

# Manual for Human Tracking Software

**Alexandru Balan**

Department of Computer Science  
Brown University  
Providence, Rhode Island 02912  
[alb@cs.brown.edu](mailto:alb@cs.brown.edu)

*Version 1.0 – November 26, 2005*

## 1. Abstract

A novel human motion dataset, baseline software, and evaluation measures are being made available to the research community in the interests of encouraging quantitative evaluation and comparison. The dataset contains synchronized video from 4 camera views and motion capture data. A basic Matlab implementation is provided for tracking a moving person in a sequence of multiple synchronized images, with known camera calibration information and body dimensions.

## 2. Introduction

The software included in this package is provided for free for research purposes only\*. It provides a basic Matlab implementation for tracking a moving person in a sequence of multiple synchronized images, with known camera calibration information and body dimensions. In addition, evaluation code is provided to compare the tracking results to the ground truth. For bug reports, questions, suggestions and comments, feel free to email me at [alb@cs.brown.edu](mailto:alb@cs.brown.edu). Improvements are also welcome.

Published research reports using this code (or a modified version) should cite present work as follows:

A. Balan, L. Sigal and M. Black. A Quantitative Evaluation of Video-based 3D Person Tracking. *IEEE Workshop on VS-PETS*, 349-356, 2005.

## 3. Method

The algorithm used in our implementation is Deutscher's annealed particle filtering [2] with a few additions. For details about our work and for a detailed set of experiments with this code see [1].

## 4. System Requirements

The software has been tested successfully in Matlab 7.0 under Linux. We expect it to run in Matlab 6.5 with little or no modifications. It should also run under Windows with minimal or no modifications.

## 5. Packages and Installation

There are three separate packages that are needed.

The **LeeTracking** software for human tracking can be downloaded from <http://www.cs.brown.edu/~alb/software.htm>

The **Brown** dataset is a sample dataset compatible with our software and documentation for it can be found at <http://www.cs.brown.edu/~ls/Software/index.html>.

**Note:** We have tested our software using only version 1.0 of this dataset.

Also needed is the Camera Calibration Toolbox for Matlab [http://www.vision.caltech.edu/bouguetj/calib\\_doc/](http://www.vision.caltech.edu/bouguetj/calib_doc/)

Other standard Matlab packages needed are the Image Processing Toolbox, the Optimization Toolbox and the Statistics Toolbox.

The three packages should be unzipped in the same directory, which in the end should contain three directories: `./LeeTrackingX.X` (X.X is the version of the software), `./WebData` and `./TOOLBOX_calib`.

## 6. Image Dataset / Model

The **Brown** dataset contains images used for tracking taken with four synchronized cameras. Also provided are: the camera calibration parameters, the binary maps for the foreground silhouettes, and the marker data from a motion capture system.

We assume an articulated model of the human body with 31 degrees of freedom (DOF) [1]. Type `"help ComputeBodyModelAngles"` for more details.

The marker data has been processed separately to obtain the body model parameters (limb length and width) and the ground truth values for the joint angles, position and orientation. The results were stored in `./LeeTrackingX.X/TestData/LeeWalkTest.mat`. Similar data from a different and more extensive motion capture session were stored in `./LeeTrackingX.X/TrainData/LeeWalkTrain.mat`.

**Note:** The ground truth obtained with the motion capture system was good but not perfect. We provide as a convenience a second ground truth file `./LeeTrackingX.X/TestData/LeeWalkTestFitImages.mat`. Here the ground truth data has been adjusted to fit the images better by optimizing over the likelihood function used in the present work. Therefore the latter data is not unbiased anymore.

## 7. Usage Instructions

There are 6 functions in `./LeeTrackingX.X` that can be used to perform tracking and evaluate the results. In most part, they can be executed in an interactive way without passing any parameters. The parameters are meant to reduce the amount of interaction. However in the current version the functions need to be executed from the directory in which they are located, namely `./LeeTrackingX.X`.

- Start by running `InitPath()`, which updates the Matlab PATH with paths to the three packages.
- To initiate tracking, run `TrackLee()`. You will need to select a file name for a tracking log where the results from tracking will be saved. If no options are provided, default values are used. Type `"help LoadTrackingParameters"` for more details on tracking options.

Examples:     **TrackLee([], 'out/results.mat');**  
                  - Tracking with the default options and saving results  
                  in the specified file

**options.NUM\_PARTICLES = 200;**  
                  **options.VIEWS = [2 4];**  
                  **TrackLee(options, 'out/results.mat');**  
                  - Tracking with modified options

*Implementation details:* The tracking log file will contain structure variables such as **PRM**, **MODEL\_PRM**, **InitParam** and **Result**. While the first three are specific to the tracking method used, the fourth one, **Result**, is method independent and is the only one necessary for evaluating the results (see **ErrorGraph** below). **PRM** and **MODEL\_PRM** hold tracking and body model parameters. **InitParam** holds the states necessary to restart tracking from any frame.

- To quickly find out the tracking parameters for a log file, use `LogInfo()`.

Example:     **LogInfo('out/results.mat');**

- If tracking is interrupted, it can be resumed using `ContinueTracking`. It can also be used to continue tracking for additional frames, or restart from an earlier frame.

Examples:     **ContinueTracking('out/results.mat', 10, 3);**  
                  - Track 3 frames starting from frame 10

**ContinueTracking('out/results.mat', 'end');**  
                  - Restart tracking from where it left off

- To visualize the results, run `ShowResults()`. You will need to select the file where the results were saved. It will also give you an option to save the

overlaid images to files as JPG. The ground truth of the body model projection is overlaid on the image using black lines. The mean particle (weighted average of all particles) is shown with color lines. The left and right sides of the body are color coded.

Examples: `ShowResults('out/results.mat', true, 'out');`

- To evaluate the results, run `ErrorGraph()`. It will generate a graph with the tracking error with respect to the ground truth.

Examples: `ErrorGraph('out/results.mat');`  
- Compute default error but do not save to log file

`ErrorGraph('out/results.mat', @weightedError);`  
- Compute using the specified error method

`ErrorGraph('out/results.mat', [], true);`  
- Save computed error to log file

*Implementation details:* The error is computed using a marker-distance metric. From the joint angles of a given particle the spatial position of 15 markers on the body is computed and compared with the true position. The error for one particle is the mean distance error over all markers. Multiple error measures can be devised to measure the error at a given frame of the entire particle set. The default error is the @optimisticError  $\hat{\Delta}_R$  as introduced in [1], but which computes the minimum error of the 10 most likely particles together with the expected pose. Other options include a weighted average of particle errors (@weightedError) and the error of the most likely particle (@MAPerror). (Note: It is possible that the way the error is computed may change in the future versions).

Even if other tracking algorithms are used, they should still record the hypotheses for each frame and their weights as `Result.poses` and `Result.weights` to the tracking log. This way `ErrorGraph` can still be used for evaluation.

Function `ErrorGraph` returns a vector with the error for each frame if needed for post-processing. It can also be saved to the log file as `ERRORS.ERR`.

Type "help ErrorGraph" for more details.

## 8. References

- [1] A. Balan, L. Sigal and M. Black. A Quantitative Evaluation of Video-based 3D Person Tracking. *IEEE Workshop on VS-PETS*, 349-356, 2005.
- [2] J. Deutscher and I. Reid. Articulated Body Motion Capture by Stochastic Search. *IJCV*, 61(2):185-205, 2004.

## 9. Copyright

\* Copyright 2005, Brown University

All Rights Reserved Permission to use, copy, modify, and distribute this software and its documentation for any non-commercial purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of the author not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTUOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.