# Chapter 9
# Seam Carving
# for Images and Videos

Distributed Algorithms
for Image and Video Processing

## Introduction

**Goals**

- Enhance the visual content of images
  - Adapted images should look natural
  - Most relevant content should be clearly visible
- Change size of images / videos
- Change aspect ratio of images / videos

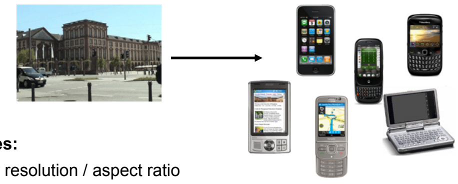
→ technique: seam carving



## Introduction (1)

**Goal**

- Improved visualization of images on handheld mobile devices



**Challenges:**

- Display resolution / aspect ratio
- Bandwidth
- Hardware (CPU, memory, battery)



## Introduction (2)

**Adaptation techniques**

- Letterboxing: details are lost
  → *use full screen*

**letterboxing**



## Introduction (2)

**Adaptation techniques**

- Letterboxing: details are lost
  → *use full screen*
- Cropping: truncated or missing objects
  → *preserve relevant objects*

**cropping**



## Introduction (2)

**Adaptation techniques**

- Letterboxing: details are lost
  → *use full screen*
- Cropping: truncated or missing objects
  → *preserve relevant objects*
- Scaling: distorted objects
  → keep *aspect ratio*

**scaling**

## Introduction (2)

**Adaptation techniques**

- Letterboxing: details are lost
  → *use full screen*
- Cropping: truncated or missing objects
  → *preserve relevant objects*
- Scaling: distorted objects
  → keep *aspect ratio*
- Seam carving

**seam carving**



## Seam Carving (1)

**Idea: Seam Carving***

- Remove a path (seam) of pixels with low relevance for the content of an image
- Each removed seam reduces the image size by 1 column or row
  – **Vertical seam**: change image width
    → path of pixels from top to bottom
  – **Horizontal seam**: change image height
    → path of pixels from left to right

*Avidan, Shamir, 2007



## Seam Carving (2)

**Vertical seam**

$$\mathrm{s} = \{(x(i), i)\}_{i=1}^{H}$$

**S**: vertical seam
*H*: image height

**Constraints**

1. One seam pixel is selected in each row
2. Distance between seam pixels: $|x(i)-x(i-1)| \leq T$

**Definition**

$$\mathrm{s} = \{s_i\}_{i=1}^{H} = \{(x(i), i)\}_{i=1}^{H}, s.t. \forall i : |x(i)-x(i-1)| \leq T$$



## Seam Carving (3)

**How do we identify the optimal seam?**

- Use energy function *E* to select pixels
- Minimize energy of seam
  $$\mathrm{s}^* = \arg\min_{\mathrm{s}} E(\mathrm{s}) = \arg\min_{\mathrm{s}} \sum_{i=1}^{H} E(s_i)$$
- Use dynamic programming to solve this minimization problem

**Energy function**

- Absolute gradient magnitude of adjacent path pixels



## Seam Carving (4)

**Algorithm: Reduce image resolution**

**WHILE** (image size > destination size) **DO**

– Find optimal seam in image
– Remove pixels of the optimal seam and fill gap (shift pixels to the left or top)

low energy seams (white) are removed first



## Video Demo: Seam Carving

## Seam Carving for Videos (1)

**1. Approach:** Use seam carving on each frame separately

→ video becomes blurred and shaky

original

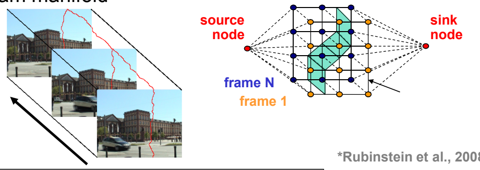
adapted



## Seam Carving for Videos (2)

**2. Approach***

- Video defines 3D space-time volumes
- Remove 2D seam manifolds (seam pixels are connected in 3D)
- Use graph cuts (max-flow min-cut) to detect optimal seam manifold




*Rubinstein et al., 2008



## Video Demo: Seam Carving



## *FSCAV* Algorithm (1)

**Requirements for optimal seams**

1. Robust: avoid shaky videos
2. Fast: calculate 1D seams instead of 2D seam manifolds

**Idea**

- Create one image that aggregates the pixel values / energy values of all frames (process each shot separately)
- Detect 1D seam in aggregated image
- Map this seam to all frames



## *FSCAV* Algorithm (2)

**Approach**

1. Use image registration techniques to estimate camera motion between adjacent frames
2. Compensate camera motion and create background image
3. Detect optimal seam in background image
4. Use inverse camera motion to transform optimal seam back to all frames



## *FSCAV* Algorithm (3)

**Advantages**

- **Seams are robust**: pixels of optimal seam represent the same visual background in all frames
- **Algorithm is fast**: seams are detected in an image (not in a 3D space-time cube)

**Problems**

- Foreground objects
- Seams of the background image are not necessarily included in all frames (e.g., camera pan)

# Camera Motion Compensation

- Use projective camera model

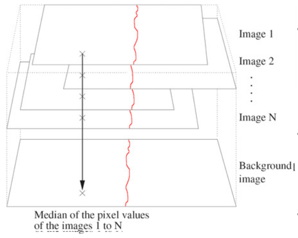
$$x' = \frac{a_{11}x + a_{12}y + t_x}{p_x x + p_y y + 1}, \qquad y' = \frac{a_{21}x + a_{22}y + t_y}{p_x x + p_y y + 1}$$

- Identify feature points (Harris with sub-pixel refinement)
- Similarity measure: sum of absolute differences
- Greedy-based assignment of corresponding features
- Calculate camera model parameters (RANSAC)
  1. Randomly draw 4 corresponding features
  2. Calculate parameters of camera model
  3. Get number of inliers and outliers
  4. Repeat with 1.



# Aggregation of Frames (1)

Calculate background image




→ Optimal seam in background image preserves background objects very well

→ Compare aligned frames to background to get objects

→ Copy object pixels to background image



# Robust Seams (1)

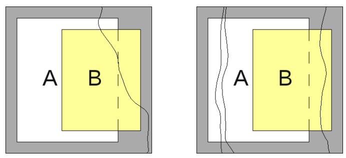
1. Calculate energy map of background image and detect optimal seam
2. Transform seam to individual frames
3. Validate seam
   - *Visible*: seam is included in all frames
   - *Complete*: a pixel is assigned to each row (vertical seam)



# Robust Seams (2)

**Challenges**

- Seams are **not included** in all frames (e.g., in case of camera motion)




→ Solution: ignore these seams



# Robust Seams (3)

**Challenges**

- **Gaps** in seams caused by camera zoom

  → Solution: interpolate missing pixels by adjacent seam pixels

- Different seam pixels of the background image **are mapped to the same pixel** in a frame (rounding errors / inexact camera model parameters)

  → Solution: detect next unoccupied pixel



# Robust Seams (4)

**Challenges**

- **No robust seams** are detected if the first and last frame do not share any visual content (e.g., in case of fast camera motions like long camera pans)

  → Solution:
  - Split sequence in the middle (recursively)
  - Process each video segment separately
  - Use short overlap to fade the video segments

# User Evaluation (1)

- Selected 45 video sequences (shots), 5 categories
- Resolution: PAL (720x568), HD (1920x1080)
- Reduce width by 45 percent (PAL → 400x568)
- 10 test users watched a video from each category in random order: original, scaled, cropped, FSCAV

| | Number of Videos Sequences | Length [frames] |
|---|---|---|
| Static | 5 | 40 – 120 |
| Camera motion only | 12 | 60 – 250 |
| Small object motion | 15 | 50 – 500 |
| High object motion | 11 | 90 – 260 |
| Very large objects | 2 | 100 – 250 |



# User Evaluation (2)

**Cropping**

- Set borders manually
- Worst results in the evaluation

**Scaling / FSCAV**

- Average quality of FSCAV is better
- But: it depends on the motion

**Categories: Static, camera motion only, small object motion**

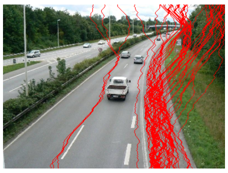
- FSCAV is significantly better



# User Evaluation (3)

**Category: High object / camera motion**

- Camera or objects move parallel to seams
  → FSCAV is better than scaling

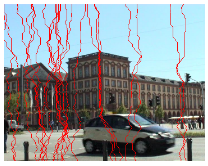





# User Evaluation (4)

**Category: High object / camera motion**

- Camera or objects move orthogonal to seams
  → quality drops significantly



# User Evaluation (5)

**Category: Very large objects**

- Quality is very low
  → video should be scaled



# User Evaluation (6)

**Category: Static**

# User Evaluation (7)

**Category: Small object motion**



# User Evaluation (8)

**Category: High object motion**



# User Evaluation (9)

**Comparison with shaky video**

**shaky**

**seam carving**



# Graph Cuts / FSCAV (1)

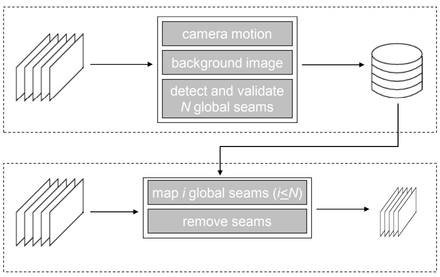
**Comparison of visual quality**

- Visual quality is very similar
- Static / camera motion:
  → visual quality of FSCAV is slightly better
- Small objects move in parallel to the seams
  → visual quality of graph cuts is higher
- Large objects / orthogonal motion
  → quality is low with both techniques



# Graph Cuts / FSCAV (2)

**Computational effort: FSCAV**






# Graph Cuts / FSCAV (3)

Runtime* and memory requirements

| | *Low res.* 120 × 68 50 frames | *PAL* 720 × 576 150 frames | *HD* 1920 × 1080 200 frames |
|---|---|---|---|
| Crop | <1 s | 5 s | 32 s |
| Scale | <1 s | 6 s | 36 s |
| FSCAV<br>- Analysis<br>- Adaptation | <br>14 s<br>1 s<br>29 MB | <br>8 min<br>11 s<br><200 MB | <br>51 min<br>83 s<br><200 MB |
| Graph Cuts | 17 min<br>290 MB | N/A<br>(44 GB) | N/A<br>(292 GB) |
| Graph Cuts (hierarchical) | N/A<br>N/A | 49 min<br>530 MB | 123 min<br>820 MB |

*Athlon 64, Dual Core, 2.4 GHz, 2GB RAM

## Conclusions

- FSCAV is suitable to adapt videos to the limited screen resolution of mobile devices
- Advantages of FSCAV:
  - fast
  - videos are stable (*robust seams*)
  - limited memory requirements
- The *motion in videos* has a great impact on the quality of adapted videos:
  - Slow motion in *parallel* to seams
    → FSCAV works very well
  - Fast motion *orthogonal* to seams
    → Scaling is more reliable



Questions ?