Testing the Tracking Software

Introduction

It seems we have at our disposal a few open-source tracking software's (or at least plug-ins). We can of course choose the tracking software on a subjective basis ('it looks more precise on the examples I have'). But this is of course wholly insufficient – not to say pretty dodgy. And even if the outcome of the selection is correct ('*Software B is the best*'), it is only a part of the story. It is not enough to know that the software is the best available to us: ***we need to know how good it is!!!***

Engineers indeed want to assess every module involved in the problem they have to deal with. In some cases it is to identify the weak link in the chain - a chain is only as strong as its weakest link – in order to devise a strategy to cope with this weakness. In other cases – when all modules are well-controlled, reliable – it is to quantify the effect every module has on the one following it so that we can estimate the behaviour of the whole chain and also what overall error is made (quality control).

Summarising the relation between outputs – or simply properties of the outputs - and inputs is usually done with ***transfer functions*** – in some very favourable cases a model might be available. These transfer functions can have deterministic components as well as stochastic components – quantifying aspects of the distribution of the output given a certain input.

In the case of software's like ours, there are not too many outputs that can be resumed by deterministic transfer functions – we are not interested by such criteria as average computing time or average computer power since we have access to powerful computers and have quite a lot of time. However, reliability – that is the error made during the tracking - is crucial!

How to Test our Software(s)

About Gold Standards

Ideally we would test our software(s) with real data. But this is not possible unless they have been reliably processed – the outcome to such processing being then used as ***gold standard*** – aka ground truths - that is as a credible approximation of the perfect outcome.
Often with medical data gold standards are produced by asking specialists (humans!!) to ***process the data manually or semi-manually***. In our case this is simply not possible (if you do not believe me, try on a few frames and see how fast you give up…): the amount of tedious work is just too great.
The nicer approach is to ***get hold of data*** that were processed by specialists – again in our case it is not really an option. Fortunately we have one last option – and one that is available to us: ***generate realistic synthetic data!***

Generating Synthetic Data from Real Data

Generating realistic synthetic data is a real hassle – to stay polite.
Simply put, the simulation of the physical process associated with the acquisition of the data – that is the sensor you use - is really complex. To convince yourself, download one of these Magnetic Resonance simulators that are available for download form the Web. And if the process of interest – say the motility of bacteria - needs to be simulated too then it gets far worse.

In our case ***we do not need to simulate with accuracy the motility of the bacteria***. We only need to simulate trajectories of bacteria with roughly similar properties – that is acceptable average velocity, tumbling time etc…
This is easily done with Matlab (see **Tutorial 1** on Motility Assay Analysis).

The tricky part is to ***turn these trajectories into realistic data***.
Fortunately our data are simple: they are movies of dark blobs moving on light backgrounds – the backgrounds having a certain texture. Therefore all we need is to obtain an acceptable approximation

- of the shape of the blobs
- of the intensity of the blobs (ideally intensity distribution)
- of the intensity of the background
- of the texture of the background (noise, etc…)

This is easily done if we have real, representative data – and we have.

Gathering the Information

The first step is a ***data selection*** step

- Select a few representative frames of your movie
- Segment the blobs from the background

At the second step, **analyse the blobs**

- ***Shape Analysis:***
  - Let us assume they are ellipse: compute for all the blobs their small and large axes.
  - More complex approach: store the shapes so they can be used at later stage
- ***Intensity Analysis:***
  - Compute the average and standard deviation for all the blobs
  - More complex approach: store the intensity of all the pixels so they can be used at later stage

Finally, **analyse the background**

- ***Intensity Analysis:***
  - Select a few regions in the background – apply a square window of size that is large enough
  - Compute the average and standard deviation for these windows
  - Even better Model the backgrounds thanks to textural models (is there a Matlab plug-in available???)
  - More complex approach: store of all the windows
- ***Features of Interests in the background***
  - If you see interesting features, analyse them as well

Making a simple Movie

Making a simple movie is easy once you have all the previous information.
First ***bacteria are treated as dots***

- Simulate the trajectories of a handful of dot-like bacteria
- Turn these trajectories into a movie (with the same frame rate as your data and the same image size) as you did in **Tutorial 1**

Now we ***dress up the dots and turn them into blobs***

- Assign a blob shape to each of these bacteria
- Turn the dots of your previous movie into blobs of appropriate size (remember the angle of the movement is given by the large axis of the blob)

Finally we ***assign the intensities***

- Assign for each blob a constant intensity
- Same for the background

More Realistic Movies

The previous movie is of course just a skeleton. To obtain realistic synthetic data, we need to make the blobs and the background more complex.

- The shape of the blobs should vary continuously with time
- If you do not like the elliptical shapes you generate, you may use the shapes you have stored
- The intensity of the blobs should also vary with time
- Ideally the pixels in the blobs would also see their intensity vary
- The background should be dressed up as well with the textural model you have
- You can also spice up your background by incorporating some slopes and interesting features you have identified.

Play with all these options! And generate a few little movies that resemble the experimental data.

While you are at it, also generate data that are more and less challenging than your experimental data (it is always instructive to know how software(s) fare according to the quality of the data). To make your data more or less challenging you can play on:

- The ratio of the intensity of the blob/the background (a version of SNR)
- The shape and size of the blobs
- The noise/texture in the background and on the blobs

Assessing the Reliability of the Software

We have realistic synthetic data – for which the trajectories of the bacteria they contain are known perfectly. We can now test the software(s).
For all your synthetic movies:

- Process them with your software(s)
- If your software(s) depend on a few parameters, do the operations for a reasonable range of parameters

Comparison is achieved by comparing the outcome of these computations to the ground truths - the ideal outcome to a tracking software - we have for each synthetic movie. Global comparison is not possible. Instead a set of well-chosen metrics measure the distance between ground truths and the actual outcomes to your software(s). There are of course many metrics you can use. I would suggest

- Metrics regarding the spatial position of the bacteria
- Metrics regarding the shapes and sizes of the tracked bacteria
- Finally metrics quantifying how well the software(s) assign bacteria to the correct trajectory – the tracking proper.

That's is it! We now have good grounds to make our choice of software. Furthermore we have gathered valuable, quantitative data on what sort of errors are committed during the tracking phase. Such data will be precious for the data analysis stage.