GPGPU Assignment #3

Author: Yu Sheng Lin Instructor: Wei Chao Chen

March 28, 2016

1 Goals

You have to implement Poisson Editing for image cloning in this assignment.

2 Description

2.1 Image Cloning

Figure 1. shows possible inputs of image cloning algorithms: (a) is the background, (b) is the target image and (c) is a boolean mask. A very naive image cloning example is shown in Figure 2. It just copy and paste the image from the target image to the background image if the mask is true. The is just what we do in the sample codes we provide.

Of course we won't let you implement that in this assignment. You will implement Poisson Editing in this assignment, which will be described in more details later in this document.

2.2 Function Signature

The function signature of this assignment is Listing 1.. `background` and `output` are interlaced RGB images of size $W_b \times H_b$ and range from 0 to 255. `target` is the same except that it's size is $W_t \times H_t$. `mask` is the mask which only has one color channel and we use `0.0f/255.0f` to represent false/true.

```
void PoissonImageCloning(
    const float *background,
    const float *target,
    const float *mask,
    float *output,
    const int wb, const int hb, const int wt, const int ht,
    const int oy, const int ox
);
```

Listing 1: Function signature

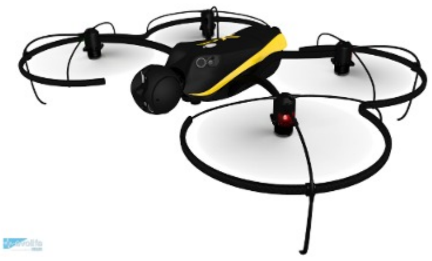
We will also assign the offset O_y, O_x, which means the offset of the target image in the background image (from the top-left corner). To generate the image shown in Figure 2. with the the sample code, run the program by Listing 2..

The image format we are using is the PGM/PPM format, which can be edited by many image processing softwares. You can generate new testcases if you desire.

```
./a.out img_background.ppm img_target.ppm img_mask.pgm 130 600 output.ppm
```

Listing 2: Execute your code

(a) The background image $W_b \times H_b$.


(b) The target image which will be pasted to the background, $W_t \times H_t$.

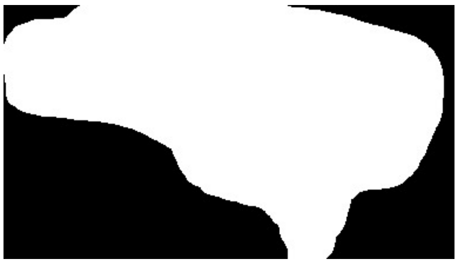

(c) The mask $W_t \times H_t$.

Figure 1: The input images of this assignment.

2.3 Poisson Editing

The 4-neighbor linear system is Equation 1. (also refer to Figure 3.), and if you are interested, please refer to the original paper of Poisson Editing.

$$4C_b - (S_b + E_b) = 4C_t - (N_t + W_t + S_t + E_t) + (N_b + W_b) \tag{1}$$

With the Jacobi Process, the iteration step is Equation 2..

$$C'_b = \frac{1}{4}\left[\underbrace{4C_t - (N_t + W_t + S_t + E_t) + (N_b + W_b)}_{\text{Fixed during iterations}} + \underbrace{(S_b + E_b)}_{\text{Current value}}\right] \tag{2}$$

It's your task to generalize (just by intuition) and figure out what if (1) the distrubution of gray points changes and (2) the point only has fewer neighbors.

If we further use the successive over-relaxation (SOR), the iteration step becomes Equation 3..

$$C'_{b,SOR} = \omega C'_b + (1 - \omega)C_b \tag{3}$$

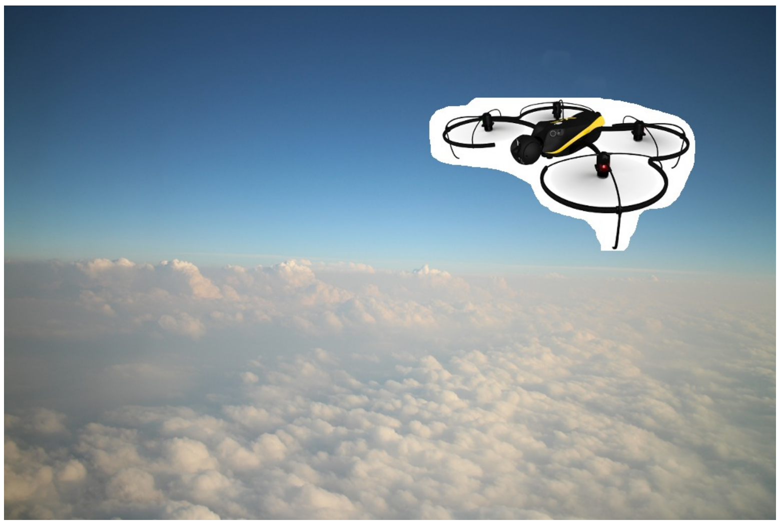
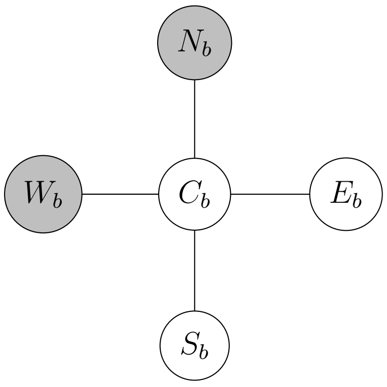
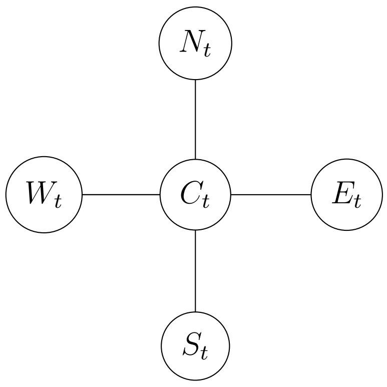


Figure 2: A very naive image cloning.



(a) The values to be solved.



(b) The corresponding target image.

Figure 3: The gray nodes are the boundary (the black pixels in the mask).

SOR is just a interpolation/extrapolation between current values and the values of the next iteration. Usually, we use $\omega < 1$ to ensure the convergence while $\omega > 1$ to accelerate the convergence. Also note that when $\omega = 1$, the SOR is exactly the original method. You could choose larger ω (maybe 1.2) initially and decrease it to 1 later after a few iterations.

Acute readers might notice that the time for a value to propagate from the boundary is just propotional to the distance from the boundary. This method will require thousands of iterations to converge, which is very impratical! A simple solution is to use the multigrid method: solve at lower resolution initially then upsample and solve again, and in this assignmnet 1/64x, 1/16x, 1/4x and 1x is acceptable (not tried yet).