Research Summary

Conclusion for my research work from August 12 to October 4. My main work focused on two parts: one is looking into the documentations of the Nuvu camera and write an interface to replace the previous code for the Andor camera; the other part is to benchmark and examine different image processing methods' fidelity. Here, I provide mainly a record of what I have tried and some of the results.

Main Work

1 Nuvu camera

1.1 Resources

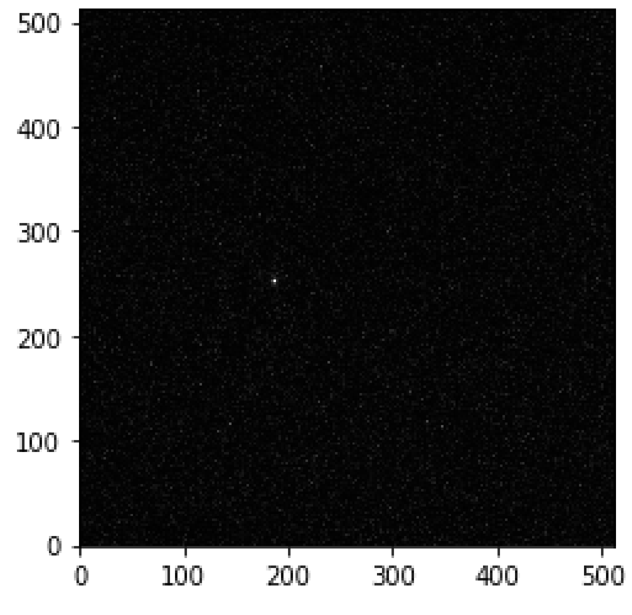
The Nuvu camera comes in with a SDK documentation, though not as detailed as the Andor documentation. Moreover, what I find very useful are the examples given by Nuvu in which some basic functions are called and gives a general idea of how the acquisition process works and what calls the camera is expecting. Another resource is the previous code on the github page, especially files of pulse_sequence.py and andorCamera.py, which are connected through andorServer.py. Those files specifies what functions are used, which is only a small fraction of all available ones. Therefore, the most efficient way should be to look for whatever function is needed and then try to find the corresponding examples in the documentation.

1.2 Test Code

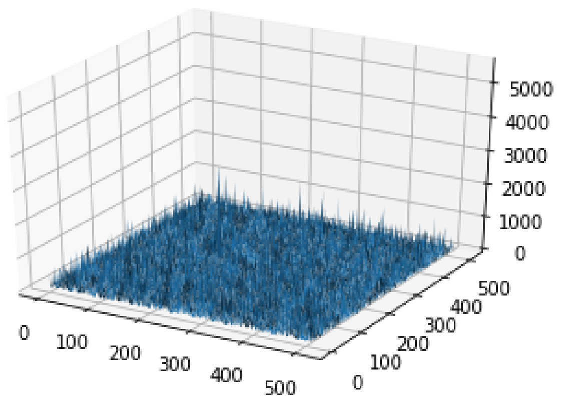
I have written a test case called test_acquisition.py, and it should be included with this report. This test case is simply written, and only some minor modifications are made compared to the example case given called simpleAcquisition.c. Because the Software Development Kit (SDK) are all written in C++, to write an interface in python, a python library called ctypes must be used to connect to dynamic link libraries (DLL) and also call C compatible data types. Therefore, test_acquisition.py is mainly a test or demonstration of using ctypes for simple acquisition.

One of the problems I met is that, different from how Andor wrote their DLL, Nuvu defined their own Structure NuvuCamera and NuvuImage, etc. Ctypes itself doesn't deal with this case, and the detailed definition of the Structures aren't present. My solution is to assign an address pointing to an integer and 'pretending' that it points to the correct Structure. Later on, use other functions to assign actual values, such as ncCamOpen for NuvuCamera.

Some other minor problems I solved are all commented in the code.



(a) Image



(b) 3D graph

Figure 1: Nuvu image of exposure time 20ms

1.3 Modified code

I have also tried to replace and modify some parts of pulse_sequence, andorCamera, and andorServer to fit with the Nuvu Camera. I have included them as well. However, since I am not able to test the code, I would suggest take my code as reference when implementing the actual code in the future.

1.4 Preliminary images

7 images of FITS format obtained from Nuvu camera are included with this report as well, and each of them has different exposure time of one bright ion. These images are taken when the staq group happens to trap an ion, so it might look differently when move to a different setup or with different parameters. A preliminary image of exposure time 20ms is given in Fig.(1).

2 State detection algorithms

The state detection algorithms are based on the fluorescence of the ions. For my test cases, I focused on the case of two ions since I only have actual experiment image for this case. For two ions, if label bright ion as 1 and dark ion as 0, then the different combinations are 00, 01, 10, and 11. Fig.(2) gives an example of each

2.1 Overview of the current algorithm

For the current algorithm, the model of the photon distribution is asuumed to be Gaussian:

$$P(r|r_c,\sigma) = Ae^{-\frac{|r-r_c|^2}{2\sigma^2}} \tag{1}$$

the steps are as follows:

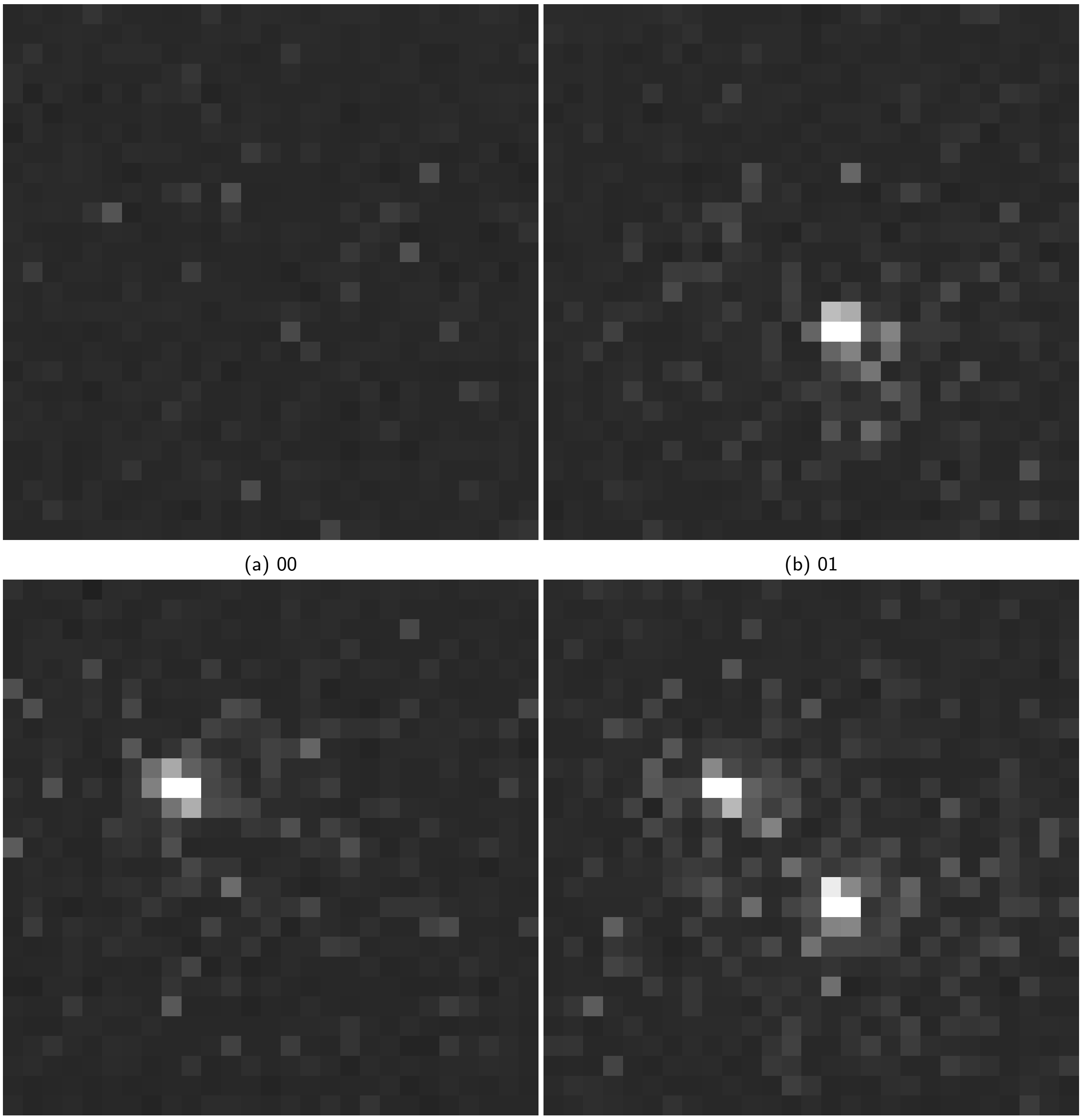

(a) 00 (b) 01 (c) 10 (d) 11

Figure 2: Image of two ions, exposure time 20ms

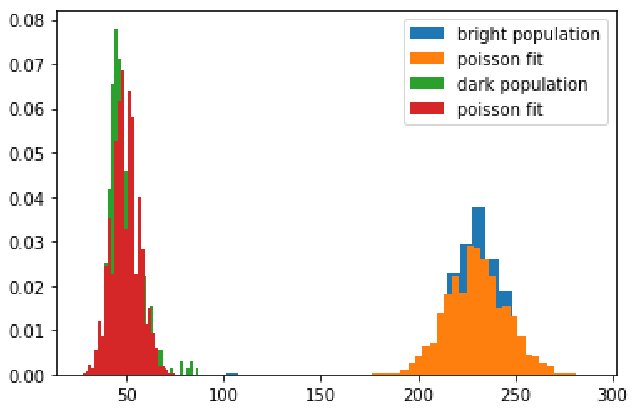


Figure 3: Populations of bright and dark ions' mean pixel count in the ROI for experiment images

1. Take reference image of state $|11\rangle$.

2. Make initial guess of parameters of model. Parameters include the coordinate of the whole system's center, σ of the distribution, amplitude A of the distribution, background level, background standard deviation, etc.

3. Use function lmfit.optimize to fit all the parameters of the distribution to get the least χ^2 for the reference image.

4. Use the optimized parameters from step 3, and calculate all the χ^2 of possible states. Output the one with smallest χ^2.

2.2 Image simulation

For experimental images, it is hard to label all the images 'correctly' because it is not so obvious what is the threshold between spontaneous decay and the randomness from noise or Poisson process. Therefore, I produced simulated image taking into account of spontaneous decay, Gaussian distribution, pixelization, and the population distribution. Therefore, for the produced images, the initial state can be set as the correct label, and the time of decay can be easily seen as well.

1. For Spontaneous decay, I took the lifetime to be $1.17 \times 10^3 ms$. The exposure time is set to be 20ms (same as experimental image).

2. For Gaussian distribution, I use the same amplitude fitted from the reference image.

3. For pixelization, by observation, the pixel size should be a sqaure of 10×10 in the .png file.

4. For the population distribution, initially I used Poisson distribution, but later on modified it to be sampling from the population of the experiment images. The population is shown in Fig(3). The ROI are shown with black boxes in Fig(4).

As a result, the simulated images are like in Fig(5). Here, we can see that the randomness of the whole simulated images is still not as large as the experimental images, which have bright spots in many regions. However, the simulated images are much easier to benchmark, therefore the later method are examined on these images.

2.3 Other methods

Other methods that I have examined include PMT threshold method and a modified Gaussian method.

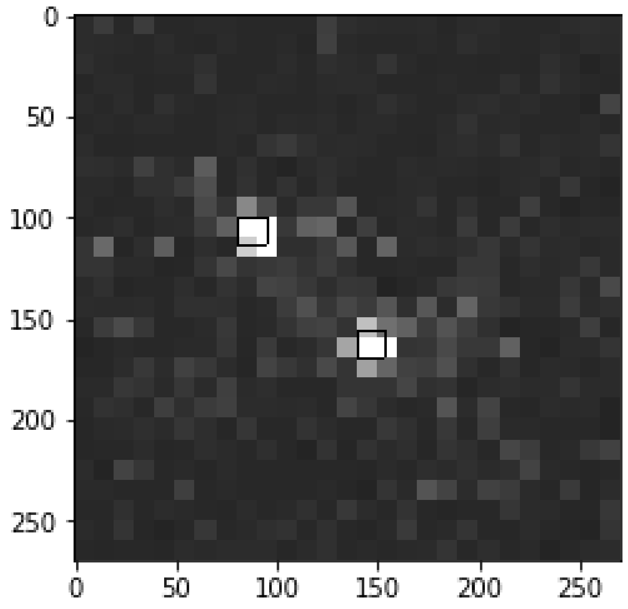
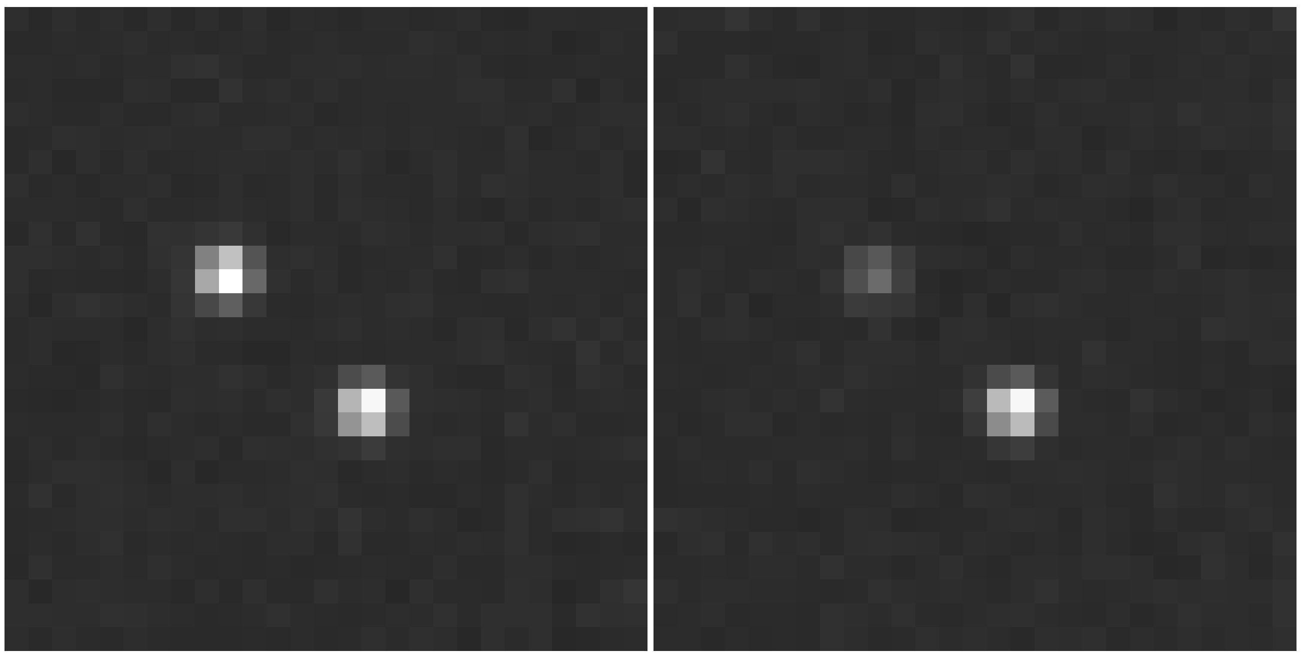


Figure 4: ROI demonstration on $|11\rangle$, with the boundaries labeled by black lines



(a) undecayed state of $|11\rangle$ (b) decayed state of $|01\rangle$. Decayed at 14ms/20ms.

Figure 5: Simulated image of two ions

2.3.1 PMT threshold

In the PMT threshold method, the image produced from EMCCD is used as segmented PMT. The processes are as follows:

1. Follow steps from current Gaussian method to find the parameters of the ROI.

2. Calculate the mean pixel count of each ROI on image, i.e. two for two ions.

3. Compare it with the threshold and output the state.

2.3.2 Modified Gaussian method

In this modified Gaussian method, the procedures are as follows:

1,2,3 are the same as the current Gaussian method

4. Fix all parameters other than the amplitudes of the Gaussian distribution, i.e. fix the centers, background level, rotation angle, etc. In principle, some of these can also be flexible and be fitted by lmfit.optimize, but in order to decrease the run time, only the amplitudes at the two ion centers are left to fit.

5. Set a threshold for each amplitude and output the state.

2.3.3 Results

Since I have been modifying the models and procedures for simulating images, I have tested the current Gaussian method and the PMT threshold method but not the modified Gaussian method. However, in principle, the modified Gaussian method is similar to PMT threshold but also takes into account of spatial distribution, so its performance should be at least as good as PMT. The question is that whether the trade-off of run time and accuracy is worth it between the modified Gaussian method and the PMT threshold method.

For the tests, 5000 simulated images are used, in which there are 770 decayed images and 49230 undecayed images. The result for current Gaussian method is fidelity of 99.0%. The result for PMT threshold is in Fig(6), which gives a maximum of 99.906% when mean pixel count threshold is set at 146. Here, the simulated images have dark region count average at around 45 and bright region count average at around 165.

Previously, I have also done similar analysis for new Gaussian method on simulated images of different models, the result is in Fig(7) to give a feeling of what it's like. The resulting trend tends to be similar to PMT threshold method. Here note the threshold is for the amplitude of the fitted Gaussian distribution.

2.4 Other attempts

I have also made some attempts, which might not be that useful but I feel that leave a record here can be a good idea.

1. For the initial guess, the previous lmfit.optimize can sometimes breakdown for me though it seems for lattice group it works fine. However, for my case I suspect is the initial guess's problem, since the initial guess makes assumption on the magnitude, position, and σ of the two Gaussian distributions, which might cause trouble especially when one pixel is of size 10×10 in the image file. To solve this

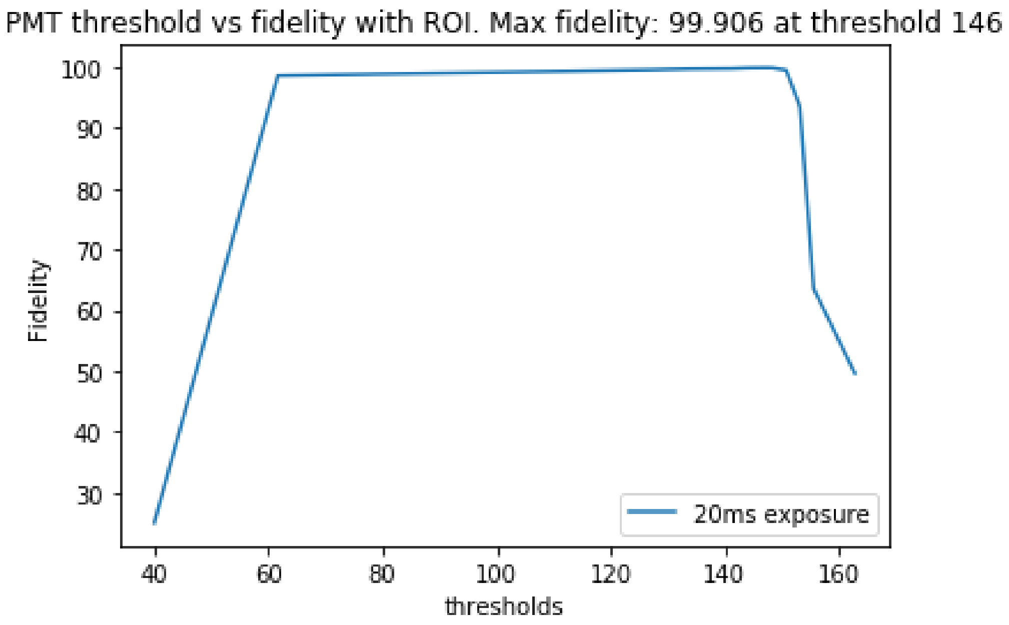


Figure 6: PMT threshold method examination

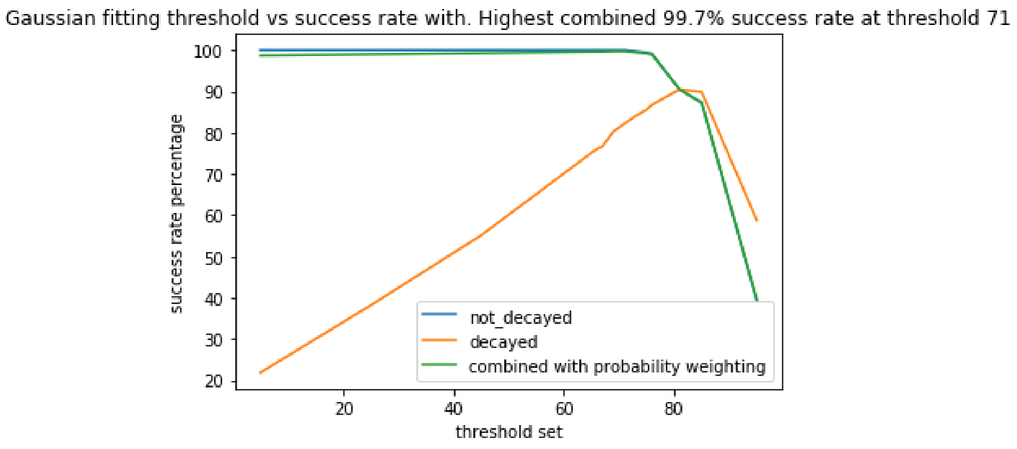


Figure 7: New Gaussian threshold method examination

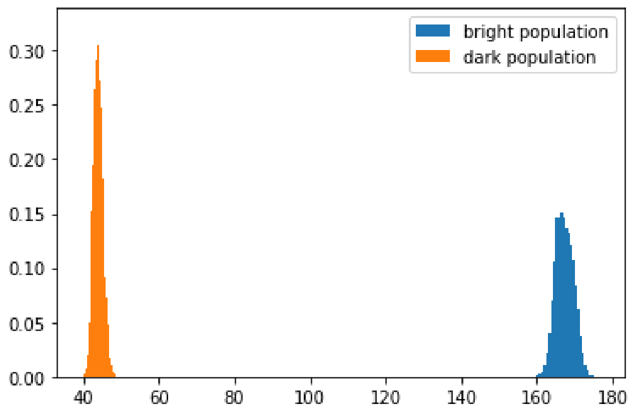


Figure 8: Populations of bright and dark ions' mean pixel count in the ROI for simulated images

issue, I implemented KMeans to cluster all the peaks of the two distributions to get a better initial guess.

2. In the beginning I was suspicious of the Gaussian model and tried to propose another model. Therefore, I simply assumed that the photons distribute like photons from an isotropic light source hit on a screen, which gives me a distribution as follows:

$$P(r|r_c, d) = A\frac{1}{(1+(r/d)^2)^{3/2}} \tag{2}$$

Although proven to be an oversimplified model without considering the effects of imaging devices, I found that this model generally has better fit than Gaussian model such that the χ^2 of this model is usually just 80% of the χ^2 for Gaussian model. Therefore, not suggesting that this model should be used, but instead we should be aware of the potential of finding a new model much more accurate than the Gaussian model.

2.5 Possible improvements in the future

The population distribution of bright and dark regions of the simulated images is as plotted in Fig(8). It is clear that for these images they are much more centralized than the experimental images, which decreased the randomness and therefore the fidelity from this report is higher than expected.

I have just found out of this problem on Thursday, and had some discussion with Joe but I didn't have the time to fix it. The potential reason for this is that I only used the segmented PMT result to sample for the random process, which was previously assumed to be Poisson process. However, I am still using Gaussian for the spatial distribution, which is both not accurate and not randomized enough. A possible brute-force way of solving this is to simply take the whole 2D matrix from the experiment image and do sampling from there, which includes the spatial distribution. This method should definitely give simulated images that resemble the experimental images. Worth noticing, in order to take into consideration of cross talk, the sampling process should be from four population of $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$. The four groups should be classified from experiment image, but this classification process itself already has bias in it, i.e. none of the methods can guarantee to have 100% accuracy. Currently, only state of $|11\rangle$ can be produced with 100% accuracy since it is the ground state.

3 Summary

For the first part on the Nuvu camera, I have completed replaced the Andor camera code with Nuvu camera code with some modification customized to the Nuvu camera. Moreover, I have written a test case, which tests the fundamental functions of the camera and can serve as a demonstration for writting interface between Python and C++ based dynamic link library. I have attempted to connect the camera to the Linux PC of lattice group and install the software packages. However, because the limit of time, I didn't have the time to debug for some of the modified codes, which should be taken care of in the future.

For the second part on the state detection algorithm, I have written and tested some methods besides the current Gaussian method. Moreover, I have produced simulated image to better benchmark the performances of the methods. However, due to imperfections in the simulation process, the result obtained now can be higher than the actual result, but I believe the whole procedure and benchmarking standard are set, and implementing the proper simulation process shouldn't require that much effort.