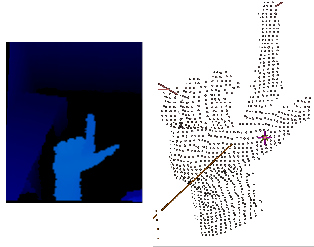
Skeletal Hand Tracking

Getting Started Guide

The library provided in this package has technology designed to track the 3D pose of the user’s hand based on depth data - such as what is provided by the Creative Interactive Gesture CameraTM through the Intel® Perceptual Computing SDK.

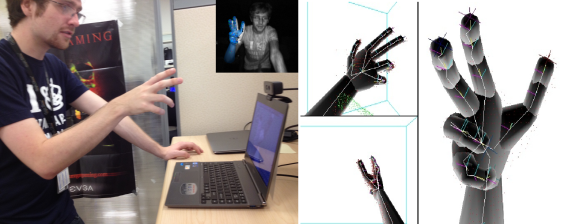
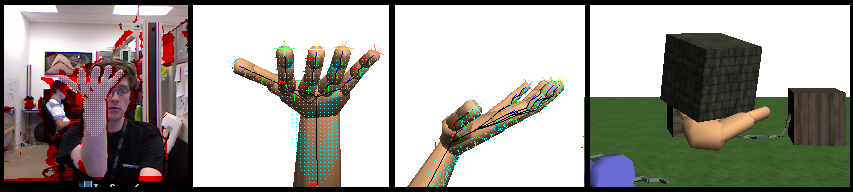
## Contents

This .zip package includes hand tracking library header **.h**, **.lib**, and **.dll** files found in the **include/**, **lib/**, and **bin/** subfolders respectively. Additionally there are some sample programs illustrating how to use the tracking in the **samples/** subfolder. With a Creative Gesture Camera plugged in and the Intel® Perceptual Computing SDK installed on the system, it should be possible to go into the bin folder and run the pre-built samples.

## Overview

Figure 1 Screen shot of handtrackviewer.exe

Running the **HandTrackViewer** sample illustrates what the software does. Raise a hand with fingers spread in front of the camera about 60 cm away and move it slowly toward the camera. You should see a rendered hand model overlaid on top of the user’s hand shown in the video stream. Try moving the hand and fingers around. If the system is working well, the rendered hand model’s pose will match that of the user. In a nutshell, the pose of this displayed articulated hand is the output of the skeletal hand tracking library. The pose is the 3D position (xyz) and orientation (xyzw) of all the bones of the hand. Consequently, the hand can be rendered from different viewpoints and used in a virtual environment for true 3D interaction.

**** 

## Tracking Behavior and Troubleshooting

The output of tracking library will not always match the pose of the user’s hand. Rest assured there are ongoing efforts to improve the robustness and performance of the software. In the meantime there are some usage tips that can help improve the reliability.

The tracking algorithm tries to fit a hand model to the closest object in its view. If there are things contributing to the depth image other than the forearm and hand then this can throw things off. This could be anything such as baggy sleeves, chair armrests, coffee mugs on the desk – anything in the field of view and close to the sensor.

Coffee Mug Interference

The internal tracking technology is based on fitting a 3D “tracking” model to the depth data (or point cloud). Consequently, the more accurately this internal geometric model matches the object (user’s hand) being tracked the more robust the tracking will be. The internal hand model is based on a human hand of average size and shape. To improve tracking accuracy for an individual user, this model should be scaled to fit the user’s hand. The API supports length and width scale factors that are applied to all the bones. The **HandTrackViewer** sample accepts **w**/**a**/**s**/**d** key presses to adjust the size on the fly. In this sample, the IR image background should approximately line up with the overlaid rendered model.

Even with a correct fit and no interference, the tracking can get messed up and may not snap back to a correct fit right away. Putting the user’s hand in a spread 5 finger pose with the palm squarely facing the camera is usually a good way for the system to regain tracking. Furthermore, pulling the hand backward to the specified depth limit will put the system into that pose.

You may find hand and individual finger motions track more reliably with the hand a bit closer to the camera (such as at 30 cm) rather than further (say 60cm). This is because there are more depth samples to use for the hand fitting at closer distances – provided the hand remains within the sensor’s view volume.

Spread 5 Reset Pose

Some motions, such as rolling a clenched fist, provide very little distinguishable detail in the depth data, and will be more susceptible to loss of tracking.

The tracking system will likely saturate one of the x86 cores with computation. Frame rate will typically range between 30 and 60 fps depending on the performance of the computer. Some laptops can operate at a battery-saving slower clock speed when unplugged which tends to slow down cycle hungry applications. Plug in the laptop if you notice poor performance.

## Articulated Hand Model Construction

Figure 3 Bone IDs of tracking model.

The tracking hand model has 17 bones and 16 joints. We use one bone for the forearm, one bone for the palm, and three bones for each finger including the thumb. We use one joint at the wrist, to connect the forearm and the palm, and three joints for the knuckles of each finger, to connect the finger bones together and to connect them to the palm. Due to the acyclic nature of the bone connection graph, the model can be expressed as a hierarchical structure with the forearm as the root of the tree. Therefore, for convenience, the position (or local origin) of each bone is the same as the location of where it is attached to its “parent” bone. Because finger joints are linearly constrained, this position will be fairly constant in the parent’s reference frame.

The geometry of the internal hand model (convex mesh for each bone) is accessible (read only) through the API should the developer wish to display or otherwise use it in an application. Alternatively, the pose can be used to drive any model such as a skinned mesh with a compatible rig.

## Typical API usage

Refer to the samples to see how tracking is integrated into an application. In a nutshell, initialize the camera and the tracking at program startup. Do some hand size calibration if necessary. Each frame make the call to Update() to get the current pose of the user’s hand then do something with it.

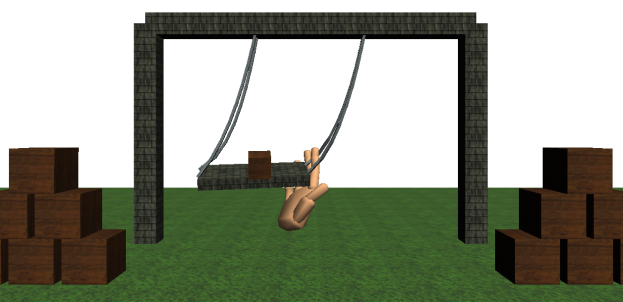
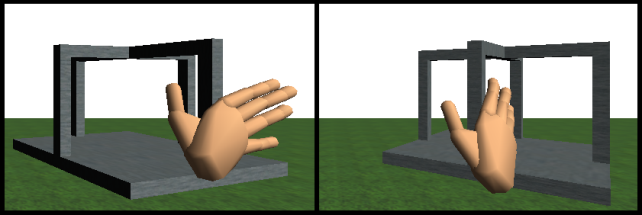
Note that there *is* a temporal element to the tracking, the tracking module should *not* be reinitialized each frame, as the tracking quality will suffer.

## Application Development Suggestions

It helps for the user to get continual feedback about where the system thinks the hand is. Often users will learn to keep their hand within the field of view and eventually prefer to use motions that are more reliable. In addition to rendering a hand model that uses the tracked pose, the API gives access to which pixels (segment) of the video stream are used for tracking and fitting the hand. It can sometimes be helpful to show the video stream with these pixels highlighted so the user can see what the camera is trying to fit.

The application ideally should be geared to make use of tracking rather than something that recognizes gestures. Furthermore, unless the application has a need for interaction within a 3D virtual setting, there is little value added by having skeletal tracking.

Within an application it helps to provide a context that encourages users to keep their hand within the field of view of the depth sensor. One method is to keep the objects of interest are reachable when the hand is in this zone such as what is shown in the snake demo figure to the right. For an environment with more things to explore, by turning the camera when the hand is at the extents of the field of view will place the object of interest that the user is reaching for in a more direct line of site encouraging his hand to keep within range. This technique, in combination with a forward or backward movement when the hand is near or far from the camera, enables a useful way for the user to navigate an environment.

The nature of the interactive objects also plays a role. In a pure physically based virtual scene, convex objects with 6dof can be challenging to grab. A user will have more success attempting to interact with objects that have more complex shape, consist of a number of articulated sections, or have constraints that limit motion or degrees of freedom. Reducing gravity and increasing damping for objects when they are near the virtual hand also helps. One-off custom programming to get the interaction experience desired for specific objects is also a possibility when general physics interaction isn’t sufficient.

An application will have more success if can utilize hand motions that are easier to track versus requiring those that would be more difficult. Experimentation with various techniques can help determine what scenarios are more feasible.

## References

* Intel Perceptual Computing Website <http://www.intel.com/software/perceptual>
* Technical info: Dynamics Based 3D Skeletal Hand Tracking, S Melax, L Keselman, S Orsten. poster/abstract from I3D symposium 2013 <http://dl.acm.org/citation.cfm?id=2448232> and paper in Graphics Interface 2013.