

# FicTrac: a visual method for tracking spherical motion and generating fictive animal paths

---

By Richard Moore ([rjdmoore@uqconnect.edu.au](mailto:rjdmoore@uqconnect.edu.au)) 26/05/2011 (updated 24/02/2014).

***IMPORTANT:*** consult the *README* text file distributed with *FicTrac* prior to use!

## **What is FicTrac?**

The FicTrac program will compute the absolute orientation of a patterned sphere from images captured directly from an attached 1394a/b or USB camera or loaded from a specified video file.

FicTrac has been designed for use with tethered animal walking experiments, in which a stationary animal subject walks on a freely rotating spherical treadmill. The animal's attempted locomotion is represented by rotations of the sphere, which are measured by FicTrac to reconstruct the animal's instantaneous walking velocity (i.e. speed and direction of motion). By integrating those motions over time, the absolute heading direction and 2D path of the animal on a fictive plane is determined.

This document contains important information for using the software, as well as a brief overview of the approach, which may help with getting the most out of the program. For a more scientific description of the algorithm, please consult

(Moore *et al.*, 2014) R. J. D. Moore, G. J. Taylor, A. C. Paulk, T. Pearson, B. van Swinderen, M. V. Srinivasan, *FicTrac: a visual method for tracking spherical motion and generating fictive animal paths*, Journal of Neuroscience Methods, Volume 225, 30<sup>th</sup> March 2014, Pages 106 – 119, <http://dx.doi.org/10.1016/j.jneumeth.2014.01.010>.

## **How does the program work?**

FicTrac estimates the instantaneous orientation of the sphere by localising the current view within a stored map of the sphere's surface. The stored surface map is generated automatically and continuously by localising and storing new aspects of the sphere's surface as it is rotated. Eventually (typically after a few seconds of undirected motion by the animal), the entire surface of the sphere will be mapped and each new view of the sphere can be localised *absolutely* (i.e. the absolute orientation of the sphere each frame can be determined independently). The sphere surface map is stored as a rectangular image using an equal-area cylindrical projection. An example input image and corresponding sphere surface map are shown in Figure 1.

Using this method, estimates of the sphere's orientation will *not* drift over time. The animal's fictive heading direction and 2D path must still be integrated from the instantaneous sphere rotation rates however, because the *sequence* of sphere rotations is important for reconstructing the animal's path. This is true of all similar ball tracking methods (i.e. optical mouse-based methods), however FicTrac is likely to be more accurate than those other methods because noise affecting the underlying measurements (i.e. rotational velocity of the sphere) will not be biased. This is guaranteed for FicTrac because the *absolute orientation* of the sphere is estimated independently each frame, rather than the relative rotation of the sphere between frames. Other methods that measure the relative motion of the sphere between frames (i.e. optical mouse) could be affected by measurement biases, which would degrade estimates of the animal's heading and path when integrated. For a more rigorous analysis of the errors affecting FicTrac and optical mouse systems, please consult (Moore *et al.*, 2014).

In order to transform the rotations of the sphere – that are measured in the coordinate frame of the camera – into animal motions, the 3D orientation of the camera with respect to the reference frame of the animal must be known. However, this rotational transformation can be estimated automatically by FicTrac during the interactive configuration procedure.



Figure 1. (left) Crop from the input webcam image and (right) the segmented sphere surface, showing (top) the instantaneously visible surface and (bottom) the complete surface map. The sphere surface is represented by an equal-area cylindrical mapping: imaginary horizontal lines drawn across the bottom, middle, and top of the sphere map would correspond to the south pole, equator, and north pole on the real sphere (in the camera's frame of reference); imaginary vertical lines would correspond to degrees of longitude on the sphere. Note that the segment of the sphere obscured by the bee is discarded (this is specified by the user with a pixel mask).

The user (you) interfaces with FicTrac via a **configuration file**, which is specified via the command line and loaded by FicTrac at run time. The configuration file contains all the various parameters that can be modified to control how **FicTrac executes (input/output directories, camera parameters, precision/quality, etc)**. Many of the parameters default to reasonable values and can be ignored, but some are important and must be set accurately for each experimental setup (see section *Configuration*). All of the information about a particular experimental setup that is required by FicTrac is stored in a special transform file that is read in by the FicTrac program at run-time. If the transform file is not found then its contents are generated automatically via an interactive configuration procedure.

### **What is required to run FicTrac?**

#### ➤ Software

- ◆ FicTrac binaries pre-compiled for the Ubuntu 12.10 operating system are available from <http://bit.ly/fictrac>. FicTrac has also been tested successfully on native Windows 7 & 8 machines using a virtual machine (VMware).
- ◆ FicTrac depends on several open-source software libraries, which must be installed on the Ubuntu operating system and present at run-time in order for the program to execute. They may be installed in the default directories (e.g. `/usr/local/lib/`). The dependencies are:
  - OpenCV (`libopencv_core`, `libopencv_highgui`, `libopencv_imgproc`)

- FFmpeg (*libavformat, libavcodec, libavutil, libswscale*)
- Cairo (*libcairo, libcairomm-1.0*)
- NLOpt (*libnlopt*)
- Miscellaneous (*libpthread, libsigc++-2.0*) – should be installed by default

Most of these libraries can be installed via synaptic package manager (Ubuntu). Alternatively, the latest versions can be obtained by downloading and building from source files.

*Hint: use the ldd unix command to print the shared libraries (and their expected locations) required at run-time by an executable.*

- ◆ FicTrac should execute in approximately 10 ms on a 3 GHz processor, using a quality setting of 6 and with all debug display disabled.


#### ➤ Camera

- ◆ Either a 1394a/b computer vision camera or USB webcam that is supported by either OpenCV (v2.4) or the Point Grey FlyCapture SDK (PGR cameras only) must be attached to the system and configured correctly (**very limited image/camera configuration can be performed via FicTrac**) prior to executing FicTrac. Alternatively, a video file (with format supported by OpenCV v2.4) can be specified in the configuration file.
- ◆ An example region of interest (ROI) from an input image is shown in Figure 1. Note that this image is cropped – you do *not* need to crop or zoom (on board the camera or by using a lens) your input images. However, if you do, you *must* specify the **correct final vertical field of view (FOV)** for the input images in the configuration file.
- ◆ The exposure time (shutter speed) of the camera should be set as short as possible to minimise blurring – any visible blurring of the sphere pattern will degrade FicTrac's performance. Exposure time should not be set so short that the program has difficulty segmenting the pattern on the sphere (i.e. if the whole image is too dark then white areas may appear similar to black areas, particularly near the “edges” of the sphere). Check the segmented region of interest (ROI) in the debugging display to ensure the segmented pattern is correct.
- ◆ The frame rate of the camera should be set high enough so that the sphere does not rotate more than ~20 degrees between frames. If the camera is used online, then the frame rate shouldn't be too high for the execution speed of the program (printed to console). **A frame rate of 10 Hz – 100 Hz should be fine under most circumstances.**
- ◆ As much as possible of the tracking sphere should be visible, and the camera should be positioned appropriately – too close and the depth of field of the camera will cause some of the sphere to be out of focus; too far and the resolution of the sphere will be poor.
- ◆ **The position of the sphere within the image and the orientation of the camera are both very important and are set only once during configuration** – if the camera is bumped or moved during or between experiments, the interactive configuration procedure must be repeated. If care is taken to keep both the camera and the relative position of the sphere stationary then a single transform file can be used for multiple experiments.

#### ➤ Tracking sphere

- ◆ A distinct and irregular (not self-similar nor tessellating) binary (two-colour) pattern

must be applied to the surface of the sphere in order for FicTrac to segment and localise each view successfully.

- The pattern should contain a mix of spatial frequencies to ensure fast and precise matching.
  -  e-drawn patterns of blobs with approximately equal areas of foreground and background work well. The ratio of the area of each blob to the total area of the sphere should be approximately equal to that indicated in Figure 1 (you should have approximately the same number of blobs on any sphere, regardless of the sphere's diameter).
  - Colour combinations other than black/white may be used, but black/white is preferred because anything less distinct than that might degrade performance. Normally, FicTrac applies an adaptive threshold to the intensity of the pixels within the ROI to segment the ball pattern (colour information is discarded). However, if *use\_ball\_colour* is enabled in the configuration file, then FicTrac converts the input pixels to the YUV colour space and segmentation is performed based on their chrominance values instead (intensity information is discarded). For colour-based segmentation, examples of the typical colours of the foreground and background regions must be specified during the automatic configuration procedure.
  - ◆ Specular reflections *must* be avoided – glossy inks, reflective sphere material, and direct lighting will all cause specular reflections on the sphere which may lead to poor matching performance. Direct or intense lighting will also lead to shadowing on some areas of the sphere, which may degrade matching performance.
- Experimental setup
- ◆ FicTrac has been designed to suit a wide variety of experimental configurations – a sphere (and thus animal) of any size and a wide variety of sphere colours/patterns may be used, and the camera may be positioned anywhere within the experimental arena, as long as **>~25% of the sphere's surface is visible to FicTrac** (the exact proportion that must be visible depends on other factors such as the ball pattern and lighting – greater visibility leads to greater robustness).
  - ◆ A square shape, plate, or pattern must be aligned with the animal's forward axis and visible within the FOV of the camera during the interactive configuration procedure, *if and only if* the exact 3D orientation of the camera with respect to the coordinate frame of the animal is unknown. The structure supporting the tracking sphere may be used for this purpose (see Figure 2).

## **Executing the program**

The only input required by the FicTrac program is the path to a configuration file (e.g. *config.txt*):

```
/path/to/FicTrac /path/to/config.txt
```

To execute the program in batch mode (i.e. for processing many videos off line) the supplied script can be used to “glob” input file names:

```
/path/to/run_fictrac.sh “/path/to/config.txt” “/path/to/input_video*.avi”
```

Note that the “quotation marks” are necessary in this case. This command will execute the program on all video files matching the pattern *input\_video\*.avi* (i.e. *input\_video\_01.avi*,

*input\_video\_02.avi*, ...) in the directory */path/to/*. The configuration parameters specified in *config.txt* will be used for all videos<sup>1</sup>.

## **Configuration**

The configuration file specifies the values of several parameters that control how the program runs. Importantly, the configuration file also specifies the path to several other files required by FicTrac to produce valid output. The syntax of the configuration file is keyword and value pairs, separated by spaces or tabs. All parameters are optional (except for some combinations of parameters, i.e. input from video enabled but no input video filename specified), and the program will default to typical values if a particular keyword is not found. Some keywords only make sense in some situations and will be ignored in others (i.e. *input\_vid\_fn* will be ignored when *cam\_input* is selected).

The supported keywords, default values, and their descriptions are given below (boolean true/false values should be given as 1/0 respectively):

Keyword	Default value	Description
<i>input_vid_fn</i>	-	Path to the input video file, if running from video.
<i>output_fn</i>	< <i>input_vid_fn</i> >.dat	Path to the output data file (produced by the program). Will default to the same as the input video file (with the extension replaced by .dat) if running from video, terminates with an error if not specified and running from camera.
<i>mask_fn</i>	< <i>output_fn</i> >-mask.jpg	Path to a mask image. Zero (black) values mask out non-sphere regions such as animal or structure, 255 (white) corresponds to good regions. Program terminates with an error if mask not found.
<i>transform_fn</i>	< <i>output_fn</i> >-transform.dat	Path to the transform configuration file. File is generated automatically by a configuration procedure if not found. Once generated, the same transform file can be specified and used for multiple experiments, as long as the camera is not moved.
<i>template_fn</i>	< <i>output_fn</i> >-template.jpg	Path to a sphere map template image. Sphere map auto-generates as the program executes and auto-saves on exit. If the same sphere is used for multiple experiments, a previously saved template can be loaded by setting <i>load_template</i> to true.
<i>closed_loop_fn</i>	-	Path to an output file, useful for closed-loop control.
<i>frame_skip</i>	0	Number of frames to skip at the beginning of a video file.
<i>frame_step</i>	1	Number of video frames to increment each time step.
<i>do_display</i>	1	Show visual debugging information. Slows execution slightly but useful for problem solving or seeing results in real time.
<i>no_prompts</i>	0	Never prompt user for input, useful for batch execution.
<i>do_config</i>	0	Force the configuration procedure to run prior to execution, will

---

<sup>1</sup> A single mask, sphere template, and transform can be used for all videos in batch mode by specifying the appropriate files in the configuration file (*config.txt*). This saves having to run the configuration procedure on every input video individually.

		auto-run if no <i>transform_fn</i> found.
<i>do_search</i>	1	Perform a global search if orientation match fails. Warning: this is very slow, only enable for offline processing; for online processing, just make sure the program never fails I guess...
<i>do_update</i>	1	Update the sphere template. Set to false if loading a sphere template ( <i>template_fn</i> ) and you don't want any additional learning during experiment.
<i>save_video</i>	0	Save debug display to video. Video is output to <i>&lt;output_fn&gt;-debug.avi</i> .
<i>save_input_video</i>	0	Save raw input frames from camera to video. Video is output to <i>&lt;output_fn&gt;-raw_frames.avi</i> .
<i>load_template</i>	1	Load sphere template if <i>template_fn</i> is found.
<i>fisheye</i>	0	Input camera model rectilinear/fisheye. Defaults to rectilinear.
<i>cam_input</i>	0	Capture directly from camera attached to computer, defaults to input from video ( <i>input_vid_fn</i> )
<i>fps</i>	-1	Target frames per second when processing offline, defaults to “as fast as possible”, a value of 0 will cause program to pause for user input each frame. Attempts to set camera frame rate when capturing directly from a camera ( <i>cam_input</i> = 1).
<i>cam_index</i>	0	Selects which camera to use if there are multiple cameras connected.
<i>use_ball_colour</i>	0	Allow user to select foreground and background colours during configuration, defaults to 0 – black and white.
<i>vfov</i>	45	Vertical field of view of the input images in degrees. It is important to specify this parameter accurately! Warning: if you crop the input images then you must reduce this parameter correspondingly.
<i>quality_factor</i>	6	Adjusts the balance between resolution (and hence precision) and execution time. Smaller numbers correspond to coarser but quicker tracking and vice versa. Default (6) executes at ~80Hz on a 3GHz processor.
<i>nlopt_tol</i>	1e-4	Termination tolerance for the non-linear optimisation algorithm used to localise the sphere's orientation.
<i>nlopt_max_eval</i>	100	Maximum number of function evaluations to perform during localisation of the sphere's orientation.
<i>error_thresh</i>	1e4	Matching threshold. Frames that result in a minimisation score greater than this threshold will be discarded and a global search will be performed to re-localise the sphere if <i>do_search</i> is true.
<i>thresh_win</i>	15	Size of neighbourhood window to use for adaptive thresholding of input sphere ROI – larger window size avoids over-segmentation of constant-colour areas, smaller window size makes segmentation more robust to intensity gradients across sphere.

		Parameter is only used if <i>use_ball_colour</i> is 0.
<i>thresh_ratio</i>	1.0	Affects adaptive thresholding of input sphere ROI. Values > 1.0 preference foreground regions (more white in segmented image), values < 1.0 preference background regions.
<i>max_bad_frames</i>	0	If set to a value > 0, tracking will reset when this many frames are dropped in a row. Allows the program to recover and continue after dropping a frame/s without performing an expensive global search. Warning: if tracking is reset, the frame of reference for the output heading direction (and hence also the integrated 2D path) will change invisibly. The output data should be split into contiguous tracking segments. The variable <i>sequence number</i> in the output data file indicates reset occurrences.
<i>do_serial_out</i>	0	Enable to output a subset of FicTrac's output data via serial.
<i>serial_baud</i>	115200	Baud rate to use if <i>do_serial_out</i> > 0.
<i>serial_port</i>	/dev/ttyS0	Serial port to use if <i>do_serial_out</i> > 0.
<i>do_led_display</i>	0	Enable to draw a small (32 px * 128 px) green (RGB: 0,255,255) window to screen containing a single dark (RGB: 0,0,0) vertical (32 px * 4 px) bar. This window may be used to provide visual feedback for the animal.
<i>do_socket_out</i>	0	Enable to output a subset of FicTrac's output data via network sockets. The socket port is selected randomly from available ports at run-time. Consult the terminal output for the correct port to connect to.

If *mask\_fn*, *transform\_fn*, or *template\_fn* are not specified, then the program will look for them in the same directory as *output\_fn*, with file names specified by *output\_fn*, plus the relevant extension.

### Configuration procedure

The interactive configuration procedure is run automatically prior to execution if no valid *transform\_fn* is found. The purpose of the configuration procedure is to specify the location of the tracking sphere within the FOV of the camera, as well as to specify the transformation from the camera coordinate frame to the laboratory (animal) coordinate frame. The output of the configuration procedure is a transform file (*transform\_fn*), which is simply a text file with the following values (these shouldn't normally be modified or set by hand, they are set automatically by the configuration procedure):

R <sub>1</sub> R <sub>2</sub> R <sub>3</sub> R <sub>4</sub> R <sub>5</sub> R <sub>6</sub> R <sub>7</sub> R <sub>8</sub> R <sub>9</sub>	Rotation matrix specifying transformation from camera coordinate frame to laboratory coordinate frame.
T <sub>x</sub> T <sub>y</sub> T <sub>z</sub>	Translation vector from camera nodal point to centre of laboratory coordinate frame (unused).
S <sub>x</sub> S <sub>y</sub> S <sub>z</sub>	Unit view vector to the centre of the sphere in the camera coordinate frame.
S <sub>FOV</sub>	Angular field of view subtended by the sphere.



FG <sub>Y</sub> FG <sub>U</sub> FG <sub>V</sub>	YUV of the foreground blobs.
BG <sub>Y</sub> BG <sub>U</sub> BG <sub>V</sub>	YUV of the background blobs.
VFOV	Vertical field of view of the input images.
CAM_MODEL	0 = rectilinear; 1 = fisheye

During the interactive configuration procedure the user is prompted for input. The user interacts with the program by clicking on the display window with the mouse. The exact input required from the user depends on the parameters specified in the configuration file. As a minimum, the user will be required to select the exact outline of the sphere and the exact outline of a square, which must be visible within the FOV of the camera so that the rotational transformation between the camera and the animal can be estimated (if this rotation transformation is known *a priori* or can be determined manually, the parameters for the correct 3D rotation matrix can be used to overwrite the corresponding values in the transform file that were generated during the interactive configuration procedure).

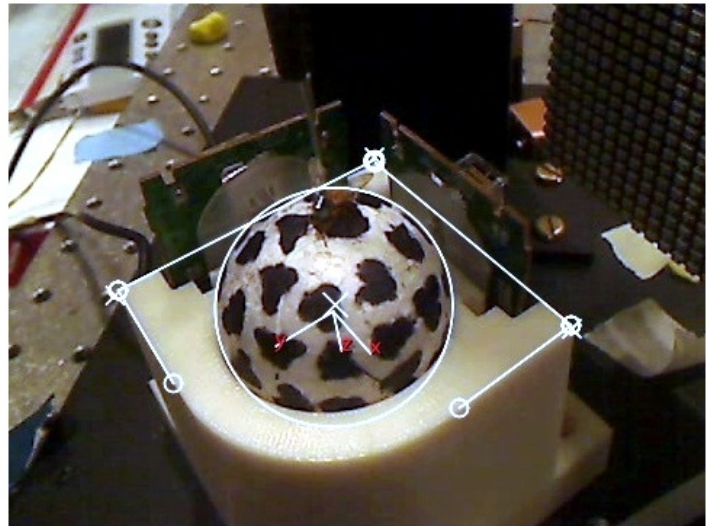


Figure 2. Visual output of the configuration procedure showing user-defined outline of the tracking sphere and square support. Centre and radius of the sphere outline is adjusted to exactly match the circumference of the tracking sphere (neglecting obscured regions); sides of the square are selected to exactly match the physical dimensions. Also shown are the computed axes for the animal.

The configuration procedure is illustrated in Figure 2. It can be seen that the circular region selected by the user corresponds exactly to the circumference of the tracking sphere (neglecting obscured regions of the sphere, these areas should be masked out by the user). The dimensions of the square are computed by selecting either the four sides or the corners of the square. The order in which the corners or sides are selected is important because the orientation of the square defines the coordinate frame of the animal with respect to the camera. After configuration, the computed axes of the lab (animal) coordinate frame are drawn onto the configuration image. If the axes are inaccurate the configuration procedure must be repeated because the data output by the program will be in the incorrect coordinate frame and will therefore be meaningless.

### **Program output**

Several files are output at different stages of the program. Header files describing the format of each output file can be found in the help docs directory.

- ◆ *output\_fn*: Data output file generated throughout the execution of the program. Data is output in comma separated variable (CSV) format – i.e. columns represent parameters and rows represent frames. The contents of the data file is as follows:



Cols	Data	Description
1	Frame counter	Corresponding video frame (starts at #1).
2-4	Delta rotation vector (cam)	Change in orientation since last frame, represented as rotation angle/axis (radians) in camera coordinates (x right, y down, z forward).
5	Delta rotation score	Error score associated with rotation estimate.
6-8	Delta rotation vector (lab)	Change in orientation since last frame, represented as rotation angle/axis (radians) in laboratory coordinates.
9-11	Absolute orientation vector (cam)	Absolute orientation of the sphere represented as rotation angle/axis (radians) in camera coordinates.
12-14	Absolute orientation vector (lab)	Absolute orientation of the sphere represented as rotation angle/axis (radians) in laboratory coordinates.
15-16	Integrated x/y position (lab)	Integrated x/y position (radians) in laboratory coordinates. Scale by sphere radius for metric position. Incorporates animal's changes in heading.
17	Integrated animal heading (lab)	Integrated heading direction (radians) of the animal in laboratory coordinates. This is the absolute direction the animal would be facing if it wasn't tethered.
18	Animal movement direction (lab)	Instantaneous running direction (radians) of the animal in laboratory coordinates. This is the direction the animal is moving in the lab frame (add to heading to get instantaneous movement direction in world).
19	Animal movement speed	Instantaneous running speed (radians/frame) of the animal. Scale by sphere radius for metric speed.
20-21	Integrated forward/side motion	Integrated x/y position (radians) of the sphere in laboratory coordinates neglecting heading – i.e. equivalent to the accumulated vertical component of the output from two optic mice sensors placed beside and behind the animal.
22	Timestamp	Either of: position (ms) in video file; or real time elapsed between 00:00 Jan 1 1970 UTC and capture time of input frame.
23	Sequence number	Frame number in current tracking sequence. Usually corresponds directly to <i>frame counter</i> but can reset to 1 if tracking is reset (see parameter <i>max_bad_frames</i> in input configuration file).

- ◆ *<output\_fn>-transform.dat*: Configuration output file generated following configuration procedure. The content of the transform file is described in the previous section.
- ◆ *<output\_fn>-configImg.jpg*: Configuration image generated following configuration procedure. Useful for confirming properties of coordinate transform. The required mask image can be generated using this image as a starting point.
- ◆ *<output\_fn>-template.jpg*: Sphere map template image generated following execution of program. Following execution, a snapshot of the sphere surface map is saved to disk so that it can be loaded for subsequent experiments. A template image can also be saved at any point during execution by pressing “s”.
- ◆ *closed\_loop\_fn*: If specified in the config file, this file is overwritten with a single line of

data each frame. Five variables are written to disk (brackets indicate column # of matching parameter from *output\_fn*):

1	Frame counter
2	Integrated forward motion (20)
3	Integrated side motion (21)
4	Integrated heading direction (17)
5	Frame counter

The data values output to *closed\_loop\_fn* are identical to the values output in the corresponding row of *output\_fn*. The five values are overwritten each frame, however, so the file may be polled regularly for use within a closed-loop setup. The current frame number is output first and last so it should be obvious if new data has successfully been written to the file, or if the data was being written as the file was polled (i.e. if both frame counter parameters match, the data is good).

### **Notes**

For closed-loop operation, it is important that the program executes in real time – *do\_search* must be disabled in the configuration file. With *do\_search* disabled, care must be taken to ensure that the program is not pushed beyond its operational limits for any input frame – i.e. if lighting conditions are not ideal and segmentation fails, then the program might lose track of the sphere's orientation and fail to match until *max\_bad\_frames* is reached (or for the remainder of the experiment if *max\_bad\_frames* = 0).

This program is still very much under development but has been tested thoroughly. Please make use of it if you can but no guarantees can be provided that it will work properly under all conditions or that the data that is output is accurate. Please contact the author if you are having issues with the software.