Adapting Pik Tracker Software for Articulated Tracking

Amber Shinsel, shinsela@eecs.oregonstate.edu

1. Introduction

An area of intense research in computer vision is the analysis of human action. To date this analysis has relied almost entirely upon data collected of a human subject wearing expensive sensors at a sparse set of points. Due to these limitations, human action recognition using this data is extremely suspect. Recently, an inexpensive 3D camera with high fidelity has been developed, and has allowed a skeletonization of a human being (called articulated tracking) to be extracted in realtime using this sensor. Software was recently developed by Soren Hauberg to achieve more accurate articulated tracking. This software is known as the Pik Tracker [1], and has the potential to be very valuable in advancing articulated tracking. The Pik Tracker software will allow us to compare the performance of articulated tracking in estimating human pose to the performance of the invasive, expensive wearable sensors such as motion capture suits. This analysis could provide a strong incentive for all human action recognition research in computer vision to rely on the data collected by our method. However, the Pik Tracker software was not in a usable state for the TeleImmersion lab. My task was to collect the various library and program dependancies, alter the program to compile on a Windows-based system, create and run the executables, and verify that the Jacobian [3] was being correctly calculated.

This paper will be structured in the following way. In section 2, basic background information on articulated tracking will be given. Next, OpenTissue —a program that is greatly depended on by the Pik Traker— will be discussed. In section 4, the main stages of the project will be outlined. Finally, section 5 will discuss the future work that will be done with the Pik Tracker code.

2. Background

Articulated tracking is used in computer vision to try to estimate the position of a body or object over time given certain sensor input. Articulated tracking is just one of many solutions to human motion tracking, but it seems to be one of the most promising ones as well. Motion capture is an example of another of these solutions, but it is invasive, expensive, and unsuitable for most data capturing outside of a laboratory.

Articulated tracking is highly related to particle filtering. Particle filters, otherwise known as Sequential Monte Carlo methods, are model estimation methods based on simulation and probability. SMC methods work to output samples that approximate the filtering distribution [5] . The basic steps of particle filtering include sampling, resampling, computing importance weights, prediction, and normalization are all present in the Pik Tracker code. Essentially, these steps simply ensure that the samples used represent a probability distribution. For more detailed information on particle filtering and Sequential Monte Carlo Methods, see [5].

A very important part of the articulated tracking process is the computation of the Jacobian [txt]. The Jacobian plays a vital role in the accuracy of the articulated tracking, and is very depended on when dealing with the covariance of calculated 'bone' positions [1]. The 'bones' are defined and determined in a skeleton file that is used by the Pik Tracker. The skeleton file is essentially a

spatial representation of the person or object that is the subject of the articulated tracking.

3. OpenTissue

There were many program dependancies for the Pik Tracker, but the most important one was OpenTissue. OpenTissue is a program that consists of a set of many data structures, algorithms, and functions that assist in modeling and simulation. The Pik Tracker used many OpenTissue functions; the one I focused most on being the Jacobian calculations.
OpenTissue requires a set of its own dependancies. They consist of:

- *Boost and the Boost Bindings*
- *Nvidia Cg*
- *Qhull, TetGen, TinyXML, and Triangle*
- *ATLAS*
- *DeVIL*
- *GLEW*
- *GLUT*

These dependancies are very important, and installing them incorrectly can lead to many problems. However, even if you install them correctly OpenTissue will occasionally run into problems with their libraries or include files. This happened to me, and I was able to rectify the problem by manually including the necessary files in Visual Studio's configuration options.
Once all the dependancies are installed, an OpenTissue Visual Studio Project can be created using CMake. OpenTissue must be compiled in Visual Studio (this is when you find out if your dependancies are correctly installed), and then OpenTissue is ready to use with the Pik Tracker.

4. Project Stages

4.1 Background Reading and Code Exploration

One of the most difficult parts of working on the Pik Tracker software is the background knowledge needed to fully understand articulated tracking. The first couple weeks of the project was spent doing extensive background research and reading, as well as examining the Pik Tracker source code. In order to grasp articulated tracking, it is important to have a strong understanding of linear algebra, forward kinematics, Sequential Monte Carlo algorithms, and robotics – to name a few. Articulated Tracking literature is heavily based in theory and mathematics, so it is a difficult read that can be quite challenging to one inexperienced in these areas.

The Pik Tracker code itself is quite confusing at first (as to be expected). Even though almost a full week was spent examining it, it was difficult to get a feel for what was happening without being able to actually step through the code with Visual Studio.

4.2 Installing Dependancies

One of the most difficult parts in getting the Pik Tracker to a functional place on a Windows machine is making sure all of the dependancies are collected and installed properly. It was known that OpenTissue was required to run the Pik Tracker, but other dependancies were learned through trial and error. It turns out the main dependancies that the Pik Tracker has are OpenCV libraries, OpenTissue and all of it's dependancies, and a few other library or include files that may or may not be included in a given operating system.

4.3 Running Executables

After OpenTissue and all of the other necessary dependancies were installed a Visual Studio project of the Pik Tracker was created using CMake. Other problems then started to surface. The Pik Tracker code was written in C++ on a Linux system, and therefore there were several consistency problems when it was used with Visual Studio's compiler. Many of these issues had to be tracked down and dealt with manually, but some of them simply required a change in the CMakeList.

After these issues were finally all resolved and a successful build had been done, it was time to examine the executables created by the Pik Tracker. The Pik Tracker created seven

executables:

- *gui_projector*: This program is perhaps the most important one, as it is the one that does the actual tracking. The results of the tracking are shown in a gui that projects the estimated poses onto the original video frame by frame. Therefore, when you run this executable you see the Pik Tracker's estimations overlaying the actual video as the video plays.

- *gui_tracker:* This program is virtually the same as the *gui_projector,* except that the results are shown differently—the data from the stereo camera is used instead. The visualization is the only difference between these two programs.

- *headless_tracker*: This program only does the tracking – there is no visualization or gui. Therefore, *headless_tracker* is primarily used only in the case of using a machine over a network or a similar situation.

- *playground*: This program is one of the least important—it simply serves as a 'playground' for experiments.

- *poser*: This program is used to set the initial 'pose' of a person for the tracker when looking at the data.

- *skelton_sizer:* This program is used to modify the skeleton that is shown over the data. It allows you to change the size of the individual bones and is used when creating person-specific skeleton models.

- *view_data*: This program simply displays the data.

Unfortunately, there were more problems with the code once the executables were run. There were several exceptions that weren't noticeable until the data was being viewed. This required more in depth debugging until most of the errors were taken care of. However, *gui_projector* had some .dll problems that were unresolvable without considerable .dll manipulation. Since *gui_projector* is almost the exact same as *gui_tracker*, it wasn't worth the hassle to fix the .dll problem when *gui_tracker* would work just as well.

4.4 Exploring Jacobian Computation

Once the executables were working, the next step was looking at the predictions the Pik Tracker was making (using gui_tracker) to see how well the articulated tracking was working. It seemed that there was a problem somewhere, as the articulated tracking wasn't making very accurate predictions. The first logical place to check was the Jacobian calculation; if the Jacobian wasn't being calculated as expected the problems would be explained.

In order to check this, the Jacobian first had to be manually calculated with a sample set of data—specifically, an example from a textbook. This is where the reading done at the beginning of the project became important; calculating the Jacobian is not easy without a certain amount of relevant mathematical experience.

After calculating the expected value of the Jacobian, it was necessary to modify the Pik Tracker code to calculate the Jacobian simply on that small example. This required a new skeleton file to be written specifying the values used in the textbook example. As the Pik Tracker was designed to deal with large numbers of bones in a skeleton, the simple three bone textbook example meant that a great deal of the Pik Tracker code had to be modified. Once the code had been sufficiently modified, additional code had to be written to output the Jacobian values in an organized manner.

Right away it became obvious that the Jacobian values being calculated by OpenTissue were not as expected. The size of the matrices were incorrect, and after close examination of the code it was found that it would be impossible for OpenTissue to output a matrix of the correct size.

Unfortunately, the duration of the project had ended before this finding was able to be explored fully, but there are three possible explanations for this behavior:

1) The textbook example was incorrect. This is very unlikely, as researching the matter found no

errors listed in this portion of the textbook.

2) The way OpenTissue is computing the Jacobian is incorrect—at least according to what the Pik Tracker is expecting. This could possible mean that the Pik Tracker is using the wrong *compute_jacobian* function, that OpenTissue has an error in the *compute_jacobian* function, or that OpenTissue does not compute the Jacobian in the manner that the author of Pik Tracker assumed it did.

3) The textbook example was interpreted incorrectly in the creation of the new skeleton file.

5. Future Plans

The issue with OpenTissue's calculation of the Jacobian will continue to be explored in further research at a later date. Once this is resolved by the scholars now working on the Pik Tracker, the Pik Tracker software will be able to be used to compare data gathered from sensor methods (such as motion capture suits) to data generated with articulated tracking—a much less expensive, less invasive, and less costly method.

6. References

1. Hauberg, S., Sommer, S., Pedersen, K.S.: Gaussian-like spatial priors for Articulated Tracking, *European Conference on Computer Vision*, 2010.

2. Ek, H., Torr, P.H., Lawrence, N.D.: Gaussian process latent variable models for human pose estimation. In *Machine Learning for Multimodal Interaction*, 2007.

3. Murray, R.M., Li, Z., Sastry, S.S.: A Mathematical Introduction To Robotic Manipulation. *CRC Press,* 1994.

4. Poppe, R.: Vision-based human motion analysis: An overview. *Computer Vision and Image Understanding,* 2007.

5. Cappé, O., Godsill, S.J., Moulines, E.: An overview of existing methods and recent advances in sequential Monte Carlo. *Proceedings of the IEEE,* 2007.

6. Balan, A.O., Sigal, L., Black, M.J.: A quantitative evaluation of video-based 3d person tracking. *Visual Surveillance and Performance Evaluation of Tracking and Surveillance,* 2005.

7. Sidenbladh, H., de la Torre, F., Black, M.J.: A framework for modeling the appearance of 3D articulated figures. I*nt. Conf. on Automatic Face and Gesture Recognition*, 2000.