The 12th International Symposium on Virtual Reality, Archaeology and Cultural Heritage VAST (2011)
M. Dellepiane, F. Niccolucci, S. Pena Serna, H. Rushmeier, and L. Van Gool (Editors)

Point Cloud Segmentation for Cultural Heritage Sites

S. Spina[1] and K. Debattista[1] and K. Bugeja[1] and A. Chalmers[1]

[1]International Digital Lab - University of Warwick

Abstract

Over the past few years, the acquisition of 3D point information representing the structure of real-world objects has become common practice in many areas. This is particularly true in the Cultural Heritage (CH) domain, where point clouds reproducing important and usually unique artifacts and sites of various sizes and geometric complexities are acquired. Specialized software is then usually used to process and organise this data. This paper addresses the problem of automatically organising this raw data by segmenting point clouds into meaningful subsets. This organisation over raw data entails a reduction in complexity and facilitates the post-processing effort required to work with the individual objects in the scene. This paper describes an efficient two-stage segmentation algorithm which is able to automatically partition raw point clouds. Following an intial partitioning of the point cloud, a RanSaC-based plane fitting algorithm is used in order to add a further layer of abstraction. A number of potential uses of the newly processed point cloud are presented; one of which is object extraction using point cloud queries. Our method is demonstrated on three point clouds ranging from 600K to 1.9M points. One of these point clouds was acquired from the pre-historic temple of Mnajdra consistsing of multiple adjacent complex structures.

Categories and Subject Descriptors (according to ACM CCS): I.3.8 [Computer Graphics]: Applications—

1. Introduction

In recent years many CH institutions have been engaged in the exercise of creating 3D virtual reproductions of artefacts and sites for which they are responsible. Large architectural heritage sites are continuously being scanned and documented (for example in the work described by Ruther [Rut10]). There are clearly many reasons why this is important including amongst others academic study, hypothesis evaluation, better preservation and CH dissemination to the general public. Traditional 3D scanning devices based on laser or structured light, using either triangulation or time-of-flight approaches are normally used. Recently with the popularisation of algorithms and tools (for example Vergauwen et al. [VG06] and Snavely et al. [SSS06]) capable of generating relatively accurate 3D point clouds from photographs without the need of expensive and specialised hardware, the amount and availablility of 3D point cloud data has also rapidly increased. This scenario is contributing to an increase in importance for algorithms which are capable of analysing and processing point clouds efficiently. As pointed out by Cignoni et al. [CS08], a major challenge is now how to manage the complexity of scanned data. In this paper we address this concern by proposing an efficient and semantically meaningful point cloud segmentation pipeline that assumes only the availability of position information within the data. This partitioning of the point cloud effectively provides for a higher level of abstraction over raw position data and allows for easier and more efficient point cloud manipulation.

Our point cloud processing pipeline consists of two main stages; a segmentation stage and a fitting stage (using the RanSaC paradigm [FB81]). For the first stage, a point cloud segmentation algorithm which is particularly suited for the processing of sites which typically exhibit rough (e.g. eroded stone) surfaces is described. The segmentation algorithm is robust enough to automatically extract most of the individual stones composing the rubble walls of the temple. Following this process RanSaC fitting is applied on the segments produced in order to try and fit primitive shapes within the data.

The Mnajdra site area used for this paper's results is approximately 60m^2. Figure 2 illustrates the rendering of the point cloud acquired from the smallest of three temples in the Mnajdra pre-historic site. When laser scanning relatively large sites, the final point cloud is the union of various scans of many different adjacent objects. As pointed out by Cignoni et al. [CS08], the acquisition process is followed

DOI: 10.2312/VAST/VAST11/041-048

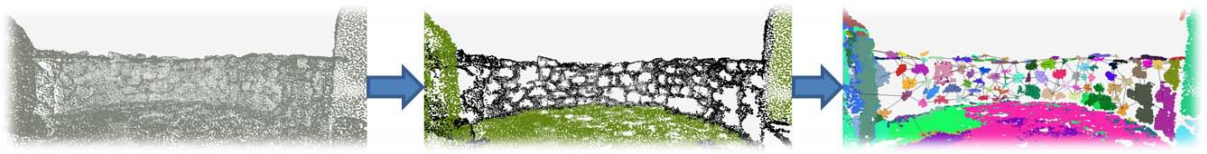


Figure 1: *Automatic Point Cloud Segmentation Pipeline - Raw data is first segmented into smaller patches then geometric planes (coloured patches) are mapped onto these segments using the RanSaC paradigm*

by substantial data processing, usually requiring user intervention, long processing times and above all tedious work. Ruther in [Rut10] describes how post-processing tasks usually take much more time than the actual acquisition process on site. This can be more relevant in the CH domain whenever points in the cloud would usually need to be grouped and tagged as specific objects for future reference. This processing time can be decreased if the point cloud generated from the scanning process is partitioned into smaller meaningful subsets of points representing distinct geometries (see Figure 1). This ability to automatically distinguish between different elements in the scene would benefit the CH professional working with the acquired point cloud. For example, tessellation problems common with complex sites such as pre-historic temples consisting mostly of weathered and eroded stone which usually require decimation, can be approached compositionally by tessellating segments individually according to their needs. The selection of parts of a site, for example a specific wall or the floor, would usually require users to learn how a specific GUI is used. Our segmentation pipeline enables the use of simple point cloud queries, where a processed point cloud can be used to efficiently query for and extract specific parts.

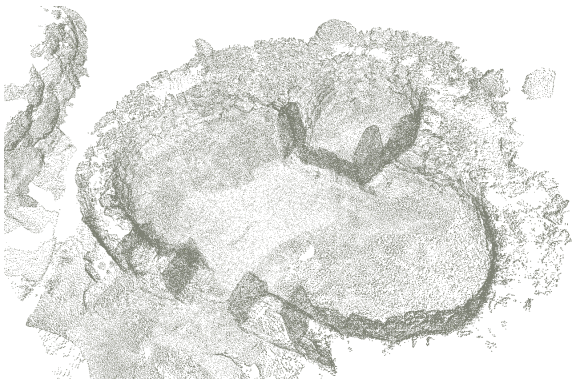


Figure 2: *Point cloud of temple (600K points)*

2. Related Work

The post-processing of scanned 3D surfaces plays an important role in the acquisition pipeline of CH artefacts. Overviews of the typical problems encountered and descriptions of the various algorithms designed to address them can be found in various sources, such as the work carried out by Bernardini et al. [BR02], Weyrich et al. [WPH*04] and Cignoni et al. [CS08]. Segmentation addresses the problem of automatically partitioning a set of points (in 3D) or pixels (in 2D) into equivalence classes representing 'meaningful' patches.

A number of point cloud segmentation (or decomposition) algorithms exist, most of them based on depth or range maps with 2.5D information. In this case traditional image processing techniques can be applied, mainly adopting either edge or surface based segmentation. In edge-based techniques, for example by Wani et al. [WB94], edges on the image are first detected then pixels within these regions grouped together. In surface-based techniques, grouping is based on similarities (e.g. normals, curvature) between spatially close pixels. For surface-based segmentation either top-down (starting from one segment and splitting accordingly) or bottom-up (starting from seed pixels and growing the segments) approaches are usually adopted. Our segmentation algorithm applies an edge-based bottom-up approach directly on unstructured 3D points.

Other segmentation algorithms working directly on point clouds have been developed which are more domain-specific (e.g. segmentation of point clouds representing trees by Ning et al. [NZW09] and buildings by Dorninger et al. [DN07]) and make a number of assumptions on the input point cloud. Golovinsky et al. [GF09] describe a min-cut based method capable of extracting objects from point clouds. Given an initial object location their algorithm computes a foreground-background segmentation. Robbani et al. [RV06] describe an algorithm based on smoothness constraints which is able to produce smoothly connected areas. Their work is however primarily focused on identifying industrial objects exhibiting smooth surfaces. Segmentation is particularly challenging in the CH scenario due to the generally more complex geometrical and surface properties (eg. weathered and eroded stone) typical for CH sites.

Segmentation can also be carried out by trying to fit primitive objects (e.g. planes, spheres, cylinders, etc) within the point cloud using the RanSaC paradigm developed by Fischler et al. [FB81]. RanSaC provides an efficient mechanism by which geometric primitves (planes, spheres, cylin-

ders, etc.) are fitted to point cloud data. Minimal point sets representing these primitives are iteratively randomly selected from the point cloud and used to determine whether the rest of the points actually fit the current model primitive. If enough points fit the model, this is chosen to represent the data. The main benefit resulting from this technique is that outliers have no influence on the result. Schnabel et al. [SWK07] demonstrate that RanSaC can be very efficiently used on point cloud data consisting of millions of points.

As opposed to point cloud segmentation algorithms, a considerable number of mesh segmentation algorithms exist. An overview of mesh segmentation algorithms is given by Chen et al. [CGF09]. Of particular interest is the work carried out by Wu et al. [WK05], in that they adopt a similar partitioning and primitive fitting sequence over the input mesh. The provision of topological information on the vertices is assumed to be correct and usually used when determining the various segments. Since tessellating a point cloud might actually introduce surface errors in the data (for example joining vertices from adjacent stones together), we opt not to tessellate point clouds before segmentation. Tessellation is actually considered as one of the applications of our segmentation pipeline.

3. Segmentation

Key to the segmentation method presented here is the observation that objects typically consist of a combination of surfaces (not necessarily flat) and edges (not necessarily straight). A simple object such as the obelisk shown in Figure 3, for example, consists of five surface geometries (rendered as green splats) connected together through one continuous edge (rendered as black splats). When sampling the surface of this obelisk into a point cloud, points would fall into two disjoint subsets storing respectively points that are found on surfaces, and others that are found on edges. On the other hand when sampling a smooth spherical object (if enough points are sampled), the set of edges would be empty since every point sampled would be a surface point. Each point is either of type surface or edge and the determination of type information for each point depends on two factors, namely, point cloud resolution and neighbourhood function implementation.

All vertices in the point cloud are stored in a spatial data structure in order to accelerate neighbourhood queries. The neighbourhood query function, $\phi(p)$, returns the closest n points to point p. It takes into consideration not just the grid cell in which the current point p is located but also points in the 26 adjacent cells. In this way, the algorithm is able to crawl across neighbouring points when segmenting the point cloud. The time complexity is $O(nm^2)$ where n is the number of points in the cloud and m is the number of points in the 26 adjacent cells.

Prior to segmentation, Principal Component Analysis (PCA) is used to determine for each point whether it is most likely to be located on a surface or an edge. In a similar fashion to the work of Hoppe et al. [HDD*92], an oriented bounding volume for a small set of neighbouring points is first calculated. The ratio of the eigenvalues of this orthogonal basis determines the type of each point. Eigenvalues are discussed in depth by Pauly et al. [PGK02]. If the third eigenvalue is much smaller than the second eigenvalue, i.e. there is minimal variance along the third eigenvector then the point is tagged as *surface*. The point is tagged as *edge* otherwise. For the point clouds used in this paper this ratio was set to 12, i.e. if the third eigenvalue multiplied by 12 is smaller than the second eigenvalue, then the point is tagged as *surface*. The neighbourhood function $\phi(p)$, returning the 30 closest points is used to determine point type. Figures 3(a) and 3(d) show the same obelisk point cloud with the latter figure having all its points labelled as either edge (black splats) or surface (green splats).

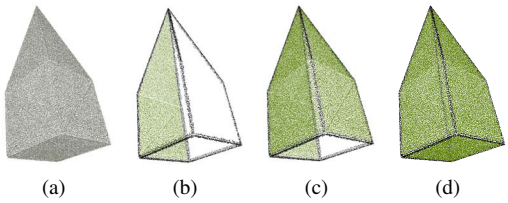


Figure 3: *Point cloud segmentation of obelisk - starting from position only information all points are progressively assigned to segments using algorithm 1.*

The output of our segmentation algorithm is a graph G, representing the various surface and edge segments extracted from the point cloud. Each segment is a collection of points with the same type. Let $G = (N, E)$ be an undirected graph with graph vertices $n_i \in N$, be the set of segments (both surface and edge) obtained after segmentation and edges $(n_i, n_j) \in E$ corresponding to pairs of adjacent segments. The union of points from all segments $n_i \in N$ is equal to the set of points P in the point cloud. The pairwise predicate *adjacent* is defined on segments as $A(n_i, n_j)$ = true if there exists a point $p_i \in n_i$ and a point $p_j \in n_j$ such that p_i and p_j fall in the same neighbourhood set and the type of p_i is different from that of p_j. If this predicate is true, an edge is created between n_i and n_j storing the spatial relationship between them. Graph edges are thus only created between segments of different type.

Pseudo code of our segmentation process can be seen in Algorithm 1. In order to improve the readability of the pseudo code the input is taken to be a point cloud P with point type (edge or surface) information already calculated. Using this type information, an algorithm based on flood-fill is used to partition the point cloud according to the predicate $A(n_i, n_j)$. In the pseudo code, the type *Segment* refers to both

surface or edge segments, and the boolean array V is used to store the visited status of each point. A point becomes visited as soon as it is associated with a segment.

Algorithm 1 Segmentation of Point Cloud P into G=(N,E)

Input: Point cloud P with point type information, Segments $\{N\}$, Segment Relations $\{E\}$, Neighbour function ϕ(p), Boolean array V of size $|P|$, Queue Q_A, Queue Q_R
Initialise: Set all elements in V to false, Enqueue Q_R with the pair (p $\in P$, new s:Segment)
while Q_R is not empty **do**
  $(p,s) \Leftarrow^{deq} Q_R$
  $p \Rightarrow^{enq} Q_A$
  while Q_A is not empty **do**
    $p \Leftarrow^{deq} Q_A$
    $V[p]$ = true
    $\{s\} \Leftarrow^{add} p$
    $\{nbr\} \Leftarrow \phi$(p)
    for $i = 1$ to $|nbr|$ **do**
      if $V[nbr_i]$ == false **then**
        if $nbr_i.T$ == activeType **then**
          $nbr_i \Rightarrow^{enq} Q_A$
        else
          $(nbr_i, new\ s' : Segment) \Rightarrow^{enq} Q_R$
          $\{E\} \Leftarrow (s,s')$
        end if
      end if
    end for
  end while
  $\{N\} \Leftarrow$ gg
end while
return $G = (N,E)$

Two queues are maintained throughout the segmentation process. The first one, Q_A, stores the currently active points, i.e. those points retrieved by the neighbourhood query to be considered next. The second queue, Q_R, stores *potential* new segments. Q_R is initialised with the pair (p_i, s:Segment), where p_i, is randomly chosen from the sparse grid cell which has the highest number of surface points. s is thus a new surface segment seeded with the point p_i. Q_A is initiailised by popping this first element from Q_R and pushing p_i onto the queue. The *activeType* property is set to the type of p_i, i.e. surface in this case.

The status of these two queues determines the control flow of the algorithm. The outer while loop checks whether there are any more potential segments that can be created whereas the inner loop checks whether there are any more points that can be added to the current segment. During the inner loop the current point p is first added to the current segment, then the points returned from the neighbourhood query $\phi(p)$ are pushed onto Q_A if they are of the current active type. If not, the point together with a new instance of a segment (as a pair) are pushed onto Q_R and the relation (edge on the graph G) between the current and this new segment is added to $\{E\}$. This essentially implements the pairwise predicate *adjacent*. In our implementation another outer loop is added to check whether there are any points not considered for segmentation. This can occur when there are distant disjoint objects in the scene. In this case the execution of the algortihm is repeated on the disjoint subset of points.

3.1. Point Cloud Resolution

An important factor determining the outcome of the segmentation algorithm is the resolution of the points in the cloud as this directly affects the point set returned by the neighbourhood function. This function is first used to determine the type of each point then used when constructing the segmentation graph. Due to the surface roughness present at the Mnajdra temple, resolution contributes mostly when determining point type. Figure 4 shows a cross-section of a hypothetical surface with samples (crosses) taken from it. When a higher resolution is used many more points will be tagged as edge points as opposed to the slightly lower resolution sampled points.

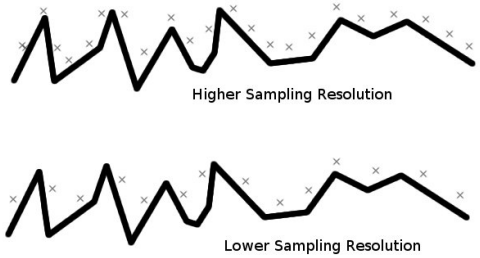


Figure 4: *High and slightly lower point sample resolution*

The result of the neighbourhood function also depends of the resolution of the sparse grid used to fit the points. The smaller the resolution the more points would surround a given point (in the 26 adjacent cells). Grid and point cloud resolution can be viewed as the two sides of the same coin. A user-defined parameter is hence used to determine the resolution of the sparse grid in order to best fit the data available.

3.2. Over and Under Segmentation

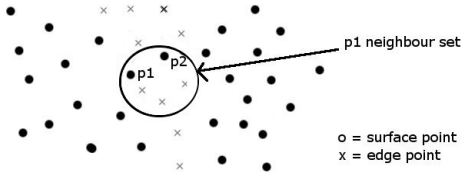


Figure 5: *Boundary between edge and surface points*

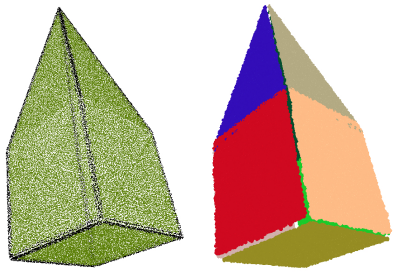
Given the nature of most surfaces in CH, especially those

where very old and weathered stone is present, over segmentation can easily occur. The other side of the coin is under segmentation, where distinct geometric entities are merged together as one segment. In order to minimise under-segmentation, a straightforward heuristic is used within the inner loop of the algorithm which determines whether the points returned by the neighbourhood function, which have the same type as the current active point, should be added to the current active queue. If the number of current active type points is less than the number of points which are not of the same type, the points returned by the neighbourhood function are added to the current segment but not added to the current active queue. Figure 5 shows a simple boundary example where if $p2$ had to be added to the current active queue, the current segment would end up with many more surface points which should really be a separate segment. This heuristic is used to minimise the possibility of under segmentation by somehow measuring the evidence that a boundary between two regions exists.

4. RanSaC Fitting

In our processing pipeline only plane primitives (parametrised with a tolerance value to include points close to the plane) are used and fitted to the patches resulting from segmentation. The main intuition behind the use of planes is that these provide for an efficient representation of more diverse geometric objects as collections of planes. For example, the three apses of the Mnajdra temple might each fit a cylinder primitive, however, finer grain segmentation can be achieved when using a number of smaller planes each represting the indivdual stones composing the surface. The second stage of the pipeline, takes each surface segment produced by the segmentation algorithm just described, and determines which plane geometric primitives fit the data. Clearly since the data (within a segment) will nearly never perfectly fit a plane, a tolerance parameter t, is attached to each plane, i.e. points are allowed to fit the plane whenever the distance d from a point to a plane is within the range $-t \leq d \leq t$. For each segment three points are randomly chosen to define the plane parameters. These parameters are then used to calculate the percentage of points from the segment which fit this model. Different plane parameters are repeatedly chosen until this percentage gets stable, i.e. does not improve over a number of iterations. If the percentage of points fitting the plane is below a certain threshold, further plane fitting is carried out on the remaining points in the segment. Figure 6 shows how the previously segmented obelisk (five surface and one edge segments) has been fitted with nine planes, i.e. the four vertical segments have been fitted to two planes each, whereas the bottom horizontal segment easily fits within one plane.

In general, when the number of iterations computed is limited, the solution (planes created in our case) obtained may not be optimal and it may not even be one that fits the data in a good way. In our case, since RanSaC fitting is done over segments and not the entire point cloud, this whole process is much more efficient timewise and the solution obtained for Mnajdra fits the data perfectly. Moreover, this new level of abstraction over the data, allows for the creation of simple point cloud queries which can be used to very efficiently select different parts of the point cloud.



(a) All five surface segments (b) Nine RANSAC planes

Figure 6: *Plane primitives fitted to obelisk segments*

5. Point Cloud Queries

The extracted planar patches, together with a neighbourhood graphs computed over these patches as shown in Figure 10(b), provide the required structure necessary to enable reasoning about and querying of point clouds. Given a seed patch (RanSaC plane), a predicate comparing the surface normals of adjacent planar patches (e.g. similar, same or opposite directions) is used to traverse the graph until the predicate is satisfied. Figure 7 shows the point cloud representing the Kalabsha temple [SCM04] and the transformation of some of the segments extracted (representing columns here) into RanSac plane primitives. The bottom left part of the image shows a cross section of one column and describes how RanSaC plane primitives are fitted. When searching the point cloud, one of these planes needs to be manually selected as the seed plane. The result of the query (or search) would consist of all the planes connected to the seed which satisfy the predicate.

6. Results

Our segmentation pipeline is evaluated on three point clouds. The first one represents part of the pre-historic site of Mnajdra that was acquired in 2005 by Heritage Malta, the national agency for museums, conservation practice and cultural heritage in Malta. The modeled surface precision was stated as +/- 2mm. The bounding volume of the point-cloud is [19.64,18.58,3.54]. The sparse grid resolution was set to 0.05, thus producing a sparse grid of maximum size 392x371x71 cells. The second point cloud, used to better assess the scalability of our segmentation pipeline, consists

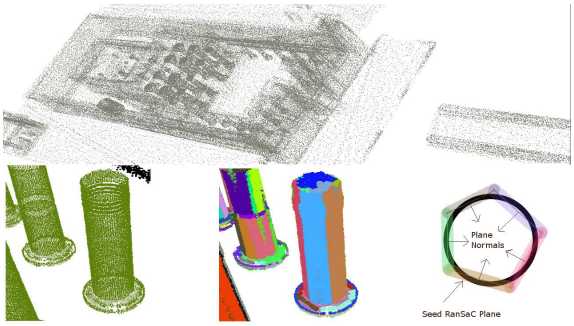
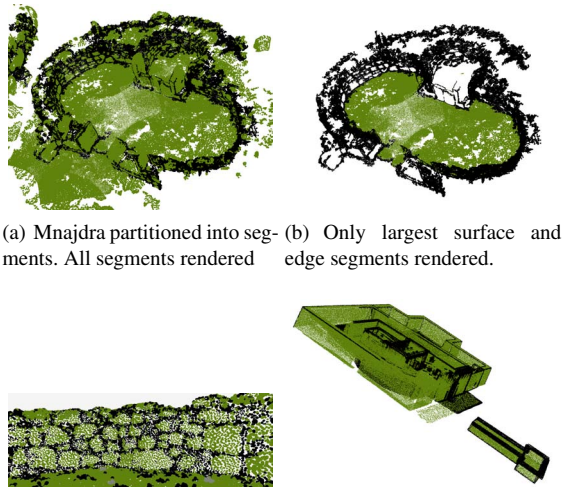


Figure 7: *Individual columns extracted via the segmentation process are fitted with planes.*



(a) Mnajdra partitioned into segments. All segments rendered

(b) Only largest surface and edge segments rendered.

(c) Closeup view of one of the apses

(d) Kalabsha partitioned into segments

Figure 8: *Segmentation of point cloud*

of the same Mnajdra point cloud upsampled to 1.2M points. Bounding volume and resolution are not altered. The segmentation pipeline is also applied on a point cloud reprseting the Kalabsha temple consisting of 1.9M points. The bounding volume of this point cloud is [43.64, 6.90, 31.70] and the sparse grid resolution was set to 0.1. Segmentation results are presented first, followed by results obtained from RanSaC fitting. Finally, two point cloud queries are described which enable the quick selection and retrieval of parts within the Mnajdra point cloud.

6.1. Segmentation

In the Mnajdra case study presented here, segmentation should be able to discriminate between the various stones composing the temple. Figure 8(a) shows the entire point cloud rendered using our point cloud viewer with black and green splats representing edge (248,761) and surface (344,248) points respectively. A total of 793 edge and 1031 surface segments are extracted.

Figure 8(b) shows the largest surface and edge segments. As would be expected the largest surface segment consists of points sampled from the floor of the main part of the temple, whereas the largest edge segment is made up of points connecting the entire rubble wall structure. Figure 8(c) illustrates a closer view (of part) of the rubble wall present in the site. Around 35 stones are automatically identified in this part alone. The edge segment contributes towards the partitioning of the rubble wall into a number of surface segments representing the individual stones making the apse. Very similar results are obtained with the upsampled 1.2M point cloud. Due to the increased number of points per grid cell, more time is spent in neighbourhood queries. With the Kalabsha 1.9M point cloud, the various walls, floors and columns are extracted as shown in figure 8(d). Table 1 shows the processing time for the first and second stages of the pipeline to compute.

PointCloud	Segmentation	Fitting	#Planes
Mnajdra 600K	55sec	60sec	930
Mnajdra 1.2M	112sec	90sec	1011
Kalabsha 1.9M	180sec	121sec	2025

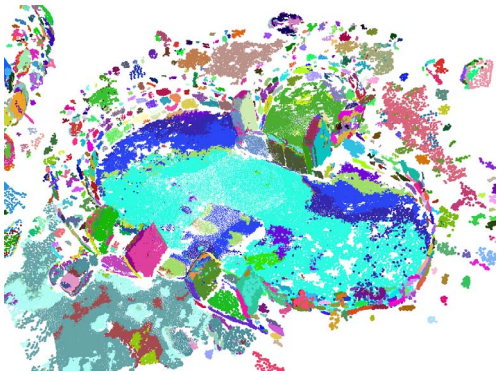
Table 1: *Statistics for the three point clouds used*

6.2. RanSaC Plane Fitting

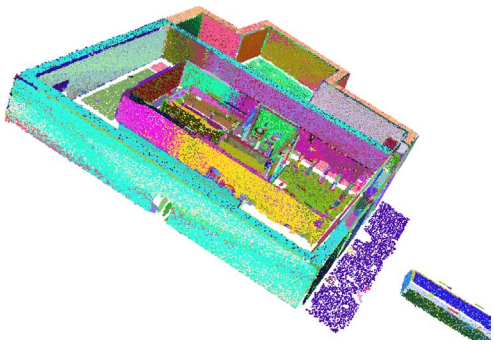
Figure 9(a) shows all the planes (rendered using different colours) fitted over all the different surface segments of the Mnajdra temple. A total of 930 planes are created fitting all surface points (344,248) in 60 seconds. Prior point cloud segmentation is critical here. Our RanSaC implementation is very efficient due to the fact that planes are fitted on subsets of related points (produced by our segmentation process) rather than the whole data set. The 98% of the surface points are fitted within the 930 plane primitives. Without prior segmentation, our implementation of RanSac does not converge properly and only two planes (see Figure 9(c)) are fitted to the floor of the temple in approximately 4 minutes. Moreover, each of the two plane primitives cover points which are found on different, unrelated parts of the temple. When segmentation is used, one can visually easily verify that the planes created fit the data in a very accurate way and that the general structure of the temple is clearly visible. Figures 10(a) and 10(b) show the transformation from surface segments to planar patches on one of the apses. A total of 1011 planes are fitted on all surface points of the upsampled Mnajdra point cloud in approximately 90 seconds. The planes created in this upsampled version of the Mnajdra temple are nearly identical to the ones created in the 600K model. This is expected since the general structure of the

temple is maintained. Given that the same grid resolution is kept, more points are present within each grid cell thus resulting in more time spent on neighbourhood queries. For the Kalabsha 1.9M point cloud, 2025 planes are extracted in 121 seconds (see figure 9(b)). Figure 9(d) shows how planes are fitted over the columns present in the temple.

Over these planes a neighbourhood graph is derived which is used in the implementation of point cloud queries. Each node in the neighbourhood graph represents a plane primitive. The median point of each plane primitive is used to calculate the distances between each plane and generate the graph. Note that if segmentation is not done before RanSaC fitting then the neighbourhood graph derived would not be correct. This can be observed in Figure 9(c), where the medians of the two planes would be very close.

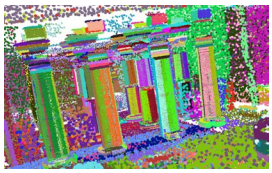


(a) Plane fitting over the segmented Mnajdra point cloud



(b) Plane fitting over the segmented Kalabsha point cloud

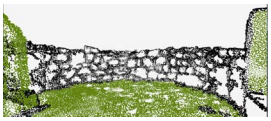
(c) RanSac plane fitting **without** prior segmentation



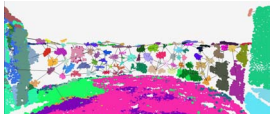
(d) View of columns within Kalabsha point cloud

Figure 9: *Fitting on Mnajdra and Kalabsha point clouds*

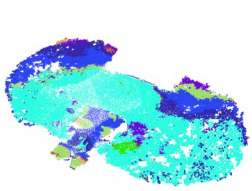
6.3. Point Cloud Queries



(a) Segmention of one apse



(b) Planes fitted over surface segments



(c) Query returns all points on the main floor

(d) Query returns internal walls of the temple

Figure 10: *Point Cloud Query results*

Two queries have been implemented to describe how related parts of a point cloud can be extracted. The first query takes a seed patch and iteratively follows the neighbourhood graph until the predicate used returns false. The predicate always compares the current node in the graph with the seed node. When p is set to one of the patches on the main floor of the temple and the predicate is set to *similar(seedNormalDir, pNormalDir, tolerance)*, the query returns all the points making up the floor of the main part of the temple as shown in Figure 10(c). The query starts searching from the seed patch and follows the neighbourhood graph until the predicate *similar* returns false, which occurs when the vertical wall patches are encountered. Note here, that although moving away from the seed patch to adjacent patches, the query always compares against the normal of the seed patch.

The second query applies the predicate on adjacent planes. Hence, if the predicate is comparing the difference in normals it would do so between adjacent patches rather than with the seed patch. This query can be used to extract the internal walls of the temple. Figure 10(d) shows the union of three such queries where the seed patches are chosen from stones located in the three different apses and the predicate used was *similar(pNormalDir, qNormalDir, tolerance)*. Note how the query now follows the rubble wall until megalith stones positioned perpendicular to the wall are encountered.

7. Further Applications

In this section further applications of our point cloud segmentation pipeline are proposed.

Texture Mapping. Aligning textures with geometry is an important post processing task especially if no colour information is available within the data. In this case, the ability of

partitioning the point cloud into small meaningful segments can be used to find correspondences between photographs of the site and the specific parts of the point cloud.

Adding Semantics. CH experts would usually require that specific parts of the point cloud are tagged with some specific meaning. For example one might label a particular segment of the point cloud as representing the ground, which is then linked to photographs from the site. In this regard, point cloud queries require further development so that they can be used to locate and extract specific objects within point clouds. Further work is being carried out on formalising a point cloud query language and develop a user friendly GUI which can be used to execute these queries.

Tessellation. Another interesting aspect of segmentation lies in the potential use of the results obtained to optimise tessellation algorithms and rendering quality. For instance, automatically creating a reasonably accurate mesh of the Mnajdra temple is not a simple task. Segmentation results can be used to project the RanSac planes extracted onto a flat surface, tessellate using traditional Delaunay triangulation, then use the topological information acquired to render the quasi-flat surface as a triangular mesh.

8. Conclusion and Future Work

The generation of 3D point clouds is becoming increasingly common in many areas of research. Given this huge amount of data, algorithms are required which are able to process, organise and extract important information about it in order to help in the post-processing effort. Our initial results have shown that the proposed automatic segmentation pipeline is a feasible approach towards achieving this goal. However, in order to better evaluate the robustness and scalability of this approach, we are currently in the process of obtaining further point clouds whose size ranges from a few million to billions of points. As the size of the point cloud increases, we plan to extend the current implementation of the segmentation pipeline to support out-of-core processing. Moreover, in order to keep the processing times within a reasonable limit, a distributed and concurrent implementation of the pipeline described here is being designed. Another important direction is the further development of point cloud queries, in order to provide an automated, user-friendly and efficient mechanism by which CH professionals can select interesting parts within a point cloud.

References

[BR02] BERNARDINI F., RUSHMEIER H. E.: The 3d model acquisition pipeline. *Comput. Graph. Forum 21*, 2 (2002), 149–172. 2

[CGF09] CHEN X., GOLOVINSKIY A., FUNKHOUSER T.: A benchmark for 3d mesh segmentation. In *ACM SIGGRAPH 2009 papers* (New York, NY, USA, 2009), SIGGRAPH '09, ACM, pp. 73:1–73:12. 3

[CS08] CIGNONI P., SCOPIGNO R.: Sampled 3d models for ch applications: A viable and enabling new medium or just a technological exercise? *J. Comput. Cult. Herit. 1* (June 2008), 2:1–2:23. 1, 2

[DN07] DORNINGER P., NOTTHEGGER C.: 3d segmentation of unstructured point clouds for building modelling. In *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* (2007), Institute of Photogrammetry and Cartography Technische Universitaet Muenchen. 2

[FB81] FISCHLER M. A., BOLLES R. C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM 24* (June 1981), 381–395. 1, 2

[GF09] GOLOVINSKIY A., FUNKHOUSER T.: Min-cut based segmentation of point clouds. In *IEEE Workshop on Search in 3D and Video (S3DV) at ICCV* (2009). 2

[HDD*92] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Surface reconstruction from unorganized points. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1992), ACM, pp. 71–78. 3

[NZW09] NING X., ZHANG X., WANG Y.: Tree segmentation from scanned scene data. In *Plant Growth Modeling, Simulation, Visualization and Applications (PMA), 2009 Third International Symposium on* (2009), pp. 360 –367. 2

[PGK02] PAULY M., GROSS M., KOBBELT L. P.: Efficient simplification of point-sampled surfaces. In *Proc. VIS '02* (USA, 2002), IEEE, pp. 163–170. 3

[Rut10] RUTHER H.: Documenting africa's cultural heritage. In *In Proceedings of the 11th International Symposium VAST. Virtual Reality, Archaeology and Cultural Heritage* (2010). 1, 2

[RV06] ROBBANI, VOSSELMAN: Segmentation of point clouds using smoothness constraint. In *Image Engineering and Vision Metrology* (2006). 2

[SCM04] SUNDSTEDT V., CHALMERS A., MARTINEZ P.: High fidelity reconstruction of the ancient egyptian temple of kalabsha. In *Proceedings of the 3rd international conference on Computer graphics, virtual reality, visualisation and interaction in Africa* (New York, NY, USA, 2004), AFRIGRAPH '04, ACM, pp. 107–113. 5

[SSS06] SNAVELY N., SEITZ S. M., SZELISKI R.: Photo tourism: Exploring photo collections in 3d. In *SIGGRAPH Conference Proceedings* (New York, NY, USA, 2006), ACM Press, pp. 835–846. 1

[SWK07] SCHNABEL R., WAHL R., KLEIN R.: Efficient ransac for point-cloud shape detection. *Computer Graphics Forum 26*, 2 (2007), 214–226. 3

[VG06] VERGAUWEN M., GOOL L. V.: Web-based 3d reconstruction service. *Mach. Vision Appl. 17*, 6 (2006), 411–426. 1

[WB94] WANI M. A., BATCHELOR B. G.: Edge-region-based segmentation of range images. *IEEE Trans. Pattern Anal. Mach. Intell. 16* (March 1994), 314–319. 2

[WK05] WU J., KOBBELT L.: Structure recovery via hybrid variational surface approximation. *Comput. Graph. Forum 24*, 3 (2005), 277–284. 3

[WPH*04] WEYRICH T., PAULY M., HEINZLE S., SCANDELLA S., GROSS M.: Post-Processing Of Scanned 3D Surface Data. In *Symposium On Point-Based Graphics* (2004), pp. 85–94. 2