**Running NARS Auto nCT (TomographyV6.4.21.py)**

Program written by: Matt Bisbee at NARS Lab Ohio State University under Dr. Raymond Cao

Instructions Date: December 16, 2021

Prerequisites to Running the code:

* If this is the first run on a computer, please see installing NARS Auto nCT instructions
* Ensure Micro-Manager (MM) and pycromanager is up to date on the nightly build.
  + Go to <https://micro-manager.org/wiki/Micro-Manager_Nightly_Builds> to download most recent build of MM and within a command line, type “py -m pip install --upgrade pycromanger”
* Find what communication port the stages are connected to by going to device manager and finding the related “Port” information, the default values for communication ports for camera, rotation stage, and XYZ stage are listed in the entry boxes already. This would most likely change if running on a different computer.
* If running from .py, ensure folder has the following: FewViewPart2.py, LeftoverTomography.py, NewCameraControl.py, SMC100Final.py, XYZMotion\_Station.py, OSU\_NRL.ico, OSU\_NRL.png
* This code is written on a Windows based machine and only tested for Windows OS, adjustments would have to be made for other forms of OS

Running the Code:

1. Open Micro-Manager 2.0 Gamma and connect to the appropriate camera. Once this is open correctly, you may run the TomographyV6.4.21.py file through command line or code editor
2. In the GUI start at “1. Input Experimental Parameters” section:
   1. **File Name:** Think of any name for the file as long as it is one string and follows general rules for Windows files
   2. **File Location:** By default it will save to the same location the program is running from, you can change this path and it is possible to enter the name for a folder that doesn’t exist, it will create a new folder for you
   3. **Exposure Time:** Enter the time in seconds for each image acquisition
   4. **Rotation (deg):** This is the smallest increment of degree you want the program to run by. All angles must be entered as a float (eg. 1.0 NOT 1)
   5. **Starting Point (deg):** What angle you’d like to start from. All angles must be entered as a float (eg. 0.0 NOT 0)
   6. **Max Rotation (deg):** This will be the last angle the stage rotates to. All angles must be entered as a float (eg. 1.0 NOT 1).
      1. If the max rotation degree is set to 360.0 and the starting point is set as 0.0 degree, the code will take an image at 0.0 degree and remove the 360.0 as they are equivalent.
   7. **EM Gain:** This is optional for the EMCCD camera use, if the EM Gain passed through is greater than 3, the program will try to adjust this value in Micro-Manager. If it is 3 or lower, it will not adjust the value.
      1. If not using EMCCD, leave the value as 3 and the code will not try to change this value
   8. **SMC Port:** The rotational stage will be connected to a communication port, find the correct port in device manager.
      1. If connected to both Rotation and XYZ stages, may have to play with connecting to each device to see which is which
   9. **Camera Port:** This should always be COM1 from my experience
   10. **Select if Running Few View**: Keep this box checked if the program will run few view method, if not, the program will run sequentially from Start to Max by Rotation increments
   11. **Enter:** Once you hit the enter button, the program will collect the inputs and check if anything is wrong.
       1. If nothing is wrong/invalid, the black box will be populated by the degree list for the run.
       2. If you input a file name/location combination that already exists, the black box will populate with a question on how you’d like to proceed. This is fairly straight forward, you can at this point overwrite the existing files, change the name, change the folder, or change both
       3. If the entered degrees are not divisible into each other, the black box will populate asking you to change these values to ensure divisibility. (eg Rotation of 7 deg in a start 0 max 180 would cause a divisibility error)
3. Proceed to the “2.0 Open Programs and Initialize Devices” section:
   1. The first button references to Micro-Manager being open and running, if it is not, the code must be closed out and this must be opened first. This is simply a check to remind the user that MM must be open first.
   2. The user can snap as many test images as they would like. These images will be automatically saved to the folder the python file is running from as TestImageX with X being a number increasing from 1
      1. This serves two functions:
         1. Ensuring the camera connects properly and takes a picture
         2. Can be used to test alignment of the object (if the object needs to be moved, click the “Show Control of XYZ Stage” button and adjust accordingly
   3. Once the user is satisfied with the images or if not necessary, select “Camera Ready”
   4. The next section tests the SMC controller to ensure the rotational stage is being moved properly. This will connect to the stage and make some moves to ensure movement resolution
      1. If the button becomes green, it has passed and the user can selected “SMC Ready” button
      2. If the button is red, check the connections on the controller or check if the port is incorrect
   5. Once there are 3 green buttons along the right side of this section; MMReady, Camera Readied, and SMC Readied, the Start button will become active
4. In “3. Run the Tomography and Monitor It” section select **Start** when you are ready
   1. The tomography experiment will start at this time, the tomography code runs on two threads
      1. The control peripherals thread is what is used to move the stage, take pictures, and create messages
      2. The Update GUI thread puts the most recent messages on the screen and keeps the GUI refreshed
         1. Due to threading complications, the pictures being updated on the screen lag behind the control at the start, you may not always be seeing the most up to date picture. The images are a reference to ensure everything is lined up at each angle. Very specifically, the first two images will not be displayed on the screen.
   2. The top message will tell the user how many images remain and estimate how much remaining time there is in the experiment based on acquisition times and time spent on movements.
   3. The middle message tells the user what image name was just saved
   4. The bottom message only will change to an error if something comes up during the run
      1. If an error occurs during the run, the code automatically sends the remaining list of degrees at time of error to a text file. Alternatively, one can press “Remaining To Text” button and it will create a text file of the remaining degrees. These can be uploaded to “LeftoverTomography.py” version of tomography to restart a run.
         1. The LeftoverTomography runs very similarly but has a box to browse for an input .txt file of degrees. These must be in form of one degree per row
   5. The “Quit” button can be used to close out of the program and upon selection will try to disconnect from the stages and cameras if it can
      1. If the user quits out of code during a run, the camera will still take the last image it had started so the code does not fully stop until the camera is disconnected. This can be shown as you cannot close MM as the camera takes and image
   6. Outside of the Run section, the code updates the most recent images replacing the previous image and updates the remaining degree list so the user can get an idea of what degrees are coming up.
5. **Optional** Control of XYZ Stage
   1. If the user has need to adjust the XYZ stage before the run, they can press the “Show Control of XYZ Stage” button. At this point a large section of GUI will be created
   2. The first section of XYZ Control has an entry to insert the COM port that is connected to XYZ
      1. The user can use this section to see which COM connects to the XYZ stage and then by process of elimination will know what COM port the rotation stage is connected to
   3. After connecting to the XYZ Stage, the user can move the stage to the desired location
      1. Note: Positive X is towards the source. Positive Y is towards East (at NRL). Positive Z is upwards
      2. The XYZ stage will throw errors if the move will send the stage past the limits
   4. **IMPORTANT:** Do not turn off or unplug the XYZ stage if it has not been homed. If the stage is not homed, the next time it turns on, it will think 0 is where it currently is at. This creates problems as the code ensures the stage does not hit limits by calculating based off a known 0 with all stages centered.

Notes on the Code

* The code is written to be a single run code, once the acquisition parameters have been entered, and the testing has started, it will continue until the end or if it encounters a fatal error. If for some reason the code needs to be shut down the user must either accept restarting a new acquisition or sending the remaining degrees needed to text file using the remaining to text button and then beginning from there in the LeftoverTomography GUI.
* Upon clicking the start button the code starts two threads which is the way for a code to “simultaneously” do two tasks at the same time by allocating them to different sections of computer processing aka a thread. Tkinter and threading are complicated and prone to errors. I was unable to find a way to stop the threads once they start therefore, once the code is started it is unable to define new threads with new inputs. This also makes it such that pausing the threads hasn’t been done even though it seems they should work. This could be a feature implemented in future versions of the code.
* One can send the remaining degrees to text file at any point and can do it multiple times in a run, if there is any fear of the code randomly crashing, it is suggested to generate this text file early and remove any degrees that did get passed from the text file before starting LeftoverTomography.py
* If the FewViewPart2 has an issue reordering the list, the black box will never populate with the list and you know there is an issue. The FewView algorithm had errors in the past and may need to be altered for a specific case.
* The rotation stage operates from -180 to 180 without being able to continuously rotate. The code creates named degree angles that are easy to interpret such as 0-360 but in reality the code is sending the stage commands that are both positive and negative. Any degree above 180 degrees is actually a negative equivalent. Aka 190 would be -170 and 350 would be -10.

Known Issues/ Common Mistakes:

* The code utilizes threading to show progress and allow for clicks on the GUI during the run, without threading the screen would look like it is not responding. The update of the images and text does not populate until the second image has been taken. After this point, in theory, the code updates the image/messages to be the most recent image.
* Taking a snap will freeze the screen as “Not Responding” as this is not threaded, the program is not failing, it is just focusing on taking the image, wait until the exposure time has elapsed before thinking the program has failed
* Pressing the quit button mid-run will close communications with the stages and stop the run however it will not send any remaining list to text and possibly will not home the rotation stage. The quit button does not send the XYZ stage to home ever so if quitting, make sure the XYZ stage has been homed.