Advanced Image Editing Tools

- Goal: Mimic a better photographer by improving image quality
  - Image retargeting (resize image)
  - Image completion (erase and fill unwanted areas)
  - Texture synthesis
  - Blur distracting background (alter depth of field)
  - Image super-resolution
  - Image "reshuffling" (object rearrangement)
  - And many more

Image Resizing / Retargeting

"Seam Carving for Content-Aware Image Resizing"
by S. Avidan and A. Shamir, *Proc. SIGGRAPH*, 2007

Thanks to K. Padalkar, S. Avidan, and A. Shamir for many of these slides

The Image Resizing/Retargeting Problem



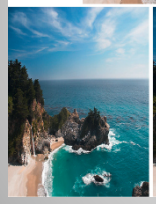
Image Retargeting Objectives

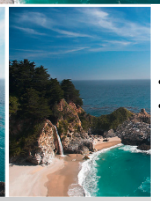
1. Change size 
2. Preserve the important ***content*** and ***structures***
3. Limit ***artifacts*** created



Traditional Methods







Scaling Cropping

- Scaling introduces distortions
- Cropping removes important parts

Many Existing Resizing Methods



Seam Carving Method

- "Seam Carving for Content-Aware Image Resizing," S. Avidan and A. Shamir, *Proc. SIGGRAPH*, 2007
- 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

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

- Entropy
- HOG (Histogram of Gradient)
- Saliency
- …

Caviet: No single energy function performs well on ***all*** images

Saliency Map

original image



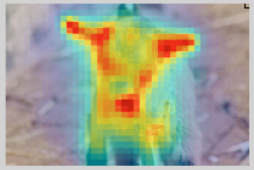
GBVS map



most salient (75%ile) parts



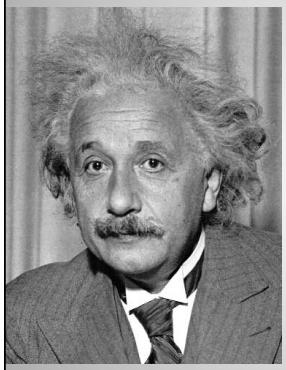
saliency map overlayed



Edges as an Energy Function

Convolution

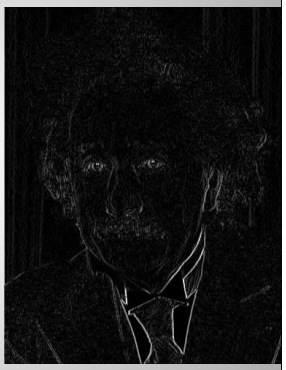
$$\left|\frac{\partial I}{\partial y}\right| = |g_v * I|$$



where g_v =

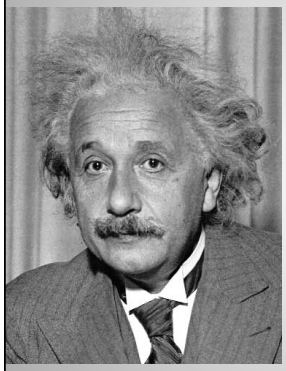
1	0	-1
2	0	-2
1	0	-1

Sobel filter



Vertical edges (absolute value)

Edges as an Energy Function

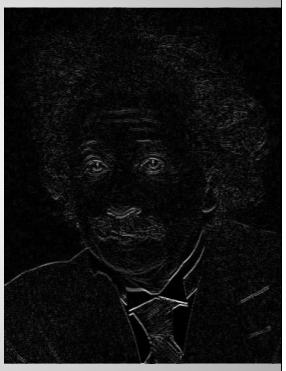


$$\left|\frac{\partial I}{\partial x}\right| = |g_h * I|$$

where g_h =

1	2	1
0	0	0
-1	-2	-1

Sobel filter



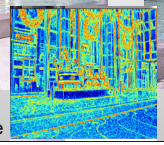
Horizontal edges (absolute value)

Pixel Removal Criterion

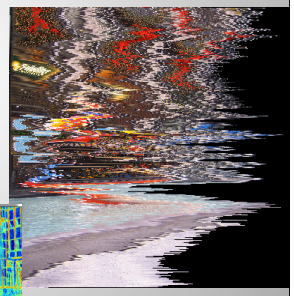
- **Optimal**: remove the *k* pixels with lowest energy
- Output image no longer rectangular



Input image



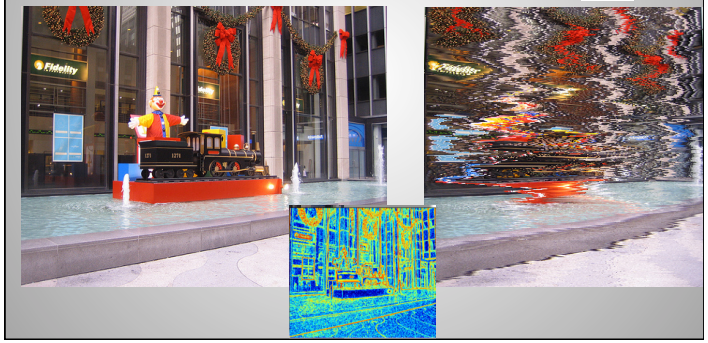
Energy image



Output image

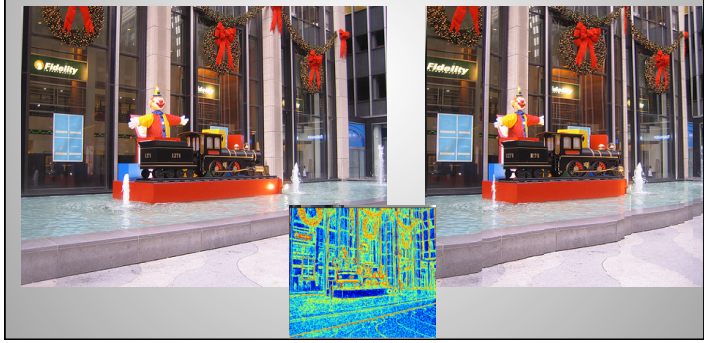
Pixel Removal Criterion

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



Pixel Removal Criterion

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



Seam Definition

- Vertical Seam

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

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



i = row number
x(*i*) = column number

Seam Energy

- Energy of a Seam

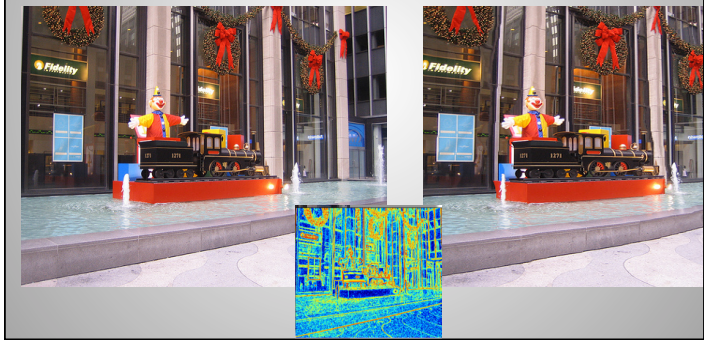
$$E(\mathbf{s}) = E(\mathbf{I}_\mathbf{s}) = \sum_{i=1}^n e(\mathbf{I}(s_i))$$

- Minimum Energy Seam

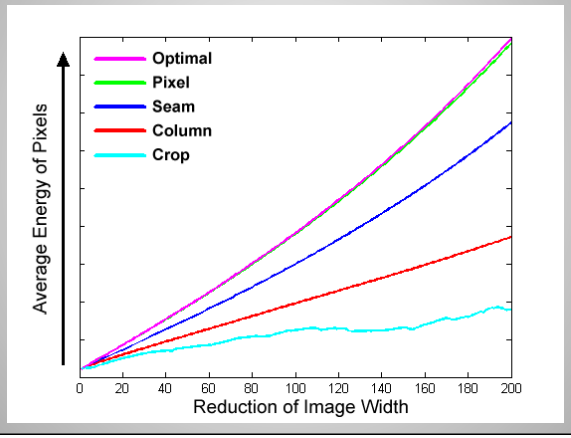
$$s^* = \min_{\mathbf{s}} E(\mathbf{s}) = \min_{\mathbf{s}} \sum_{i=1}^n e(\mathbf{I}(s_i))$$

Pixel Removal Criterion

- **Seam**: Remove the vertical curve of lowest energy



Pixel Removal Effectiveness



How to Efficiently Compute Best Seam?

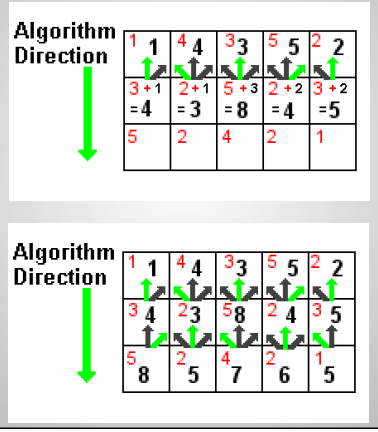
- Use **Dynamic Programming** algorithm to examine the previously solved subproblems and combine their optimal solutions to give the optimal solution for the given problem
- Requires problem to satisfy the **Principle of Optimality**: An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision

How to Efficiently Compute Best Seam?

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

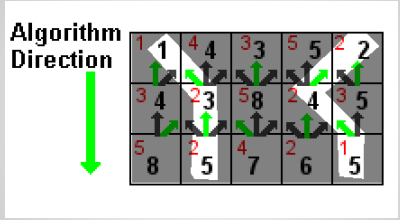
1. **Forward Pass** (top row to bottom row for finding vertical seam)
   - Define $M(r, c)$ = total energy of path ending at (r, c)
   - $M(1,c) = e(1, cj)$
   - $M(r, c) = e(r,c) + \min(M(r\text{-}1, c\text{-}1), M(r\text{-}1, c), M(r\text{-}1, c\text{+}1))$
   - $B(r, c) = \text{argmin}_{k=c\text{-}1,\, c,\, c\text{+}1} M(r\text{-}1, k)$
   - 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



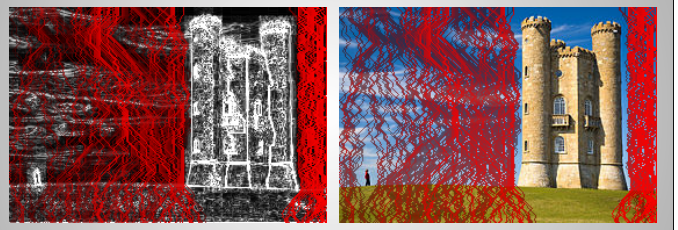
Credit: Wikipedia

Backward Pass



Credit: Wikipedia

Seams



Seams over energy image

Seams over input image

Shrink Image in 1 Dimension

- Change the image from size n x m to n x m'
  - assume $m' < m$
- Remove m-m' = c seams successively



Seam Carving

Shrink Image in 1 Dimension

- Change the image from size *n* x *m* to *n* x *m´*
  - assume *m´* < *m*
- Remove *m*-*m´* = *c* seams successively





Scaling

Shrink Image in Both Dimensions: Optimal Seam Ordering

- Change the image from size *n* x *m* to *n´* x *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:

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

where $k = r+c$, $r = (m-m')$, $c = (n-n')$ and α_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





Enlarging Images

Method 2: To enlarge width by *k*, compute top *k* vertical seams (for removal) and duplicate *each* of them once




Content Amplification

1. Scale the image; this will scale everything, "content" as well as "non-content"
2. Then shrink the scaled-image using seam carving, which will (hopefully) carve out the non-content part




Object Removal

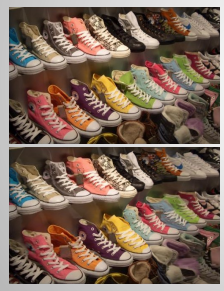
- User marks the target object to be removed
- Force seams to pass through marked pixels
- Seams are removed from the image until all marked pixels are gone
- To obtain the original image size, use seam insertion




Object Removal

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

input




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
  - Pre-compute removal order for every pixel in image
  - Compute Index map, V, of size n x m that encodes, for each pixel, the index of the vertical seam that removes it, i.e., $V(i, j) = t$ means pixel (i, j) is removed by the t^{th} vertical seam removal iteration

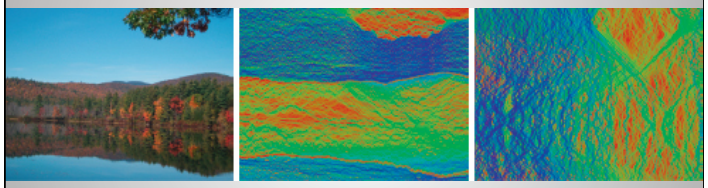
Multi-Size Images

- 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



Input H V

Blue seams are removed first, red seams removed last

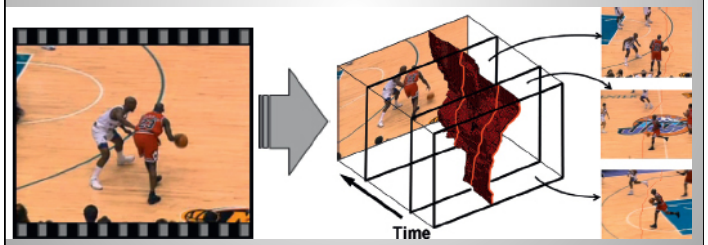
Failures

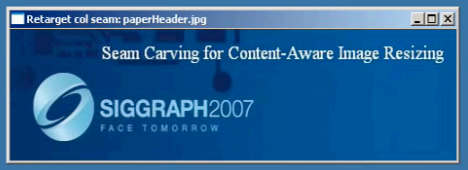


- Too much content
- No space for seam to avoid content

Video Resizing

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





Shai Avidan
Mitsubishi Electric Research Lab
Ariel Shamir
The interdisciplinary Center & MERL