flat geometry
Affinity Propagation	Not scalable	Many cluster, non-flat geometry
Mean-Shift	Not scalable	Many cluster, non-flat geometry
Spectral Clustering	Medium n samples	Few cluster, non-flat geometry
Agglomerative Clustering	Large n samples	Many cluster, non-flat geometry
DBSCAN	Very large n samples	Non-flat geometry, outlier detection
Gaussian Mixtures	Not scalable	Flat geometry
Birch	Large n samples	Flat geometry, Remove outlier

Figure 2. The comparison of different clustering algorithm

From figure 2, in terms of scalability, we can infer that clustering algorithm such as Affinity Propagation, Mean-Shift, Spectral Clustering, Gaussian Mixtures are not suitable as they are either not scalable or only scalable to a small to medium samples.

In terms of non-flat geometry, we can infer that clustering algorithm such as K-Means, Birch are not suitable as they are only catered to flat geometry.

In terms of no outlier detection, DBSCAN is not suitable as it detects outlier which could cause shrines to be classified as outlier instead of part of a cluster. Hence, the only suitable clustering algorithm is the agglomerative clustering algorithm and it will be used for this application.

For this application, we will use the agglomerative clustering algorithm from the sci-kit learn clustering library. The metric used to define inter-cluster similarity for the merging strategy will be the ward method. The ward method for the inter-cluster similarity works by minimizing the sum of squared differences within all cluster [7].

The agglomerative clustering algorithm will then be applied to all the geographical coordinate points of all shrines. All the shrines location will be color coded according to the cluster they belong to. (Figure 3)

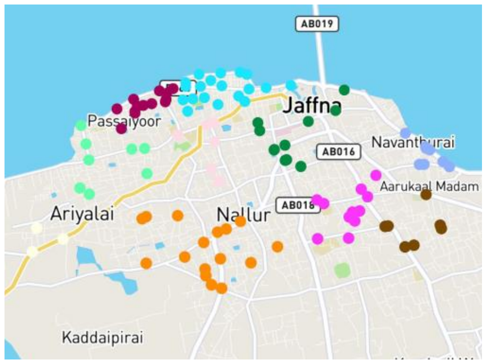

Figure 3. Each point represents one shrine and they are clustered, and color coded according to the cluster.

C. Convex Hull Algorithm

The clustering done in Figure 3 can be difficult for users to visualize the cluster, and in terms of usability, it is not easy for users to download the individual points.

To enable better visualization of cluster, a better approach can be taken with the use of the convex hull which will be used to join all points in a cluster to form polygon. A convex hull is basically a convex polygon in which all the points of a cluster are inside the polygon.

To create the convex hull, the Quickhull algorithm created in 1995 by C. B. Barber, D. P. Dobkin, and H. Huhdanpaa [8] will be used, and modify it such that the x and y coordinates are replaced with latitude and longitude.

The Quickhull algorithm is a recursive method where each step, partition points into several groups [9]. This step will then be repeated recursively. To use the Quickhull method, n should be more than 2 points (>= 3 points) as any number less than 3 points will be impossible to create a hull.

The pseudocode for the Quickhull algorithm is as follows:

```
Algorithm 1: Quickhull algorithm to draw convex hull
1  function QuickHull ($S$);
   Input  : A set S of n (latitude and longitude) points
   Output: Convex Hull
2  $ConvexHull := \emptyset$;
3  Find left and right most points, say A and B, and add A and B to convex
     hull;
4  Segment AB divides the remaining (n-2) points into 2 groups S1 and S2
     where S1 are points in S that are on the right side of the oriented line
     from A to B, and S2 are points in S that are on the right side of the
     oriented line from B to A;
5  FindHull (S1, A, B);
6  FindHull (S2, B, A);
7  function FindHull ($Sk, P, Q$);
8  if Sk has no point then
9  |   return;
10 end
11 From the given set of points in Sk, find farthest point, say C, from
     segment PQ;
12 Add point C to convex hull at the location between P and Q;
13 Three points P, Q, and C partition the remaining points of Sk into 3
     subsets: S0, S1, and S2 where S0 are points inside triangle PCQ, S1 are
     points on the right side of the oriented line from P to C, and S2 are
     points on the right side of the oriented line from C to Q.;
14 FindHull (S1, P, C);
15 FindHull (S2, C, Q);
```

Figure 4. Quickhull algorithm for drawing convex hull

The time complexity for the Quickhull algorithm is as follows [10]:

- Best Case Time Complexity = O(nlogn)
- Worst Case Time Complexity = O(n2)

After the convex hull algorithm is ran on all the points and the points are joined to form convex hulls (Figure 5), users can simply tap on the convex hulls(cluster) to download that specific cluster.

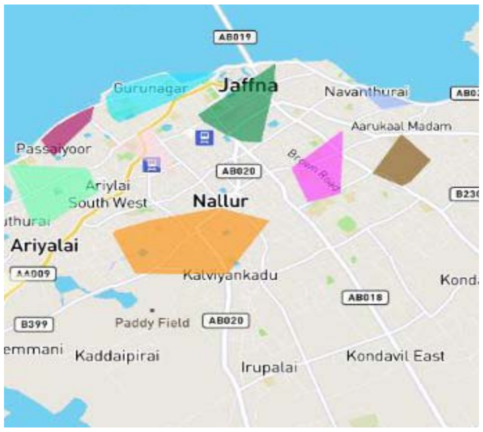

Figure 5. Shrines points joined to form convex hulls.

D. Augmented Reality using ARCore

To implement the augmented reality feature, all the shrine images will be downloaded and stored in the ARCore's augmented database. The feature points (visually distinct feature of an image) will then be extracted from the image and will be used to recognize any similar shrine image [11].

Once ARCore finds a similar image, it gives a pose [12], world coordinate space, for each object. These poses will be used to create anchors and position the shrine videos based on the anchor. ARCore will track the user's motion and changed the perspective of the video due to the motion.

An example can be seen in Figure 6, in which the shrine image will be first recognized and then in Figure 7, the shrine video is positioned on top of the image.

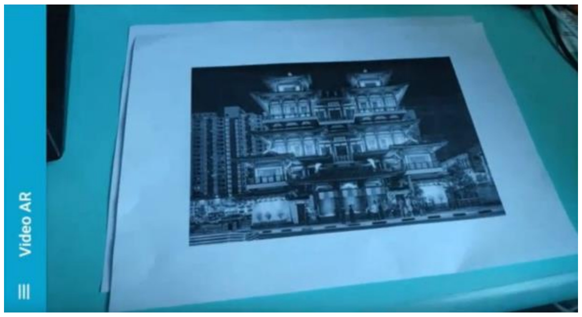

Figure 6. Shrine image recognized based on feature points

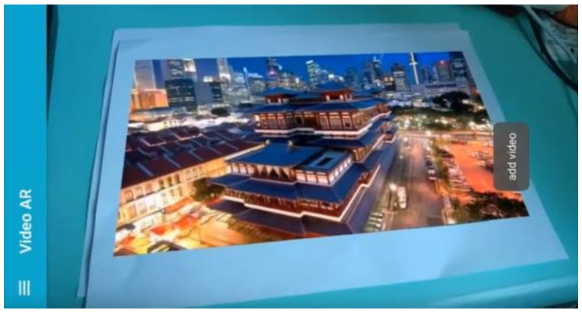

Figure 7. Videos is placed on recognized shrine image

IV. Conclusion

With this mobile application, it will improve the accessibility of shrines for users and they will also be more informed and educated about the culture and heritage of shrines.

In addition, with the implementation of Agglomerative clustering and the Quickhull algorithm to improve both usability and visualization, users will find this application intuitive and visually appealing.

Furthermore, with the use of augmented reality feature to discover shrine, users will have an exciting way to learn and discover more about shrines.

Acknowledgements

The research in this paper is supported by the Heritage Research Grant of the National Heritage Board of Singapore for the digital humanities research project, "The Hidden Shrines of Singapore: Mapping and Narrating Multi-Religious Heritages." Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Heritage Board, Singapore.

References

[1] H. Wen. (2010). *Augmented reality: Pure hype or Next Big Thing in mobile?* [Online]. Available: https://www.networkworld.com/article/2243332/augmented-reality--pure-hype-or-next-big-thing-in-mobile-.html

[2] V. Vlahakis *et al.*, "Archeoguide: first results of an augmented reality, mobile computing system in cultural heritage sites," presented at the Association for Computing Machinery, 2001.

[3] J. Keil, L. Pujol, M. Roussou, T. Engelke, M. Schmitt, and S. Eleftheratou, "A Digital Look at Physical Museum Exhibits : Designing Personalized Stories with Handheld Augmented Reality in Museums," presented at the Digital Heritage International Congress, 2013.

[4] J.-B. Kim and C. Park, "Development of Mobile AR Tour Application for the National Palace Museum of Korea," presented at the VMR, 2011.

[5] O. N. N. Fernando, C. Deshan, N. Pang, and R. Nakatsu, "Mobile Augmented Reality for Bukit Brown Cemetery Navigation," in *11th ACM SIGGRAPH International Conference*, 2012.

[6] Scikit-Learn. *2.3.1. Overview of clustering methods* [Online]. Available: https://scikit-learn.org/stable/modules/clustering.html

[7] Scikit-Learn. *2.3.6. Hierarchical clustering* [Online]. Available: http://scikit-learn.org/stable/modules/clustering.html#hierarchical-clustering

[8] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The Quickhull Algorithm for Convex Hulls," *ACM Transactions on Mathematical Software,* 1995.

[9] A. Hausner. (1997). *Quick-Hull* [Online]. Available: http://www.cs.princeton.edu/courses/archive/spr09/cos226/demo/ah/QuickHull.html

[10] J. S. Greenfield, "A Proof for a QuickHull Algorithm," Syracuse University1990, Available: https://surface.syr.edu/cgi/viewcontent.cgi?referer=https://www.google.com.sg/&httpsredir=1&article=1058&context=eecs_techreports.

[11] Google. (2019). *Recognize and Augment Images* [Online]. Available: https://developers.google.com/ar/develop/java/augmented-images/

[12] Google. (2018). *Pose* [Online]. Available: https://developers.google.com/ar/reference/java/arcore/reference/com/google/ar/core/Pose