Efficient Removal of Inconsistencies in Large Multi-Scan Point Clouds

Thomas Kanzok[1] thomas.kanzok

Falk Süß[1]

Lars Linsen[2] l.linsen

Paul Rosenthal[1] paul.rosenthal

[1] Chemnitz University of Technology
Department of Computer Science
Visual Computing Laboratory
Straße der Nationen 62
09111 Chemnitz, Germany
[e.mail]@informatik.tu-chemnitz.de

[2] Jacobs University
School of Engineering & Science
Visualization and Computer Graphics Laboratory
Campus Ring 1
28759 Bremen, Germany
[e.mail]@jacobs-university.de

ABSTRACT

When large scale structures are to be digitized using laser scanning, usually multiple scans have to be registered and merged to cover the whole scene. During the scanning process movement in the scene and equipment standing around – which is not always avoidable – may induce artifacts. Since the scans often overlap considerably another scan of the same area might be artifact-free. In this paper we describe an approach to find these "temporal" artifacts (so-called, because they only appear during one single scan) based on shadow mapping, which makes it implementable as a shader program and therefore very fast. The effectiveness of the approach is demonstrated with the help of large-scale real-world data.

Keywords

Outlier Removal, Point Cloud Processing, Laser Scanning

1 INTRODUCTION

3D-scanning has gained increasing popularity in a wide range of applications, including content creation for games and movies [ARL+10], reverse engineering [IM09] and acquisition of geometric data for documentation of archaeological sites [BGM+09, GSS08, LNCV10] and large-scale structures [PV09, SZW09, WBB+08], where new rendering approaches [KLR12, DRL10] have already helped to reduce the postprocessing time for a dataset. In these applications, usually several scans have to be done and registered in order to capture a complete site or building. During this process it is not always possible to completely close the site – or street, in our particular case – for traffic or visitors, or to ensure that no people or equipment of the scanning team are present in the scene.

This leads to ghost geometry that is only captured by one or few scanners and is sometimes not even consistently colored, since colors are acquired independently from the geometry by different terrestrial laser scanning systems in a second scanning pass. This means that moving objects – which we call "temporary" for the remainder of this paper, because they are not persistent in the scene – introduce artifacts in the scan that compromise the rendering quality considerably (see Figure 1).

A simple observation we made, was that the artifacts induced by temporary geometry are in most cases only present in one of the many merged scans. Since the scans have to overlap significantly to allow robust registration, there is usually at least one scanner that can literally see through such artifacts. The question we have to ask in order to decide whether a point can be considered an artifact is therefore: "Is this point invisible for any other scanner that could potentially see it?" or shorter "Can any other scanner see through this point?"

The second formulation is equivalent to the question whether this point casts a shadow in all other scans that cover it. This suggests applying some variation of shadow mapping to the problem.

In this paper we present an implementation of a shadow-mapping based algorithm for artifact removal in preregistered point cloud datasets. Since all calculations can be done in a shader program on the GPU we can show that the running time of the algorithm is only bounded by the transfer speed of the storage device that holds the data. To achieve this we have to sacrifice some precision, because we can not hold all available

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

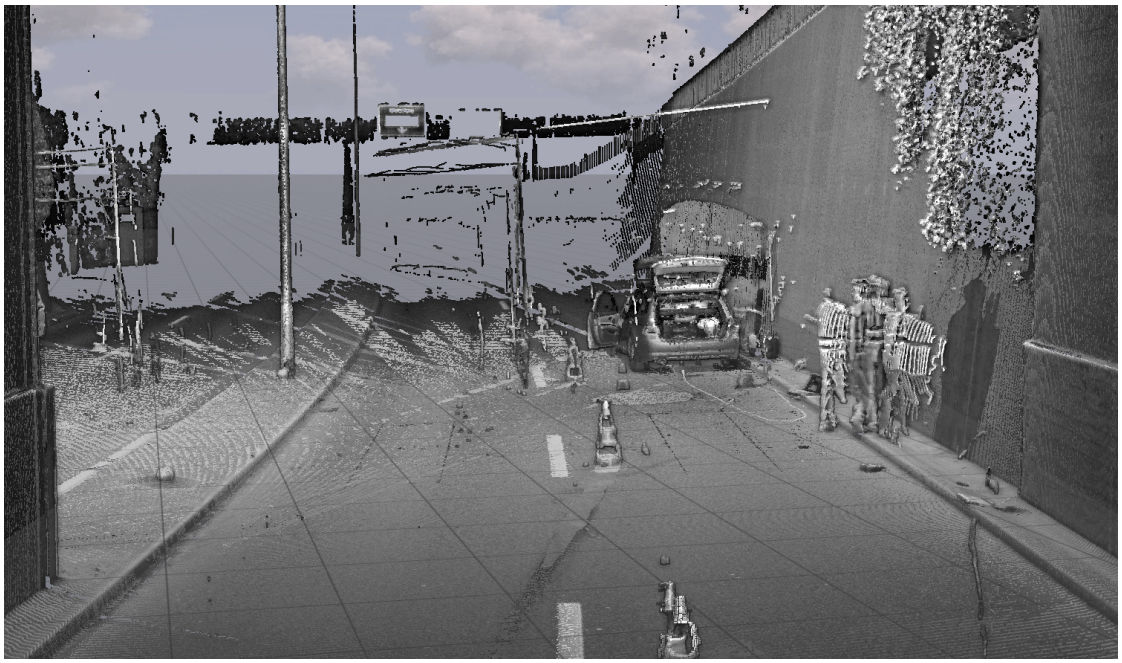


Figure 1: An example for typical temporal artifacts in a scene that was merged from multiple scans. The people in the foreground and the car in the background were only present during one scan. Additionally, there are lots of stripes induced by passing trucks. The scene was shaded to enhance geometry perception.

information on the GPU. This is only critical in the vicinity of edges, however, and can in most cases be treated by detecting these edges in the image.

2 RELATED WORK

Artifacts or outliers are a problem every automated scanning system has to deal with. There are basically two causes for them to occur: inaccuracies in the scanning equipment itself and inconsistencies in the scene that is scanned. Most published approaches do not differentiate between the two sources, so every kind of artifact is treated in the same way and is often characterized as diverging from a statistical distribution. This way the local consistency of the data with a fitted model can be evaluated to identify outliers.

Papadimitriou et al. [PKGF03] detect outliers by comparing the local sampling density of a point to the average local sampling density of its neighbors in a certain radius. A similar approach is used by Sotoodeh [Sot06] to define a *local outlier factor* that is based on the relative local size of the k-neighborhood of a point. Another algorithm based on local coherence was presented by Weyrich et al. [WPK+04]. They define three criteria for outlier classification: the divergence from a fitting plane in the k-neighborhood, the distance of a point to a ball fitted to its k-neighborhood (without the point itself), and the "nearest neighbor reciprocity", which represents the number of nearest neighbors of a point $\mathbf{p}$ that do *not* have $\mathbf{p}$ as a nearest neighbor themselves.

Large scale architectural datasets are mostly comprised of planar elements (walls, floors, etc.), outliers can therefore be assumed to form lines or uncorrelated clusters [WBB+08]. Artifacts can then be found by analyzing the eigenvalues of a point's covariance matrix. Schall et al. [SBS05] use a kernel density estimation scheme to identify outliers in the data as those points, that have a lower local density than the rest of the data, according to an appropriately chosen threshold.

Directly on the scanned range image operates the method of Rusinkiewicz et al. [RHHL02]. They triangulate the range images acquired by the scanner, reject triangles that have exceptionally long edges or are facing away from the scanner and finally delete single points that are not part of any triangle. Other approaches do not remove outliers at all but try to re-integrate them into the surface by some kind of weighted smoothing scheme [MT09, PMG04].

However, these approaches do not take into account that we actually have much more information than raw point coordinates and possibly color. As already pointed out by Köhler et al. [KNRS12] they therefore fail to recognize ghost geometry, because it is consistent with a model – even though temporal – and tend to smooth close outliers into the surface. Their alternative approach takes an additional camera in a structured-light system for surface recognition and uses the light projector for correcting the measured positions.

In the following paper we show another property of real world data that can be used to detect artifacts – the consistency of valid geometry over multiple partial scans of a large scene.

3 GENERAL APPROACH

In this paper we make a clear distinction between "artifacts" and "outliers". While our definition of outliers follows the conventional concept as being points that do not belong to the model according to some local consistency criterion, we denote as *artifacts* points or larger groups of points that are not part of the global model, although they can be perfectly consistent for one single scan (meaning that they are not technically outliers for this scan), since they may represent temporary geometry. Artifacts therefore include all outliers, but also encompass all temporary geometry in the scene.

The property of merged point clouds we want to exploit to identify these artifacts is illustrated in Figure 2. It is based on the simple observation that "permanent" objects, i.e. objects that are not movable and should therefore appear at the same position in each scan, block the laser beams from any scanner involved. This means, if we compute a shadow map for each distinct scanner using only its "own" points, permanent objects cast a shadow in each of these maps. In contrast, "temporary" objects, i.e. objects that moved during scans, only cast a shadow in one or few scans. We can use this to decide whether a point of the final scan is temporary by checking it against each of the computed shadow maps. If the point is shadowed in *all* shadow maps, it is likely to be permanent, otherwise, if it is *not* shadowed in one or more shadow maps, it is most likely temporary. In the following we describe in detail how we come from the raw scanner output to a classification of each point and show results and timings we achieved with the method.

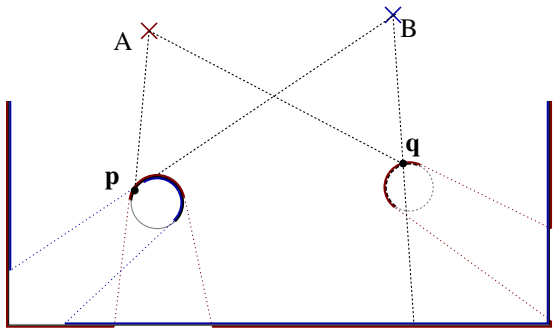


Figure 2: A 2D illustration of the reasoning behind the algorithm. The distance of **p** to B is larger or equal than the respective depth buffer value stored for B, while the distance of **q** (which was only seen by scanner A) to B is smaller than the respective depth buffer value stored for B. This means that **q** has not been seen when scanning from B, while **p** has been seen (unless it is occluded by a point that is even closer to B). Consequently, **q** must have been a removable object, while **p** is a steady object.

Large scale laser scanning systems typically use one laser beam that is reflected by a movable mirror into a spherical direction θ, φ, with θ being the azimuthal and φ being the polar angle. From the time it takes the laser signal to be reflected back to the scanner the spherical distance r can be calculated, which gives the spherical coordinates $\mathbf{p}_s = (r, \theta, \varphi)$ for the point. They then get converted to Euclidean coordinates $\mathbf{p}_e = (x, y, z)$, which is the output of the scanner.

Different scans in one scene then have to be registered into one global system using strategically placed markers and/or local optimization methods, e.g. multiple variants of the well-known ICP algorithm [CM92]. For this work we assume that this registration has already been done as exact as possible and will not go into further detail here.

In order to identify points that are transparent for at least one scanner, we employ a map that facilitates occlusion lookups. Analogous to the original shadow-mapping algorithm we allocate a texture for each scanner that stores the distances from the scanner to the surrounding geometry. Note that this only conveys useful information if we generate the shadow map for each scanner only on the data that was actually acquired by this scanner.

Usually, the numbers of discrete steps taken for θ and φ are fixed and independent, leading to a uniform sampling in θ and φ. To preserve this uniformity and to lose as little information as possible to sampling errors, we store all information for a scanner in one single texture that is parameterized over θ and φ.

The scanner's sampling rate of k steps in θ and l steps in φ then gives us a very good estimate for the optimal resolution of a spherical shadow map. Using a map with domain $[0,k] \times [0,l]$ would ensure that on average each pixel is hit by exactly one laser beam. In practice, this would also mean that we would have to store one third of the size of the entire dataset on the GPU (one float for the spherical radius r instead of three floats for (x, y, z)), which is normally not possible. Additionally, if the sampling grid is not given for the dataset, we have to reconstruct θ and φ from the Euclidean coordinates, which introduces considerable jitter leading to holes in the map. Due to these limitations, the maximum resolution for the shadow map on the GPU should not exceed $\frac{k}{2} \times \frac{l}{2}$ to ensure closed surfaces and should probably be even smaller ($\frac{k}{f} \times \frac{l}{f}; f \geq 2$, where f is chosen such that the maps of every scanner fit in GPU memory).

3.1 Removing Inconsistencies with Shadow Mapping

If the sampling grid of the scanner is not given in advance, each point $\mathbf{p}_e$ of a single scan in Euclidean coordinates can be converted back to spherical coordinates in a shader program and its r-component can be rendered into a shadow texture image S of size $\frac{k}{f} \times \frac{l}{f}$:

$$S(\hat{\theta},\hat{\varphi}) = r \quad \text{with} \quad \hat{\theta} = \frac{k\theta}{2f\pi} \quad \text{and} \quad \hat{\varphi} = \frac{l\varphi}{f\pi};$$

with r being the Euclidean distance of $\mathbf{p}_e$ to the respective scanner origin and $\hat{\theta},\hat{\varphi}$ being the mapping of polar and azimuthal angle to texture coordinates of the render target.

Figure 3: The color texture for the spherical surrounding area of one example scan. The shadow map created from this scan can be seen in Figure 7a.

Having computed a shadow map for each of the n scanners (see Figure 3 for an example of a scanned environment) the shadow maps are store in a layered 3D texture. Using this texture in connection with the original scanner positions, the complete registered point cloud is processed by a vertex shader program, that executes Algorithm 1 and returns its result via transform feedback.

In practice we have to make several tweaks to the algorithm in order to account for different problems which we describe in detail in the following chapters.

3.2 Handling Anisotropy

The first problem is caused by the fact that we will almost never have a spherical scene around a scanner. On the contrary, since scanners usually stand on level ground the angle between scanner and surface declines rapidly with increasing φ. The same is true for long walls, where the angle is a function of θ. Looking up points in shadow map pixels under a very small angle introduces a considerable sampling error, as illustrated in Figure 4a. This could be handled by increasing the threshold ε. However, a higher threshold also increases the rate of false negatives and is therefore to be avoided.

To overcome this problem, we reconstruct the surface equation during texture lookup by fitting a plane to the 7×7-neighborhood of the respective shadow map texel. The distance from a point to this plane then gives a much better estimate for the visibility of said point (see Figure 4b).

For a plane given by a normal $\mathbf{n}$ and a distance l from the origin in the local coordinate frame of a scanner and a point $\mathbf{p}$ in the same coordinate frame we can compute the deviation d of the point as follows:

$$d = <\mathbf{n},\mathbf{p}> - l \tag{1}$$

This solves the problems we experienced in areas with a large but constant depth gradient, for example roads or walls.

Data:
attribute:

- Euclidean point coordinates $\mathbf{p}_e$;

uniform:

- an array of n shadow maps S;
- the positions $\mathbf{s}$ of the n scanners

Result:

- a boolean flag c that indicates, whether the point is an artifact;

```
c = false;
forall the shadow maps S_i do
    calculate (θ̂, φ̂) from p_e;
    if distance(p_e, s_i) + ε < S_i(θ̂, φ̂) then
        c = true;
    end
end
```

Algorithm 1: Shader pseudocode for naive artifact removal. The ε is a tolerance threshold that has to be defined with respect to the data range. The result of this approach can be seen in Figure 6a

3.3 Missing Data

Although we used a smaller size for the shadow map than the original spherical scanner resolution it can still happen that pixel-sized holes remain in the shadow map. Since we are approximating planes over the neighborhood of each pixel this does not pose a serious problem. To make sure that the map is mostly filled, we use a point size of two pixels during rendering of the shadow maps.

The larger problem is how to cope with missing data in the texture. There are three principal reasons for not having any range data from the scanner for a given (θ,φ):

1. The laser beam did not hit any geometry within the maximum distance for the scanner. Consequently, the beam is classified as background.
2. The beam did hit reflecting or scattering geometry, seen at the bottom right of Figure 3, where the note-

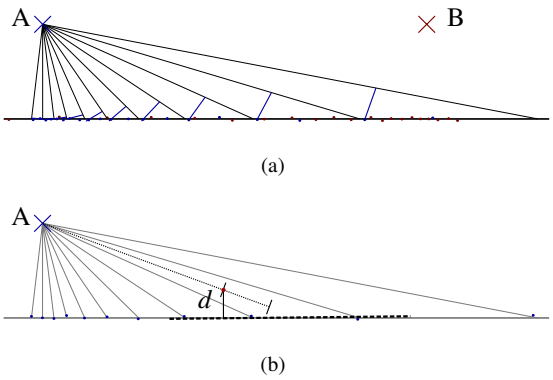


Figure 4: Illustration of the anisotropy problem in spherical shadow maps. (a) The distance from scanner A in the region of densest sampling of scanner B does not suffice to make a decision for the points of B without increasing the threshold considerably. (b) Fitting planes to the neighborhood allows the computation of the exact divergence for a given point **p**. Here the blue points have been reconstructed from the shadow map information and a plane (dashed) was fitted to the neighborhood of the pixel to which **p** was projected. Calculating the distance d to this plane gives a much better estimate of the actual distance than the average distance to the shadow map values (dotted line), especially under acute angles.

book screen and the reflective bands of the cone are missing.

3. The data has already been processed in some way, which was the case with our dataset, where some – but by no means most – erroneous geometry has been removed by hand.

If it was only for the background being hit, we could treat such shadow map pixels as infinitely far away and discard any point in the final dataset that would correspond to this pixel (since obviously at least one scanner saw the background through this point). However, since there are other possible causes for missing data we chose to disregard such empty areas and to make no assumptions on the visibility of any point that would be mapped to there.

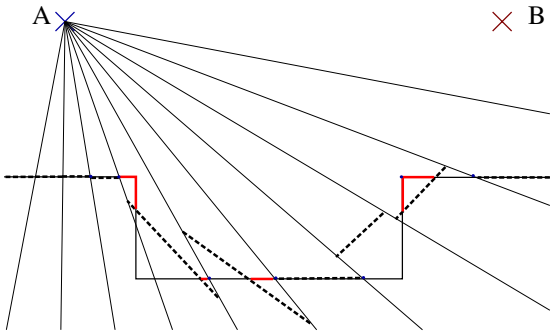
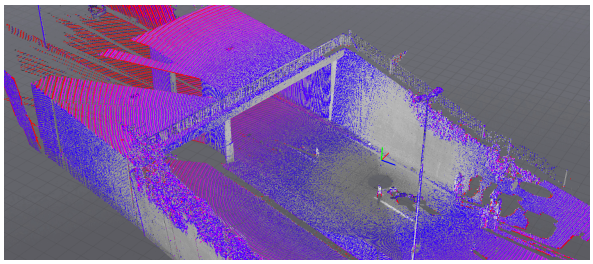
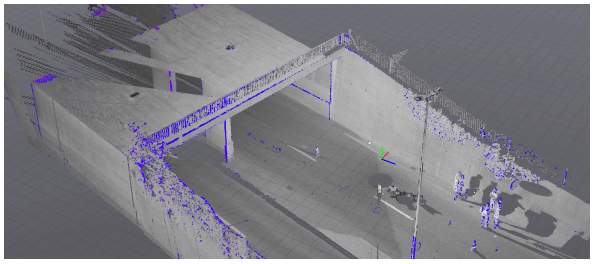


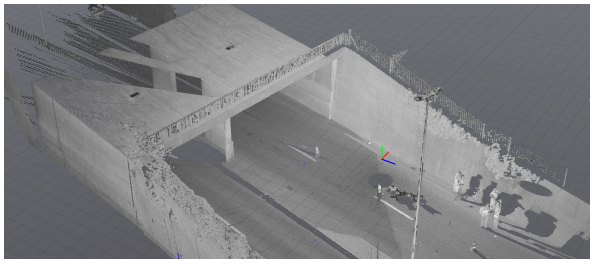
Figure 5: A 2D example for the sensibility of fitting planes in the vicinity of edges. For every texel of the shadow map of scanner A the best fitting plane to its 3-neighborhood is shown as a dashed line, indicated in red are the areas that would have a positive distance to their respective planes, putting them at risk to be classified as outlier.



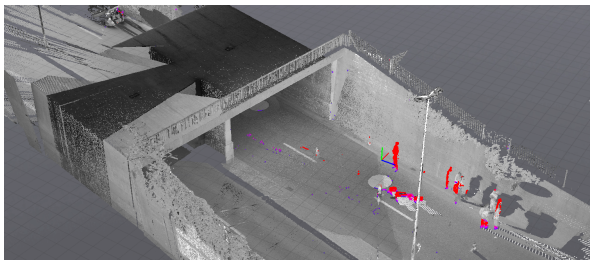
(a)



(b)



(c)



(d)

Figure 6: The different problems we encountered and the solutions found by our algorithm: a) Actual distance minus map distance for a scan tested against its own shadow map using Algorithm 1 and the same when using fitting planes (b). Note that there are still misclassification at the corners, even though they are very small. c) After applying the confidence map nearly all misclassifications have been eliminated. d) The introduction of a second scan facilitates recognition of first artifacts (color coded from blue to red is the respective confidence value).

3.4 Confidence Map

One problem that remains, occurs in areas with diverging gradients, mainly edges and corners, where there are still issues caused by the shadow map resolution. Usually the shadow map resolution is considerably smaller than the resolution of the original dataset, which introduces large errors in the computation of the osculating planes, leading to misclassifications in the vicinity of corners and edges (see Figure 5).

To facilitate more control over the classification we abandon the binary classification of Algorithm 1 in favor of a real certainty value that indicates how likely a given point is an artifact, with $c = 1$ meaning that the point is an artifact for sure and $c = 0$ meaning that the point surely belongs to the model. We can then sum up the weighted distances over all shadow maps and apply a threshold to the mean c to infer a binary classification.

The actual weighting of the classification is done using an additional confidence map for each shadow map, which is 0 on edges between valid points and background and otherwise, in areas that are completely covered by the shadow map, is computed as $1 - \tilde{\sigma}$:

$$\tilde{\sigma}_{u,v} = \sqrt{\frac{1}{|N|}\sum_{i=1}^{|N|}(< \mathbf{n}, \mathbf{p}_i > -l)^2};$$

with $\mathbf{p}_i$ being a reconstructed point from the neighborhood N of a given texel at (u,v) and $\mathbf{n}, l$ being the parameters of the fitted plane, as in Equation 1. This is nothing else than the root mean square error (RMSE) of the fitted plane for a texel. The final confidence value e for a texel is therefore:

$$e_{u,v} = 1 - \begin{cases} \tilde{\sigma}_{u,v} & \Leftrightarrow \quad valid \\ 1 & \Leftrightarrow \quad else \end{cases} \qquad (2)$$

A shadow map texel is considered *valid*, if the four 3×3-corners of the pixel contain enough points to be able to robustly fit a plane to them also (we only required the corners to contain at least four points each themselves, the planes are not actually reconstructed). Otherwise we can assume that there is a large patch of background in the neighborhood of the given texel and filter these border areas out.

Since it can be computed together with the planes themselves, the confidence map does not have to be stored explicitly. Here it is only it is given for convenient illustration (see Figure 7).

After these optimizations we arrive at the refined Algorithm 2.

Overall the algorithm needs none of the usual preprocessing steps like constructing a space partitioning or estimating normals. It simply works by streaming all available data through the GPU twice, comparing the points with each shadow map in the second run, which yields a total runtime in the order of $O(k \cdot n)$, with $k \ll n$ being the number of scans. However, the shadow maps lie in fast GPU memory and the lookups therein are done using specialized hardware units, so the dominant factor is clearly n. The following section gives detailed timings and classification results for a real world dataset with 138 million points.

Result:

- a confidence value c that indicates, how likely the point is an artifact;

```
c = 0;
j = 0;
forall the shadow maps S_i do
    calculate (θ̂, φ̂) from p_e;
    reconstruct set of Cartesian coordinates N over the
    neighborhood of S_i(θ̂, φ̂);
    fit a plane P to the points in N;
    calculate d using (1);
    calculate e using (2);
    if d > 0 then
        c = c + (e * d);
        j++;
    end
end
c = c/j;
```

Algorithm 2: Shader pseudocode for the refined classification algorithm. The input is the same as in Algorithm 1.

4 RESULTS AND DISCUSSION

We tested our approach on a notebook using an Intel i7 Quad-core processor with 2,3 GHz, and an Nvidia GeForce GTX 485m graphics card, running a 64bit Linux. We deliberately chose mobile hardware since we wanted the software to be used under field conditions. The data stems from an architectural scan of a bridge and consists of 138 million points distributed over five scans.

The timings we achieved (see Table 1) were very satisfying. In fact, the actual processing time carries little weight compared to the I/O latency of the hard drive. Although this could be alleviated to some extent by using for example an SSD it would still dominate the running time of the algorithm.

Classification quality depends highly on certain design choices. We present a comparison of results with the different optimization steps of Section 3 in Figure 6. There the progression of the algorithm through Sections 3.1 to 3.4 is depicted on an example dataset (the same dataset that was used to generate the example shadow and confidence maps). One has to note that

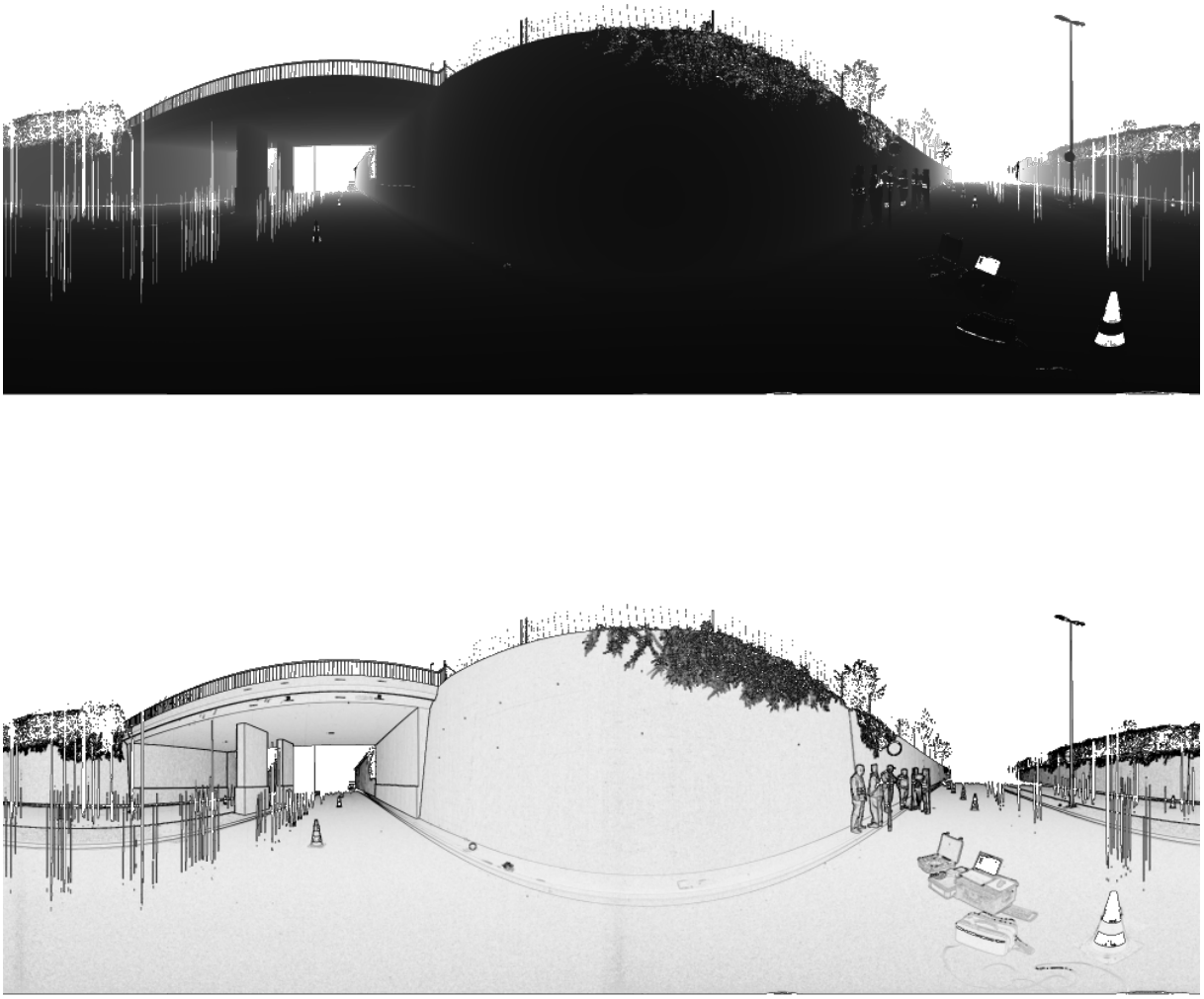


Figure 7: The spherical shadow map produced from the scan seen in Figure 3 (top) and the respective confidence map (bottom). One can see that values taken from texels that constitute edges are assigned a lower weight due to the confidence map and therefore do not compromise the artifact classification.

the confidence value introduced in Equation 2 was chosen for data on a centimeter-scale. That means that the RMSE for different scales has to be adjusted to this, since the deviations may have very small fractional values otherwise, making the confidence extremely high. In our case the RMSE can be interpreted as deviation from the plane in centimeters, implying that all planes with a RMSE higher that 1 cm are rejected (confidence 0).

Another issue that is obvious is that knowing the *exact* scanner position for each scanner is crucial to the performance of the algorithm. If this information is lost after registering the scans, one has to put some effort into aligning the positions exactly. If a new scan is being produced, however, this information should be easily accessible and can be exported into the data. An additional prerequisite is an as-exact-as-possible registration of the data. Otherwise large portions of the dataset may receive a positive c that has to be thresholded appropriately (see Figures 9a). Normally the relative scanning position for each scan is in the origin, making the generation of the maps easy. The absolute positions in the scene are then found during registration. This implies that in order to account for a maximum misregistration of x cm the minimum threshold has to be at least x cm also.

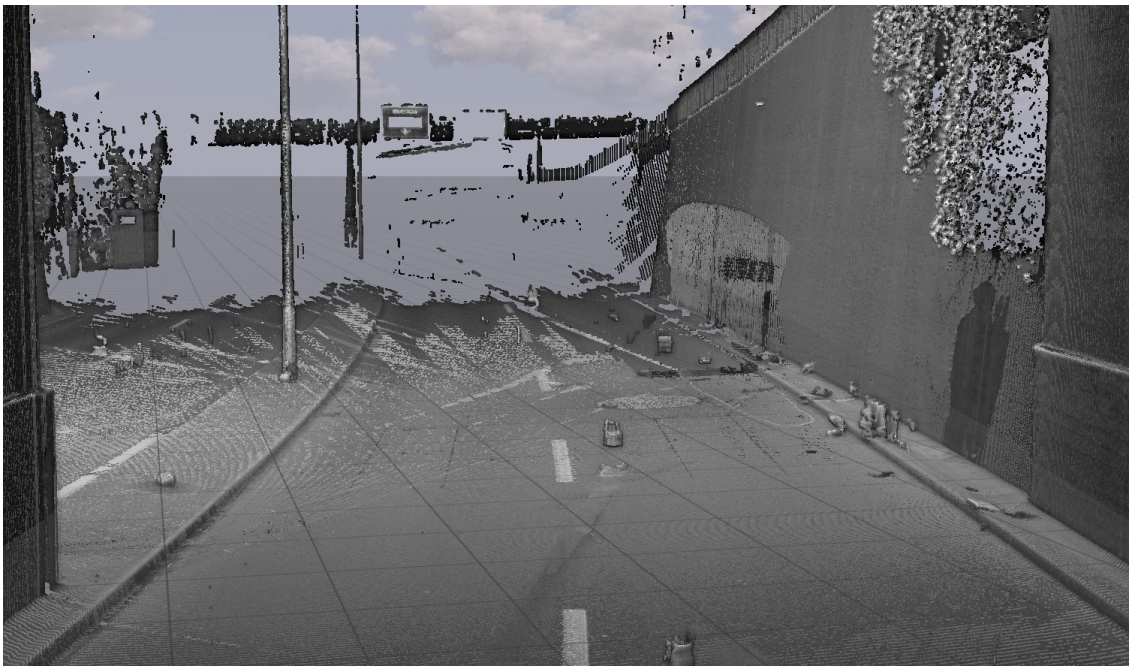


Figure 8: The same view as Figure 1 after applying our artifact removal procedure. The people and the car have been completely removed. However they may leave a shadow, if the lighting in two merged scans differed considerably. Additionally, nearly all of the stripes and parts of the cones were removed.

Scan	# Points	Load	Init	Classify
0	23.0m	5 699 ms	48 ms	1 852 ms
1	38.5m	9 482 ms	82 ms	3 265 ms
2	23.5m	6 414 ms	49 ms	1 943 ms
3	38.2m	9 279 ms	82 ms	3 099 ms
4	15.2m	3 766 ms	31 ms	1 318 ms
$\sum$	138.4m	34 659	292	11 477
Total Time		80 s ($\approx 2 \cdot \text{Load} + \text{Init} + \text{Classify}$)		

Table 1: Processing times for a 138 million point dataset. The complete processing took approximately 80 seconds. In the table the times are divided in "Load", i.e. the transfer of the data from the hard drive to the GPU, "Init", i.e. the initialization and rendering of the shadow-map and "Classify", i.e. the classification of all points of the dataset according to the shadow-maps of all scans using Algorithm 2. The classification has to use all shadow maps, but since they are of equal size the classification time per map equals to the total time divided by the number of scans, i.e. $\frac{\text{classification time}}{5}$ in our case. Note that in the total processing time the "Load" component appears twice, since we streamed the data from the hard drive again for classification.

A peculiar observation we made was that some of the objects in the scene, in particular a traffic cone under the bridge, have only been slightly moved in between scans – probably by accident. This slight dislocation allowed for a removal of one of the partial cones (Figure 9b). Knowing this, it might be beneficial to at least slightly move equipment that can for some reason not be removed completely from the scene in between the scans.

5 CONCLUSION

We have presented an approach for artifact classification in large point clouds comprised of multiple scans. The algorithm can for the most part be implemented on the GPU with linear asymptotic complexity. It results in a confidence value for each point that can easily be evaluated by the user using color coding. With this information the user is able to choose a threshold via a slider to remove the found points. Thanks to the edge-sensitivity it works very conservative, although that means that certain artifacts can remain in the dataset, because no other scanner could confidently reject them. However, since the classification can be done within minutes, this can also be used to infer a position for additional scans during a campaign. To improve the robustness of the approach, slight dislocations of light equipment between scans can already have a huge effect.

6 ACKNOWLEDGMENTS

The authors would like to thank the enertec engineering AG (Winterthur, Switzerland) for providing us with the data and for their close collaboration. This work was partially funded by EUREKA Eurostars (Project E!7001 "enercloud - Instantaneous Visual Inspection of High-resolution Engineering Construction Scans").

7 REFERENCES

[ARL+10] O. Alexander, M. Rogers, W. Lambeth, J. Chiang, W. Ma, C. Wang, and P. Debevec. The Digital Emily Project: Achieving a Photorealistic Digital Actor. *IEEE Comput. Graph. Appl.*, 30(4):20–31, July 2010.

[BGM+09] F. Bettio, E. Gobbetti, F. Marton, A. Tinti, Emilio Merella, and Roberto Combet. A Point-based System for Local and Remote Exploration of Dense 3D Scanned Models. In Debattista et al. [DPPS09], pages 25–32.

[CM92] Y. Chen and G. Medioni. Object modelling by registration of multiple range images. *Image and Vision Computing*, 10(3):145 – 155, 1992.

[DPPS09] Kurt Debattista, Cinzia Perlingieri, Denis Pitzalis, and Sandro Spina, editors. *VAST09: The 10th International Symposium on Virtual Reality, Archaeology and Intelligent Cultural Heritage*, St. Julians, Malta, 2009. Eurographics Association.

[DRL10] P. Dobrev, P. Rosenthal, and L. Linsen. Interactive Image-space Point Cloud Rendering with Transparency and Shadows. In Vaclav Skala editor, *Communication Papers Proceedings of WSCG, The 18th International Conference on Computer Graphics, Visualization and Computer Vision*, pages 101–108, Plzen, Czech Republic, 2 2010. UNION Agency–Science Press.

[GPAR04] Markus Gross, Hanspeter Pfister, Marc Alexa, and Szymon Rusinkiewicz, editors. *SPBG'04 Symposium on Point - Based Graphics*, Zürich, Switzerland, 2004. Eurographics Association.

[GSS08] L. Grosman, O. Smikt, and U. Smilansky. On the application of 3-D scanning technology for the documentation and typology of lithic artifacts. *Journal of Archaeological Science*, 35(12):3101–3110, 2008.

[IM09] L. Iuliano and P. Minetola. Enhancing moulds manufacturing by means of reverse engineering. *The International Journal of Advanced Manufacturing Technology*, 43(5–6):551–562, 2009.

[KLR12] T. Kanzok, L. Linsen, and P. Rosenthal. On-the-fly Luminance Correction for Rendering of Inconsistently Lit Point Clouds. *Journal of WSCG*, 20(2):161 – 169, 2012.

[KNRS12] J. Köhler, T. Nöll, G. Reis, and D. Stricker. Robust Outlier Removal from Point Clouds Acquired with Structured Light. In *Eurographics 2012-Short Papers*, pages 21–24. The Eurographics Association, 2012.

[LNCV10] J. L. Lerma, S. Navarro, M. Cabrelles, and V. Villaverde. Terrestrial laser scanning and close range photogrammetry for 3D archaeological documentation: the upper palaeolithic cave of parpalló as a case study. *Journal of Archaeological Science*, 37(3):499–507, March 2010.

[MT09] H. Masuda and I. Tanaka. Extraction of Surface Primitives from Noisy Large-Scale Point-Clouds. *Computer-Aided Design and Applications*, 6(3):387–398, 2009.

[PKGF03] S. Papadimitriou, H. Kitagawa, P. B. Gibbons, and C. Faloutsos. Loci: Fast outlier detection using the local correlation integral. In *Data Engineering, 2003. Proceedings. 19th International Conference on*, pages 315–326, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

[PMG04] M. Pauly, N. J. Mitra, and L. J. Guibas. Uncertainty and Variability in Point Cloud Surface Data . In Gross et al. [GPAR04], pages 77–84.

[PV09] S. Pu and G. Vosselman. Knowledge based reconstruction of building models from terrestrial laser scanning data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 64(6):575–584, November 2009.

[RHHL02] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy. Real-time 3d model acquisition. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '02, pages 438–446, New York, NY, USA, 2002. ACM.

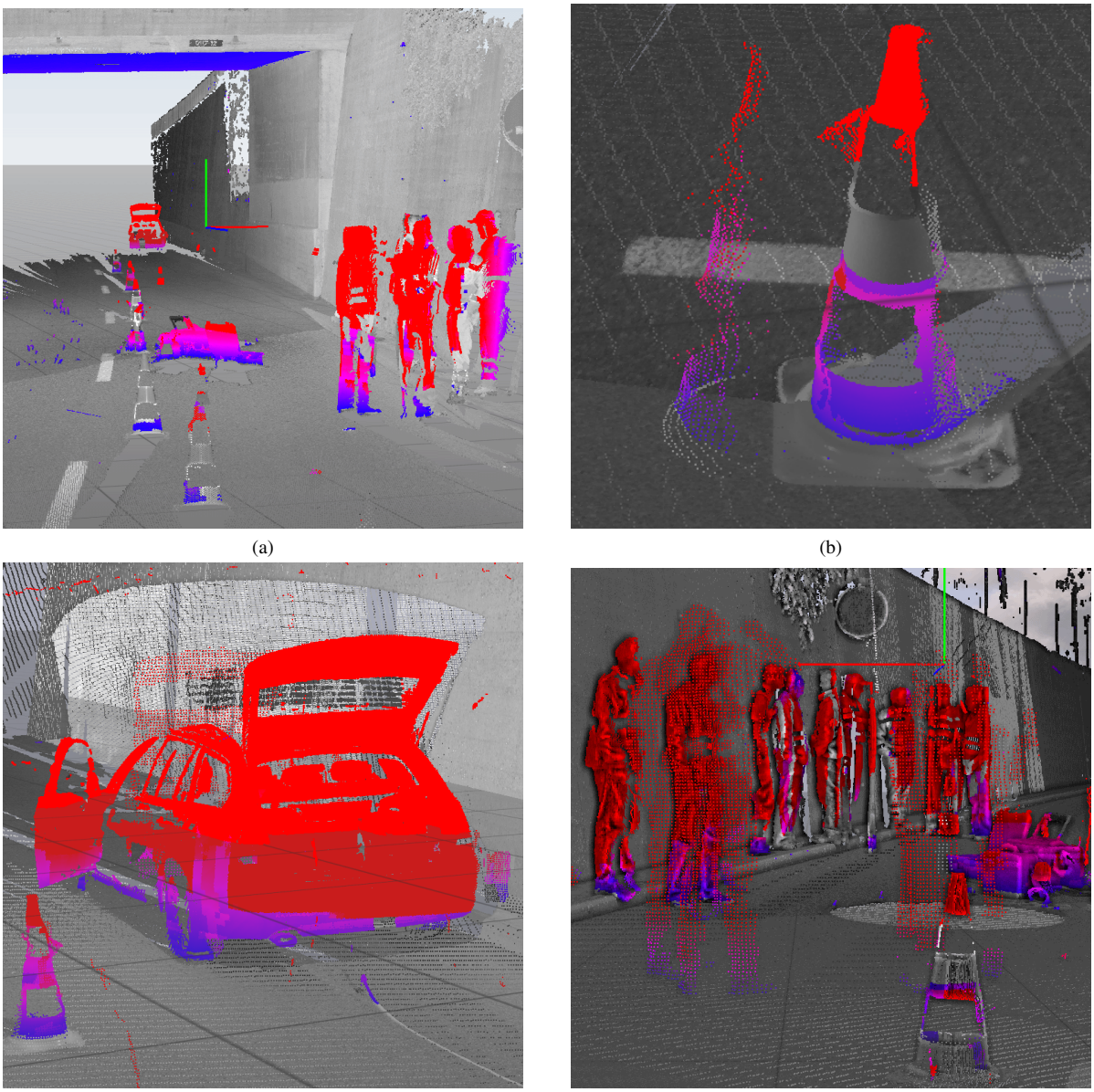
[SBS05] O. Schall, A. Belyaev, and H. Seidel. Robust filtering of noisy scattered point data. In *Proceedings of the Second Eurographics / IEEE VGTC conference on Point-Based Graphics*, SPBG'05, pages 71–77, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.

[Sot06] S. Sotoodeh. Outlier detection in laser scanner point clouds. In *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences XXXVI-5*, pages 297–302, 2006.

[SZW09] C. Scheiblauer, N. Zimmermann, and M. Wimmer. Interactive Domitilla Catacomb Exploration. In Debattista et al. [DPPS09], pages 65–72.

[WBB+08] M. Wand, A. Berner, M. Bokeloh, P. Jenke, A. Fleck, M. Hoffmann, B. Maier, D. Staneker, A. Schilling, and H. Seidel. Processing and interactive editing of huge point clouds from 3D scanners. *Computers & Graphics*, 32(2):204–220, 2008.

[WPK+04] T. Weyrich, M. Pauly, R. Keiser, S. Heinzle, S. Scandella, and M. Gross. Post-processing of Scanned 3D Surface Data. In Gross et al. [GPAR04], pages 85–94.



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

Figure 9: Some classifications inferred by our algorithm. For these images a threshold of 2 cm was used. The blue area on the ceiling in (a) is due to inaccurate registration of the scans and has to be taken care of by choosing an appropriate threshold. The cone in (b) was slightly displaced during scans, allowing at least the artifact from one scan (left) to be completely recognized. c) The car was apparently completely removed for at least one scan, which made it almost completely recognizable. Note the slight shadow of the tailgate, indicating that this car was slightly displaced also present in a second scan. d) A group of people that was present at roughly the same place during different scans. Note that parts of the group in the background could not be classified, since they were shadowed by other people closer to the scanners (Shading was added to improve geometric structure perception).