Image Resizing / Retargeting

“Seam Carving for Content-Aware Image Resizing”

Slides by Chuck Dyer and thanks to K. Padalkar, S. Avidan, and A. Shamir for the raw materials
The Image Resizing/Retargeting Problem
Image Retargeting Objectives

1. Change size
2. Preserve the important *content* and *structures*
3. Limit *artifacts* created
Traditional Methods

- Scaling introduces distortions
- Cropping removes important parts
Many Existing Resizing Methods

<table>
<thead>
<tr>
<th>Source</th>
<th>CR</th>
<th>MULTIOP</th>
<th>SCL</th>
<th>SM</th>
<th>SNS</th>
<th>WARP</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="1ArtRoom_0.75width.png" alt="Image" /></td>
<td><img src="1CR.png" alt="Image" /></td>
<td><img src="1MULTIOP.png" alt="Image" /></td>
<td><img src="1SCL.png" alt="Image" /></td>
<td><img src="1SM.png" alt="Image" /></td>
<td><img src="1SNS.png" alt="Image" /></td>
<td><img src="1WARP.png" alt="Image" /></td>
</tr>
</tbody>
</table>

| ![Image](2CR.png) | ![Image](2SV.png) | ![Image](2MULTIOP.png) | ![Image](2SC.png) | ![Image](2SCL.png) | ![Image](2SM.png) | ![Image](2SNS.png) | ![Image](2WARP.png) |

| ![Image](3CR.png) | ![Image](3SV.png) | ![Image](3MULTIOP.png) | ![Image](3SC.png) | ![Image](3SCL.png) | ![Image](3SM.png) | ![Image](3SNS.png) | ![Image](3WARP.png) |

[1] ArtRoom (0.75 width)

[2] Brasserie L'Aficion (0.50 width)
Seam Carving Method


• In Photoshop called “content aware scaling”

• Main idea: **Remove the least noticeable pixels**
  – How? Define an “energy function” that measures how perceptually noticeable each pixel is

• **Remove the pixels with “low energy”** and avoid removing pixels with “high energy”
  – How? Define a criterion for picking which pixels to remove
Possible Energy Functions

- Edgeness
  - Gradient magnitude
  \[ e_1(I) = \left| \frac{\partial}{\partial x} I \right| + \left| \frac{\partial}{\partial y} I \right| \]
- Entropy
- HOG (Histogram of Gradient)
- Saliency
- ...

Caviet: No single energy function performs well on all images.
Pixel Removal Criteria

- **Optimal**: remove the \( k \) pixels with lowest energy
- Output image no longer rectangular 😞
Pixel Removal Criteria

- **Pixel**: Remove $k$ pixels with lowest energy in each row
- No visual coherence between adjacent rows 😞
Pixel Removal Criteria

- **Column**: Remove whole column with lowest energy
- Frequently introduces artifacts 😞
Seam Definition

• Vertical Seam

is an 8-connected path of pixels in an $n \times m$ image from top to bottom, containing one, and only one, pixel in each row of the image:

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

- Energy of a Seam

\[ E(s) = E(I_s) = \sum_{i=1}^{n} e(I(s_i)) \]

- Minimum Energy Seam

\[ s^* = \min_{s} E(s) = \min_{s} \sum_{i=1}^{n} e(I(s_i)) \]
Pixel Removal Criteria

- **Seam**: Remove the vertical curve of lowest energy 🎉
Pixel Removal Effectiveness

![Diagram showing the effectiveness of different pixel removal methods. The y-axis represents the average energy of pixels, and the x-axis represents the reduction of image width. The methods compared are Optimal, Pixel, Seam, Column, and Crop. The graph shows that Optimal and Pixel methods have the highest average energy, followed by Seam, Column, and Crop.](image-url)
How to Efficiently Compute Best Seam

- Use **Dynamic Programming** to find lowest energy seam in linear time

1. **Forward Pass** (top row to bottom row for finding vertical seam)
   - Define $M(i, j) = \text{total energy of path ending at } (i, j)$
   - $M(1, j) = e(1, j)$
   - $M(i, j) = e(i, j) + \min(M(i-1, j-1), M(i-1, j), M(i-1, j+1))$
   - $B(i, j) = \arg\min_{k=j-1, j, j+1} M(i-1, k)$
   - Find minimum value in last row: $\min_j M(n, j)$

2. **Backward Pass** (bottom row to top row)
   - Trace back path from pixel in bottom row with min value to top row using $B$
Forward Pass

Algorithm Direction

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>4</th>
<th>3</th>
<th>5</th>
<th>2</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>e</td>
<td>3+1</td>
<td>2+1</td>
<td>5+3</td>
<td>2+2</td>
<td>3+2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>3</td>
<td>8</td>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>M</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$e = \text{red numbers}$

$M = \text{black numbers}$

$B = \text{green arrows}$

Credit: Wikipedia
Backward Pass

Algorithm Direction

Credit: Wikipedia
Seams

Seams over energy image            Seams over input image
Shrink Image in 1 Dimension

• Change the image from size $n \times m$ to $n \times m'$
  – assume $m' < m$

• Remove $m-m' = c$ seams successively

Seam Carving
Shrink Image in 1 Dimension

• Change the image from size $n \times m$ to $n \times m'$
  – assume $m' < m$

• Remove $m - m' = c$ seams successively
Shrink Image in Both Dimensions: Optimal Seam Ordering

• Change the image from size $n \times m$ to $n' \times m'$
  – assume $m' < m$ and $n' < n$

• What is the best order for seam carving?
  – Remove vertical seams first?
  – Horizontal seams first?
  – Alternate between the two?
Optimal Seam Ordering

• Solve optimization problem:

\[
\min_{s^x, s^y, \alpha} \sum_{i=1}^{k} E(\alpha_i s^x_i + (1 - \alpha_i) s^y_i)
\]

where \( k = r + c \), \( r = (m - m') \), \( c = (n - n') \) and \( \alpha_i \) is a parameter that determines if at step \( i \) we remove a horizontal or vertical seam: \( \alpha \in \{0,1\} \)
Enlarging Images

- Method 1: Compute the optimal vertical (horizontal) seam $s$ in image and duplicate the pixels in $s$ by averaging them with their left and right neighbors (top and bottom in the horizontal case).
- Often will choose the *same* seam at each iteration, producing noticeable stretching artifact.
Method 2: To enlarge width by $k$, compute top $k$ vertical seams (for removal) and duplicate each of them.
Content Amplification

- Scale the image; this will scale everything, “content” as well as “non-content”
- Shrink the scaled-image using seam carving, which will (hopefully) carve out the non-content part
Object Removal

- User marks the target object to be removed
- Force seams to pass through marked pixels
- Seams are removed from the image until all marked pixels are gone
- To obtain the original image size, use seam insertion
Object Removal

- One shoe removed (and image enlarged to original size)
Object Removal

- Object marking to prevent unwanted results: mark regions where seams must *not* pass
Multi-Size Images

• Methods mentioned so far are not real-time
• We calculate best seam, remove it, calculate the next seam based on new image, etc.
• For real-time resizing
  – Precompute removal order for every pixel in image
  – Compute Index map, $V$, of size $n \times m$ that encodes, for each pixel, the index of the seam that removed it, i.e., $V(i, j) = t$ means pixel $(i, j)$ was removed by the $t^{th}$ seam removal iteration
Multi-Size Images

- Horizontal Index map (H)
- Vertical Index map (V)

- To get an image of width $m'$
  - need to remove $m-m'$ pixels from each row
  - concatenate, in each row, all pixels with seam index greater than or equal to $m-m'$

- Same for changing the height
Seam Index Maps

Blue seams are removed first, red seams removed last.

Input  \( H \)  \( V \)
Failures

- Too much content
- No space for seam to avoid content
Evaluation

- Benchmark and evaluation methodology for image retargeting

RetargetMe

http://people.csail.mit.edu/mrub/retargetme/

- 80 (classified) images
- 8 retargeting operators, 6 objective measures
- 600+ participants and counting
- 400+ human hours, 28,000+ user votes
- 1 graduate student ...
Video Resizing

- Resizing each frame independently is bad
- Instead, find 2D surface in 3D x-y-t video volume