Parallelization of Instagram Photomosaics and a Study of Photo Tile Reutilization

Stephanie Yeung (syeung1)
Tyler Hedrick (thedrick)

Background

- Utilizing a database of Instagram photos to generate photomosaics where each Instagram photo is a "pixel" in the original image
- Parallelized over finding image matches
  - Challenge in removing duplicates atomically
  - # subimages to give best results with least overhead?
- Also studied data compression through reusing previous image data for new images

Algorithm

1. Divide each photo into M x M subphotos
2. Divide each subphoto into N x N grid and average RGB values in each grid section
3. Do same for all images in database
4. Find closest match, $\sqrt{(r_1-r_2)^2+(g_1-g_2)^2+(b_1-b_2)^2}$

```
Load and slice image into M x M squares of equal size
    for each sub-image in parallel:
        store average RGB values of N x N grid pieces into an array
        for each stored photo in parallel:
            compare RGB values with grid and update current best match
        add photo pixels to shared final image buffer
```

Approach

- Serial implementation in C++
  - ImageMagick Magick++ Library
  - Custom LibJPEG Image Tiler
    - Given image path and index into final image, can easily parallelize writing matching images to array
- Parallelization in CUDA
  - Graphics Card, GEForce GTX 480:
    - SMs: 15
    - Compute CAP: 2.0
    - GPU Memory: 1536 MB

Results

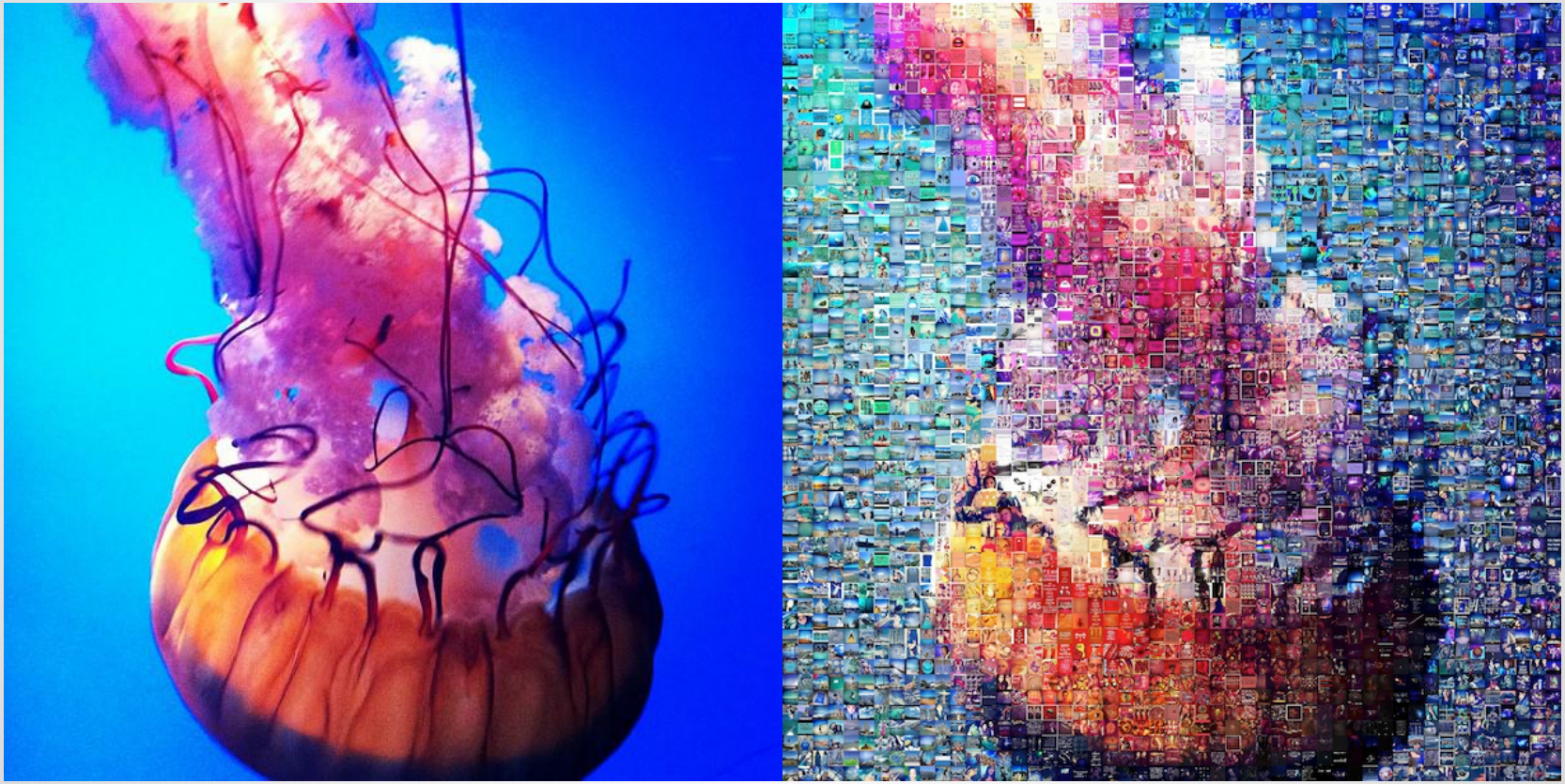
- 39.1x speedup w/ CUDA for 204x204 mosaic

	Serial	GTX 480 (15 SMs)	GTX 650 (2 SMs)	GTX 670 (7 SMs)	GTX 680 (8 SMs)
Find Matches (s)	1180.7	25.4	169.4	46.0	42
Total Time* (s)	1187.7	30.4	174.4	51.0	47
Speedup	1x	39.1x	6.8x	23.3x	25.3x

* Additional 3 seconds to load remote database of images and 2 seconds to tile (5 for serial)

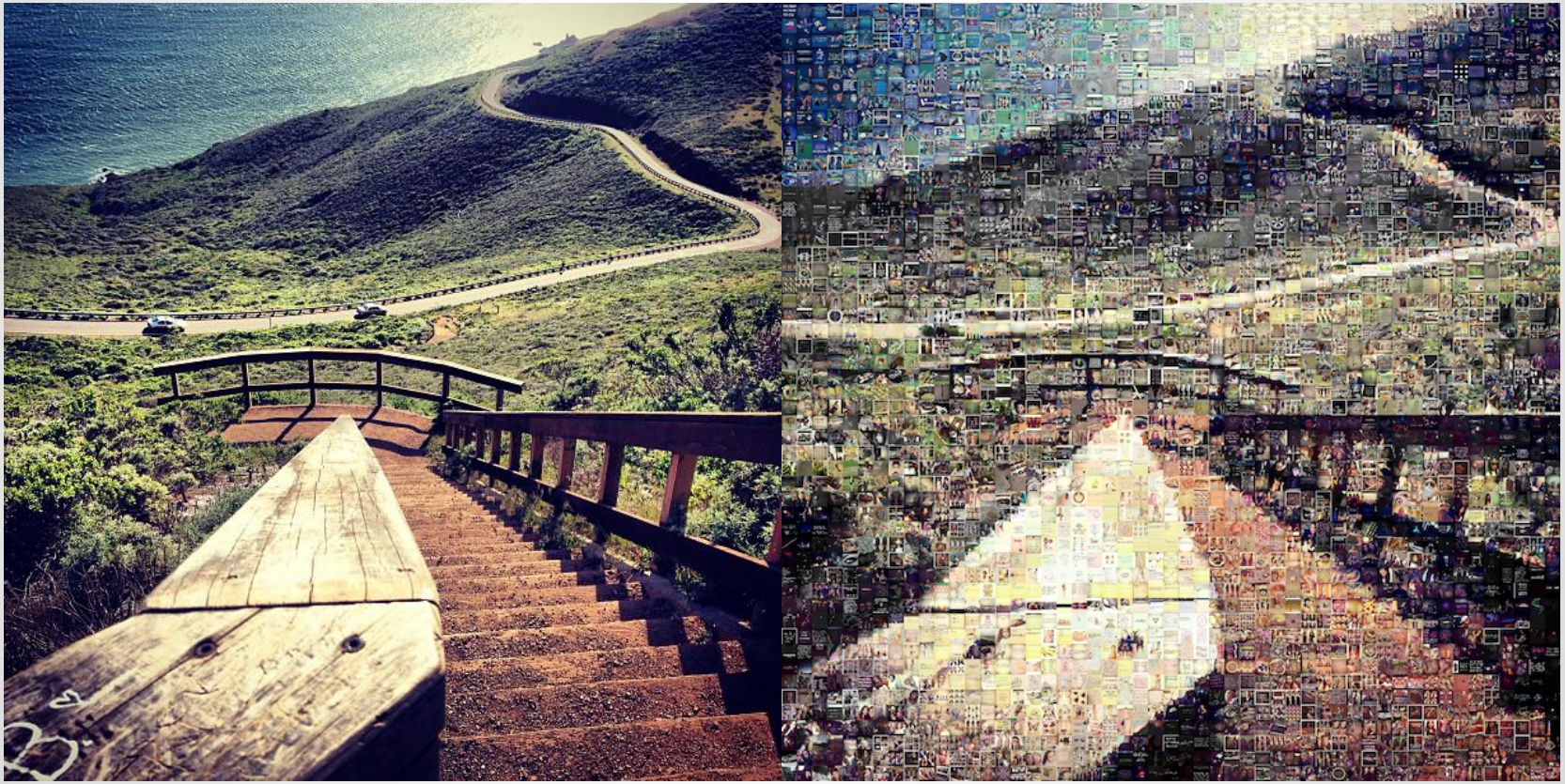
- 0-21.2% images for potential data reusage
  - Images with borders yield higher reuse potential

Results



102 x 102

Results



Results

Thanks!

Questions?