SUBMISSION DECLARATION PAGE

This page shall be included as the first page of all papers submitted for DTTC Review.

IP Declaration

[✓] My check mark here indicates that to the best of my knowledge the contents of this submission are not classified as Intel Top Secret or Restricted Secret, or otherwise embargoed for publication.

Check one of the following two.

[✓] Consider my submission for Presentation at DTTC and publication on the DTTC web

[ ] Consider my submission only for publication on the DTTC Web

Has your submission or parts of its contents been presented or published previously at another open attendance conference (external or internal)? [No]

If yes, please review the DTTC Previously Published Work Policy and explain why your submission should be considered for DTTC.

Check the one that best applies to your submission

- [ ] Illustration of established Intel/Industry BKM with results and learning for the benefit of other projects
- [ ] Advancement of Intel/industry BKM with new solutions to known problems and/or improvement in the costs, performance, schedule, effort, customer value.
- [✓] Discussion of new, emerging or future issues and solutions
- [ ] New untried ideas and innovations with promise

Check all that apply to your submission

- [ ] Proposal implemented in hardware/software and results available from actual deployment or product
- [ ] Proposal implemented in hardware/software but results awaited.
- [✓] Proposal is a POR and in various stages of Execution in a product/project
- [ ] Proposal under consideration for POR
- [ ] Proposal is a new idea or innovations for Intel products to consider
- [ ] PoC established by prototype, simulation, or by other methods
- [ ] Proposal in need of product sponsor

Contents of my submission are based on (product(s)
KabyLake_____

Contents of my submission should be of interest to (product(s)
TigerLake_____

Analysis of Large Scale 3D Reconstruction and Its Implications to VR/MR Platform Design

Muley, Makarand R, UMPE5-A7-SRR2 (makarand.r.muley@intel.com)

Nadiger, Chetan Hemanth, UMPE5-A13-SRR2 (chetan.hemanth.nadiger@intel.com)

Krishnaswamy, Prasanna, SRR1 (Prasanna.krishnaswamy@intel.com)

Hong, Dongho, JF3-2-30-76, 2111 (dongho.hong@intel.com)

Desai, Akshata A (akshata.arvind.desai@intel.com)

Abstract

3D reconstruction, needed for VR/MR, is a process of capturing shape and appearance of real objects. Reconstruction activity includes tracking a user in a scene so that part of scene visible to the user in real world can be reconstructed and rendered in virtual world. This needs heavy computation and large memory. These storage and memory requirements have been quantified for various scenes which has led a pathway for developing architectural design to meet these compute and memory requirements on Intel's future client platforms. It would efficiently split reconstruction process across CPU, Intel graphics and VPU. At the time of writing this paper, very few algorithms could do real time scene reconstruction at room scale. This paper proposes an efficient IP which when coupled with to-be proposed architecture provide good reconstruction ability on client platform. This paper discusses results of various experiments and their impact on architectural design.

Keywords- Reconstruction, VR, MR, InfiniTAM, Ray cast, Tracking, Fusion, Client, CCG, Head Mounted Display

Introduction

3D reconstruction is a process of capturing shape and appearance of real objects. In recent times, there is a demand for 3D content for Virtual Reality / Mixed Reality (VR/MR) mainstream needs. Supporting 3D reconstruction on Intel client platforms will help in enabling intriguing VR/MR experiences for the end users. This will be an important value add to Intel platforms.

Efficient localization as well as mapping is required for solving 3D reconstruction problem. Localization is a process of finding the position/orientation of camera with respect to its surroundings. Mapping is a process of estimating the scene. Based on location of user, view of reconstructed world will be rendered. If location changes, surface visible to user also changes. Most of the modern day 3D reconstruction algorithms perform at 'small' scale. Additionally, the frame rate of reconstruction is not high enough to consider them for real time operations. Storing and processing information of large volume of space, needs high storage and compute capabilities. Thus, performing 3D reconstruction at room-scale in real time becomes a challenge. The base data required for 3D reconstruction is the object's distance from sensor. Various methods have been evolved based on the method of obtaining depth information. Among them, camera based depth sensors are affordable. Using these depth sensors, one can develop a 3D reconstruction framework to scan the geometry of the environment. InfiniTAM [2] is one such framework developed at the University of Oxford. It uses innovative voxel hashing techniques for solving storage problem. The compute problem is solved by using external GPU. The authors in this paper have analyzed the InfiniTAM framework for suggesting platform architecture to perform real-time reconstruction.

At present InfiniTAM can be built and executed either on CPU or on GPU. In CPU version, 3D reconstruction is slow. For GPU version, external graphics card is needed. The aim of this work is to design platform architecture for providing large scale real time 3D reconstruction on Intel client platforms, by using combination of CPU, internal GPU and, if needed, VPU (Vision Processing Unit). This paper attempts to provide data points needed to develop such architecture. Experiments have been performed as part of this

work to measure compute and storage requirement of 3D reconstruction. Results of these experiments and their implications on architecture design are discussed in this paper.

The authors' analysis draws conclusions that 3D reconstruction is a compute and storage problem. For sufficiently large area, more than 2TFLOPS compute is needed. Also, quality and processing time needed for 3D reconstruction depends on voxel size, a unit of 3D volume representation. Small voxel size results in dense reconstruction and will need more processing time. Based on use case, voxel size should be altered. For use cases like obstacle detection large voxel size can be used. For games where minute details are of importance, small voxel size can be used. By utilizing CPU, internal GPU and VPU together, client platforms will be able to perform 3D reconstruction.

The paper is organized as follows. Section 1 describes working of ORB-SLAM2, another candidate for reconstruction. Section 2 describes working of InfiniTAM framework. Section 3 describes tools and hardware used for analysis of InfiniTAM. Section 4 discusses the results and insights into the performance numbers of InfiniTAM. Section 5 provides recommendations for platform architecture. Section 6 concludes the paper with summary.

Initially, two 3D reconstruction algorithms – ORB-SLAM2 and InfiniTAM were considered and evaluated for room-scale reconstruction.

1) ORB-SLAM2

ORB-SLAM2 [3] algorithm works by tracking sparse ORB features and obtaining the 6DOF (Degree Of Freedom) parameters of camera. The map generated is also sparse. Though it has other features such as loop closure, this algorithm was not taken forward for this work. This work needs dense maps which are not achievable using ORB-SLAM2.

2) InfiniTAM

InfiniTAM is a framework which 'houses' the Kinect-fusion 3D reconstruction algorithm [4]. It has modular software components, which makes it easy plug-and-play of different sub-systems. At fixed voxel size, for large reconstruction area, large storage is needed. Due to scarcity of memory, most algorithms fail to do large scale reconstruction. To handle this, InfiniTAM uses hashing techniques. The depth images are captured using any depth sensor. In this work, Kinect V2 is used. Figure 1 explains the basic working of InfiniTAM.

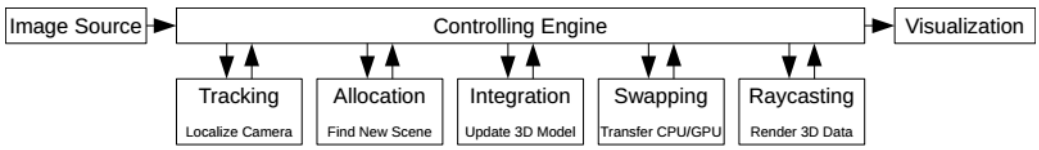

Figure 1. InfiniTAM processing pipeline

Tracking: The camera pose for the new frame is obtained by fitting the current depth (or optionally color) image to the projection of the world model from the previous frame.

Allocation: Based on the depth image, new voxel blocks are allocated as required and a list of all visible voxel blocks is created.

Integration: The current depth and color frames are integrated within the map.

Swapping In and Out: If required, map data is swapped in from host memory to device memory and merged with the present data. Parts of the map data that are not required are swapped out from device memory to host memory.

Raycasting: The world model is rendered from the current pose – (i) for visualization purposes and (ii) to be used by the tracking stage at next frame.

3) Analysis tools and platform

This section details about the analysis software tools and Intel platforms on which the InfiniTAM was tested.

Platform: All experiments were run on the following systems:

a) SKL H (4+ 2E), NVidia GTX 970M, Ubuntu, 16GB RAM

b) Asus I7 Laptop, NVidia GTX 1070M, Windows 10, 16GB RAM

c) KBL-U2+3e, Windows 10

Tools: Vtune for Intel CPU related analysis, NVidia NSight for GPU related analysis, SoCWatch, ETL traces

Datasets: All experiments used TUM Dataset [1]

4) Results and Observations

This section discusses various performance metrics obtained during the experiments.

a) Frames per second (FPS) analysis

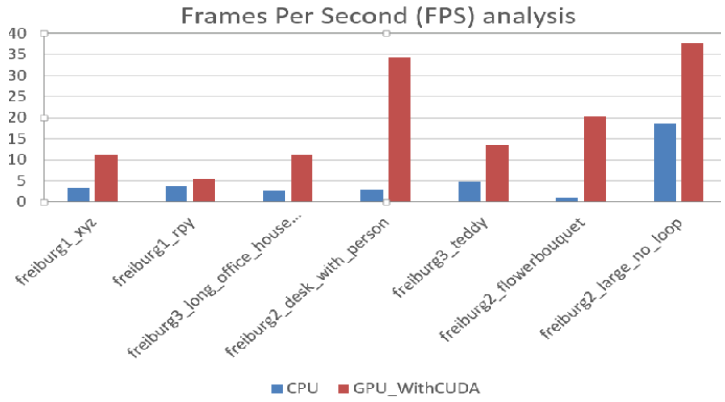

Figure 2. FPS Analysis for TUM Datasets (CPU vs GPU with CUDA optimization)

InfiniTAM has GPU and CPU version. GPU version runs on NVidia Graphics card and uses CUDA optimization library. It was run on multiple datasets with varying level of scene complexity to check if GPU consistently outperforms CPU. GPU always gives 4 to 7 times better Frames Per Second (FPS). So, the FPS range in which the to-be proposed architecture, which is a combination of CPU, internal GPU and, if needed, VPU, can perform lies somewhere in the middle of FPS range for only CPU and only GPU version. Also, more FPS achieved by GPU version on every dataset indicates that huge parallelization opportunities are present in scene reconstruction activity.

b) Identification of key parallelization candidates

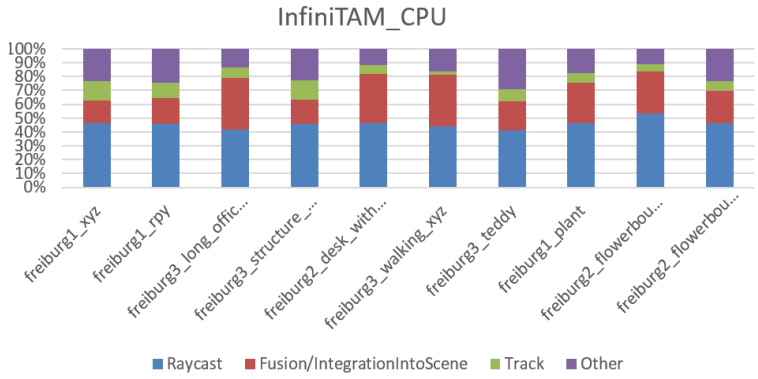

Figure 3. InfiniTAM Key bottleneck analysis (TUM Datasets).

Identifying Key bottleneck functions which are seen on CPU but not on GPU would lead a path towards parallelization efforts required for the to-be proposed architecture. In CPU version, Raycast, Fusion and Track modules take significant time (>70%). In GPU version, CUDA parallelizes such functions. This implies that these bottlenecks should be targeted as key parallelization candidates in the to-be proposed architecture.

c) Impact analysis of multithreading

By default, CPU version of InfiniTAM runs single threaded. As seen in section 4a) InfiniTAM has high scope for parallelization. Utilization of all available cores is required. As InfiniTAM supports Open MP, performance of its Open MP enabled CPU version was tested against its Open MP disabled (single threaded) CPU version and results are shown in Table 1.

Dataset	Number of Images	System DDR BW (GB/sec)		Avg CPU residency				Avg time/frame(mSec)		CPI		P_CPU	
				Core0		Core1							
		CPU	CPU+OpenMP	CPU	CPU + OpenMP	CPU	CPU + OpenMP	CPU	CPU+OpenMP	CPU	CPU + OpenMP	CPU	CPU + OpenMP
Table with Pers	798	0.706	0.872	41.69	98.05	59.34	99.4	423.5	243	0.499	0.87	5.96	10.16
Box Structure	1099	0.704	0.889	47.51	98.13	53.95	99.39	423.1	240	0.485	0.843	5.94	10.22
Vase with plant	1141	0.672	1.028	57.19	99.21	43.82	99.72	686.2	384	0.5	0.857	5.93	10.02

Table 1. InfiniTAM on CPU without Open MP and with Open MP, executed on KBL U2+3e

KBL U2+3e setup gives approx. 100% performance gain (compute time reduction) on Open MP enabled CPU version of InfiniTAM.

Compute Possibilities

d) Estimation of average compute requirement

For this Paper compute requirements for 3D reconstruction were required to be investigated. FLOPS analysis performed on GPU version of InfiniTAM found that for sufficiently large sequences more than 2TFLOPS were needed to perform dense reconstruction. Compute needs for key bottleneck components are described in Table 2.

Function	Module	Average TFLOPS
IntegrateIntoScene	Fusion	1
BuildHashAllocAndVisibleType	Fusion	0.6
PrjoectAndSplitBlocks_Device	Raycast	0.25
GenericRaycast_Device	Raycast	0.08
DepthTracker	Tracking	0.04
FilterSubSampleHoles_Device	Tracking	0.04

Table 2. FLOPS calculation for GPU version of InfiniTAM

e) Compute optimization possibilities

While performing dense reconstruction, each frame brings in new information about scene. At the time of writing this paper, all frames are being processed in InfiniTAM. So, the performance measure (quality, compute and memory requirement) of InfiniTAM with less frames was required.

Experiments performed varied from providing only key frames (identified by running ORB SLAM2 on same dataset) to skipping every other frame. This leads to an observation that processing frames with significantly less new information is not necessary. Few non-key frames can be skipped and yet similar performance can be obtained. This reduces CPU time proportionally to number of frames pruned and mesh size by factor of 8% to 10%.

Storage Possibilities

f) Voxel size influence on 3D reconstruction

InfiniTAM was executed for various voxel sizes and its impact on quality and storage was analyzed. Consequently, voxel size of 5mm is able to provide dense reconstruction with optimal mesh size. Large voxel size reduces storage and processing time at the expense of mesh quality. At voxel size lesser than 5mm, no significant improvement in reconstruction quality is observed but storage size rapidly increases. Based on use case, voxel size should be set.

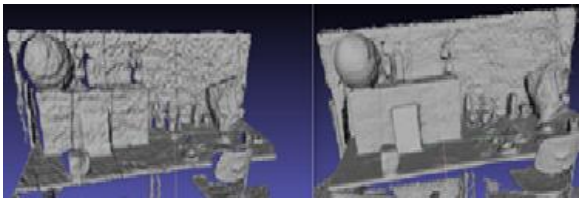

Figure 5. Impact of voxel size on reconstruction quality (7mm vs 5mm)

5) Architecture Recommendation

Platform Recommendation

To get performance equivalent of discrete GPU on client platform with internal graphics, VPU will be needed. The paper recommends to split task among CPU, GPU and VPU. Tasks which need high computation, memory but not parallelization can be run on VPU. Movidius VPU can fulfil such needs.

From per frame analysis, this paper identified key functions which need high storage and can support parallelization. This paper recommends to add more memory to graphics card and allocate more percentage of Execution Units (EU) for reconstruction. Providing sufficient large graphics memory will save time spent on swapping and will significantly improve performance.

Software Recommendation

Reconstruction is sensitive to number of voxels. Small voxel size means more number of voxels for storage and processing. This paper recommends usage of variable size of voxel based on object geometry. For instance, in Figure 6, by identifying planes in a scene, storing one plane will require one voxel because the entire plane has same voxel information. To enable this, appropriate data structure needs to be used to store plane's region.

Figure 6. Plane detection in scene for storage optimization. Left: Scene. Right: Planes in unique colors

Size of reconstructed world is in order of Gigabytes. So, optimization is required to reduce 3D reconstructed mesh size. Pruning redundant frames will save compute and storage and provide better FPS.

6. Summary

For VR/MR, 3D reconstruction is an important capability. Present day solutions for 3D reconstruction are GPU bound and can reconstruct small area. CCG wants to investigate whether this compute problem can be solved using client platforms. The experiments conducted here, conclude that it is possible to give comparable performance on client platforms by carefully splitting reconstruction workload across CPU, internal GPU and VPU where CPU can perform compute intensive tasks, VPU with enough availability of memory can execute activities which need storage and compute but less parallelization and internal GPU can be used for parallelization activities and rendering.

7. Acknowledgments

Neb A; Pillai, Girin G; V, Sukruth H; Varadarajan, Ramesh; Anuraj Sen; Sankhagowit, Peter.

8. Bibliography/References

[1] https://vision.in.tum.de/data/datasets/rgbd-dataset/download#

[2] Prisacariu, Victor Adrian; Kähler, Olaf; Cheng, Ming Ming; Yuheng Ren, Carl; Valentin, Julien; Torr, Philip H. S.; Reid, Ian D.; Murray, David W.,"A Framework for the Volumetric Integration of Depth Images",eprint arXiv:1410.0925,10/2014

[3] Raul Mur-Artal, Juan Tardos D, "ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras", https://github.com/raulmur/ORB_SLAM2

[4]https://www.microsoft.com/en-us/research/publication/kinectfusion-real-time-dense-surface-mapping-and-tracking/