GP-Tool: A user-friendly graphical interface to apply GP-FBM

GM Oliveira

1 Installation

GP-Tool was tested in Linux, Windows 10 and Mac OSX. It can be easily compiled under these systems given that compiling tools (gcc, visual studio or xcode compatible with standard c++17), OpenGL 4+, CMake and Git are available. To clone the repository along with all submodules use

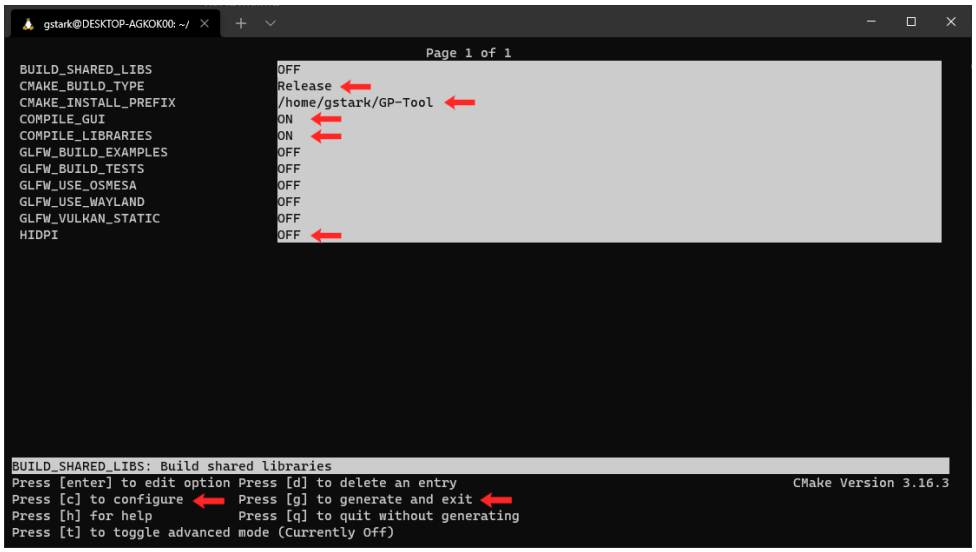
```
git clone --recurse-submodules -j4 https://github.com/guilmont/GP-Tool.git
```

1.1 Linux and Mac OSX

Once cloning is completed, create a build folder and load ccmake as follows

```
gstark@DESKTOP-AGKOK00:~/doc$ ls
GP-Tool
gstark@DESKTOP-AGKOK00:~/doc$ mkdir build
gstark@DESKTOP-AGKOK00:~/doc$ cd build
gstark@DESKTOP-AGKOK00:~/doc/build$ ccmake ../GP-Tool
```

which will open ccmake. Type "c" to configure. You shall see



These are the options pointed in red arrows to be addressed:

1. `CMAKE_BUILD_TYPE`: Conventionally should be set to "Release", unless debugging features are needed;
2. `CMAKE_INSTALL_PREFIX`: Set to the path in which GP-Tool shall be installed;
3. `COMPILE_GUI`: If "ON", the graphical user interface will be compiled and installed to set path;
4. `COMPILE_LIBRAIES`: If "ON" shared libraries from main methods will be generate for the purpose of batching (see section 3);
5. `HIDPI`: If "ON", the size of all widgets will be scaled so compensate for high dpi monitors.

Once all the options are set, press "c" to reconfigure and "g" to generate. For Linux and Mac OSX, type:

```
make -j install
```

everything should be compiled and installed at the set installation prefix.

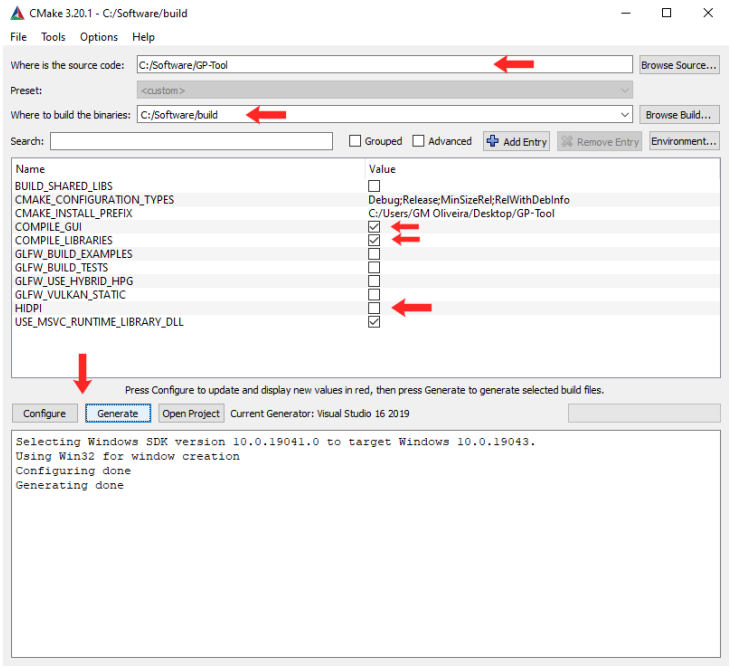
1.2 Windows 10

1.2.1 Binaries

Simply uncompress zip-compressed folder and move it to whichever convenient place you would like and you are ready to go.

1.2.2 Source code

If the user plans to install GP-Tool from source, I would advice to install "cmake-gui.exe". Once you open it, set the source code address and where you would like to build the project. After that, click the button "Configure" at the bottom and specify the generator (I'm using 2019) and platform x64. Click "Finish". You will see something like this



Setup the arrows as follows:

1. `CMAKE_INSTALL_PREFIX`: Set to the path in which GP-Tool shall be installed;
2. `COMPILE_GUI`: If "ON", the graphical user interface will be compiled and installed to set path;
3. `COMPILE_LIBRAIES`: If "ON" shared libraries from main methods will be generate for the purpose of batching (see section 3);
4. `HIDPI`: If "ON", the size of all widgets will be scaled so compensate for high dpi monitors.

Then reconfigure and generate the configuration files by clicking in "Generate". Once configured, open the console and navigate to the build directory and type:

```
cmake.exe --build . --config Release --target install
```

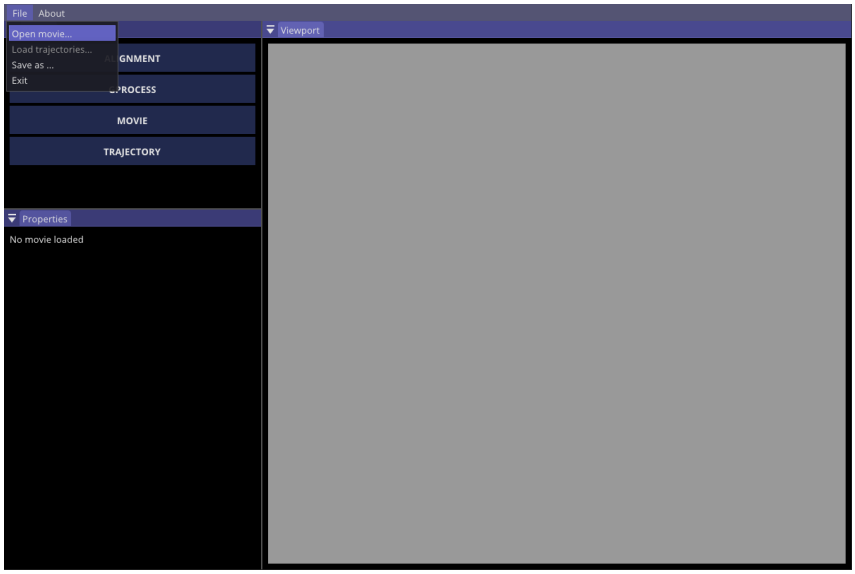
Once the project is built, GP-Tool will be installed at the prefixed location.

2 Plugins

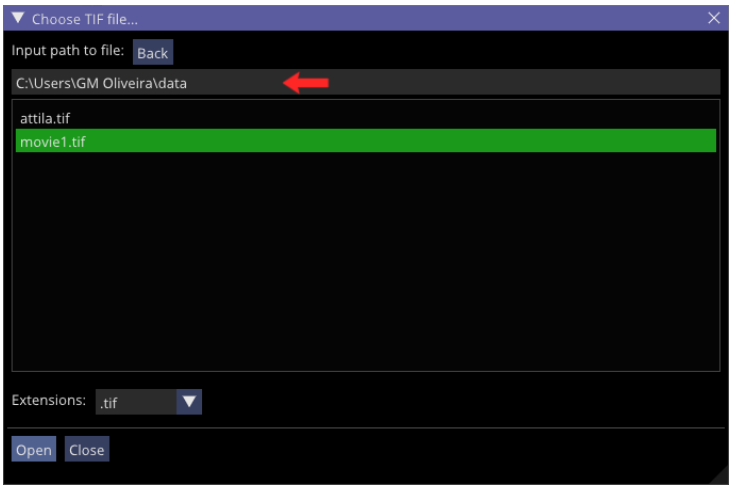
GP-Tool is segmented into several interrelated plugins that work in synchrony. These are responsible by opening movies and handling their metadata, loading tracked trajectories in multiple formats, applying GP-FBM and correcting multi-channeled movies alignment. In the next few sections, we are going to learn how to use each plugin in detail and how to save the final results of the analysis.

2.1 Movie

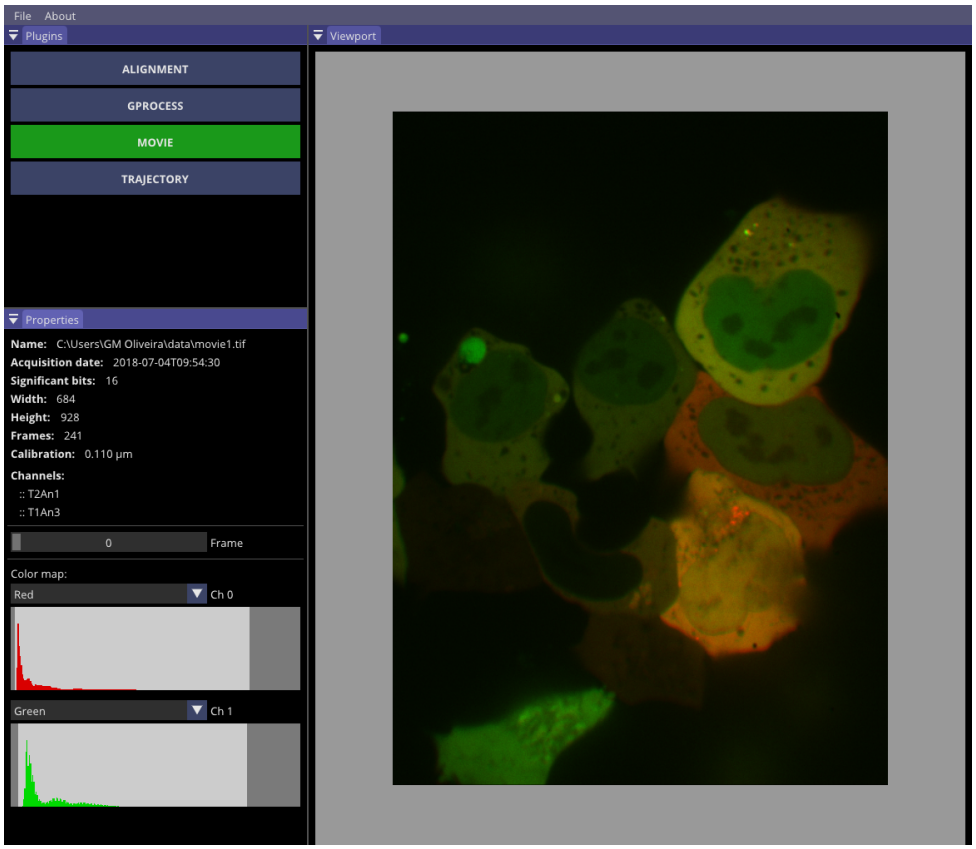
Once GP-Tool is opened, the user can load tiff movies via *Files* > *Open movies...* or by the key combination "Ctrl+O". There are 3 formats of metadata that GP-Tool can parse: basic ImageJ, extended ImageJ and OME. GP-Tool can also handle movies compressed via LZW algorithm.



This selection will open the following file dialog, where the user will be able to navigate until the desired movie. For convenience, we can write or paste the address directly at the path box (red arrow);



Loading the movie might take a few minutes depending on size, compression and operational system in use. Once the movie is finally loaded, the user will be presented with an interactive view port, where we can zoom in or out and move the image around. Similarly, the properties panel will display basic metadata if available. These are movie name(path), acquisition date, number of bits, image height and width, the number of frames, calibration from pixels into pertinent units and the name of each channel. We also have an slider from which we can change the frame display in the view port.

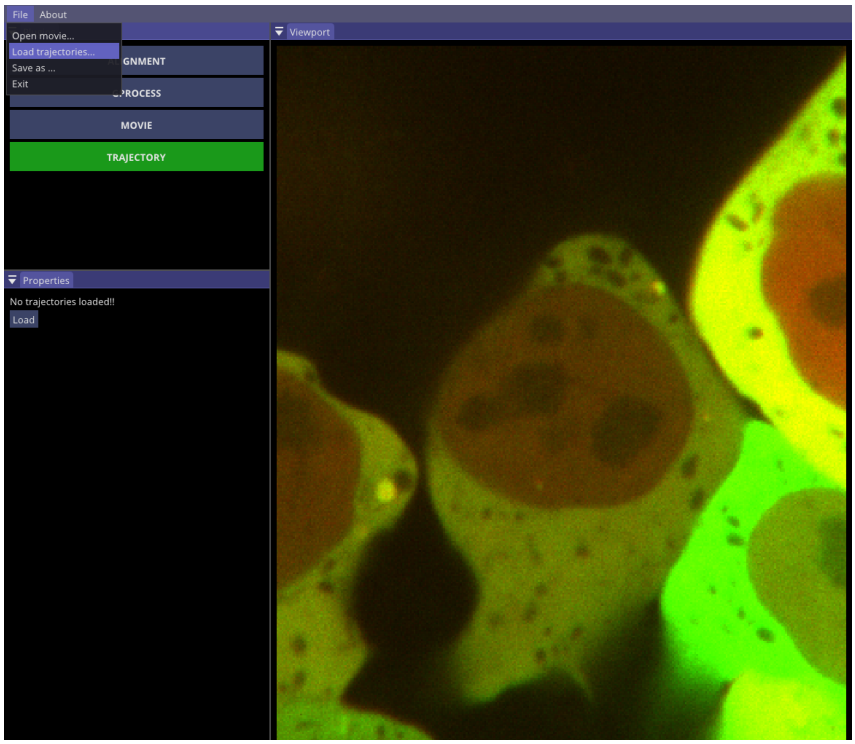


The color map will allow the user to chose which look-up table will be use for each channel from a selection of colors. Differently, we also can opt for none, hence hiding the corresponding channel. Finally, by dragging darker rectangles in either side of histogram widgets, we can set different and channel independent contrast.

2.2 Trajectory

Once the movie was imported, the user will be able to load tracked trajectories into GP-Tool. There are 3 possible ways to do so:

1. Clicking on *File* > *Load trajectories...*;
2. Selecting Trajectory plugin under plugins tab and the "Load" button;
3. Key combination "Ctrl + T".

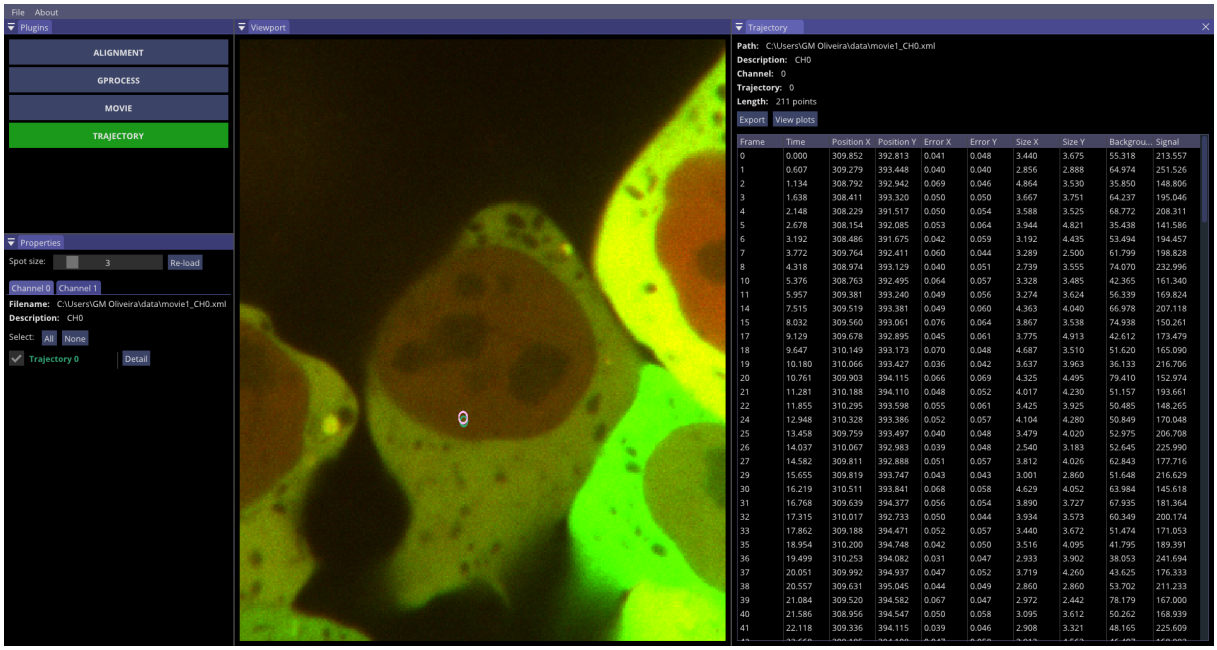


Any of these options will open a dialog where the user will be able to load a track file per channel. At this point, the user should also set an average size expected for spots. This value will be later used to enhance localization and estimate positional errors among other properties for each spot. If, for any reason, the user wants to change this value, there is the possibility to reload trajectories with a different expected size from within the plugin's properties tab.

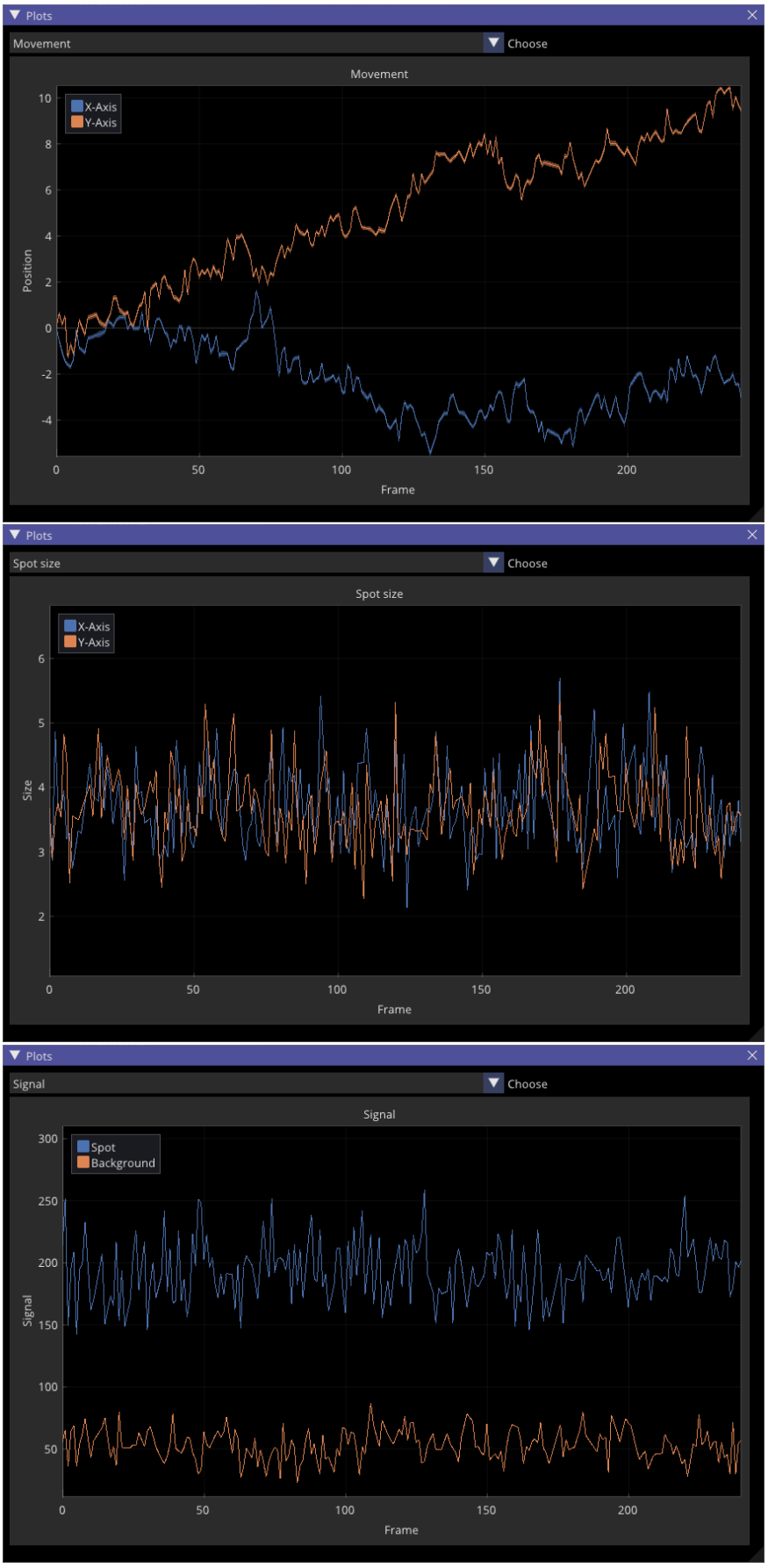
By default, GP-Tool accepts XML track files from ICY software or "handmade" CSV files. The following image shows how these files should resemble.

By clicking on load, GP-Tool will launch a set of algorithms to characterize each spot and remove possible outliers, such as false positives. Upon completion, the properties panel will open a tab for each channel in which all trajectories should have an unique color code assigned. The same color will be used to circulate corresponding spot in the view port if spot is checked.

The user can check further detail about the trajectory described by each spot clicking in the button "detail". This option will open a tab containing information about detected frames, corresponding elapsed times in seconds (if metadata is properly configured), position in X and Y, localization error in X and Y, spot size in X and Y, background and above background signal. All coordinates, errors and sizes are given in pixels.



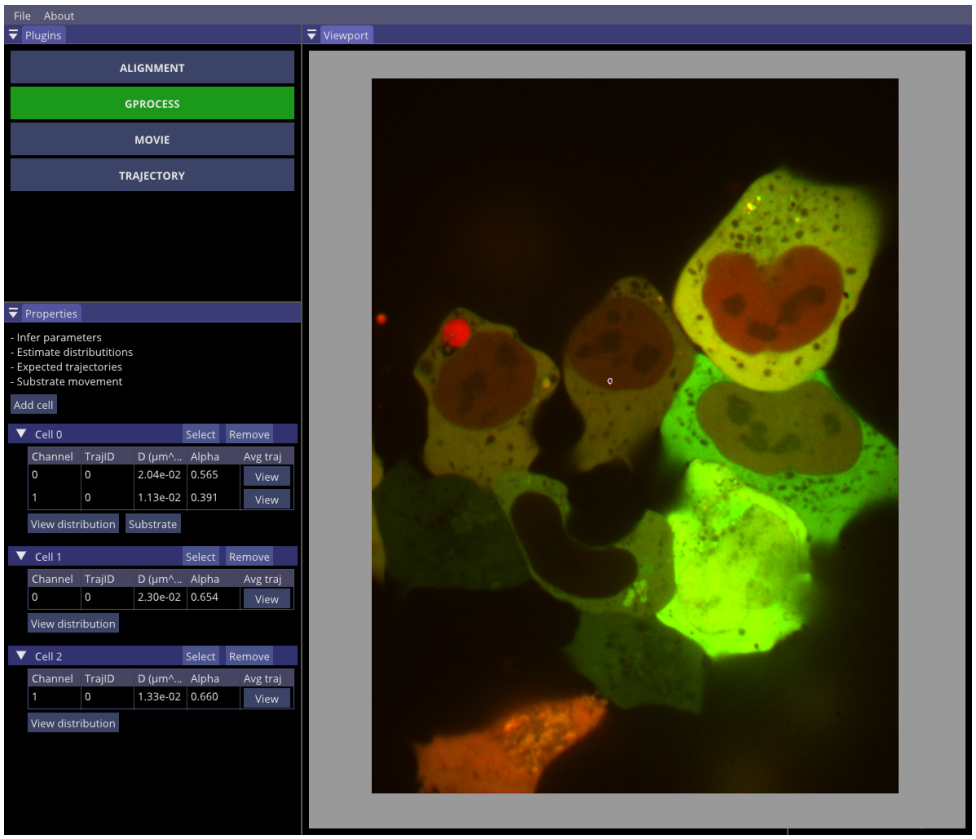
Aiming for a more intuitive analysis, clicking in "View plots" on top of the table, the user will be presented with plots containing information about the spot's position over time with a 95% credible interval, spot size and signal as present next.



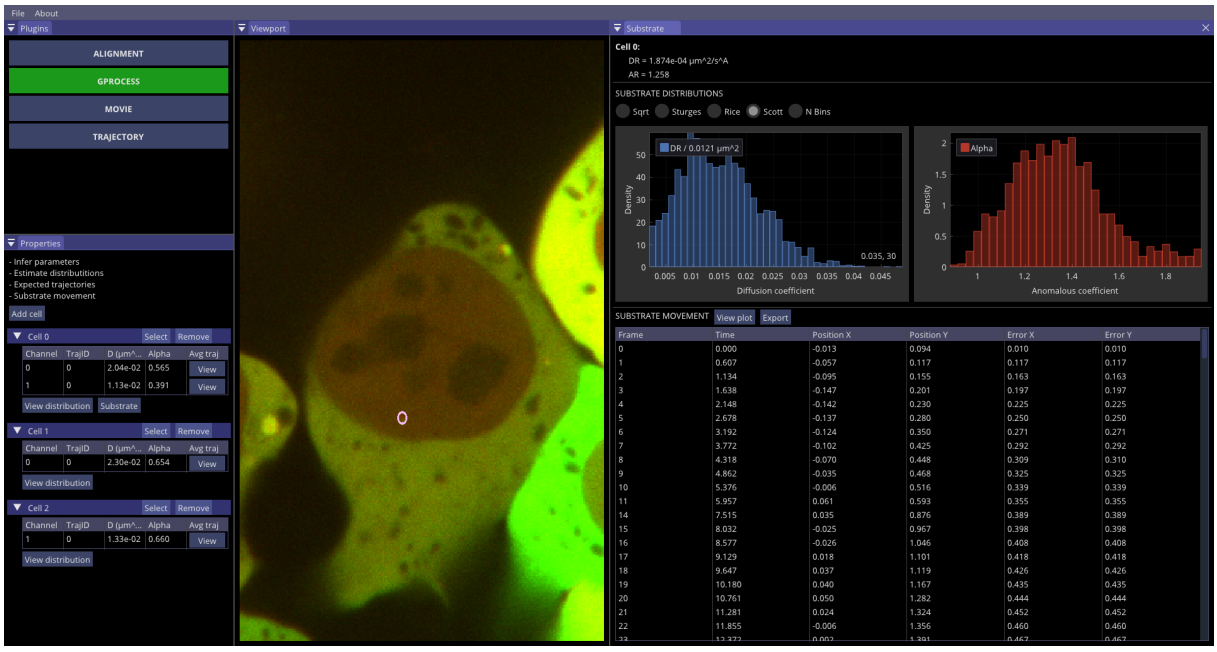
Finally, the user can choose to save this table in a CSV file clicking on the “Export” button, which will open a save dialog.

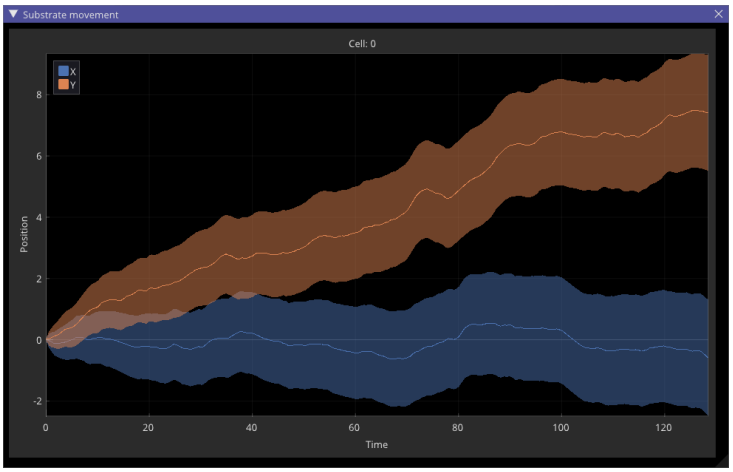
2.3 G-Process

After all trajectories are imported, it is time to determine the diffusion properties of spots. The user has an option to analyze spots individually, by selecting one spot at the time under Trajectory plugin or to group spots. Once grouped, GP-Tool will assume that all selected spots are under a similar context, hence subject to similar background movement. In that sense, we could, for example, select all the spots in the nucleus of a cell and, like so, filter out the cellular movement from the apparent diffusion and anomalous coefficients. Differently, by selecting two or more spots close to fluctuating membrane, the dynamics properties inferred will attempt to filter out said fluctuations. In the next image, we first group both spots for a first analysis and, later, treat them independently. Comparing the results, we can observe how background movement affects inferred values.



Evidently, by clicking on the button "Substrate" we are presented with a series of data estimated for the substrate from selected group of trajectories. In the substrate tab, we have inferred apparent diffusion and anomalous coefficients for the substrate as well as a probability density plot for these parameters. Additionally, GP-FBM will provide an estimation for the trajectory described by the substrate in the form of a table. The user can save this table by clicking in the button "Export" or visualize the data with "View plot", where the trajectory will be displayed with a 95% credible interval.



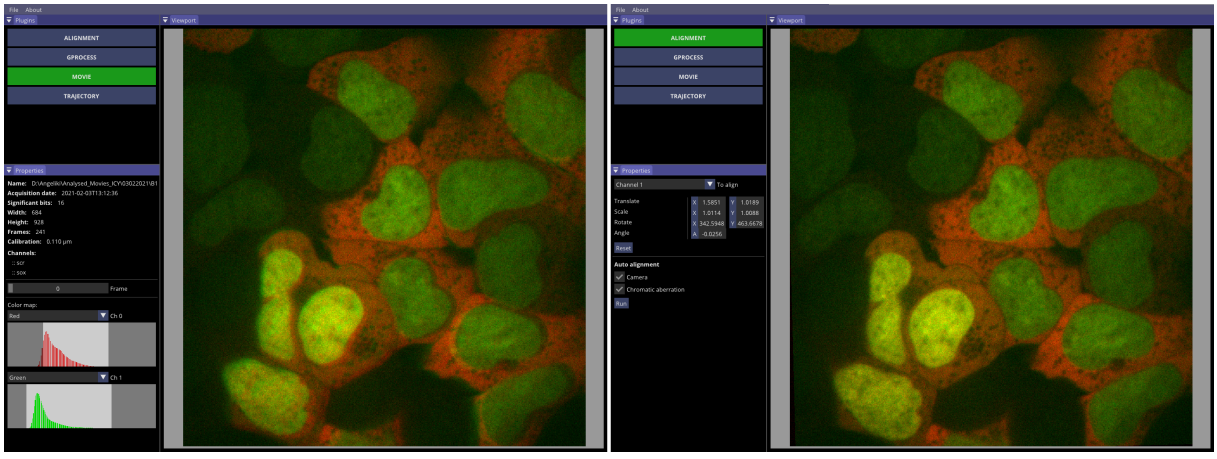


An additional benefit of Gaussian processes, is the possibility to use measured spot locations and estimated dynamics parameters to interpolate the position of spots in time within a credible interval. GP-Tool allows the user observe these curves using button "View" at the column "Average trajectory" in the properties panel. In the following image, points represent measured positions whilst continuous lines and shaded areas correspond to most probable trajectory in time and a 95% credible interval.

Finally, to determine how stable the inference of apparent diffusion and anomalous coefficients are, the user can consult the probability density function of each parameter estimated via Metropolis-Hastings clicking in the button "View distribution". For convenience, several standard binning methods are provided. Differently, the user can manually select any given number of bins.

2.4 Alignment

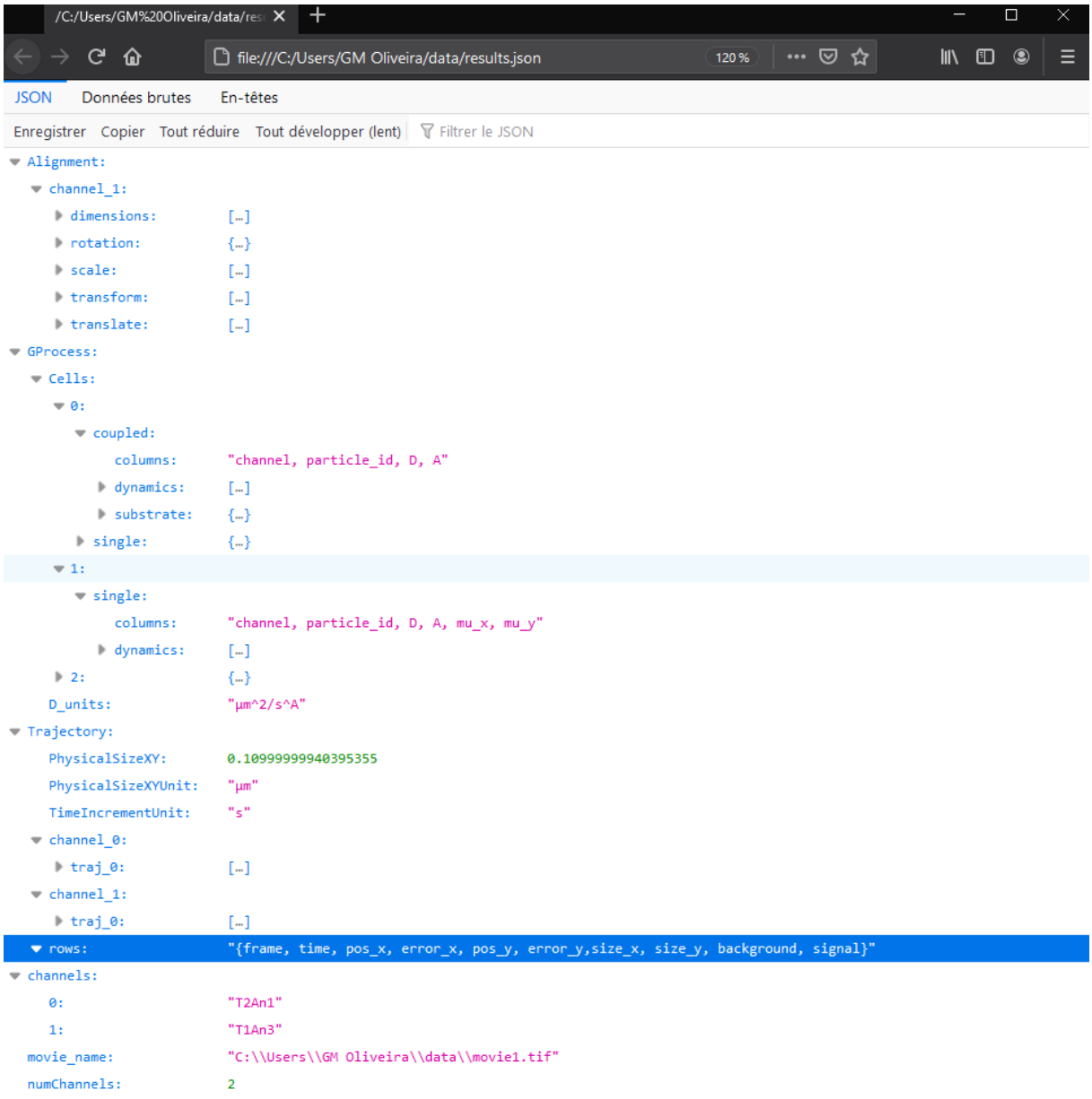
Even though channel alignment is not strictly necessary for inference of diffusion properties as fractional Brownian motion is orientation independent and chromatic aberration is negligible for short trajectories, it becomes relevant once we are interested in the average distance between spots. For that reason, GPTool includes a plugin that will attempt to calculate a matrix based on generic affine transformations that will match common shapes in both channels. In the left most of the following images, we present an movie in which the dual-camera setup was badly calibrated. On the right, we present the results of an automatic alignment. Differently, the user might setup parameters manually.



The plugin provides two types of auto-correction aiming for different types of errors. The first will combine translations and rotations in the attempt to correct for a system with multiple badly calibrated cameras. Evidently, this type of correction is not necessary in the case of a setup with a single camera. On the other hand, any experimental protocol using multiple wavelengths is subjected to chromatic aberration. Its effect is not so relevant at image's center, but it might become non-negligible towards its edges.

2.5 Saving results

Once the analysis is complete, the user can save results by navigating to *File* > *Save as...* or the key combination "Ctrl + S". By its flexibility, "ease-of-use" and wide range of programming languages that can parse it, the JSON format was chosen. The final file will look as in the next image.



In a hierarchical organization, the file will show the original movie name and basic metadata on which the analysis was made. It will present enhanced non-aligned trajectories per channel (the contents of each row are presented) and inferred diffusion parameters. GProcess will always present results determined from individual trajectories, the reason being its usage on position interpolation. In cells (or groups) containing multiple spots, the coupled analysis will be provided along with substrate data. Finally, the alignment plugin will save its own data, that is, image dimensions and optimal translation, scaling and rotation parameters. For convenience, a ready-to-use transformation matrix is also provided. This matrix can be directly used to alignment trajectories if necessary.

3 Batching

In order to facilitate the analysis of hundreds of movies at a time, the core functions available in GP-Tool are wrapped in shared C++ libraries that can be linked to any C++ code. In the next hundred or so lines of code, function "runSample" implements the basic pipeline for analysis. First we load a movie and ICY tracks in xml format using a clever naming convention. Then, we enhance these trajectories, that is, we optimize localization and determine error ranges, sizes and signal. We also keep a few frames from each channel for later alignment. After, we show how to analyze single trajectories in an independent fashion, where no substrate is accounted. Next, we analyze these trajectories by assuming coupling via substrate. Finally we apply camera and chromatic aberration correction.

I don't propose any standard mechanism to save this data as different types of analysis require alternative results to be saved. In my case, for example, I batched all movies recorded in a microscopy session and saved in a single HDF5 file, which is more efficient than JSON for a large amount of data. One might also consider to save a few frames from all movies of said session for alignment, which is beneficial in the case of non-regular noise (blobs of super bright fluorescent material).

```
/***************************/
#include <iostream>
#include <filesystem>
#include <thread>

// GP-Tool libraries
#include <movie.h>
#include <trajectory.h>
#include <gp_fbm.h>
#include <align.h>

void runSample(std::filesystem::path &path)
{
    std::string mov_name = path.string(),
    base = path.parent_path().string() + "/" + path.stem().string();

    std::vector<MatXd> vCH[2];
    std::array<std::string, 2> xml_name = {"_CH0.xml", "_CH1.xml"};

    Movie mov(mov_name);
    const Metadata &meta = mov.getMetadata();
    const uint32_t nChannels = meta.SizeC;

    ////////////////////////////////////////////////////
    // Setting up trajectories
    Trajectory traj(&mov);
    traj.spotSize = 3; // selecting spot size

    for (uint32_t ch = 0; ch < nChannels; ch++)
    {
     // Getting a few frames for alignment
     for (uint32_t k = 0; k < 5; k++)
     vCH[ch].push_back(mov.getImage(ch, k));

     // Importing trajectories
     traj.useICY(base + xml_name[ch], ch);
    }
    // Enhancing tracks
    traj.enhanceTracks();

    ////////////////////////////////////////////////////
    // Determining diffusion parameters
    const MatXd &mat0 = traj.getTrack(0).traj[0];
    const MatXd &mat1 = traj.getTrack(1).traj[0];

    if (mat0.rows() < 50 || mat1.rows() < 50)
    {
```

```
        std::cout << "WARN: Ignored due to short trajectory >> " << mov_name << std::endl;
        return;
    }

    GP_FBM gp(std::vector<MatXd>{mat0, mat1});
    GP_FBM::DA
        *da0 = gp.singleModel(0),
        *da1 = gp.singleModel(1);

    GP_FBM::CDA
        *cda = gp.coupledModel();

    if (cda == nullptr || da0 == nullptr || da1 == nullptr)
    {
        std::cout << "WARN: Ignore because didn't converge >> " << mov_name << std::endl;
        return;
    }

    ////////////////////////////////////////////////////
    // Aligning channels
    Align align(uint32_t(vCH[0].size()), vCH[0].data(), vCH[1].data());

    if (!align.alignCameras())
        std::cout << "WARN: Couldn't align cameras >> " << mov_name << std::endl;

    if (!align.correctAberrations())
        std::cout << "WARN: Couldn't correct chromatic aberrations >> " << mov_name << std::endl;

    // Getting correction values
    const TransformData &trf = align.getTransformData();

    ////////////////////////////////////////////////////
    // Save results as necessary here

} // runSample

static void runParallel(uint32_t tid, char **argv, uint32_t argc, uint32_t nThreads)
{
    for (uint32_t l = tid + 1; l < argc; l += nThreads)
    runSample(std::filesystem::path(argv[l]));
}

int main(int argc, char *argv[])
{
    const uint32_t nThreads = 4;
    std::vector<std::thread> vThr(nThreads);

    for (uint32_t k = 0; k < nThreads; k++)
    vThr[k] = std::thread(runParallel, k, argv, uint32_t(argc), nThreads);

    for (std::thread &thr : vThr)
    thr.join();

    return EXIT_SUCCESS;
}
/*****************************/
```

To compile this code, one can simply use the command line, however, I propose the following CMake script. Notice that my GP-Tool installation was done at the Desktop.

```
##############################
cmake_minimum_required(VERSION 3.0.0)
project(batch VERSION 0.1.0)
```

```
set(CMAKE_BUILD_TYPE Release)
set(CMAKE_CXX_STANDARD 17)
find_package(Threads REQUIRED)

if (WIN32)
    set(GP_INCLUDE "C:/Users/GM Oliveira/Desktop/GP-Tool/include")
    set(GP_LIB "C:/Users/GM Oliveira/Desktop/GP-Tool/lib")
else()
    set(GP_INCLUDE "/home/gstark/GP/include")
    set(GP_LIB "/home/gstark/GP/lib")
endif()

add_executable(batch main.cpp)
target_include_directories(batch PUBLIC ${GP_INCLUDE})
target_link_directories(batch PUBLIC ${GP_LIB})
target_link_libraries(batch PUBLIC  Threads::Threads movie align trajectory gpfbm)
##############################
```

For the sake of efficiency in the case of Windows 10 users, I suggest installing WSL 2. On my personal computer, I got an improvement of three to four times in terms of speed.