Beam Profiling Application – Manual

1. Introduction

1.1. About This Manual

The porpuse of this document is to describe a beam profiling application. This application helps to monitor and characterize laser beams.

The manual guides the user through the installation process and describes the features of the program. It also sketches how to make it compatible with other camera types. The software has been tested with a Ximea xiQ camera and several VRmagic USB cameras. There also exists an interface to uEye cameras, which has been tested with an old version of the program.

1.2. Copyright and License

1.3. Contributors

- **Cristian Groß** christian.gross@mpq.mpg.de

- **Timon Hilker** timon.hilker@mpq.mpg.de
- **Michael Höse** michael.hoese@physik.uni-muenchen.de
- **Konrad Viebahn** kv291@cam.ac.uk

For questions on the source code refer to **Michael Höse.** Questions on the uEye interface should be directed to **Konrad Viebahn.**

1.4. Helpful Hints

Testing was always done with black and white images (b/w) with 8 bits per pixel (mono8 format). For other formats, the program is not guaranteed to work.

2. Installation

2.1. Preparations

2.1.1. Python

Make sure that a current version of Python 2.7 is installed on your computer (64-bit recommended). If not, install a Python distribution that contains the relevant scientific packages (Numpy, Scipy, etc.). For testing Anaconda was used. To install Anaconda go to https://www.continuum.io/downloads. Download and install the suitable version (64-bit recommended).

2.1.2. Pyqtgraph

There are two options to get "pyqtgraph" running on your computer

- If "pyqtgraph" is not yet installed on your computer, go to http://pyqtgraph.org/ and follow the installation instructions
- In case you do not want to install "pyqtgraph" or if any problems occur, you can directly download the source code (as a Zip-folder) or clone into the "pyqtgraph" repository here https://github.com/pyqtgraph/pyqtgraph. The following steps will be explained later.

2.2. Necessary Camera Software

2.2.1. VRmagic USB Cameras

To run the beam profiling software with VRmagic USB cameras, go to https://www.vrmagic.com/de/imaging/downloads/ and download and install the

"VRmUsbCam DevKit". If your Python installation is 32-bit use the 32-bit version, otherwise download the 64-bit version.

2.2.2. Ximea xiQ Cameras

When using Ximea xiQ cameras, go to https://www.ximea.com/support/wiki/apis/APIs to download and install the "XIMEA API Software Package" that suits your system.

2.3. Retrieving Files from GitHub

The most convenient way to get the necessary files to run the beam profiling application is to clone the Git repository from GitHub. Therefore Git has to be installed on your computer. To clone the repository, open the Git Bash and move to the directory, where you want to clone the repository to. Run the command **`git clone https:// github.com/mhoese/beam-cam`** in the Git Bash. Then change to the "mpq" branch by calling **`git checkout mpq`**. Now you should see in the directory you have chosen a new folder "beam-cam". This folder contains the program files. Instead of cloning, you can also just download the files from https://github.com/mhoese/beam-cam.

To run the program with VRmagic USB cameras, you have to copy the "vrmusbcam2.dll" file into the "beam-cam" folder. You find this file in the VRmagic directory (by default in CommonFiles).

When you want to use Ximea xiQ cameras, copy the "xiapi64.dll" (in case your Python platform is 32bit use "xiapi32.dll") to the "beam-cam" folder. This file can be found in the XIMEA directory. The program recognizes if your Python platform is 32bit automatically.

By default, the program loads both of these files, so make sure that both of them are in the "beam-cam" folder. If you only want to use one camera, you will have to specify this in the "BeamProfilingApplication.py" file. For details see the section on how to configurate different camera types.

In case you do not want to install "pyqtgraph", you have to copy the "pyqtgraph" -folder from the repository of "pyqtgraph" (see section 2.1.2.) into the "beam-cam" folder.

To start the program, run "BeamProfilingApplication.py" with Python.

3. Preparations

This chapter specifies how to adapt the program to different camera types. It is also possible to only install the software of one of the two cameras described above. If doing so, some default settings in the source code have to be changed.

3.1. Modifications for Using a Single Camera Type

If only a single camera type (VRmagic **or** Ximea) is used, there is no need to install the software for the other camera type. To run the program, some changes in the source code have to be made. In this part of the source code, comment out the options that do not meet your needs. You find those at the beginning of "BeamProfilingApplication.py".

```
'''!!COMMENT OUT THE OPTIONS THAT YOU DO NOT WANT!!'''


'''Using both camera types (VRmagic and Ximea)'''
cameratypes = ['VRmagic USB','Ximea xiQ']
cameraapinames = ['VRmagicUsbCamAPI','XimeaxiQCamAPI']



# '''Using VRmagic cameras only'''
# cameratypes = ['VRmagic USB']
# cameraapinames = ['VRmagicUsbCamAPI']



# '''Using Ximea xiQ cameras only'''
# cameratypes = ['Ximea xiQ']
# cameraapinames = ['XimeaxiQCamAPI']



# '''Only use demo'''
# cameratypes = []
# cameraapinames = []

'''-----------------------------------------------'''
```

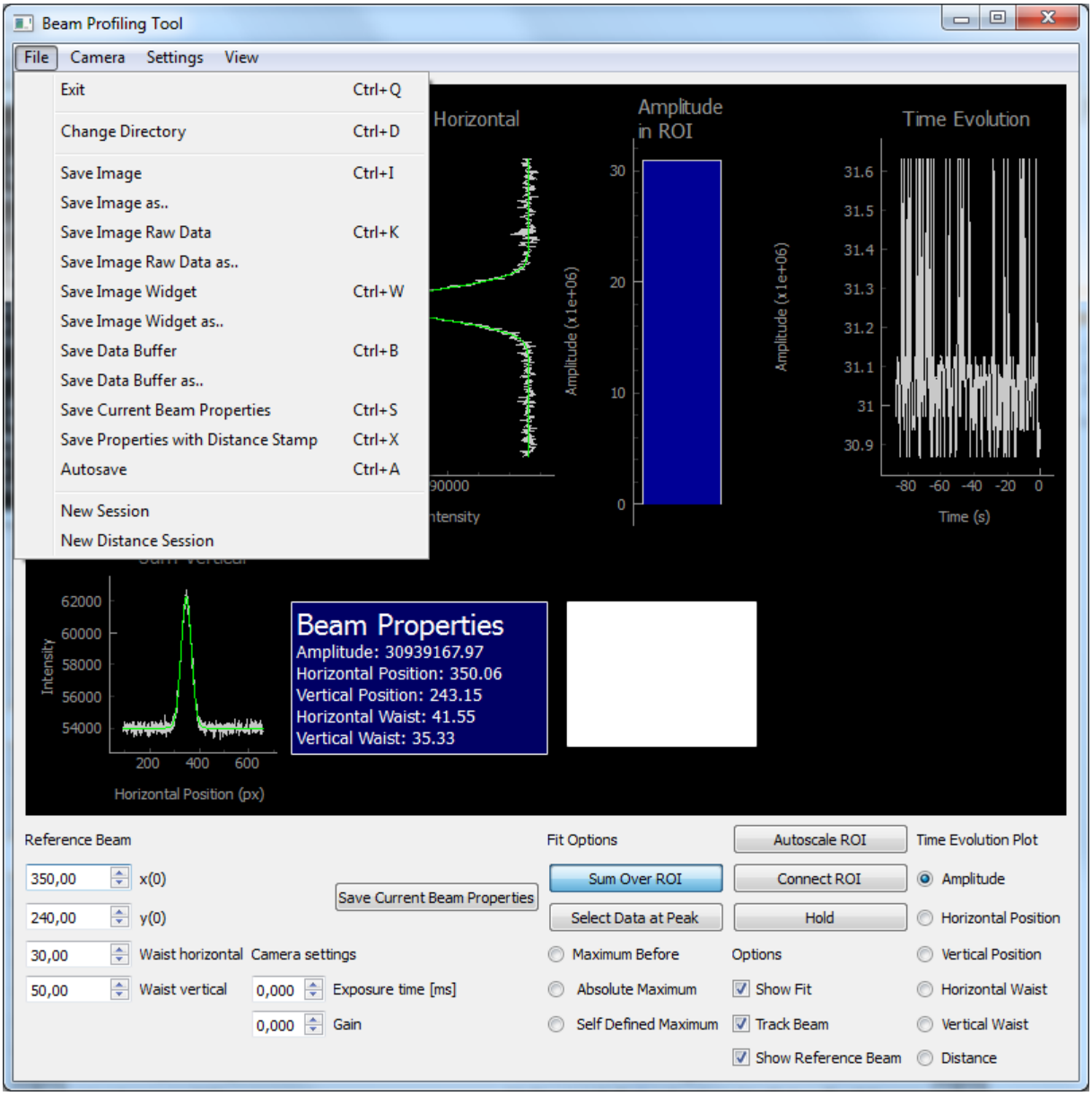
3.2. Modifications for Using Another Camera Type

To use a camera type that is not compatible with any of the available interfaces, a new subclass that extends the methods used in the superclass (contained in “CameraAPI.py”) has to be written. This new interface has to overwrite all the methods specified in the superclass. The datatypes of arguments and the structure of returned values have to match the specifications exactely to ensure a working program. For examples see “XimeaxiQCamAPI.py” or “VRmagicUsbCamAPI.py”. Finally you have to define a new name for the GUI in the list “cameratypes” and the name of the corresponding interface in the list “cameraapinames”. Examples can be seen in section 3.1. These lists can be found at the beginning of the file “BeamProfilingApplication.py”.

4. Functions of the Program

This section gives an overview of the functions of the program. All the options are shortly described.

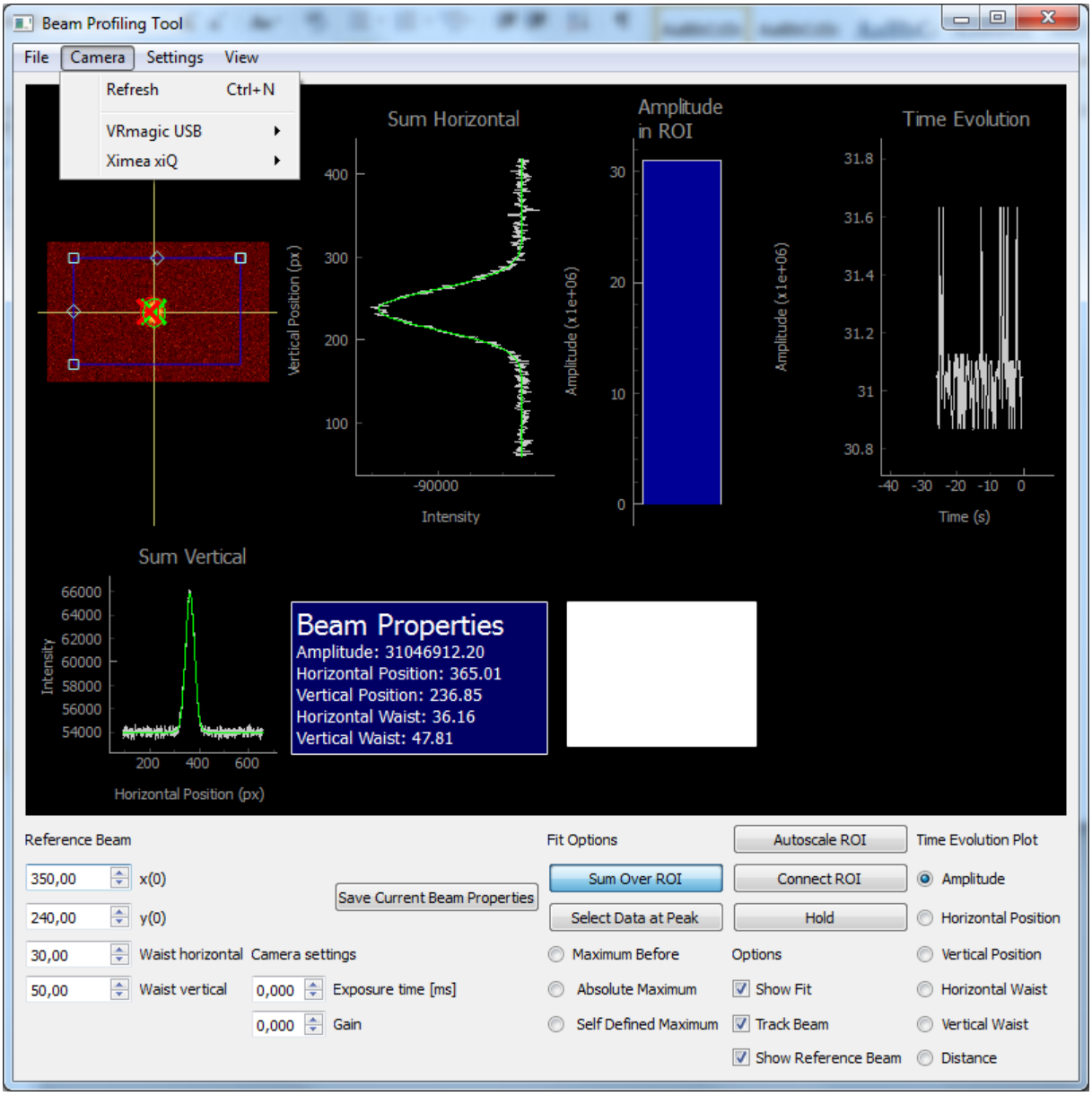
4.1. Menubar – File



- **Exit:** Quit the application
- **Change Directory:** Change the directory where the data or images are save
- **Save Image:** Save the image as .png file
- **Save Image as:** Save the image as .png file with self-specified name
- **Save Image Raw Data:** Save the image raw data as .npy file
- **Save Image Raw Data as:** Save the image raw data as .npy file with self-specified name
- **Save Image Widget:** Save a part of the displayed interface as .png file
- **Save Image Widget as:** Save a part of the displayed interface as .png file Save the current data buffer as .csv file

- **Save Data Buffer:** Save the current data buffer as .csv file
- **Save Data Buffer as:** Save the current data buffer as .csv file Save the current data buffer as .csv file
- **Save Current Beam Properties:** Creates a .csv file in case it not exists yet and ads a line with the current beam properties (also accessible from the "Save Current Beam Properties" button in the GUI)
- **Save Properties with Distance Stamp:** Same as "Save Current Beam Properties", but a distance stamp has to be entered manually (possible applications: measure Gaussian beam).
- **Autosave:** If checked: Creates a .csv file and saves the current beam properties after a time interval that has to be specified when starting the function.
- **New Session:** Opens a new .csv file when "Save Current Beam Properties" is used afterwards.
- **New Distance Session:** Opens a new .csv file when "Save Properties with Distance Stamp" is used afterwards.

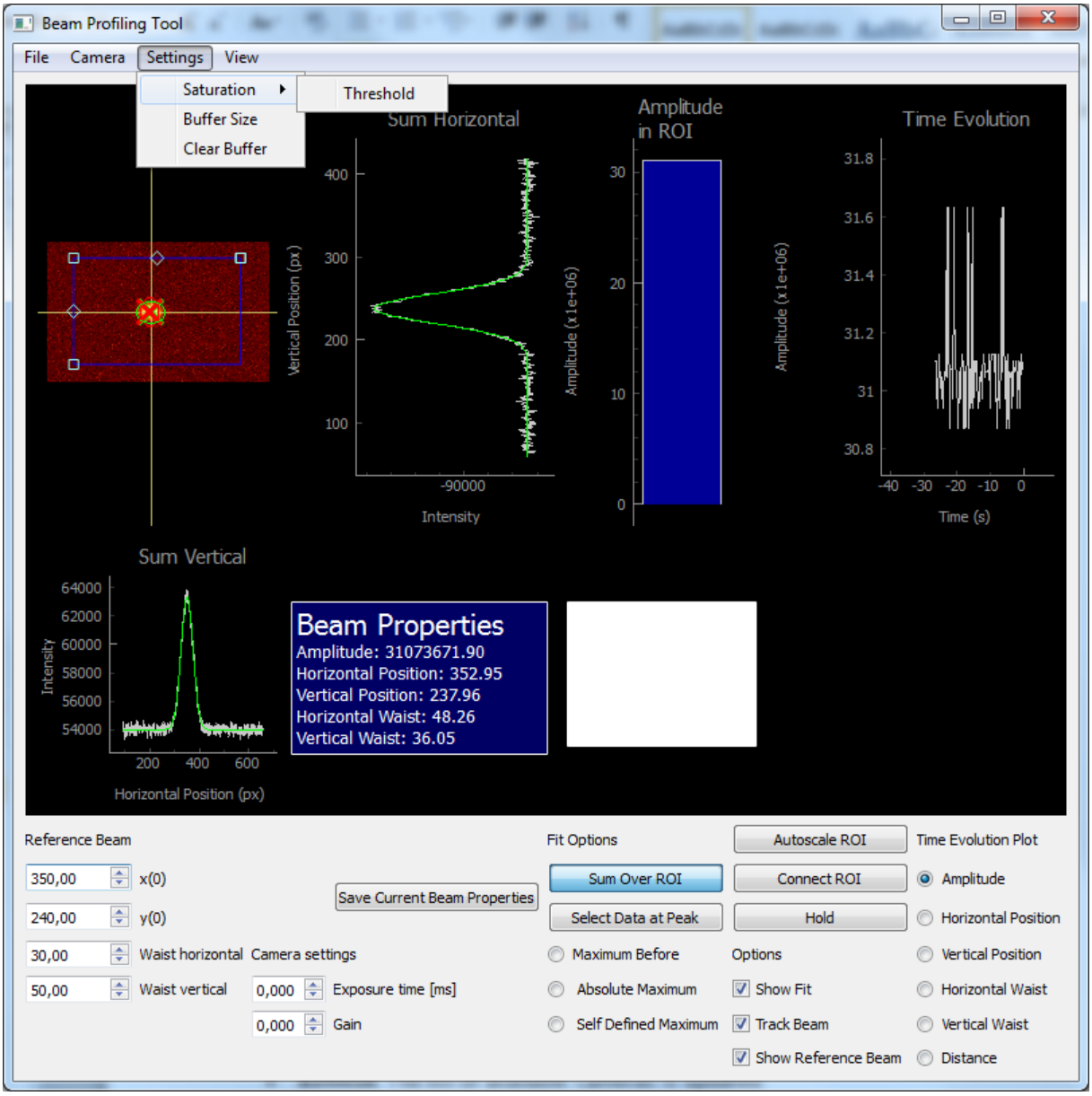
4.2. Menubar – Camera



- **Refresh**: The list of available cameras is updated

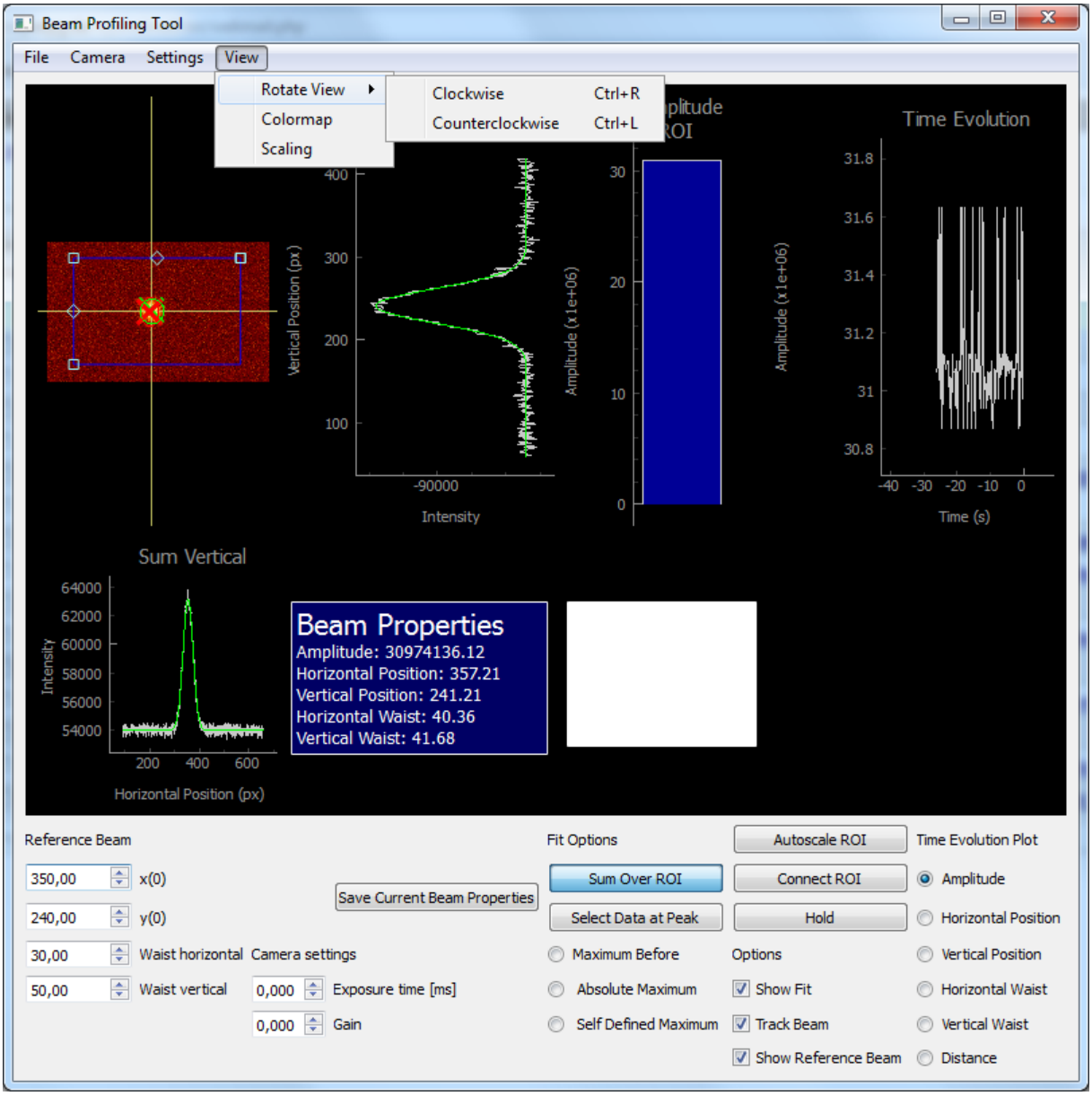
Below, the compatible camera types are shown. In the according sub-menus the available cameras are displayed. By clicking on them, the current operating camera is changed.

4.3. Menubar – Settings



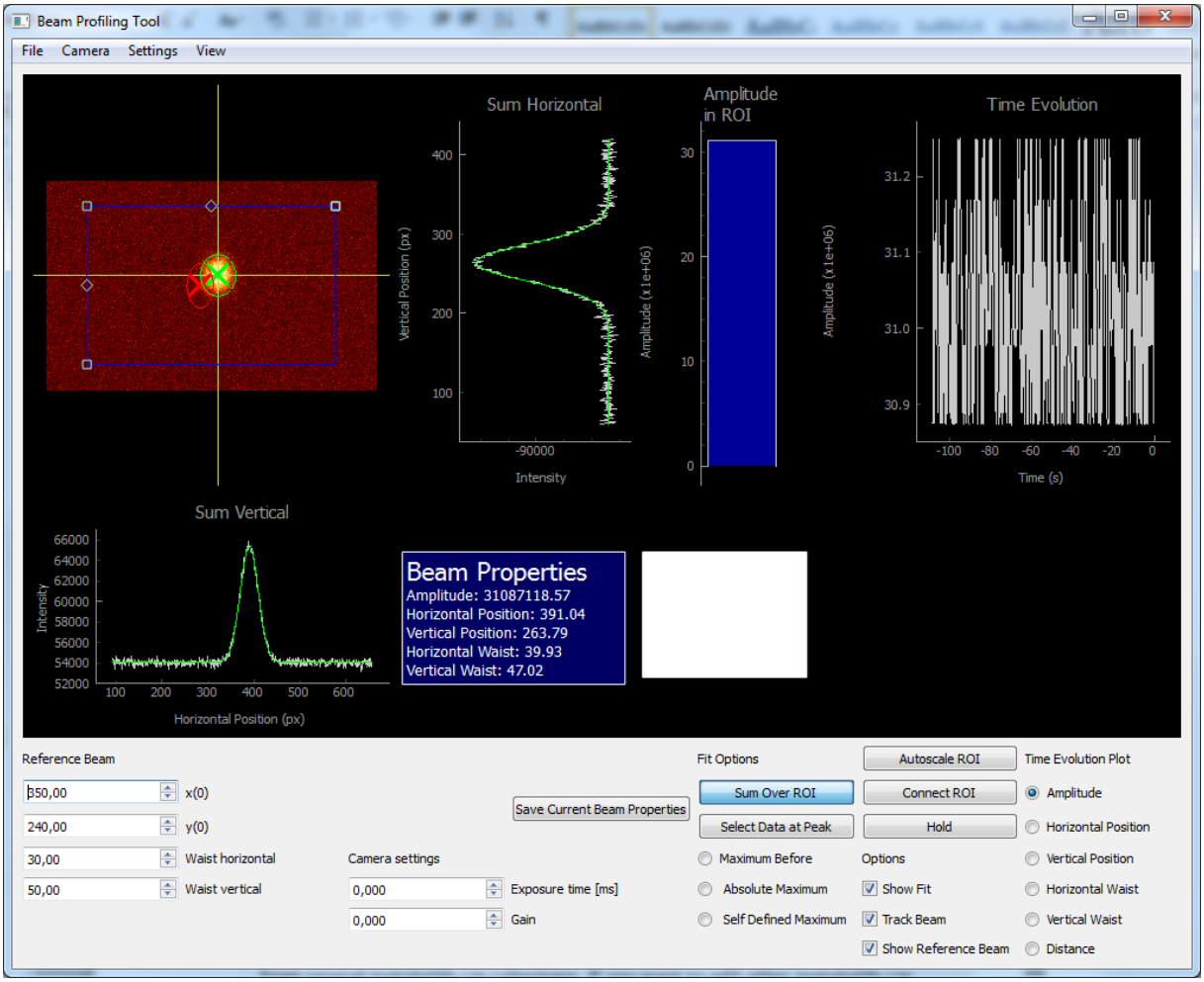
- **Saturation – Threshold:** The minimal number of pixel that has to be saturated for the saturation warning to show up can be changed here
- **Buffer Size:** The size of the data buffer can be adjusted here
- **Clear Buffer:** Clear the data buffer

4.4. Menubar – View



- **Rotate View:** The displayed image can be rotated clockwise or counterclockwise
- **Colormap:** Here the colormap of the image can be changed. It can be chosen from several matplotlib.cm colormaps. If you want to add other matplotlib.cm colormaps, in the method "ChangeColorMap" in the source code (in BeamProfilingApplication.py) the name of the colormap has to be added to the "cmaps" list and a name for the GUI has to be specified in the "cmapnames" list
- **Scaling:** The pixel size (in micrometer per pixel) can be entered. The data in the buffer is recalculated automatically and the scaling of the graphs is adapted.

4.5. Overview of the Window



At the bottom of the window, the program offers several different options. In the far left column, the properties of the reference beam can be specified. The reference beam will be shown in the image in red, if the “Show Reference Beam” option is checked.

In the second column from the left, the exposure time and the gain of the camera can be adjusted. The current configuration is stored in a “config” file. When reopening the program and starting the same camera on one computer, the settings are automatically changed to the previously used ones.

The “Save Current Beam Properties” button does the same as the “Save Current Beam Properties” option in the file menu.

In the “Fit Options” column it can be chosen wether the horizontal or vertical sum over the ROI or a single line and column of the ROI is used for fitting (Select Data at Peak). When using single lines for fitting, it can be chosen if the lines should be taken at the peak (maximum) of the image before (good when beam is moving

slowly), at the absolute maximum of the current image or if one wants to define the lines by oneself. In all three cases a second cross-hair appears which indicates the two lines used for fitting.

The “Autoscale ROI” button scales the ROI size to four times the beam waist. When the “Connect ROI” button is checked, the ROI moves with the beam. Pressing the “Hold” button pauses the live view. The options below offer the possibility to choose if the fitted curves, the detected beam or the reference beam should be shown.

In the far left column one can choose the property whose time evolution is plotted on the left side.

In case several pixels of the image are saturated, a red warning appears in the white box in the window. By right – clicking on the plots, the automatic scaling can be switched off and the scaling can be adjusted manually.

4.7. Further Options

In the “FitGaussian” method in “MathematicalTools.py” two different options to improve the performance of the fitting process are implemented. They are not used, as the Fit has been observed not to be limiting performance. The Boolean variables “usesplit” and “useslice” in the arguments of the “FitGaussian” function define if one of those methods is used.