7/7/16

Fluid Simulator/DFT Image Processing Program Summary

***Fluid Simulator***: This tool animates the evolution of density through a dynamic velocity field to simulate realistic fluid flow (in this case, rising smoke). The solver models fluid state as a velocity vector field and smoke as a continuous density function in which the value at each grid cell corresponds to the number of particles present. Both density and velocity are defined at cell centers. Principal components of the solver include a routine for density diffusion, density advection/velocity self-advection, pressure projection, and treatment of boundary conditions. Both density and each component of the velocity field are represented by two arrays each, the values of which undergo a repeated swapping/updating procedure in the animation loop. Every iteration of the loop calculates changes in density over a single time step and visualizes the updated state. Solver algorithms run in a reasonable time for grid sizes around 100-200. Notably, the grid allocates an additional layer of wall cells to simplify treatment of boundaries, so users should choose N for target grid size (N + 2) \* (N + 2). Visualization methods include real-time animation in a separate window (Automatic graphics backend) or a series of stills printed in the console (Inline graphics backend). The latter includes an option to save array files as the solver computes them, which themselves can be animated at a later time with the code provided in “Commented\_\_Load + Animate Saved Arrays.” Users can modify backend settings in Spyder by going to Tools>Preferences>IPython Console>Graphics.

***High-Res Rendering with Fourier Transform Image Processing***: Employs routines of the regular fluid dynamic solver, but sequentially incorporates saved low-res density array data to dictate the overall behavior of a higher resolution simulation. Each iteration runs a high-pass filter on the present density array and a low-pass filter on the corresponding low-res array. The sum of remaining density elements from both filters is passed in for the subsequent iteration’s density state. Filters employ a 2D Discrete Fourier Transform (DFT) to attenuate high or low frequencies (cutoff frequency defined by radius r) in the Fourier spectrum before reverting to a modified image in the spacial domain. Ideally, the solver preserves desirable overall behavior obtained in a low-res environment but renders the simulation with the detail of a larger grid size. Note: optimal animation involves uploading saved array data with the program below, since real-time animation generally proceeds inconveniently slowly.

***Load & Animate Saved Arrays***: Animates saved array files from a previously run simulation. Useful for producing an animation that progresses at a reasonable speed when real-time animation exhibits inconvenient lag.

**“advection2” module: Instructions for compiling Cython extension:** The fluid simulator imports advection and interpolation routines from a Cython extension module to expedite program execution. “advection2.pyx” contains the original Python source code and “setup.py” is the associated setup file. To build the Cython file, navigate to the appropriate directory in the command line and enter “python setup.py build\_ext --inplace”. A number of files should appear in your local directory, including the C source file and a .so or .pyd file depending on whether you’re working in unix or Windows, respectively. The fluid simulator imports the latter file like a regular python module, and the necessary import statement is already included in the simulator’s initial few lines. For further details, see <http://docs.cython.org/src/tutorial/cython_tutorial.html> (basic Cython tutorial)

All script files were written in Python and developed in Spyder, a scientific IDE for the Python language.