CS4442/9542b
Artificial Intelligence II
prof. Olga Veksler

Lecture 14

Computer Vision

Edge Detection

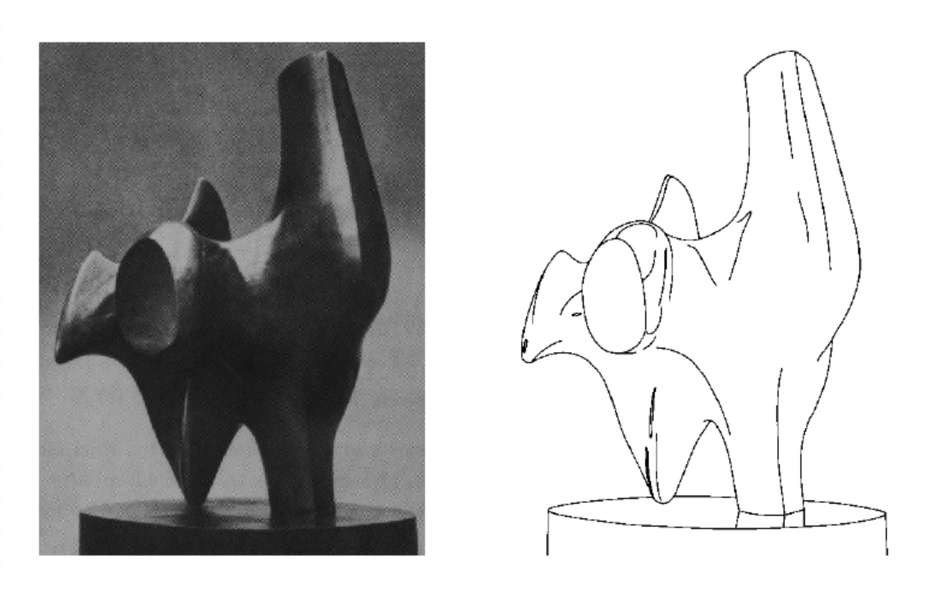
Some slides from: S.Seitz, D. Jacobs, D. Lowe, H. Man, K. Grauman, D. Hoiem, S. Lazebnik

Outline

- Edge Detection
  - Edge types
  - Image Gradient
  - Canny Edge Detector
- Application
  - intelligent image resizing: Seam Carving

Edge Detection

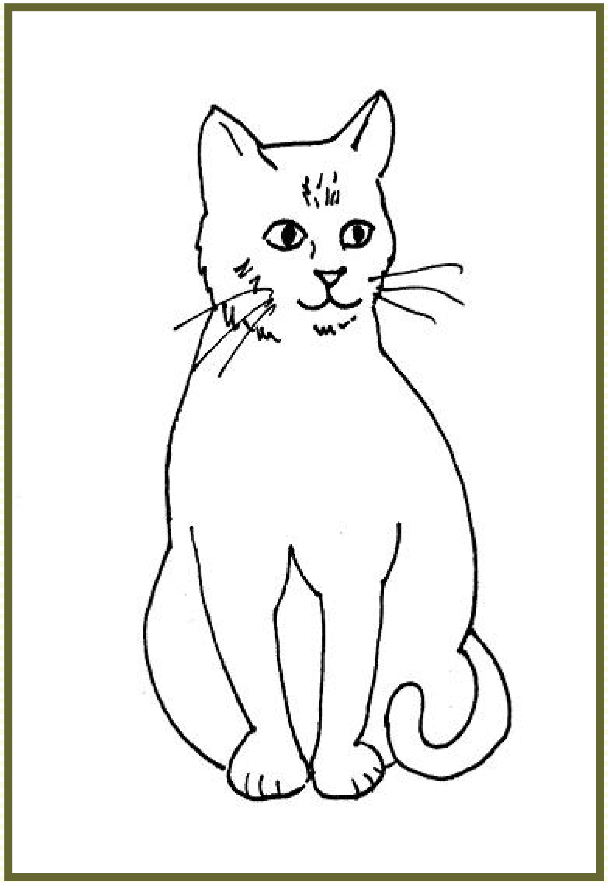
- Convert intensity image into binary (0 or 1) image that marks **prominent** curves
- What is a prominent curve?
  - no exact definition
  - intuitively, it is a place where abrupt changes occur



- Why perform edge detection?
  - most shape and semantic and information is encoded in edges
  - edges are stable to lighting and other changes, makes them good features for object recognition, etc.
  - more compact representation than intensity

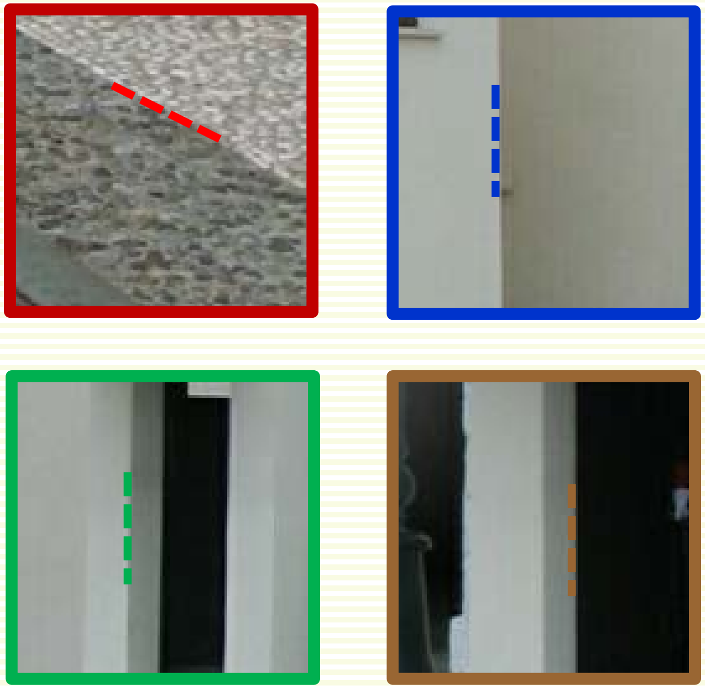
Line Drawings

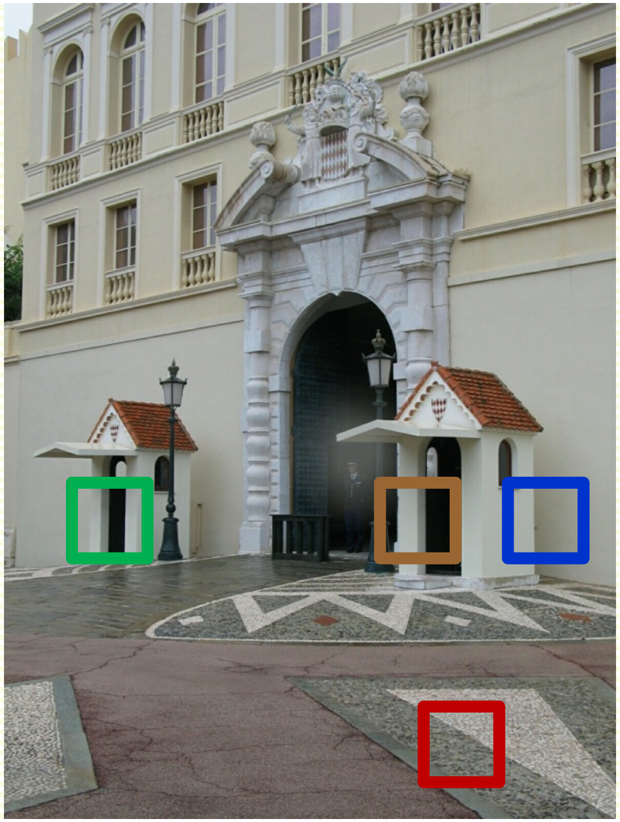
- Artists do it
  - and much better, as they use high level knowledge which edges are more perceptually important



Origin of Edges

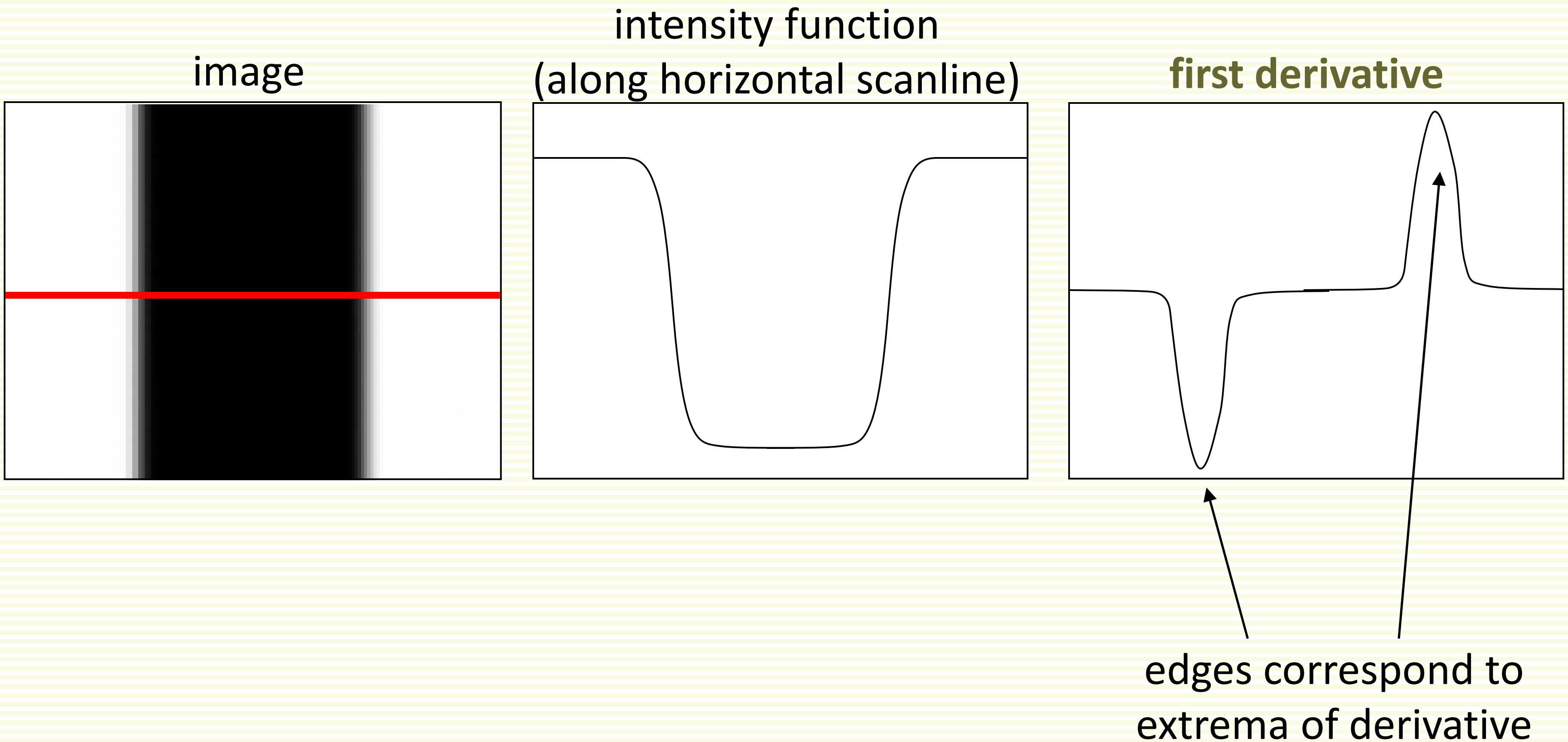
- Many **discontinuity** causes:
  - surface color or texture discontinuity
  - depth discontinuity (object boundary)
  - surface normal discontinuity
  - illumination discontinuity (shadows)





Derivatives and Edges

- An edge is a place of rapid change in intensity



Derivatives with Convolution

- For 2D function $f(x,y)$, partial derivative in horizontal direction

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

- For discrete data, approximate

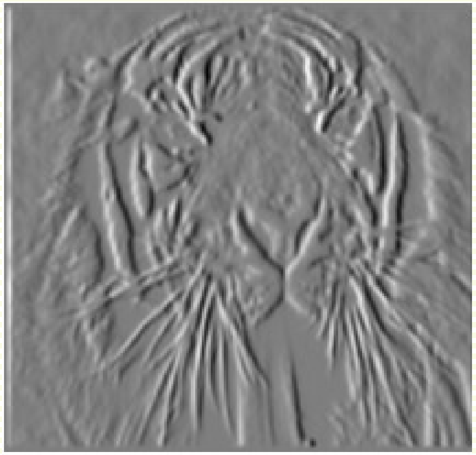
$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1, y) - f(x,y)}{1}$$

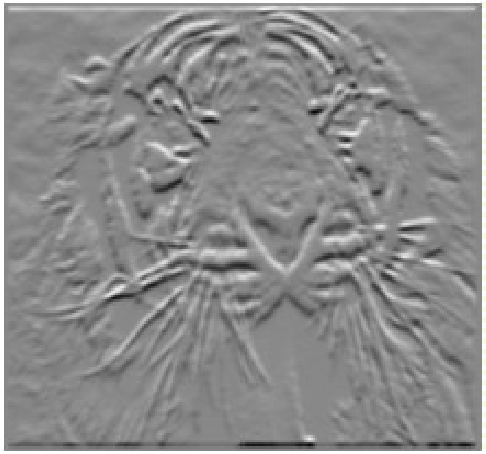
- Similarly, approximate vertical partial derivative (wrt y)

- How to implement as a convolution?

Image Partial Derivatives

Which is with respect to x?





$$\frac{\partial f(x,y)}{\partial x}$$

-1	1

or

1	-1

$$\frac{\partial f(x,y)}{\partial y}$$

-1
1

or

1
-1

Finite Difference Filters

- Other filters for derivative approximation

Prewitt:

$$H_x = \frac{1}{6}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad H_y = \frac{1}{6}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Sobel:

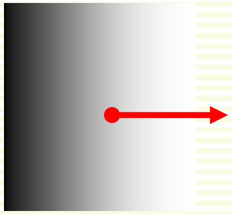
$$H_x = \frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad H_y = \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Image Gradient

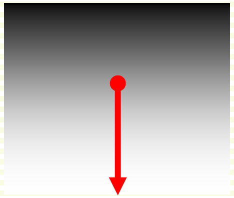
- Combine both partial derivatives into vector $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$

image gradient

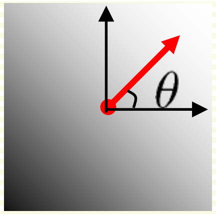
- Gradient points in the direction of most rapid increase in intensity



$$\nabla f = \left[\frac{\partial f}{\partial x}, 0 \right]$$



$$\nabla f = \left[0, \frac{\partial f}{\partial y} \right]$$



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$$

- **Direction** perpendicular to edge:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} \middle/ \frac{\partial f}{\partial x} \right)$$

gradient orientation

- Edge **strength**

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

gradient magnitude

Application: Gradient-domain Image Editing

- Goal: solve for pixel values in the target region to match gradients of the source region while keeping background pixels the same

P. Perez, M. Gangnet, A. Blake, Poisson Image Editing, SIGGRAPH 2003

Simplest Edge Detector

- Compute gradient magnitude at each pixel

$$g(x, y) = \|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

- Threshold gradient magnitude to get binary edge map *e*(*x*,*y*)

$$e(x, y) = \begin{cases} 1 & \text{if } g(x,\mathrm{y}) > \mathrm{T} \\ 0 & \text{otherwise} \end{cases}$$

Effects of Noise

original image

$$\frac{\partial f}{\partial y}$$

$$\frac{\partial f}{\partial x}$$

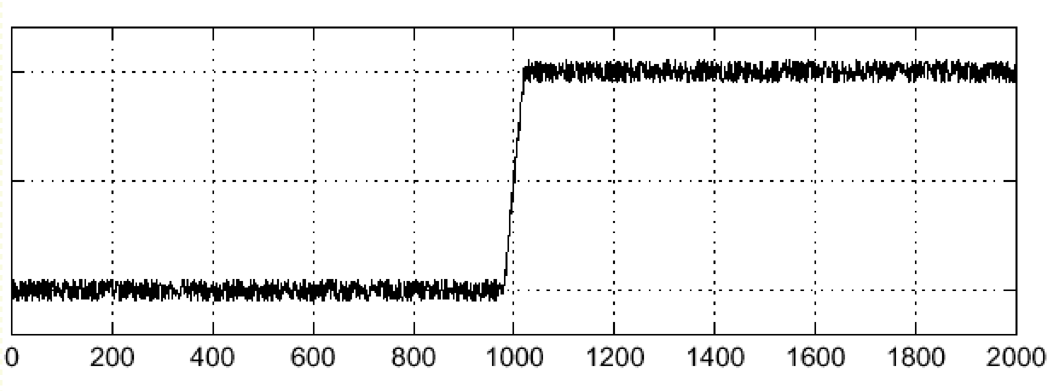
$$\|\nabla f\|$$

- Too many pixels with large gradient magnitude due to image noise

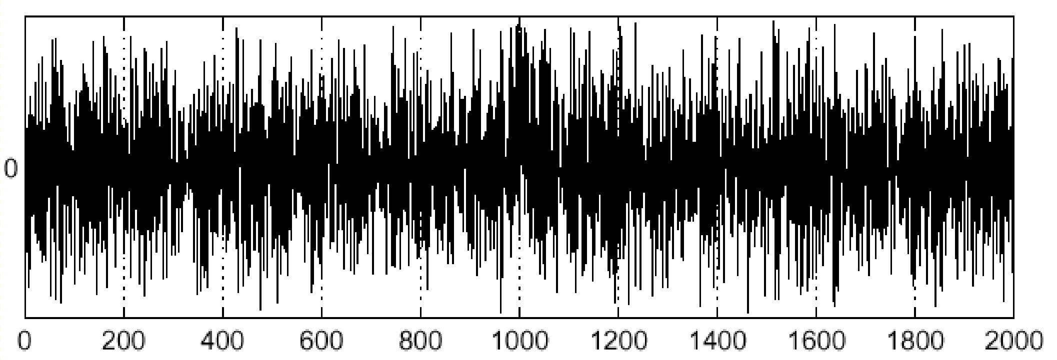
Effects of noise

- Consider a single row of the image
- Plot intensity as a function of *x*

$$f(x)$$



$$\frac{\partial}{\partial x} f(x)$$



- Where is the edge?

Effects of Noise

- How do we deal with noise?
- We already know, filter the noise out using Gaussian kernel
- First convolve image with a Gaussian filter
- Then take derivative

Derivative Theorem of Convolution

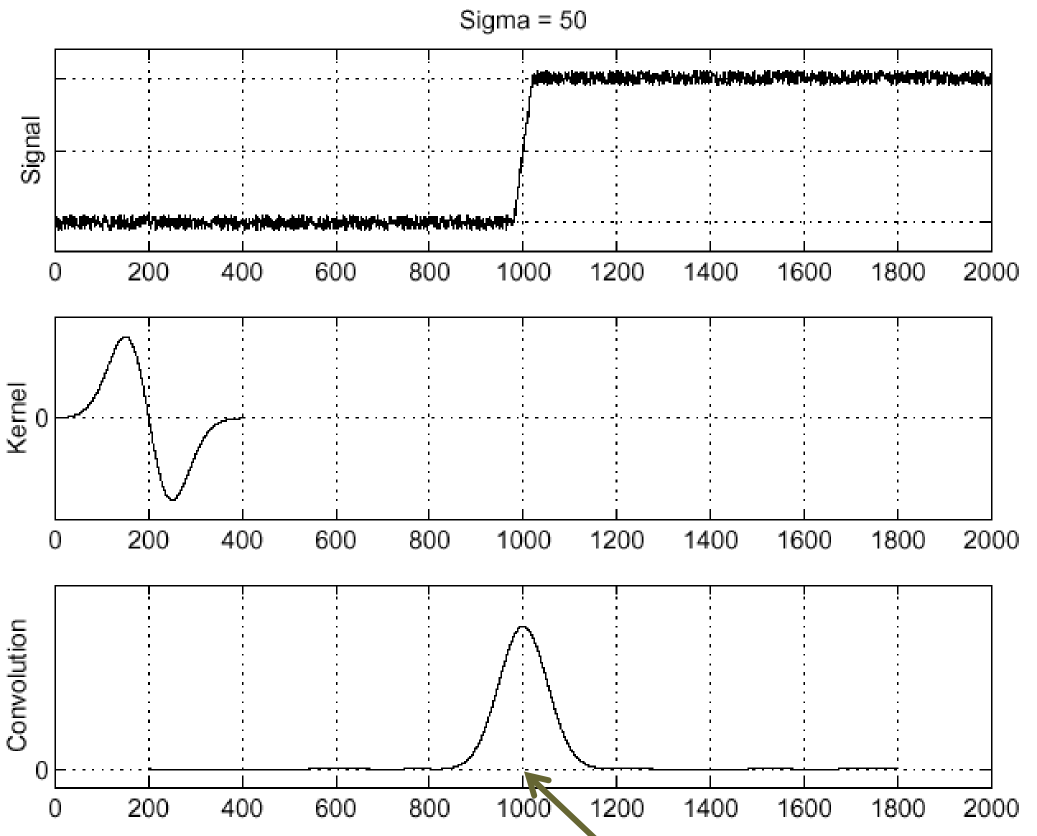
$$\frac{\partial}{\partial x}(H * f) = \left(\frac{\partial}{\partial x} H\right) * f$$

- This saves us one step

$$f$$

$$\frac{\partial}{\partial x} H$$

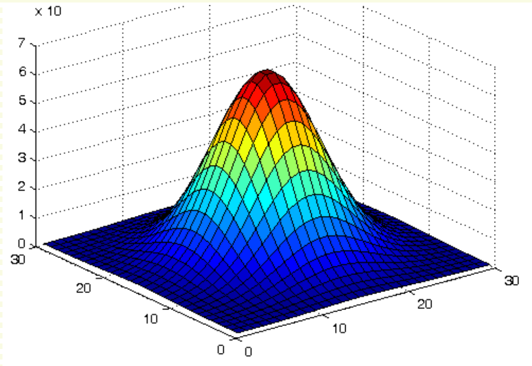
$$\left(\frac{\partial}{\partial x} H\right) * f$$

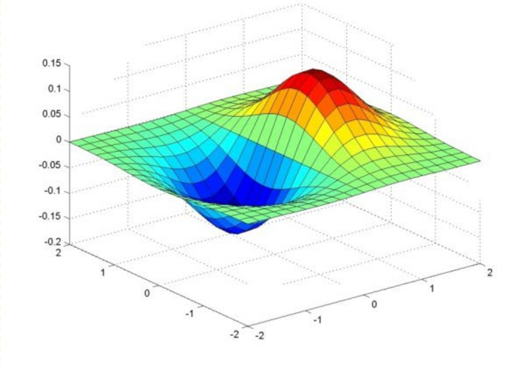


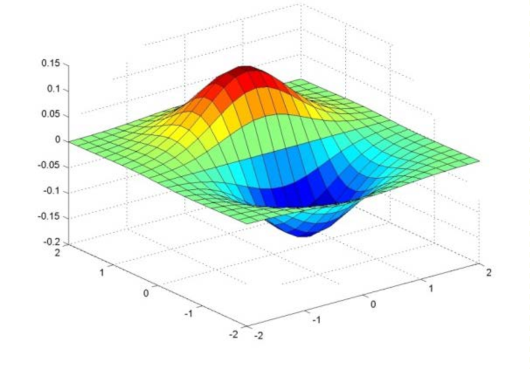
edge

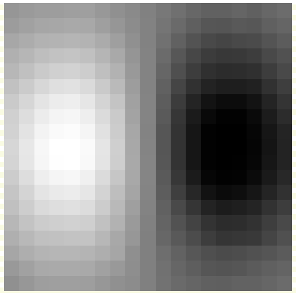
Derivative of Gaussian

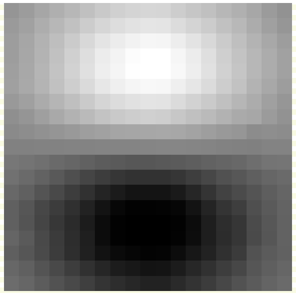
$$G_\sigma \qquad \frac{\partial}{\partial x} G_\sigma \qquad \frac{\partial}{\partial y} G_\sigma$$











white is positive values, dark negative, gray zero

- Which finds horizontal, which vertical edges?

Derivative of Gaussian: Example

- Ignoring constant:

$$G_\sigma(x,y) = e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

- Differentiate:

$$\frac{\partial}{\partial x} G_\sigma(x,y) = -\frac{x}{\sigma^2} \cdot e^{-\frac{(x^2+y^2)}{2\sigma^2}} \qquad \frac{\partial}{\partial y} G_\sigma(x,y) = -\frac{y}{\sigma^2} \cdot e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

- Plug in σ = 5, and take 5×5 window

(-2,-2)	(-1,-2)	(0,-2)	(1,-2)	(2,-2)
(-2,-1)	(-1,-1)	(0,-1)	(1,-1)	(2,-1)
(-2,0)	(-1,0)	(0,0)	(1,0)	(2,0)
(-2,1)	(-1,1)	(0,1)	(1,1)	(2,1)
(-2,2)	(-1,2)	(0,2)	(1,2)	(2,2)

coordinates in window

0.04	0.08	0	-0.08	-0.04
0.16	0.37	0	-0.37	-0.16
0.27	0.61	0	-0.61	-0.27
0.16	0.37	0	-0.37	-0.16
0.04	0.08	0	-0.08	-0.04

H_x

-0.04	-0.16	-0.27	-0.16	-0.04
-0.08	-0.37	-0.61	-0.37	-0.08
0	0	0	0	0
0.08	0.37	0.61	0.37	0.08
0.04	0.16	0.27	0.16	0.04

H_y

Example Continued

H_x

0.04	0.08	0	-0.08	-0.04
0.16	0.37	0	-0.37	-0.16
0.27	0.61	0	-0.61	-0.27
0.16	0.37	0	-0.37	-0.16
0.04	0.08	0	-0.08	-0.04

H_y

-0.04	-0.16	-0.27	-0.16	-0.04
-0.08	-0.37	-0.61	-0.37	-0.08
0	0	0	0	0
0.08	0.37	0.61	0.37	0.08
0.04	0.16	0.27	0.16	0.04

121	121	122	123	122	123
121	121	122	123	122	123
122	123	124	123	124	123
120	122	122	123	122	123
121	121	124	123	124	123
125	120	124	123	124	123

apply H_x to pixel in red: -0.78

apply H_y to pixel in red: 0.46

121	121	122	123	20	20
121	121	122	123	22	22
122	123	124	123	24	21
120	122	122	123	22	22
121	121	124	123	24	23
125	120	124	123	24	24

apply H_x to pixel in red: **217**

apply H_y to pixel in red: 0.69

Example Continued

H_x

0.04	0.08	0	-0.08	-0.04
0.16	0.37	0	-0.37	-0.16
0.27	0.61	0	-0.61	-0.27
0.16	0.37	0	-0.37	-0.16
0.04	0.08	0	-0.08	-0.04

H_y

-0.04	-0.16	-0.27	-0.16	-0.04
-0.08	-0.37	-0.61	-0.37	-0.08
0	0	0	0	0
0.08	0.37	0.61	0.37	0.08
0.04	0.16	0.27	0.16	0.04

121	121	122	123	20	20
121	121	122	123	22	22
122	123	124	123	24	21
120	122	122	123	22	22
121	121	124	123	24	23
125	120	124	123	24	24

apply H_x to pixel in red: **217**

apply H_y to pixel in red: 0.69

121	121	122	120	121	125
121	121	123	122	121	120
122	122	124	122	124	124
123	123	123	123	123	123
20	22	24	22	24	24
20	22	21	22	23	24

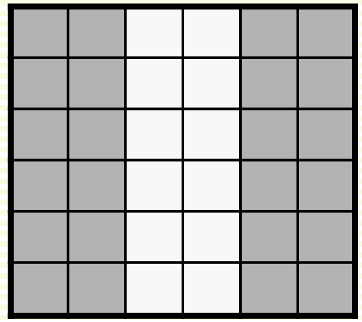
apply H_x to pixel in red: -0.69

apply H_y to pixel in red: **-217**

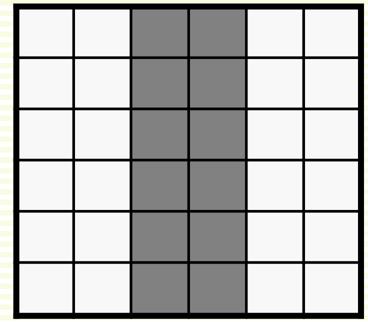
Mask looks like the pattern it is trying to detect!

What does this Mask Detect?

2	2	-4	-4	2	2
2	2	-4	-4	2	2
2	2	-4	-4	2	2
2	2	-4	-4	2	2
2	2	-4	-4	2	2



strong negative response

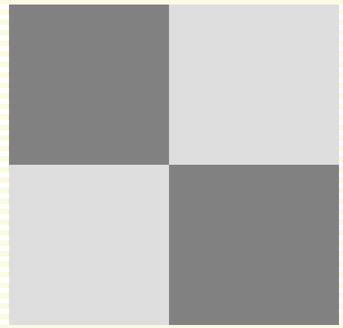


strong positive response

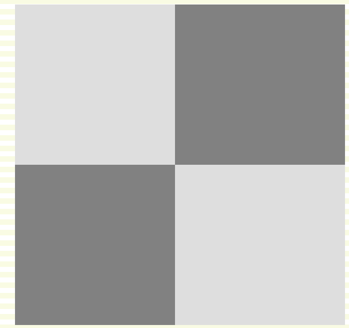
What Does this Mask Detect?

2	2	-2	-2
2	2	-2	-2
-2	-2	2	2
-2	-2	2	2

strong negative response



strong positive response



Canny Edge Detector

input image

Canny Edge Detector

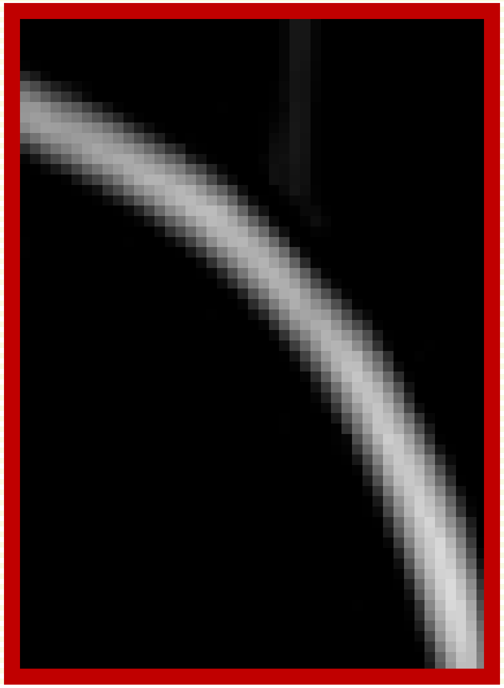
gradient magnitude

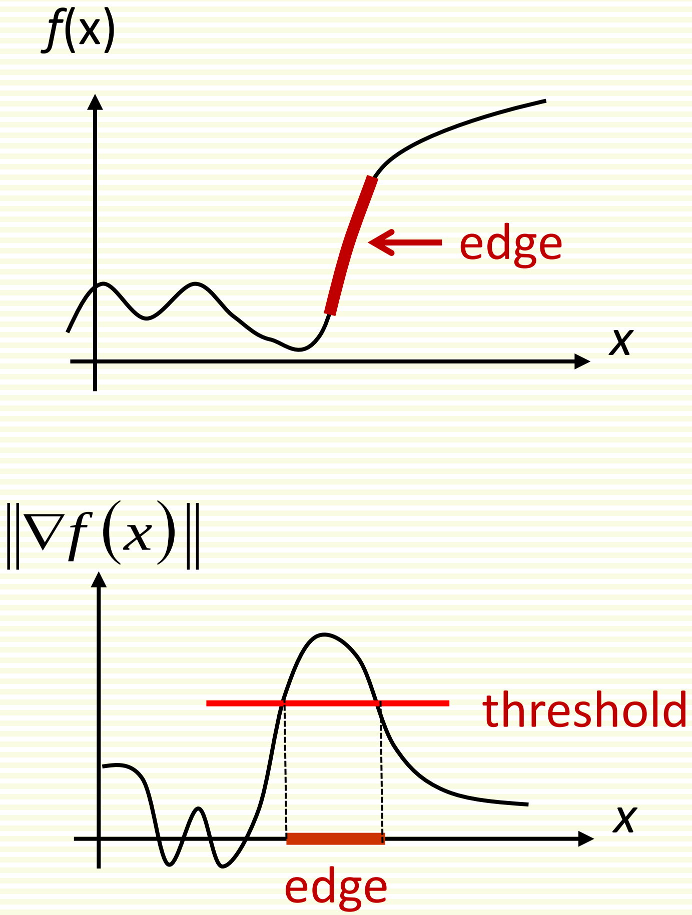
Canny Edge Detector

thresholding

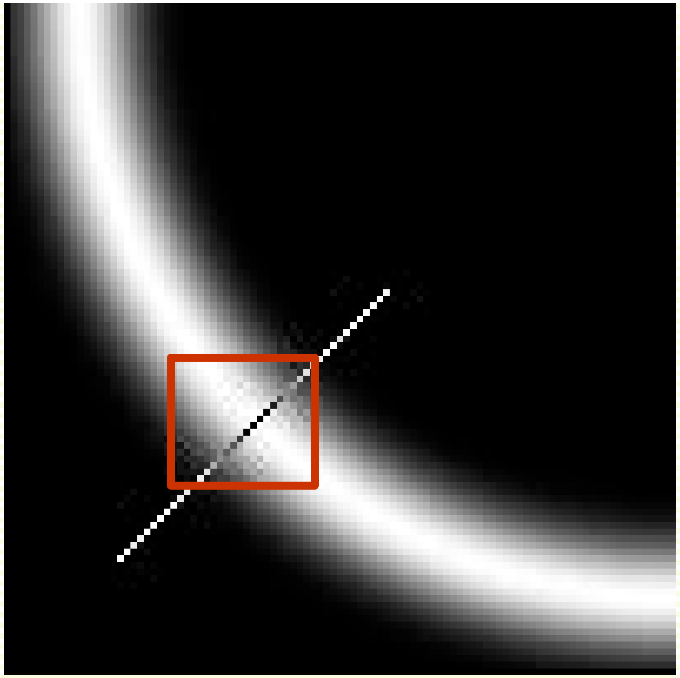
Canny Edge detector

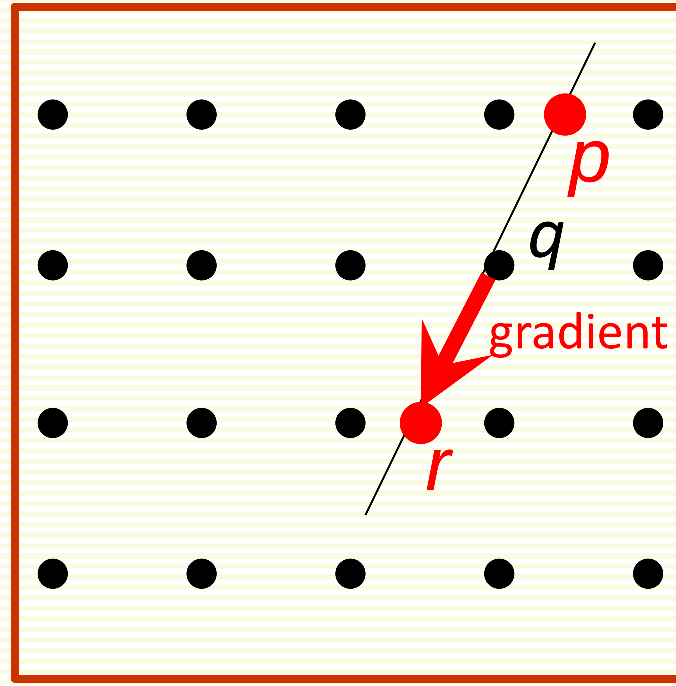
- Why we get thick regions after thresholding?





Edge Thinning: non-maximum suppression





- Check if pixel *q* is local maximum along gradient direction
  - take two neighbors in *p* and *r* in the gradient direction
    - requires checking interpolated pixels *p* and *r*
  - turn off edge at pixel *q* if $g(q) < g(p)$ or $g(q) < g(r)$

The Canny Edge Detector

- Another problem: some weak edge pixels do not survive thresholding

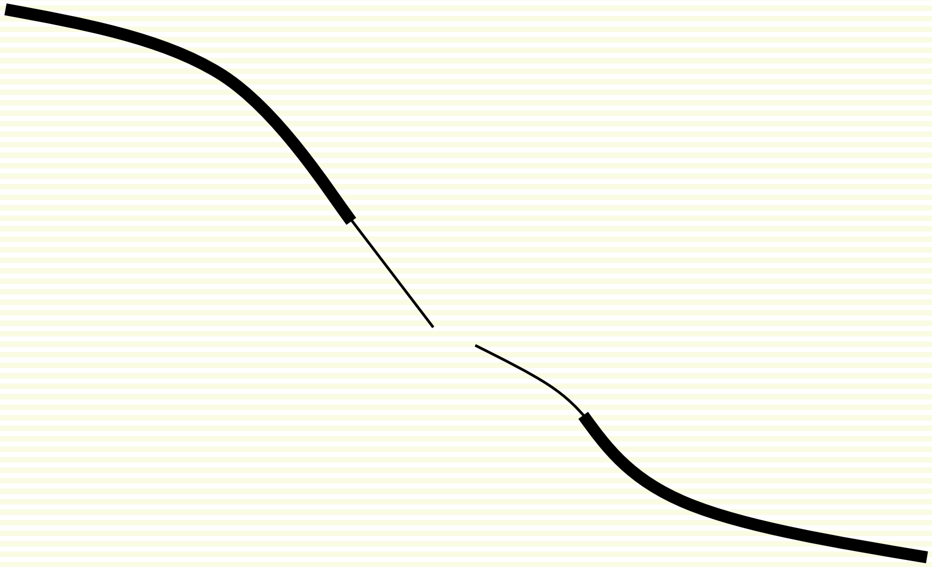
after thinning

The Canny Edge Detector

- Try a smaller threshold?
  - too many weak edges

Hysteresis Thresholding

- Specify a high and low thresholds
- Use high threshold to start edge curves
  - Continue edge in the gradient direction
  - Use low threshold for continuation



The Canny Edge Detector

low threshold

high threshold

hysteresis with low and high thresholds

Effect of Kernel Size and Spread

original image

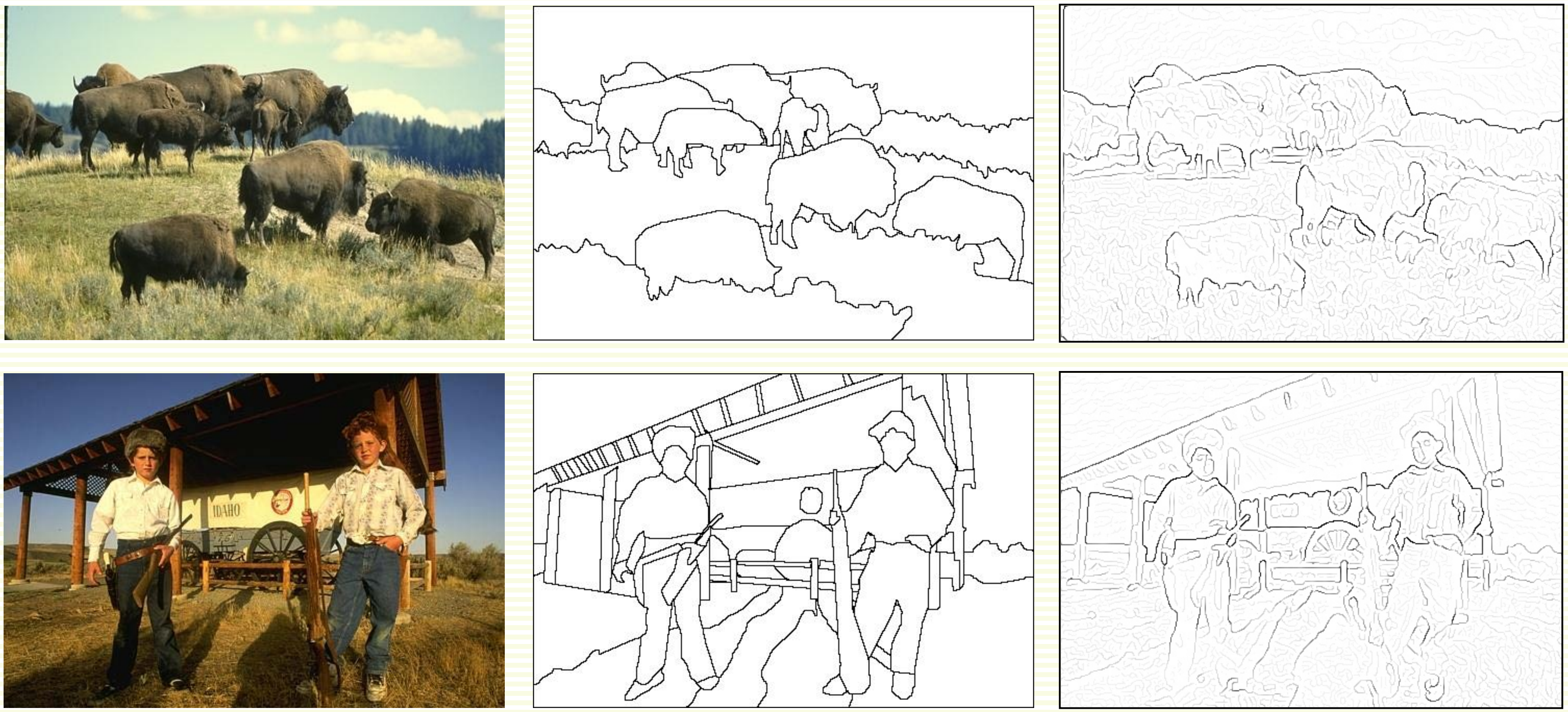
Canny with $\sigma = 1$

Canny with $\sigma = 2$

- Smaller σ/mask size detects fine scale edges
- Larger σ/mask detects large scale edges

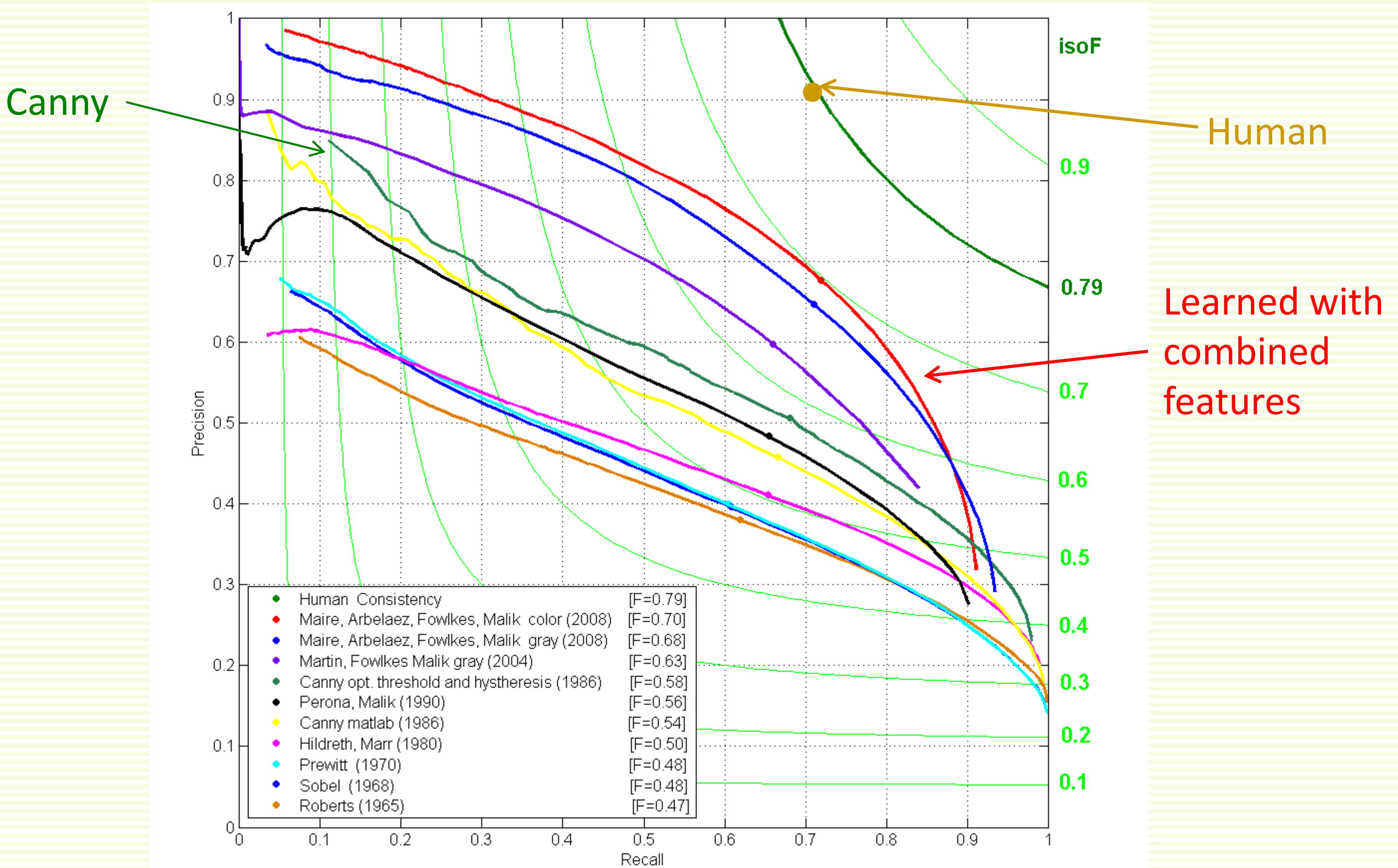
Still Far From Human Vision

image | human segmentation | gradient magnitude



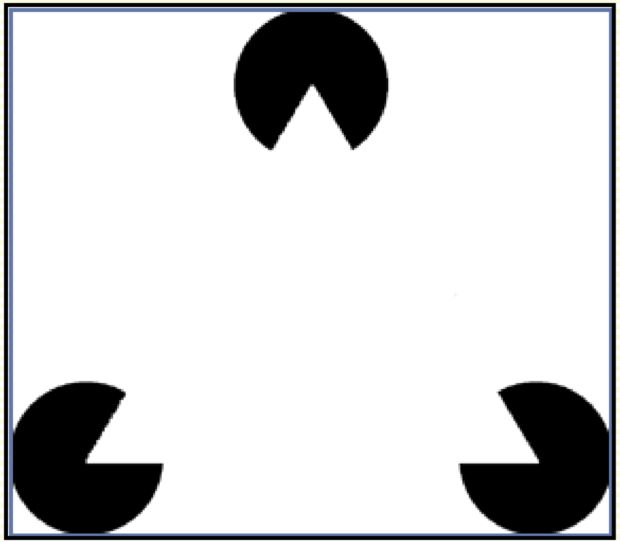
- Berkeley segmentation database:
  http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

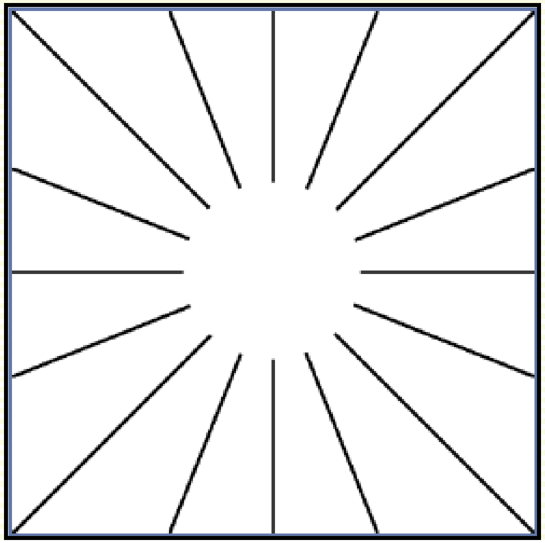
State-of-the-Art in Contour Detection

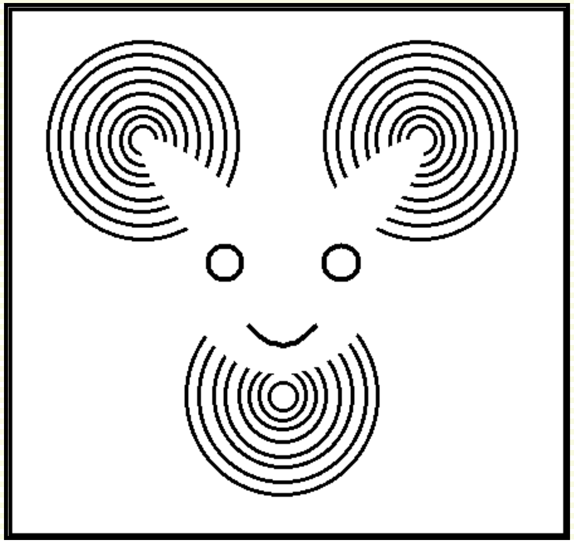


Source: Jitendra Malik: http://www.cs.berkeley.edu/~malik/malik-talks-ptrs.html

Illusory Contours



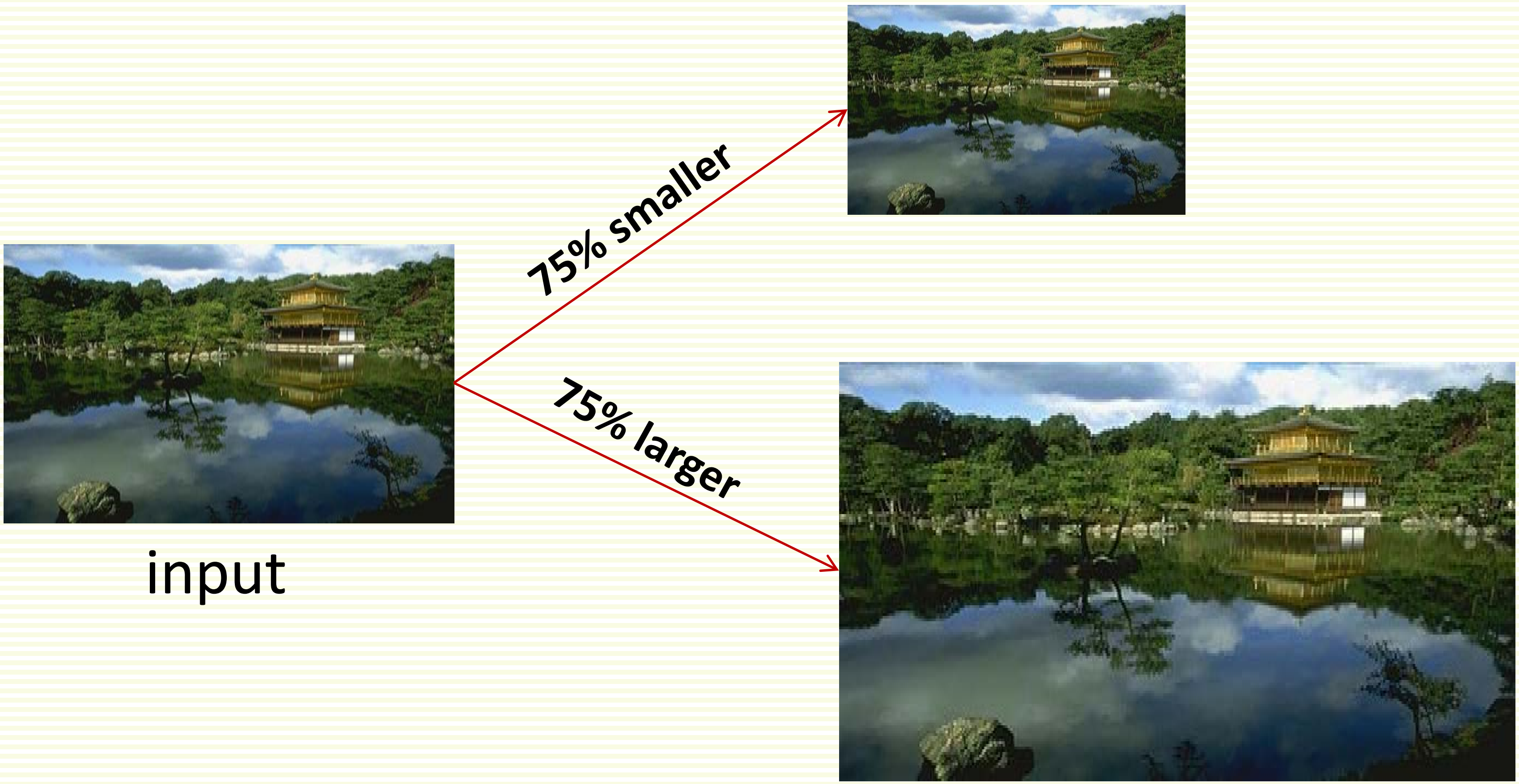




- impossible detect the "illusory" contours using only local image gradients

Application of Gradients: Intelligent Resizing

- In traditional image resizing, all dimensions change by the same ratio

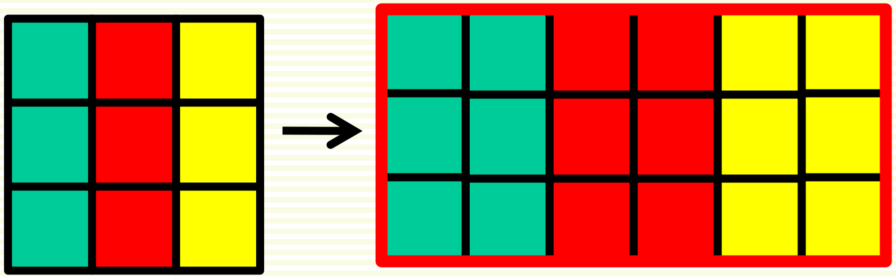


Application of Gradients: Intelligent Resizing

- What if need to fit to a mobile device? Or resize to a web page?
  - often need different ratio in different dimensions

- Change width, height unchanged

- Object proportions are not preserved:



Application of Gradients: Intelligent Resizing

- Intelligent resizing "seam carving"
  - Shai & Avidan, SIGGRAPH 2007

seam carving

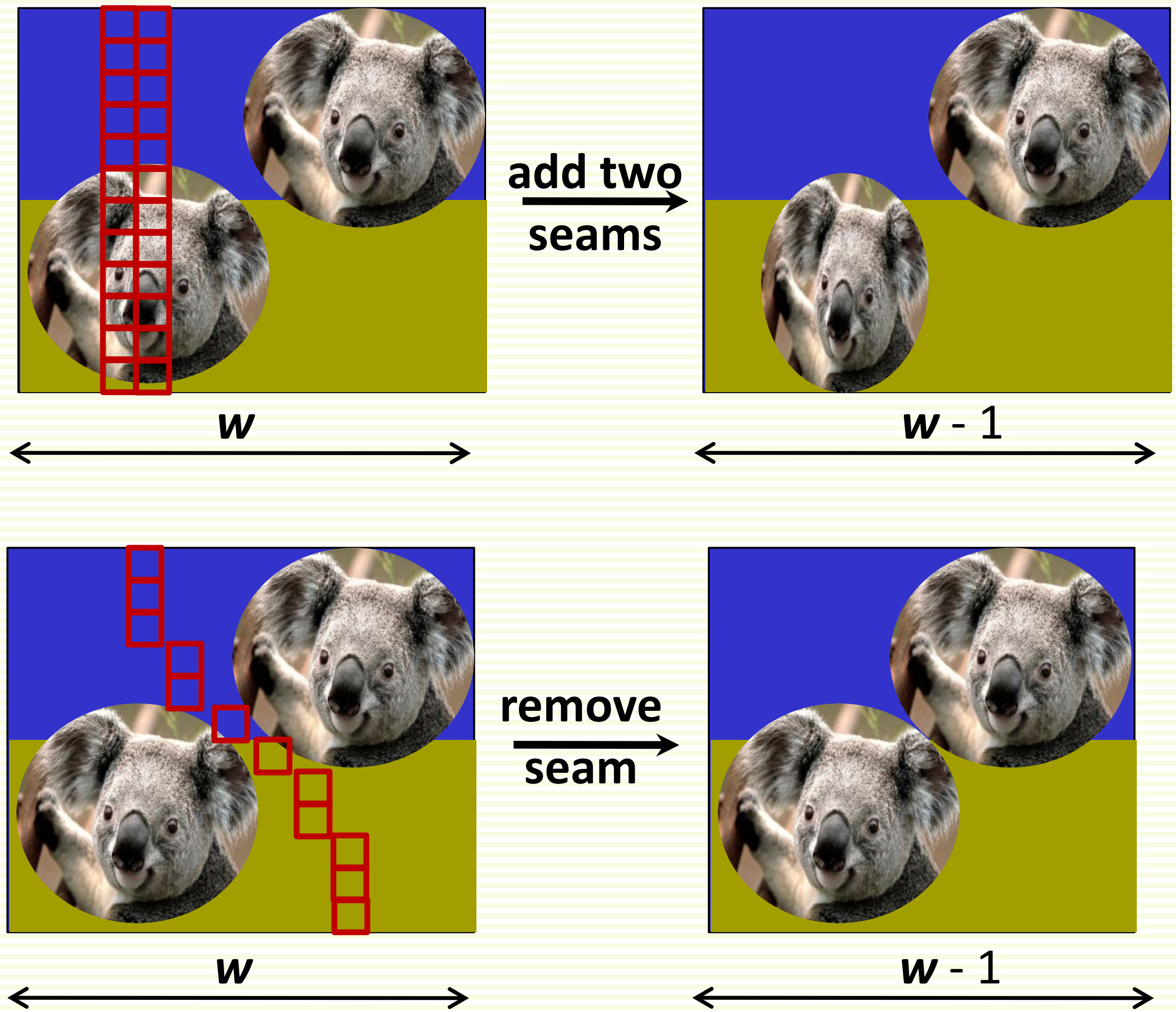
naïve resizing

Seam Carving: Main Idea

- Preserve the most “interesting” content
  - large gradient magnitude = interesting
  - small gradient magnitude = uninteresting
- ➢ **Prefer changes around low gradient magnitude pixels**

Reducing Width by One Pixel

- Traditional resizing
  - works on regular seams
  - through random pixels
- Seam carving
  - find irregular seams
  - through low gradient (uninteresting) pixels

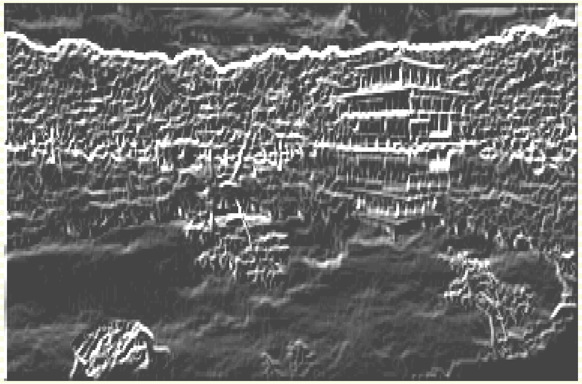


Seam Carving: Main Idea

- **Prefer changes around low gradient magnitude pixels**
  - to reduce size in one dimension, **remove** irregular seams
  - to enlarge size in one dimension, **insert** irregular seams
- Many “uninteresting” seams
  - find the best (most boring) seam
  - with dynamic programming

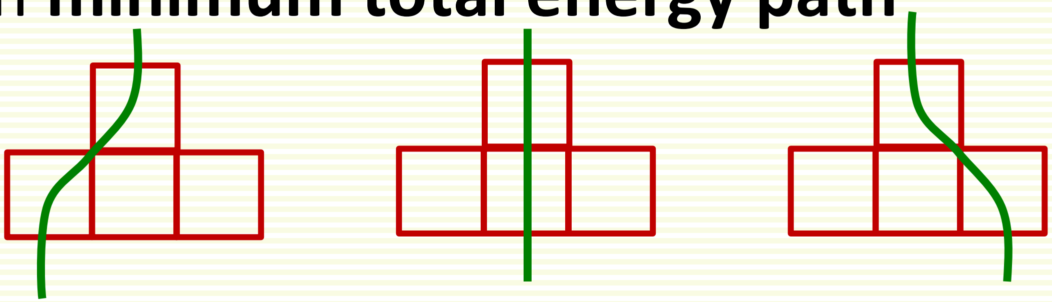
Seam Carving: Main Idea

seams



$$Energy(f) = \|\nabla f\|$$

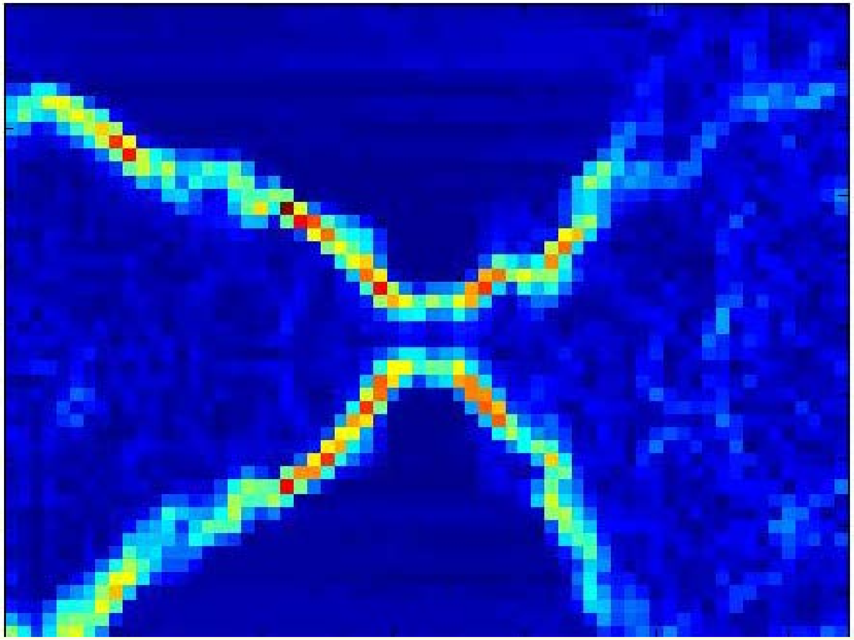
- Measure **energy** as gradient magnitude
- Removing low energy seam makes change less visible
- Choose seam based on **minimum total energy path**
- Path is **8-connected**



Example

Original Image

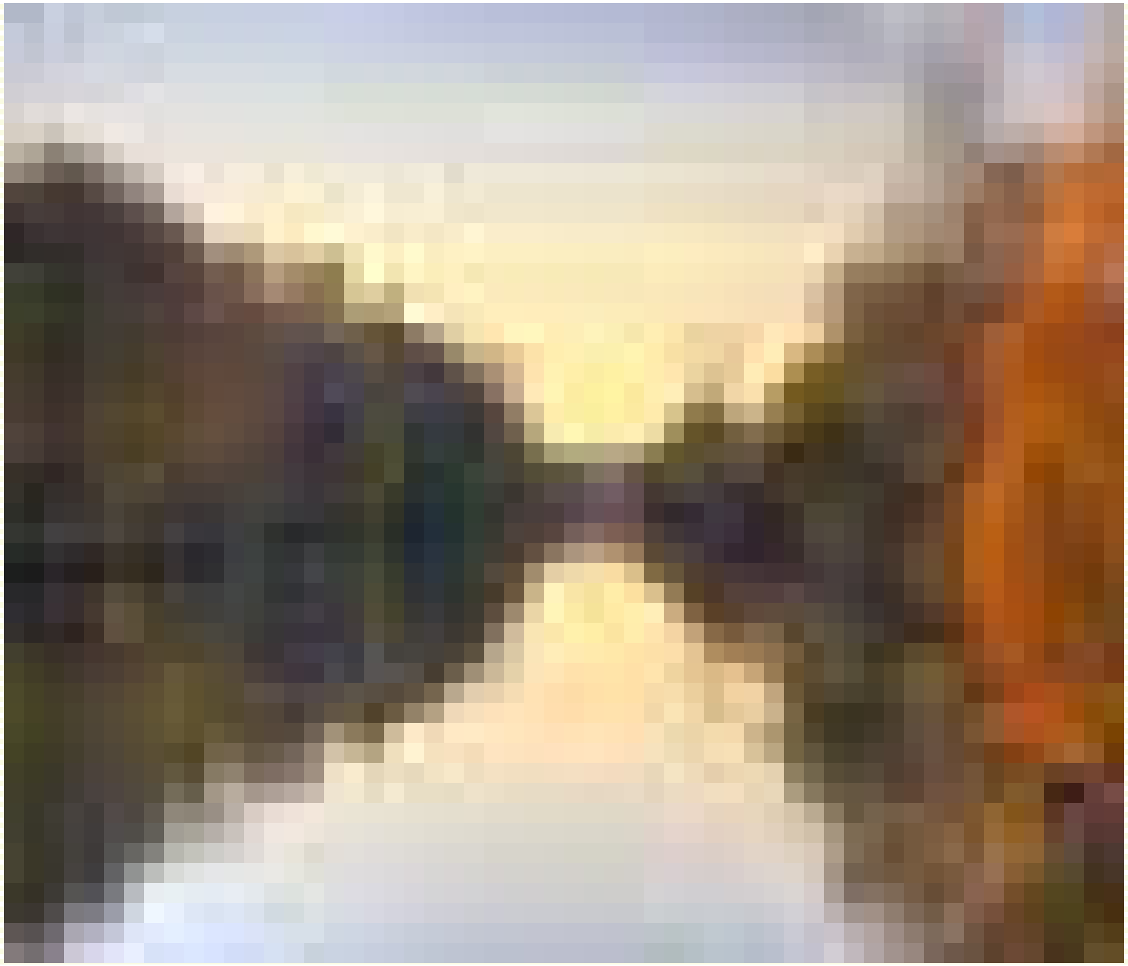
Gradient Energy



blue = low energy

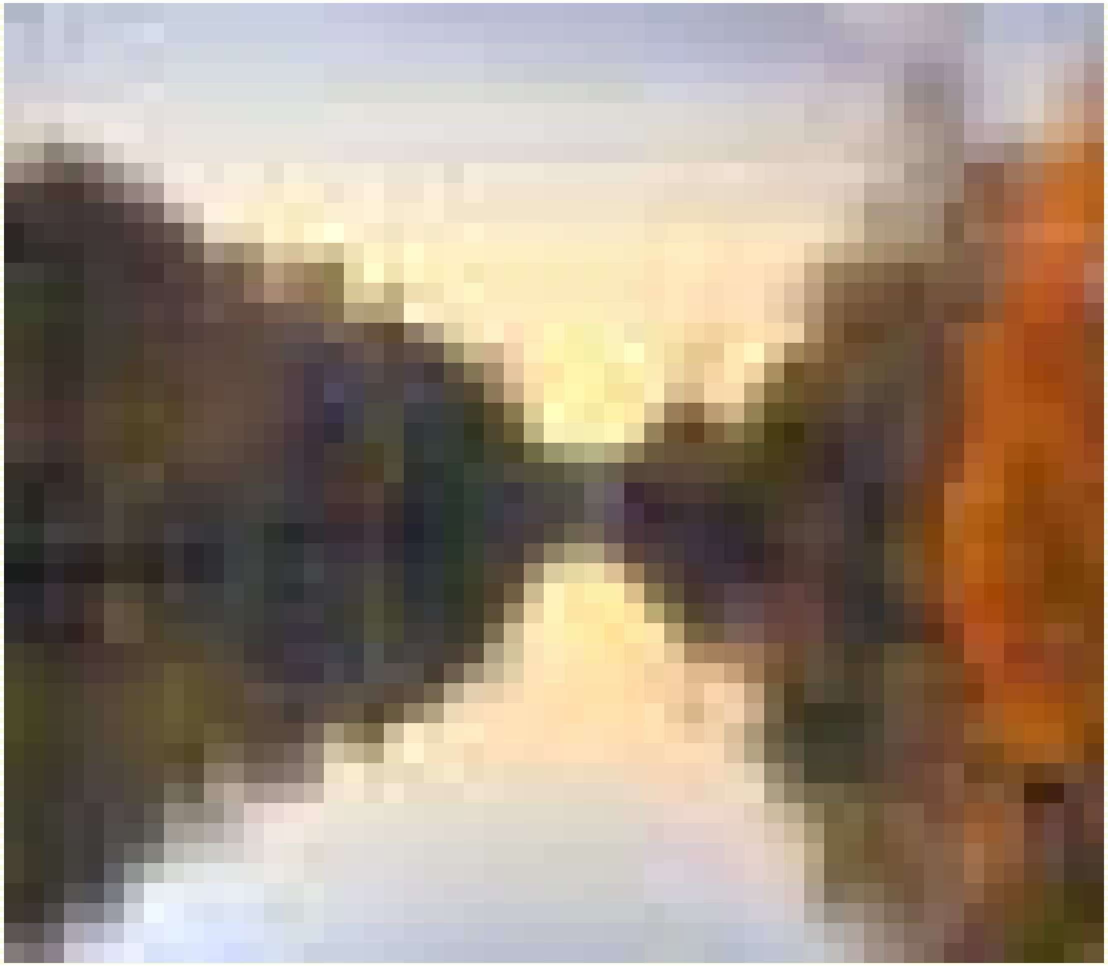
red = high energy

Example

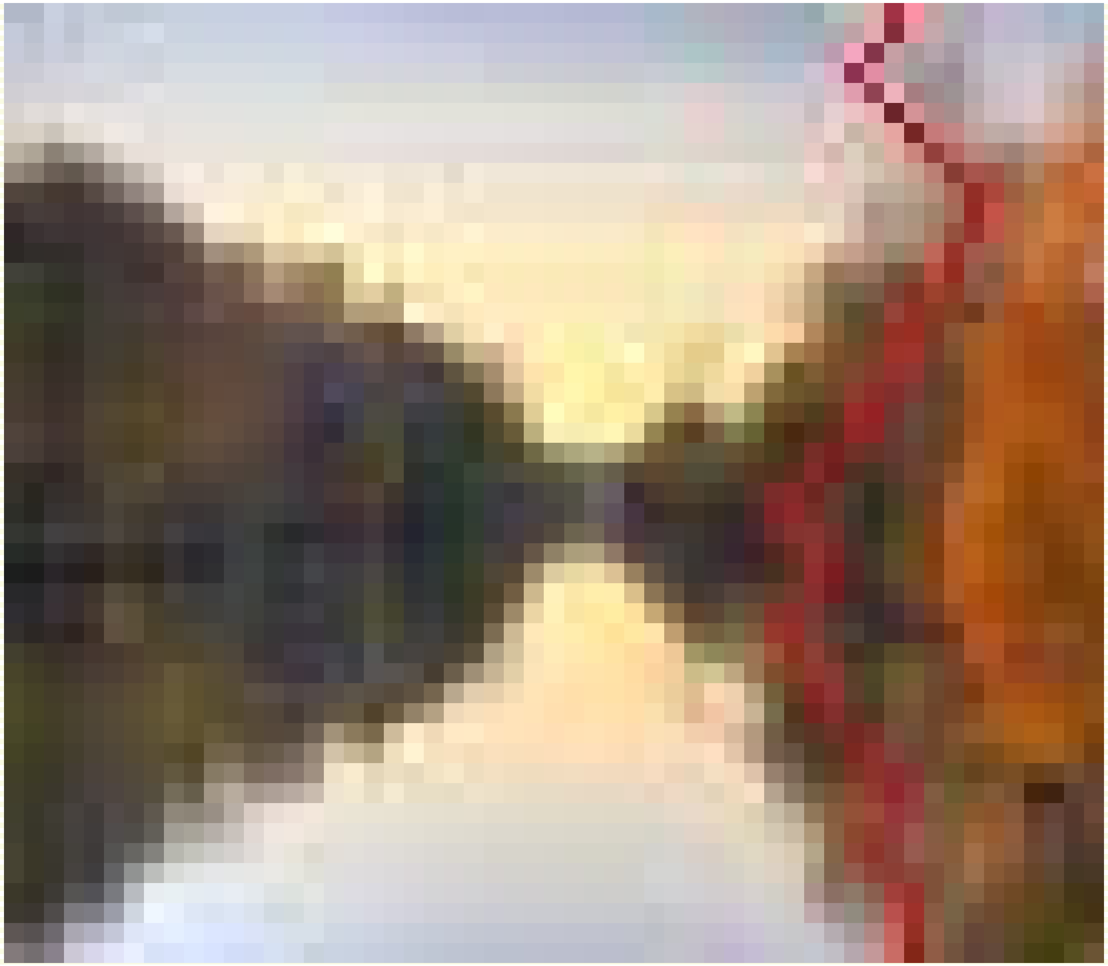


Example

Example

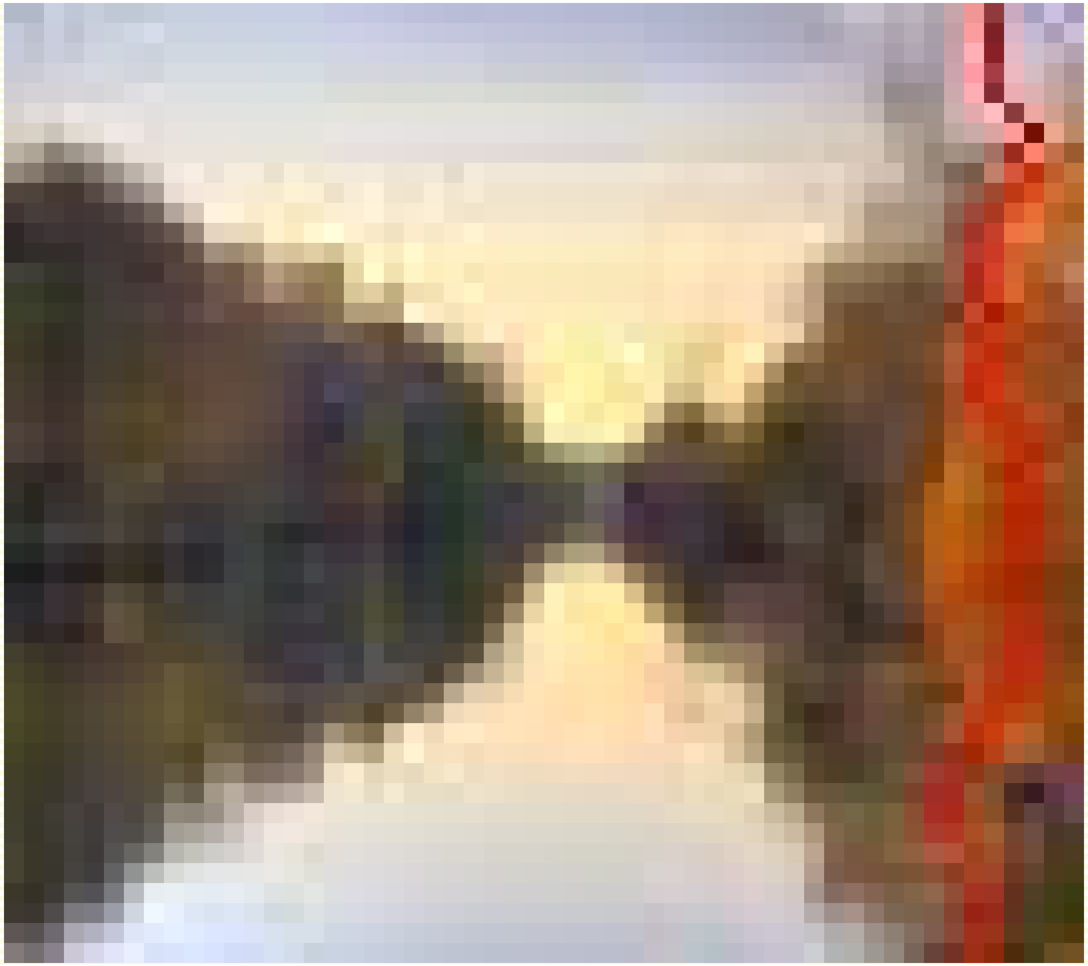


Example



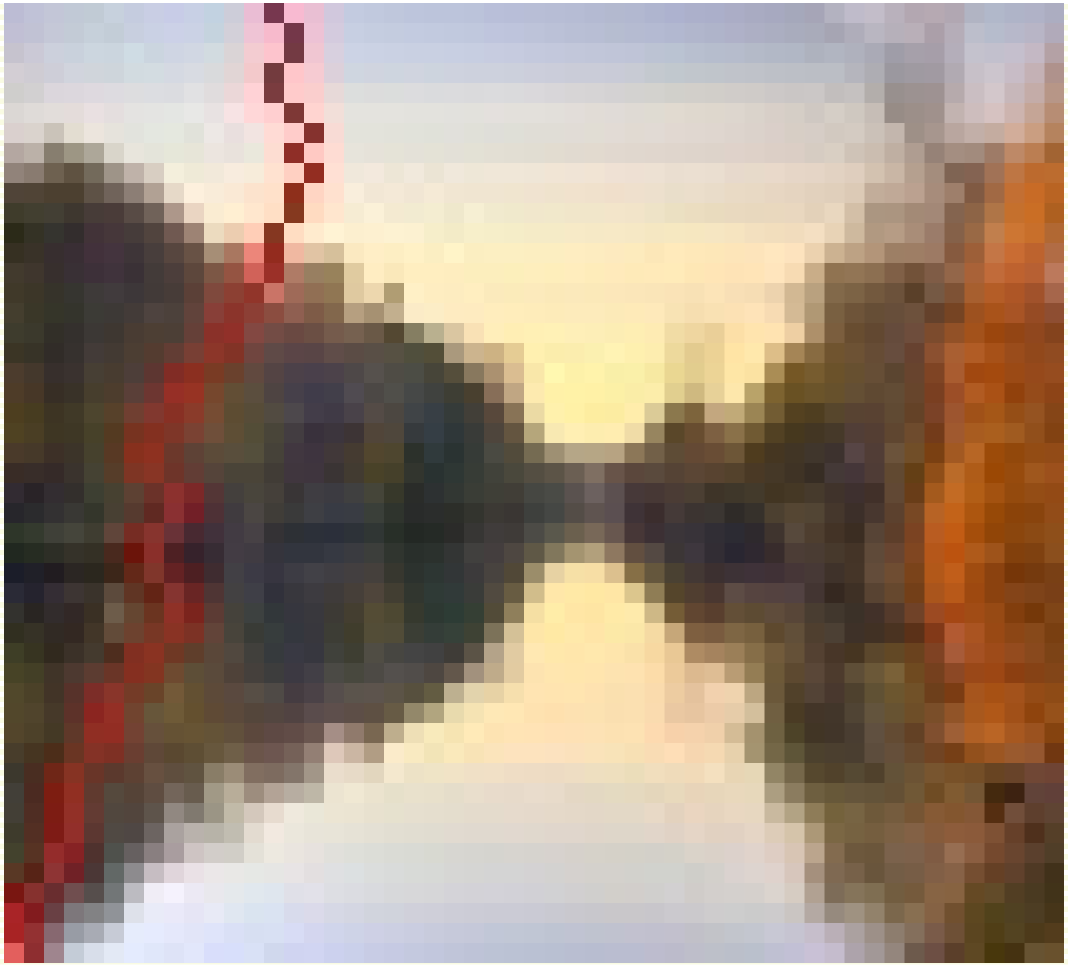
Example

Example

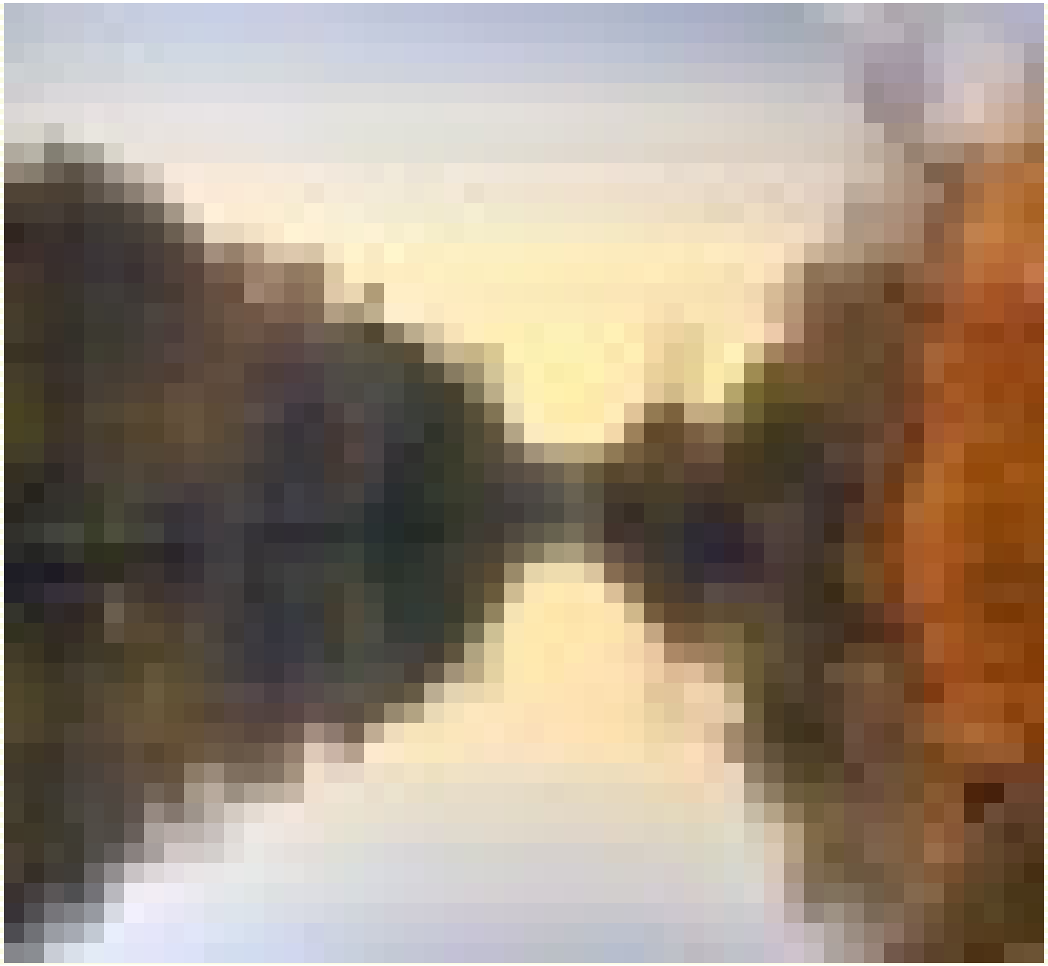


Example

Example

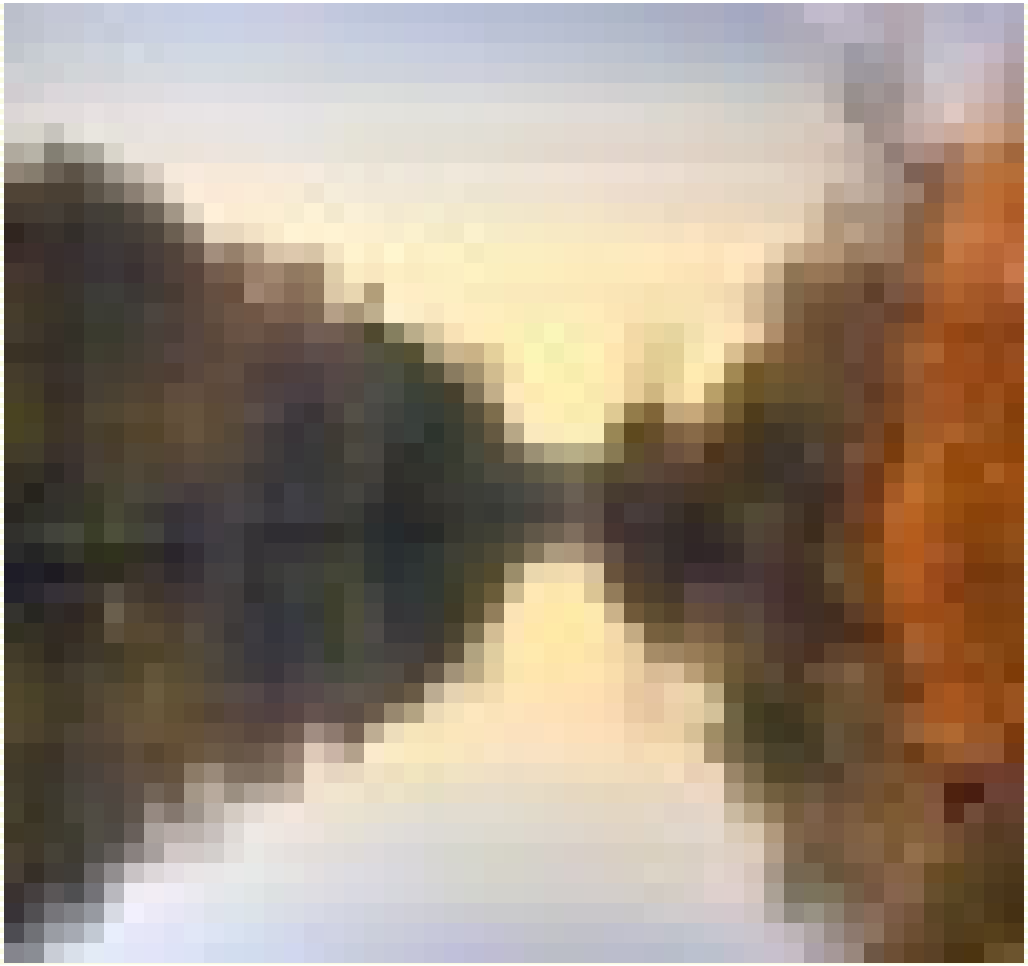


Example



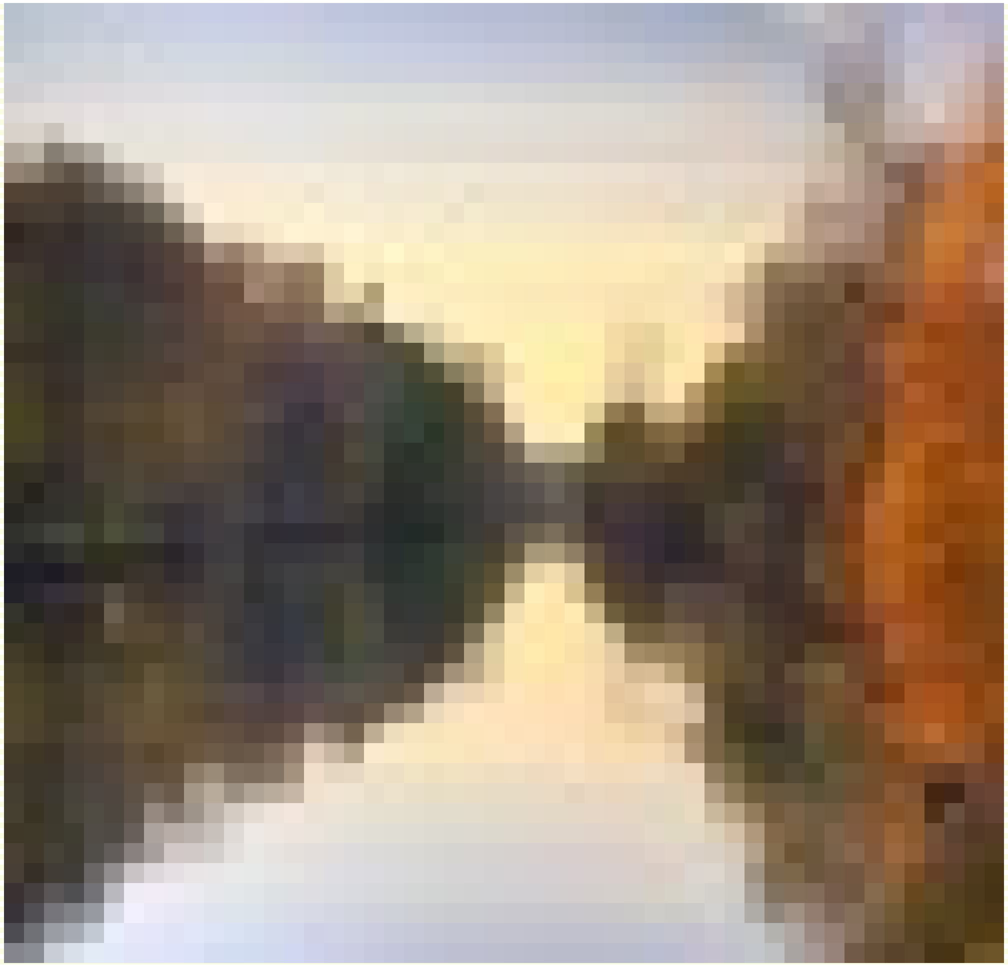
Example

Example



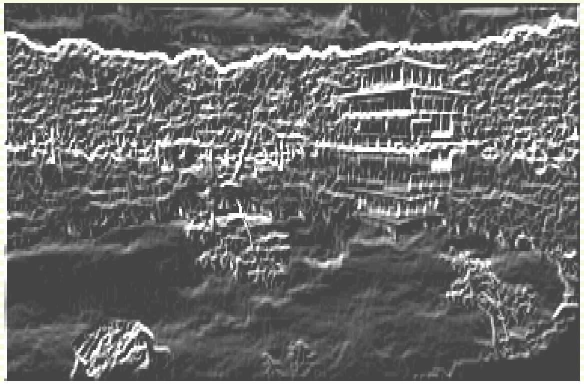
Example

Example



Seam Carving: Algorithm

3	0	6	4	2
1	3	6	6	4
4	3	4	6	2
4	6	0	0	4
1	6	5	6	6



$$Energy(f) = \|\nabla f\|$$

- Vertical seam ***s*** consists of ***n*** positions that form a path
  - ***s*** = ($\boldsymbol{s}_1$, $\boldsymbol{s}_2$,…, $\boldsymbol{s}_n$): one pixel in every row
- Seam cost ***M***(***s***) = Energy(***s***) = Energy($\boldsymbol{s}_1$)+Energy($\boldsymbol{s}_2$)+…+Energy($\boldsymbol{s}_n$)
  - red seam has cost 0 + 6 + 3 + 6 + 1 = 16
  - green seam has cost 4 + 4 + 2 + 4 + 6 = 20
- **Optimal** seam minimizes this cost

$$\mathbf{s}^* = \arg\min_{\mathbf{s}} M(\mathbf{s})$$

How to Find the Minimum Cost Seam?

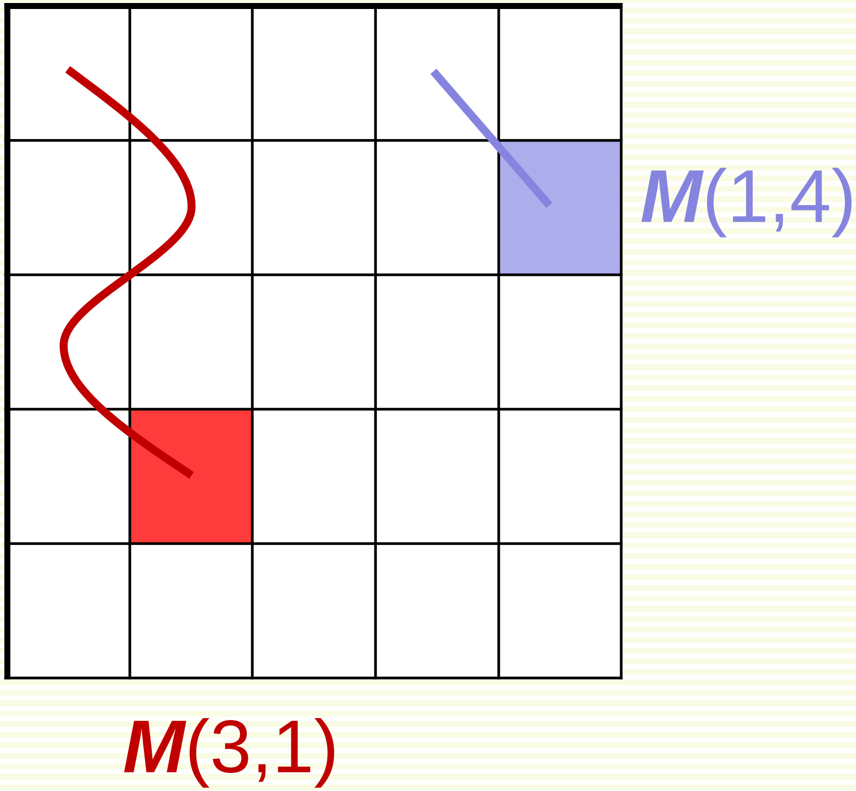
- First, consider a **greedy** approach on a small image
  - smaller number corresponds to smaller gradient

3	0	6	4	2
1	3	6	6	4
4	3	4	6	2
4	6	0	0	4
0	6	5	6	6

- Greedy seam cost: 0 + 1 + 3 + 0 + 5 = 9
- Is this the best vertical seam?

Optimal Seam Carving Algorithm

- Dynamic programming can find the best seam
  - recall POS tagging in NLP lectures
- Work from the top row to the bottom row
- ***M***(***r***,***c***) is best seam cost that starts anywhere in row 0 end ends at position (***r***,***c***)
- After computed all of ***M***, the best cost path is the smallest value of ***M*** in the last row
- Also keep track of the parent on the path, ***P***(***r***,***c***)



Seam Carving Algorithm: Initialization Step

```
Compute Energy image E
for c = 0 to maxCol
        M(0,c) = E(0,c)
        P(0,c) = null
```

3	0	6	4	2
1	3	6	6	4
4	3	4	6	2
4	6	0	0	4
0	6	5	6	6

E

3	0	6	4	2

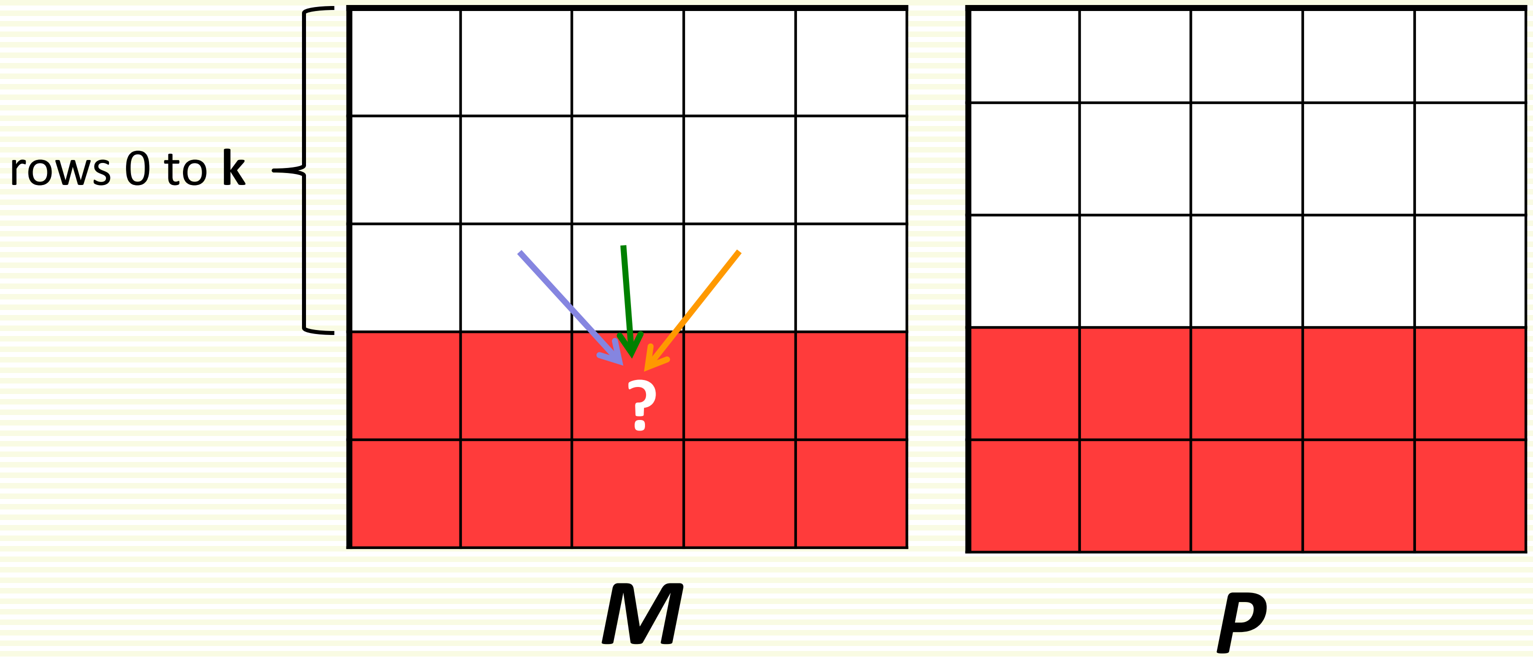
M

null	null	null	null	null

P

Seam Carving Algorithm: Iteration Step

- Computed ***M***, ***P*** for rows 0 to ***k***
- How to compute ***M***, ***P*** for row ***k***+1?



- $M(r+1,c) = E(r+1,c)$ + smallest in$\{M(r,c-1), M(r,c), M(r,c+1)\}$
- $P(r+1,c)$ stores corresponding column
  - either c-1, or c, or c+1

Optimal Seam Carving Algorithm: Iterations

```
for r = 0 to maxRow
    for c = 0 to maxCol
        option1 = M(r-1,c-1)
        option2 = M(r-1,c)
        option3 = M(r-1,c+1)
        if option1 ≤ option2 and option1 ≤ option3
            M(r,c)= E(r,c) + M(r-1,c-1)
            P(r,c)=c-1
        elseif option2 ≤ option1 and option2 ≤ option3
            M(r,c)= E(r,c) + M(r-1,c)
            P(r,c)= c
        else
            M(r,c)= E(r,c) + M(r-1,c+1)
            P(r,c)= c+1
```

!!!Note: have to implement matrix out of bounds check!!!

Example: Initialization

3	0	6	4	2
1	3	6	6	4
4	3	4	6	2
4	6	0	0	4
0	6	5	6	6

E

3	0	6	4	2

M

null	null	null	null	null

P

Example: Iteration 1

3	0	6	4	2
1	3	6	6	4
4	3	4	6	2
4	6	0	0	4
0	6	5	6	6

E

3	0	6	4	2
1	3	6	8	6

M

null	null	null	null	null
1	1	1	4	4

P

Example: Iteration 2

3	0	6	4	2
1	3	6	6	4
4	3	4	6	2
4	6	0	0	4
0	6	5	6	6

E

3	0	6	4	2
1	3	6	8	6
5	4	7	12	8

M

null	null	null	null	null
1	1	1	4	4
0	0	1	2	4

P

Example: Iteration 3

3	0	6	4	2
1	3	6	6	4
4	3	4	6	2
4	6	0	0	4
0	6	5	6	6

E

3	0	6	4	2
1	3	6	8	6
5	4	7	12	8
8	10	4	7	12

M

null	null	null	null	null
1	1	1	4	4
0	0	1	2	4
1	1	1	2	4

P

Example: Iteration 4

- Best seam has cost 8, better than what greedy algorithm finds
  - end of the best seam is in column 0

3	0	6	4	2
1	3	6	6	4
4	3	4	6	2
4	6	0	0	4
0	6	5	6	6

E

3	0	6	4	2
1	3	6	8	6
5	4	7	12	8
8	10	4	7	12
8	10	9	10	13

M

null	null	null	null	null
1	1	1	4	4
0	0	1	2	4
1	1	1	2	4
0	2	2	2	3

P

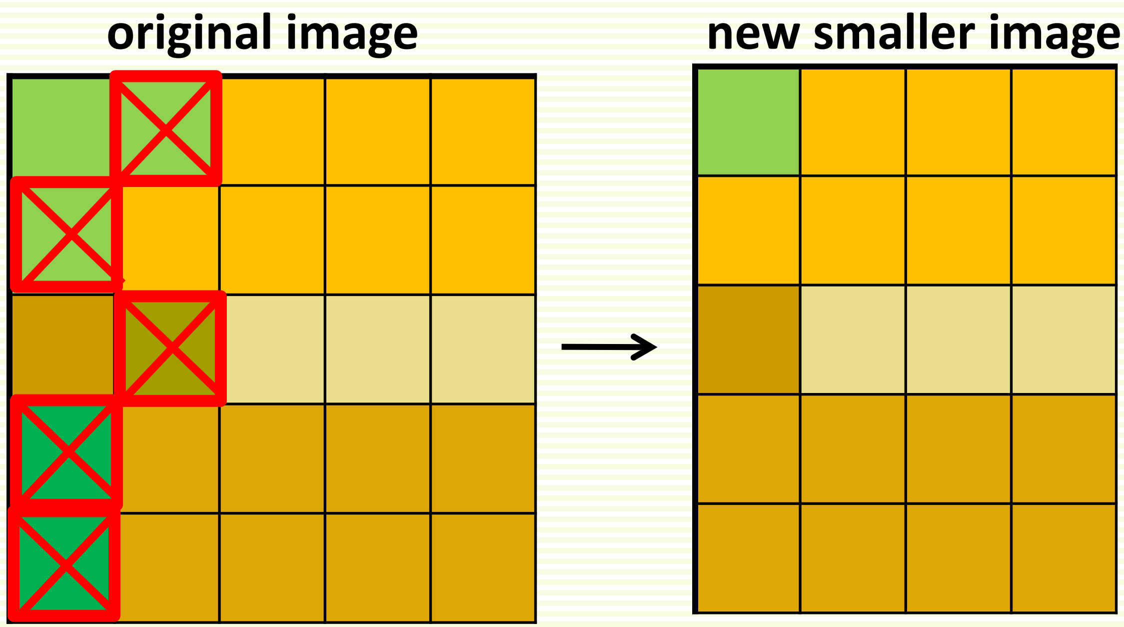
Example: Finishing Up

M

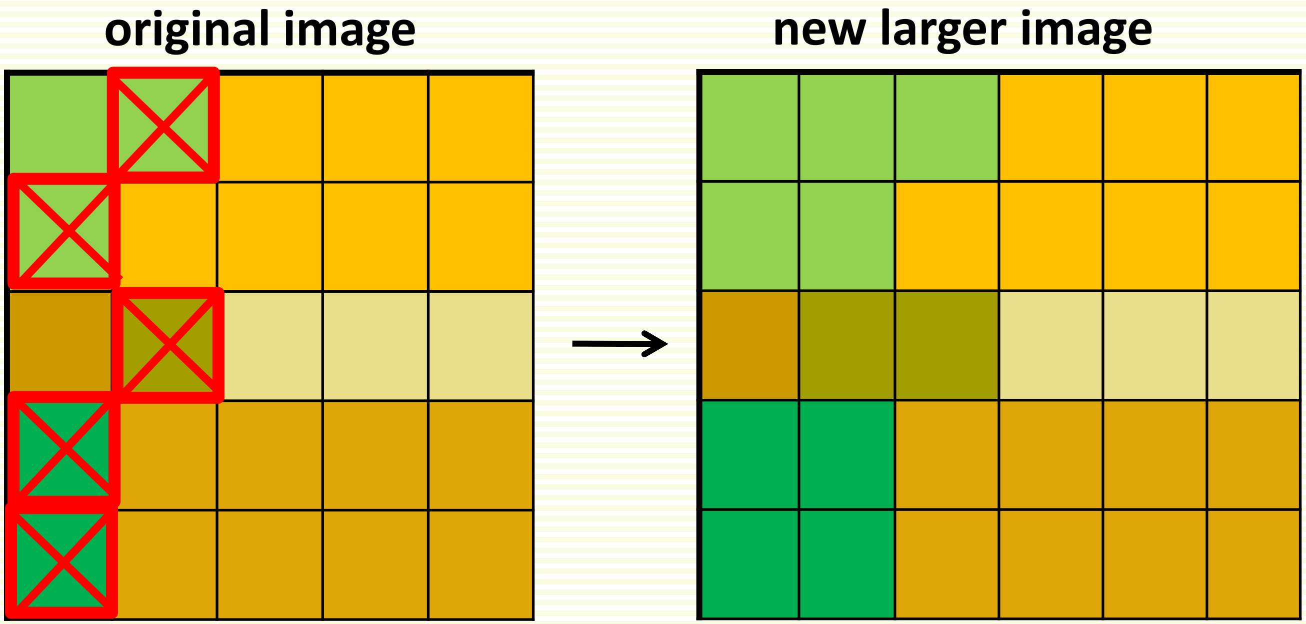
3	0	6	4	2
1	3	6	8	6
5	4	7	12	8
8	10	4	7	12
8	10	9	10	13

P

null	null	null	null	null
1	1	1	4	4
0	0	1	2	4
1	1	1	2	4
0	2	2	2	3



Other notes on seam carving



- Can also insert seams to *increase* size of image
  - duplicate optimal seam, averaged with neighbors
- Analogous procedure for horizontal seams
- Other energy functions may be plugged in
  - e.g., color-based
- Can remove (or keep, or enlarge) marked objects

Some Results

brings friends closer

or draws them apart

Include Color in Energy

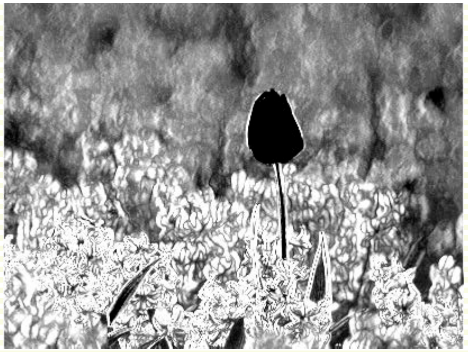
- Want to remove objects of red color
  - R_f , G_f, B_f are red, green blue color channels of image *f*

$\|\nabla f\|$

+

$-2\cdot R_f + G_f + B_f$

=



$energy(f)$

input image *f*

carving out red

Include Color in Energy

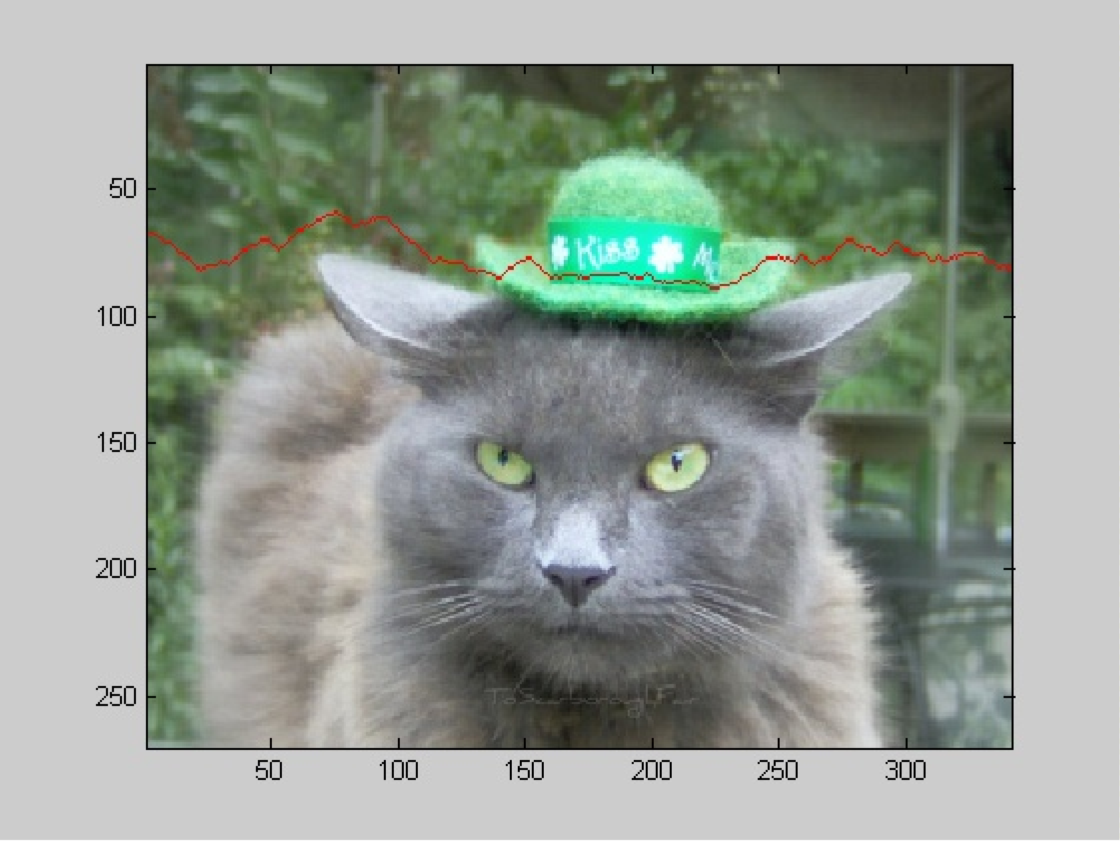
- That hat is too big - get rid of some green

$$energy(f) = \|\nabla f\| - 2G_f + R_f + B_f$$

input image f

carved

Some Results



Removal of a Marked Object

- Mask image ***M*** is 1 for object, 0 otherwise
  - remove vertical and horizontal seams

$\|\nabla f\|$

- 1000·

M

=

$energy(f)$

Insert More Marked Object

- Same energy, now insert vertical seams

$\|\nabla f\|$

- 1000·

$\boldsymbol{M}$

=

$energy(f)$

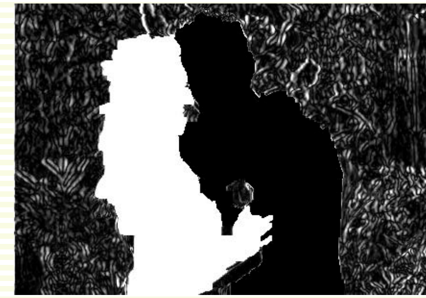
Sometimes Remove Mask not Enough

Remove and Preserve Mask

- ***M*** is 1 for pixels to remove, -1 for pixels to keep, 0 for neutral

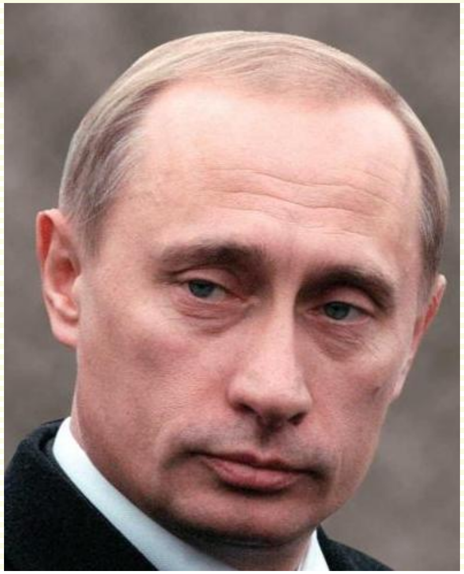
- 1000·

=



$$\|\nabla f\| - 1000 \cdot \boldsymbol{M} = energy(f)$$

Carving a Caricature



Sometimes it Fails