|  |  |  |  |
| --- | --- | --- | --- |
| **Revision History** | | | |
| **Revision** | **Date** | **Name** | **Change Description** |
| 01 | 07/15/2021 | Yang Hyo Kim | New Document |

1. Purpose
   1. Code explanation for SV Beam Measure.
2. Scope
   1. Engineering/Manufacturing
3. Responsibility
   1. Engineering and Manufacturing is responsible in keeping this document updated
   2. Procedure must be followed by Engineering/Manufacturing
4. References
   1. List any References

# Development environment

1. OS (Windows 10)
2. Python package (Anaconda3-2020.07-Windows-x86\_64, <https://www.anaconda.com/products/individual>)
3. IDE (Spyder 4.1.4)
4. Python 3.8.3
5. GUI (wxPython 4.1.0 msw (phoenix) wxWidgets 3.1.4, <https://wxpython.org/>)
6. Plotting (matplotlib 3.3.4, <https://matplotlib.org/>)
7. exe freezing (PyInstaller 4.2, <https://www.pyinstaller.org/>)

# Source code location

D:\Optical Biosystems\Regular work\RND\2020-07-14 Beam position\Python codes\2020-08-23 update

# Manual location

\\ma2files\Production systems\04\_Documentation\Calibration\PROC-10XX\_02, SV Beam Measure.pdf

Delivered exe file location  
\\ma2files\Production systems\09\_Production Tools\Python\SV Beam Measure

# Background information

**Peak value ratio definition**

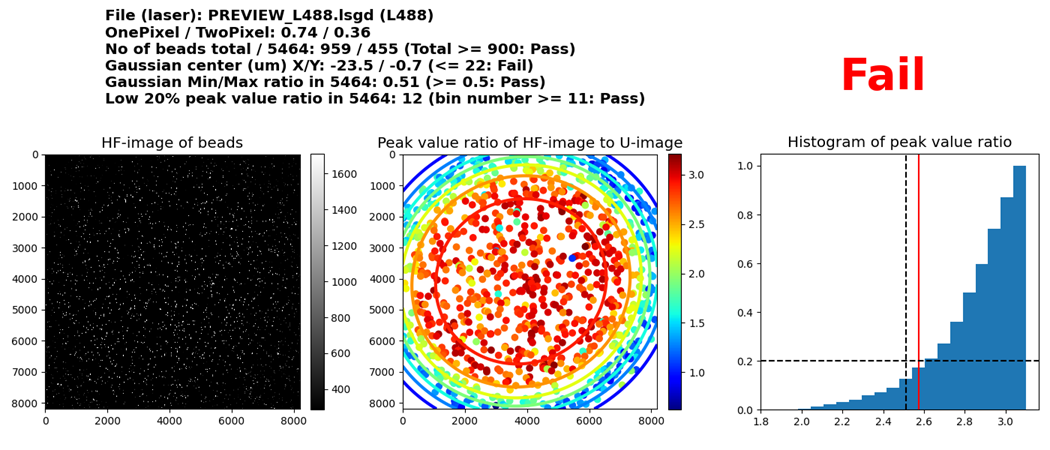
|  |  |
| --- | --- |
|  |  |

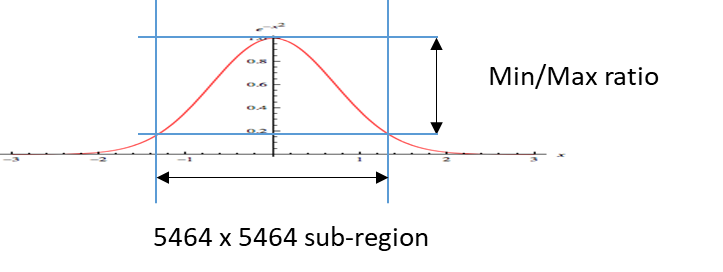
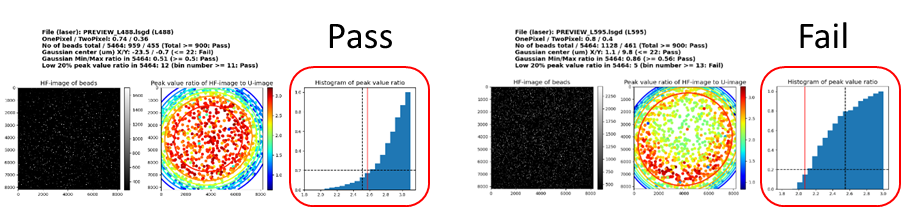
U image is the uniform illumination image which is given by averaging 3 raw images with phase shift. HF is the SAO reconstructed image with higher resolution. U image normalization is necessary to compensate the illumination intensity and bead-to-bead variation.

**Super-Gaussian function**<https://www.rp-photonics.com/flat_top_beams.html>

|  |  |
| --- | --- |
|  |  |

**Pass/Fail criteria**



1. **No of beads total / 5464**  
   The algorithm tries to find a possibly single bead instead of aggregated beads. This item shows the number of single beads detected by the algorithm inside a whole field of view and 5464 x 5464 sub-region. Enough number of beads must be tested to guarantee the reliability of Pass/Fail decision.
2. **Gaussian center (um) X/Y**  
   Peak value ratios between HF-image and U-image from single beads generates a scattered 2D data. This data is fitted to a 2D super-Gaussian function. This item shows the center of the Gaussian function in x and y coordinates with a um unit.
3. **Gaussian Min/Max ratio in 5464**  
   From the Gaussian fitting, the ratio between the minimum and maximum peak values inside a 5464 x 5464 sub-region. This item measures the image quality degradation from a center to an edge.  
   
4. **Low 20% peak value ratio in 5464**  
   From the bead data, a histogram for peak value ratio is generated. This item measures the overall image quality inside a 5464 x 5464 region by checking lower 20% location in the histogram.  
   

# Program flow overview

## [SVBeamMeasure\_20201103.py]

|  |
| --- |
| GUI implementation using wxPython To keep the GUI responsive when the application has to do long running tasks, wx.PostEvent and Threads are used. Finding isolated single beads takes long computation time, so the workload is splitted into multiple pieces utilizing multi-core CPUs.  **runBtnClick()** run button click event handler (initiate data processing)  -> runData1Thread(), Worker Thread Class  -> SAOMeasureLib.findLocMulti(), *Find Isolated Single Beads*  -> BeadCalibLib.LsgdToImageStack(), *image stack of 12 images from a lsgd file*  -> BeadCalibLib.EstimateBackgroundNoise(), *black background intensity value*   -> BeadCalibLib.SelectIsolatedTargets\_Multi1()  -> BeadCalibLib.SelectIsolatedTargets\_Multi2(), *multiprocessing*  -> BeadCalibLib.SelectIsolatedTargets\_Multi3()  -> runData2(), *Post event hander for worker thread (Data processing and showing result)*  -> SAOMeasureLib.analPeak(), *peak intensity ratio between u-image and hf-image.*   -> SAOMeasureLib.findPeakLoc(), *peak intensity location (row, col)* |

## [SAOMeasureLib.py]

|  |
| --- |
| **findLocMulti()** Find Isolated Single Bead with multiprocessing (multiple-cores)  -> BeadCalibLib.LsgdToImageStack()  -> BeadCalibLib.EstimateBackgroundNoise()  -> BeadCalibLib.SelectIsolatedTargets\_Multi1()  -> BeadCalibLib.SelectIsolatedTargets\_Multi2()  -> BeadCalibLib.SelectIsolatedTargets\_Multi3()  **analPeak()** RETURN A PEAK INTENSITY RATIO MATRIX BETWEEN UNIFORM ILLUMINATION AND RECONSTRUCTED IMAGES.  **findPeakLoc()** FIND THE PEAK INTENSITY LOCATION (ROW, COL). |

## **[BeadCalibLib.py]**

|  |
| --- |
| Conversion of existing Matlab codes into Python  - LsgdToImageStack.m - EstimateBackgroundNoise.m - SelectIsolatedTargets.m  **LsgdToImageStack()** Return an image stack of 12 images from a lsgd file  **EstimateBackgroundNoise()** Return estimated black background intensity value from the camera  **SelectIsolatedTargets()** Return an Nx2 (N number of targets and (row,col) location) array of isolated single bead location from an input image SelectIsolatedTargets() is split into the following 3 parts to utilize multiple CPU cores for speed-up  **SelectIsolatedTargets\_Multi1()** Rule out possible aggregated beads and pick up isolated single beads.  **SelectIsolatedTargets\_Multi2()** SELECT ISOLATED TARGET BEADS. THE LONGEST RUNNING TASK (MULTIPROCESSING).  **SelectIsolatedTargets\_Multi3()** UNIFORM NUMBER DENSITY CONTROL |

## **[GaussFitLib.py]**

|  |
| --- |
| 2D Gaussian fitting is based on least-square optimization method. The initial starting point for the optimization is calculated from moments and fed into the iterative optimization algorithm. This optimization algorithm needs an error function to minimize. The error function takes Gaussian parameters as arguments with given data.  **gaussian()** Return a Gaussian function with the given parameters  **moments()** Initial Gaussian parameters before fitting  **fitgaussian()** Gaussian parameters by 2D Gaussian fitting  **errgaussianPercent()** Calculate deviation from a 2D Gaussian function  Above functions are developed for ArgoLight data (grid data), while Optic Check deals with bead data (randomly scattered data). The following new functions were developed for bead data (randomly scattered data) based on above functions.  **superGaussian()**  **momentsBead()**  **fitSuperGaussian()** |