Content-Aware Image Resizing

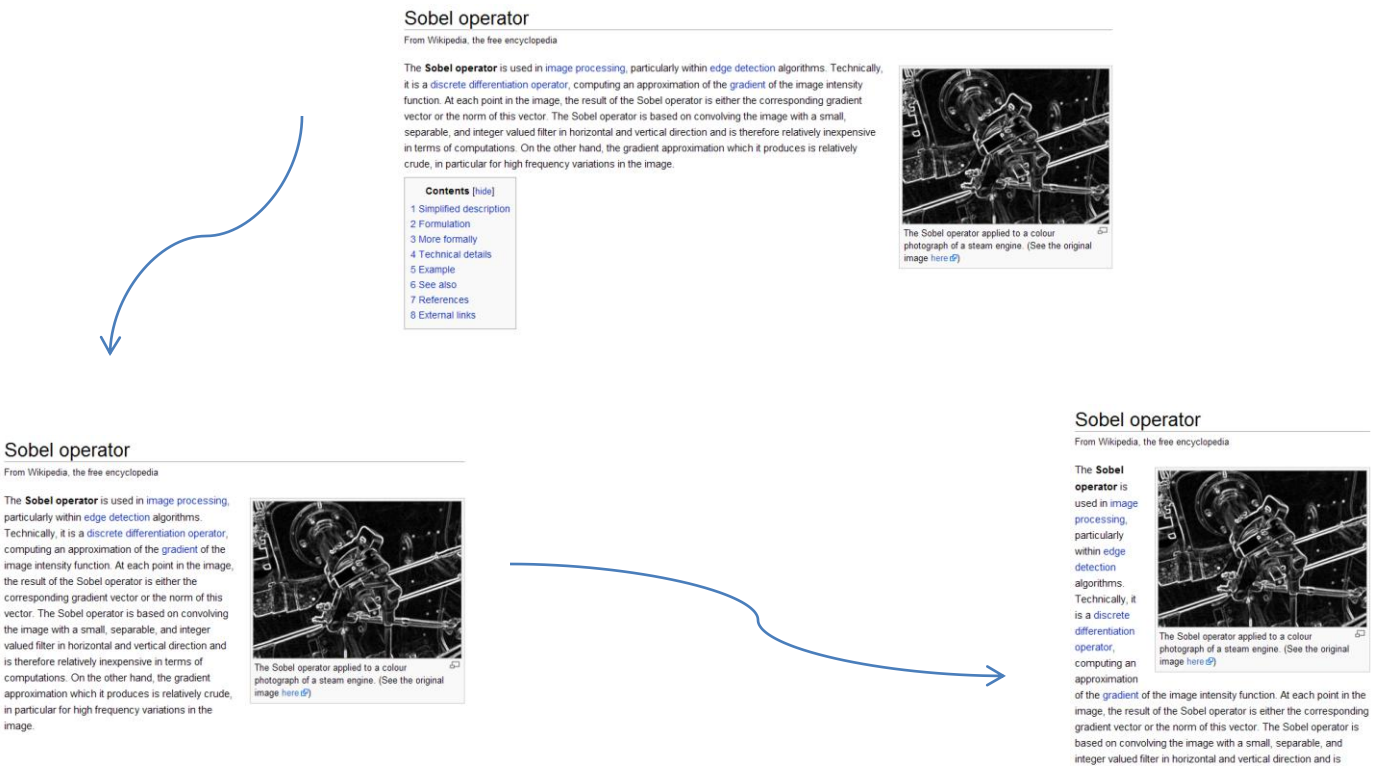
Final Project Report
601 Computer Graphics and Image Processing
Vuk Malbasa
Based on work by S. Avidan and A. Shamir

Overview

- The problem
- The proposed solution
- Problems with the solution
- Technical details
- Conclusion

The Problem

- The internet uses dynamic formatting to arrange pictures and text into available space



Sobel operator

From Wikipedia, the free encyclopedia

The **Sobel operator** is used in image processing, particularly within edge detection algorithms. Technically, it is a discrete differentiation operator, computing an approximation of the gradient of the image intensity function. At each point in the image, the result of the Sobel operator is either the corresponding gradient vector or the norm of this vector. The Sobel operator is based on convolving the image with a small, separable, and integer valued filter in horizontal and vertical direction and is therefore relatively inexpensive in terms of computations. On the other hand, the gradient approximation which it produces is relatively crude, in particular for high frequency variations in the image.

Contents [hide]

1 Simplified description
2 Formulation
3 More formally
4 Technical details
5 Example
6 See also
7 References
8 External links

The Sobel operator applied to a colour photograph of a steam engine. (See the original image here)

Sobel operator

From Wikipedia, the free encyclopedia

The **Sobel operator** is used in image processing, particularly within edge detection algorithms. Technically, it is a discrete differentiation operator, computing an approximation of the gradient of the image intensity function. At each point in the image, the result of the Sobel operator is either the corresponding gradient vector or the norm of this vector. The Sobel operator is based on convolving the image with a small, separable, and integer valued filter in horizontal and vertical direction and is therefore relatively inexpensive in terms of computations. On the other hand, the gradient approximation which it produces is relatively crude, in particular for high frequency variations in the image.

The Sobel operator applied to a colour photograph of a steam engine. (See the original image here)

Sobel operator

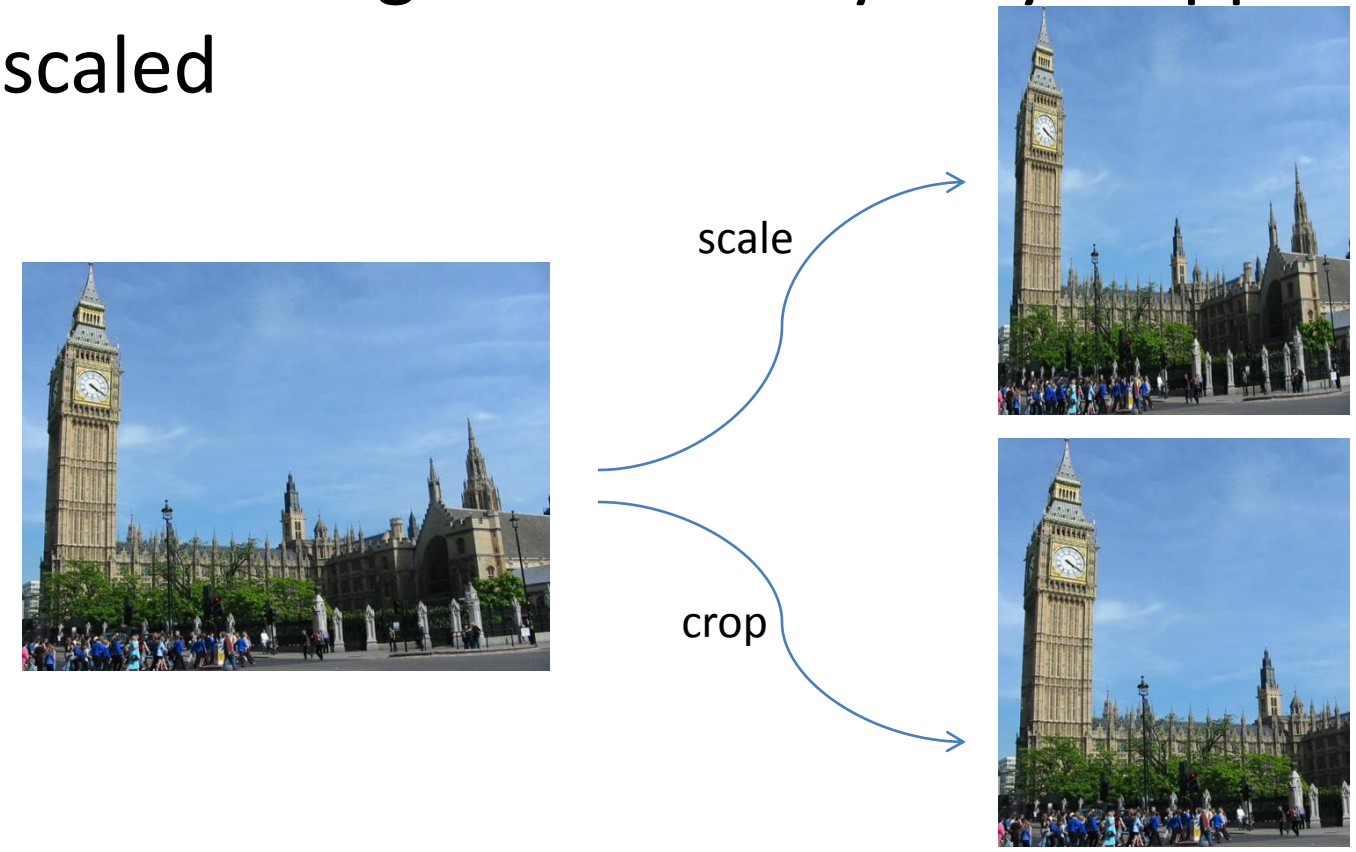
From Wikipedia, the free encyclopedia

The **Sobel operator** is used in image processing, particularly within edge detection algorithms. Technically, it is a discrete differentiation operator, computing an approximation of the gradient of the image intensity function. At each point in the image, the result of the Sobel operator is either the corresponding gradient vector or the norm of this vector. The Sobel operator is based on convolving the image with a small, separable, and integer valued filter in horizontal and vertical direction and is

The Sobel operator applied to a colour photograph of a steam engine. (See the original image here)

The Problem

- While text can be rearanged to better fix the screen images are usually only cropped or scaled



The Problem

Scaling distorts images

Cropping cuts out details

Some solutions

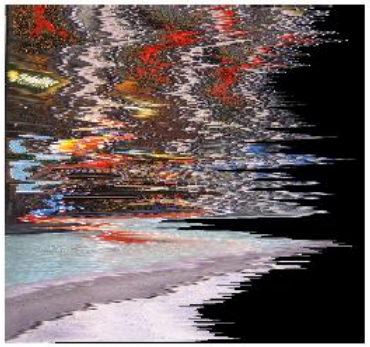
Original Image and per pixel energy

cropped

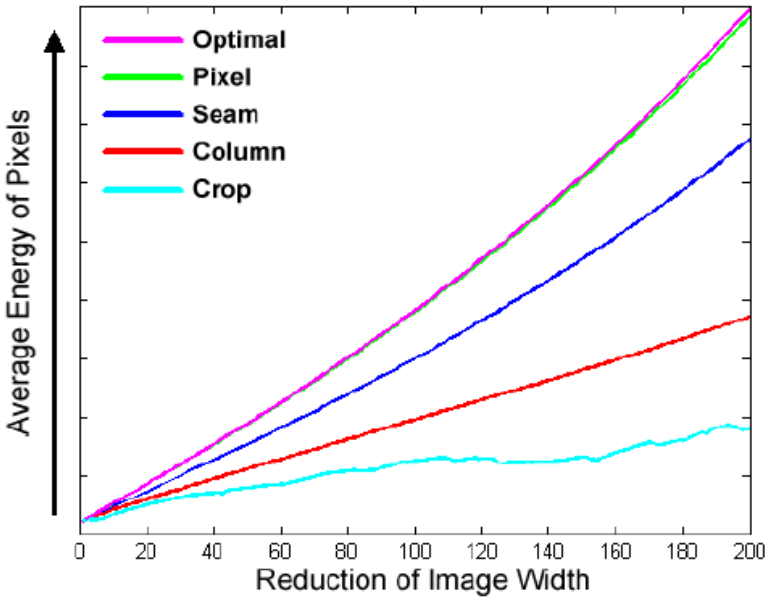
column

pixel

seam



optimal



Solution

- Retargetting: Keeping important details while removing visually uninteresting parts

Re-targeting

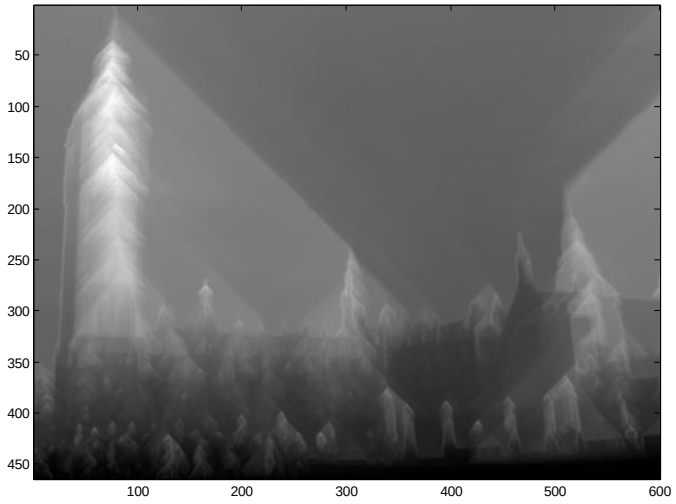
Problems with the solution

Re-targeted

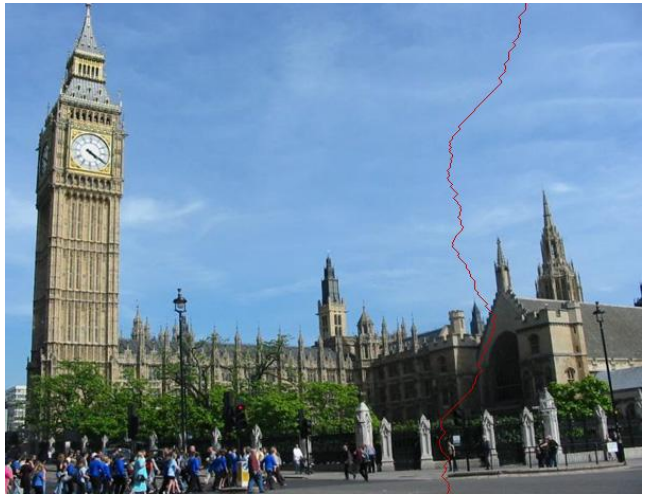
Technical Details

Detect interesting parts

Find minimal path



Remove minimal seam



Detecting interesting parts

Sobel operator

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

$$\mathbf{G} = \sqrt{\mathbf{G}_x^{\,2} + \mathbf{G}_y^{\,2}}$$

Find Minimal Path

- Assume we have the energy matrix E
- We need to find the minimal path from top to bottom.
- To solve this we need to check all possible paths
- Luckily we have a constraint: the path needs to be 4 connected

Find Minimal Path - trivial

```
function out = find(E)
   for i = 1:maxx
      temp = findM(i,1)
   if temp < current_minimum
      current_minimum
   end

function out = findM(x,y,E)
  if y > maxy
     out = 0
  else
     out = E(x,y) + min(findM(x-1,y+1,E), findM(x,y+1,E), find(x+1,y+1,E))
  end
```

E =

0.3	0.2	0.4	0.3
0.2	0.3	0.4	0.1
0.1	0.2	0.4	0.3
0.4	0.1	0.3	0.2

Complexity: $O(maxx \times 3^{maxy})$ ~ too much

Find Minimal Path - memoization

```
function out = find(E)
   for i = 1:maxx
      temp = findM(i,1)
   if temp < current_minimum
      current_minimum
   end
```

```
function out = findM(x,y,E)
   if isnan(table(x,y))
      table(x,y) = rfindM(x,y,E)
      out = table(x,y)
   else
      out = table(x,y)
   end
```

```
function out = rfindM(x,y,E)
  if y > maxy
     out = 0
  else
     out = E(x,y) + min(findM(x-1,y+1,E), findM(x,y+1,E), find(x+1,y+1,E))
  end
```

Complexity:

O(maxx × maxy) - for making the table

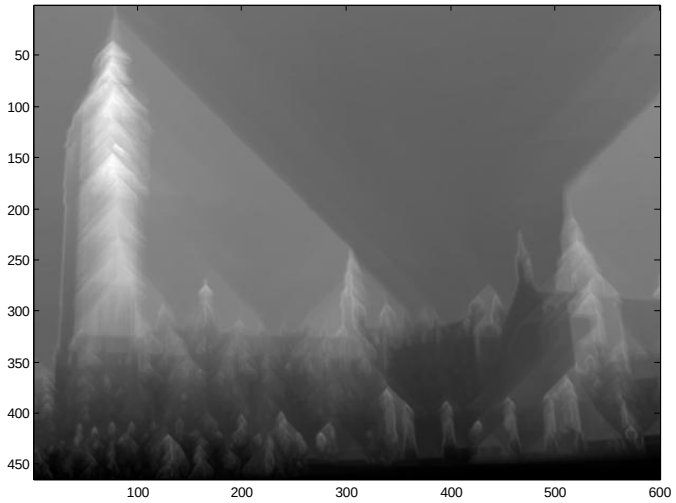
O(maxx) - for the main procedure

STILL A PROBLEM!

Find Minimal Path – dynamic programming

```
function out = findM(E)
   for i = maxy:-1:1
      for j = 1:maxx
         table(x,y) = E(x,y) + min(M(x,y+1),M(x-1,y+1),M(x+1,y+1))
      end
   end
```

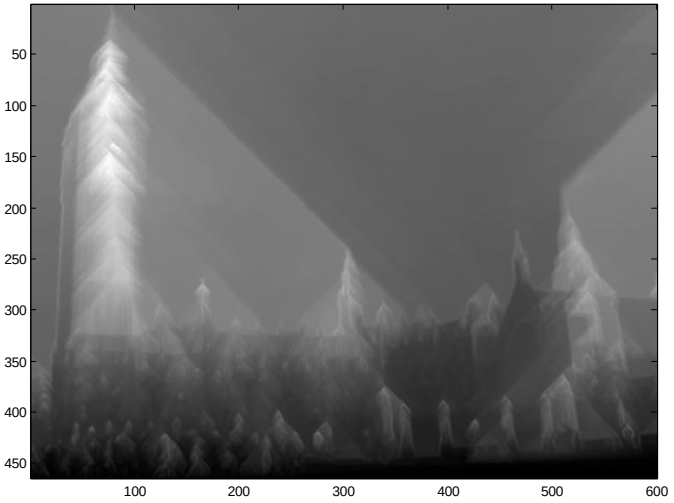
findM



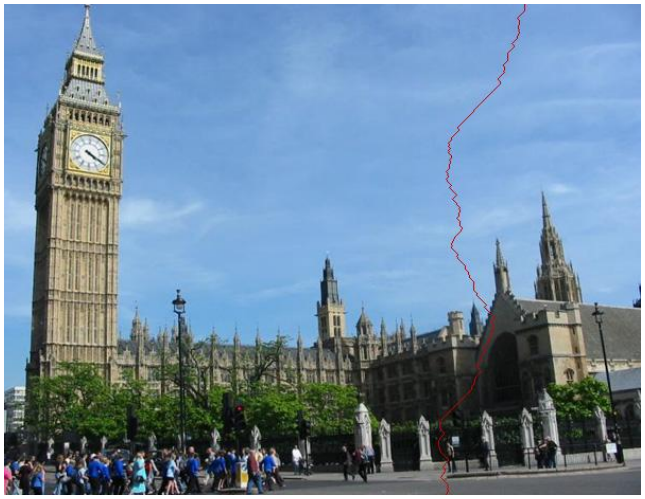
Now we can just do a greedy search

Find Minimal Path - greedy

```
function seam = findS(table)
[v s] = min(table(:,1))
seam = s
for y = 2:maxy
    [vn sn] = min(table(s-1,y),table(s,y),table(s+1,y))
    s = s + sn - 2
    seam(y) = s
end
```



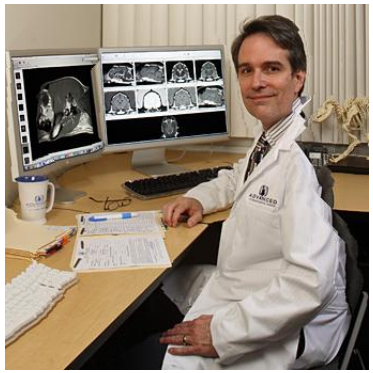
findS



Improvements

- Improving performance
  - It is possible to code the program in C or Java which will improve the running time by several orders of magnitude.
  - Saving the previous calculations so that the program doesn't compute the same things over and over again
  - Precomputing all possible minimal paths and keeping them in order so that no on-the-fly computing is necessary
- Improving quality
  - Different energy functions that preserve the objects better
    - Entropy, visual saliency, eye gaze movement
  - User corrected seam removal
    - Marking desirable or undesirable objects in frame for removal

Conclusion



With people and other objects of recognizable shape the algorithm when removing a lot of the image creates undesirable effects

With buildings and similar objects the algorithm performs fairly well, however artifacts such as the interruption of straight lines is obvious