EUROGRAPHICS Workshop on Graphics and Cultural Heritage (2016)
C. E. Catalano and L. De Luca (Editors)

Accelerating point cloud cleaning

R. L. Mulder[1] and P. Marais[1]

[1]University of Cape Town, Computer Science Department, South Africa

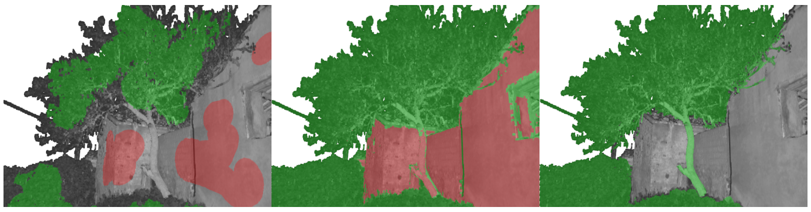


Figure 1: *Machine assisted segmentation: 1. labeling regions that need to be differentiated, 2. run the classifier and 3. touch up the result.*

Abstract

A laser scanning campaign to capture the geometry of a large heritage site can produce thousands of high resolution range scans. These must be cleaned to remove noise and artefacts. To accelerate the cleaning task, we can i) reduce the time required for batch-processing tasks, ii) reduce user interaction time, or iii) replace interactive tasks with more efficient automated algorithms. We present a point cloud cleaning framework that attempts to improve each of these aspects. First, we present a novel system architecture targeted point cloud segmentation. This architecture represents 'layers' of related points in a way that greatly reduces memory consumption and provides efficient set operations between layers. These set operations (union, difference, intersection) allow the creation of new layers which aid in the segmentation task. Next, we introduce roll-corrected 3D camera navigation that allows a user to look around freely while reducing disorientation. A user study showed that this camera mode significantly reduces a userś navigation time between locations in a large point cloud thus accelerating point selection operations. Finally, we show how boosted random forests can be trained interactively, per scan, to assist users in a point cleaning task. To achieve interactivity, we sub-sample the training data on the fly and use efficient features adapted to the properties of range scans. Training and classification required 8-9s for point clouds up to 11 million points. Tests showed that a simple user-selected seed allowed walls to be recovered from tree and bush overgrowth with up to 92% accuracy (f-score). A preliminary user study showed that overall task time performance was improved. The study could however not confirm this result as statistically significant with 19 users. These results are, however, promising and suggest that even larger performance improvements are likely with more sophisticated features or the use of colour range images, which are now commonplace.

Categories and Subject Descriptors (according to ACM CCS): I.3.1 [Picture/Image Generation]: Digitization and Image Capture—Scanning

1. Introduction

Point cloud cleaning is the process of removing unwanted data points from laser range scans and other types of point cloud data. It is part of the pipeline [BR02] that converts laser range scans of heritage sites into 3D models. Unwanted points typically correspond to objects, such as people or cars that were present at scan time as well as noise [TH05]. Heritage scanning expeditions can produce more than 2000 range scans that need to be cleaned manually. Cleaning a single scan can take a person between 30 minutes and 2 hours [RH11]. The two most time consuming parts of the cleaning task are interaction time, and processing time. We can thus either improve the system interface so that the user is more efficient, or improve the algorithms such that they consume less time. A third option is to replace interactive tasks with more efficient automation. In this paper we aim to reduce overall cleaning time by addressing all 3 areas. We demonstrate a roll corrected camera mode that reduces navigation time in editing software. Then we introduce an innovative architecture for representing layers such that less memory is consumed and processing time is reduced. Finally we show how boosted random forests can be used to assist a users with segmentation tasks.

2. Navigation

Heritage point clouds or range scans are 3D virtual environments that the user interacts with during the cleaning process. These interactions can be grouped into *navigation*, *selection*, and *system control* [JHST14, Han97]. Developing an efficient 3D navigation technique for interactive 3D environments is complicated by the need to control 3 degrees of rotational and translation freedom using a 2 dimensional mouse and keyboard inputs. Most 3D modelling and point cloud editing packages use variations of the ArcBall [Sho92] or Virtual Sphere [CMS88] to achieve 3D rotation with 2D inputs. Rotating around a central point is best suited for exploratory movement or object manipulation, rather than the targeted movement used for cleaning. Another 3D navigation to use a first person perspective (FPP) as employed by many games. In this mode the user typically translates the camera position along a horizontal plane using arrow keys while using the mouse to change the camera orientation. FPP navigation, however, requires one to make a trade-off between rotational freedom and potential disorientation. If the camera can be rotated freely it is easy for a person to end up in non-upright position that can be disorientating.

We developed a variation of the FPP camera that mitigates the problem of accidental disorientation without restricting camera rotations. With this technique a person is allowed to freely rotate the camera around the yaw and pitch axis of the local reference frame. This allows the user to look around naturally without being obstructed by invisible barriers. In allowing this level of freedom, the camera can become rolled relative to the world axis. A roll state only becomes undesirable when the horizon is along the line of sight. When looking straight ahead along the horizon, a person expects to be upright. We can use the camera's pitch relative to the ground to develop a heuristic to determine when a roll state is undesirable. As the pitch angle approaches 0, roll becomes undesirable. Roll states are more tolerable as the pitch moves away from 0 since the user's intention is likely to look up or down. This heuristic can be used to determine when to unroll the camera. In our system we apply a small correctional roll rotation to the camera with every interaction when the heuristic indicates that a roll state is undesirable. With successive interactions the effect of the correction slowly nudges the camera back upright. To control the speed of the roll correction a damping coefficient d is used. The correction factor is $d(1-|\cos(\theta)|)$ when $|\cos(\theta)| \leq 45^\circ$ and 0 otherwise.

Evaluation: If a roll corrected FPP camera reduces disorientation, a user should be able to navigate a virtual 3D environment faster with roll correction when encountering disorientated reference frames. To test this hypothesis, 19 users were given two timed navigation tasks. The tasks required them to use our system to navigate from one position in a scan to another. The starting camera orientation was rolled and the final camera orientation had to be level with the ground. Each task was performed under two conditions. In the control condition, roll correction was disabled and in the experimental condition roll correction was enabled. A within-subject counterbalanced design was used. User's were randomly assigned to one of two groups that determined the order in which the experimental and control conditions were presented. The first group started with the control task. To counter learning effects users where primed by giving them time familiarise themselves with the environment. Users were also given a trial run with each task. Users were asked to repeat each navigation task 3 times under control and experimental conditions in order to reduce random error.

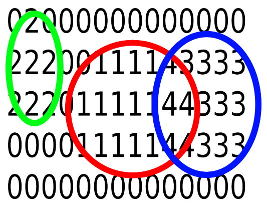
Users performed both tasks significantly faster with roll correction on. In the first task the mean time without roll correction was 60.77s $\pm$ 39.74s. With roll correction the mean time was 25.65s $\pm$ 18.11s. This amounts to a 35.13s improvement that is significance with $p < 0.05$. In the second task the mean time for the control condition was 34.24s $\pm$ 26.60s. Roll correction reduced the time to 24.02s $\pm$ 17.73s. That is a 10.22s reduction that is significance with $p < 0.05$. A double tailed repeated measures t-test was used to compare samples.

3. Layers

The manner in which a user tackles point removal depends largely on the feature set of the software used. *Selections* allow one to temporarily specify what points an operation should apply to. The most relevant action for cleaning is to remove points. *Layers* are generally used to as a more persistent store of point sets, potentially for future operations. Most software packages allow one to hide the points in a layer. This mechanism supports a more iterative cleaning work flow. Instead of immediately discarding points, areas can be hidden from view for later refinement.

In a system aimed specifically at point cloud cleaning, the only requirement is to *annotate* points while maintaining interactivity. Duplicating points during the creation of new layers may cause thrashing when dealing with large point clouds or many layers. Representing layers as a binary maps can be more computationally and memory efficient. An ideal technique would however support extra layers without a linear increase in memory consumption.

We devised a novel layering technique that supports the same operations as maps, with similar performance, without a linear increase in memory consumption for each additional layer. It also supports constant time set operations between layers. The technique allocates n bits per point in the associated point cloud. This allows one to keep track of 2^n layers for the best case and n for the worse case. The n bits associated with each point are used for labelling points. n bits lets us create 2^n unique labels. We use a separate data structure to keep track of layers. Layers are represented as sets of associated labels. Initially each point is assigned the "0" label that we designate as being the state in which a point is not associated with any layer (see figure 2). To create a layer, we assign each associated point a new label, and then associate the label with the layer. In our example we assign a label of 1 to the points in the new red circle, and then associate the label 1 with the red layer (see figure 2). To add an additional non overlapping layer the same process is followed. Creating overlapping layers is not as simple. In figure 2 a new blue layer is created that overlaps with the red layer. Assigning a new integer label to the points in the blue segment would remove the overlapping points from the red layer. This problem is solved by creating two new labels. First we assign "3" to



Layer	Label set
red	1, 4
green	2
blue	3, 4
red ∩ blue	4

Figure 2: *Managing layer labels.*

the points that do not overlap with the red layer. This label is added to the blue layer's set. The overlapping points are given the label "4". This label is then added to both the set of the red and blue label. The blue and red layer end up having the "4" label in common. To create a layer containing the intersection between the red and blue layers we simply find the intersection between the two label sets (i.e. 4), and create a new layer (cf. Figure 2). Union and difference operations can be achieved similarly. Set operation with naive maps would incur a computational and memory cost of $0(n)$ where n is the point cloud size. The number of labels that we can create is limited by the number of layer intersections. In the worst case, each newly created layer overlaps with every other layer and the number of bits allocated for the label will be the maximum number of layers. A 16 bit implementation therefore be limited to 16 layers. If no overlaps occur, 65536 layers could be created.

4. Machine assisted segmentation

The cleaning of terrestrial range images in the heritage domain has not been the topic of much research. The tools used to clean these scans are mostly limited to manual techniques such as brush and polygon lasso selection. Automated tools include automated ground point extraction, wall selection, rooftop extraction, and general purpose clustering. Such tools are mostly lacking when applied to heritage applications since the accuracy is insufficient, or only high enough for very specific use cases.

An effective automated technique should adapt to different contexts while achieving good accuracy. The accuracy should be such that any manual touch up work required should not extend the total task time beyond what it would be when performed.

In this study we focus on segmenting grey scale range images. Working with range images is challenging because the point density of the cloud is non uniform. Range images can however also be interpreted as 2D depth images, which could allow one to apply 2D segmentation techniques such as "GrabCut" [RKB]. Golovinskiy [GKF09] uses graph cuts to segment objects in uniform point clouds of cluttered city streets. We have found that constructing such a graph over a range scan lead to bad segmentations due to non uniform density. Anguelov et al. [ATC*05] demonstrates how graph cuts can be successfully applied to segment range images when the edge weights are determined via supervised learning. Training was performed on 30 thousand training points, the training duration was not reported. Selecting features specific to the objects being targeted is problematic in heritage cleaning as segmentation targets hard to anticipate.

The technique we developed is similar to "GrabCut" [RKB]. Instead of using a graph cut to produce segmentations we apply boosted random forest learning in a similar manner to [CLSB08]. Random forest allows us to use features that are weakly correlated with the target object to build a strong classifier. As in "GrabCut" [RKB] our system lets a user roughly label areas in the scene with different colours using a simple painting interface. When the labelling is complete supervised on-line learning is applied to the training set. This is followed by classification of the remainder of the points (see Figure 1). The principal benefit of this approach is that we can learn a classification scheme tailored to *each* scan that we segment — this should suit the unpredictable nature of heritage scan content far better than trying to learn a general point classifier.

To ensure interactivity is maintained, the training set is sub sampled so that only 5% of the points are used. We use features that are inexpensive to compute and have discriminative power in our sample of heritage sites: 3D positions, normals, intensity returns, and principal components. Normals correlate with surfaces that face the same direction and provided that the range image neighbourhood structure is intact they are inexpensive to compute [KAWB09]. Ordered coefficients of a principal component analysis in a neighbourhood has been shown to discriminate well between edges, corners and planes [BF12, ATC*05]. In our system the neighbourhood is a 200 cm radius. The intensity return of the laser scanner is known to strongly correlate with the type of surface that it reflected off [TP12]. The 3D coordinates of each point enforces some level of spacial coherence between points. We use [CLSB08]'s implementation of boosted random forests with 100 trees, with a maximum depth of 10, and 10 training iterations. This implementation discard weak trees.

Evaluation: Participants (13 men, 6 women) with varying levels of technical experience were recruited for this evaluation. Users were presented with a timed segmentation task. Users were asked to recreate a reference segmentation that was presented to them in a layer. The segmentation had to be recreated with 97% or 95% accuracy, depending on the task. In the control condition for this experiment users started with a clean project file with nothing selected. In the experimental condition users were presented with an existing segmentation that was created by applying machine learning. In order to remove additional sources of variance from the test procedure, one set of segmentation results were presented to all users as the baseline for the machine assisted condition. These segmentations were the result of the labellings depicted in Figure 1 and Figure 3 that were created in 20s each. In the first task the user was required separate a tree and some shrubbery from a house wall (see Figure 1). In the experimental condition the tree and shrubbery was pre-segmented with 91.58% accuracy. The user was tasked to achieve a final accuracy of 97%, from scratch or from the above mentioned starting point. In the second task the windows in a courtyard had to be segmented from a wall (see Figure 3). Our segmentation algorithm achieved 78.69% given the training data as shown in panel one of Figure 3. Users were initially tasked to also achieve a segmentation accuracy of 97%, a pilot study however revealed users required an excessive amount of time to achieve this level of accuracy. The require accuracy was then dropped to 95% which some users still had difficulty achieving. Segmentation accuracy was measured with an F-score. A within-subject counter-balanced design was used again. Users were randomly assigned to

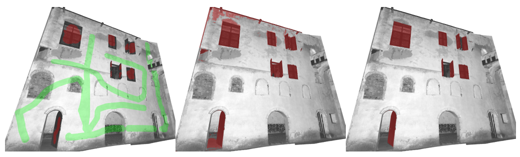


Figure 3: *Machine assisted segmentation (courtyard) 1. labeling, 2. running the classifier and 3. touching up the result.*

two groups. The first group was given the control segmentation task first. Users were primed before starting the experiment. During the priming procedure users were trained how to use the lasso, brush and plane selection tools. Users were then given 3 targets to that they could test each tool on.

Given a coarse user labelling, machine assisted segmentation reduced time to label the remaining points dramatically but no significant reduction in *overall* task duration was found for either task. For task 1 users completed the manual segmentation in an average time of 207s $\pm$ 100s. The machine assisted segmentation was performed in 142s $\pm$ 73s, which is significant for $p < 0.5$, when the estimated labelling and processing time of 29s is added the overall time increases to 163s $\pm$ 78s with no significance ($p < 0.5$). For task 2 users completed the manual segmentation in an average time of 268s $\pm$ 140s. The machine assisted segmentation was performed in 206s $\pm$ 148s, which was not significant, when labelling and processing overhead of 29s time is added the total time is on average 235s $\pm$ 148s also with no significance ($p < 0.5$). A double tailed repeated measures t-test was used to compare samples. It is worth emphasizing that, given the coarse labeling, the time required for user interaction was much lower than for the regular manual cleaning process. However, the results show that the learning process produces an initial scan segmentation that requires a fair amount of "touch-up" time. One possible reason for this is because we only use (X, Y, Z) coordinate values to enforce coherence. Less accurate segmentation results tend to be generated when isolated scan regions are not labeled. This effect can be seen in the top left corner of the second panel of Figure 3. Applying a graph cut approach as in [RKB] or [ATC*05] may produce better results. Better features may also lead more accurate results which could reduce the touch up time required. Most notably, utilising the extra information in colour channels in scans that have them is likely to have a big impact. Statistical significance is expected to be achievable with a larger user sample.

5. Conclusion

We have presented a framework which addresses the acceleration of the cleaning process in three ways. First, we define a roll-corrected mouse interface to allow users to rapidly navigate through the range scan point cloud during editing; secondly, we provide an efficient layer-based point labelling data structure which supports fast memory efficient set-based layer operations; and finally, we introduced a machine learning point classification scheme based on boosted random forests to classify all scan points based on coarse user labelling. The learning algorithm is retrained for each scan based on labelled data provided by a simple paint-by-numbers interface. Preliminary findings show that our framework greatly reduces the time to classify points into related clusters during cleaning, when compared to a full manual point clustering/classification. However, the time required to tweak the generated point labelling, to get high accuracy, is still greater than it needs to be. This can be attributed to (1) an unoptimized collection of features and the lack of a strong spatial neighbourhood constraint, (2) the boosted random forest implementation we used was unoptimized, so additional time could be saved by utilising one of the many highly optimized machine learning libraries that now exist.

With regards to future work, we plan to utilise additional colour features and to explore more complex features and learning algorithms, whilst ensuring that more complex learning algorithms do not unduly push up the overall processing time.

Acknowledgements: spatial data provided by the Zamani heritage documentation project, University of Cape Town.

References

[ATC*05] ANGUELOV D., TASKARF B., CHATALBASHEV V., KOLLER D., GUPTA D., HEITZ G., NG A.: Discriminative learning of markov random fields for segmentation of 3d scan data. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on* (2005), vol. 2, IEEE, pp. 169–176. 3, 4

[BF12] BARNEA S., FILIN S.: Extraction of Objects from Terrestrial Laser Scans by Integrating Geometry Image and Intensity Data with Demonstration on Trees. *Remote Sensing 4*, 12 (jan 2012), 88–110. 3

[BR02] BERNARDINI F., RUSHMEIER H.: The 3d model acquisition pipeline. In *Computer Graphics Forum* (2002), vol. 21, Wiley Online Library, pp. 149–172. 1

[CLSB08] CHRISTIAN A. S., LEISTNER C., SANTNER J., BISCHOF H.: On-line Random Forests. 3

[CMS88] CHEN M., MOUNTFORD S. J., SELLEN A.: A study in interactive 3-d rotation using 2-d control devices. In *ACM SIGGRAPH Computer Graphics* (1988), vol. 22, ACM, pp. 121–129. 2

[GKF09] GOLOVINSKIY A., KIM V. G., FUNKHOUSER T.: Shape-based recognition of 3d point clouds in urban environments. In *Computer Vision, 2009 IEEE 12th International Conference on* (2009), IEEE, pp. 2154–2161. 3

[Han97] HAND C.: A Survey of 3D Interaction Techniques. *Computer Graphics Forum 16*, 5 (1997), 269–281. 2

[JHST14] JANKOWSKI J., HACHET M., SURVEY A., TECHNIQUES I.: A Survey of Interaction Techniques for Interactive 3D Environments. 2

[KAWB09] KLASING K., ALTHOFF D., WOLLHERR D., BUSS M.: Comparison of surface normal estimation methods for range sensing applications. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on* (2009), IEEE, pp. 3206–3211. 3

[RH11] RUTHER H., HELD C.: Challenges in Heritage Documentation with Terrestrial Laser Scanning. In *Proceedings of AfricaGeo* (2011). 1

[RKB] ROTHER C., KOLMOGOROV V., BLAKE A.: GrabCut - Interactive Foreground Extraction using Iterated Graph Cuts. 3, 4

[Sho92] SHOEMAKE K.: ARCBALL: A user interface for specifying three-dimensional orientation using a mouse. *Proceedings of Graphics Interface '92* (1992), 151–156. 2

[TH05] TULEY J., HEBERT M.: Analysis and Removal of artifacts in 3-D LADAR Data. 1

[TP12] TATOGLU A., POCHIRAJU K.: Point cloud segmentation with LIDAR reflection intensity behavior. *2012 IEEE International Conference on Robotics and Automation* (may 2012), 786–790. 3