Installation

Prerequisites

Prior to the installation of OASIS, a few requirements must be met. Most importantly, Python has to be installed on the machine.

While both 32-bit and 64-bit versions are supported, a 64-bit Python installation is strongly recommended, as the double precision floating point image arrays take up huge amounts of memory. For rendering images with resolutions exceeding 8 Megapixels, a 64-bit system is inevitable.

OASIS also requires a suite of Python libraries to run. The list below shows all Python packages that must be installed and accessible from the desired OASIS installation path.

- Cython
- NumPy
- SciPy
- OpenEXR
- OpenCV
- Imath

Furthermore, the installation of the open source version control system Git is recommended, but not mandatory.

Download and Installation

The most convenient installation method is cloning the OASIS repository from Github. With Git installed on the machine, the desired installation directory should be navigated to in a command prompt. Then, the following command clones the repository to the local hard drive.

```
git clone https://github.com/maxbhr/OpticalAberrations.git
```

The pull command below should be run to ensure that the local master branch is up to date.

```
git pull https://github.com/maxbhr/OpticalAberrations.git master
```

Alternatively, a zip-file containing the code can be manually downloaded from `https://github.com/maxbhr/OpticalAberrations` and then extracted to the preferred installation path on the machine.

Now having all required files on the hard drive, the root folder of the installation should be navigated to in a command prompt to compile the Cython code. This is done by running the following Python command.

```
python setup.py build_ext --inplace
```

If the necessary libraries are found on the machine, the installation is successfully finished after this step.

Running simulations

Preparing the input files

For the input images, OASIS supports a variety of popular file extensions, listed below.

- jpg / jpeg / JPG / JPEG
- png / PNG
- exr

Those files must be RGB images. If a black-and-white image is used as input and stored as a grayscale with only a single color channel, it must be converted to a true RGB image first. This requirement applies to any of the chosen file types. When using EXR files, the color channels must be designated with 'R', 'G' and 'B'.

OASIS can render many images in a sequence with the same simulation parameters, as long as they are located in a common folder.

If a custom lens file is supplied, it must match the same file type criteria mentioned for the input images above. Additionally, it should be a square image, meaning equal width and height. This lens file should ideally not be placed in the same directory as the input images, to avoid it being used as input file.

Modifying the parameter file

The camera model and file management, as well as render and performance settings can all be adjusted from a single parameter file provided in the root directory of OASIS. By default this file is named params.txt and should not be renamed, unless the Python code is modified accordingly.

In the following, each section of the parameter file will be explained in detail.

```
# general
input_folder = C:/Users/name/documents/OASIS/input
output_folder = C:/Users/name/documents/OASIS/output
lens_file = C:/Users/name/documents/OASIS/lens.jpg
output_type = exr
monochromatic_sensor = 0

# render dimensions
x_min = 0
x_max = 0
y_min = 0
y_max = 0

# sample size
samples = 100
```

```
16      exposure = 1
17
18      # optical aberrations
19      aberration = coma
20      aberration_size = 4
21      chromatic_aberration = 4
22
23      # image noise
24      dark_current_noise = 2
25      readout_noise = 4
26      shot_noise = 1
27
28      # distortion
29      simulate_distortion = 1
30      fx = 6724.33
31      fy = 6724.33
32      cx = 995.163
33      cy = 970.568
34      k1 = 0.0256868
35      k2 = -1.77445
36      p1 = -0.00152939
37      p2 = -0.00046356
38      k3 = 13.7814
```

In the first three parameters, the preferred locations for the input and output folder are specified, along with the lens file. The directories must be set as absolute file paths or relative from the OASIS root directory. Is is important to note, that the first two parameters, `input_folder` and `output_folder`, must be set to a directory, as their name suggests, while the `lens_file` parameter must be pointing to an image file, as shown in the default configuration.

If no lens file is supplied, this value must be set to `0`. In this case the aberration intensity will vary with a pixel's distance from the image center.

The `output_type` parameter defines the file type for all exported images in the current folder. It can be set to any of the following file extensions.

`jpg, jpeg, JPG, JPEG, png, PNG, exr`

The `monochromatic_sensor` parameter defines whether or not to model a monochromatic camera sensor. This setting does not depend on the input images. To simulate a monochromatic sensor using the average method, this parameter should be set to `1`. Alternatively, OASIS models the weighted average method when the parameter is set to `2`.

In lines nine to twelve, the desired render dimensions are specified. By default every value is `0` which renders the entire image area but when dealing with high sample sizes on lower end machines, it might be advisable to test certain settings on a smaller image area first, before applying it to the entire image. Setting up those values is straightforward, as the origin is the top left image corner. For example, a configuration of `x_min = 0`, `x_max = 200`, `y_min = 0`, `y_max = 100` would render the top left image portion with a width of 200 and a height of 100. Naturally,

the render dimensions must be positive integers with maximum values of $width - 1$ and $height - 1$. Render dimensions only influence the simulation of point spread functions and image noise. Other aberrations remain unaffected by those four parameters.

The sample size is modified by the `samples` parameter, which must be a positive integer. This value defines the average number of samples per pixel, not per image. This ensures that shot noise remains constant, even when image dimensions change. Modifications to this value have the strongest impact on computation time.

The exposure compensation can be changed in line sixteen with the `exposure parameter`. Doubling this floating point value, produces an image that is twice a bright as the input image but there is no practical use of modifying its default value of `1`. If a change in exposure is desired, it is more advisable to choose `exr` as output file type and adjust the exposure level in post production.

In line nineteen, the `aberration` parameter defines the type of PSF modelled by OASIS. This string value can be set to one of the following options.

`coma, astigmatism_sagittal, astigmatism_tangential`

The maximum strength of the optical aberration is then defined by the `aberration_size` as a double value. By default this maximum is reached at the image edges, but when a lens file is provided, this value will be applied in regions of maximum brightness on the lens file. Its unit has been scaled to match the OPIC sensor resolution, so when rendering a 2048x2048 image with the default value of `4`, then the coma length will be 4 pixels at its maximum. This parameter is tied to the image dimensions, meaning a 1024x1024 input would produce a visually identical result with the coma length adjusted to 2 pixels. For astigmatism, aberration strength defines half of its length, in pixels. By default, a positive value will produce the more common external coma. Setting the size to a value smaller than 0 generates internal coma.

In line 21, the amount of chromatic aberration is specified. The `chromatic_aberration` parameter CA is a double value and it has the following relationship to the scale factor s of the red R and blue B color channels.

$$s_R = 1 - \frac{CA}{4000} \qquad\qquad s_B = 1 + \frac{CA}{4000}$$

Above equation represents the chromatic aberration properties for most camera lenses. However, a small number of lenses magnify the red channel the most and would therefore produce a different output image. Inverted magnification is modelled by OASIS, when the `chromatic_aberration` parameter is set to a negative double value.

The next section in the parameter file marks the modification of image noise. `dark_current_noise` is defined in line 24 by a double value. This value is internally divided by 1000 and then used as the standard deviation for the Gaussian noise generation, in a linear color space ranging from 0 to 1. An identical procedure applies to the `readout_noise` parameter.

Shot noise is modelled in a different way. The `shot_noise` parameter in line 26 is an integer that can either be set to `0` or `1`, while the actual amount of noise is solely dependent on the sample size parameter mentioned above. Disabling `shot_noise` improves render times noticeably, as only half as many random numbers are generated.

The final section is reserved for distortion settings with the initial `simulate_distortion` parameter in line 29 turning the distortion simulation either on or off by setting its integer value to `1` or `0`. In the next four lines, the camera model is defined, where `fx` is the focal length in x-direction, and `fy` is the focal length in y-direction. Both values expressed in pixel units. `cx` and `cy` define the principle point in pixels, which is ideally the center of the sensor. The distortion coefficients `k1`, `k2`, `p1`, `k2` and `k3` are found in the last five lines of the parameter file. The camera model and distortion coefficients are filled with OPIC's calibrated parameters by default.

Running a batch job

Once all parameters are set and the parameter file saved, a simulation can be run. To start the batch job, the following command should be run in a command prompt after navigating to the root directory of OASIS.

```
python batch_run.py
```

Depending on the parameter settings, the console output will look similar to the lines shown below.

```
Image 1/6: opic_test.jpg
Reading image file:   100 %
Reading lens file:    100 %
Applying chromatic aberration: 100 %
Distortion applied.
Applying coma with shot noise: 100 %
Render time: 00:04:42
Dark current noise applied.
Readout noise applied.
Writing file:  100 %
Total time: 00:04:56
```

The first two and last two lines will always be displayed, whereas the content in between shows a real-time progress of the optical aberrations simulated to the image. While the total computation time is always presented at the end, the render time is only visible when a point spread function or shot noise is simulated. In the output folder, images are stored with modified names. The example input shown above would create the following file in the output folder.

```
opic_test_rendered_YYYYMMDDHHMMSS.exr
```

Where `YYYYMMDDHHMMSS` is replaced with the current date and time.