Implementation of Game Integration Pipeline for Rapid Scene Reconstruction using KinectFusion

Alex Rohregger, Gleb Gorbachev, Ivan Hernandez, Farid Yagubbayli

1 Introduction

The availability of low cost commodity RGB-D sensors, such as the Microsoft Kinect, has opened up many opportunities for 3D reconstruction in practical applications. The aim of this project is to implement a pipeline that allows rapid scene reconstruction using the Microsoft Kinect and integration of that scene into a working gaming environment created in Unity3D. The pipeline is tested using a custom driving video game for the following parameters:

1. Runtime performance of handcrafted and scanned environment.
2. Time taken to create the scenes, with focus on time spent by the user.

2 Rapid Scene Reconstruction Pipeline

The rapid scene reconstruction pipeline is made up of two sub-pipelines as shown in figure 1. The KinectFusion sub-pipeline is implemented in C++ and runs on the CPU. The Unity3D importer pipeline is implemented in Unity3D running a special version of C#. The framegrabbing from the Kinect is achieved using the NUI-framegrabber package supplied with the Windows Kinect SDK.

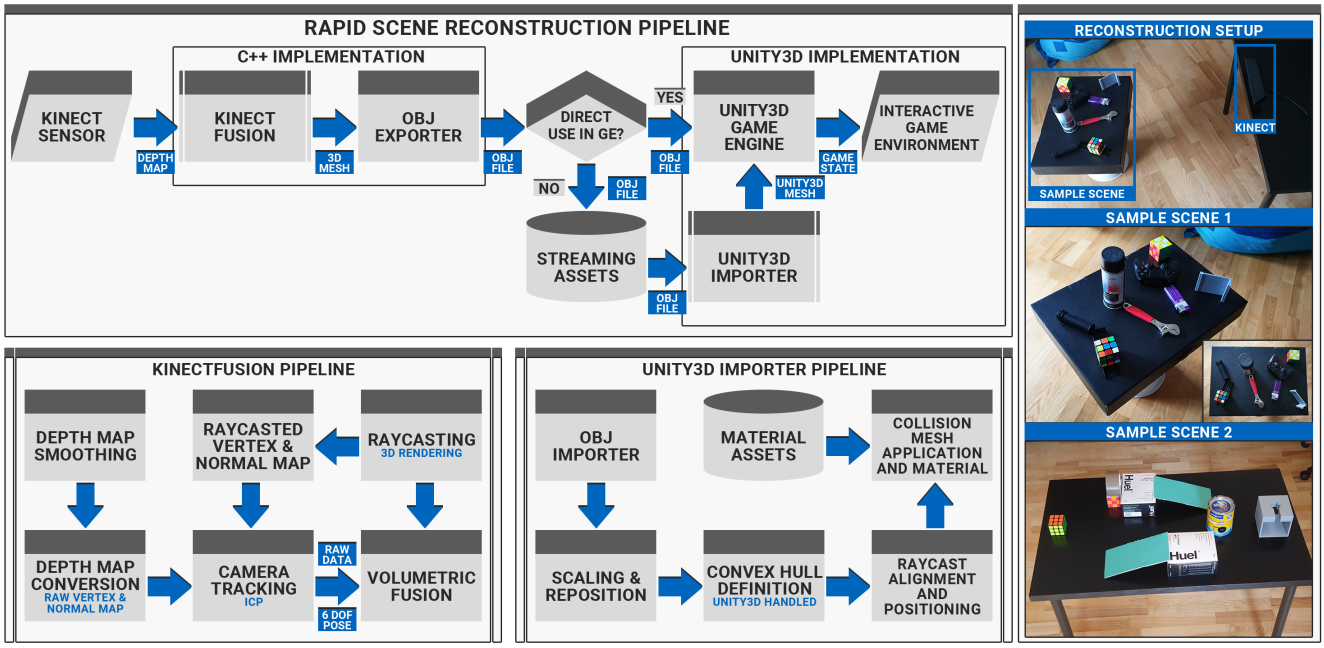

Figure 1: (Left) Game integration pipeline with the sub-pipelines KinectFusion and Unity3D OBJ-Importer. (Right) Scanning setup and sample scenes constructed with basic objects.

Two simple sample scenes were created consisting of both basic and complex shapes. The performance benchmarks were performed using a Nvidia GTX 1080Ti and an AMD Ryzen 1700X.

3 Results of Reconstructed and Handcrafted Sample Scenes

The results in figure 2 indicate a clear difference in quality between the 3D reconstructed and handcrafted sample scenes. However, table 1 shows that the time the user spends generating each scene is considerably less for the 3D reconstructed scenes, as much of the work load is carried out by the pipeline.

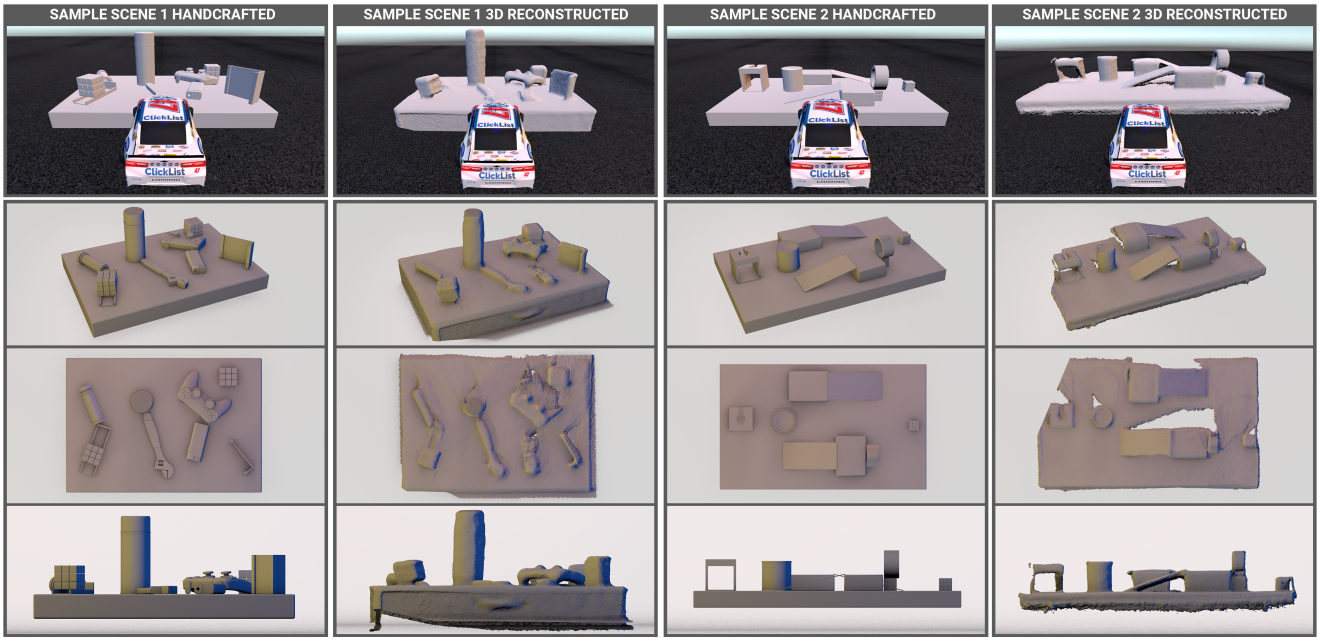

Figure 2: Two sample scenes used for testing, showing side-by-side visual comparison between the handcrafted and reconstructed scenes.

Table 1: Benchmarking results for various test scenes. (H) indicates handcrafted version of the scene, whilst (K) indicates 3D reconstructed scene using the KinectFusion pipeline.

Scenes	Performance [Frames/s]			Throughput Time [min]		Vertices	Faces
	Min	Max	Avg	Total	User Time		
Baseline (Plane)	36	120	108	NA	NA	4	2
Sample Scene 1 (H)	51	122	107	144	144	38092	38296
Sample Scene 1 (K)	43	121	104	30	10	759114	253038
Sample Scene 2 (H)	56	123	105	42	42	2206	2230
Sample Scene 2 (K)	53	122	105	30	10	97336	194170

4 Conclusion and Outlook

The results outlined in 1 indicate a great potential for rapidly reconstructing complex scenes, as the processing time only scales linearly with the time scanned, regardless of scene complexity. Handcrafted scenes can take a lot more time, especially for complex scenes. However, the trade-off lies in worse quality with reference to the ground truth and more vertices and triangles, although this does not have any significant impact on performance given that modern graphics hardware can handle millions of triangles. Considering the results from our testing, the use case for our current pipeline lies more in giving users the ability to create custom scenes for games without requiring any modelling knowledge. Several of the above outlined deficits of the game integration pipeline, compared to handcrafting game scenes, can be resolved in future iterations by exploring improvements outlined below:

1. Throughput Time: GPU implementation of the KinectFusion pipeline and OBJ-Importer pipeline to achieve real-time performance.

2. Functionality Extension: Increase scanning range with an extended KinectFusion pipeline and a loop-closure algorithm to reduce distortions caused by drift error.

3. Quality: Increasing the resolution of the voxel grid to capture finer details of the scanned scene and using RGB input data to texture the scanned scenes.