Efficient Colorization of Large-scale Point Cloud using Multi-pass Z-ordering

Sunyoung Cho[§], Jizhou Yan[†], Yasuyuki Matsushita[†], and Hyeran Byun[§]

Yonsei University[§], Microsoft Research Asia[†]

Abstract

We present an efficient colorization method for a large scale point cloud using multi-view images. To address the practical issues of noisy camera parameters and color inconsistencies across multi-view images, our method takes an optimization approach for achieving visually pleasing point cloud colorization. We introduce a multi-pass Z-ordering technique that efficiently defines a graph structure to a large-scale and un-ordered set of 3D points, and use the graph structure for optimizing the point colors to be assigned. Our technique is useful for defining minimal but sufficient connectivities among 3D points so that the optimization can exploit the sparsity for efficiently solving the problem. We demonstrate the effectiveness of our method using synthetic datasets and a large-scale real-world data in comparison with other graph construction techniques.

1. Introduction

This paper addresses the issue of assigning colors to a large-scale 3D point data from multi-view images as illustrated in Fig. 1. With the recent developments in various 3D sensing modalities, it is becoming feasible to obtain a large-scale 3D point data. For the tasks of 3D visualization or segmentation of a 3D point cloud, assigning color appearances to the points is an important process. This is a trivial problem if we neglect the camera calibration error, inconsistent appearances across views and occlusions, because simply projecting 3D points to the image coordinates of any of the views would accomplish the goal. However, in reality, camera calibration is not exact, and appearances may vary across views due to view-dependent reflectances or exposure variations and projection errors. In addition, the massive 3D point data that can be easily obtained by modern devices renders another challenge to this problem in terms of computational tractability.

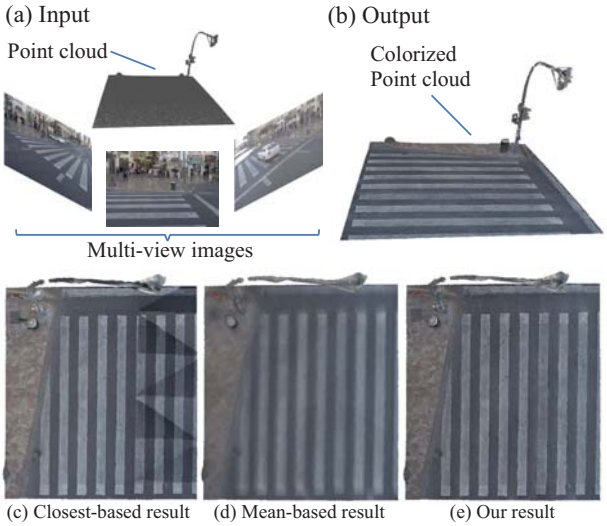


Figure 1: First row shows our problem setting. (a) Our input is point cloud and multi-view images, (b) Output is a colorized point cloud. The second row is an example showing two problems of 3D point cloud colorization. (c) Colorization using combination of closest views, (d) Mean color of all views, (e) Our colorization. There are color inconsistency in (d) between distinct view images and blurry boundary from projection error in (e).

In this paper, we present an efficient 3D point cloud colorization method that takes a large-scale 3D point cloud and corresponding multi-view images as input, which may contain calibration error and inconsistent view-dependent appearances. Our method casts the color assignment problem as an optimization using a sparse graph structure, which effectively ensures smooth and faithful color assignments. At the heart of the proposed method, we introduce a *multi-pass Z-ordering technique* for fast creation of a *sparse* and *connected* graph structure from a set of unordered 3D points on the unstructured grid. The proposed multi-pass Z-ordering method is advantageous over a conventional k-

Part of this work has been done while the first and second authors are visiting MSRA as research interns.

nearest neighbor (k-NN) graph construction, because it ensures the connected graph as output, which is crucial for the colorization. It is also computationally efficient in comparison with the Delaunay triangulation-based mesh creation technique.

We conduct quantitative and qualitative evaluation of the proposed method using synthetic data that contains camera calibration error and multi-view color inconsistencies, and compare our method with other graph generation methods, such as k-NN and Delaunay triangulation. We also show the result of our method applied to a large-scale real-world outdoor data.

2. Related Work

Point-based surface representation has been shown effective for large-scale or complex geometric data as it naturally avoids the expensive computation time for generating consistent triangular meshes. The output of our point colorization method can be used for point-based rendering [22, 28, 24], and also for segmentation of a point cloud [25, 2].

One of the major challenges in assigning colors to the 3D point cloud is to reduce the color discrepancy among multi-view images. To deal with this issue, color correction methods have been studied in multi-view image and video stitching, which is the process of correcting the color differences between neighboring views that arise due to different exposure levels and viewing angles. Among them, exposure compensation [17, 5, 27] has studied color balancing in the multi-view stitching problem, and color transfer techniques [21, 26, 18] have been developed for achieving the similar goal with a more emphasis on establishing the mapping function. Kim and Pollefeys [18] apply their color transfer algorithm to stereo sequences, and its resulting radiometrically aligned images are used for generating a texture-mapped 3D model without color inconsistency. They propose a likelihood approximation method of the brightness transfer function in the joint histogram between two overlapped images. Bannai *et al*. [5] also apply their color correction method for multiple texture maps, and its corrected texture maps are used for constructing the consistent 3D model. Their method first constructs a graph of texture map patches, and spreads the color matching error across all multi-view images using unconstrained nonlinear optimization. Our work also reduces the color inconsistency in multi-view images; however, we perform the color fusion using all multi-view observations simultaneously without setting any particular reference image.

There are some color blending approaches for assigning colors given an uncolorized 3D mesh and scan images. Chuang *et al*. [9] define the Laplace-Beltrami operator using restrictions of 3D functions to the surface. Their operator allows an efficient multi-resolution hierarchy for solving the Poisson equation, and produces the seamless colorization result. Li *et al*. [19] reconstruct the texture of 3D surface using a method of [6, 7] that recovers illumination as well as reflectance from a single image based on shading cues. Most of these methods mainly focus on colorization of small 3D objects. Unlike them, our goal is to develop a scalable colorization technique for a large-scale 3D point cloud. In this context, our work is closely related to Shan *et al*. [23], which estimates albedo and normal for each vertex in the triangle mesh given a city-scale 3D model. Their method optimizes an energy function that consists of ambient + diffuse shading model, lighting, and surface normal in the mesh. While effective, their method requires several optimization steps for each vertex because of various unknown parameters. Our method can be viewed as a simpler form of their problem, where we are interested in assigning colors (albedo), resulting in an efficient closed form solution once a sparse graph is defined on the entire point cloud.

3. Proposed Method

We first introduce the notations that are used in the rest of paper. Let us assume that multiple colors for each point are obtained from its corresponding image pixel colors of calibrated multi-view images. Let $P = \{p\}$ denote a set of point indices of the 3D point cloud, where p is the point index, and let $V = \left\{\mathbf{v}_p^i\right\}$ denote the observed color vector of the corresponding image pixel in the i-th view image and $C = \{\mathbf{c}_p\}$ be the color vector that is going to be assigned to the point p.

The proposed method begins with creation of a sparse graph, *i.e*., construct a sparse graph $G = \{P, E\}$ with the given point cloud P, where a node corresponds to a 3D point p, and an edge $e = (p, q) \in E$ represents the connection between nodes corresponding to 3D points p and q. Our goal is to optimize $\{\mathbf{c}_p\}$ based on $G = \{P, E\}$ and V.

3.1. Graph construction

Unlike image pixels on a regular grid, our 3D point cloud has an unstructured distribution. Therefore, it is not straightforward to define edges among nodes (or points). One may define a fully or densely connected graph by spanning many edges. However, given a large-scale point cloud, the graph needs to be sparse, containing as small number of edges as possible, in order to make the problem computationally tractable. One of the most straightforward approaches to defining a sparse graph to the 3D point cloud may be to compute k-nearest neighbors (k-NN) for each node and link them together. However, such an approach may yield isolated subgraphs, which prohibits the global optimization, unless a sufficiently large k is used, which in turn introduces computational difficulty.

There are various methods to obtain a connected graph with geometric approximations, and they can be categorized

into two types of approaches; geometry-based and graph-based approaches. In geometry-based approaches, the connection between points is determined by geometric ordering of points, whereas graph-based approaches use subgraph search algorithms based on the graph topology. The geometry-based approaches, such as traversal of spatial indexing trees [14, 15] or space filling curves [16, 20], can be immediately applied to point data, while graph-based approaches, such as spanning tree [13] or graph traversal [11], require conversion of the original point data into a graph representation, whose construction process generally takes $O(n \log n)$ as a pre-processing [4]. For our problem dealing with massive data, a geometry-based approach is more suitable because of its computational efficiency.

To achieve this goal, we employ a Z-order curve method proposed by [20, 10], which connects the all points in the order of points' Z-value calculated by interleaving the binary representation of its coordinate values. Suppose, for example, there are three 3D points, $\mathbf{x}_{p_1} = (2,3,1)^T$, $\mathbf{x}_{p_2} = (1,4,5)^T$, and $\mathbf{x}_{p_3} = (1,2,7)^T$ with the integer coordinates $0 \leq x,\ y,\ z \leq 7$. Their Z-values become `000110011`, `011000101`, and `001011101`, respectively. Finally, the points are connected with the order of $p_1 \rightarrow p_3 \rightarrow p_2$. The method produces the recursively Z-shaped curve that multi-dimensional data is mapped to one-dimensional data while the locality of the points is preserved.

The Z-ordering method is one of the space filling curves and has the time complexity of $O(n \log n)$ without pre-processing; thus, it has an advantage in its efficiency. Figure 2 shows the conceptual illustration of the Z-order curve as compared to other graph generation methods such as k-NN graph and graph of Delaunay triangulation. We can see in this particular example that k-NN graphs with $k = 1$ and 2 produce isolated subgraphs. Although k-NN graph with $k = 3$ yields a connected graph, k may need to be a large value to guarantee a connected graph due to a large amount of points in more realistic settings. The Delaunay triangulation is a conventional method for generating a graph from point data. The graph obtained from Delaunay triangulation is connected and closes to a complete graph from the large number of edges, while the Z-ordering curve produces a connected graph with a much smaller number of edges.

3.2. Point color optimization

Once a sparse connected graph G is obtained by the method described in the previous section, we take an optimization approach for assigning a color vector to each node. Namely, we minimize the following energy function that consists of data and smoothness terms :

$$E(C) = \sum_{p \in P} D_p(C) + \lambda \sum_{(p,q) \in N} S_{p,q}(C), \qquad (1)$$

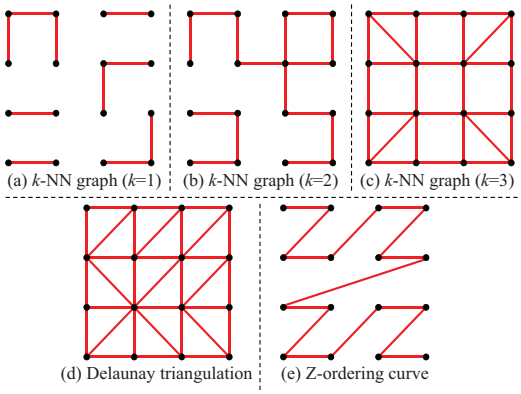


Figure 2: Conceptual illustration of various graphs. (a), (b) and (c) indicate k-NN graphs with $k = 1$, 2, and 3, respectively. (d) Delaunay triangulation, (e) Z-ordering curve.

where C is the set of color vectors to be assigned to the point cloud P. The first term D_p is a data term for the point p, $S_{p,q}$ is a smoothness term of the points p and q, N is the set of node pairs linked in G, and λ is a weighting factor for the smoothness term. In our implementation, we consistently set $\lambda = 1.0$. We now explain the definitions of the data and smoothness terms.

Data term. The data term encourages the color vector c_p to become similar to the mean color of all the point p's corresponding pixel colors in multi-view images. The data term for point p is defined as

$$D_p(C) = a_p \|\mathbf{c}_p - \overline{\mathbf{v}_p}\|_2^2, \qquad (2)$$

where $\mathbf{c}_p$ is the color vector to be assigned to the point p, and $\overline{\mathbf{v}_p}$ is the mean color of the point p in all the visible images. a_p is a binary mask that indicates whether the point p has a valid data term ($a_p = 1$) or not ($a_p = 0$), and is used for activating *good* data terms. This is to avoid the blurry result depicted in Fig. 1(d) that is obtained by activating terms of all points. We choose to use the mean color $\overline{\mathbf{v}_p}$ instead of a single color sampled from one view in order to avoid the view-dependent appearances shown in Fig. 1(c). The validity a_p can be activated randomly, or by choosing points p where multi-view colors are consistent. In our experiment, we randomly select 10% of points from the point cloud P and set their $a_p = 1$.

Smoothness term. Our smoothness term S enforces the smooth color transition between connected nodes p and q with avoiding over-smoothing. It is defined as the difference of color gradients as

$$S_{p,q}(C) = \|(\mathbf{c}_p - \mathbf{c}_q) - \mathbf{g}(p,q)\|_2^2, \qquad (3)$$

where $\mathbf{g}(p,q)$ is the color difference between points p and q. Among multiple colors at a scene point observed from

different views, we choose the set of views within a certain distance to avoid errors caused by large mis-alignment. Therefore, the color difference $\mathbf{g}(p,q)$ is designed as

$$\mathbf{g}(p,q) = \sum_{i \in (U(p) \cap U(q))} \left(\mathbf{v}_p^i - \mathbf{v}_q^i\right) / |(U(p) \cap U(q))|, \quad (4)$$

where $U(p)$ is the set of views within the threshold distance for point p. In this manner, we choose the view indices i that are close to both p and q.

Optimization. The energy function of Eq. (1) is a typical optimization problem that can be represented as a linear system $\mathbf{Ax} = \mathbf{b}$. By setting the first-order derivative of E of Eq. (1) to be zero, we obtain

$$a_p(\mathbf{c}_p - \overline{\mathbf{v}_p}) + \lambda \sum_{(p,q) \in N} (\mathbf{c}_p - \mathbf{c}_q - \mathbf{g}(p,q)) = \mathbf{0}. \quad (5)$$

We optimize all the colors $\mathbf{C}$ at once. By re-arranging $\mathbf{c}_p$ across all the point p in a vector $\mathbf{x}$, Eq. (5) can be written in a matrix form $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is a $m\,|P| \times m\,|P|$ matrix for data with m color channels, and $\mathbf{b}$ is a vector that includes the color differences between neighboring nodes and mean colors $\overline{\mathbf{v}_p}$ for the nodes where $a_p = 1$. The p-th diagonal element of the matrix $\mathbf{A}$ has the value of $\sum_{(p,q) \in N} 1 + a_p$, and its horizontal neighbors have the values of -1. Therefore, the matrix $\mathbf{A}$ has a sparse structure and can be efficiently solved even though the dimensions of the matrix $\mathbf{A}$ is large. We use a SuiteSparse QR solver [12] for solving the large sparse linear system.

3.3. Multi-pass Z-ordering

In the connected graph obtained by Z-ordering, each node has two neighboring nodes except for the first and last nodes. For the purpose of point colorization, it typically requires more supports from neighboring nodes in order to ensure the visual smoothness, otherwise it produces visible artifacts as shown in Fig. 3(b). To avoid this problem, we propose to perform *multi-pass* Z-ordering, *i.e.*, Z-ordering is applied multiple times by altering coordinate systems. Since our problem is based on 3D coordinates, there are four possible passes ($1:(x,y,z)$, $2:(-x,y,z)$, $3:(x,-y,z)$, and $4:(x,y,-z)$) obtained by reflecting each coordinate. For the n-dimensional space, there are 2^{n-1} different possible passes for Z-ordering, which are the combinations of positive and negative directions for each dimension. We observed that using the full set of passes produces the best result compared with the result with its subset. For the case of using all passes, each node has 4.49 neighboring nodes on average. Figure 3 shows the comparison between single and multi-pass Z-ordering results.

(a) Two-pass Z-ordering curve (b) Colorization result of one-pass Z-ordering (c) Colorization result of two-pass Z-ordering

Figure 3: Comparison between Z-ordering and multi-pass Z-ordering results. (a) shows the illustration of our connected graph. In (a), black points are nodes. The dark lines show the one-pass of the Z-ordering curve and red lines are another pass of the Z-ordering curve using the different coordinates. (b) and (c) are the results of single and two-pass Z-orderings.

3.4. Massive-scale 3D point cloud colorization

While the solution method described in Sec 3.2 can handle a large-scale point cloud by exploiting the sparse graph structure, the size of the graph is practically limited by the PC's memory size. To handle a massive amount of observations that cannot be accommodated on PC's memory, we solve the problem by splitting the entire 3D point cloud into several chunks.

This problem is equivalent to partitioning a graph into several subgraphs. When generating subgraphs, it is important to minimize the number of edges spanning across separated subgraphs. This type of problems has already been studied as a graph partitioning problem [8]. If there are no edges among subgraphs, the matrix $\mathbf{A}$ becomes a block diagonal matrix. If $\mathbf{A}$ is a block diagonal matrix with q submatrices on the diagonal, then $\mathbf{A}$ is invertible if and only if $\mathbf{A}_a$ is invertible for $1 \le a \le q$. In this case $\mathbf{A}^{-1}$ is also a block diagonal matrix, identically partitioned to $\mathbf{A}$, with $\left(\mathbf{A}^{-1}\right)_a = (\mathbf{A}_a)^{-1}$. It means that results of global and local methods are same if we can split a graph without having edges spanning across subgraphs. However, this graph partition problem is unfortunately an NP-complete problem [3]. In addition, when the original graph is too large to store in the memory, we cannot directly apply a graph partitioning method.

We, therefore, take a strategy of first splitting the entire point cloud into several chunks with a constant size, *i.e.*, segmenting the original data using a cubic grid, and performing colorization by constructing a graph for each chunk. Because of the fact that our graph is extremely sparse, *i.e.*, each node only has $2 \sim 4$ connected nodes among several hundred million nodes, we have observed that this local method does not suffer much from the simple partitioning as long as the size of sub-matrix is sufficiently large. To further reduce the potential color discrepancy across neighboring chunks, we split the entire point cloud by allowing some overlaps among neigh-

boring chunks. These overlaps enable to include the same points within boundary regions across neighboring chunks for colorization and consequently produce consistent colors among neighboring chunks. In our implementation the ratio of overlap is set to $\omega = 0.1(= 10\%)$.

4. Results

We first qualitatively and quantitatively evaluate our method using three synthetic datasets and compare our method with other graph generation methods. Next, we apply our large-scale colorization method on large-scale datasets that are taken from the real-world.

4.1. Experiments on Synthetic Data

Using the *Bunny*, *Dragon*, and *Armadillo* models [1], we render multi-view images in 1280×960 resolution from 48 different viewpoints using the Lambertian shading model. For each model, we generate observation datasets by adding noise to (1) camera extrinsic parameters, (2) pixel intensities, and (3) both camera extrinsics and intensities. The input point cloud is generated by uniformly sampling the original mesh model.

We compare our method with other graph generation methods: k-NN with $k = 5$ and 20, and Delaunay triangulation (DT). We consider two variants of our approach; single-pass (1p) and 4-pass Z-ordering (4p Z-order).

To assess the colorization quality, we use the average intensity error per point (AEP), and ratio of the number of erroneous points whose error is greater than an intensity threshold $t(= 20)$ (REP). We also compare the computation time measured on a system equipped with an Intel(R) Xeon(R) 16 cores 2.90 GHz CPU and 256 GB memory.

Table 1 shows the quantitative evaluation for synthetic datasets under various types of noise and graph generation methods. Our method outperforms the other graph generation methods in terms of error and computation time. For the k-NN graph, a small k produces significant artifacts due to isolated subgraphs. Although k-NN graph with $k = 20$ produces a similar result to our method, the computation time becomes much longer than our method. We also use DT to generate the graph from point data. While DT does not guarantee to produce a connected graph in the 3D space, in these examples, it consistently produces plausible result. However, due to the density of the graph, it requires significantly longer computation times.

Figure 4 shows the colorization result of the synthetic scenes. The closest-view method, where a color is assigned directly from the point's closest view, produces erroneous results containing visible color inconsistency due to the noise in camera extrinsics. The mean color based method ends up with blurry results due to the same error factor. On the other hand, our method avoids the color inconsistencies and produces more vivid result than these simple techniques. The remaining results are from various graph generation methods. We can see that k-NN graph with $k = 5$ produces many artifacts for all models due to a large number of isolated subgraphs where data terms are not properly defined, and $k = 20$ removes color inconsistency with reduced blurriness. DT produces quite good result close to our method (1p Z-order and 4p Z-order) because it produces many edges in the graph, although the computational cost is expensive. The 1p Z-order results often contain a small color inconsistency, but 4p Z-order deals well with these color inconsistency. As the result indicates, our method consistently produces visually plausible result with significantly lower computational cost in comparison with k-NN graph with $k = 20$ and DT methods.

4.2. Experiments on Large-scale Data

Our large-scale test dataset is a dense 3D point cloud of an urban scene. It is re-sampled from original 3D points generated from a LiDAR sensor mounted on a vehicle driving in the city. Our dataset contains $215,920,001$ 3D points and 120 of $6M$-pixel images. To remove the erroneous LiDAR firings, *e.g.*, reflections on glasses and no laser return, we applied noise filtering as a pre-processing.

Table 2 shows the comparison of the computation times for various graph construction methods on several chunk sizes. We first compare the computation time of Z-ordering with varying numbers of passes $(1 - 4)$. For each multi-pass Z-ordering, we use an average computation time of all possible passes, *e.g.*, all pairs of $(-x, y, z)$, $(x, -y, z)$, and $(x, y, -z)$ for 2-pass Z-ordering. We also evaluate the computation times of two graph construction methods, k-NN and DT. For the multi-pass Z-ordering, the computation time increases as the number of passes goes up because of the increasing number of edges. In the k-NN graph, we use 20 as k, which produces a similar quality result with our method. The result is consistent with the evaluation using the synthetic dataset; for all chunks, our method produces qualitatively similar result with k-NN with $k = 20$ and DT but with much lower computation time.

To process the whole dataset, we split the original data into 151 chunks. The total computation time is 2.92 hours for colorizing the 215 million points. Figure 5 shows the colorization result of the entire dataset. The result is visually pleasing without blurry boundaries or color inconsistencies. Especially, since our method uses the graph partitioning scheme with overlap for the large-scale data, we can observe that the colorization result does not contain any boundary artifacts between chunks. To show the details of the result, we show close-up views from different view points in the figure.

Table 1: Quantitative evaluation for synthetic datasets. Each element in the center columns consists of AEP and REP, which indicate the colorization error. The last column shows the average computation times.

		Error for noise type			
3D model	Method	Camera + Intensity	Camera	Intensity	Avg time (sec)
Bunny	5-NN	127.02 / 0.66	125.59 / 0.66	126.11 / 0.67	71.72
	20-NN	9.71 / 0.15	8.77 / 0.13	4.24 / **0.02**	822.77
	DT	9.47 / 0.14	8.53 / 0.12	4.21 / **0.02**	2196.51
	1p Z-order	**7.45 / 0.09**	**6.81 / 0.09**	4.32 / **0.02**	**28.45**
	4p Z-order	8.59 / 0.12	7.73 / 0.10	**4.19 / 0.02**	219.81
Dragon	5-NN	124.72 / 0.99	123.02 / 0.99	126.87 / 0.99	57.09
	20-NN	6.17 / **0.06**	7.10 / **0.07**	2.79 / **0.02**	496.28
	DT	5.95 / **0.06**	6.88 / **0.07**	32.13 / 0.49	2061.80
	1p Z-order	6.33 / **0.06**	7.20 / 0.08	3.31 / **0.02**	**20.80**
	4p Z-order	**5.90 / 0.06**	**6.81 / 0.07**	**2.71 / 0.02**	200.28
Armadillo	5-NN	124.24 / 0.94	133.39 / 0.95	126.51 / 0.94	45.06
	20-NN	11.46 / 0.17	12.30 / 0.18	5.18 / 0.04	354.29
	DT	10.99 / 0.15	11.88 / 0.17	4.92 / 0.04	834.25
	1p Z-order	**10.26 / 0.14**	**11.29 / 0.16**	5.25 / 0.04	**18.73**
	4p Z-order	10.43 / **0.14**	11.49 / **0.16**	**4.89 / 0.03**	109.10

Table 2: Comparison of the computation times [second] for various graph construction methods on varying size of data.

ID	# of points	1-p zorder	2-p zorder	3-p zorder	4-p zorder	5-NN	20-NN	DT
1	64993	5.55	5.86	7.61	9.72	5.39	16.55	37.53
2	885888	31.33	41.49	60.24	116.85	13.16	437.59	1158.10
3	976314	32.67	45.83	62.26	123.05	14.14	482.38	1288.08
4	1138103	38.03	53.54	74.09	162.46	14.96	508.84	1180.50
5	1672754	54.01	76.53	122.08	236.36	18.84	627.54	2060.49
6	5980147	222.79	286.62	324.78	922.66	63.48	3019.42	29014.66

5. Conclusions

We have presented an efficient method for colorizing a large-scale 3D point cloud in presence of noisy camera parameters and color inconsistencies across multi-view images. The proposed multi-pass Z-ordering method is shown effective in terms of its computational cost and quality of the colorization. In comparison to the conventional techniques, such as k-NN and DT, it is advantageous particularly in its computational efficiency. We believe that the presented method will be useful for various tasks where we wish to assign vectored values, not necessarily limited to color values but also be other quantities, such as model parameters of a certain reflectance model. Our current method has a limitation in that it produces blurry result for models having a fine and uneven geometric structure and/or high-frequency texture within a small region. In the future, we would like to resolve the blurriness problem for such complex surfaces by simultaneously looking into geometry and appearance cues. We also wish to explore the possibility of extending the method for explicitly accounting for view-dependent appearances.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government(MEST) (No. NRF-2013R1A2A1A01015870).

References

[1] *The Stanford 3D Scanning Repository*. https://graphics.stanford.edu/data/3Dscanrep/. 5

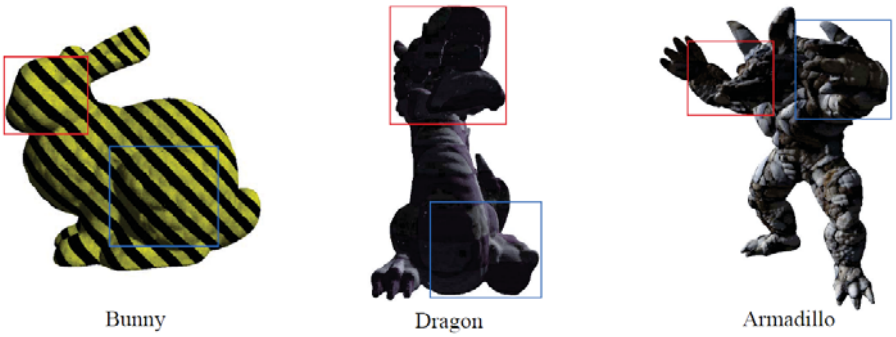
[2] A. K. Aijazi, P. Checchin, and L. Trassoudaine. Segmentation based classification of 3d urban point clouds: a super-voxel based approach with evaluation, 2013. Remote Sensing. 2

[3] K. Andreev and H. Räcke. Balanced graph partitioning, 2004. ACM symposium on Parallelism in algorithms and architectures. 4

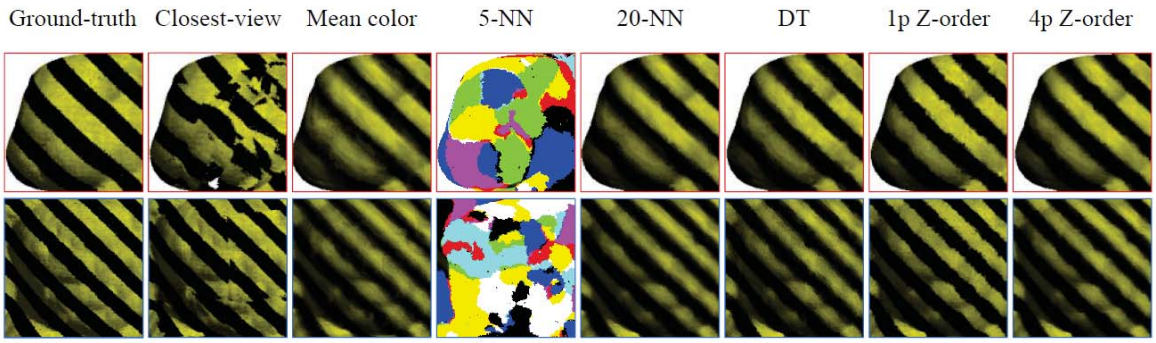
[4] F. Aurenhammer. Voronoi diagrams - a survey of a fundamental geometric data structure, 1991. ACM Computing Surveys. 3

[5] N. Bannai, R. B. Fisher, and A. Agathos. Multiple clor texture map fusion for 3d models, 2007. 2

[6] J. T. Barron and J. Malik. Shape, albedo, and illumination from a single image of an unknown object, 2012. CVPR. 2

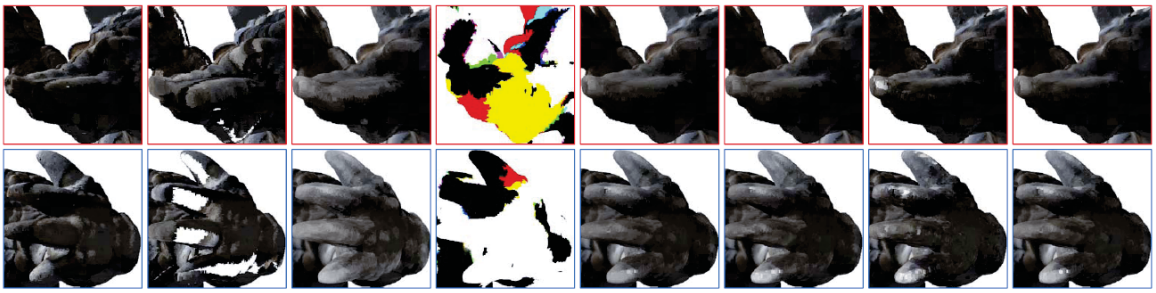


(a) Three synthetic datasets for quantitative evaluation



(b) Colorization results for *Bunny* model under various types of graph generation methods

(c) Colorization results for *Dragon* model under various types of graph generation methods



(d) Colorization results for *Armadillo* model under various types of graph generation methods

Figure 4: Colorization results of synthetic datasets. (a) *Bunny*, *Dragon*, *Armadillo* models. For each model, we magnify parts of colorization result. (b), (c) and (d) are magnified results for *Bunny*, *Dragon*, and *Armadillo*, respectively.

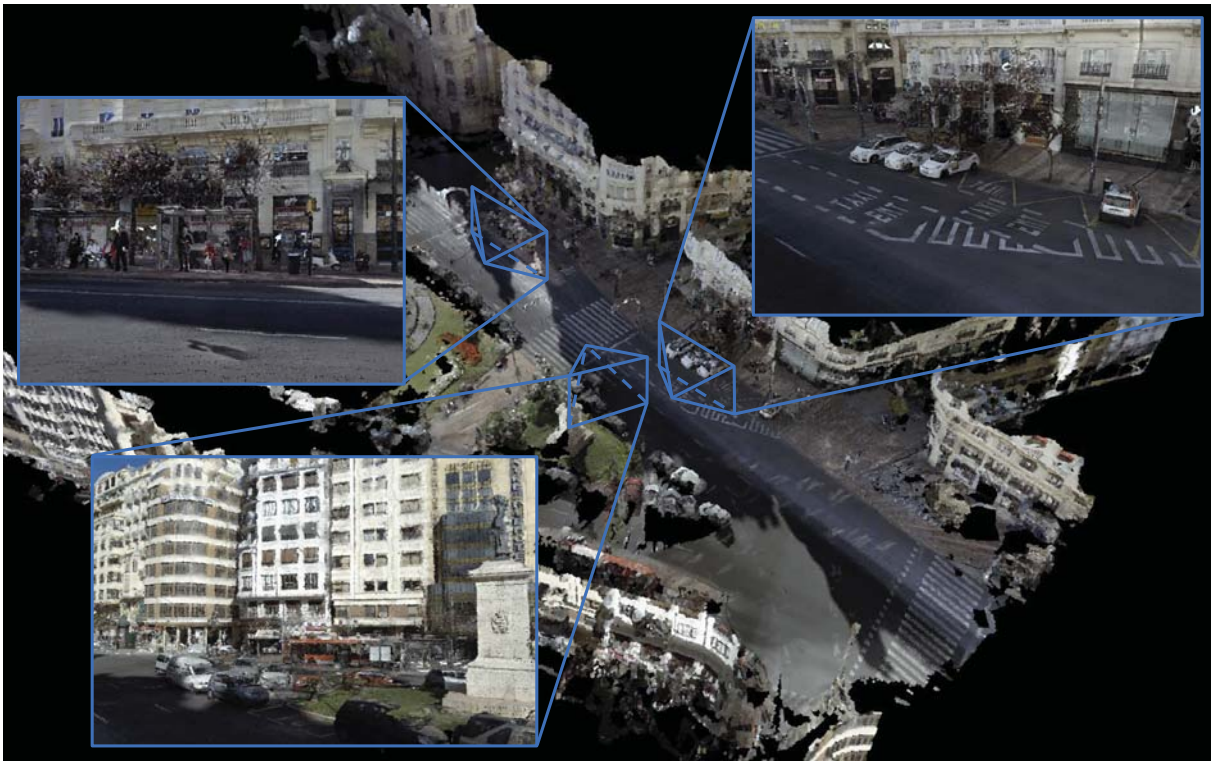


Figure 5: Colorization result of large-scale dataset. We magnify three regions depicted as blue-colored box, and each apex in the polyhedron indicates the view point.

[7] J. T. Barron and J. Malik. Intrinsic scene properties from a single rgb-d image, 2013. CVPR. 2

[8] C. E. Bichot and P. Siarry. Graph partitioning: optimisation and applications, 2011. ISTE-Wiley. 4

[9] M. Chuang, L. Luo, B. J. Brown, S. Rusinkiewicz, and M. Kazhdan. Estimating the laplace-beltrami operator by restricting 3d functions, 2009. Eurographics Symposium on Geometry Processing. 2

[10] M. Connor and P. Kumar. Fast construction of k-nearest neighbor graphs for point clouds, 2009. 3

[11] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. Introduction to algorithms, 2001. MIT Press and McGraw-Hill. 3

[12] T. A. Davis. Algorithm 915, suitesparseqr: Multifrontal multithreaded rank-revealing sparse qr factorization, 2011. 4

[13] J. Eisner. State-of-the-art algorithms for minimum spanning trees: a tutorial discussion, 1997. Manuscript. University of Pennsylvania. 3

[14] R. Finkel and J. L. Bentley. Quad trees: A data structure for retrieval on composite keys, 1974. 3

[15] A. Guttman. R-trees: A dynamic index structure for spatial searching, 1984. SIGMOD. 3

[16] D. Hilbert. Über die stetige abbildung einer linie auf ein flächenstück, 1891. Mathematische Annalen. 3

[17] J. Jia and C. K. Tang. Tensor voting for image correction by global and local intensity alignment, 2005. 2

[18] S. J. Kim and M. Pollefeys. Robust radiometric calibration and vignetting correction, 2008. 2

[19] H. Li, E. Vouga, A. Gudym, L. Luo, J. T. Barron, and G. Gusev. 3d self-portraits, 2013. SIGGRAPH Asia. 2

[20] G. M. Morton. A computer oriented geodetic data base: and a new technique in file sequencing, 1966. Technical Report, Ottawa, Canada: IBM Ltd. 3

[21] F. Pitit, A. C. Kokaram, and R. Dahyot. N-dimensional probability density function transfer and its application to color transfer, 2005. ICCV. 2

[22] S. Rusinkiewicz and M. Levoy. Qsplat: a multiresolution point rendering system for large meshes, 2000. SIGGRAPH. 2

[23] Q. Shan, R. Adams, B. Curless, Y. Furukawa, and S. M. Seitz. The visual turing test for scene reconstruction, 2013. 3DV (International Conference on 3D Vision). 2

[24] M. Stamminger and G. Drettakis. Interactive sampling and rendering for complex and procedural geometry, 2001. Eurographics Workshop on Rendering Techniques. 2

[25] J. Strom, A. Richardson, and E. Olson. Graph-based segmentation for colored 3d laser point clouds, 2010. IROS. 2

[26] Y. W. Tai, J. Jia, and C. K. Tang. Local color transfer via a probabilistic segmentation by expectation-maximization, 2005. CVPR. 2

[27] K. Yamamoto and R. Oi. Color correction for multi-view video using energy minimization of view networks, 2008. 2

[28] M. Zwicker, H. Pfister, J. Baar, and M. Gross. Surface splatting, 2001. SIGGRAPH. 2