EXR Stitcher

EXR stitching tool

We would like you to create an EXR image stitching tool that can be used to stitch together multiple EXR images into a single output EXR image.

It should be a command line the tool that is invoked by providing a folder containing the input images and the path where the output file is expected to be placed:

```
$ python exrstitch.py exr-input-folder exr-output-file-path
```

The tool should only pick up EXR image files from the input folder that follow the naming convention: `u#_v#.exr` or `u#_v#_#.exr` (where # denotes whole numbers or positive non-zero integers). All other files should be ignored. The file names inform the tool about the configuration of the grid that is to be expected in the output image e.g. if you have the following files in the input folder: `u1_v1.exr, u1_v2.exr, u1_v3.exr`, the tool will generate the output image by stacking the input images one after the other in a horizontal line.

While if the input folder had the following files: `u1_v1.exr, u1_v2.exr, u2_v1.exr, u2_v2.exr`, then the tool will generate a grid of `2 x 2` cells.

We should be able to optionally add multiple images for a given grid cell. In case there are multiple images for a given cell, the file names for that cell would follow the u#_v#_#.exr naming convention. In this case the ending number in the file name would specify a sort of z-index to stack/paste/merge the images onto each other in ascending order. So for example if the input folder contains the following files for the same cell: `u3_v4_1.exr, u3_v4_2.exr, u3_v4_3.exr` the output image in that part of the grid should be the result of pasting `u3_v4_3.exr` on to `u3_v4_2.exr` and then pasting that on to `u3_v4_1.exr`. Of course you're expected to do some optimization here and not literally paste/overwrite the pixels on to each other. Note that only the first file for that grid cell needs to adhere to the resolution (width and height) requirements for that cell, the other files can be of smaller resolution but not larger in its respective dimension.

Salient Features

- The application should have a tiny memory footprint. This can be achieved by writing out the output image one scan line at a time. So as long as the filesystem supports it, it should be able to handle any size images or grids

- It should support both single channel and multi channel EXR images, of course all the input images need to have the same channel configuration
- Depending on the input image names, it should be able to generate a grid of any size e.g. `2 x 2` or `4 x 1` or `6 x 8` etc.
- Different rows can have varying heights and different columns can have varying widths, of course all the images for the same row need to have the same height and all the images for the same column need to have the same width. This is adequately illustrated by the images in the test folder
- Should have a validation step to identify any missing files or incorrect/corrupt input images before the processing actually starts

Installation

The tool should be written in Python 3.x

Please include a `requirements.txt` file, so it can be installed by running:

```
$ pip install -r requirements.txt
```

Test

The attached zip contains some test images which you can test your application against by entering the following at the prompt:

```
$ python exrstitch.py ./test1 ./test1_out.exr
```

Feel free to create or add more tests, especially for the case of multiple images per grid cell since that feature is not represented in the test images provided. Kindly submit any test data that you generate along with your application.