Seam Carving

Konwoo Kim

Contents

Contents

Quick Background

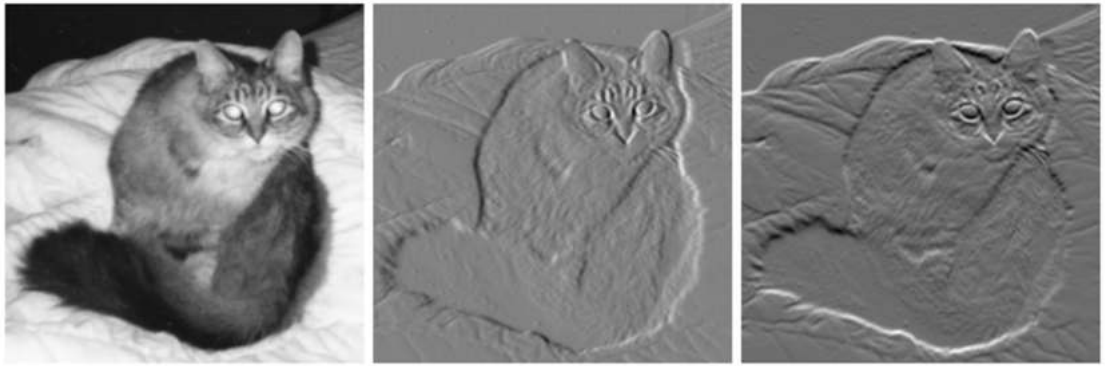
- Motivated by need for dynamic web content
- How do we do **image resizing**?
- We want to consider **image content** not just **image size**

Contents

How to Remove Pixels?

$$e_1(\mathbf{I}) = |\frac{\partial}{\partial x}\mathbf{I}| + |\frac{\partial}{\partial y}\mathbf{I}|$$

- Want to remove “unnoticeable pixels” or pixels with **low energy**
- Define energy in terms of image gradient



Sub-Optimal Ideas

- Find window with highest energy
- Remove lowest energy pixels
- Remove low energy pixels per row
- Remove columns with low energy

(a) Original

(b) Crop (c) Column (d) Seam (e) Pixel (f) Optimal

Main Idea

- Vertical/horizontal seams are more flexible columns/rows

$$\mathbf{s}^{\mathbf{x}} = \{s_i^x\}_{i=1}^n = \{(x(i), i)\}_{i=1}^n, \text{ s.t. } \forall i, |x(i) - x(i-1)| \leq 1$$

- “8-connected path of pixels in the image from top to bottom”
- One pixel in each row of the image
- **Note:** removing a seam is equivalent (size-wise) to removing a column/row

Main Idea (cont.)

- Goal: $$s^* = \min_{\mathbf{s}} E(\mathbf{s}) = \min_{\mathbf{s}} \sum_{i=1}^{n} e(\mathbf{I}(s_i))$$

- Solve using DP (Dynamic Programming)

$$M(i,j) = e(i,j) + \min(M(i-1,j-1), M(i-1,j), M(i-1,j+1))$$

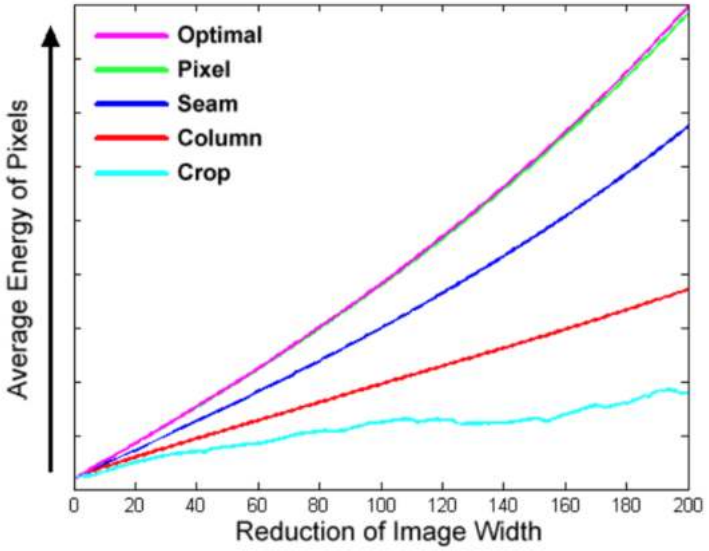
- M(i, j) = minimum (vertical)-seam energy value ending at pixel (i,j)
- DP is quadratic, backtrack to recover seam

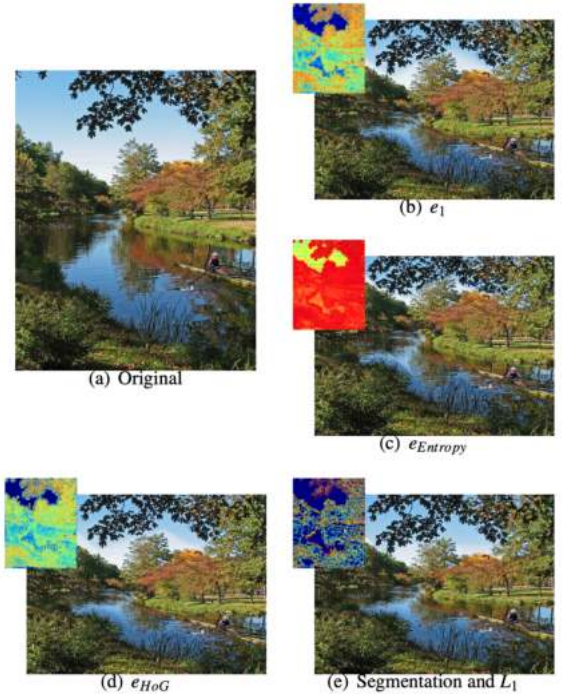
Contents

Visual Demonstration

Credit: https://andrewdcampbell.github.io/seam-carving

Energy Preservation Results





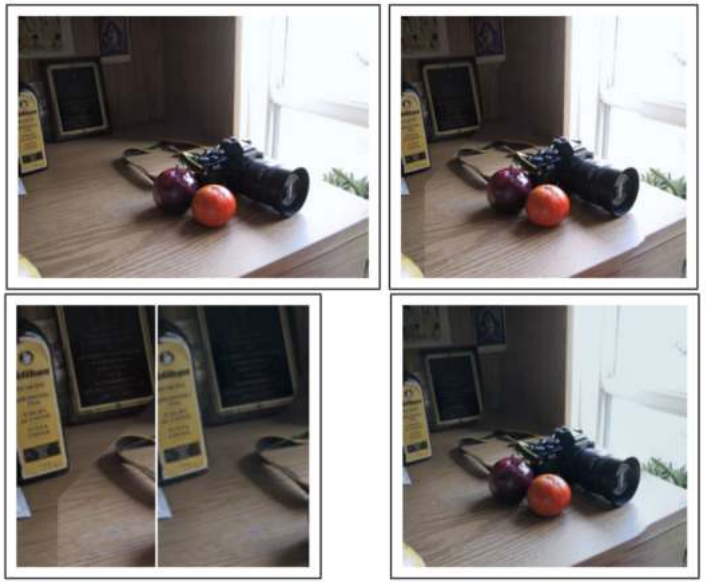
$$e_{HoG}(\mathbf{I}) = \frac{|\frac{\partial}{\partial x}\mathbf{I}| + |\frac{\partial}{\partial y}\mathbf{I}|}{\max(HoG(\mathbf{I}(x,y))},$$

Contents

Other Applications - Content Amplification

Other Applications - Seam Carving Gradients

- Poisson reconstruction for image recovery



Other Applications - Object Removal

Contents

Image Retargeting

- Retarget m x n image to m' x n' where m' < m, n' < n

$$\min_{\mathbf{s^x},\mathbf{s^y},\alpha} \sum_{i=1}^{k} E(\alpha_i \mathbf{s_i^x} + (1-\alpha_i)\mathbf{s_i^y})$$

- DP again — T(r, c) = min cost to obtain image of size (n - r) x (m - c)

$$\mathbf{T}(r,c) = min(\mathbf{T}(r-1,c) + E(\mathbf{s^x}(\mathbf{I_{n-r-1\times m-c}})), \mathbf{T}(r,c-1) + E(\mathbf{s^y}(\mathbf{I_{n-r\times m-c-1}})))$$

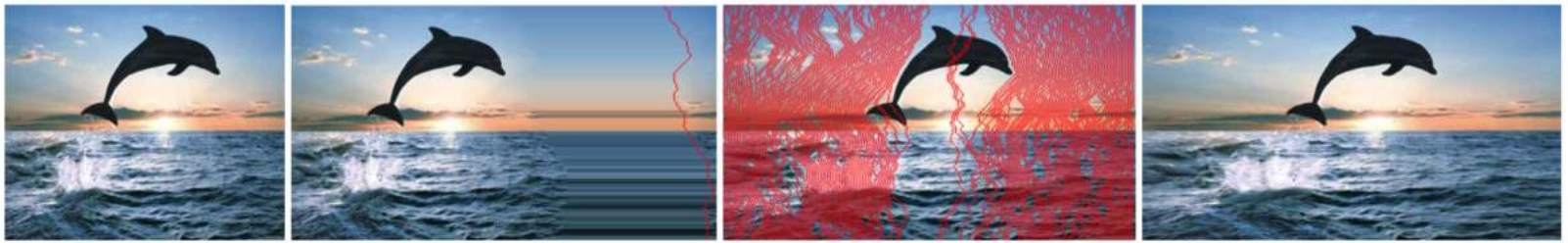
- Save choices in a bitmap to reconstruct

Multi-Size Images

- Resize image to a range of different sizes
- For every pixel, store the index of the seam which removed it
- Example: Resize n x m to n x m’ where m’ < m
  - Take pixels with indices >= m - m’
- Index maps for both directions is more difficult
  - Inconsistent seams

Image Enlarging

- Insert ‘artificial seams’ by considering seams to remove and averaging with neighbors
- Choosing multiple seams avoids stretching artifacts
- Splitting the process up into batches is better!



Conclusion

- Efficient method for resizing with good results
- Limitations with dense content
- Lots of applications!

Figure 16: Examples when resizing using seams fails: images that are too condensed (left) or where the content layout prevents seams to bypass important parts (right). In such cases the best strategy would be to use scaling.

Sources

- http://graphics.cs.cmu.edu/courses/15-463/2007_fall/hw/proj2/imret.pdf
- https://andrewdcampbell.github.io/seam-carving