# Scalable Surface Reconstruction in the Mobile Edge

Dániel Varga
ELTE Eötvös Loránd University
Budapest, Hungary
vaduaci@inf.elte.hu

Sándor Laki
ELTE Eötvös Loránd University
Budapest, Hungary
lakis@elte.hu

## ABSTRACT

Various augmented and virtual reality (AR/VR) applications have emerged in the past years and their popularity are still increasing. Most of these applications work near real-time, posing low-latency requirements against the algorithms to be used. The high computational complexity of 3D mesh generation from continuous 3D point cloud streams provided by depth cameras requires strong processors that have high energy consumption, leading to low battery life of mobile devices. By moving the processing to mobile edge computing (MEC) servers, the smart phones can be offloaded and thus their battery life can be extended. In this paper, we propose a highly-scalable method for near real-time 3D mesh generation by exploiting the advances of MEC.

## CCS CONCEPTS

• **Computing methodologies** → **Parallel algorithms**;

## KEYWORDS

surface reconstruction, parallel computing, mobile edge cloud

**ACM Reference Format:**
Dániel Varga and Sándor Laki. 2018. Scalable Surface Reconstruction in the Mobile Edge. In *Proceedings of SIGCOMM Posters and Demos '18, August 20–25, 2018, Budapest, Hungary.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3234200.3234243

## 1 INTRODUCTION

Mobile Edge Computing (MEC) aims at providing a tool to transfer computational intensive tasks to MEC servers, offloading mobile phones and thus resulting longer battery time. In the recent years, a few application candidates like real-time video processing and gaming for MEC infrastructures have emerged, but there seems to be a long way to go before some true, market-ready "killer" applications are proposed. The increased popularity of augmented reality (AR) and the advent of cheap depth cameras may result in new use cases where MEC infrastructures can be exploited. AR applications require 3D surfaces about the real environment, however, these surface reconstruction methods generally have high computational complexity.

In this poster paper, we propose a surface reconstruction approach exploiting the benefits of MEC infrastructures that provides computational resources with extremely low latency. Our approach 1) can offload computational tasks from mobile phones, 2) result in the surface in real-time (total delay of processing and data transmission is below the inter-frame time, which is 200ms with our Tango compatible smart phone), 3) scale well with the number of CPU resources. We first present the current status of research and the method we have implemented so far and overview the possible future directions of extension.

## 2 METHOD

Our aim was not to develop a new surface reconstruction method but to make an existing explicit approach scalable and feasible for MEC environment, addressing the previously defined goals. We used an existing triangulation algorithm called Greedy Projection (GP) [1]. An overview of the proposed architecture is depicted in Fig. 1: 1) One or more user equipments (UEs) are located in the same observation area, generating point cloud streams from different viewpoints. 2) They are connected to MEC servers deployed close to the base station they are connected to. 3) The MEC servers processes the point cloud frame in a parallel fashion. 4) The reconstructed and updated mesh is sent back to the UEs.

The processing step running in the MEC servers can be split into the following phases: 1) **Workload splitting:** A continuously updated Octree (search tree) is used to segment the incoming point cloud into different partitions which will be processed in parallel. Each leaf node of the Octree represents a subset of the incoming point cloud. These subsets are then sent to the worker thread responsible for that part of the scene. 2) **Partial mesh generation:** After filtering and smoothing the received subset of the point cloud frame, a partial 3D mesh is calculated by the GP method (see Fig. 2b).

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. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
*SIGCOMM '18, August 20–25, 2018, Budapest, Hungary*
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5915-3/18/08…$15.00
https://doi.org/10.1145/3234200.3234243

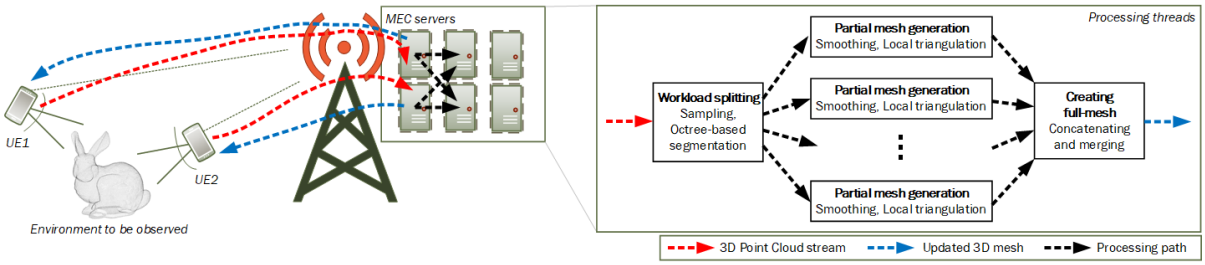
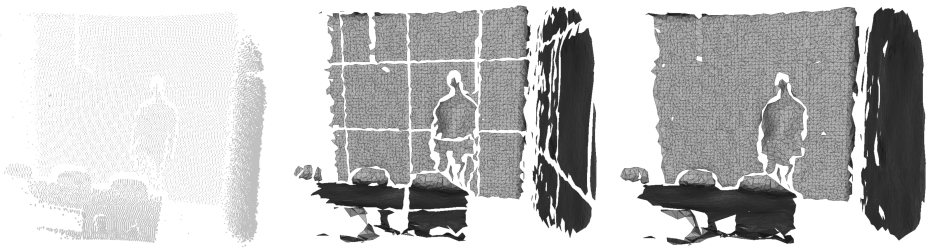



**Figure 1: System architecture**



**(a) A point cloud frame** **(b) Partial meshes** **(c) Mesh after concatenation**

**Figure 2: The captured point cloud, triangulated mesh parts and the mesh with the concatenation**

3) **Creating full-mesh:** A method is applied to merge the generated partial 3D meshes (see Fig. 2c).

In contrast to other approaches [3] our method is not bound to GP and thus independent of the applied triangulation technique.

## 3 RESULTS

We have implemented the proposed method in C++ using the Point Cloud Library (PCL) [2] and evaluated its performance on various point cloud streams collected by a Tango compatible smart phone (Lenovo Phab 2 Pro) with depth sensor. An example point cloud frame can be seen in Fig. 2a. For the tests we used a single user equipment, but the framework may support multiple point cloud sources. The method has been evaluated for two data sets. 1) "room-fixed" case: 370 688 points (28 frames) - the phone position was fixed. 2) "room-dynamic" case: 3 480 640 points (291 frames) - the phone's position and the pose were changed.

The time needed for the surface reconstruction of a single point cloud frame was measured in our testbed (AMD Ryzen Threadripper 1900X CPU (8 core, 16 thread, 3.8GHz). One frame contains about 13 000 points. Fig. 3 shows the average reconstruction time based on 280 iterations, using 1,2 and 4 processing threads. One can observe that the processing time decreases proportionally with the number of threads used.

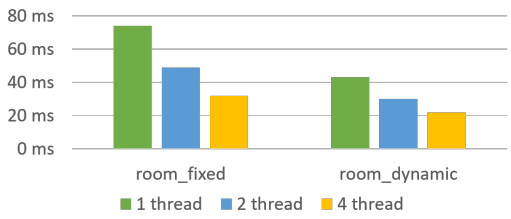



**Figure 3: Average per frame reconstruction time.**

## 4 FUTURE WORK

We see several future directions for extension: 1) Currently the same triangulation method is used for merging, and it is computed in a single thread. A custom method for this step can speed up the triangulation. 2) The resolution of the Octree used for segmentation is hard coded. We plan to extend the solution so that it should adapt for the properties of the input and the computation capacity of the MEC. 3) By moving computations from the phone to the mobile edge we gain lower energy consumption at processing but with the network communication we may lose time and also energy. Therefore, in the future it is also important to consider this factor finding a good trade-off.

**Acknowledgement.** The project was supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00002). The authors also thank the support of Ericsson.

## REFERENCES

[1] Zoltan Csaba Marton, Radu Bogdan Rusu, and Michael Beetz. 2009. On Fast Surface Reconstruction Methods for Large and Noisy Datasets. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Kobe, Japan.

[2] Radu Bogdan Rusu and Steve Cousins. [n. d.]. 3d is here: Point cloud library (pcl. In *In Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 1–4.

[3] N Wongwaen, Sasin Tiendee, and Chanjira Sinthanayothin. 2012. Method of 3D mesh reconstruction from point cloud using elementary vector and geometry analysis. 1 (06 2012).