Image Resizing & Seamcarve

CS16: Introduction to Algorithms & Data Structures
Outline

- Image resizing
- Seamcarve
Q: How can you resize an image w/o affecting proportions?
Image Resizing

- Preserve important elements
- Remove/reduce repetitive areas
  - water, sand, …
Image Resizing

Fail  Fail  Fail  Success
Image Resizing

- To shrink image
  - remove unimportant pixels

- Quantify pixel importance
  - How much it varies from neighbors
  - Sum of differences in intensity with neighbors
Image Resizing

- Grayscale 3x3 image with the following pixel intensities
- Importance of the center pixel?
Image Resizing

- Pixel importance
  - Sum of differences in intensity with neighbors
Image Resizing

1 min

Activity #1
Image Resizing

Activity #1

0 min
Image Resizing

- Grayscale 3x3 image with the following pixel intensities
- Importance of the center pixel?

1\+2\+3\+3 = 9
Image Resizing

- Quantify importance of every pixel
- Determine most and least important pixels
Image Resizing: Approach 1

- Remove all pixels with importance below some threshold
- Problem?
  - removing different amount from each row
  - causes jagged right side
Image Resizing: Approach 2

- Remove $n$ least important pixels in each row
- Still not great, too much shifting between adjacent rows
Image Resizing: Approach 3

- Remove column whose total importance is smallest, and repeat
- Much better! But not perfect...
Image Resizing

- Problem
  - removing entire column or entire row distorts image
- What pixels should we remove to resize this image?
Seamcarve

- **Idea:** remove *seams* not columns
  - (vertical) seam is a path from top to bottom
  - that moves left or right by at most one pixel per row
Seamcarve

Near Perfection!
Object Removal via Seamcarve

- Mark object to remove as “unimportant”
  - artificially deflate the importance of its pixels
- Pixels will be removed by algorithm
Seamcarve

- Input
  - 2D array of importance values

- Output
  - Vertical seam with lowest importance
7x3 Importance Array

- Find and circle the best seam

Activity #2
7x3 Importance Array

Activity #2

1 min
7x3 Importance Array
### 7x3 Importance Array

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>3</td>
<td>8</td>
<td>15</td>
<td>1</td>
<td>11</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>13</td>
<td>9</td>
<td>5</td>
<td>10</td>
<td>4</td>
<td>14</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>7</td>
<td>9</td>
<td>14</td>
<td>7</td>
<td>11</td>
</tr>
</tbody>
</table>
7x7 Importance Array

- Find and circle the best seam

Activity #3

1 min
7x3 Importance Array

Activity #3

1 min
7x3 Importance Array

Activity #3

0 min
### 7x3 Importance Array

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>3</td>
<td>1</td>
<td>10</td>
<td>8</td>
<td>11</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>4</td>
<td>11</td>
<td>12</td>
<td>5</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>14</td>
<td>10</td>
<td>7</td>
<td>14</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>12</td>
<td>10</td>
<td>15</td>
<td>13</td>
<td>3</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>8</td>
<td>15</td>
<td>1</td>
<td>11</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>13</td>
<td>9</td>
<td>5</td>
<td>10</td>
<td>4</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>7</td>
<td>9</td>
<td>14</td>
<td>7</td>
<td>11</td>
<td></td>
</tr>
</tbody>
</table>
10x10 Importance Array

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>6</th>
<th>9</th>
<th>12</th>
<th>6</th>
<th>5</th>
<th>12</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>11</td>
<td>14</td>
<td>10</td>
<td>6</td>
<td>15</td>
<td>9</td>
<td>9</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>13</td>
<td>4</td>
<td>1</td>
<td>7</td>
<td>10</td>
<td>4</td>
<td>12</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>15</td>
<td>12</td>
<td>11</td>
<td>4</td>
<td>7</td>
<td>15</td>
<td>8</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>15</td>
<td>11</td>
<td>12</td>
<td>4</td>
<td>14</td>
<td>3</td>
<td>10</td>
<td>1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>13</td>
<td>8</td>
<td>15</td>
<td>6</td>
<td>13</td>
<td>3</td>
<td>13</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>14</td>
<td>6</td>
<td>14</td>
<td>4</td>
<td>13</td>
<td>14</td>
<td>7</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>8</td>
<td>4</td>
<td>11</td>
<td>14</td>
<td>6</td>
<td>12</td>
<td>10</td>
<td>12</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>12</td>
<td>13</td>
<td>2</td>
<td>11</td>
<td>6</td>
<td>6</td>
<td>8</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>15</td>
<td>9</td>
<td>8</td>
<td>12</td>
<td>10</td>
<td>8</td>
<td>6</td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>
## 10x10 Importance Array

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>6</th>
<th>9</th>
<th>12</th>
<th>6</th>
<th>5</th>
<th>12</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>6</td>
<td>9</td>
<td>12</td>
<td>6</td>
<td>5</td>
<td>12</td>
<td>5</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>11</td>
<td>14</td>
<td>10</td>
<td>6</td>
<td>15</td>
<td>9</td>
<td>9</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>13</td>
<td>4</td>
<td>1</td>
<td>7</td>
<td>10</td>
<td>4</td>
<td>12</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>15</td>
<td>12</td>
<td>11</td>
<td>4</td>
<td>7</td>
<td>15</td>
<td>8</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>15</td>
<td>11</td>
<td>12</td>
<td>4</td>
<td>14</td>
<td>3</td>
<td>10</td>
<td>1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>13</td>
<td>8</td>
<td>15</td>
<td>6</td>
<td>13</td>
<td>3</td>
<td>13</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>14</td>
<td>6</td>
<td>14</td>
<td>4</td>
<td>13</td>
<td>14</td>
<td>7</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>8</td>
<td>4</td>
<td>11</td>
<td>14</td>
<td>6</td>
<td>12</td>
<td>10</td>
<td>2</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>12</td>
<td>13</td>
<td>2</td>
<td>11</td>
<td>6</td>
<td>6</td>
<td>8</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>15</td>
<td>9</td>
<td>8</td>
<td>12</td>
<td>10</td>
<td>8</td>
<td>6</td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>
Seams

- Approximately \( cx3^r \) seams in \( cxr \) image
- For \( 10x10 \)
  - 590,490 seams
- For \( 500x500 \)
  - \( 1.81801... \times 10^{241} \) seams (242 digits)
- Age of the Universe
  - \( 4.3 \times 10^{17} \) seconds
The Seamcarve Algorithm

- Function **find_least_important_seam(vals)**
  - **input**: vals is a 2D array of importance values
  - **output**: sequence of column indices that represents a seam

```
[[-S--],
 [S--],
 [S--],
 [-S--],
 [-S--]]
```

\[ [1, 0, 1, 2] \]
# 7x3 Importance Array

<table>
<thead>
<tr>
<th></th>
<th>13</th>
<th>3</th>
<th>1</th>
<th>10</th>
<th>8</th>
<th>11</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>10</td>
<td>4</td>
<td>11</td>
<td>12</td>
<td>5</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>14</td>
<td>10</td>
<td>7</td>
<td>14</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>12</td>
<td>10</td>
<td>15</td>
<td>13</td>
<td>3</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>8</td>
<td>15</td>
<td>1</td>
<td>11</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>13</td>
<td>9</td>
<td>5</td>
<td>10</td>
<td>4</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>7</td>
<td>9</td>
<td>14</td>
<td>7</td>
<td>11</td>
<td></td>
</tr>
</tbody>
</table>

Seam = [6, 5, 4, 5, 4, 5, 5]
Data Structures Needed

- **costs**: 2D array filled in from bottom to top
  - `costs[row][col]` holds total importance of lowest cost seam starting from the bottom row and ending at `costs[row][col]`

- **dirs**: 2D array filled in at the same time as `costs`
  - `dirs[row][col]` holds the direction (-1, 0, or 1) of the previous pixel in the lowest cost seam ending at `costs[row][col]`

```
vals

3   6   8
5   7   2
4   9   3

costs

3   6   8
5   7   2
4   9   3

dirs

0   1
-   -
```

\[ costs[row][col] = \min(costs[row+1][col-1 \text{ to } col+1]) + vals[row][col] \]
Finding Least Important Seam

- Once `costs` is filled in
  - cell in top row with minimum value is the first pixel in least important seam
- Starting from that pixel
  - follow directions in `dirs` to find least important seam
  - and build its column index representation
function find_least_important_seam(vals):
    dirs = 2D array with same dimensions as vals
    costs = 2D array with same dimensions as vals
    costs[height-1] = vals[height-1] // initialize bottom row of costs

    for row from height-2 to 0:
        for col from 0 to width-1:
            costs[row][col] = vals[row][col] +
                min(costs[row+1][col-1],
                    costs[row+1][col],
                    costs[row+1][col+1])
            dirs[row][col] = -1, 0, or 1 // depending on min

    // Find least important start pixel
    min_col = argmin(costs[0]) // Returns index of min in top row

    // Create vertical seam of size ‘height’ by tracing from top
    seam = []
    seam[0] = min_col
    for row from 0 to height-2:
        seam[row+1] = seam[row] + dirs[row][seam[row]]

    return seam
What’s `argmin`?

- What does `min` do?
  - returns minimum output of a function

- What does `argmin` do?
  - given function $f(x)$ returns $x$ that minimizes $f(x)$

- $f(x) = -1 + x^2$
  - $\min f = -1$
  - $\argmin f = 0$ // value for which $f$ is -1

- Array $A = [5, 4, 1, 3, 9]$
  - $\min(A) = 1$
  - $\argmin(A) = 2$ // the index of the minimum value
... 
costs[height-1] = vals[height-1] // initialize bottom row of costs

for row from height-2 to 0:
    for col from 0 to width-1:
        costs[row][col] = vals[row][col] + 
        min(costs[row+1][col-1],
            costs[row+1][col],
            costs[row+1][col+1])
        dirs[row][col] = -1, 0, or 1 // depending on min

// Find least important start pixel
min_col = argmin(costs[0]) // Returns index of min in top row

// Create vertical seam of size ‘height’ by tracing from top
seam = []
seam[0] = min_col
for row from 0 to height-2:
    seam[row+1] = seam[row] + dirs[row][seam[row]]

return seam
costs[height-1] = vals[height-1] // initialize bottom row of costs

for row from height-2 to 0:
    for col from 0 to width-1:
        costs[row][col] = vals[row][col] +
            min(costs[row+1][col-1],
                costs[row+1][col],
                costs[row+1][col+1])
        dirs[row][col] = -1, 0, or 1 // depending on min

// Find least important start pixel
min_col = argmin(costs[0]) // Returns index of min in top row

// Create vertical seam of size ‘height’ by tracing from top
seam = []
seam[0] = min_col
for row from 0 to height-2:
    seam[row+1] = seam[row] + dirs[row][seam[row]]

return seam
... 
costs[height-1] = vals[height-1] // initialize bottom row of costs

for row from height-2 to 0:
    for col from 0 to width-1:
        costs[row][col] = vals[row][col] +
                        min(costs[row+1][col-1],
                            costs[row+1][col],
                            costs[row+1][col+1])
        dirs[row][col] = -1, 0, or 1 // depending on min

// Find least important start pixel
min_col = argmin(costs[0]) // Returns index of min in top row

// Create vertical seam of size ‘height’ by tracing from top
seam = []
seam[0] = min_col
for row from 0 to height-2:
    seam[row+1] = seam[row] + dirs[row][seam[row]]

return seam
... costs[height-1] = vals[height-1] // initialize bottom row of costs

for row from height-2 to 0:
    for col from 0 to width-1:
        costs[row][col] = vals[row][col] +
            min(costs[row+1][col-1],
                costs[row+1][col],
                costs[row+1][col+1])
        dirs[row][col] = -1, 0, or 1 // depending on min

// Find least important start pixel
min_col = argmin(costs[0]) // Returns index of min in top row

// Create vertical seam of size ‘height’ by tracing from top
seam = []
seam[0] = min_col
for row from 0 to height-2:
    seam[row+1] = seam[row] + dirs[row][seam[row]]

return seam
Readings

- The original Seamcarve paper
- Don’t expect to understand it all but has nice examples and is a worthwhile read
Announcements

- Section starts on Monday!
  - Sign up
- HW1 is out tomorrow
- Seamcarve is out
- Python lab next week