Image Hole Filling

Introduction

The goal is to build a small image processing module that fills holes in images, along with a small command line utility that uses that module. Note that while the required tool functionality now is very basic we expect more features to be added to the tool in the future, therefore flexibility in the design and good separation of concerns is something we care about.

The exercise has to be implemented in Python. You may use open-source libraries, provided that they do not implement hole filling logic.

Hole Filling Algorithm

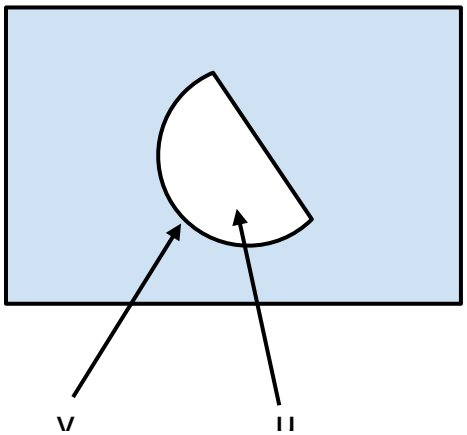
Definitions:

- I: the input image.
- $I(v)$: color of the pixel at coordinate $v \in \mathbb{Z}^2$.
- B: set of all the boundary pixel coordinates. A boundary pixel is defined as a pixel that is connected to a hole pixel, but is not in the hole itself. Pixels can be either 4- or 8-connected. See this for more info.
- H: set of all the hole (missing) pixel coordinates. You can assume the hole pixels are either 4- or 8-connected.
- $w(v,u)$: weighting function which assigns a non-negative float weight to a pair of two pixel coordinates in the image.

Algorithm:

Find the boundary $B$and the hole Hin a given image. For each $u \in H$, set its value to:

$$I(u) = \frac{\sum_{v \in B} \quad w(u,v) \cdot I(v)}{\sum_{v \in B} \quad w(u,v)}$$



Module implementation requirements

1. The default weighting function for the algorithm is $w_z(u, v) = \frac{1}{||u-v||^z+\epsilon}$, where ϵ is a small float value used to avoid division by zero, and $||u - v||$ denotes the euclidean distance between u and v. The values z, ϵ should be configurable.
2. The module needs to support any arbitrary weighting function.
3. The module needs to support both the 4- and the 8-connectivity options.
4. For simplicity, you can assume that every image has only a single hole.
5. There's no need to handle the corner case of holes that touch the boundaries of the image.

Command-line tool requirements

The hole filler should be a command-line tool that is executed with four arguments - the path to the image or images, a prefix for the mask image name, an output path and the configuration of the weight function.
For example:

```
> python hole_filler.py ./data/test.png mask_ ./output/test_filled.jpg
weight_conf.json
```

The tool should support paths to a specific image as well as to folders with multiple images.

It is assumed that for each image file, e.g. image.jpg, there is a matching image file for the mask using the same name and the added prefix, e.g. mask_image.png. The mask image should be a single channel image where any value larger than 0 is considered a hole pixel. The image formats may vary between image and mask.

The output path can be an explicit image name or a folder in case of multiple images. Give a reasonable solution for naming of multiple outputs.

The weight function configuration implementation should support the parameters of the default weight function and allow for any other configuration that may be supported by extensions to the underlying module.

[Bonus] Persistence

The tool should be able to continue filling from roughly the point it stopped, in case it has been prematurely stopped (by, for example, terminating the process). It is allowed to have a small amount of work re-done by the tool, but the vast amount of work should not be repeated.

[Bonus] Weight Caching

The tool should avoid, as reasonably as possible, recalculation of the weights for the same input parameters to the weight function. This should be persistent across runs of the tool.

Good luck!