Treinin Lab
Worm Movement Analyzer

Table of Contents

2011/02/11	0.01	Basic functionality: reading movie, binarization, skeleton, head-tail identification, motion analysis, all based on using OpenCV
2011/02/20	0.02	Skip loops and don't crash
2011/03/22	0.03	Add begin mark in track; calculate displacements differently
2011/09/18	0.04	Improve loop handling, improve skeleton detection at ends, add initial eggs identification
2011/10/26	0.05	Assume separate mmpix for stage to get both metrics and track right (workaround for possible bug in tracker data)
2011/11/01	0.06	Fix bugs with movement segments and resulting displacement and speed calculations
2011/11/28	0.07	Add measurement of angle on large segments, frequency, and option to avoid edge detection. Add scale on wave output. Scale track to fframe to lframe segment only.

1. Installation and Setup

To install the software you only need to copy the executable.

Note that the software itself is only a small part of the disk space that you will use. There are also the movies and the results of the analysis. It may therefore be advisable to use a separate disk or partition for this project. In this manual we'll assume this is disk **D**. Alternatively, you can just use a folder such as **C:\worms**. The only restriction is that there can be no spaces anywhere in the folders or file names. In other words, a folder under **My Documents** is out.

It is suggested to create two folders:

- **D:\movies**
- **D:\analysis**

Copy the installation files into **D:\analysis**. This includes the following:

- The executable, which is called **TLWMA.exe**. This is the only file that is really required.
- A sample parameters file, called **params.txt**. This is provided as an example that you can edit for your needs. Alternatively, you can create one from scratch.
- This user manual.

Copy the movies you want to analyze (and the associated stage movement data files) into **D:\movies**.

System Requirements

This software was developed for a PC running Windows XP.

The movies and analysis require lots of memory. In our setup, a 1 minute movie occupies about 25 MB of disk space. The output files also require several MB.

It is recommended to run this program on a system with plenty of RAM, say 1 GB or more.

2. Running an Analysis

Inputs

The analysis requires three input files:

- A parameters file as specified below. Among other things, it tells the program where to find the two files specified next. The parameters file must be located in the same folder as the executable.
- The movie to be analyzed. This must be an AVI file with suffix ".avi". The codec used must be supported by the system; in our system this was MJPG.
- A data file describing the movements of the stage as it follows the worm. This should have the same name as the movie itself, but with a suffix of ".log.csv". The format of this file is comma-separated values. The first line gives the headers of the different columns. In subsequent lines there are 5 fields separated by commas, providing the following data:
  1. Timestamp (min:sec or date and hr:min:sec)
  2. Media time (min:sec or hr:min:sec), i.e. the same timestamps but starting from 0
  3. The word "STAGE"
  4. X coordinate in microns
  5. Y coordinate in microns

  Seconds and microns are given as decimal values.

Parameters

Analysis parameters are given in the parameters file. This file is called **params.txt**. It must be located in the same folder where the program is being run.

The format of the parameters file is a sequence of lines, where each line has the format
name = value
lines starting with '#' are comments and ignored.

At the end of the analysis, the parameters used are saved in a file together with the results files. This is in the same format as the original parameters file, and can be used as input to repeat the analysis. However, some parameter values may be overwritten with actual computed values (e.g. **lframe** will be the actual last frame analyzed).

Most analysis parameters have reasonable defaults. Using an empty parameter (that is, not providing the value) is equivalent to not specifying the parameter at all, and the default will be used. Unrecognized parameters (including spelling errors) are simply ignored.

Parameters for specifying the input movie are:

- **moviedir**: [*Required*] directory in which movie(s) are located. May not include spaces. String.

- **moviename**: [*Required*] movie file to analyze, without suffix (for example, if the movie name is n2_2010_06_01__08_21_38__1.avi, this parameter should be n2_2010_06_01__08_21_38__1). May not include spaces. String.
- **fframe**: [0] first frame to analyze. Integer.
- **lframe**: [0] last frame to analyze. If 0, analyze till end of movie. Integer.
  fframe and **lframe** are used to analyze the movie in parts, or to stop the analysis in time if the worm is known to crawl out of the frame (which makes the program crash).
- **ignorefrm**: frames that should be ignored in the analysis. Use this to redo an analysis that failed on some frames. For a single frame, just give its number. For a sequence of frames, use the format from-to (inclusive). This parameter may be repeated as many times as needed, and all the specified frames will be ignored. If multiple frames are specified, they should be given in the correct order with no repetitions.
- **mmpix**: [*Required*] magnification used in recording the movie, expressed as pixels per millimeter. This parameter is extremely important in order to measure distances and speeds correctly. To obtain it, create a short movie of a ruler in the same magnification as used to record moving worms. Do not change the magnification. Integer.
- **frmps**: [-1] frame rate to use. If specified, this overrides the data in the movie. Should not be used unless the movie data is clearly wrong, as it affects all speed calculations. Default (-1) is to use data from the movie. Floating point.

Parameters affecting the analysis are:

- **smooth**: [4] smoothing factor when creating a histogram of grey levels to find a suitable threshold. Should be an integer that divides 256, e.g. 1 (=use all gray levels without smoothing), 2, 4, or 8. Integer.
- **thresh**: [-1] the threshold to use in binarization. This overrides the calculation using the histogram; leave empty or use -1 to use the histogram-based threshold. Integer.
- **useedges**: [1] use edge detection as part of the binarization. This usually helps identify the worm better, especially in the head region that sometimes has very light areas. However, it may cause problems in movies where the worm leaves very pronounced tracks, because these tracks may be identified as part of the worm. If this happens and many frames are flaged as bad, try turning edge detection off by setting this parameter to 0. Boolean.
- **track**: [stage] given that the camera is set up on a moving stage, we need to know its location in each frame to calculate movement correctly. Regrettably, the provided data is not completely reliable. Therefore two options are given. One is to use the stage data anyway, indicated by a value of stage. The other is to calculate the correlation between pairs of successive frames, and try to find the displacement that gives the best correlation. This is indicated by a value of corr. Experience indicates it is not really better, but is a lot slower. String.
- **stageskip**: [1] as the timestamps associated with stage movement may be inaccurate, it is possible to skip several frames before and after each indicated movement. Skipping like this avoids bogus movements that are a result of the stage

moving in one frame, but we think it moved in another frame. This parameter specified how many to skip. Integer.

- **mmpixstage**: [-1] empirically it seems that the data about stage movement is off by a factor of 2. In order to avoid hardcoding a correction for this, this parameter allows you to define a separate mmpix value for use in tracking the movement. If it is -1 (or just left undefined) then the regular **mmpix** is used. Integer.
- **seglen**: [0.5] the motion analysis is based on dividing the movie into short segments, and analyzing the motion in each such segment. Later, successive segments with consistent motion are united into longer segments. **seglen** is the length in seconds of the basic short segments. This is also used for smoothed speed measurements. Floating point.
- **pausespdlim**: [0.05] one of the motion types identified is a pause, where the worm does not move. This parameter sets the pause speed limit: if the motion is slower, it will be considered a pause. The units are millimeter per second. Correct calculation of the speed depends on correct setting of the **mmpix** parameter. Floating point.
- **displen**: [0.5] displacement measurements are done over a certain interval. This is that interval, in seconds. Floating point.
- **anglen**: [10] track angle measurements are done over a certain interval. This is half of that interval, in seconds (the angle at a point is the angle between lines connecting it to points that are a half-interval in either direction). Floating point.
- **minloopradius**: [0.02] minimum loop radius in mm, and reflects the flexibility of the worm. If the value is too big then small loops won't be recognized, and if it is too small then gloss might be recognized by mistake as a loop.

Parameters affecting the output are:

- **outdir**: [*Required*] directory in which output files will be stored. Actually a subdirectory named after the movie will be created, and all the output files will be stored there. May not include spaces. String.
- **shade**: [0.2] level of shading to add to tracking image, for each frame of the movie. Floating point.
- **star**: [0] if 1, put a star '*' to mark the starting position in the track image and an 'x' to mark the ending position. Boolean.
- **maxskellen**: [350] maximum skeleton length in pixels. This is used to set the Y dimension of the wave image. Integer.
- **distbox** [0]: if 1, create data for boxplots to show distributions of metrics. 0 means to create percentiles. Boolean.
- **emptyrows** [0]: if 1 include empty rows to indicate that data about certain movement types (backwards, omega) is missing, because there was no such movement. The default is not to show such rows with no data, but including them may be helpful if the data is loaded into a spreadsheet. Boolean.
- **eggs**:[0] if 1, finds eggs that were laid, marks them with circles on the track image, and generates an output file listing them. Works only if the stage movements are reported correctly. Boolean.

Parameters useful for debugging are:

- **screenop**: [0] if set to 1, details of the analysis will be displayed on screen during execution. 0 means not to show such details. Boolean.
- **debugimg**: [0] show images of binarization, skeleton, and head-identification at runtime.

Executing the Program

To execute the program, simply double-click on its icon. This will cause a console window to open – a small black window where the program displays text output during its execution.

The program will read the parameters file, and start analyzing the movie. Note that the parameters file must reside in the directory (folder) where you run the program.

As the program runs, it will first print the name of the movie it is analyzing:

Analyzing D:\movies\n2_2010_06_01__08_21_38__1.avi

Then, after a short pause, it will print the number of each frame it is working on:

#0 #1 #2 #3 #4 #5 #6 #7 #8 #9 #10 …

If the **screenop** parameter is set, more data will be printed out.

If the **debugimg** option is set, the movie will be shown at a reduced size such that all movements can be accommodated within the frame. Thus if the worm moves large distances the original frames will be reduced a lot, but if it only moves a bit the reduction will be mild. Importantly, when the stage moves you should see the *background* jump, but the worm should stay more or less in the same place. If the worm jumps back and forth, consider increasing **stageskip**. Another window with the current frame, binarization, and skeleton will also be shown.

When the program finishes reading the movie, you need to close the output images to allow it to finish the analysis.

3. Reading the Output of the Analysis

The output of the analysis is contained in several text files and some picture files. All these files are located in the results folder named after the movie, under the **outdir** folder. In our example this would be

D:\analysis\n2_2010_06_01__08_21_38__1\

All output file names are also prefixed with the movie name. This enables them to later be copied elsewhere without losing their identity or clobbering each other. For example, you can copy all the **wave.bmp** files to a single folder and still know which belongs to which movie.

The files are

- **params.txt** – a copy of the parameters used in the analysis, for the record. This file can also be used as the parameters file for a new analysis, to check that the results are reproduced.
- **wxs.txt** – output concerning worm movement metrics, as described below.
- **wxo.txt** – output concerning worm shape metrics, as described below.
- **wxd.txt** – a dump of the raw data used to create wxo.txt.
- **spd.txt** – output giving the speed in each frame.
- **dsp.txt** – output giving displacement in successive segments of length **displen**.
- **ang.txt** – output giving angles in successive segments of length **anglen**.
- **eggs.txt** – output listing of eggs that were found, if the **eggs** parameter was set.
- **track.bmp** – a picture of the worm's track throughout the movie, as described below.
- **wave.bmp** – a picture mapping the worm's curvature over time, as described below.
- **skel.csv** – a record of the skeleton points found in each frame of the movie. This enables further analysis of the movement and posture without repeating all the image-processing part of the analysis. The format is a line for each point, with the *X* and *Y* coordinates separated by a comma.

The wxs.txt File

This file contains data about the worm's movement throughput the movie.

The first part of the file is a listing of the movement segments identified in the movie. As explained above, the program works by first dividing the movie into short segments of length **seglen** seconds (this is a parameter; the default is 0.5). It then classifies these segments into five types:

- *Forward movement*. This is identified by computing the distance from the worm's midpoint to its head in the first frame, and comparing it with the distance from the worm's midpoint in the *last* frame to the head in the first frame. If the midpoint moved towards the head, this is forward movement.
- *Backward movement*: if the midpoint moved away from the head.

- *Pause*. This is defined as movement slower than **pausespdlim**, which is a parameter (default of 0.05 mm/sec). Note that movement is measured at the midpoint. Thus the head can perform foraging movements but the segment will still be classified as a pause. This can be verified by looking at the midpoint of the skeleton, which is specially marked in the output frames.
- *Omega*. This label is used if three shape-related conditions are met:
  - There are only two cutpoints, i.e. the worm's skeleton does not intersect the straight line from its head to its tail, and
  - Its amplitude (the maximal distance from the head-tail line to any point on the skeleton) is at least 33% of the skeleton length, and
  - The average distance of skeleton points from the head-tail line is at least half the maximal amplitude.

  Note that "omega" is contagious – if any frame in the segment is an omega, the whole segment is classified as omega.
- *Loop*. This is used when the skeleton is misformed and doubles on to itself. "Loop" is even more contagious than omega; it spreads to any adjacent frames that are "inconsistent", meaning that their skeleton length varies by more than 9% from the median. All of them are considered a single segment.

After labeling the short segments, the program attempts to unify adjacent segments into longer segments. This is done according to the following rules:
- Adjacent movement segments are unified if they are both in the same direction (that is, both are forward or both are backward).
- Adjacent pause segments are unified.
- Adjacent omega segments are unified.

After the list of movement segments some statistics of the movement are given (average forward speed, average backward speed, percent time moving forward, moving backward, or pausing, etc.). These are self explanatory.

The last metric listed is the **roam ratio** (or, rather, a better name may be the **dwell ratio**). This attempts to quantify the degree to which the worm moved consistently in the same direction and covered some distance, as opposed to moving back and forth in the same area or pausing a lot. It is calculated by the following formula:

$$\text{roam\_ratio} = 1 - \frac{\text{end\_to\_end\_dist}}{\text{accum\_dist}}$$

where
- **end_to_end_dist** is the distance along a straight line from the midpoint in the first frame of the movie to the midpoint in the last frame of the movie.
- **accum_dist** is the cumulative distance covered in all the movement segments in the movie (that is, in all segments of length **seglen,** including those labeled as a pause, *before* adjacent similar segments are unified). The distance in each segment is measured along a straight line from the skeleton midpoint in the first frame of the segment to the skeleton midpoint in the last frame of the segment, and these are summed to get the cumulative distance.

Thus if the end_to_end_dist is very small relative to the accum_dist, meaning that the worm moved a lot but essentially stayed in the same place, the roam_ratio will be close to 1. If the worm moved consistently and managed to cover considerable distance, the end_to_end_dist will be close to the accum_dist, and the roam_ratio will be low. Thus the roam ratio actually measures the degree to which the worm tends to dwell in the same place. However, it will never reach 0 because accum_dist accounts for the sinusoidal pattern of the midpoint's movement to some degree, and will always be bigger than the end_to_end_dist. Somewhat counterintuitively, the roam ratio will also be low for worms that hardly move at all, because then both the end_to_end_dist and the accum_dist are small.

Note that the roam ratio depends on the movie length, and it is dangerous to compare roam ratios from movies with different lengths (which reflect movements for different durations).

The wxo.txt File

This file contains data about the worm's shape and how it changed throughout the movie.

A slew of metrics are evaluated. Each is measured separately for the different types of movement (moving forward, moving backward, pausing, or looping to form an omega shape) that were defined above, and also for the whole movie together. Movement is measured at the worm's midpoint, so foraging movements of the head do not count. Pauses are defined to be movement of up to **pausespdlim** mm/sec, which is a definable parameter, as measured over a duration of **seglen** (another parameter).

For each metric/movement combination, the program calculates the average of the metric over those segments of the movie that exhibit this type of movement. It also characterizes the distribution of metric values, by providing either data for a boxplot or deciles. If the parameter **distbox** is set, the data provided will include the minimum value, the 5^{th}, 25^{th}, 50^{th}, 75^{th}, and 95^{th} percentiles, and the maximum. Otherwise deciles will be given, meaning the minimum value, the 10^{th}, 20^{th}, 30^{th}, 40^{th}, …, 90^{th} percentiles, and the maximum.

The metrics are the following.

numsklpts is the number of pixels in the skeleton. It thus depends on the resolution of the movie.

len is the length of the worm in pixels. The difference from **numsklpts** is that **len** takes into account the actual distances involved, and doesn't just count pixels (pixels along a diagonal represent a larger distance than pixels along a horizontal line). It is calculated as follows:

- Divide the skeleton into 20 parts with the same number of pixels in each.

- Calculate the length of each part as the distance from the first pixel to the last one, using Pythagoras's theorem, and sum these distances
- If any pixels were left over in the first step, add this number of pixels.

cutpnums is the number of cutpoints. This is the number of times the worm's skeleton cuts a straight line from its nose to its tail, including the end points. As pixels are discrete, "cuts" means a distance of less than a pixel. A sequence of 3 or more skeleton points at a distance of more than 1.4 pixels is required between successive cutpoints.

avgamp is the average amplitude, in pixels. This is the average distance of the points on the skeleton from the straight line connecting the nose to the tail.

ampsym is a metric for symmetry that is related to **avgamp**. It is the average of the *signed* amplitudes of skeleton points, i.e. their distances from the straight line connecting the head to the tail where a distance to one side of this line is considered positive and a distance to the other side negative. A value near 0 implies symmetry.

maxampL and **maxampR** are the maximal amplitudes to the left and to the right. This is the maximal distance of any point on the skeleton from the straight line connecting the nose to the tail. "Left" and "right" are relative to the direction from the nose towards the tail.

Navgamp is the average amplitude as above, divided by the skeleton length (**len**).

NmaxampL and **NmaxampR** are the maximal amplitudes to the left and to the right as above, divided by the skeleton length (**len**).

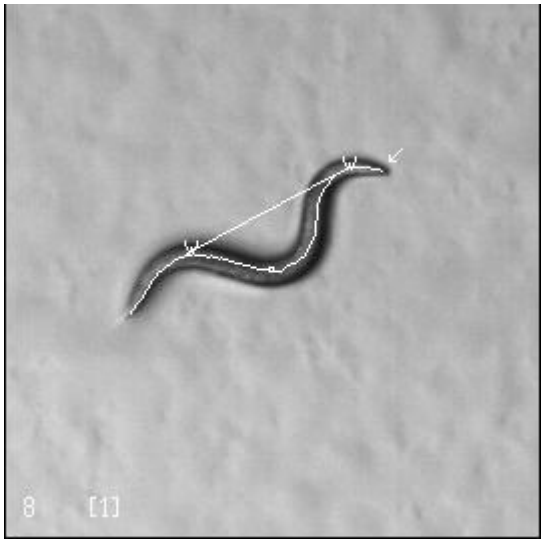
avgangle is the average angle, which is a measure of the curvature along the worm. Angles are measured from the head towards the tail, and are in the range of ±180°, indicating an inclination to the left (positive) or to the right (negative).
Angles are measured by defining a step size that is one twelfth of the skeleton length (in pixels). Consider three skeleton points *a*, *b*, and *c*, that are one step apart. The angle at point *b* is the angle between the line from *a* to *b* and the line from *b* to *c*. Thus if the worm is completely straight, the angle will be 0.
The average angle is the average of the absolute value of the angle as measured for *all* skeleton points (except near the ends). Due to using the absolute value, positive and negative angles (inclination to left or right) do *not* cancel out.

anglesym is another metric symmetry, this time related to **avgangle**. It is based on averaging the actual values, which may be positive or negative depending on whether the worm tends to the left or the right at each point. Thus a value near 0 implies symmetry. If the worm does an omega-like shape all the angles will be in the same direction, leading to a large (positive or negative) value.

maxangL and **maxangR** are the maximal positive and negative angles, respectively.

area is the number of pixels covered by the worm.
thickness is the average "width" of the worm in pixels: it is the area divided by the length of the skeleton (**len**). Both these metrics depend on the resolution.

wavelen is the wavelength of the worm's shape, measured in pixels. It is only calculated in frames where the shape is roughly sinusoidal; if not, it is given as an undefined value. To determine the shape, the program divides the skeleton into two parts at the midpoint. It then looks for the maximal and minimal curvature angle in both parts. If there are two such maxima or two such minima, they are considered as extremum points of the waveform and the distance between them is the wavelength.



Nwavelen – a normalized version of the wavelength, i.e. **wavelen** / **len**.

freq – frequency of worm undulations. This is calculated as the speed of propagation divided by the wavelength, so is undefined if the wavelength was undefined. Speed is calculated based on the movement of the center of mass of the binary silhouette of the worm from the previous frame to this one. The whole thing is only done for frames that are consistent.

disp is the displacement, in pixel units, of the midpoint, over a time of **dispLen** seconds. Unlike other parameters, this is not measured for every frame, but rather once at the beginning of each such time interval.

straightfs is an attempt to quantify how straight the worm is. It is a normalized metric in the range [0..1], with values above 0.9 indicating that the worm is "straight".
The original metric was calculated as $1-\sqrt{(stamp^2 + stang^2)}$, where stamp = max(0.707, 1.5***maxamp**), and stang = max(0.707, **avgangle**/40). The limit to 0.707 is designed to limit each term to ½ when squared, so that their sum will be limited to 1.
Currently this metric is calculated in a simpler manner, which divides the distance from the nose to the tail by the actual length of the worm (**len**). A value near 1 thus indicates straightness, whereas a value near 0 indicates a loop.

sinusfs is an attempt to quantify how "sinus-like" the worm is, but actually it is more of a metric of symmetry. It is also normalized to [0..1], with values above 0.8 considered "sinus like". It uses the symmetry metrics defined above, with the notion that symmetry implies sinusoidal shape. It is calculated as $1-\sqrt{(sinamp^2 + sinang^2)}/2$, where sinamp = **ampsym** / **avgamp**, and sinang = **anglesym** / **avgangle**. Thus if the amplitudes and angles tend to cancel out, we will get a value near 1. If they do not, the value will be low.

The wxd.txt File

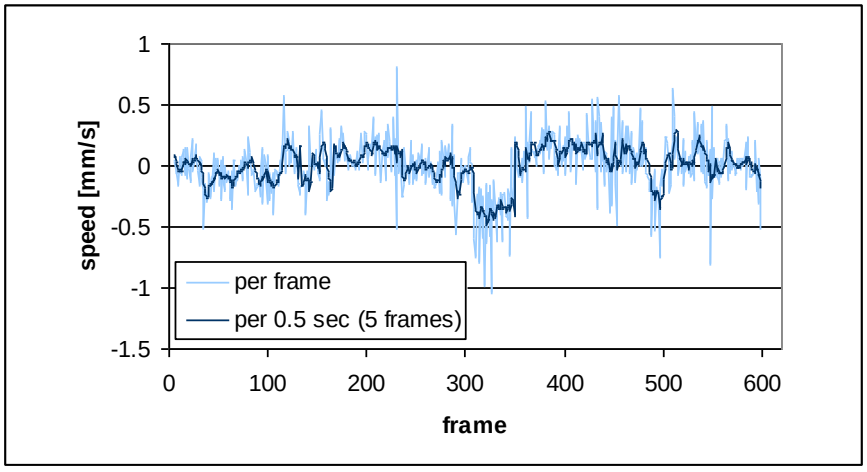
This file contains a dump of all the above data fields for every frame of the movie. The fields are separated by commas. This may be useful for further analysis.

The spd.txt File

This file provides data about the momentary speed of the worm in each frame. Positive values indicate forward movement, and negative values indicate backwards movement. The classification of movement segments is also indicated.

Two values are given for each frame. The first is the speed as calculated between this frame and the previous frame. The second is the speed as calculated between this frame and the frame **seglen** time ago (for example, for movies captured at 10 frames per second, and a parameter **seglen** of 0.5 seconds, this would be 5 frames back). In either case, the speed is calculated as the distance traveled by the midpoint divided by the time span. The units are millimeter per second, and depend on correct setting of the **mmpix** parameter.

The data is tab-delimited. It can be loaded into Excel and plotted (or plotted using some other program). A typical output is as follows:



The dsp.txt File

This file provides data about the worm's displacement in adjacent, non-overlapping segments of duration **displen** seconds. Each line has two values: the frame number and the displacement from the previous one. The units are mm.

Displacements are only calculated between consistent frames. If the desired one or the one **displen** away are inconsistent, the following frames are used. If a duration of **displen**/2 is scanned and consistent frames are not found, this point is skipped. Thus given data is never more than **displen**/2 away from the specified points.

The ang.txt File

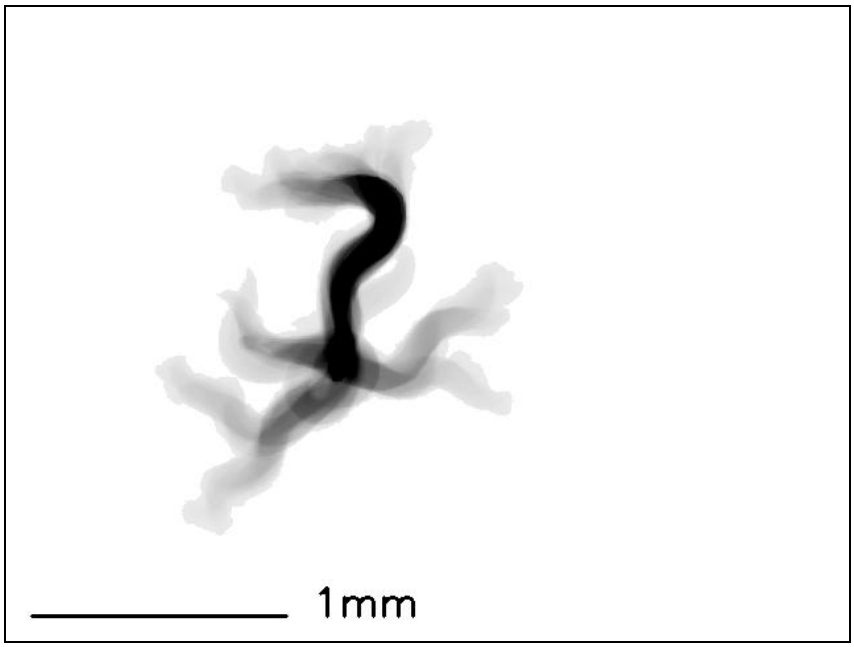
This file provides data about the worm track large-scale curvature, to enable the identification and quantification of situations where the worm seems to be moving consistently in large circles. Given three points on the track that are **anglen** apart, the value given is the angle between the continuation of the first segment and the second segment. Like displacements, this is calculated once per **anglen** interval.

In more detail, the calculation issubject to the following:

1. The frames at the three points must be consistent and not ignored.
2. The distances between them must be at least 0.9 **anglen**.
3. The average speed in each of the two segments must be above the **pausespdlim** threshold. This average is calculated as the end-to-end distance traveled in the segment divided by its exact duration. Thus angles are not calculated during pauses.

The track.bmp File

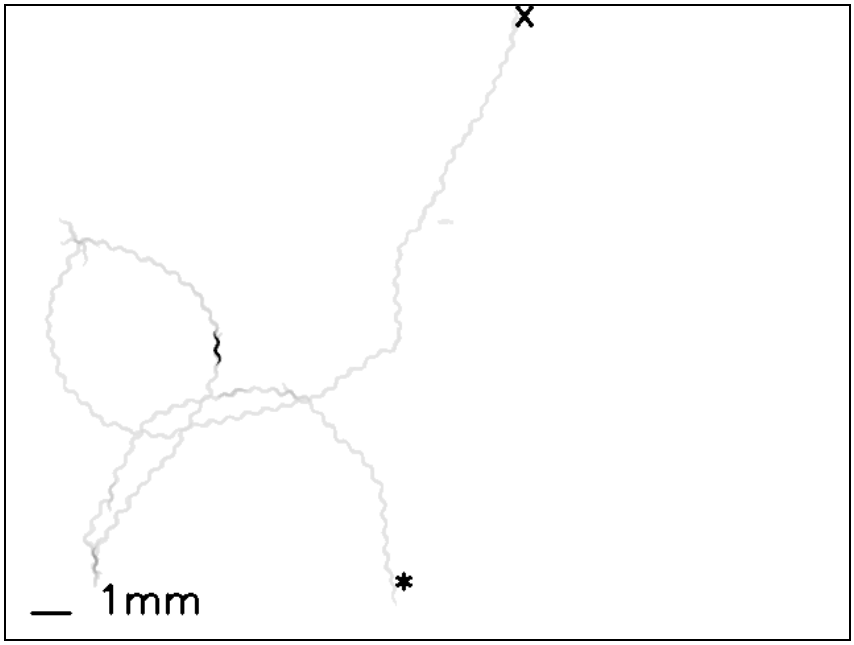
This picture gives a visual summary of the worm's movement throughput the movie (or rather, in the segment defined by **fframe** and **lframe**). A typical picture (of a worm that mostly moved back and forth in the same place) looks like this:



The shading shows the worm's movements. Areas where the worm paused for a long time, or visited repeatedly, will have a darker shading. Areas that were only passed through once will have a light shading.

The degree of shading is controlled by the **shade** parameter. If the worm moves all the time the default value might be too low, and you might want to raise it to make the track darker. If it stays in the same area the track might become saturated, and you might need to reduce the shading so as not to lose detail.

Another example, where the worm moved quite a lot, looks like the following. Note that the scale is much smaller in order to accommodate all the movement:



If the **star** parameter is set to 1, an asterisk will mark the beginning of the track, and an x will mark its end. This is useful then the worm moves a large distance. If **eggs** is set, a circle will mark the location of each egg that was identified.

The scale bar shows one millimeter. Its accuracy depends on the **mmpix** parameter.

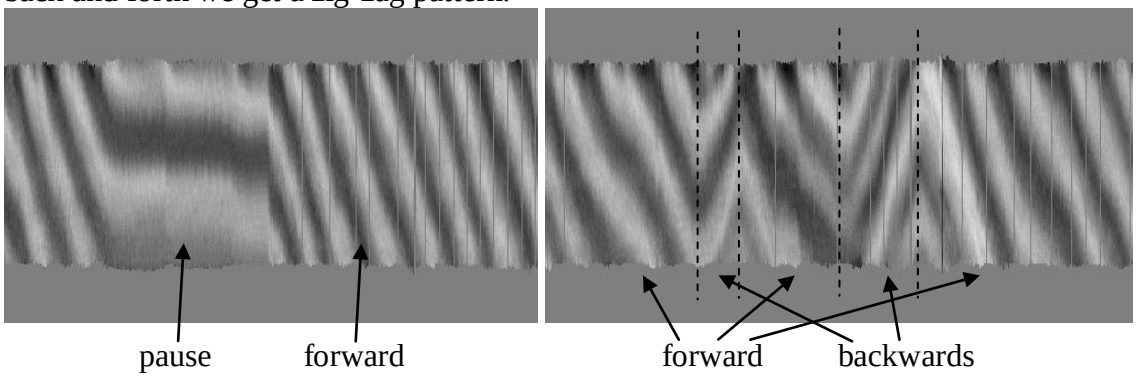
The wave.bmp File

This picture gives a visual summary of the worm's curvature throughout the movie.

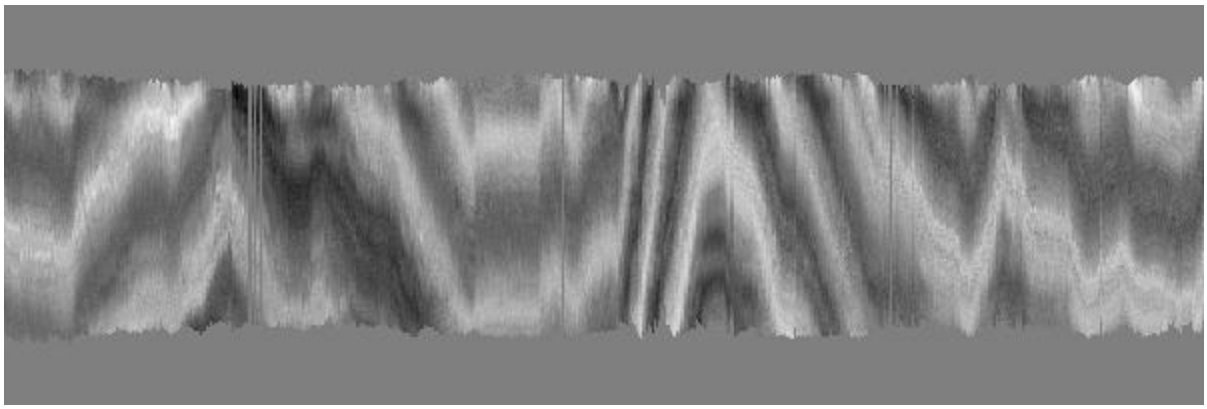
The *X* axis is the frame number – one pixel for each frame. To help keep track, a black line is drawn every 100 frames (pixels), and an indication is given every minute.
The *Y* axis represents the worm's skeleton points. It is governed by the input parameter **maxskellen**, which should be adjusted if different magnifications are used. The midpoint is centered to the middle of the *Y* axis. If you don't get a gray boundary at the top and bottom, increase **maxskellen**. If the boundary is too wide, decrease it.

The gray level represents the curvature, with light shades signifying positive angles (to the left), and dark shades signifying negative angles (to the right). They are normalized to use the full range of gray values from the middle level (meaning angle near 0) to the maximal angle that was measured in the movie.

If the worm moves with a regular sinusoidal movement, we will get relatively straight diagonal lines with alternative light and dark shading. The vertical distance between adjacent lines is the wavelength, and the horizontal distance correlates with speed. The angle of the lines represents direction of body-bend propagation; thus if the worm moves back and forth we get a zig-zag pattern.



If the worm shape and movement are irregular, we will get a much messier picture.



4. Development

This program is developed under Microsoft Visual Studio 2008 in Visual C++. It uses the OpenCV library of image processing functions from Intel.

To set up the development environment, the following steps were taken:

1. Download and install OpenCV version 2.1.
   The download file was OpenCV-2.1.0-win32-vs2008.exe.
   When run, it installs OpenCV in C:\OpenCV2.1.
2. in Microsoft Visual Studio, create a new project of type "Win32 Condole Application".
   The project name used was TLWMA.
3. In the **Tools** menu, open the **Options** item (near the bottom) and navigate to **Projects and Solutions → VC++ Directories**
   In the "Show directories for" selection box on the right, select **Include files**.
   Add the following line to the list:
   C:\OpenCV2.1\include\opencv
   In the "Show directories for" selection box on the right, now select **Library files**.
   Add the following line to the list:
   C:\OpenCV2.1\lib
4. In the **project** menu, open the **TLWMA Properties** item (located at the bottom), and navigate to **Configuration Properties → Linker → Input**
   In the **Additional Dependencies** field, make the entry
   "cv210.lib" "cxcore210.lib" "highgui210.lib"