GPU Accelerated Seam Carving

Matt Sarett, Adu Bhandaru

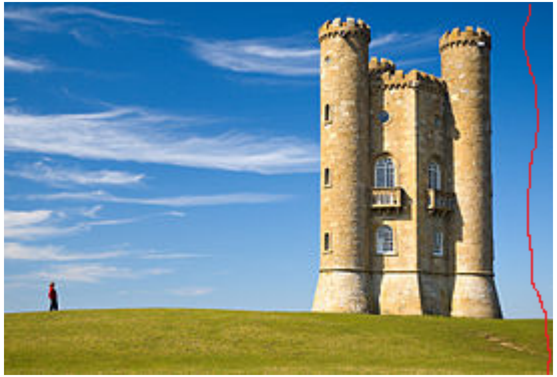
18-645 How to Write Fast Code

What is Seam Carving?

- Proposed by Shai Avidan, Mitsubishi Electric Research Laboratories.
- Content aware image resizing
  - Remove the pixels of *least importance*

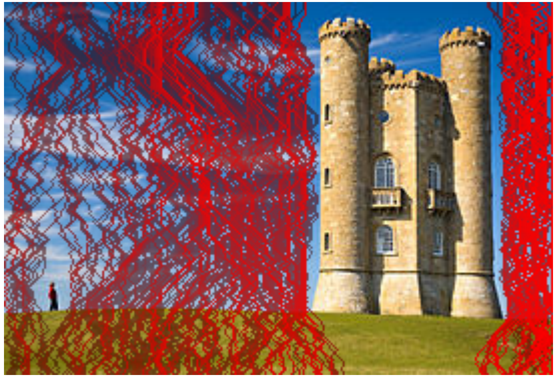
What is Seam Carving?

Original

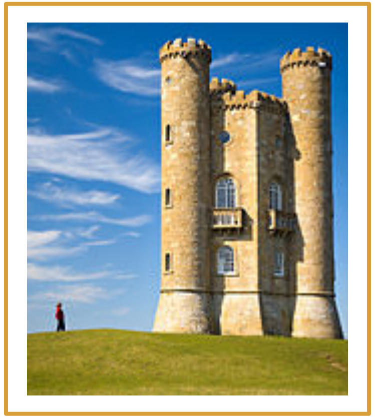


1 Seam

What is Seam Carving?



50 Seams



Finished

Previous Work

- Adobe, iResizer, ImageMagick, …
  - Mostly closed source
- Parallel computation of *importance* (energy)
  - Jacob Stultz (MIT) using multi-threading
  - Thread inter-dependence overhead

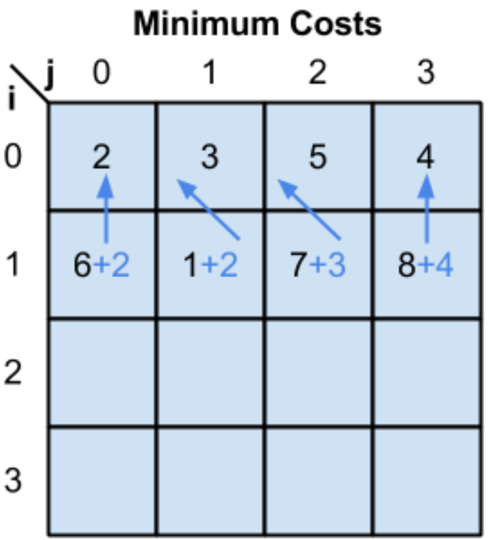
Algorithm

- Compute pixel energies (importance)
- Use dynamic programming to generate a minimum cost table.
- Backtrack to find the cheapest seam.

Algorithm

Pixel Costs

i j	0	1	2	3
0	2	3	5	4
1	6	1	7	8
2	2	7	1	2
3	10	6	7	8

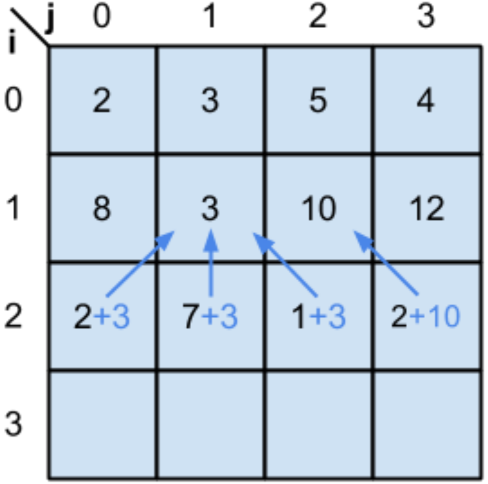


Minimum Costs

i j	0	1	2	3
0	2	3	5	4
1	6+2	1+2	7+3	8+4
2				
3				

Algorithm

2nd Row Iteration

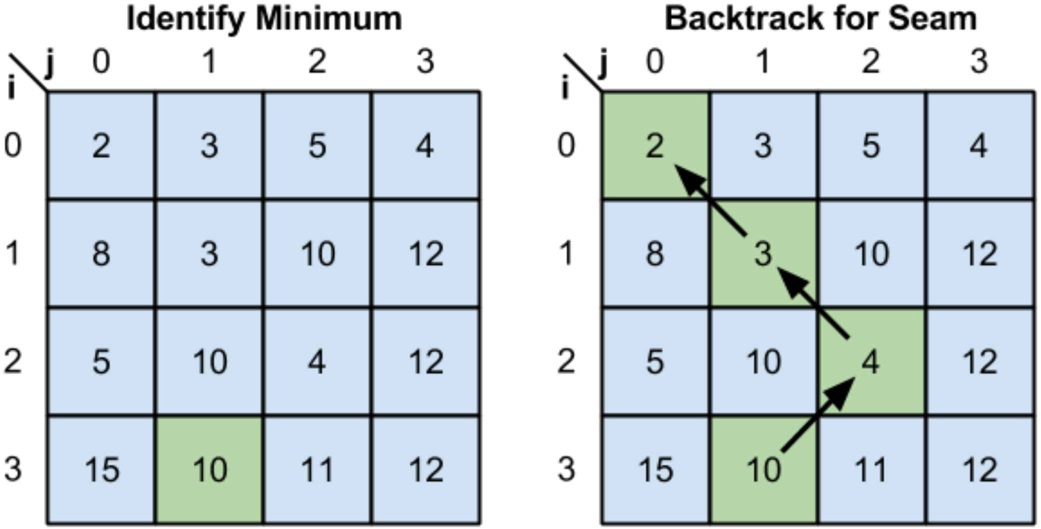


i \ j	0	1	2	3
0	2	3	5	4
1	8	3	10	12
2	2+3	7+3	1+3	2+10
3				

Finished Min-Costs

i \ j	0	1	2	3
0	2	3	5	4
1	8	3	10	12
2	5	10	4	12
3	15	10	11	12

Algorithm



Identify Minimum

i \ j	0	1	2	3
0	2	3	5	4
1	8	3	10	12
2	5	10	4	12
3	15	10	11	12

Backtrack for Seam

i \ j	0	1	2	3
0	2	3	5	4
1	8	3	10	12
2	5	10	4	12
3	15	10	11	12

Energies on the GPU

- Energy table computation is embarrassingly parallel

$$E(i,j) = [I(i,j) - I(i+1,j)]^2 + [I(i,j) - I(i+1,j)]^2$$

- One thread for each pixel
- Utilization of shared memory

Minimum Cost on the GPU

- The cost of each pixel depends only on its energy and the cost of the above pixels

```
C(i,j) = E(i,j) + min(C(i-1,j-1), C(i-1,j), C(i-1,j+1))
```

- One thread for each column
- Utilization of shared memory

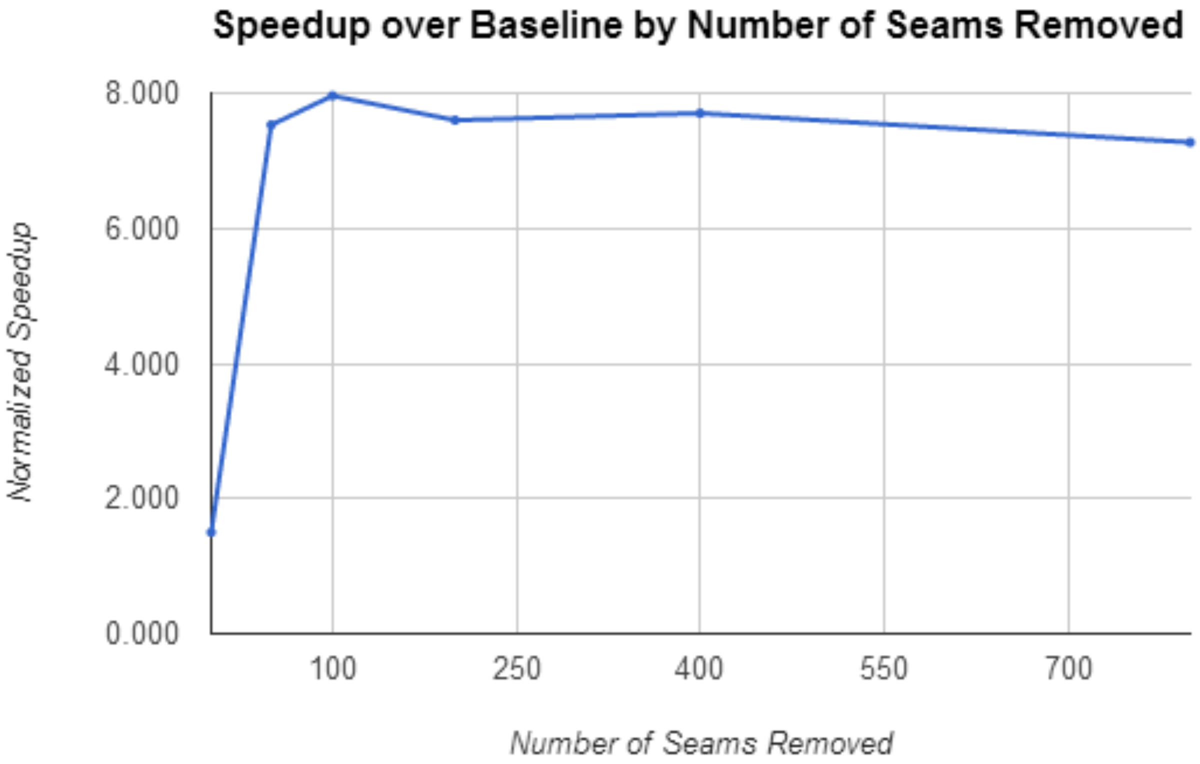
Minimum Reduction on the GPU

- This is accomplished very similarly to the reduction in k-means
- Threads compare values in parallel until we reach a single minimum
- If the width is greater than 1024, the final steps of the reduction are performed on the host

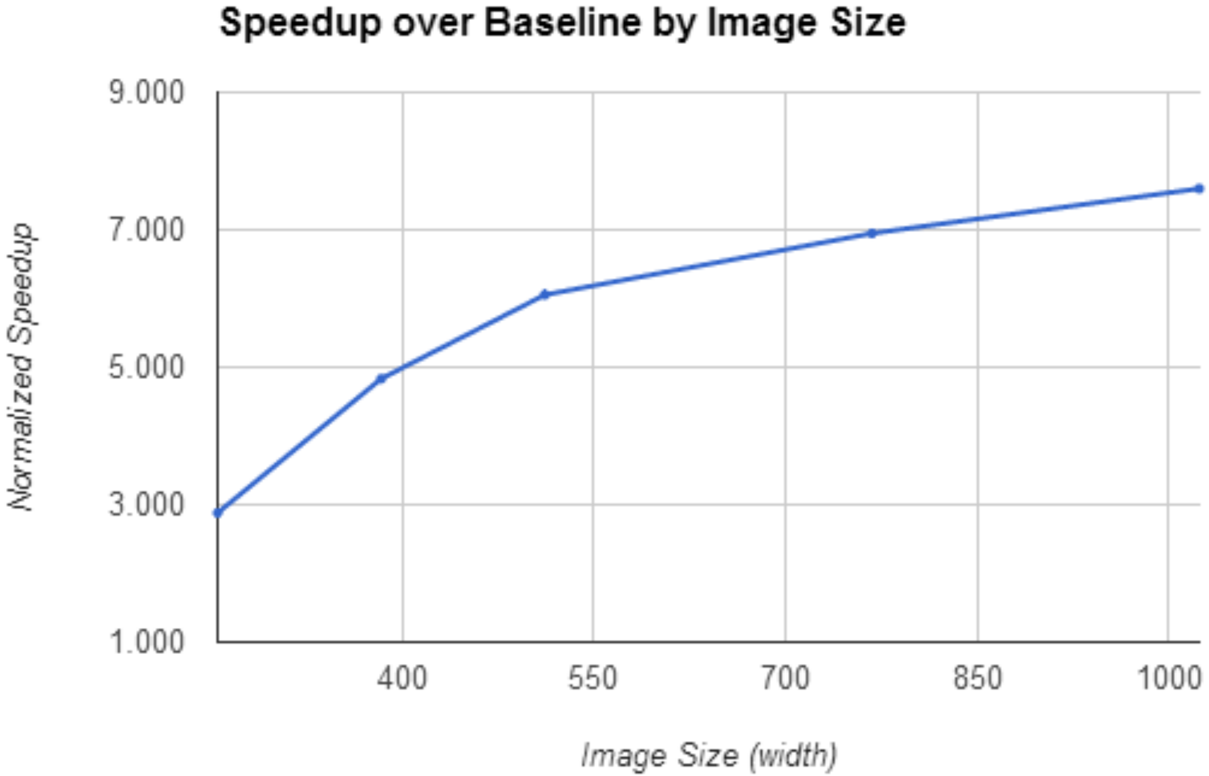
Evaluation Methodology

- Sequential baseline implementation in C++
- Diff the output images for correctness.
- Performance measured using `ctime` on the Gates cluster machines.

Results


Speedup over Baseline by Number of Seams Removed
8.000
6.000
4.000
2.000
0.000
Normalized Speedup
100
250
400
550
700
Number of Seams Removed

Results


Speedup over Baseline by Image Size
9.000
7.000
5.000
3.000
1.000
Normalized Speedup
400
550
700
850
1000
Image Size (width)

Further Work

- Maintaining the energies and minimum cost tables in GPU memory would eliminate the need to copy back and forth
- After a seam is removed, only a partial recomputation of energies and minimum cost is required

Source Code

The source code can be found here:

https://github.com/abhandaru/gpu-seamcarving