Real-time Collection and Analysis of 3-Kinect v2 Skeleton Data in a Single Application

Serguei A. Mokhov,* Miao Song,† Jonathan Llewellyn, Jie Zhang, Alexander Charette, Ruofan Wu
mDreams Pictures, Inc.
Shuiying Ge‡
RCSC, CAS, Beijing, China

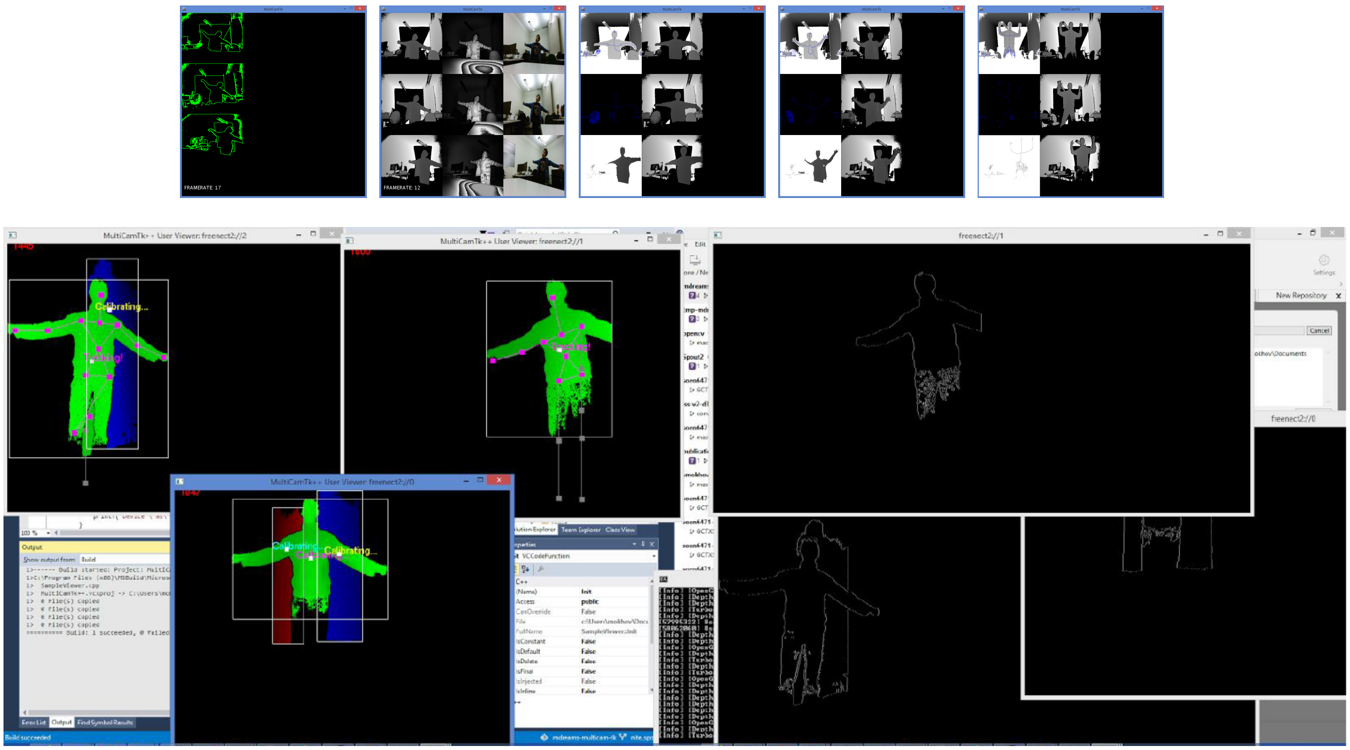

Figure 1: *Sample Multiple Kinect v2 Rendering from MultiCamTk (top) and MultiCamTk++ (bottom)*

Abstract. It was not possible to do reliable 3D skeletal tracking with the currently publicly available inexpensive consumer grade hardware/software tools, such as depth cameras and their SDKs using multiple of such sensors in a single application (e.g., a game, motion recording for animation, or 3D scanning).

We successfully attached 3 Kinect v2 sensors to a single application to track skeletal data without using Microsoft's Kinect 2 SDK. We created a new toolkit – MultiCamTk++ for 3 or more Kinects v2 with skeleton support in C++. It is a successor of our previous version, MultiCamTk, done in Processing/Java that had no skeletal tracking. We achieve high resiliency and good frame rate even if 1-2 Kinects are disconnected at runtime. We are able to receive the skeleton data from the multiple sources to correlate the coordinates for spatial 3D user tracking.

*Contact: mokhov@encs.concordia.ca
†contact@mdreamspictures.com
‡shuiying.ge@rcsc.cas.cn

SIGGRAPH '16, July 24-28, 2016, Anaheim, CA,
ISBN: 978-1-4503-4371-8/16/07
DOI: http://dx.doi.org/10.1145/2945078.2945131

Keywords: motion capture, real-time, 3D sekeletal tracking, OpenGL, OpenCV, frameworks, multi Kinect v2, libfreenect2

Concepts:•Human-centered computing → Graphics input devices; •Computing methodologies → Motion capture; Tracking;

1 Toolkit (Tk) Solution Overview

The primary motivation for this work is to get reliable skeletal tracking of the user using 2-3 Kinect v2 sensor while playing motion-based games.

Tk's content includes a number C++ components of sample functionality to support multiple Kinect v2 in C++ and VS2015 with skeleton tracking, contours, floor detection, and a decent framerate—a subject to hardware and software optimization. The toolkit has configurable options to enable and disable certain functionality.

Microsoft Kinect v2 SDK currently does not support natively multiple Kinect v2. The solution thus has to reply on a version of `libfreenect2` [Xiang et al. 2016] (natively) – an open-source C++ driver for depth cameras that does support multiple Kinect v2 sensors. However, natively `libfreenect2` does not support skeleton tracking despite a recent driver patch added to enable an OpenNI2 driver within `libfreenect2`. With the later driver added, however, one can use NiTE samples with `libfreenect2` as a driver since PrimeSense was acquired by Apple. The issue with NiTE-to-libfreenect2 solution is it was impossible with the current versions to instantiate multiple devices. Therefore a multiprocess solution was created with interprocess communication using optionally OSC (to gather skeleton coordinates) and Spout2 (to gather frames). This solution allows simultaneous tracking of skeletons from 3 Kinects and gather the coordinates in the single process for analysis. The same may be enabled for depth and other frames. The multiprocess solution provides good isolation in-memory between devices and more reliability – e.g., if one or two processes are knocked out, the others continue to operate. It has support options for optimization using threads and CUDA (currently CUDA is not tested).

This solution can be made portable in full or in part to OS X and Linux due its use of `libfreenect2`, OpenNI2, OpenCV, tinythread, and oscpack. Spout2 can be replaced with Syphon. The solution will need to be adapted to `cmake` and make from Visual Studio 2015.

A deployment schematic of the components and their interaction are depicted further.

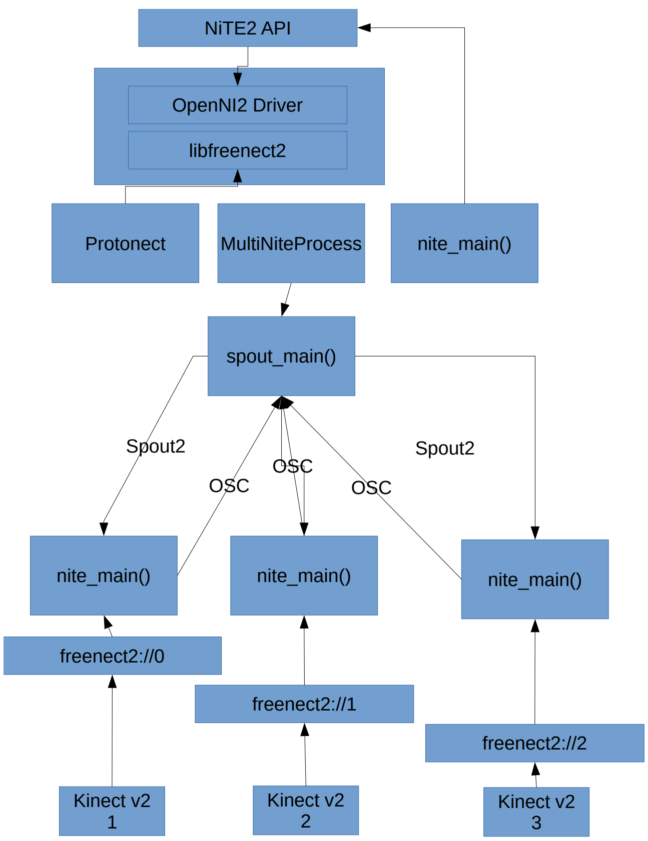

Figure 2: *Deployment Layout of Multiprocess Solution*

Prerequisites. For convenience these prerequisites are pre-packaged and provided along with MultiCamTk++, but can be downloaded and installed separately individually on a clean system.

- `libfreenect2` with OpenNI2
- OpenCV with OPENNI support option enabled
- GLFW (comes with scripts to be pre-installed in `libfreenect2`)
- NiTE support and its dependencies (PrimeSense samples and OpenNI2)
- `libusb` (used by `libfreenect2`) with libusbK backend
- oscpack – OSC support for IPC
- Spout2 – optionally enabled to send frames. If only skeleton frames are needed, then simply just use OSC. (Syphon can be used on OS X instead Spout to achieve the same framesharing effect.)
- `tinythread` – included as a part of `libfreenect2`, we use it for threaded receiving of OSC skeleton data and

2 Summary

We use C++ with various OSS platforms and our own development 3 skeleton tracking works from 3 Kinects in a multiprocess solution relying on NiTE with libfreenect2 driver with OpenNI2 built. The skeleton data are gathered in one place in the parent process MultiNiteProcess via OSC from all three Kinects. Single process solution for 3 Kinects only works with Protonect without skeletons or a single Kinect in NiTE mode. Contours work in both. Floor detection only works in NiTE mode. CUDA/CL optimizations currently work only in Protonect mode.

Points of note.

- Not desirable to render both viewer types at the same time – slow.
- Using NiTE API from OpenNI2 has a side benefit that NiTE API is support such as floor detection is included in it by default.
- Relying on libfreenect2 allows support for other sensors, like Structure and many graphics platforms.
- Of course, better hardware, such as USB3 expansion cards, graphics cards and memory are recommended to get higher performance rates.
- Debug and instrumentation output has been disabled in the code provided since outputting large amount of data into the terminal slows the whole process down and fills up the buffer memory of the terminal, so enable only when needed.
- Tinythread to receive spout and OSC messages in parallel
- If ported to OS X, Spout would be replaced by Syphon.

Optimization. We have moved from the the default OpenGL and CPU pipelines to OpenCL, which improves depth processing needed for skeletons.

- CUDA can speed things up, but need VS2013
- Direct X with Spout2 can be optimized to send textures faster
- Remove label strings from OSC messages to speed deliver and consider using 3 tiny threads, one per device and collect results via /device1, /device2, /device3 filters and aggregate in 3 dimensional arrays in MultiNiteProcess
- Debug flag variable if enabled slows down things considerably. Better to use with a single device when testing or two devices – can stall 3rd device if too many file descriptors are open or USB3 PCIe card is not stable – set the environment in main to reduce transfer rates

Limitations.

- CUDA currently would require VS2013; our solution uses VS2015, but can be adapted to VS2013.
- Some USB3 controllers may need tuning USB transfer rates per the troubleshooting page.

Future Work. To improve performance and handling can consider doing the following:

- CUDA integration to speed up
- Split OSC sending per child processes per device

In 2007, Nvidia introduced CUDA, an API that gives developers direct access to GPU's virtual instruction set and parallel computational elements. Graphical processing can be accomplished on the CPU, but it will be much slower than if the same task is processed on the GPU. Utilizing CUDA, access to this GPU's special architecture will be enabled. Thus, this will significantly improve the performance. `libfreenect2` supports CUDA already, with our build environment we need to adjust VS2015 to take advantage of that, which is not officially supported yet.

References

XIANG, L., ECHTLER, F., KERL, C., BLAKE, J., ET AL., 2016. libfreenect2: Release 0.2, Apr. https://github.com/OpenKinect/libfreenect2.