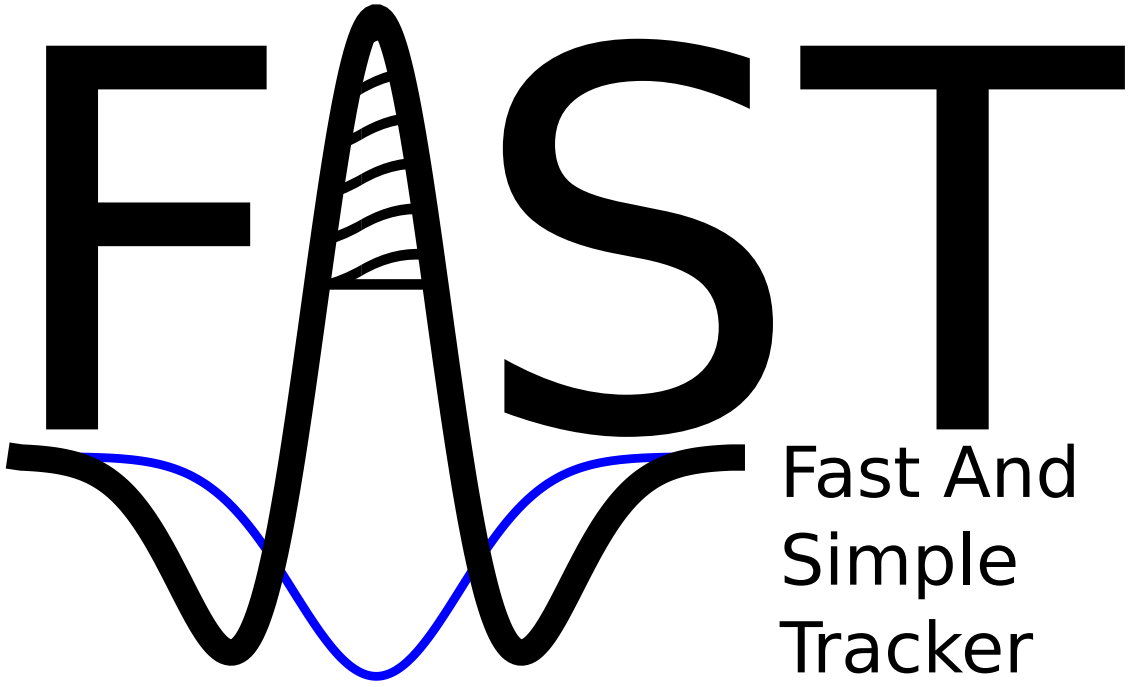

FAST
Fast And
Simple
Tracker

FAST (Fast And Simple Tracker): a short documentation

Thomas Combriat

Installation

Installation in Linux

The simplest way to use FAST on Linux is to compile it from sources:

- Download or clone the code from GitHub: https://github.com/tomcombriat/FAST
- Install the dependencies: OpenCV 3.0 developments packages,[1] CMake and ffmpeg.
- In the "FAST" folder execute:

```
cmake .
make
```

You can now execute FAST running:

```
./FAST input_video output_text_file -karg
```

Where `input_video` is the path to the video you want to analyse, `output_text_file` will be the path to the file where the tracks will be stored, and `-karg` can be `-test` if you want to execute the program in test mode, `-low-ram` if you want FAST to be less RAM-comsuptive, `-inv` if you are trying to track bright particles on a black background and `-no-BG` if you do not want any background to be used.

Installation in Windows

You can download a compiled version of FAST for Windows (64 bits) on Github.

After extracting the archive, you can execute FAST just clicking on FAST.exe. Unlike its Linux counterpart, the Windows version of FAST will ask you for the input and output files on start[2].

Use

Detection

FAST uses a LOG[3] (Laplacian of Gaussian) filter to find particles. This filter needs a few arguments:

- the `blur size` for Gaussian blur, this number should be positive and odd. If t is your blur size, the filter will be more sensible to blobs of size $\sqrt{2t}$.
- the `Laplacian kernel`: define the kernel used to compute the Laplacian[4]. By default you can put this value to 3. An higher value (typically 5) can be selected if your movie if really noisy.

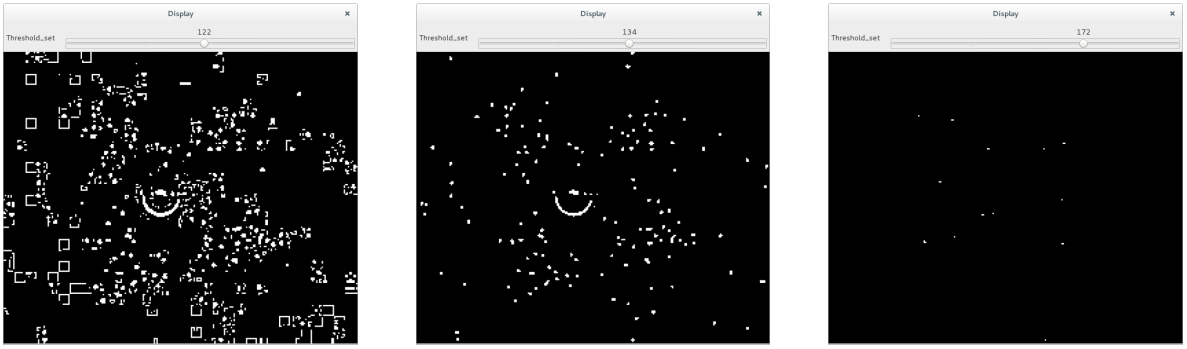
[1] as an example, on Debian these are the packages `libopencv-core-dev`

[2] NB: when asked to give the input video by the program, you can simply drag and release the video inside the command prompt

[3] https://en.wikipedia.org/wiki/Blob_detection#The_Laplacian_of_Gaussian

[4] https://en.wikipedia.org/wiki/Discrete_Laplace_operator

Figure 1: Set the threshold



From left to right: threshold too low, good threshold, threshold too high.

- the `search radius` will be the maximum distance where the tracker will look from a particle position on frame N to find its match on frame $N + 1$ (see Section Linking).

You will also be asked by the program to set a threshold of detection, this threshold has to be high enough so your particles will be displayed white and the background dark. A criteria on the size of the particles can also be set.

NB: for the time being, the program detects black particles on a white background. If you are in the other situation you can easily inverse your movie using a image analysis software.

Linking

Linking particles can be done using two different algorithms explained in figure 2.

By default, the program uses the *nearest neighbor* algorithm. This algorithm is useful if the particles have a random motion or when the displacement of a particle between two time steps is actually small compared to the mean distance between particles.

The *predictive* algorithm uses a linear extrapolation to predict the position of a particle at the next step. If the motion of the particle not deterministic this algorithm will be able to track particles for a longest time.

The `gap closing` parameter allow you to specify the program the number of frames during which it can lose a particle and still keeping the track active. It can be very useful if some particles disappear for only a few frames.

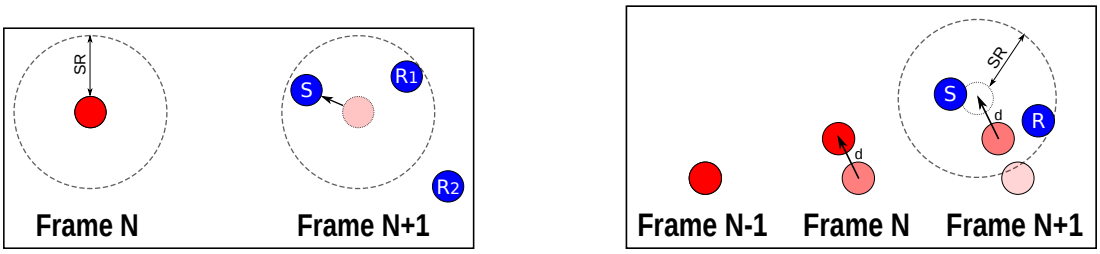
If you are tracking particles which a globally moving in the same direction and at the same speed (for example for in flow measurments) you can speficy this flow velocity for better results. Even if you are doing so, the resulting tracks will be given in the initial movie referential. If you choose to remove a flow velocity, you will have to specify its x component and its y component in pixels/frame, float numbers allowed.

Modes

Test mode (`-test`)

When executing in test mode, FAST will play the whole movie showing the found particles in the display window.

Figure 2: Linking algorithms



Left: nearest neighbor algorithm. A particle from frame N (red) will assigned to the nearest particle of frame $N+1$ (blue labeled "S") in the research perimeter delimited by the search radius (SR). The "R2" labeled particle is rejected because it is outside the research perimeter, and the "R1" particles because it is farer than the "S" particle.
Right: predicitive algorithm. From position of a particle at frame $N-1$ and N, using linear extrapolation, one can expect a position for this particle at frame $N+1$ (dashed line). The selected particle will be the nearest of this expected position within the research perimeter.

Low-ram mode (`-low-ram`)

In this mode, FAST will be a lot less RAM-comsuptive (around 8 times...) and perhaps a little slower. This mode is useful for big datas.

Inversed mode (`-inv`)

Inverse the video (black becomes white and vice-versa...) in order to track bright particles.

No background mode (`-no-BG`)

Do not substract any background. This is useful if displacements are globally slow: in this case, particles will appear on the calculated background, leading to a bad detection.