**Blending**

We implemented a Poisson Blend as described in *Pérez et al.* to combine our target and source images. For our purposes, the Poisson Blend simplifies to: for each pixel in our blended region, the result of a Laplacian operator on that pixel should be equal to the Laplacian if the corresponding pixel in the source region. Thus, the gradient across the inserted object should be preserved as well as the gradient between that object and its original background.

We are able to calculate these gradients across each color channel using a system of linear equations. Each variable in the system of equations represents one pixel in a rectangle which contains the source object placed over the target background. Using a binary mask we obtained from our image segmentation, we determine which pixels will be taken directly from the background and which pixels are from the source object and need to be blended.

* For each blended section pixel *x*
  + If pixel is background: ***x* = *t***, where *t* is the corresponding pixel in the target image.
  + If pixel is blended object: **laplacian(*x*) = laplacian(*s*)**, where *s* is the corresponding pixel in the source image.

We are able to solve this quite large system of equations using matrix division. ***A\b = x*** where *A* is the matrix that represents the left side of the equations above and b is the vector that represents the right side. Since the matrices involve get to be quite large, but have at most five non-zero values in any given row, we create them using the sparse() command in Matlab. This stores the zero values in our matrices more efficiently and helps our blender run in a reasonable amount of time (as well as prevents it from crashing James’ computer if used incorrectly).

**Blending Results**

There are some circumstances where the blending does not produce seamless results as intended. For instance, if the source object is segmented such that there is too much of its original background blended, the result image will have sort of a blurry halo around the source object which is especially noticeable if the target background image has complex patterns. 

There is a blurry halo around Bucky's head since too much of the source was selected. The underwater plants make this blur much more noticeable.

Also, if something goes wrong with segmentation and the segmented object has holes in it, the holes will be filled by the background, and the object will appear semi-transparent. 

While this segmented tiger looks nice on its own, it will actually only select the orange parts of the tiger.



The tiger is a little more transparent than we intended.

**References**

* Poisson Image Editing by Patrick Pérez, Michel Gangnet and Andrew Blake

**Our Code**

Poisson Blender code – Roughly 80 lines.

Inspiration taken from:

* <http://cs.brown.edu/courses/cs129/asgn/proj2/>
* http://eric-yuan.me/poisson-blending/
* https://github.com/asteroidhouse/gradient-blend