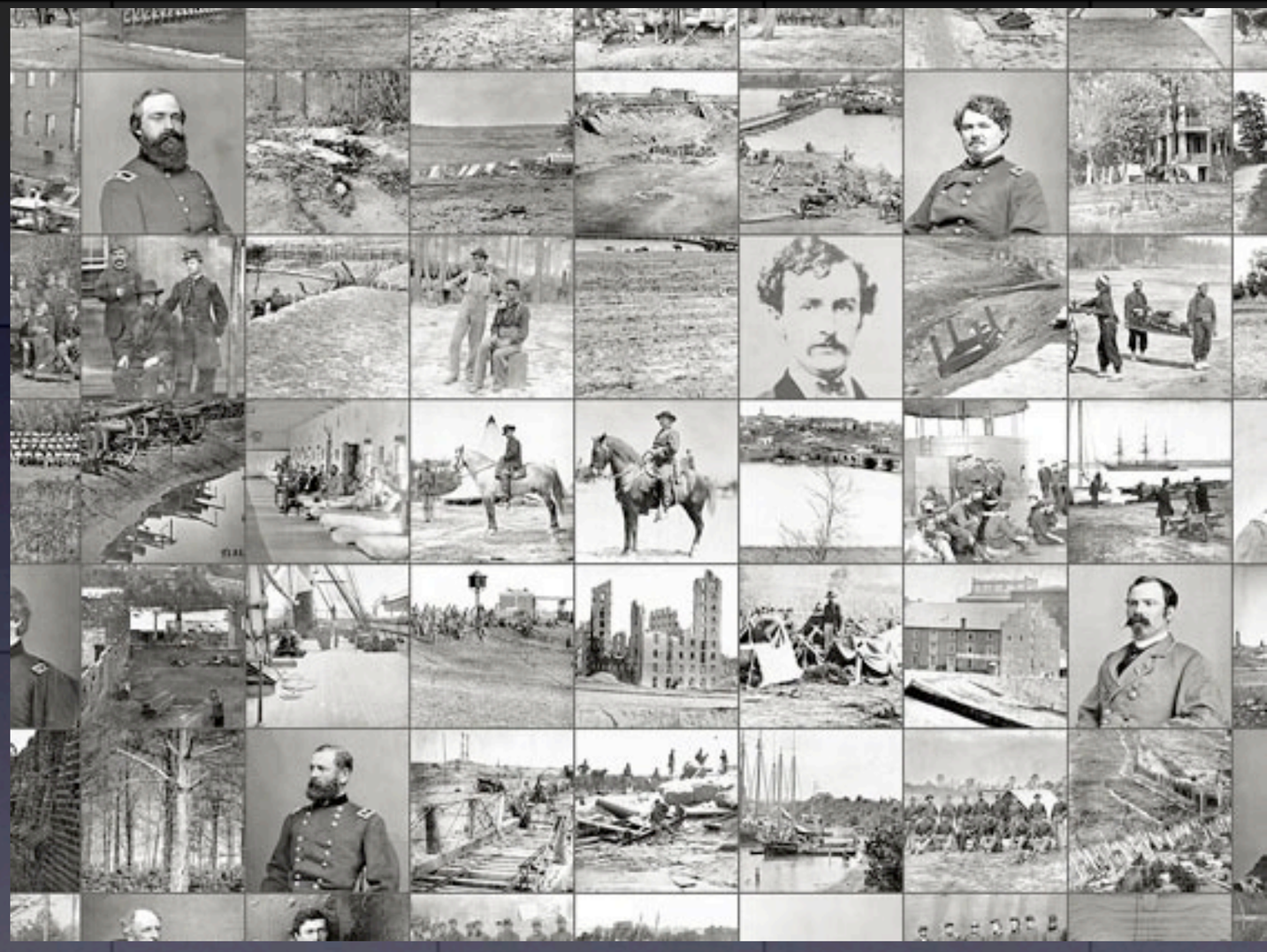
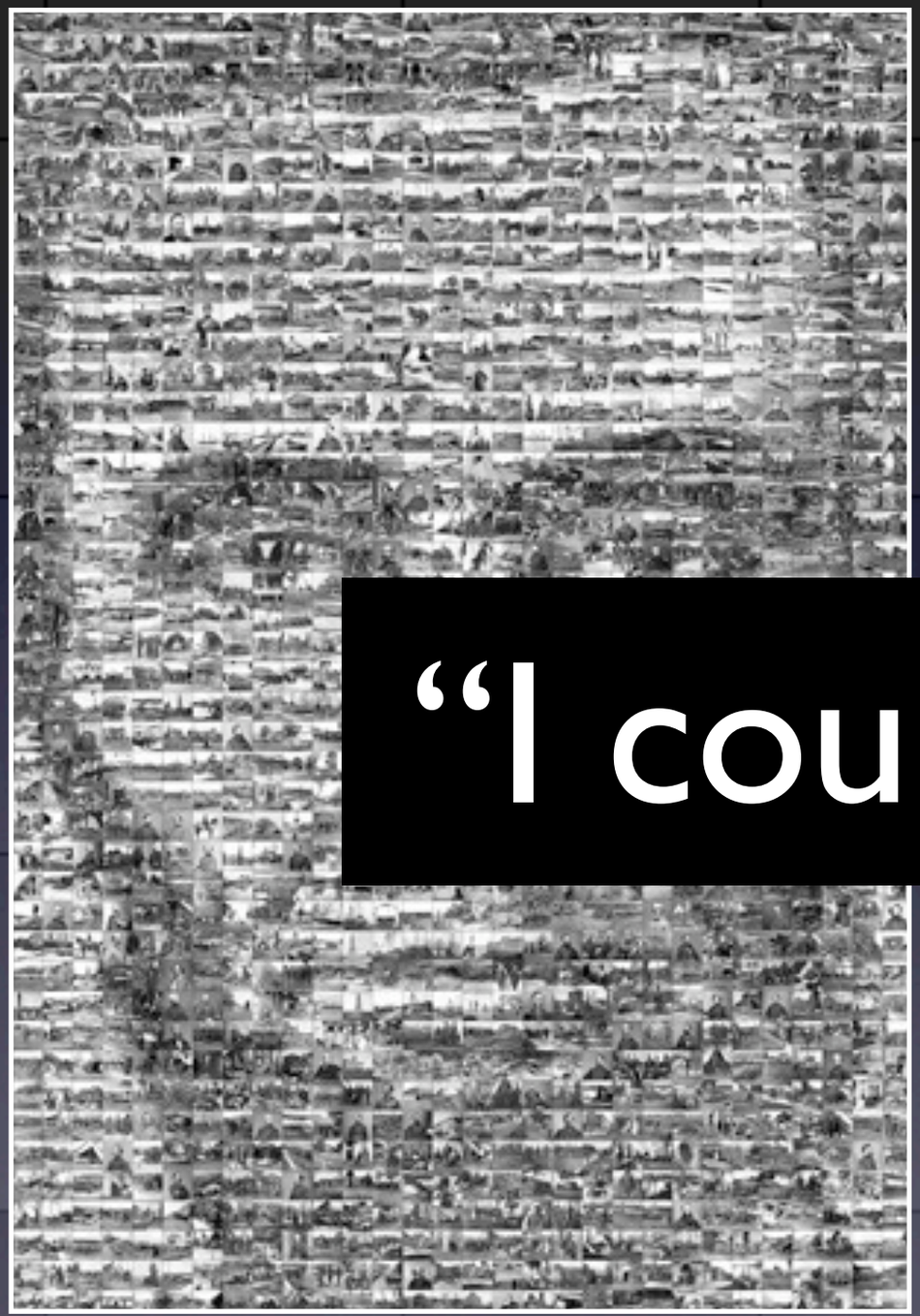
Who Needs Photoshop?

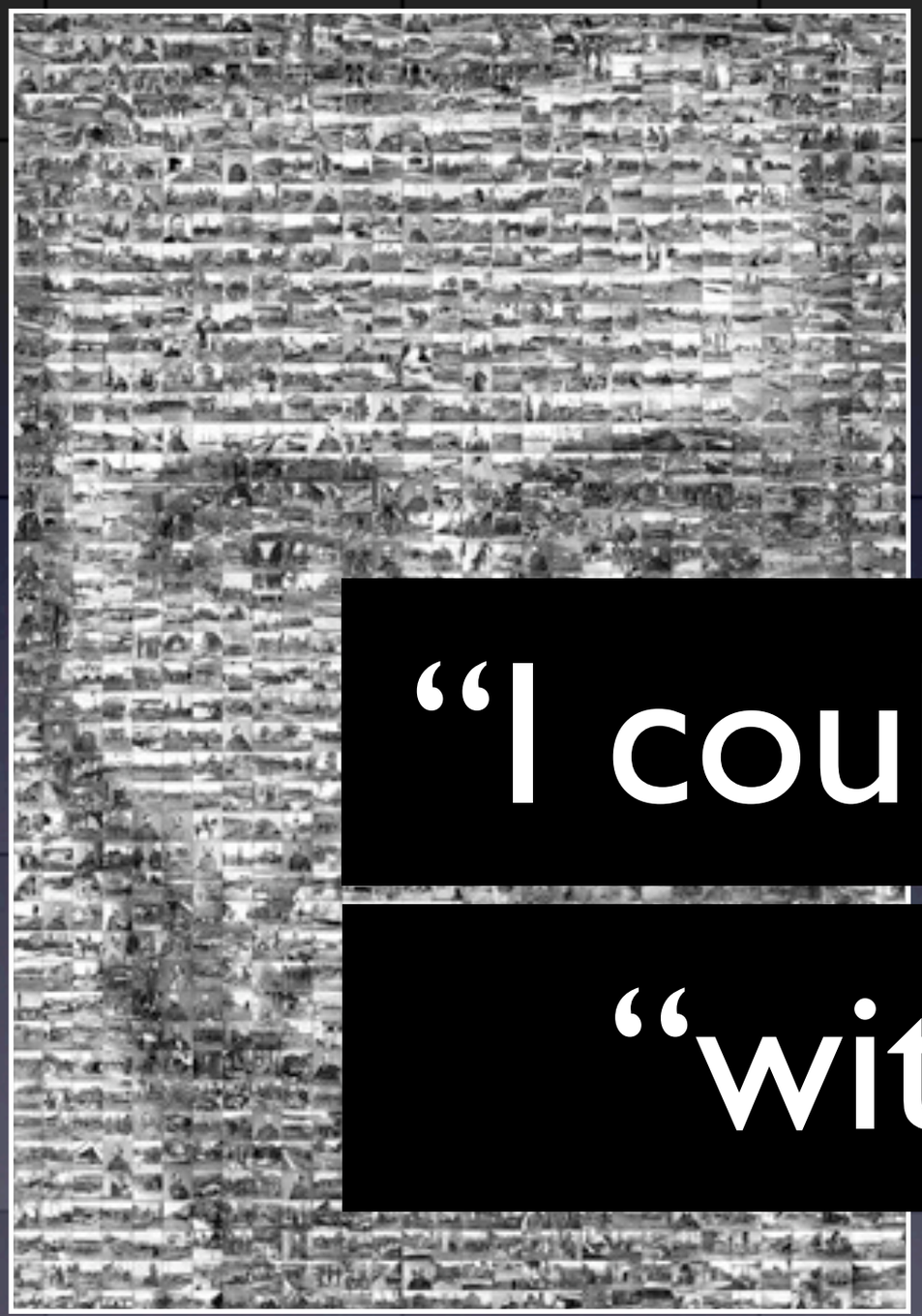
Creative image manipulation and experimentation with Ruby

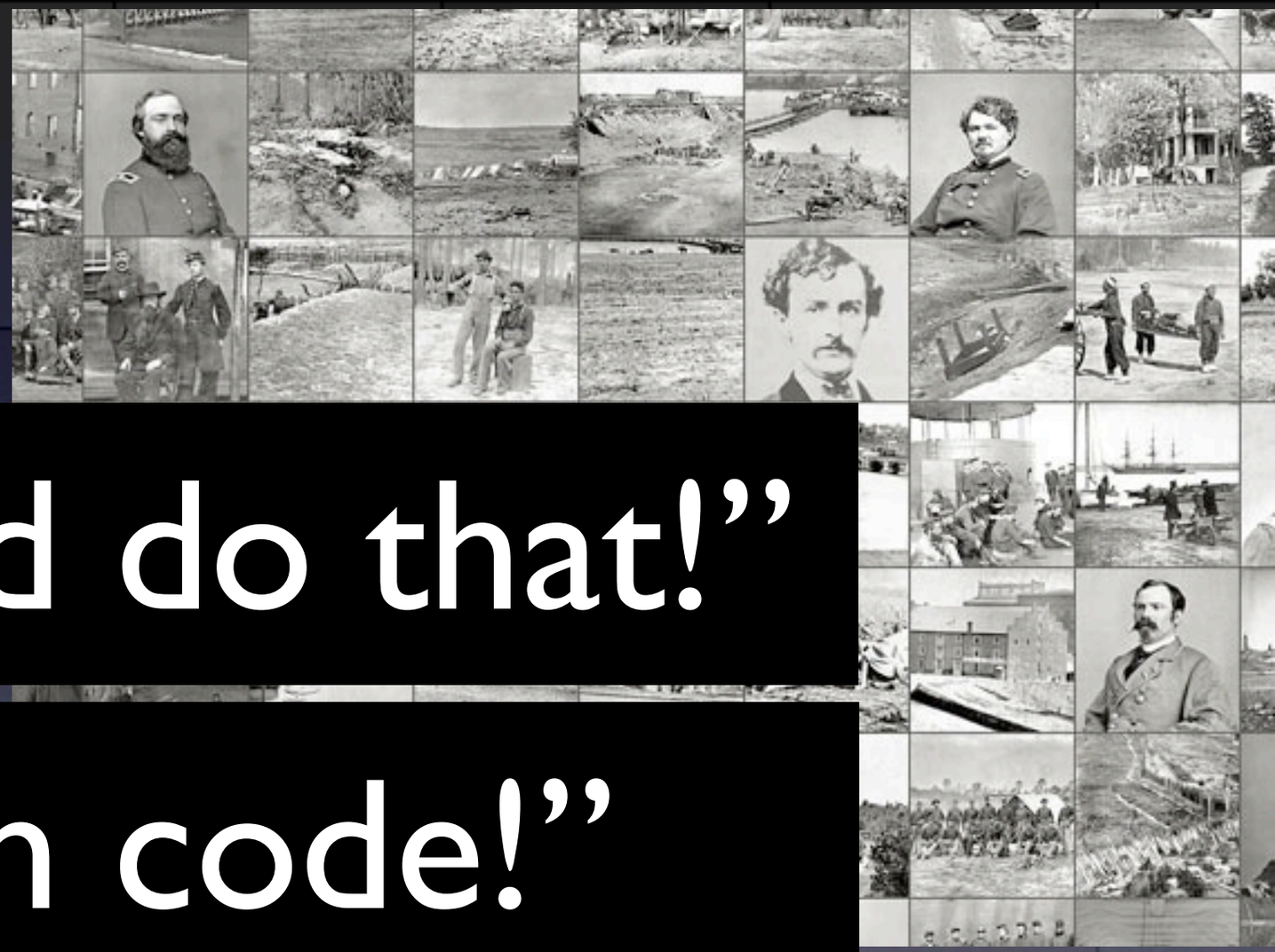



“I could do that!”



“I could do that!”





“I could do that!”

“with code!”

code is at http://github.com/bantic/image_labs /lib/blender

GUI

code is at http://github.com/bantic/image_labs /lib/blender

GUI

LIBRARIES/API

code is at http://github.com/bantic/image_labs /lib/blender

GUI

LIBRARIES/API

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

code is at http://github.com/bantic/image_labs /lib/blender

GUI

Contrast

Blur

Change Hue

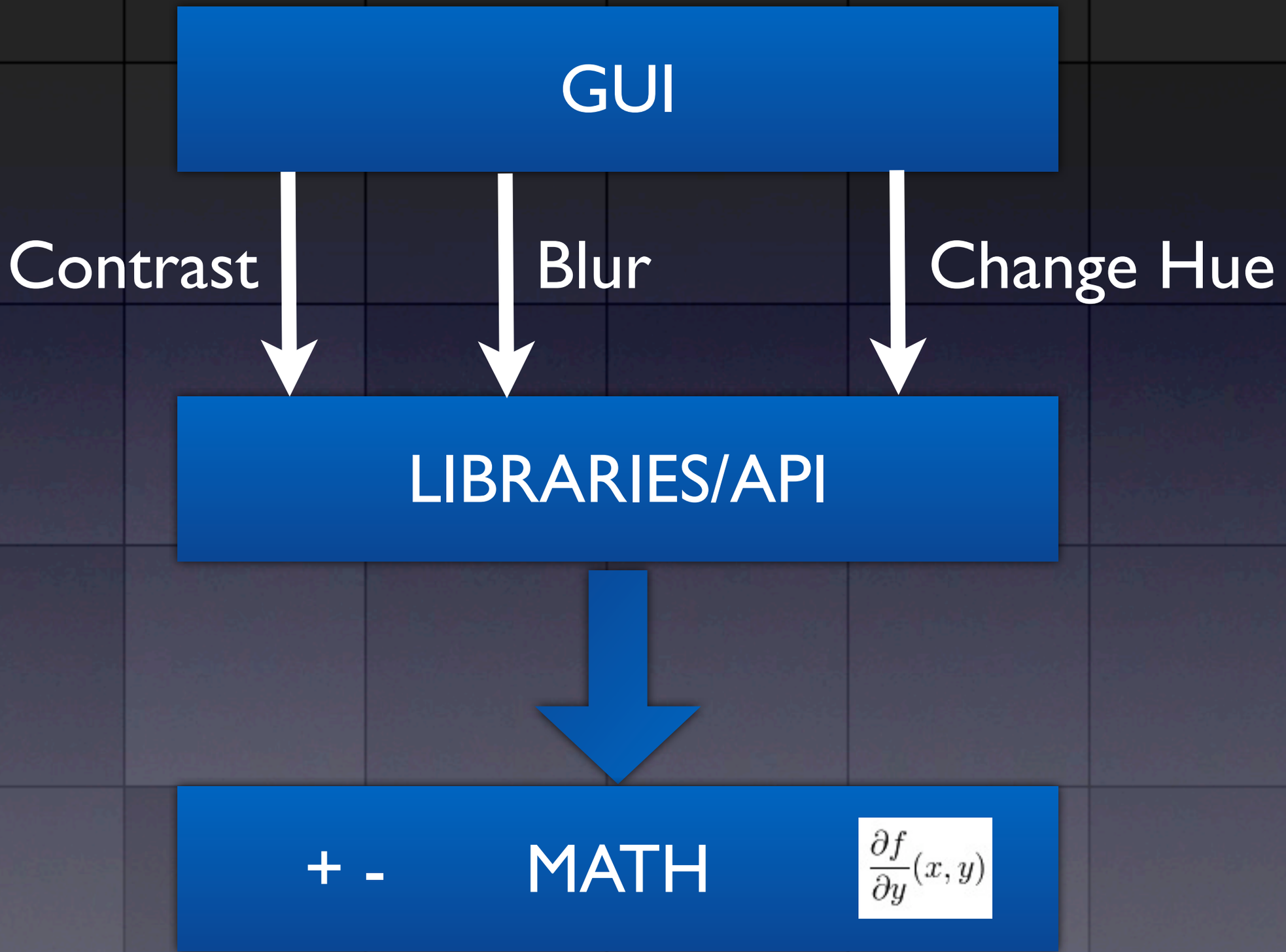
LIBRARIES/API

+ -

MATH

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

code is at http://github.com/bantic/image_labs /lib/blender



code is at http://github.com/bantic/image_labs /lib/blender

code is at http://github.com/bantic/image_labs /lib/blender

Ruby

code is at http://github.com/bantic/image_labs /lib/blender

Ruby

RMagick

ImageMagick

code is at http://github.com/bantic/image_labs /lib/blender

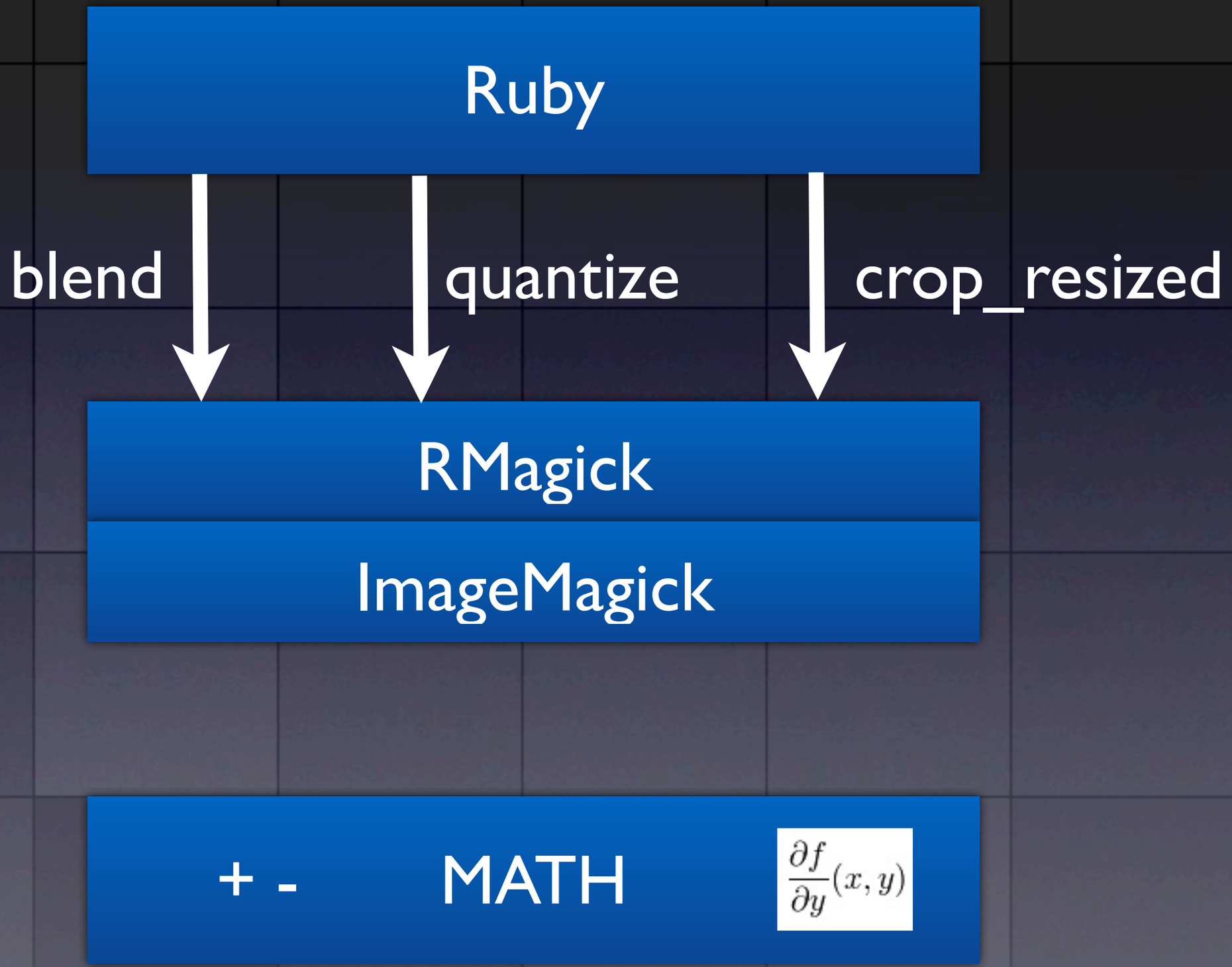
Ruby

RMagick

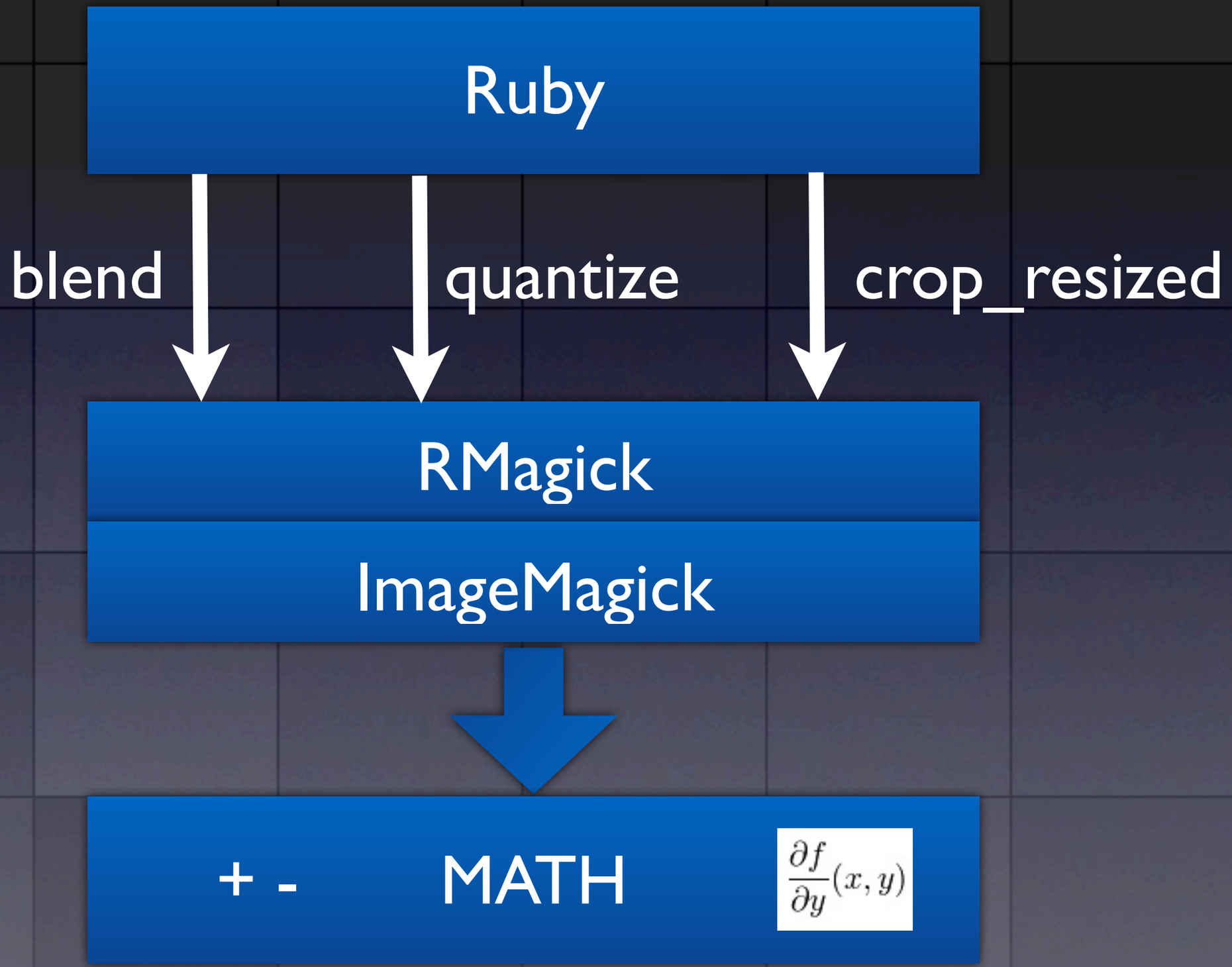
ImageMagick

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

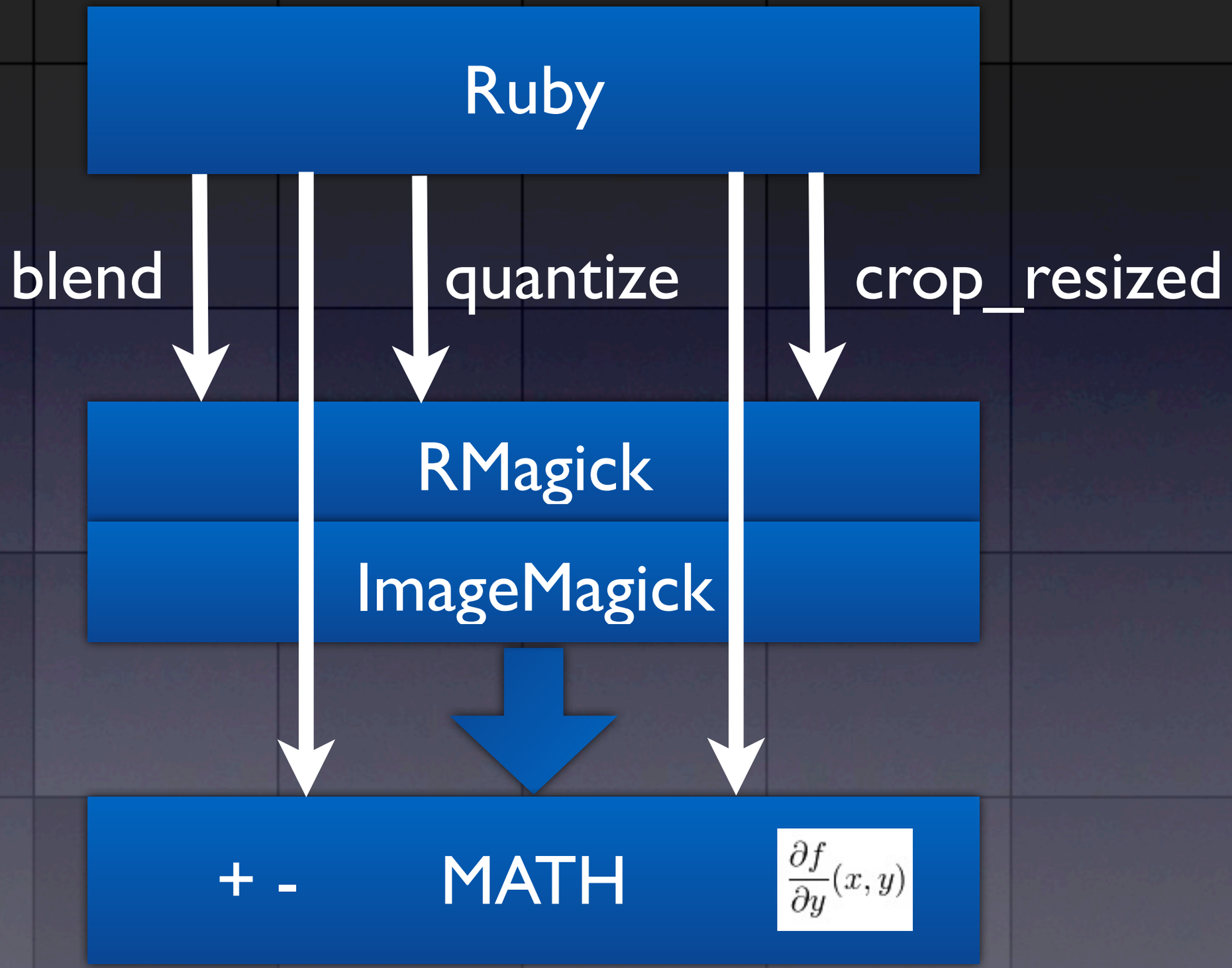
code is at http://github.com/bantic/image_labs /lib/blender



code is at http://github.com/bantic/image_labs /lib/blender



code is at http://github.com/bantic/image_labs /lib/blender



code is at http://github.com/bantic/image_labs /lib/blender

state of the art

thumbnails

- RMagick
- PhotoMosaic
- SeamCarving
- Face Detection with OpenCV

code

- available on github: http://github.com/bantic/image_labs
- libraries for PhotoMosaic creation, layer blending, Seam Carving, etc.

code is at http://github.com/bantic/image_labs

RMagick Intro

- wrapper around ImageMagick C API
- can be used for manipulating images
- also be used for drawing on images
- known for being a memory hog

code is at http://github.com/bantic/image_labs

RMagick Glossary

- Pixel.new(r,g,b)
- ImageList.new(path_to_image)
  - pixel iterators
- Draw object

code is at http://github.com/bantic/image_labs


flickr credit: mgrenner57

Tuesday, May 12, 2009

RMagick Pixel by Pixel

out

- Image#export_pixels
- Image#get_pixels
- Image#dispatch

in

- Image#import_pixels
- Image#store_pixels
- Image#constitute

Image#view (never use)

code is at http://github.com/bantic/image_labs

pure rmagick functions

out

in

code is at http://github.com/bantic/image_labs

image_labs

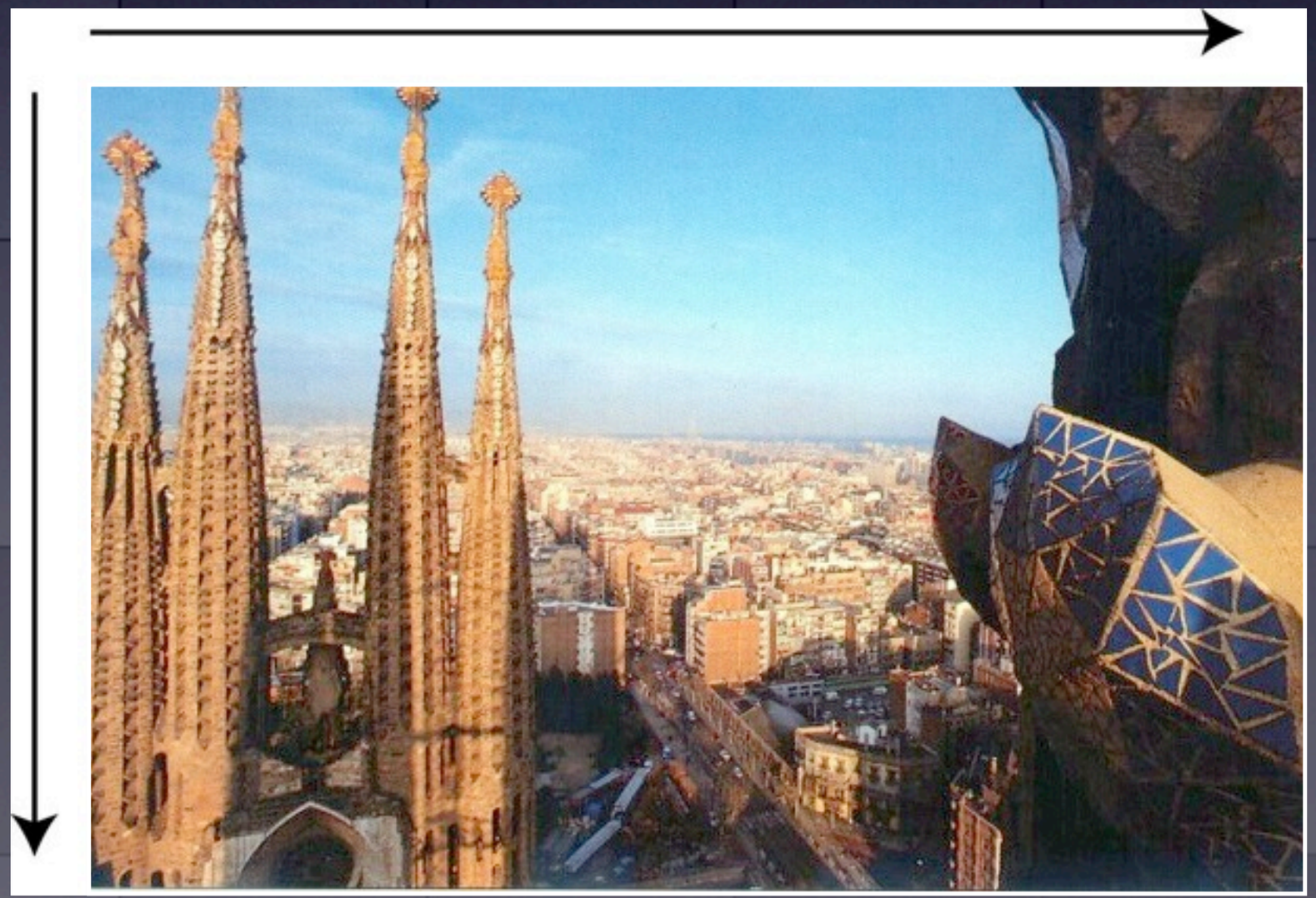
- bin/rmagicksh -- interactive shell
- Convenience functions
  - img(file) => Magick::ImageList.new(file)
- Pixel iterators
  - Image#all_pixels => array of pixels
  - Image#all_pixels_as_arrays => [[r,g,b]*]
  - Image#two_d_array_of_pixels
    - array = img.two_d_array_of_pixels
    - array[column][row] = pixel at column,row

code is at http://github.com/bantic/image_labs

RMagick Pixel by Pixel

x
width
columns

0,0



y
height
rows

- RMagick
- PhotoMosaic
- SeamCarving
- Face Detection with OpenCV

- RMagick
- PhotoMosaic
- SeamCarving
- Face Detection with OpenCV

create a photomosaic

- analyze regions of the image for intensity
- replace them with new “pixels:”
  - images of the same intensity

code is at http://github.com/bantic/image_labs /lib/mosaicer

code is at http://github.com/bantic/image_labs /lib/mosaicer

code is at http://github.com/bantic/image_labs /lib/mosaicer

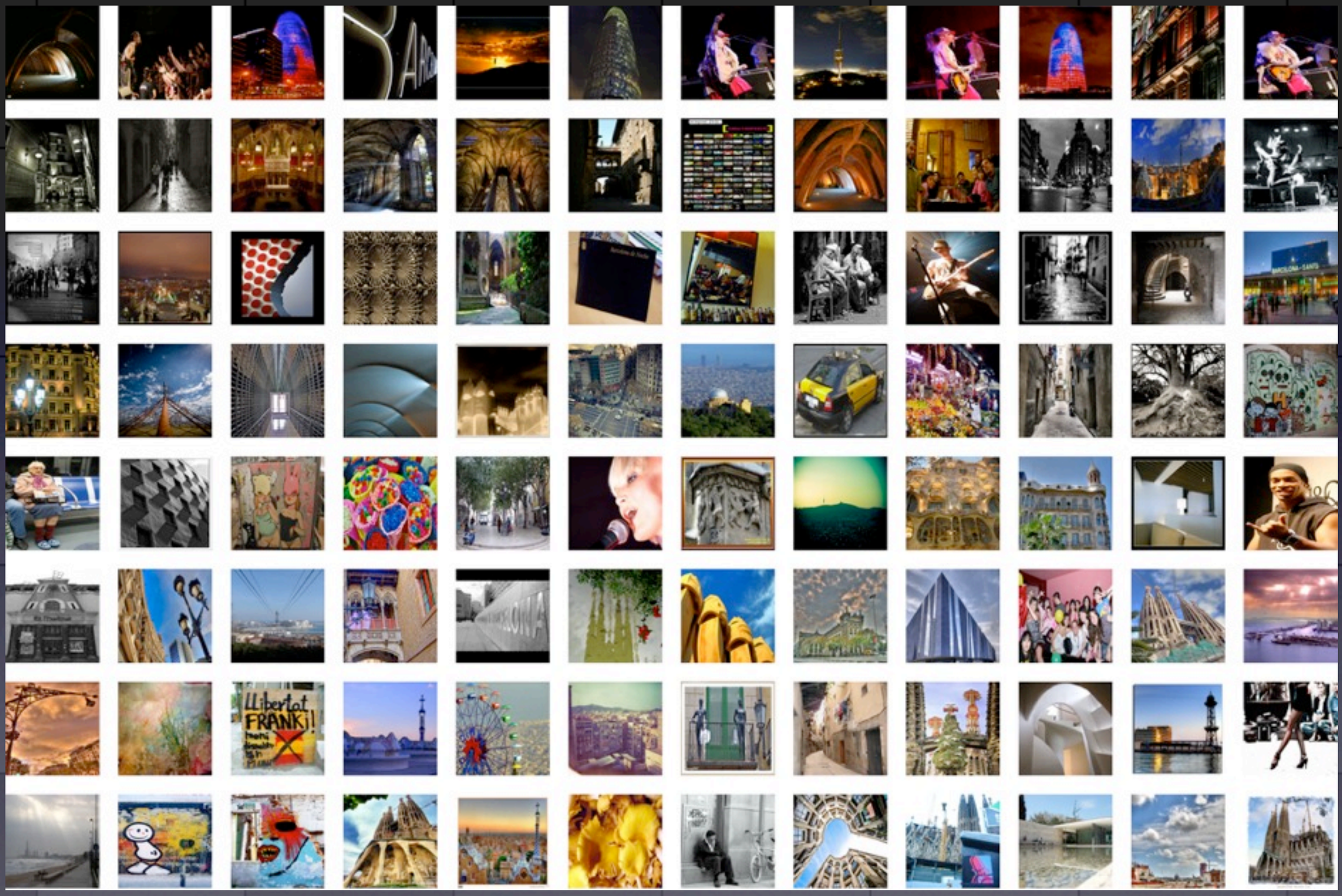
```
>> cfour:image_labs(master) cory$ bin/rmagicksh
>> img = img("images/barcelona-casa.jpg")
>> pm = PhotoMosaicer.new(img,100,100)
>> pm.create_photo_mosaic!
```

```
  def intensity_for_image_portion(x_offset, y_offset, columns, rows)
    intensity = 0
    pixel_count = 0
    x_offset.upto(x_offset + columns - 1) do |column|
      y_offset.upto(y_offset + rows - 1) do |row|
        break if column >= input_image_pixels.size
        break if row    >= input_image_pixels.first.size

        intensity   += input_image_pixels[column][row].intensity
        pixel_count += 1
      end
    end
    intensity = intensity.to_f / (pixel_count)
  end
```

code is at http://github.com/bantic/image_labs /lib/mosaicer

code is at http://github.com/bantic/image_labs /lib/mosaicer

code is at http://github.com/bantic/image_labs /lib/mosaicer



code is at http://github.com/bantic/image_labs /lib/mosaicer

```
>> pm = PhotoMosaicer.new(img,100,100)
>> pm.source_images_dir = "images/barcelona-thumbnails"
>> pm.create_photo_mosaic!
```

```ruby
def calculate_source_image_intensities!
  @scaled_images = ImageList.new(*Dir.glob(tmp_dir + "/*"))
  @intensities = []
  @image_hash = {}

  @scaled_images.each do |img|
    img_intensity = img.intensity
    @intensities << img_intensity
    @image_hash[img_intensity] = img
  end

  @intensities.sort!
end
```

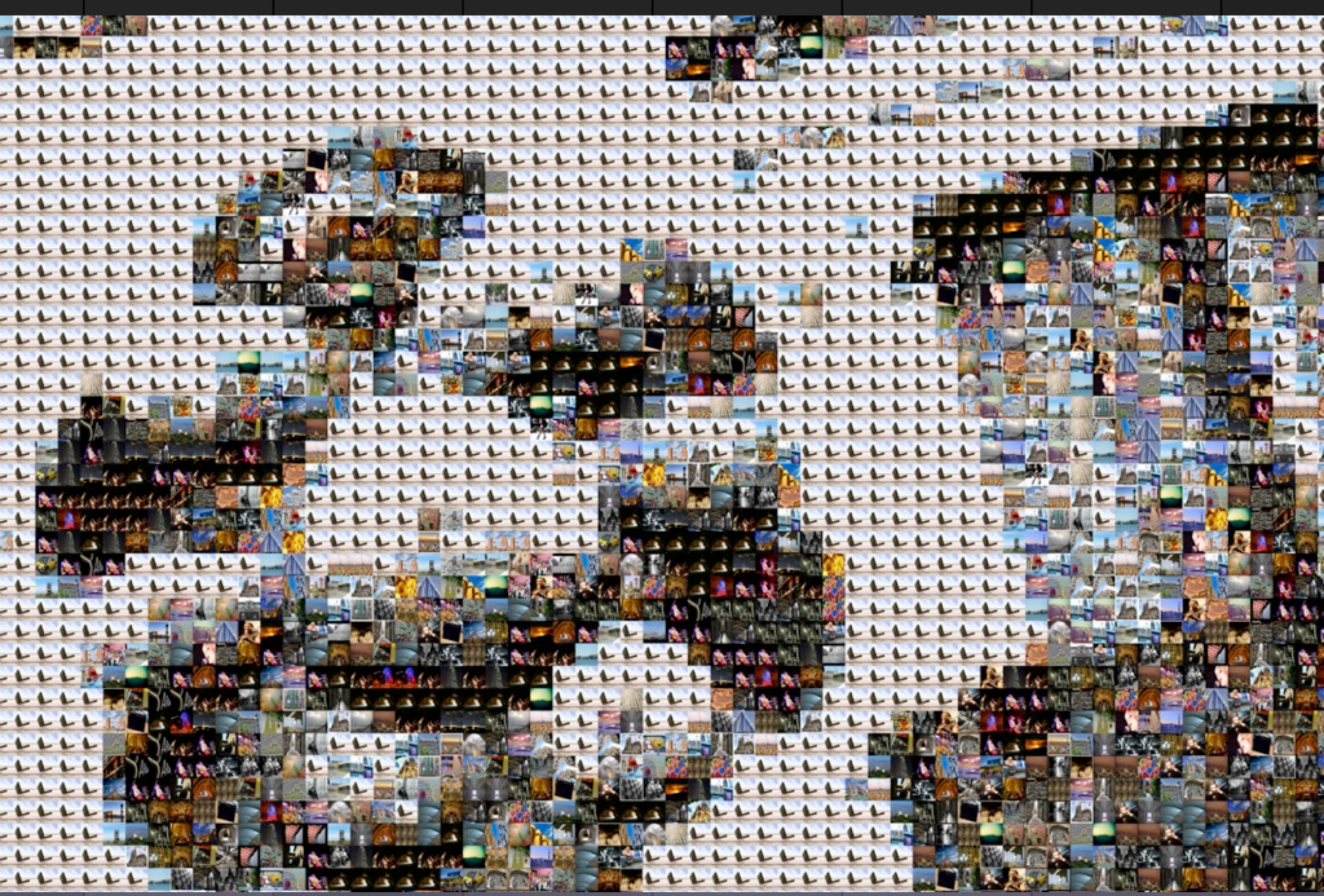
```ruby
# Find the closest image in our hash of source images
nearest_intensity = @intensities.detect {|i| intensity <= i} || @intensities.last
nearest_intensity_image = @image_hash[nearest_intensity]
```

code is at http://github.com/bantic/image_labs /lib/mosaicer

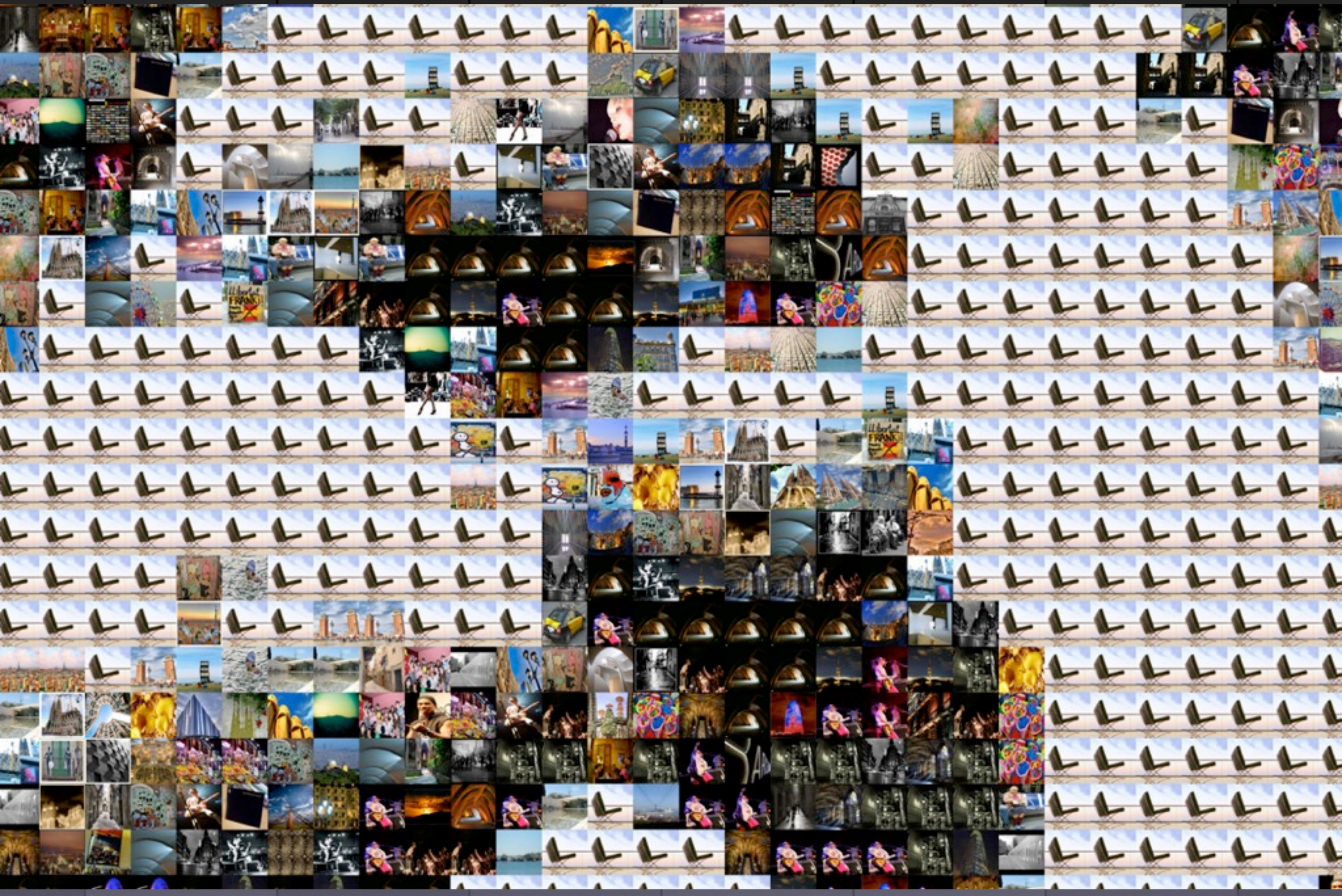
code is at http://github.com/bantic/image_labs /lib/mosaicer

code is at http://github.com/bantic/image_labs /lib/mosaicer

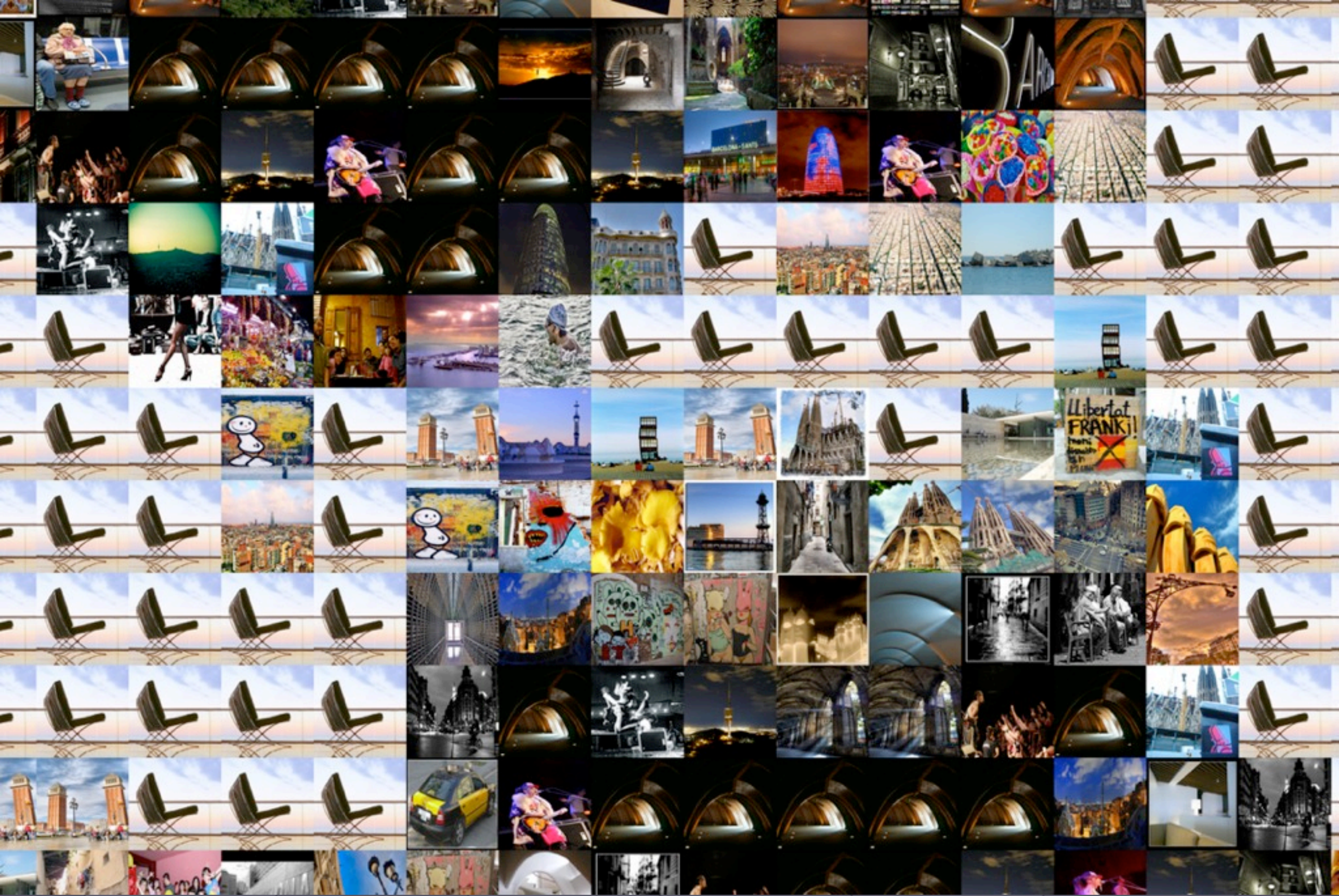
code is at http://github.com/bantic/image_labs /lib/mosaicer



code is at http://github.com/bantic/image_labs /lib/mosaicer

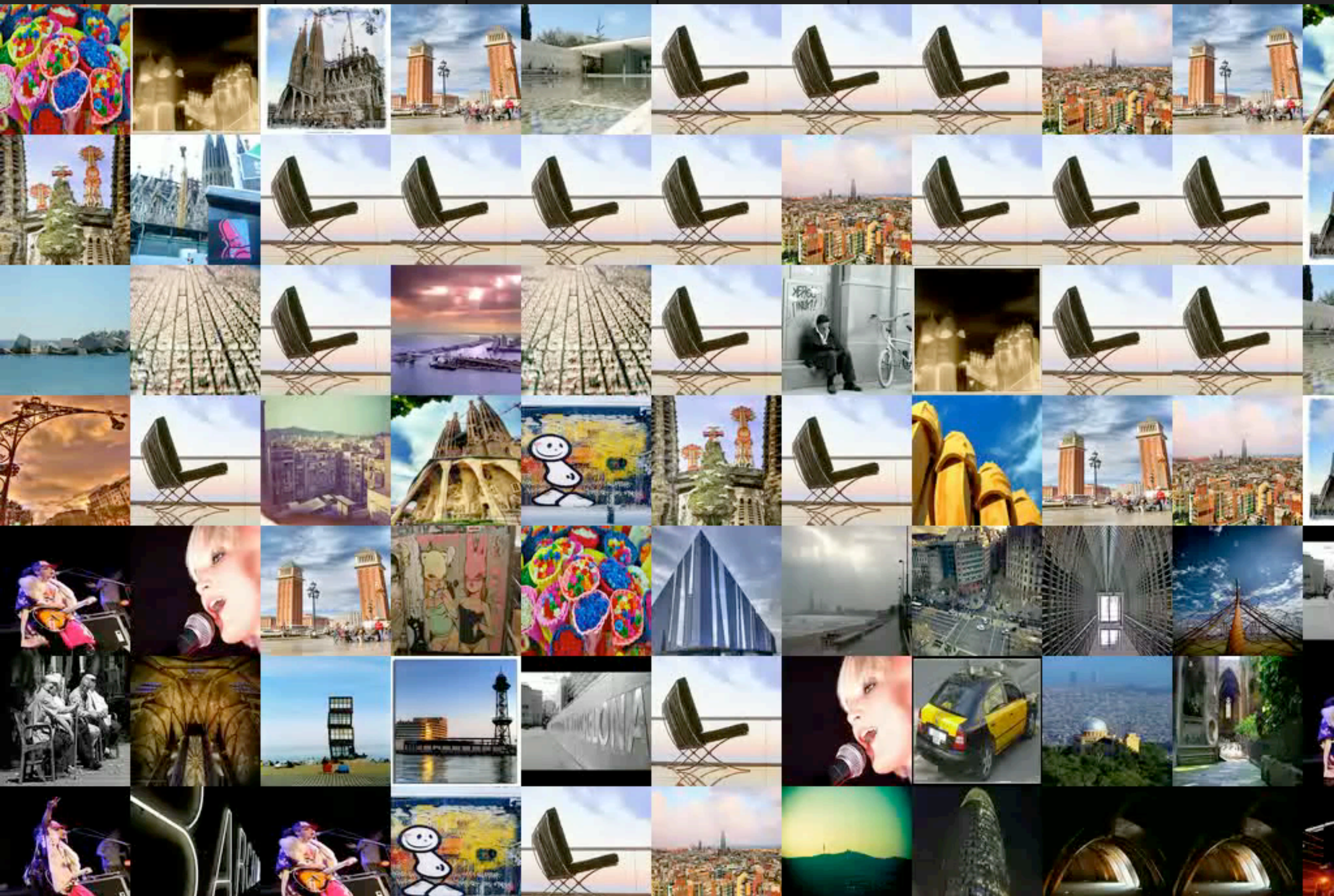


code is at http://github.com/bantic/image_labs /lib/mosaicer



code is at http://github.com/bantic/image_labs /lib/mosaicer

code is at http://github.com/bantic/image_labs /lib/mosaicer



code is at http://github.com/bantic/image_labs /lib/mosaicer

- RMagick
- PhotoMosaic
- SeamCarving
- Face Detection with OpenCV

code is at http://github.com/bantic/image_labs /lib/seamcarver

Tuesday, May 12, 2009

Seam Carving

- also called Content-Aware Image Resizing
- find edges
- create energy map
- find lowest energy path (“seam”)
- carve that seam
- repeat

code is at http://github.com/bantic/image_labs /lib/seamcarver

Tuesday, May 12, 2009

Edge Detection

- Find places where the pixels change suddenly
- Look in the X- and Y- directions
- Put those together = edges!

code is at http://github.com/bantic/image_labs /lib/seamcarver

Prewitt Operator

x-derivative

```
-1  0  1
-1  0  1
```

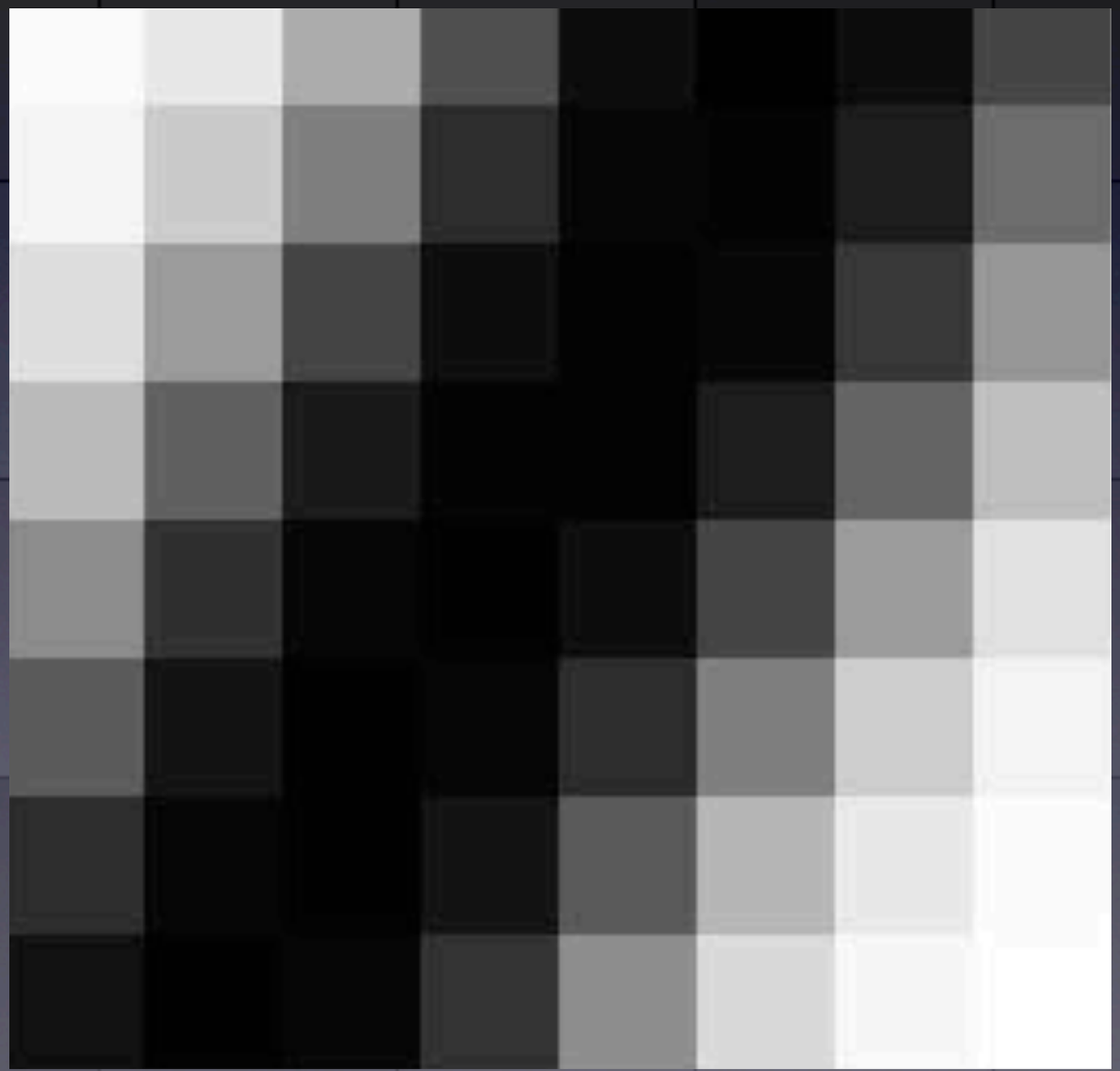
* pixel

y-derivative

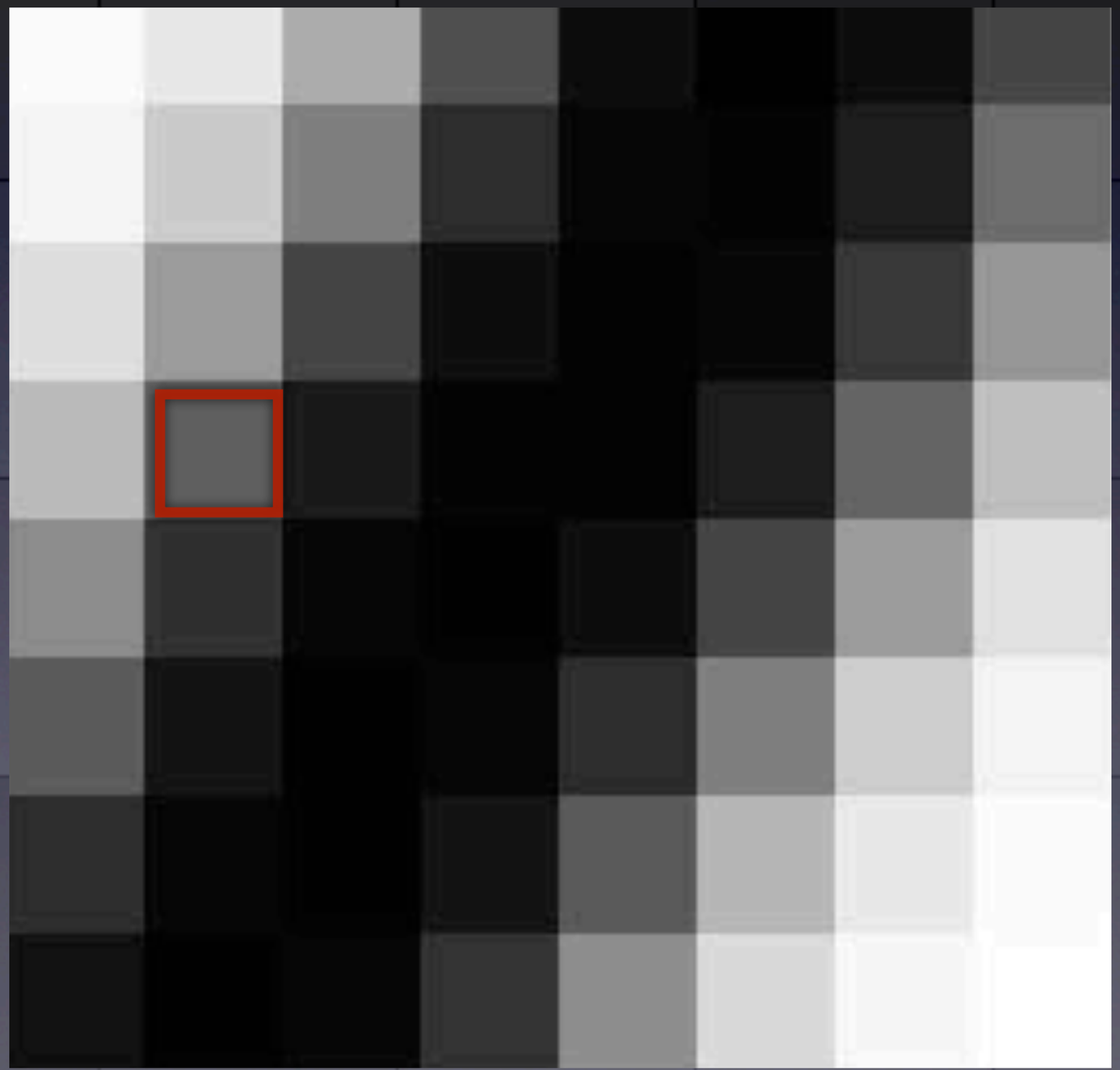
```
-1 -1 -1
 0  0  0
 1  1  1
```

* pixel

code is at http://github.com/bantic/image_labs /lib/seamcarver

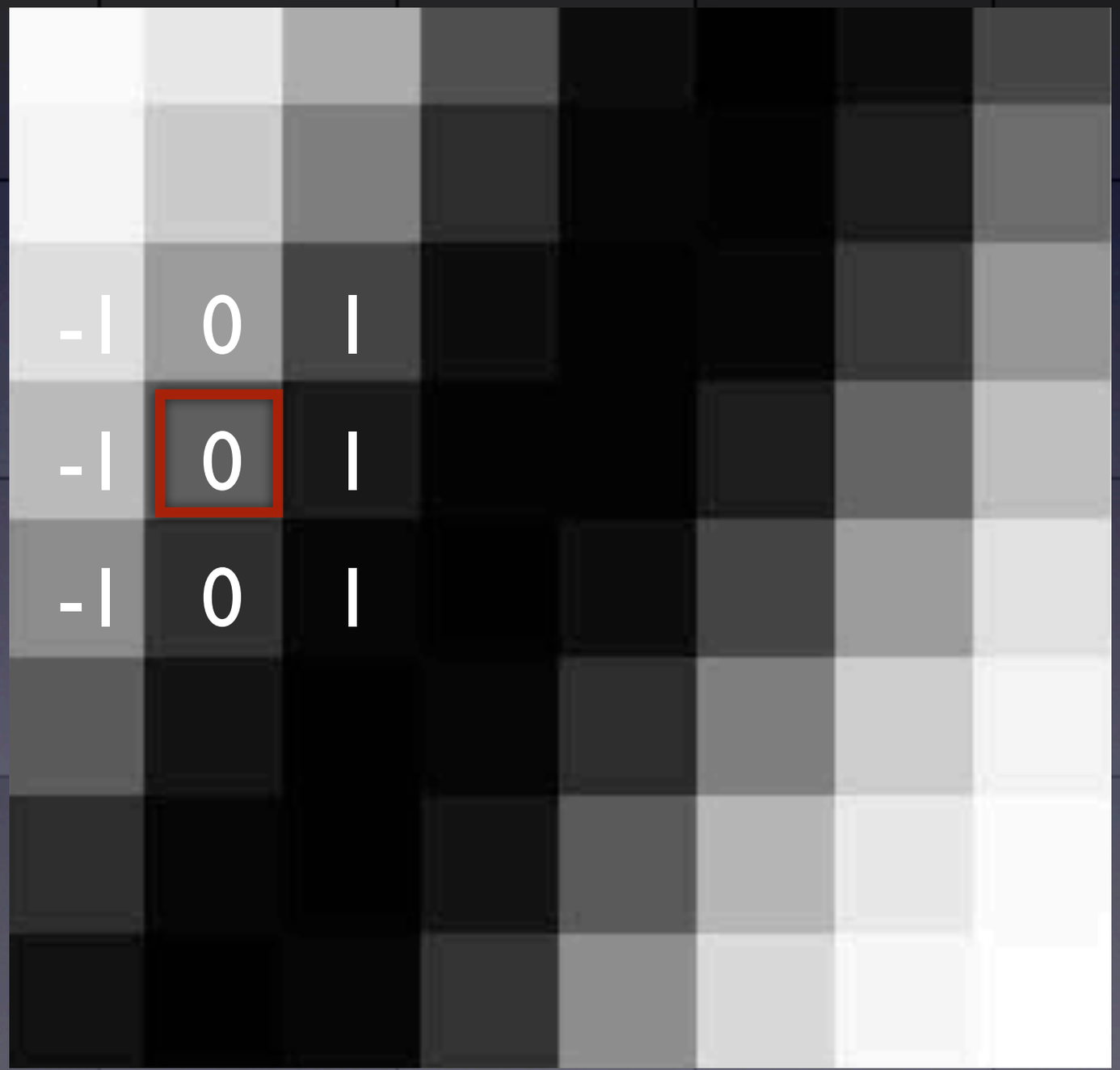


code is at http://github.com/bantic/image_labs /lib/seamcarver

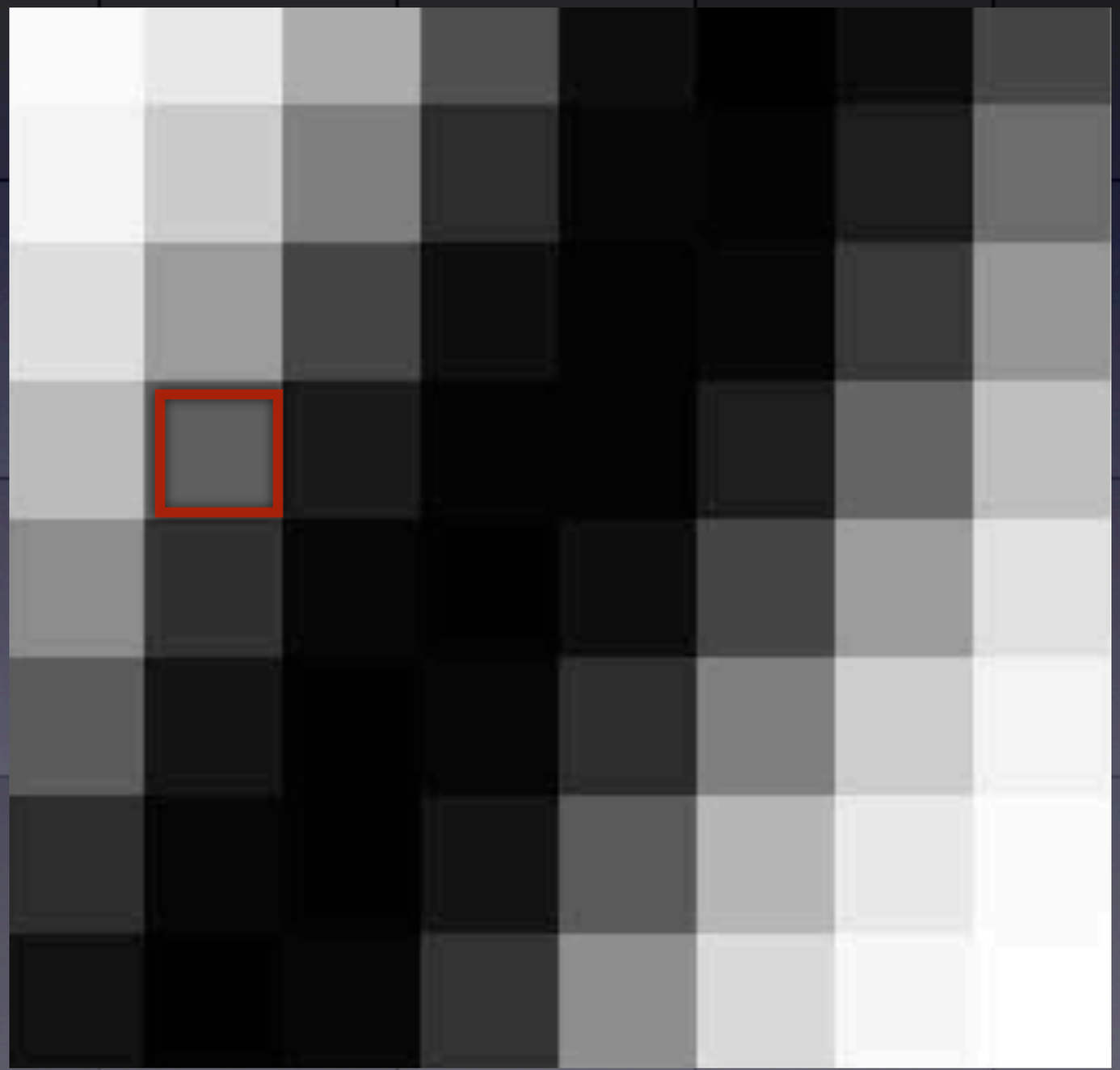


code is at http://github.com/bantic/image_labs /lib/seamcarver

x

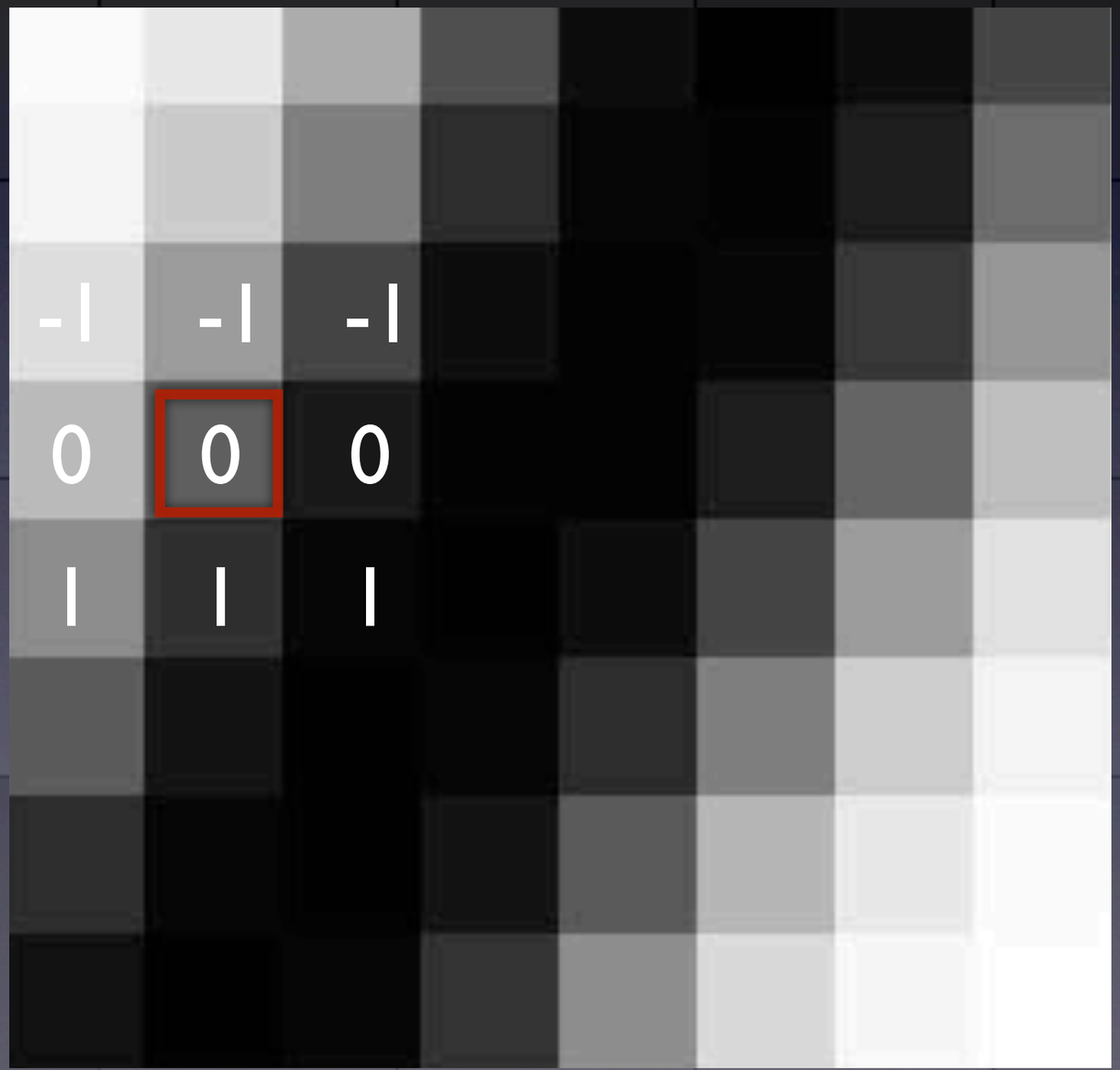


code is at http://github.com/bantic/image_labs /lib/seamcarver



code is at http://github.com/bantic/image_labs /lib/seamcarver

y

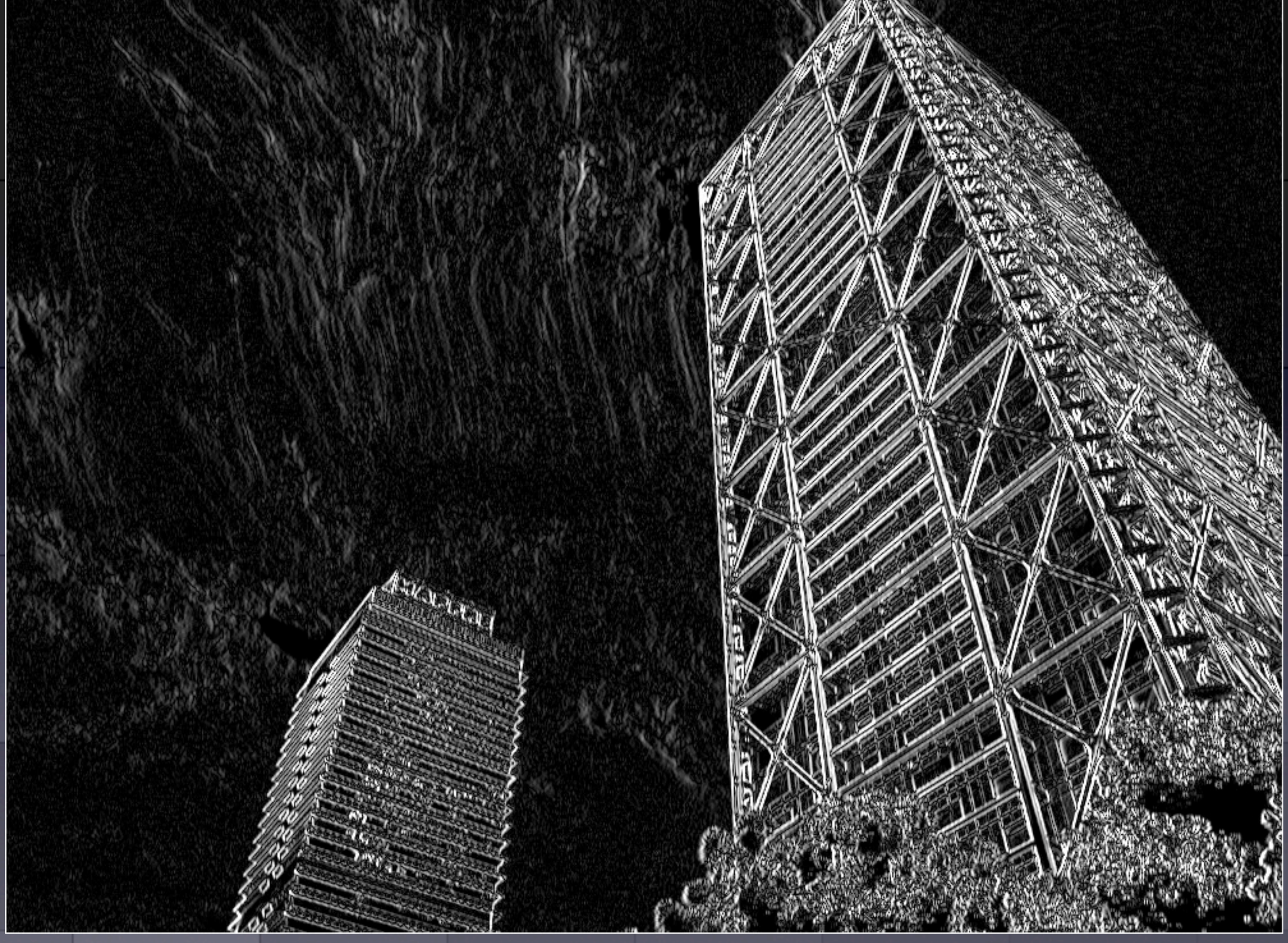


code is at http://github.com/bantic/image_labs /lib/seamcarver

edges

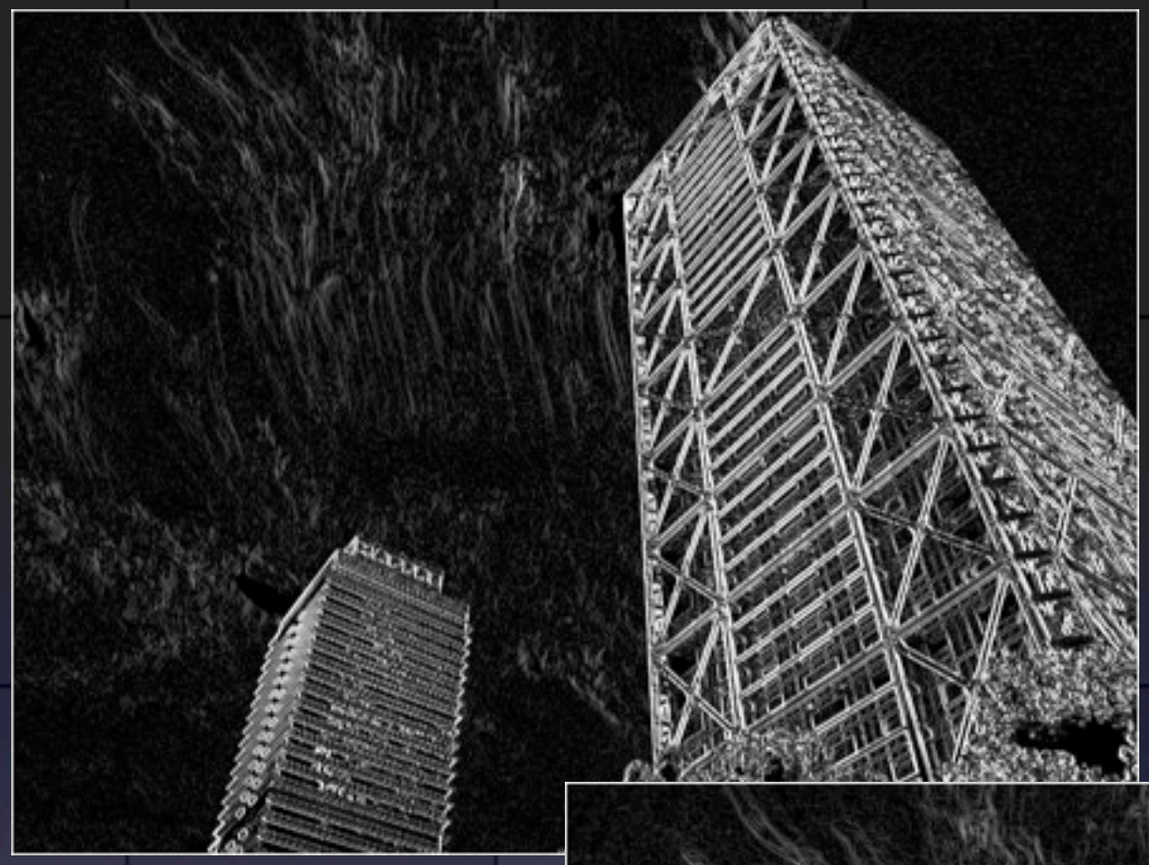
```
# for the 8 surrounding pixels to x,y, apply the filter
[-1,0,1].each do |x_offset|
  [-1,0,1].each do |y_offset|

    if orientation == :both || orientation == :x
      xsum += pixel_values[column + y_offset][row + x_offset] * x_mask(x_offset,y_offset)
    end
    if orientation == :both || orientation == :y
      ysum += pixel_values[column + y_offset][row + x_offset] * y_mask(x_offset,y_offset)
    end

  end
end

magnitude = Math.sqrt(xsum * xsum + ysum * ysum)
```

code is at http://github.com/bantic/image_labs /lib/seamcarver

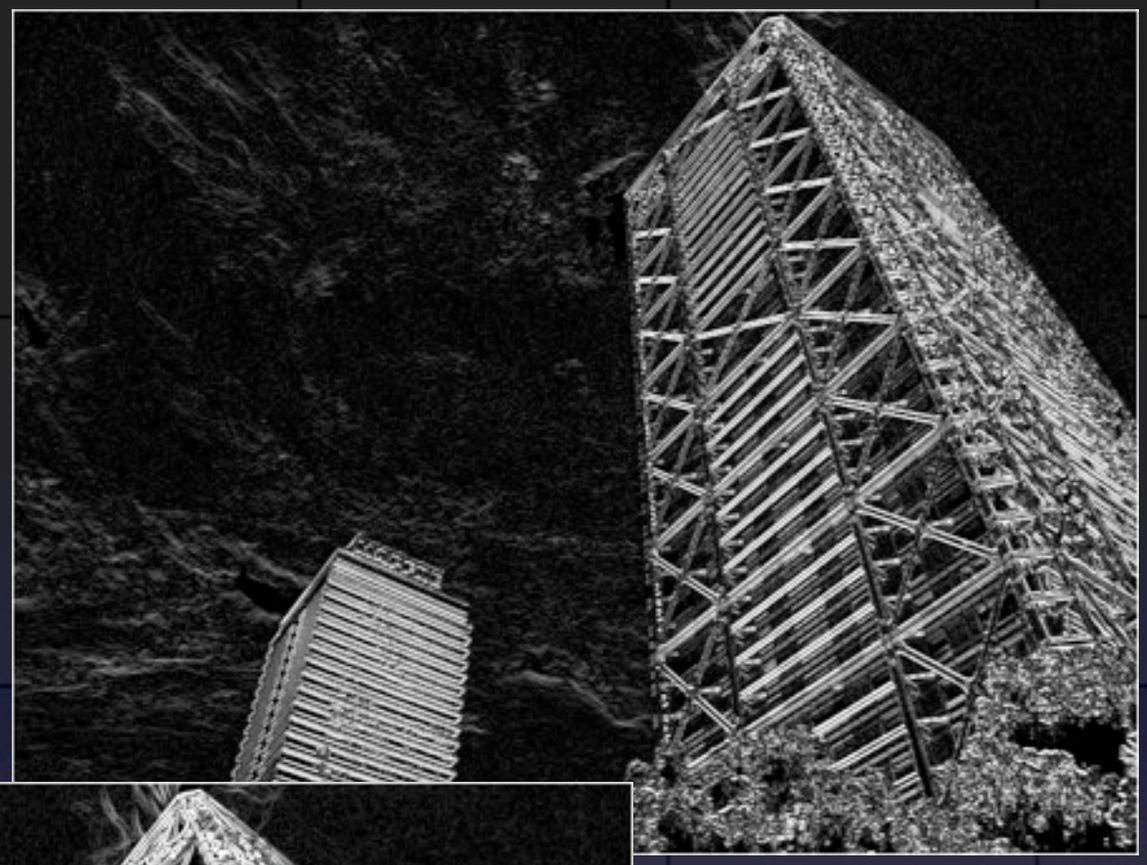


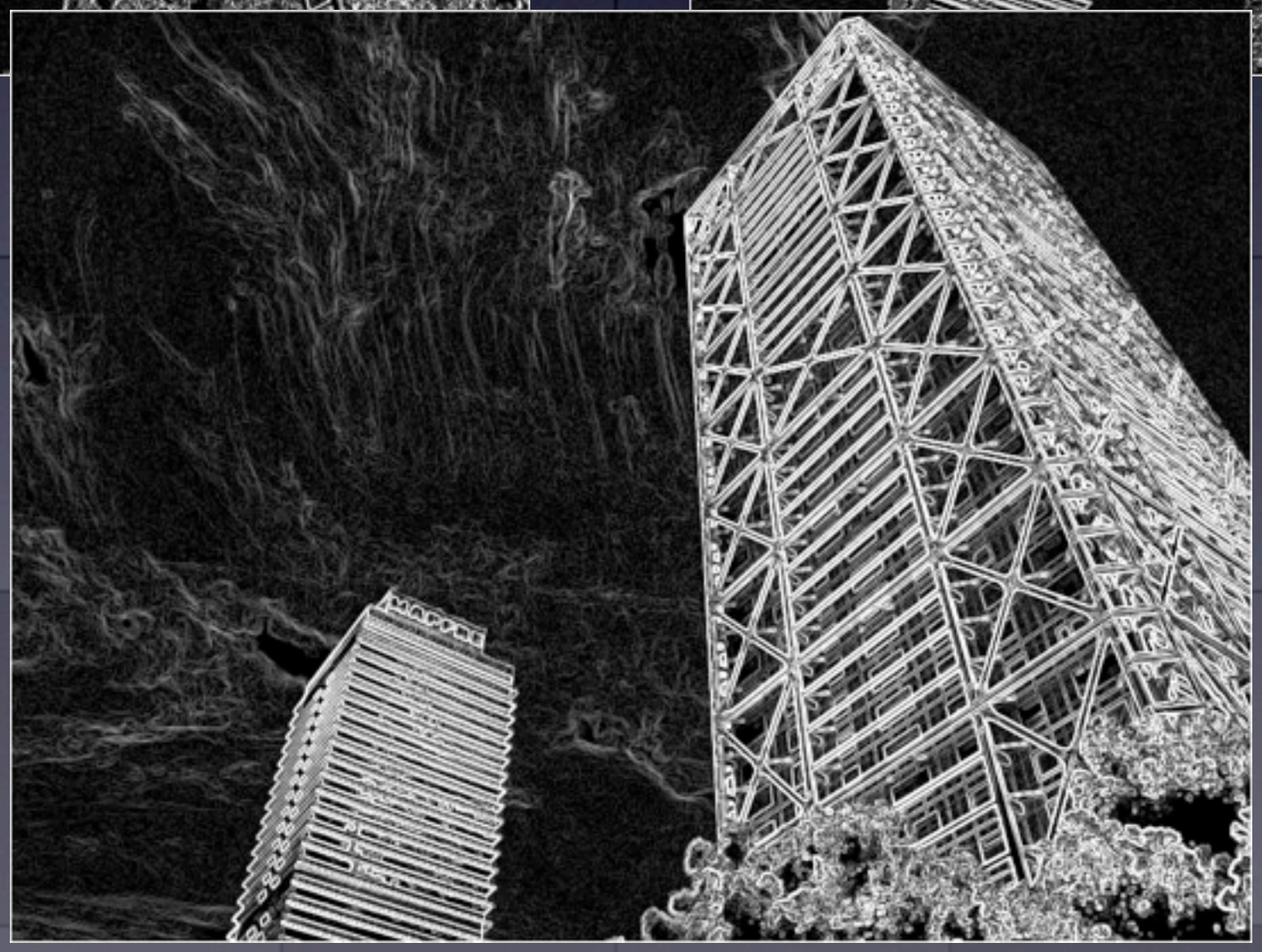
code is at http://github.com/bantic/image_labs /lib/seamcarver

code is at http://github.com/bantic/image_labs /lib/seamcarver



+



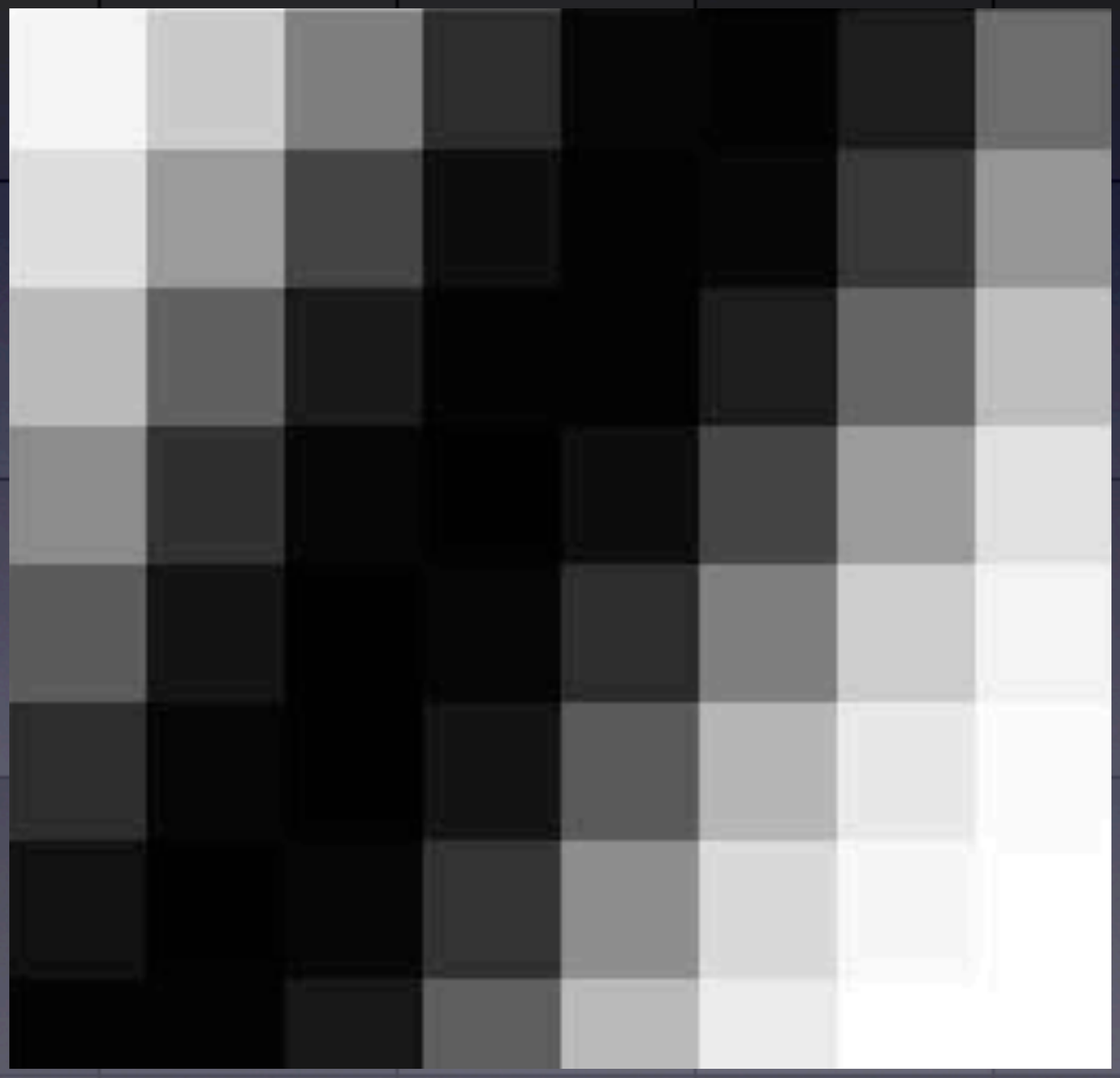


code is at http://github.com/bantic/image_labs /lib/seamcarver

energy map

- find the cheapest 8-connected path
- each pixel is assigned an energy value based on the cheapest energy path to that pixel
- dynamically calculate values pixel-by-pixel in a row
- repeat with the next row

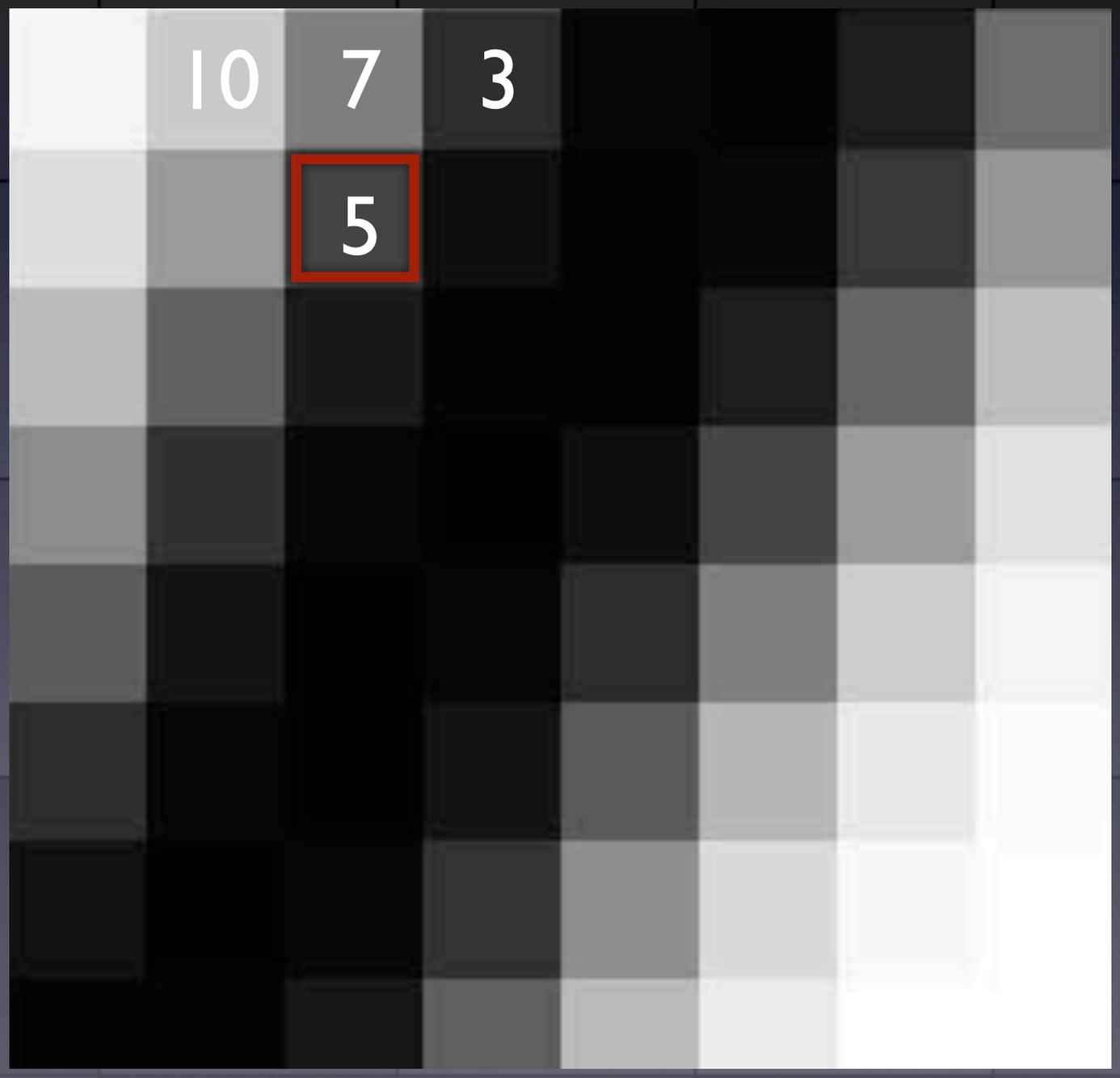
code is at http://github.com/bantic/image_labs /lib/seamcarver



code is at http://github.com/bantic/image_labs /lib/seamcarver

code is at http://github.com/bantic/image_labs /lib/seamcarver

code is at http://github.com/bantic/image_labs /lib/seamcarver



code is at http://github.com/bantic/image_labs /lib/seamcarver

code is at http://github.com/bantic/image_labs /lib/seamcarver

code is at http://github.com/bantic/image_labs /lib/seamcarver

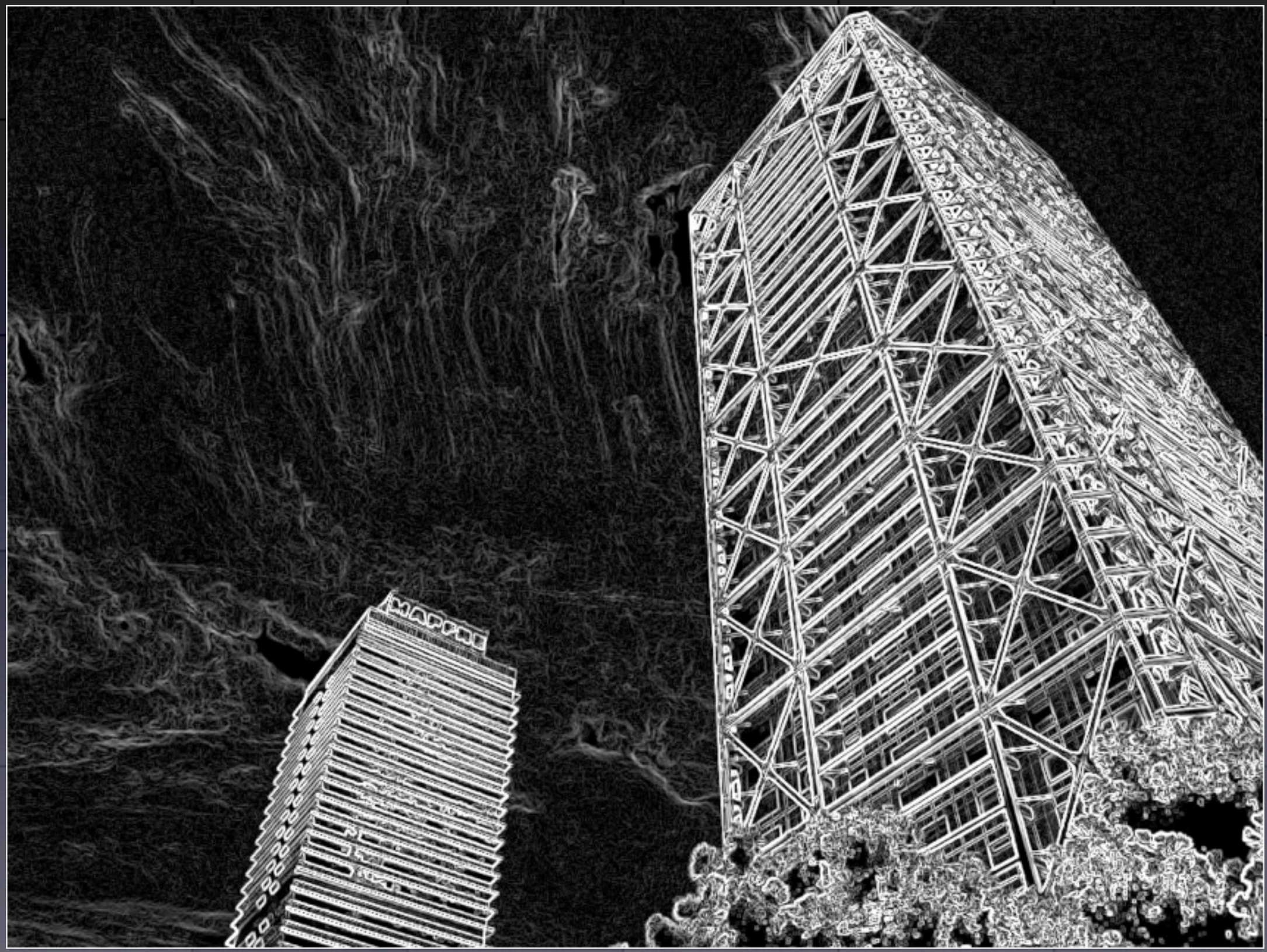
code is at http://github.com/bantic/image_labs /lib/seamcarver

calculate energy map

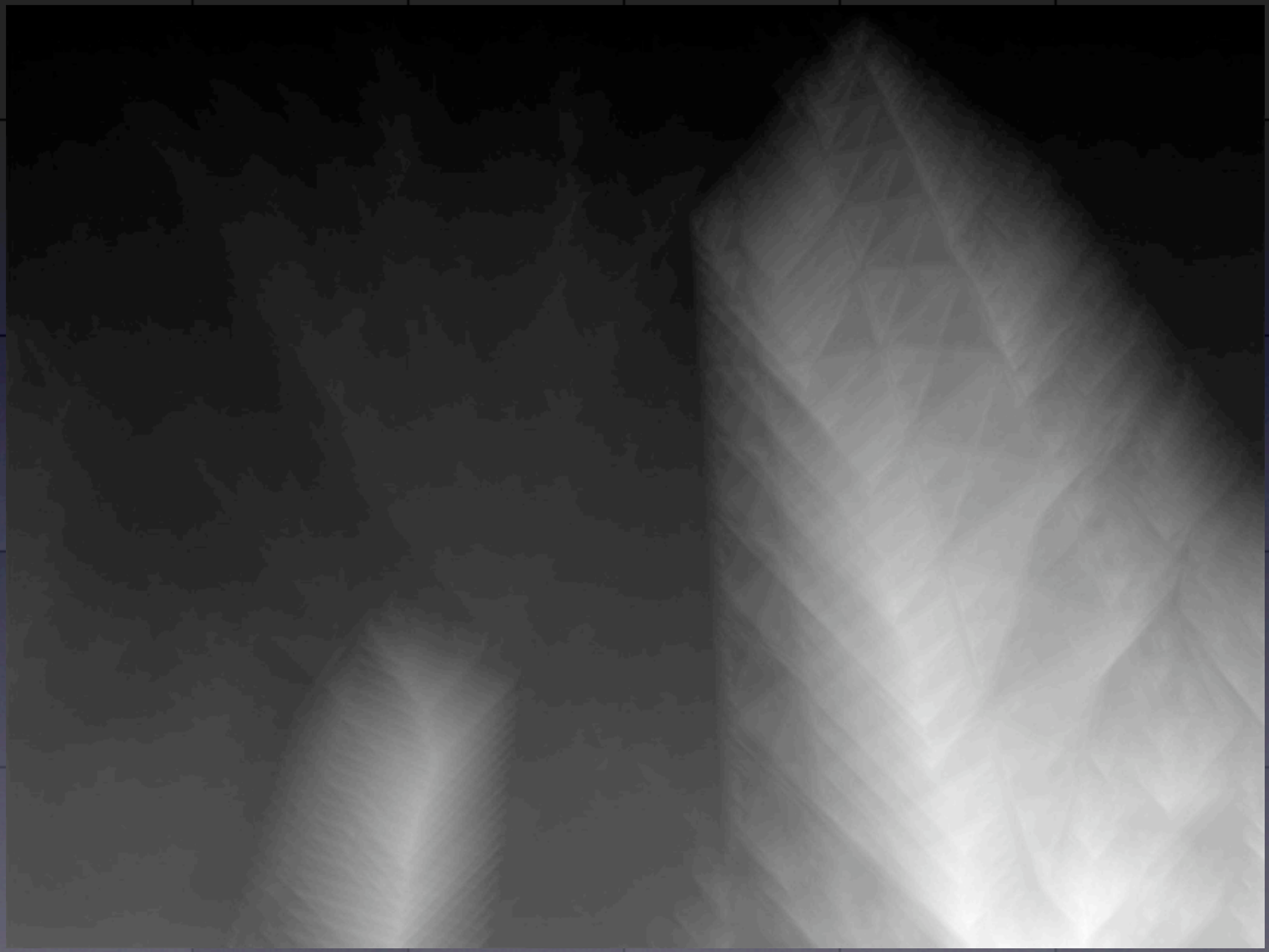
```
base_energy = energy_map[column][row]

if row == 0                    # first row. energy is just edge value.
  energy = base_energy
elsif column == 0 || column == width
  if column == 0  # left edge, only two pixels below
    energy = base_energy +  [energy_map[column    ][row - 1],
                             energy_map[column + 1][row - 1]].min
  else        # right edge, only two pixels below
    energy = base_energy +  [energy_map[column - 1][row - 1],
                             energy_map[column    ][row - 1]].min
  end
else           # not an edge, check all three pixels below
  energy   = base_energy +  [energy_map[column - 1][row - 1],
                             energy_map[column    ][row - 1],
                             energy_map[column + 1][row - 1]].min
end

energy_map[column][row] = energy
```

code is at http://github.com/bantic/image_labs /lib/seamcarver

code is at http://github.com/bantic/image_labs /lib/seamcarver



code is at http://github.com/bantic/image_labs /lib/seamcarver



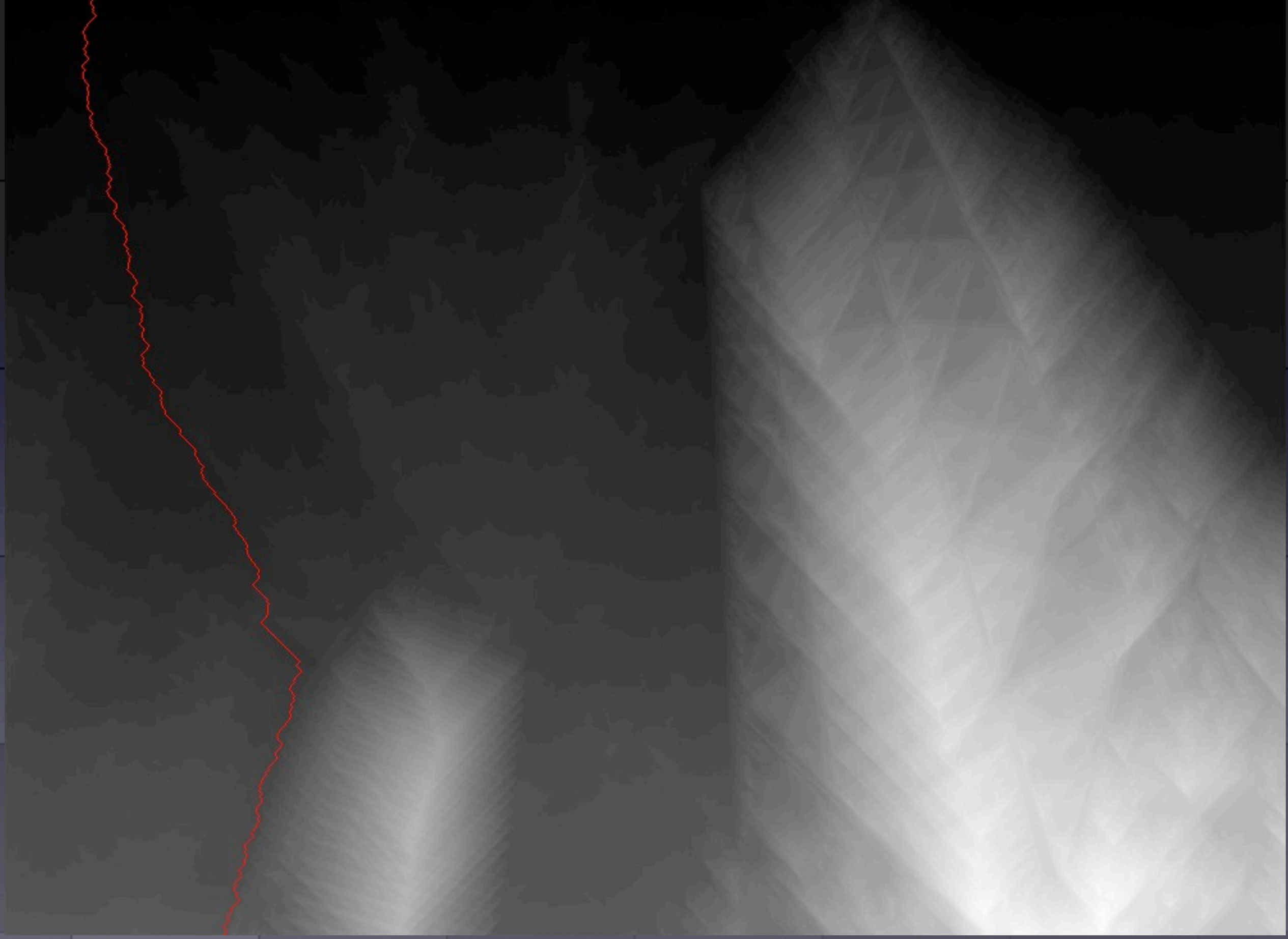
code is at http://github.com/bantic/image_labs /lib/seamcarver

find seam

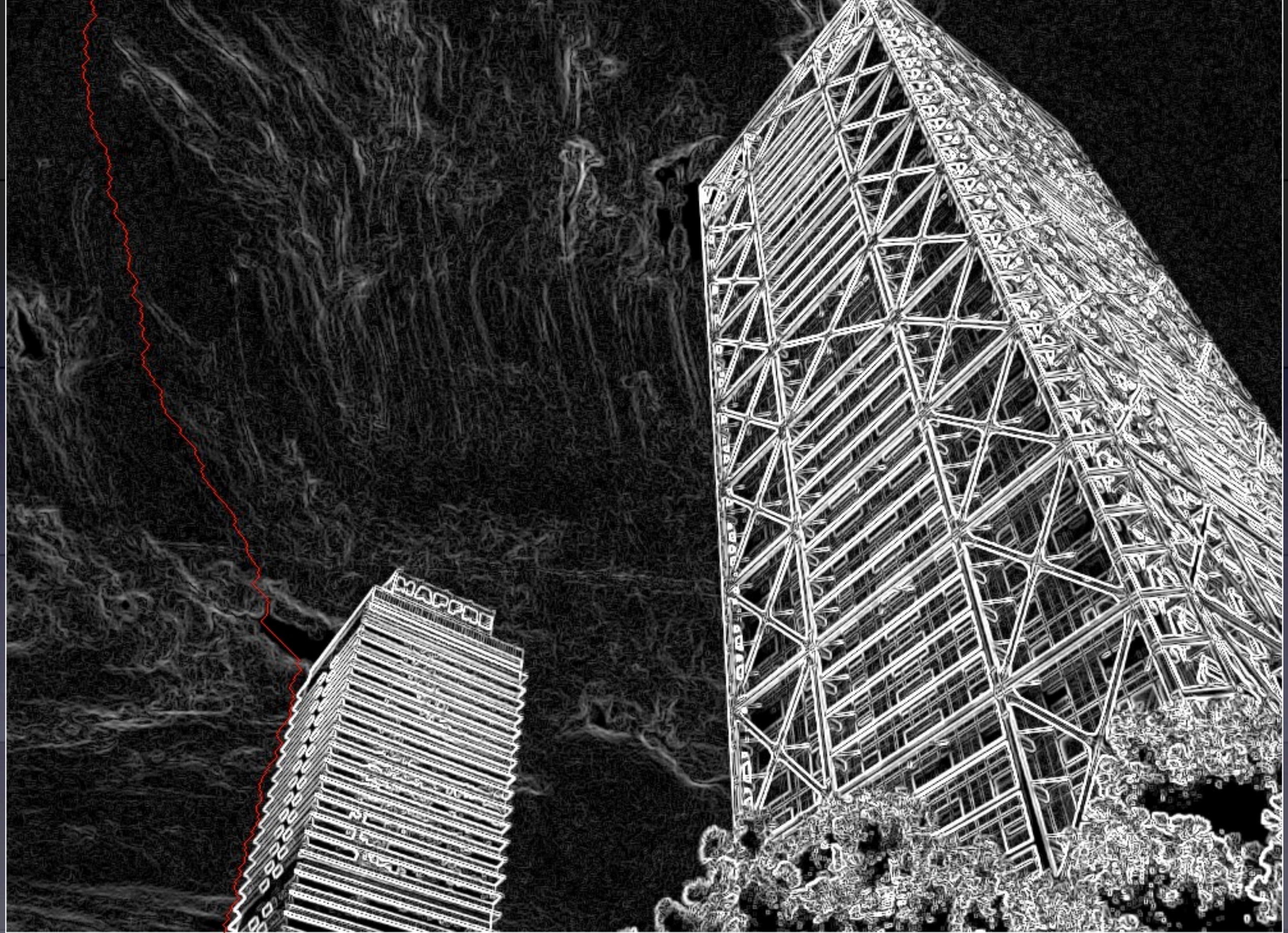
```
if min_pos == 0 || min_pos == width # at an edge, only search two spaces
  if min_pos == 0
    min_val = energy_map[row_num][min_pos, 2].min
    min_pos = energy_map[row_num][min_pos, 2].index(min_val) + min_pos
  else # min_pos == width
    min_val = energy_map[row_num][min_pos - 1, 2].min
    min_pos = energy_map[row_num][min_pos - 1, 2].index(min_val) - 1 + min_pos
  end
else                                  # find the minimum from the three spaces above this
  min_val = energy_map[row_num][min_pos - 1, 3].min
  min_pos = energy_map[row_num][min_pos - 1, 3].index(min_val) - 1 + min_pos
end
```

```
$ bin/rmagicksh
>> sc = SeamCarver.new("images/barcelona-small.jpg")
>> seam = sc.find_seam!
>> barcelona = img("images/barcelona-small.jpg")
```

code is at http://github.com/bantic/image_labs /lib/seamcarver



code is at http://github.com/bantic/image_labs /lib/seamcarver



code is at http://github.com/bantic/image_labs /lib/seamcarver

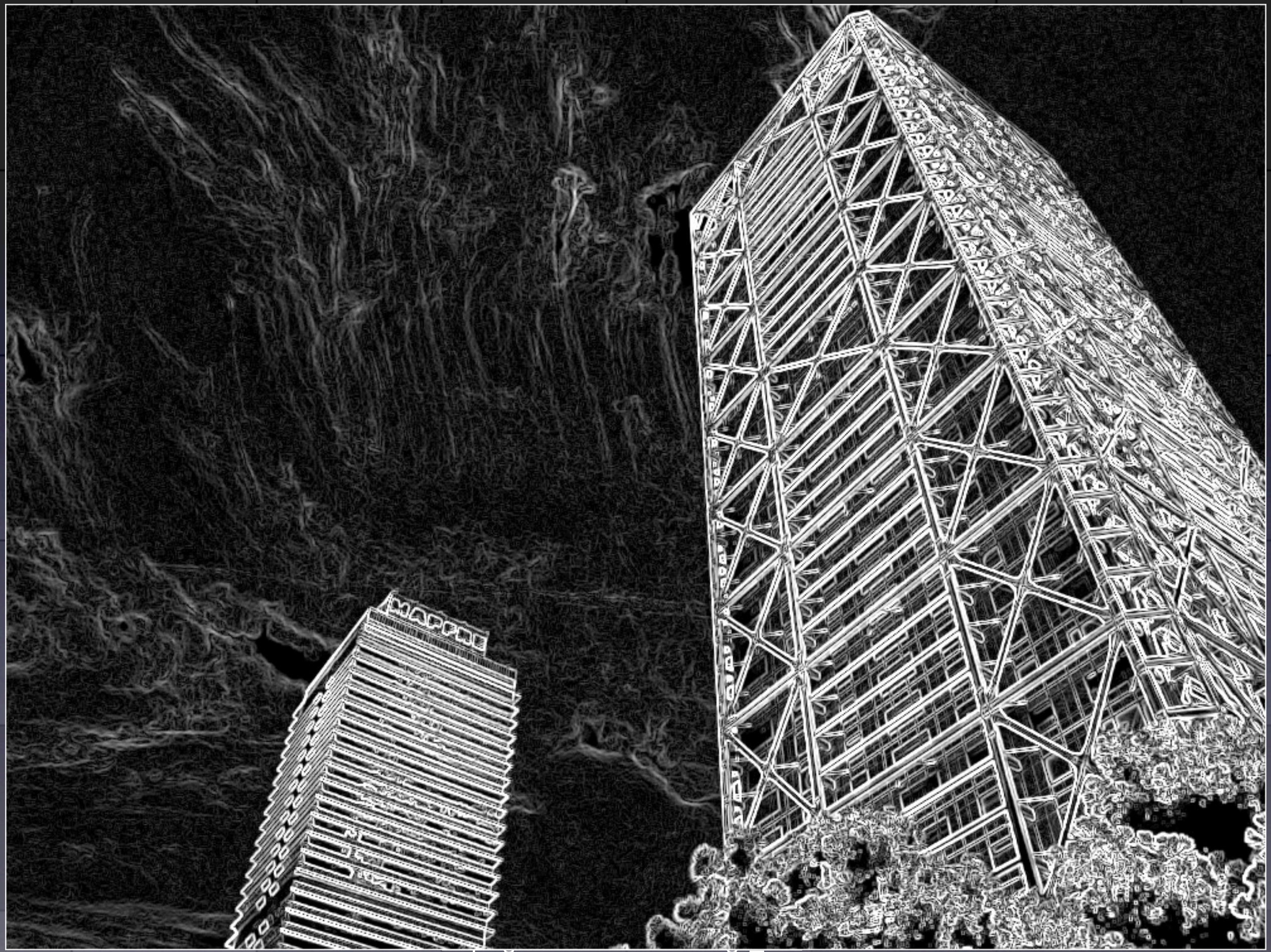
code is at http://github.com/bantic/image_labs /lib/seamcarver

CARVED

RESIZED

code is at http://github.com/bantic/image_labs /lib/seamcarver

code is at http://github.com/bantic/image_labs /lib/seamcarver



Tuesday, May 12, 2009

Tuesday, May 12, 2009

Tuesday, May 12, 2009

Tuesday, May 12, 2009

- RMagick
- PhotoMosaic
- SeamCarving
- Face Detection with OpenCV

OpenCV

- developed by Intel
- awesome
  - object detection (faces)
    - Hough transform to find lines and circles
- con: difficult to install
- Install instructions and gem:
  - http://github.com/bantic/ruby-opencv

code is at http://github.com/bantic/ruby-opencv and http://github.com/bantic/image_labs => /lib/facer

Face Detection

- "Haar Cascade"
- Simple light-dark features ("Haarlike")
- Binary Classifiers working together ("cascade")

code is at http://github.com/bantic/ruby-opencv and http://github.com/bantic/image_labs => /lib/facer

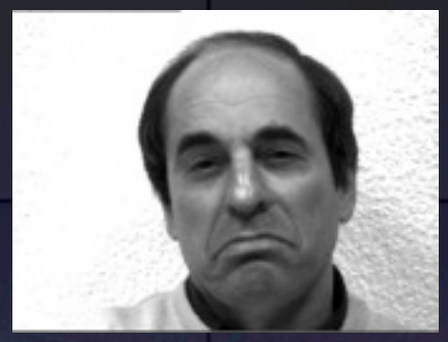
Face Detection

- "Haar Cascade"
- Simple light-dark features ("Haarlike")
- Binary Classifiers working together ("cascade")

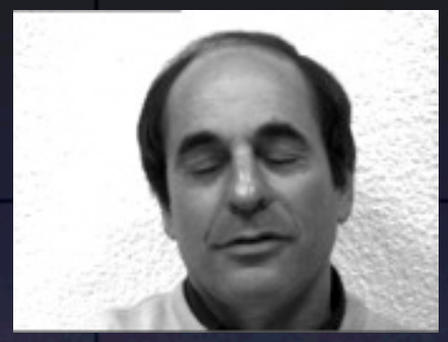
code is at http://github.com/bantic/ruby-opencv and http://github.com/bantic/image_labs => /lib/facer

Face Detection

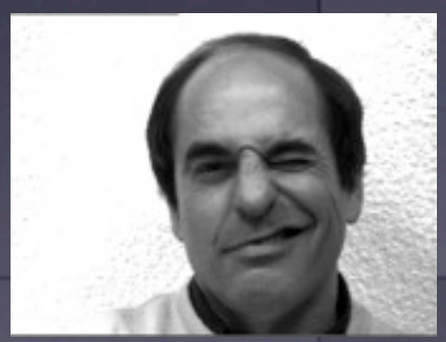
glasses



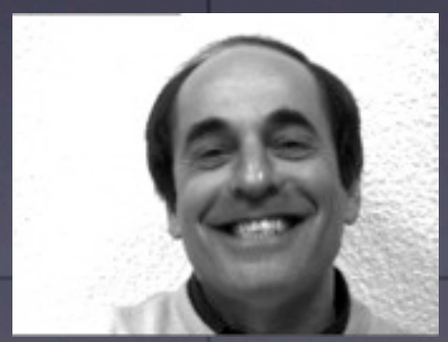
sad



sleepy



wink

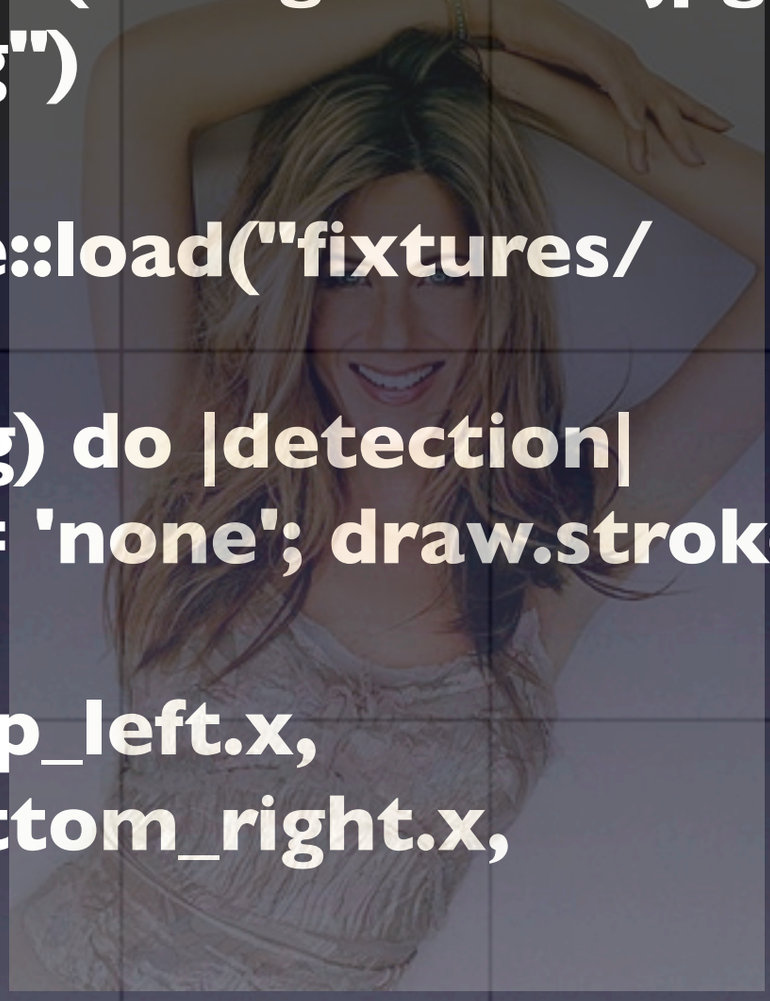


happy

surprised

code is at http://github.com/bantic/ruby-opencv and http://github.com/bantic/image_labs => /lib/facer

```
$ bin/rmagicksh
>> require 'opencv'
>> cv_img = OpenCV::IplImage.load("images/face.jpg")
>> rm_img = img("images/face.jpg")
>> detector =
OpenCV::CvHaarClassifierCascade::load("fixtures/
haarcascade_frontalface_alt.xml")
>> detector.detect_objects(cv_img) do |detection|
?>      draw = Draw.new; draw.fill = 'none'; draw.stroke
= 'red'
>>     draw.rectangle( detection.top_left.x,
detection.top_left.y, detection.bottom_right.x,
detection.bottom_right.y)
>>     draw.draw(rm_img)
>> end
>> rm_img.write("detected_face.png")
```



code is at http://github.com/bantic/ruby-opencv and http://github.com/bantic/image_labs => /lib/facer

```
$ bin/rmagicksh
>> require 'opencv'
>> cv_img = OpenCV::IplImage.load("images/face.jpg")
>> rm_img = img("images/face.jpg
>> detector =
OpenCV::CvHaarClassifierCascade
haarcascade_frontalface_alt.xml")
>> detector.detect_objects(cv_img
?>      draw = Draw.new; draw.fill =
= 'red'
>>     draw.rectangle( detection.top
detection.top_left.y, detection.bot
detection.bottom_right.y)
>>     draw.draw(rm_img)
>> end
>> rm_img.write("detected_face.png")
```

code is at http://github.com/bantic/ruby-opencv and http://github.com/bantic/image_labs => /lib/facer

>> Facer.put_funny_hat_on_celebrity

```
def self.put_funny_hat_on_celebrity(celebrity_img)
  rm_img = celebrity_img
  cv_img = OpenCV::IplImage.load(r_img.filename)

  detector = self.face_detector
  face_rectangles = detector.detect_objects(cv_img)

  face_rectangles.each do |rect|
    rect_w = rect.top_right.x - rect.top_left.x
    rect_h = rect.bottom_left.y  - rect.top_left.y

    hat = self.funny_hat.resize(rect_w.to_f * 0.8, rect_h.to_f * 0.8)

    x_offset = rect.top_left.x - (hat.columns.to_f * 0.1)
    y_offset = rect.top_left.y - (hat.rows.to_f * 0.95)
    rm_img.composite!(hat, x_offset, y_offset, Magick::OverCompositeOp)
  end
  rm_img
end
```

code is at http://github.com/bantic/ruby-opencv and http://github.com/bantic/image_labs => /lib/facer

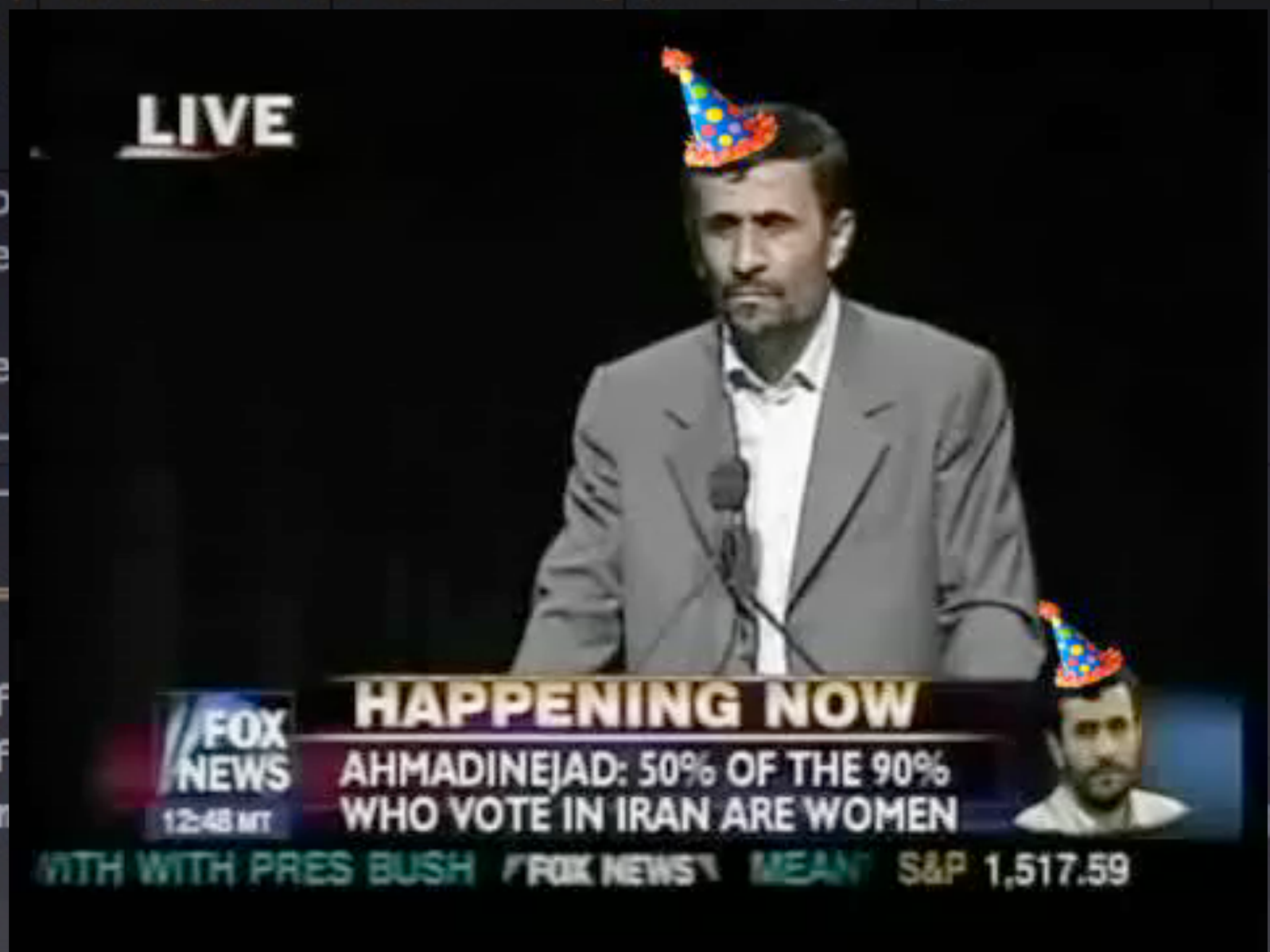
>> Facer.put_funny_hat_on_celebrity

```
def self.put_funny_hat_on_celebrity(celebrity_img)
  rm_img = celebrity_img
  cv_img = OpenCV::IplImage.load(r_img.filename)

  detector = self.face_detector
  face_rectangles = detector.detect_objects(cv_img)

  face_rectangles.each do |rect|
    rect_w = rect.top_right.x - rect.top_left.x
    rect_h = rect.bottom_left.y  - rect.top_left.y

    hat = self.funny_hat.resize(rect_w.to_f * 0.8, rect_h.to_f * 0.8)

    x_offset = rect.top_left.x - (hat.columns.to_f * 0.1)
    y_offset = rect.top_left.y - (hat.rows.to_f * 0.95)
    rm_img.composite!(hat, x_offset, y_offset, Magick::OverCompositeOp)
  end
  rm_img
end
```

code is at http://github.com/bantic/ruby-opencv and http://github.com/bantic/image_labs => /lib/facer

>> Facer.put_funny_hat_on_celebrity

```
def self.put_funny_hat_on_celebrity(celebrity_img)
  rm_img
  cv_img

  detecto
  face_re

  face_re
    rect_
    rect_

    hat =                                          * 0.8)

    x_off
    y_off
    rm_im                                          positeOp)
  end
  rm_img
end
```



code is at http://github.com/bantic/ruby-opencv and http://github.com/bantic/image_labs => /lib/facer

Thank you

- http://github.com/bantic/image_labs
- http://github.com/bantic/ruby-opencv
- http://delicious.com/eoligarry/image_labs