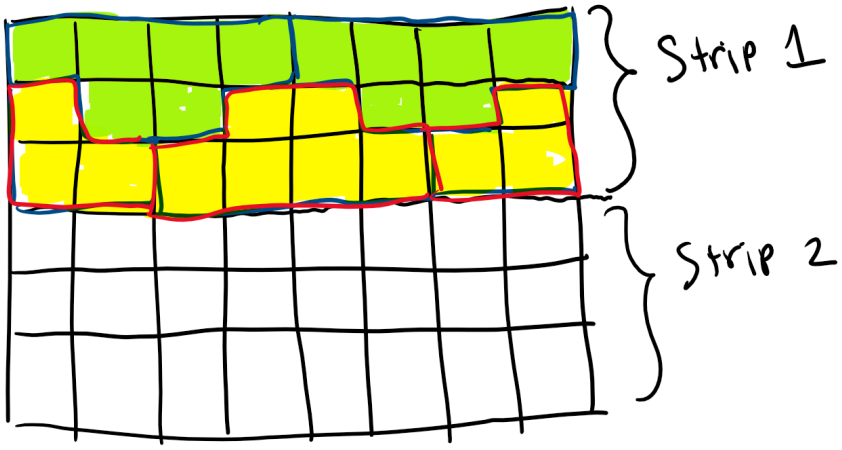
Project URL Link:
https://www.andrew.cmu.edu/user/jsliu2/seam-carving/418_project_milestone_report.pdf

Work we've completed so far

Currently, we have created a working parallel version for removing both vertical and horizontal seams from a given image, which parallelizes via OpenMP. We ended up deciding to stick with the source code we initially submitted from our proposal that uses C++ and OpenCV to create a sequential seam carver. Initially, we had a few hiccups when we realized that OpenCV was not available on the GHC machines. We tried building the libraries in Andrew but to no avail. We also briefly considered looking for other pieces of source code that didn't use OpenCV so we could leverage the GHC machines. However, we realized that if we wanted to use C++ as our programming language, then we needed OpenCV to help parse an image's pixels. We were then successfully able to install OpenCV locally and figure out how to compile the source code because the source did not include a Makefile or any documentation on how to compile the source code. We chose to parallelize via OpenMP and suspend the CUDA approach because neither of our laptops has an NVIDIA GPU and the GHC machines don't have OpenCV, so testing would have been very difficult.

In terms of how our code is parallelized, we first implemented the naive approach of parallelizing over the columns of a given row, as outlined in our proposal. This achieved moderate speedups of 15-20% on one processor, which was expected since we knew that this approach would have drawbacks. After exploring some other ideas, we settled on our current approach, which is as follows:



First, we separate the image into strips (with the example above being when we remove vertical seams). The height of these strips is flexible. We then iterate strip by strip, so Strip 1 is processed before Strip 2. Then we parallelize across all the green triangles in the strip and find the cumulative energy of each pixel in each triangle. After that's done we parallelize across all the yellow triangles in the strip and find the cumulative energy of each pixel in each triangle. Within each triangle, we go from up to down and left to right.

Describe how you are doing with respect to the goals and deliverables stated in your proposal. Do you still believe you will be able to produce all your deliverables? If not, why? What about the ”nice to haves”? In your milestone writeup we want a new list of goals that you plan to hit for the poster session.

We are running a little ahead of schedule with respect to the goals and deliverables stated in our initial proposal. We have accomplished everything initially set for Tuesday April 16, which is having a working parallel version of our Seam Carving code. Moreover, we have managed to optimize an initial parallel version that parallelizes by columns into our current approach that parallelizes by triangles. In terms of future work, we have set a JUMP parameter that determines how many rows compose a strip, and we are working on finding the ideal number for that as of now.

We decided not to write a correctness verifier for our parallel version because we are going to try trading off accuracy (exactly the same as the sequential version) with speed (having a slightly worse seam-carved image in favor of doing more work in parallel).

We believe that we will still be able to produce all of our deliverables because the only things left for our initial set of goals is to further optimize our parallel version of the starter code.

In terms of the “nice to haves”, we think that parallelizing our code to remove multiple seams at once is cool- it would be interesting to see how batched seam removals would affect how the carved image turns out. We would probably have to explore the tradeoff between speed and correctness.

Here is a new list of goals we plan to hit for the poster session:

- A parallelizable version of the DP algorithm implementation of a Seam Carver that carves both vertical and horizontal seams.
- A version of the DP implementation of a Seam Carver that carves both vertical and horizontal seams where the speedup from the sequential version is 8x.
  If we assume that we cannot achieve perfect speedup on the PSC machines and the PSC machines can employ at least 16 processors, then we think that achieving at least half of 16x speedup (8x) is reasonable.
    - We also want to find the ideal JUMP parameter for our parallelized triangle approach.
- Have GIFs that show the process of how images we provide get seam carved for an extra cool demo. We hope to have our code be able to create either GIFs that show the seam carving process or timestamped pictures of in-between stages that a user can click between on their own.

What do you plan to show at the poster session? Will it be a demo? Will it be a graph?

As a bare minimum, we will be showing pictures of original images versus their sequential seam-carved results and their parallel seam-carved results. We’re hoping to create GIFs of the seams being carved out to make the demo extra cool.

We will also be showing speedup graphs that compare the sequential version to the parallel version for both the naive parallel version as well as our optimized version with various JUMP parameters. For the graphs concerning our optimized version, we will have 1 axis for the number of rows (JUMP parameter value) and another axis for speedup.

Depending on the progress we make for batching, we also plan to have graphs for the speedups it gives, e.g. speedup as a function of the number of seams found in parallel, and also some images to assess how it affects the image quality.

Do you have preliminary results at this time? If so, it would be great to included them in your milestone write-up.

We have a few preliminary results. The following sets of measurements were made locally on a single processor.

For each of the following series, note that we are targeting the cumulative energy map time taken:

The first series is a purely sequential version of our code

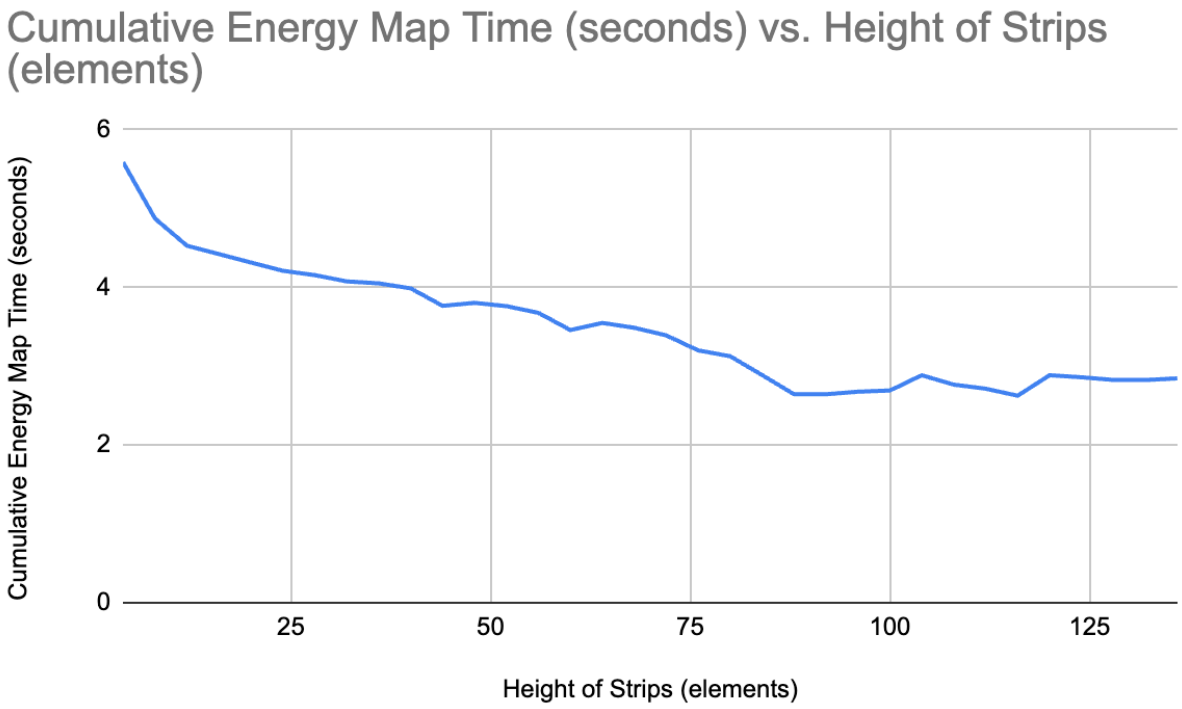
```
jonathanliu@Jonathans-MacBook-Pro-7 seam-carving % ./seam-carving
Please enter a filename: a.jpeg
Reduce width or reduce height? (0 to reduce width | 1 to reduce height): 0
Reduce width how many times? 300
Final image size: 720x980
energy image time taken: 0.8748593
cumulative energy map time taken: 6.3956809
find seam time taken: 0.0124854
reduce time taken: 0.1583089
total time taken: 7.6938739
```

The next series is our naive parallel version where we parallelized by columns.

```
jonathanliu@Jonathans-MacBook-Pro-7 seam-carving % ./seam-carving
Please enter a filename: a.jpeg
Reduce width or reduce height? (0 to reduce width | 1 to reduce height): 0
Reduce width how many times? 300
Final image size: 720x980
energy image time taken: 0.9385486
cumulative energy map time taken: 4.6876903
find seam time taken: 0.0294378
reduce time taken: 0.2096631
total time taken: 6.1067762
```

This last series is our optimized parallel version with a JUMP parameter of 90. We will be working on finding the best JUMP parameter in the next few days.

```
jonathanliu@Jonathans-MacBook-Pro-7 seam-carving % ./seam-carving
Please enter a filename: a.jpeg
Reduce width or reduce height? (0 to reduce width | 1 to reduce height): 0
Reduce width how many times? 300
Final image size: 720x980
energy image time taken: 0.9324683
cumulative energy map time taken: 2.9417882
find seam time taken: 0.0322188
reduce time taken: 0.2198520
total time taken: 4.3595161
```



(1 processor, 300 vertical seams removed on an image of size 720x980.)

List the issues that concern you the most. Are there any remaining unknowns (things you simply don't know how to solve, or resource you don't know how to get) or is it just a matter of coding and doing the work? If you do not wish to put this information on a public web site you are welcome to email the staff directly.

Right now, it is just a matter of coding and doing the work. Things look good on our end as of now, so we just have to find the best JUMP parameter and then further testing on the PSC machines. Since we've pivoted from both a CPU and GPU implementation to just a CPU implementation with the exploration of other avenues of parallelization, we should have all the resources available.

Updated Schedule

Week	Deadline	Task4(s)
5	Fri April 19	- Analyze the bottlenecks of the OpenMP implementation, e.g. memory bandwidth, cache utilization, and scheduling (Jonathan) - Fine-tune hyperparameters and obtain experimental data for speedup on images of varying size (Jonathan) - Run experiments to find the best JUMP parameter (Jonathan)

		- Test on PSC machines to obtain speedups on higher core counts (Abby)
5.5	Tues April 23	- Explore how to batch seams to remove and how to address the issue of intersecting seams (Abby) - Implement the batch approach (Jonathan)
6	Sat April 27	- Evaluate the different batch approaches (Jonathan) - Establish how we will parallelize finding multiple seams (Abby) - Collect data on what batch parameter works best (Jonathan) - Test on PSC machines to obtain speedups with batching alone and with parallel cumulative energy map (Abby)
6.5	Tues April 30	- Obtain visual results of seam carving: producing a GIF of the seams being removed in order, producing visuals for the energy map (Abby) - Begin project report (Abby)
7	Fri May 3	- Complete demo of the project (Abby + Jonathan) - Finish final report (Abby + Jonathan) - Begin poster design (Abby + Jonathan)
7.5	Sun May 5	- Finish poster and prepare presentation (Abby + Jonathan)