Version 1.0

Joaquim Rocha (jrocha@igalia.com)

Skeltrack is a Free Software library that performs human skeleton tracking from depth images.

Introduction

With the release of Microsoft's Kinect, there was finally a camera capable of giving depth information at an affordable consumer price. This, together with the fact that the device has an open USB connection, opened to doors for many innovative projects from independent developers.

One of the common applications of depth cameras, and the primary purpose of the Kinect, is the usage of its information to detect people and their movements by tracking their skeleton.

Controlling the Kinect's basic features and retrieving its information is already successfully accomplished by the Free Software library libfreenect but this gives only raw information like the depth, color, orientation of the device, etc. so skeleton tracking solutions should then use this information as a base for detecting the human skeleton.

The most well-known solutions for skeleton tracking with the Kinect device are the Microsoft Kinect SDK, the Microsoft Kinect for Windows and the OpenNI framework.
The two Microsoft's solutions differ in the way that the Kinect SDK doesn't allow its usage for commercial work whereas the Kinect for Windows does allow it but will require its users to purchase a differently branded Kinect. Needless to say, Microsoft's solutions, besides being closed, only support the Windows operating system.
Even though the OpenNI framework is multi-platform and allows commercial use, it's skeleton tracking module is also closed source software.

To solve this problem, Igalia has developed Skeltrack, a truly Free Software library capable of tracking the human skeleton from depth images.

Skeltrack's Approach

Skeltrack was not created to be a full-blown SDK for development with the Kinect. In fact, Skeltrack does not even require a Kinect device. It is device agnostic and just needs to be given a buffer containing the depth information.

Apart from device independence, another goal when developing Skeltrack was that it didn't need to use a database of poses with which to compare the ones

being detect. It should rather use mathematics and heuristics to detect and identify the human skeleton's joints.

How it Works

Preparing the buffer

Before the actual algorithm is used, the given depth buffer should be preprocessed in order to contain only the user. This can be accomplished by thresholding the buffer until it only contains the plane where the user or by performing background subtraction.

Skeltrack's algorithm performs computations that might turn out to be heavy depending on the machine and the buffer's size used. To make these easier, the buffer's dimension should be reduced before supplying it. Skeltrack has a property that determines this reduction so it will return the joint's coordinates according to the original buffer's size.

Finding the extremas

Skeltrack is partly based on a research paper by Andreas Baak called “A Data-Driven Approach for Real-Time Full Body Pose Reconstruction from a Depth Camera" but, as mentioned, does not use any database, apart from other changes to the algorithm.

Skeltrack's algorithm starts by finding the extremas in the depth buffer. That is accomplished by creating a graph based on the depth information, that is, each graph's node represents a depth point from the buffer and connects to other nodes (or points) if their euclidean distance is less than a configurable value.

Due to possible noise from the camera and other factors, the graph might end up having more than one component. Since Skeltrack expects to run on a connected graph, if the graph has more than one component, these should be joined by connecting the closest two nodes between each component.

After having the graph ready, it starts calculating the distances from a starting point (node) to every other node using Dijkstra's algorithm. After finding all the distances, it takes the node with the longest distance and creates a 0-cost edge between this node and the starting point. Then it restarts the algorithm using this newly connected node as the starting point.

Skeltrack currently focuses in finding the upper-body extremas so the starting point of the procedure explained above is the lowest point vertically aligned with a centroid point. This will assure that the upper-body extremas are the ones computed. Because it aims for the upper body extremas , the number of extremas that are currently computed is 3: hopefully the head and hands but not always, as it'll be explained further in this paper.

Identifying the joints

Having had computed the 3 extremas, we will use heuristics to determine if one of them should be a head joint. This is accomplished by going over each extrema and looking for the two points below them where the shoulders should be which is determined by checking the euclidean distance between the two points' and the current extrema using some configurable thresholds.

If one of the extremas and respective shoulders' points obey those thresholds, they are considered as successfully identified so the algorithm then focuses on the other extremas.

For the remaining two extremas, it is simply not enough to consider that the one that is on the left is the left hand and the one on the right is the right hand. For example, in case the user has extended the arms and crossed them, the above assumption would be wrong as the right and left extremas would correspond to opposite hands.

To prevent such cases, the way to identify which extrema is right or left used by Skeltrack is to make use of the Dijkstra's algorithm again and calculate the distance from the two extremas to the shoulder nodes identified before. If an extrema is closer to a shoulder's node than the other is, then it belongs to that shoulder and is considered left or right according to it. This should work even when the user's arms are crossed.

As mentioned before, the 3 extremas should represent the head and hands but it doesn't always happen because e.g. if the user has the arms down and close to the torso, the hands will not generate a long distance when calculating the extremas. However, under such conditions and in other cases, it was observed that the extremas would rather match the head and elbows.

To identify whether an extrema is a hand or an elbow, the previously calculated distance between it and its shoulder is used. If this distance is greater than a configurable threshold, it is considered a hand, otherwise, it is considered an elbow.

If an extrema is a hand, the distances previously used can be used again to find that arm's elbow. This is done by going over each node in the list of nodes used to calculate the mentioned distance and considering the first node with a distance less than the one that is defined for the elbows.

At this point, the algorithm finishes and a list holding the detected joints is returned.

Future Work

Skeltrack is a very new library so there is surely improvements that need to be done. In this section, some of those are mentioned.

As mentioned in the explanation of how Skeltrack works, currently it focuses on the upper-body so future work will include the detection of more skeleton joints (feet, hips, etc.).

Although the elbows can successfully be identified after the hands are already identified, the opposite does not work. That is, if the elbows are identified as

extremas from the beginning (when the user as the arms down, for example), Skeltrack currently cannot infer the hands from them. This will also be part of future work.

By working on a buffer with its dimension reduced and due to possible noise generated by the camera used, the joints usually jitter so a way to diminish this will be implemented. This might be accomplished by using the joints' information present at each frame and this information can also be used to improve and ease the detection of the joints.

Another improvement that should be done is the detection of more than one user. To do this, when connecting the graph's components, it should not connect them all but rather only those whose closest points' distance is less than a certain value. This will allow to consider the components as independent graphs, apply the skeleton detection algorithm to them and also discard components that fail to obey certain rules so it can ignore eventual objects that might be recorded in the users' area.

Usage

Skeltrack is written with Glib which is part of the GNOME project. It offers a synchronous and asynchronous API to track the skeleton's joints so it can be used for online or offline skeleton tracking.

The library is shipped with documentation and an example so it should be easy to get started with it.

Joaquim Rocha, Igalia, S.L.

Resources

Skeltrack repository: https://github.com/joaquimrocha/Skeltrack

Skeltrack's documentation: http://people.igalia.com/jrocha/skeltrack/doc/latest/

Video of Skeltrack's example application: https://vimeo.com/38875885

Video of using Skeltrack to control the GNOME desktop: https://vimeo.com/39660879

Libfreenect: http://openkinect.org/wiki/Main_Page

Kinect for Windows: http://www.microsoft.com/en-us/kinectforwindows/

OpenNI: http://www.openni.org/