|  |  |  |  |
| --- | --- | --- | --- |
| **Revision History** | | | |
| **Revision** | **Date** | **Name** | **Change Description** |
| 01 | 07/12/2021 | 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

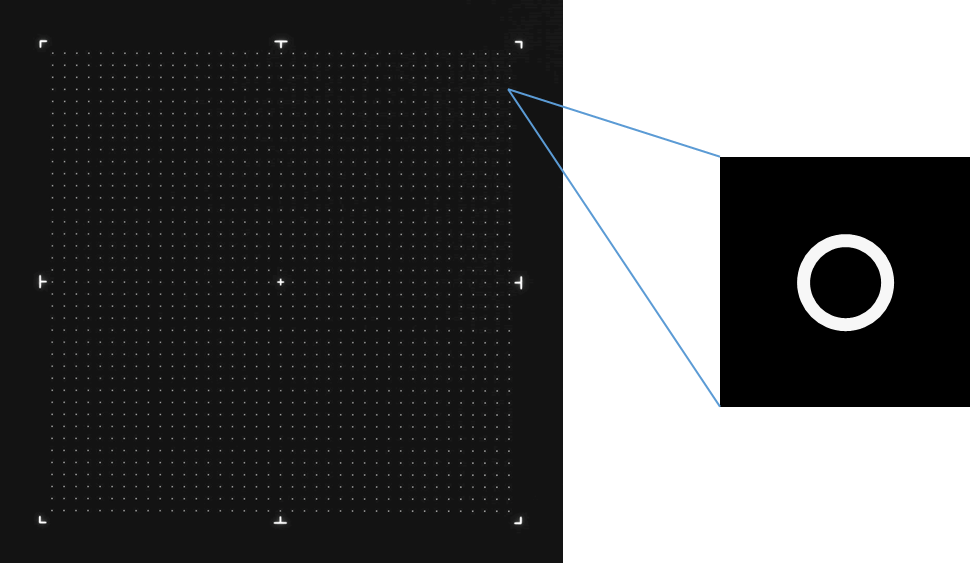
# Manual location

\\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

**Argolight fluorescent calibration slide**<http://argolight.com/widefield-super-resolution-and-confocal-microscope-calibration-slides/>



**OBI custom Argo-LM**

This pattern consists in a matrix of **33 × 33 rings, separated by 20 μm**, on a total field of 680 × 680 μm² (?). The field of rings is surrounded by eight landmarks, and exhibits no cross in its center. The ring size is 7 μm (?) which makes the initial alignment job faster because it is relatively big and bright.

Vinh uses this slide with the Matlab code (**AnalyzeArgoLM\_Custom\_SnapZ.m**).

**Argo-Check Homogeneity**

This pattern consists in a matrix of **39× 39 rings, separated by 15 μm**, on a total field of 600 × 600 μm². The field of rings is surrounded by eight landmarks, and exhibits a 7.5 μm long cross in its center. The ring size is 1 μm (?) which makes the beam characterization job more accurate, but it is more adequate for advanced users because it is relatively small and dim.

Irene uses this slide with the Matlab code (**AnalyzeArgoHomogeneity\_NPh.m**).

**Andor Zyla 4.2 PLUS sCMOS camera**

<https://andor.oxinst.com/products/scmos-camera-series/zyla-4-2-scmos>

2048 x 2048 pixels with 6.5 x 6.5 μm pixel size. With 20x magnification, on the sample plane, the pixel size is 0.325 μm and field-of-view is 665.6 x 665.6 μm².

# Program flow overview

## [SVBeamMeasure\_20201103.py]

|  |
| --- |
| GUI implementation using wxPython    **runBtnClick()** run button click event handler (initiate data processing)  -> runData()  **runData()** read a data file, processes the data, and show the result  -> ArgoDataLib.argoAnal(), *Read a data file and process the data* -> sysInfoFileName(), *File name string about system information* -> plotFig(), *Post-data processing and displaying result figure*  -> GaussFitLib.fitgaussian(), *Gaussian parameters by 2D Gaussian fitting*  -> GaussFitLib.moments(), *Initial Gaussian parameters before fitting*  -> GaussFitLib.gaussian(), *Return a Gaussian function with the given parameters*  -> processParamsFig(), *Convert Gaussian fit parameters (pixel -> um)*  -> GaussFitLib.errgaussianPercent(), *Calculate deviation from a 2D Gaussian function*  -> titleFig(), *Figure title string with system information*  -> tickFig(), *Ticks and ticklabels*  -> showResultTxtFig(), *Text in the figure*  -> saveFig(), *Save the final result into a file* -> changeDataFileName(), *Change the input data file name with one containing system info*  When reference and testing arms are measured separately, Merge files button behaves like the following.    When reference and testing arms are measured together, Merge files button behaves like the following.      **mrgBtnClick()** MERGE BUTTON CLICK EVENT HANDLER  -> ArgoDataLib.checkCh(), *Check if all 4 channel files exist* -> ArgoDataLib.mrgRefTest(), *Reference and test arms are separately measured*  -> ArgoDataLib.mrgHor(), *Merge result images files horizontally*  -> ArgoDataLib.mrgVer(), *Merge result images files vertically* -> ArgoDataLib.mrgHor(), *Merge result images files horizontally*  -> ArgoDataLib.fnameOut(), *Returns merged output file name string* |

## [ArgoDataLib.py]

|  |
| --- |
| **argoAnal()** Analyze ArgoLight data Call different function according to file type (tif vs lsgd)  -> argoTif(), tif file  -> argoSnapZ(), *Return laser beam heatmap from a raw image.*  -> argoNPh(), lsgd file containing 12 raw images  -> argoSnapZ(), *Return laser beam heatmap from a raw image.*  **argoSnapZ()** Conversion of existing Matlab codes into Python  - AnalyzeArgoHomogeneity\_NPh.m - AnalyzeArgoLM\_Custom\_SnapZ.m  Distinguish between LM and Homo slides Low pass filter and crop image Get intensities of all grid spot Make 2D Heat-map by picking up 10th highest intensity from each grid  **Merge image files functions**  **checkCh()** Check if all 4 channel files exist  **mrgRefTest()** Reference and test arms are separately measured  **mrgHor()** Merge result images files horizontally  **mrgVer()** Merge result images files vertically  **fnameOut()** Return merged output file name string |

## **[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 |