Updated 07.12.2022

Ray Tracing Code Documentation

Basic Overview

- Highly computationally intensive ray tracing code with base code utilizing CUDA and Python for running simulations
- Simulates BOS and PIV experiments depending on input parameters and camera system settings and outputs results as .tiff image files

Code Repository: https://github.com/lalitkrajendran/photon
Extra Scripts Repository: https://github.com/liu2730/FileShares

Setup Steps

- Install Ubuntu or use another Linux distribution on a PC with a GPU
    - Better with 2 GPUs so 1 for displays and the other for calculation to not be timed out
    - The Linux version should be compatible with the latest version of drivers compatible with the GPU
        - https://www.nvidia.com/download/index.aspx
        - https://developer.nvidia.com/cuda-gpus
            - Use this to find the compute capability of the GPU which is important for setup
- In Terminal: sudo apt update
- In Terminal: sudo apt upgrade
- In Terminal: sudo apt install build-essential
    - Make sure the distribution is up to date
- Install CUDA Toolkit with version greater than 10.1 Version 2
    - Make sure it is compatible with current drivers and the Ubuntu distribution
    - https://docs.nvidia.com/deploy/cuda-compatibility/index.html
        - For checking CUDA compatability
    - https://developer.nvidia.com/cuda-toolkit
        - Follow instructions from site to install properly
        - New drivers may be installed which could cause compatibility issues and lead to Linux starting up improperly (see Potential Issues section for fix)
- Download VSCode or and IDE of choice to be able to edit and compile the code
- install pip through terminal for python3
    - pip install all dependencies listed on the Github README (numpy, scipy, matplotlib, libtiff)
- install teem
    - https://teem.sourceforge.net/build.html
    - cmake should be installed before building
        - sudo apt install cmake
    - install the cmake-curses-gui as necessary following the instructions in the link
- install libpng version 1.5 (1.5.15)
    - https://sourceforge.net/projects/libpng/files/libpng15/older-releases/
    - ./configure –prefix=/usr
        - Ensure that it is built in the right place then the rest of instructions in the make/build can be followed
- install zlib
    - https://sourceforge.net/projects/libpng/files/zlib/1.2.11/

    - this is a libpng dependency so make sure to unzip and build correctly
- Other dependencies listed on the README for the code will not have to be installed
- Clone/Download the repository and navigate to the “makefile_simplified” in cuda_codes/Debug
    - ensure that the nvcc sections are pointed towards the correct folders
        - /usr/local/cuda-11.4/bin/nvcc for CUDA Toolkit 11.4 installed on current Linux PC
    - Everywhere containing: arch=compute_35,code=compute_35
        - Change the number to the compute capability corresponding to the GPU used for calculation
    - After these changes, the file should compile all the C++ and CUDA code
        - Some warnings will be present but everything should be fine if no errors are thrown
    - The rest of the README can then be followed exactly as it says
- Some of the python files may have to be edited to get rid of the errors before running. The terminal will tell you which ones after the files have been run and an error is returned
    - Most are just syntax errors or wrong folder, etc. which should be a relatively easy fix

Running Simulations and Changing Parameters (Default/First Start Settings)

- Navigate to the python_codes folder
- Create_simulation_parameters.py
    - Contains all parameters for PIV and BOS simulations such as number of dots or number of rays to trace and the camera parameters
    - Values can be edited as desired to fit simulations
    - Specify which density gradient file to use here
- Create_sample_simulation_parameters.py
    - Reads the create_simulation_parameters.py and then creates the .mat file to the specified directory
- Batch_run_simulation.py
    - Reads the .mat files above and calls on run_simulation_02.py to begin simulations
- run_simulation_02.py
    - The process for both BOS and PIV simulations
    - perform_ray_tracing_03.py is called which communicates with the compiled C++ code
    - For BOS: image without density gradients is rendered followed by an image rendered with information from .NRRD file (nearly raw raster data) for density gradients
        - Both images are saved as bos_pattern_image1.tif and bos_pattern_image2.tif and is located under sample-data/bos/images/tif folder
- perform_ray_tracing_03.py
    - Uses the the C++ and CUDA codes to complete the ray tracing calculations
- sample_run_script.sh
    - Executes the files to run the simulations
    - Add “python3 create_sample_simulation_parameters.py” to the file before the batch run so that simulation parameters are edited before every run
    - Add line to NRRD file creation for generating different density gradients before every run
- Sample NRRD file creation
    - https://purr.purdue.edu/publications/3560/1
        - Unzip bundle and navigate to analysis-package/optimization-study/src for the “create_nrrd_file_02.py”
        - Contains other files used in the writing of the research paper utilizing this code

    - Will refer to this as the Purdue repository or photon-extras in rest of documentation

Functions of the Code

- The codes uses python to interact with the base RTX code written in C and CUDA specific language
- With the python functions and script parameters for the camera setup and the BOS or PIV simulations can be edited
  - These parameters are saved to a .mat file which is read to C code to interact with the base code
- Density gradients are implemented with a .nrrd file which is essentially a more descriptive array
  - This is only important for BOS simulations as PIV uses a set of other parameters
  - Many files contained within my current setup of the documents can create variations of the NRRD file implementing various shapes
- A BOS image background is generated with a number of dots specified by the simulation parameters
- First image is created without the effects of the density files and second is supposed to be an image with shifted dots based on the given density gradients
- Resulting files are in a .tiff format (Tag Image File Format) put in the sample-data/bos/images/tif folder by default
- Run with [./sample_run_script.sh bos] in terminal and a number of python scripts is executed until the output of the final images
  - Shell script can be edited as needed to include the use of other python scripts necessary to debugging of code or producing new parameters

Basic Use Instruction for Each Simulation

1. Every time the PC is started, run this command in terminal first
   - export LD_LIBRARY_PATH=/home/liu_fl/Downloads/photon-master/cuda_codes/lib:/home/liu_fl/Downloads/photon-master/cuda_codes/lib64
   - With new code package install or different computer account, this may be different and the new path can be seen with instructions in the original code README

2. Go into the "create_simulation_parameters.py" to check and change parameters for simulation.py

3. Create a density gradient file with "NRRD_Make.py". The parameters for the function are located there. To have a deeper look, use "create_nrrd_file_02.py" which contains the function. From there change the file name if desired or even add a new function for new density gradient shapes.
   - If a new install of the files are there, both of these files will not exist. Both can be imported following directions from other parts of the documentation or can be found on github

4. Use "nrrdslice.py" to generate slice images of your density gradient. This is an optional step, and the location for where the images will be can be changed. This file does not exist in the default build so will have to be downloaded from github

5. Go into “create_sample_simulation_parameters.py” to make sure the following information is correct:
   - simulation type
   - file path for the creation of the .mat parameters file
   - the name that you want for the file

   When these are correct and you run this file, the .mat file will be generated or edited in the desired location.

6. As set currently, the parameters file should be under sample-data/bos/parameters folder. With a different set of parameters under a different file name, the undesired parameter file should be moved outside of this folder.
7. With all these files setup, the simulation can be run with “batch_run_simulation.py” and most of these steps will be written in the shell script “sample_run_script.sh”
   - With a new download, the shell script will be very basic but add the following lines in this order:
     - python3 NRRD_Make.py
     - python3 nrrdslice.py
     - python3 create_sample_simulation_parameters.py

   These can be commented out as necessary if you dont want to make a new gradient file, etc.

8. View the simulation results and the image files in sample-data/bos/images/tif folder. If save_lightrays is set to true in parameters, the binaries will be located in folders in sample-data/bos/images with separate folders for the positions and directions and within these folders, the light ray data is divided by image 1 or 2.
9. Use “ray_compare.py” for functions to comprehend the light tracing raw data (binaries).
   - On new install, this file does not exist so will have to be downloaded from the github

Files Not Part of Original Install

- /python_codes/
  - NRRD_Make.py
    - Uses “create_nrrd_file_02.py” and lays out all parameters of NRRD creation function in a way that makes it easier to understand and edit
    - Can set filepath for where the file will be created but cannot set the file name here
  - nrrdread.py
    - Reads a given NRRD file and splits into header and data (numpy array)
    - Currently is being used to scale the original density file to try and achieve working results
  - ray_compare.py
    - Used to compare the binaries generated by the simulation regarding light ray positions and direction before and after density interaction
    - Saves the first 2000 elements of the positional coordinates to a txt file
    - Other commented functions to manipulated the translated binaries
  - create_nrrd_file_02.py
    - Not created by me but imported from the extra files in the photon-extras folder
    - Base for the NRRD_Make.py file with other functions to get a numerical calculation instead of visual calculation for some parts of the simulation
  - nrrdslice.py

        - Used to take 2D slices of the NRRD files to visualize the composition and see if density gradient was correctly generated
        - Only works with cubic arrays (same number of elements in all directions) at the moment
            - Cubic arrays does not mean the the density gradient volume is itself a cube as the elements are used to divide up the predefined lengths for the volume in the density gradient file
        - Code can be changed to get a slice for all types of arrays
- /cuda_codes/
    - Any new files here are from the compilation step in the README
- /sample-data/
    - /Sample_Test_colors
        - Stores slice images for all 3 axes of the "sample-density.nrrd" and the "test-density.nrrd" (Gaussian at the time of slicing)
    - /OLDSLICE
        - Stores slice images for various functions that were able to be created for the density files
        - Name between the underscores indicate the type of function sliced
            - lin = linear function
            - quad = quadratic
            - diagonalGauss = Gaussian (sliced for file used to create weird diagonal shift results that will be described later)
            - newCreate = function generated by "createNRRD.py"
    - /bos/Original Parameters
        - contains default sample parameters that came with code download
    - /bos/images
        - /DefaultSetting_Images – Set of images I sent to the creator of the code to show that the sample does not show visible change and a set of images that show and awkward interaction between density gradient and the final results (uncharacteristic diagonal shift)
        - /Trial TIF – folder containing simulation results of various manipulated parameters and NRRD files (will be explained more in results section)
        - /TrialTIF2 – folder containing simulation results of Gaussian density file manipulation (also in appendix section)
    - test-density.nrrd
        - File for which the density profiles I created were written to for the simulations
        - Naming for this file could be changed but should also be changed in the simulation parameters if a different density in a different file is desired
- /photon-extras
    - Contains all code and plots used by the first author of paper to write paper
    - Includes parameters, density files, and the resulting images
    - Imported from repository mentioned in set up steps

Simulation Attempts and Results

- Tried running code using Windows set up and compilers
    - Code cannot compile with Windows set up
    - **<u>Don't waste your time</u>** trying to do this as code will just not work and having to download everything on windows is quite a pain

    - Future: Port the necessary files to useable format for Windows after knowing code functions correctly or if it ever does

- Attempting to run code with older graphics card (NVIDIA Quadro 2000)
    - Initially installed newer version of Ubuntu (22.04) but could not interface with the necessary version of CUDA without installing incompatible drivers for this graphics card and breaking the Linux build leading to reinstalls
    - Reinstalling an older version of Ubuntu (16.04) led to the graphics card working but many issues with python 3 and CUDA which are essential to code function
    - This graphics card was as old as the one used by Rajendran (Tesla GPU) but it did not have the same capabilities
    - The simulations also timed out at certain points because the kernal was running for more than 5 seconds for operations
    - <u>**Don't waste your time**</u> as a dead end will be reached going this route with the code compiling but the actual simulation unable to work

- Running code with NVIDIA GeForce GTX TITAN and GeForce GT 710 (Compute Capability 3.5)
    - This is the current set up with which all the rest of the simulations are running on
    - Steps to complete set up for this is decribed in the beginning section
    - GT 710 is used to run Xorg which is the code running the GUI on Linux and the GTX TITAN is used to run all the calculations so that the kernel watchdog timer is not affected thus leading to no time out in calculation unless due to memory errors
    - Both GPUs were compatible on the same drivers but it is the minimum necessary for the newer version of Ubuntu (20.04) to run the newest CUDA toolkit

- Using default simulations setups with sample NRRD and sample parameters
    - Visually shows no shift and this is confirmed by DaVis
    - Scaling up the gradient density by 10 – 10000x still yields no result
        - This was done by reading the "sample-density.nrrd" and multiplying each value by a scalar and rewriting the file to "test-density.nrrd"
        - The script for this is "nrrdread.py"
    - Analyzing the binaries with "ray_compare.py" seem to show extremely small shifts around less than a pixel
        - Shows "nan" for the first 2000 coordinates when scaled up very high

- Manipulating simulation parameters
    - Implement_diffraction = False → Does not have much effect on the final images
    - Intensity_rescaling = True → Keep true in order to see dots on final image
    - grid_point_number – changes number of dots for the background
    - grid_point_diamter – changes the size of the dots
    - X_Min, X_Max, Y_Min, Y_Max – appears to also effect the number of dots
        - +- 5e4 seems to generate the most dots
        - Other effects of this are unknown
    - generate_bos_pattern_images = True → Leave true or the simulation won't work
    - Image writing parameters
        - image_directory: the filepath to which all of the image and calculations will be saved
        - crop_image: throws key error when set to true

        - save_lightrays: useful if desiring to analyze binaries for results of the ray tracing and see why a shift might not have happened due to the calculations
            - Turn off for most cases as it is about 16x slower when trying to save the shear amount of data
        - save_intermediate_ray_data: Throws memory error when set to true

- Utilizing "create_nrrd_file_02.py" to generate own gradient files with the help of "createNRRD.py" for easier manipulation of function variables
    - Can create density gradients in functions of linear, quadratic, Gaussian, and error function (erf)
    - A variety of situations were simulated using the Gaussian function and their results will be displayed below in the appendix A

Potential Issues

- CUDA Error Launch Timeout (code = 6)
    - Cause: Kernal execution time of only 5 seconds allowed for GPU connected with display
    - Multiple solutions: https://nvidia.custhelp.com/app/answers/detail/a_id/3029/~/using-cuda-and-x
        - Use two GPUs (one for calculations and the other one for display)
        - Boot computer without GUI or display
            - Utilize SSH or remote desktop to boot into PC and run all commands and operations through terminal from remote PC
        - Optimize code for faster calculations and operations on kernel
        - Change settings mode in NVIDIA Driver with xorg.conf files
            - Not recommended as I broke the Linux build a couple of times
- CUDA Memory error
    - Cause: Parameters or density files mismatch in calculations
        - Occured for some of the other density files I tried to use from the Purdue repository
    - Solution: Edit your paramters or density (variable depending on the situation)

Current Conclusions on the Code

While not much success was achieved with the code from my end, the paper for which the code was used for showed some promising results that we had hoped to replicated. After many hardware issues, as described in the results section, we were able to get simulation results. The sample that the author provided appeared to show no worthwhile results as original and result pictures showed no shift. Even after correspondance with the author of the paper (Appendix B), the analysis into the actual ray tracing binaries showed shifts that were less than a pixel. Perhaps this analysis could be redone to ensure that this is indeed the result as I could have made some errors.

The program was able to generate certain images following the gaussian density gradients, but the results were not what one with knowledge of BOS would expect. At certain points, the simulation results were broken, such as images in which there was only a diagonal quadrant shift that was produced with a Gaussian density gradient. More investigation into this is necessary, such as changing the volume or decoding the coordinate system used in these simulations.

With more instructions or better documentation from the author of the code, the processes may have been easier to figure out. However, much of the current knowledge comes with experimentation on the code and some incorrect assumptions may have been made. There may also be parts of the code that have not been working at all, but that is unknown due to no error messages. With more empirical testing or reverse engineering of the code, results more similar to the paper could be produced. Although, it can be said with certainty that the program does not work in a variety of cases and this boundary of which cases it works and which ones it doesn’t should be documented for the future.

Appendix A

- Initial observatons (no picture saved) with a Gaussian distribution with settings: X and Y min = 0, x and y max = 0.5, nx = 100, ny = 100, nz = 200, z min = 0, zmax = 5
  - Variations of changing z_object or delta_x and delta_y do not seem to affect results
  - Changing min and max for x and y while keeping either min or max at zero seems to shift the quadrant in which a change can be noticed.
  - Pretty bad documentation for initial testing
- Playing with coordinate system: +-5e4 for X and Y min and max in simulation parameters, with grid point number of 250 for x and y, z_object = 10
  - Not every image is saved. Saved images are in sample-data/bos/images/Trial TIF folder
  - With first gradient volume settings: linear does not work even with pixel displacements (15, 2) changed
  - Quadratic: shifts entire image to the right (pixel displacements reverted and blur factor of 2 with same settings as before)
  - erf: same as quadratic apparently
  - Even for scaled up versions of gaussian, certain volume values do not work to create any change

Table 1. Observations for abov settings

X (Min, Max)	Y (Min, Max)	Z (Min, Max)	Number of elements	Result / Observation
-1, 1	-1, 1	-1, 1	200 (300)	Zoom out of entire picture. Changing increasing z values results in greater shift (Z_min = -5, Z_ max = 5) while keeping xy min max the same
-0.25, 0.75	-0.25, 0.75	-1, 1	200	Diagonal shift from top left to bottom right of entire picture
-0.75, 0.25	-0.75, 0.25	-1, 1	200	Diagonal shift from bottom right to top left of entire picture
-0.5, 0.5	-0.5, 0.5	-1, 1	200	Slight zoom out of entire pictures
0, 1	0, 1	-1, 1	200	Diagonal shift from top left (TL) to bottom right (BR) of top left quadrant and bottom right quadrant
-1, 0	-1, 0	-1, 1	200	Diagonal shift from bottom right to top left (only BR and TL quadrants with most movement in TL)
-1, 0	0, 1	-1, 1	200	Diagonal Shift TR to BL with most movement in BL Quadrant (TR and BL quadrant move only)
0, 1	-1, 0	-1, 1	200	Diagonal shift BL to TR with most movement in TR Quadrant (TR and BL Quadrant move
0.25, 0.75	0.25, 0.75	-1, 1	200	No Changes

-1, 1	-1, 1	0, 1	200	No Changes
-1, 0	-1, 0	0, 1	200	LR Quadrant shift towards TL
0, 1	0, 1	0, 1	200	TL Quadrant shift towards LR

- Empirical analysis of Gaussian Density simulations
  - Basic Settings: array cube dimensions, z_object= 50, pitch = 17 um
  - images located in sample-data/bos/images/Trial TIF 2 folder

Table 2. Another Table of Observations

Volume	Number of Elements per dimension	Peak Density	Std	Observation/file Name
[-1,1],[-1,1],[-1,1]	200	5	1	test1
[-1,1],[-1,1],[-1,1]	200	5	0.5	test2
[-1,1],[-1,1],[-1,1]	200	5	0.1	test3
[-1,1],[-1,1],[-1,1]	200	5	0.05	test4
[-1,1],[-1,1],[-1,1]	200	1	0.05	test5
[-0.5,0.5],[-0.5,0.5],[-1,1]	200	1	0.05	test6
[-0.1,0.1],[-0.1,0.1],[-1,1]	200	1	0.05	test7
[-0.5,0.5],[-0.5,0.5],[0,1]	200	1	0.05	Test8 – Accidentally overwritten but the zoom happened in opposite direction
[-0.5,0.5],[-0.5,0.5],[-0.5,0.5]	100	1	0.05	test9
[-0.1,0.1],[-0.1,0.1],[-0.5,0.5]	100	1	0.05	test10
[-0.1,0.1],[-0.1,0.1],[-0.01,0.01]	100	1	0.05	test11
[-0.1,0.1],[-0.1,0.1],[-0.01,0.01]	100	1	0.5	No changes – did not save
[0,0.1],[0,0.1],[-0.5,0.5]	100	1	0.05	test12
[0,0.1],[0,0.1],[-0.5,0.5]	150	1	0.05	test13

- Trial TIF 2 folder also contains interesting results produced by playing with the Gaussian distribution
  - gaussian_xy_0.05- changed the standard deviation of the Gaussian which produces a rounded zoomed out image
    - Have not put this in DaVis but may be worth doing as it seems to show a non-uniform zoom, which is different than the other results

- 25peakGaussianzoom – peak density scaled to 25 * ambient density
- 10peakGaussianZoom_negativeZ_only – peak density scaled to 10 * ambient density with z min in negative and z max at 0
- 10peakGaussianZoom – Same as above but with z max at a positive number

Updated 07.12.2022

Appendix B

- Email Correspondance with Rajendran
- Located on github with extra files as needed