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


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

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**
- ...

Caveat: No single energy function performs well on all images
Saliency Map


Edges as an Energy Function

\[ \frac{\partial f}{\partial x} = |g_h * I| \]

where 

\[ g_h = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \]

Sobel filter

Horizontal edges (absolute value)

Pixel Removal Criterion

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

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

**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:

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

  - $i = \text{row number}$
  - $x(i) = \text{column number}$

**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 Criterion

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

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.

Pixel Removal Effectiveness

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 $(r, c)$
   - $M(1, c) = e(1, c)$
   - $M(r, c) = e(r, c) + \min(M(r-1, c-1), M(r-1, c), M(r-1, c+1))$
   - $B(r, c) = \arg\min_{k = c-1, c, c+1} M(r-1, k)$
   - 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

\[
\begin{array}{cccccc}
1 & 1 & 1 & 1 & 1 & 1 \\
3 & +1 & =4 & 2 & +1 & =3 \\
5 & =2 & =8 & 2 & =4 & =5 \\
5 & 2 & 4 & 5 & 1 & 1 \\
\end{array}
\]

\(e = \text{red numbers}\)
\(M = \text{black numbers}\)
\(B = \text{green arrows}\)

Backward Pass

Algorithm Direction

\[
\begin{array}{cccccc}
1 & 1 & 1 & 1 & 1 & 1 \\
5 & 8 & 2 & 4 & 5 & 1 \\
3 & 4 & 1 & 3 & 8 & 5 \\
5 & 3 & 2 & 5 & 7 & 6 \\
\end{array}
\]

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
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:
  $$\min_{s, s', \alpha} \sum_{i=1}^{k} E(\alpha s_i^x + (1-\alpha) s_i^y)$$
  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
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)
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 $n \times m$ 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

- Blue seams are removed first, red seams removed last

Input  H  V

Input: H V
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