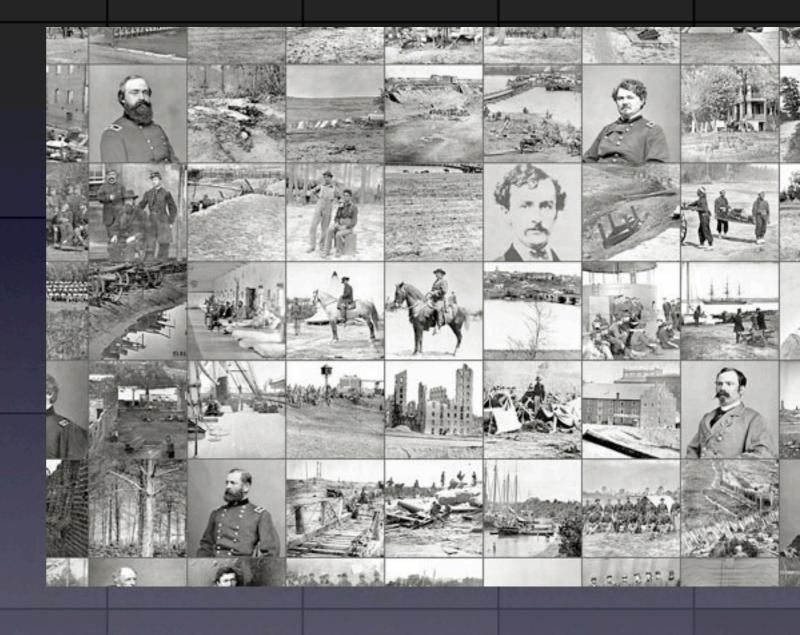
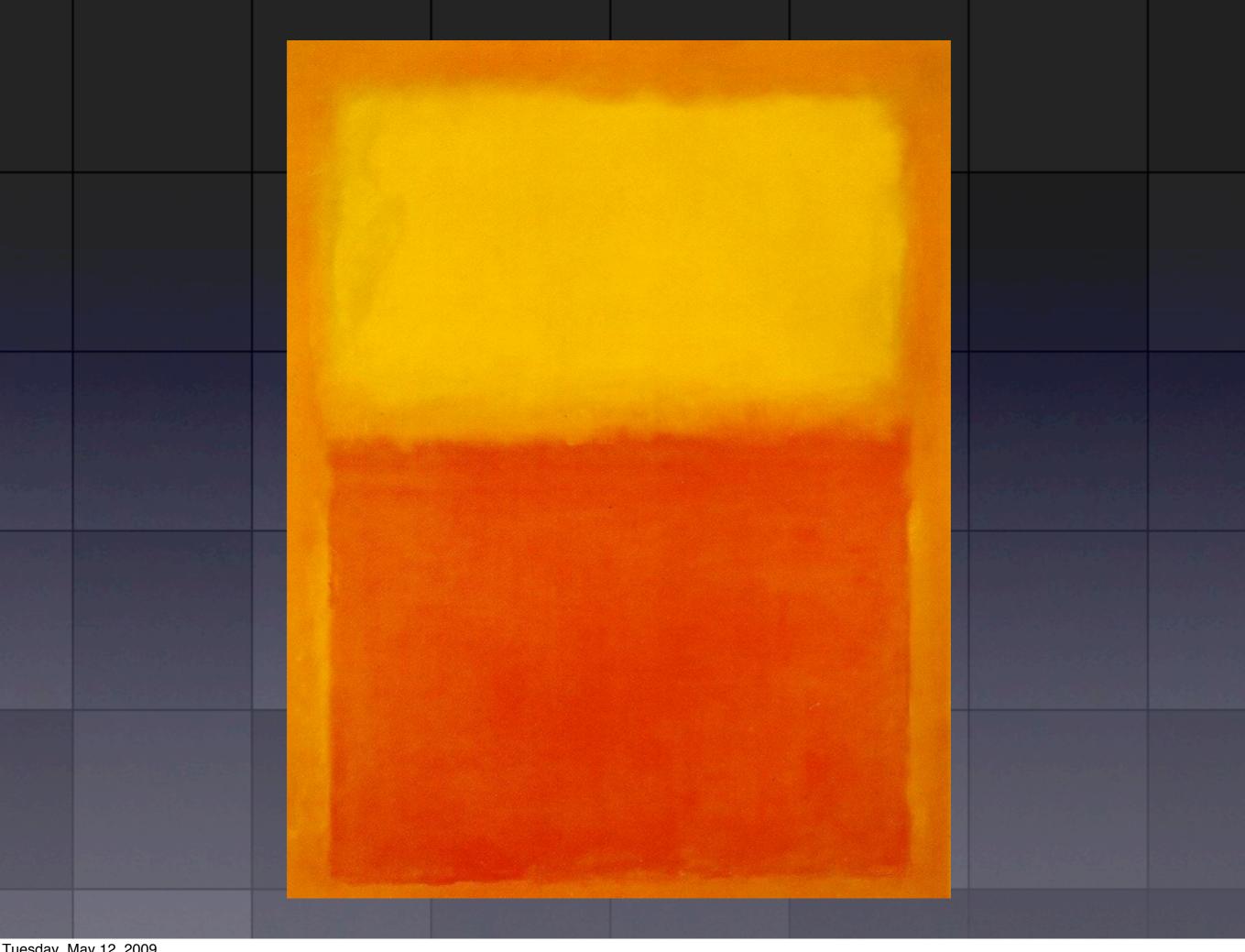
#### Who Needs Photoshop? Creative image manipulation and experimentation with Ruby

Palater a section of the section A state of the second stat A Shall a state to the state of Children and the second second second The second part of the second se Card Talat the design of the second se 100 mg martin Manager Martin & Strenger Ma Arrows 11 A - HATE ATTE AL 26 in the state of the state of the Logar Lund Bagar Br and the second sec Ante Man - A warm -100 A State of the state of the 

355 一致4一块 一般的 一 an some ANER r Hig the Palatestic Contraction and Storage A State Aught and the second THE R. H. W. C. - network P di k Construction of Toninker BAR ... And Barry and した 口 2423 465 第二十四日 第二十四日 上部 百十五 - GATE ATT ANTRIAL State of the local state of the - Richard State Barriers and Street and a lot new og & Street Married & William Street 27 20 W 40 P 243-03 





## "I could do that!"

"
 could do that!"

e c. : 222 4

CLOCK CREAK

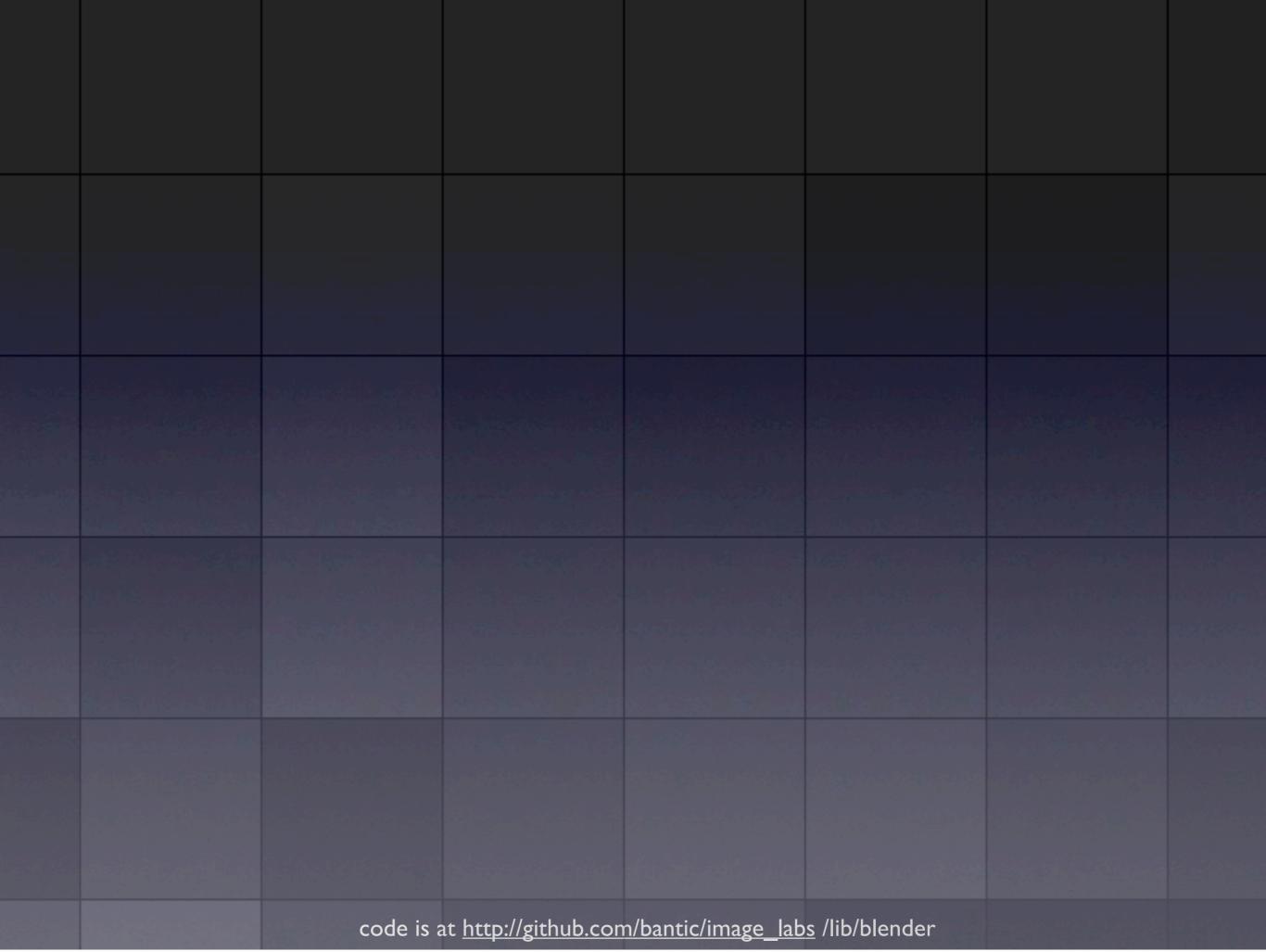
# " could do that!"

10-13

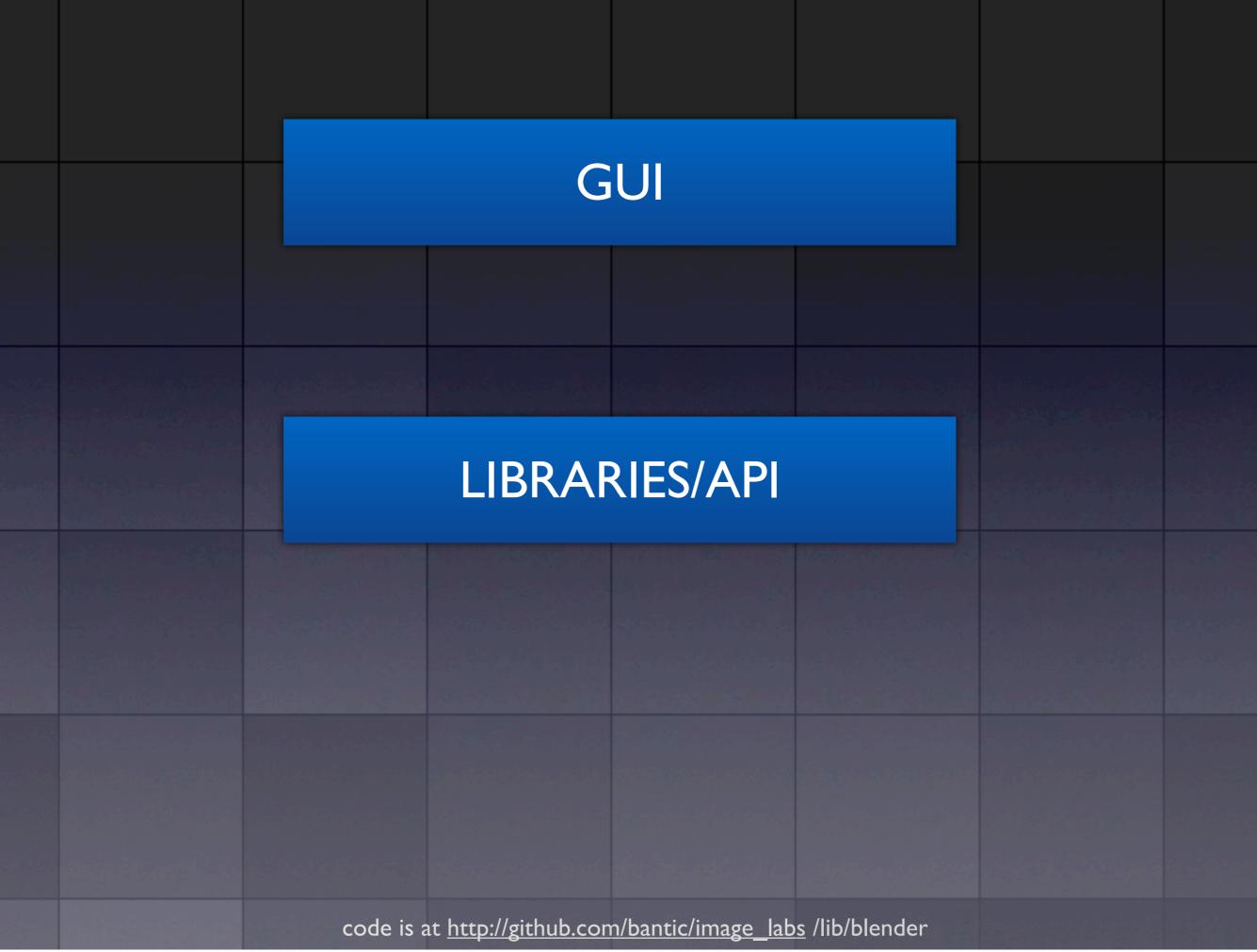
B4934

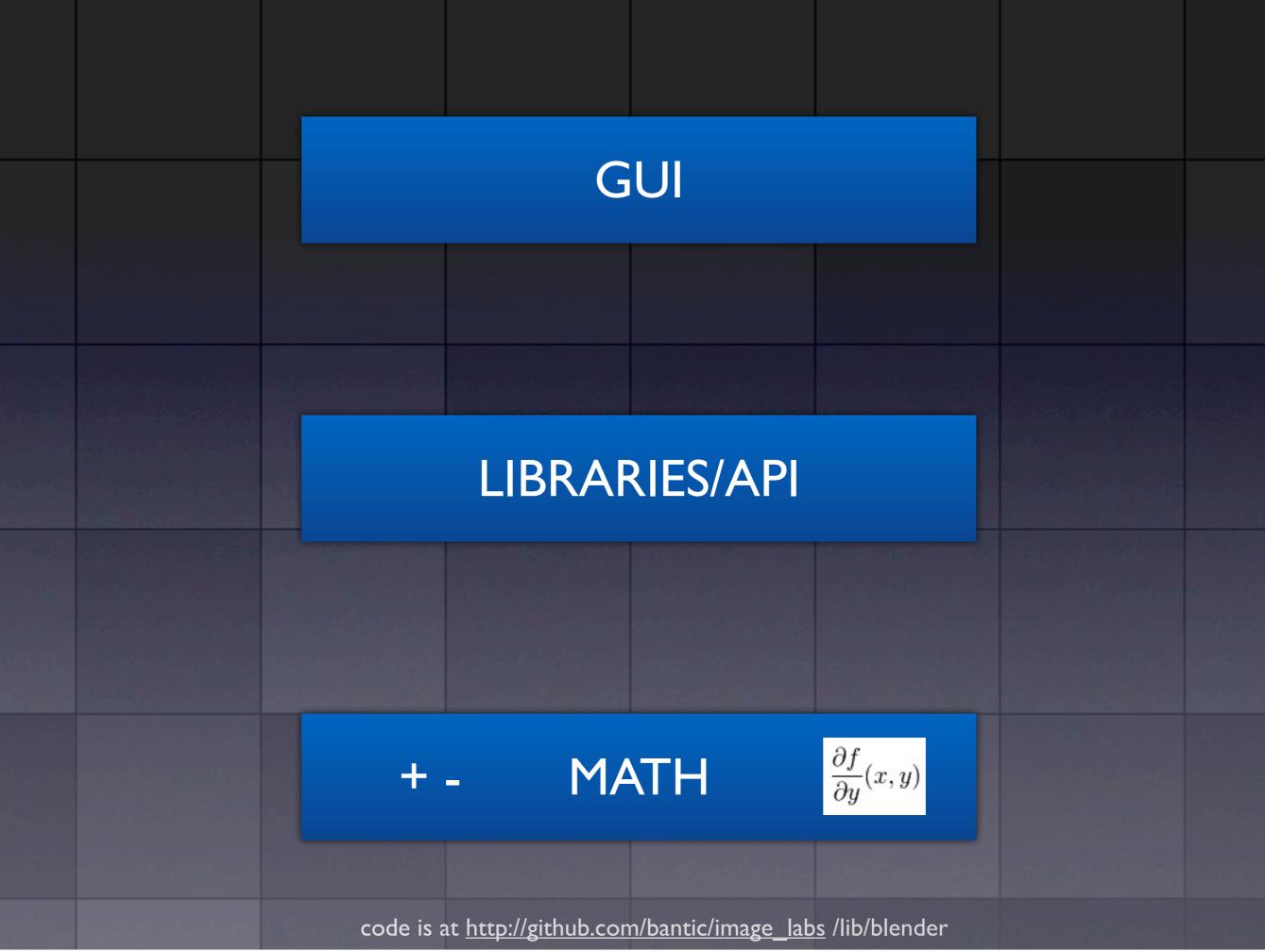
# "with code!"

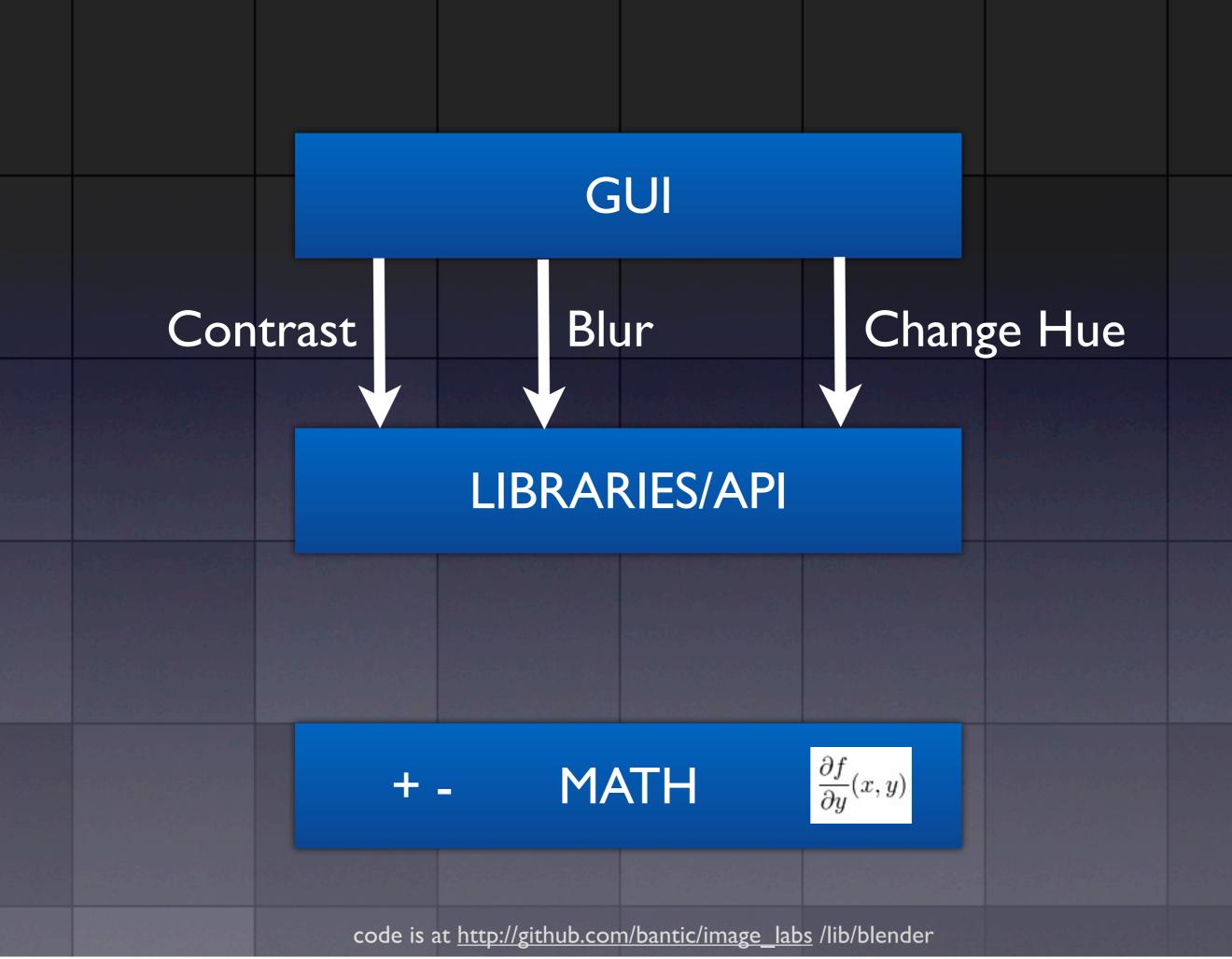
0. 2 2 2 2 2

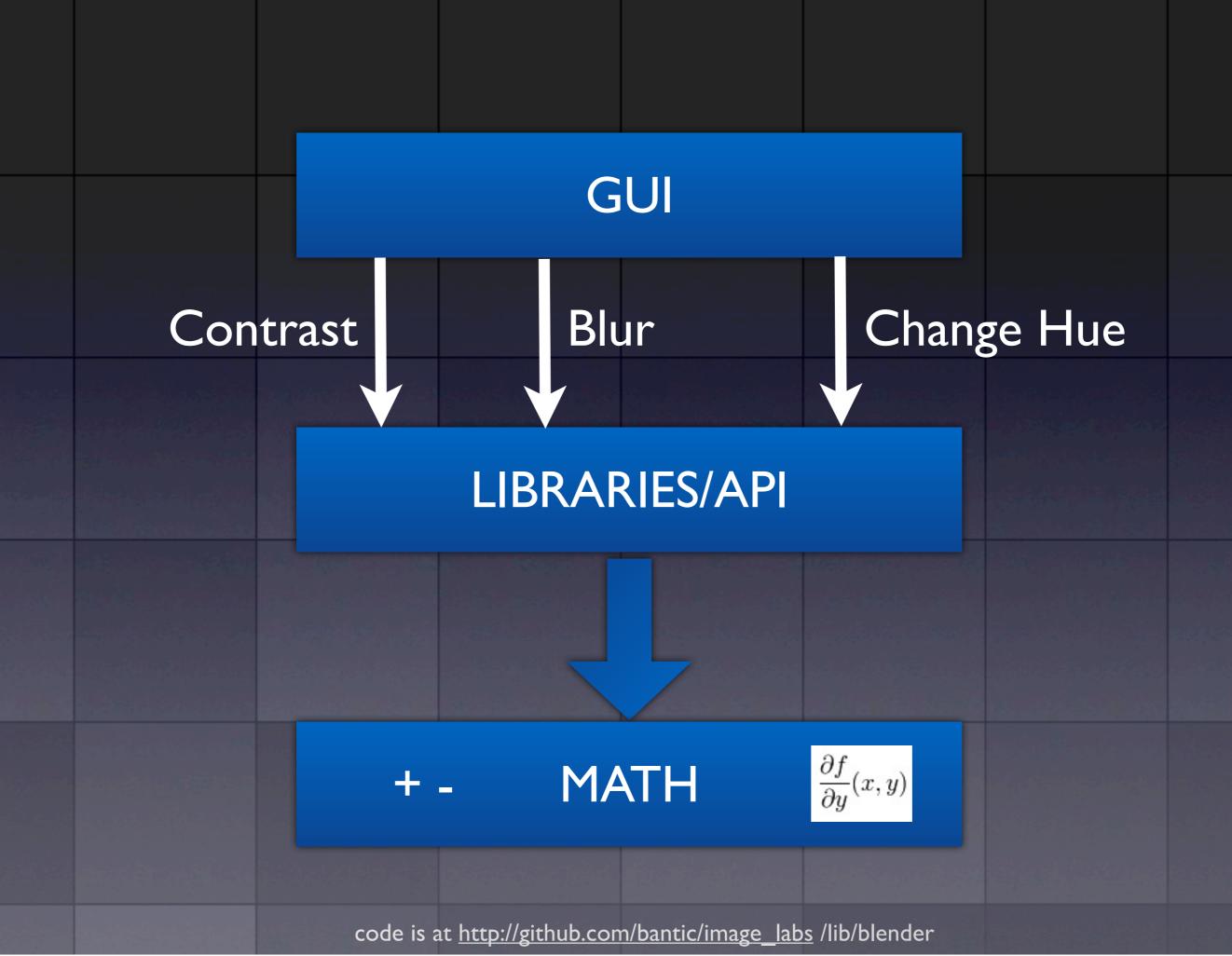


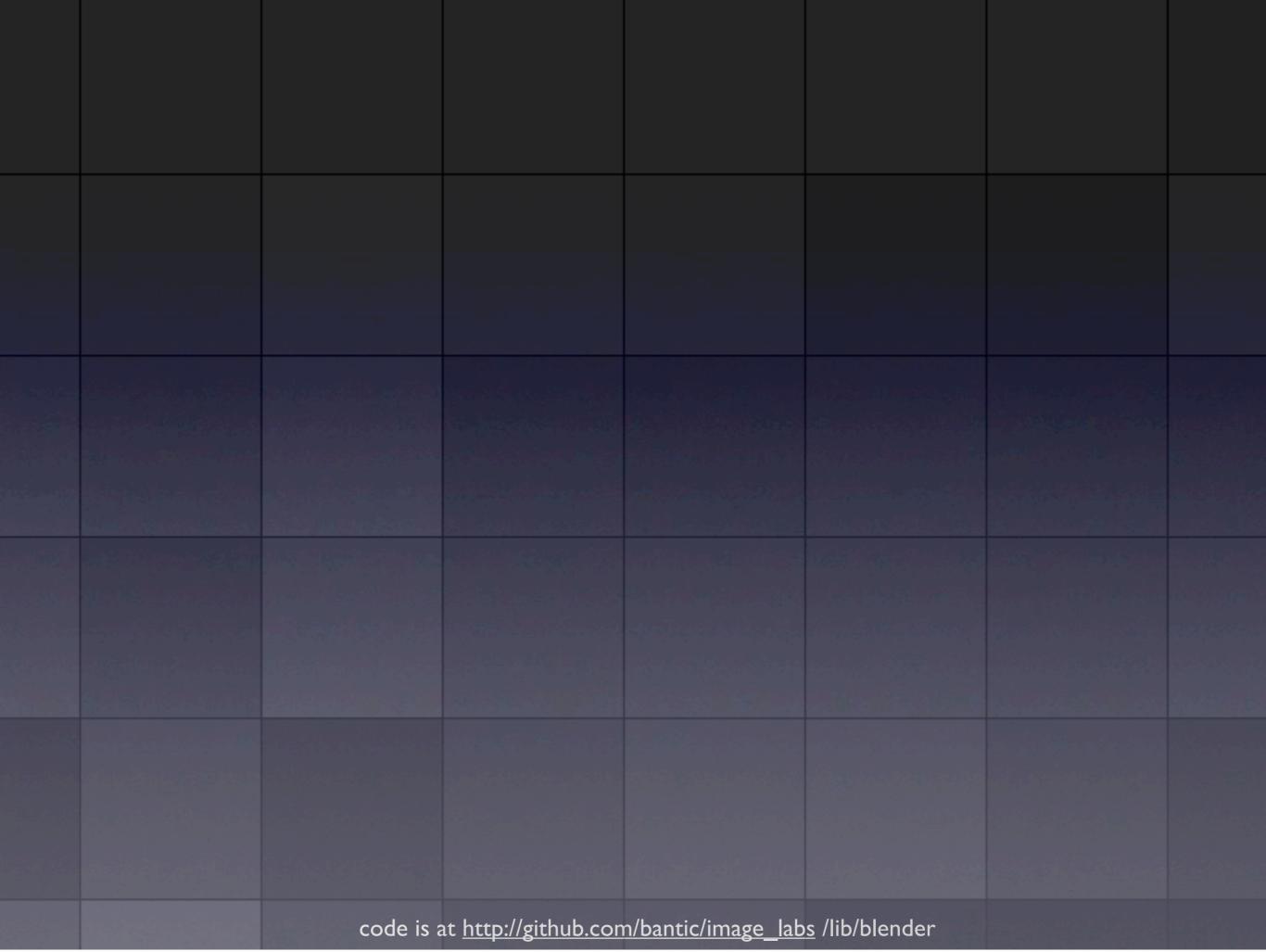


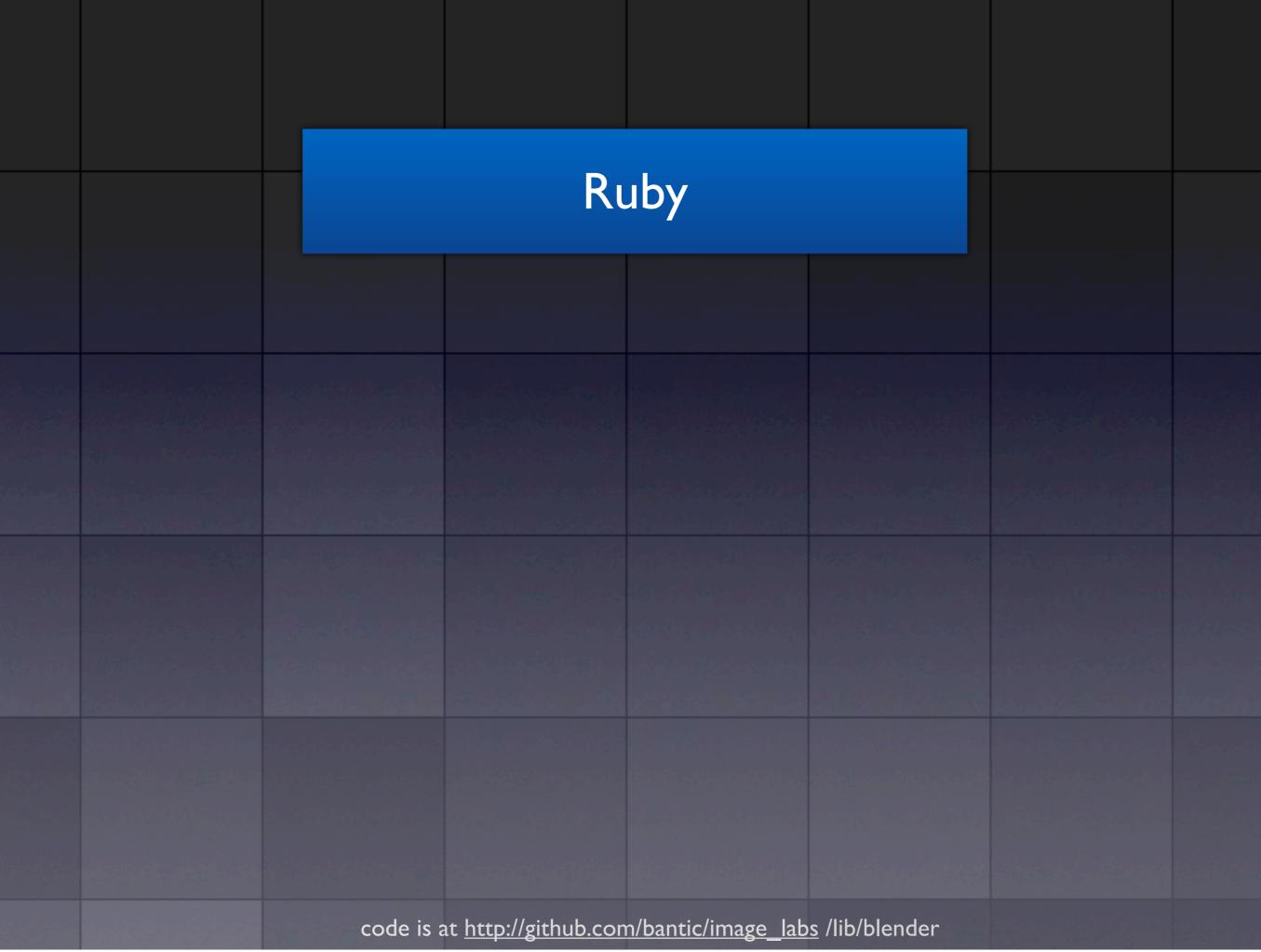


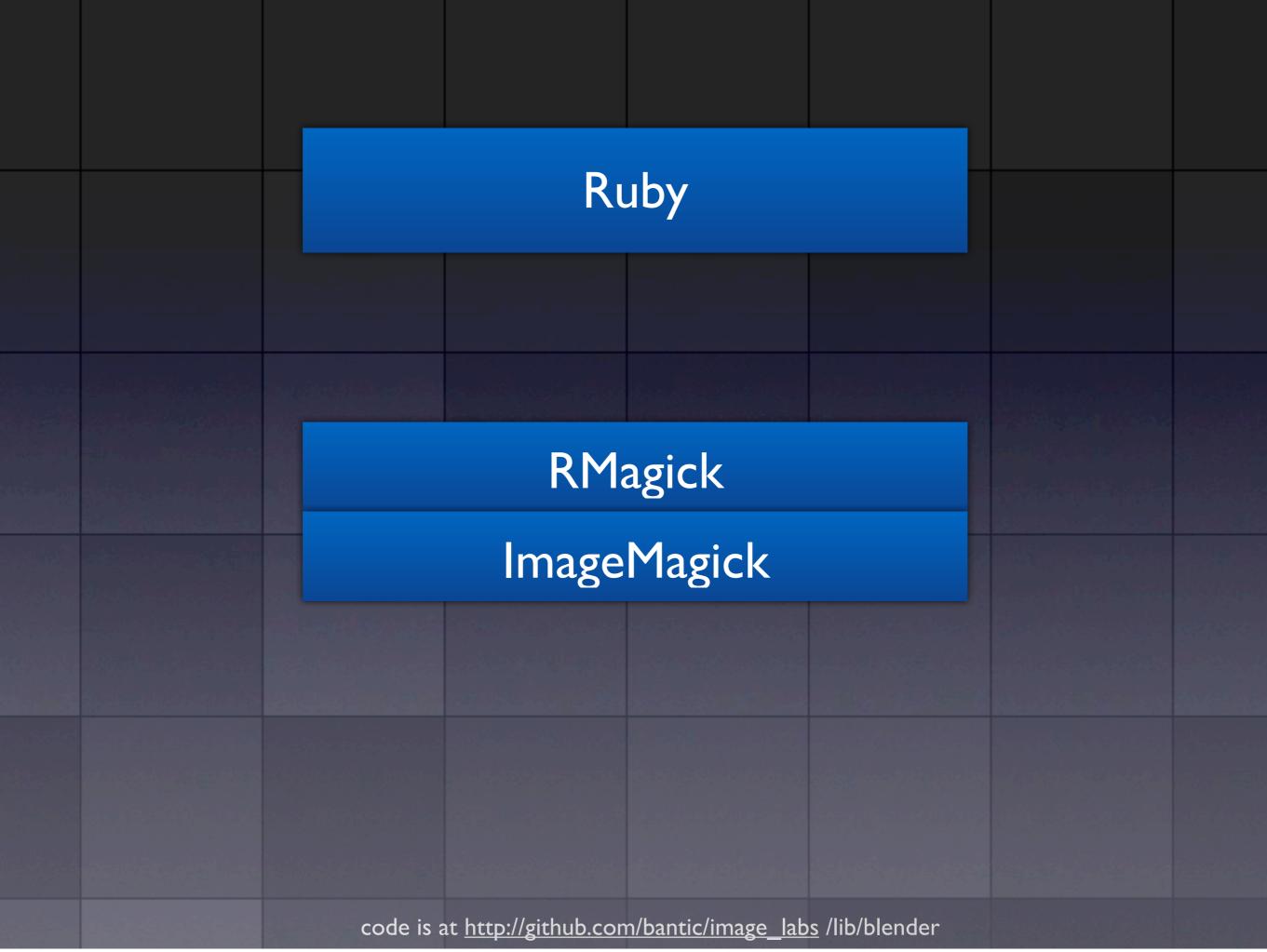


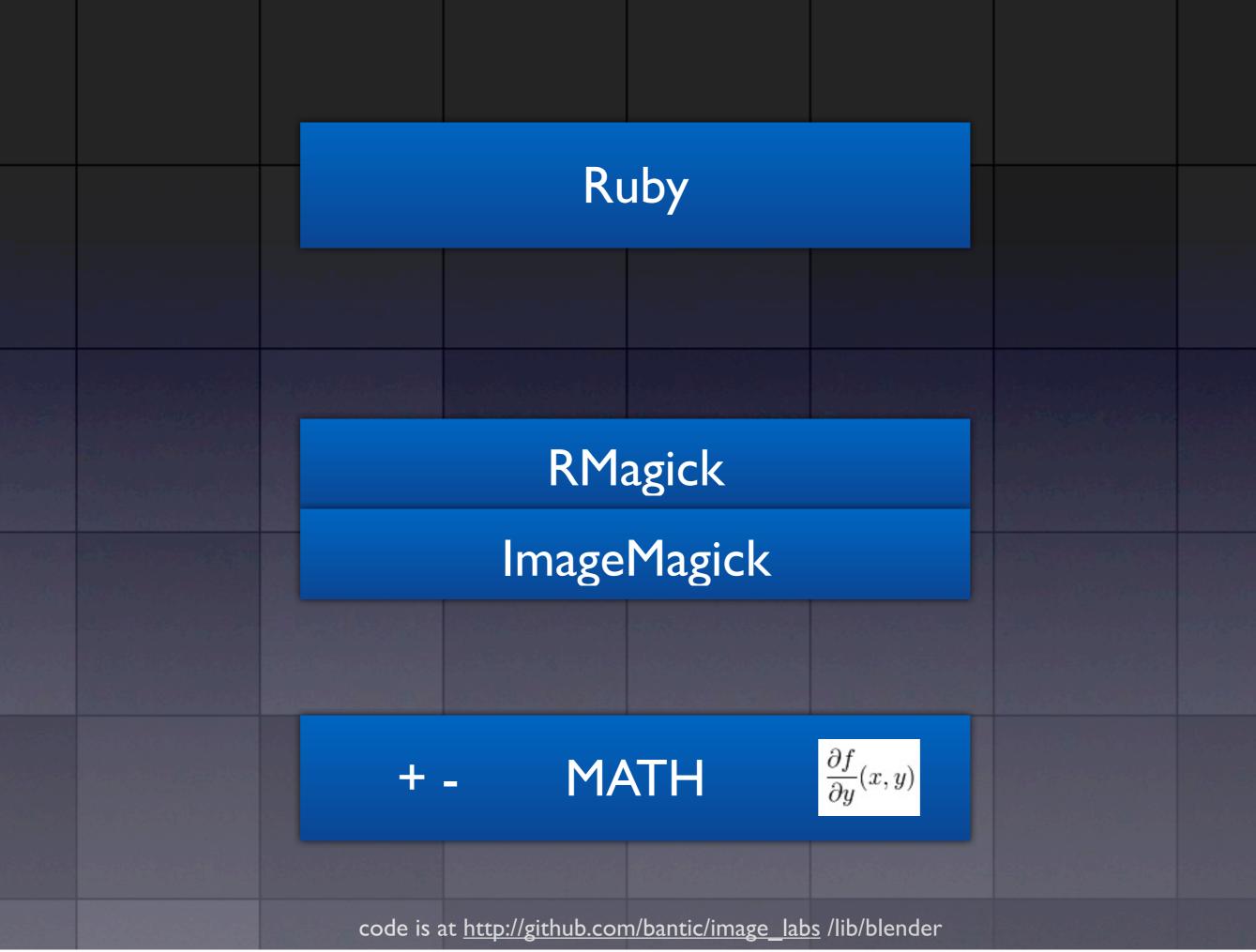


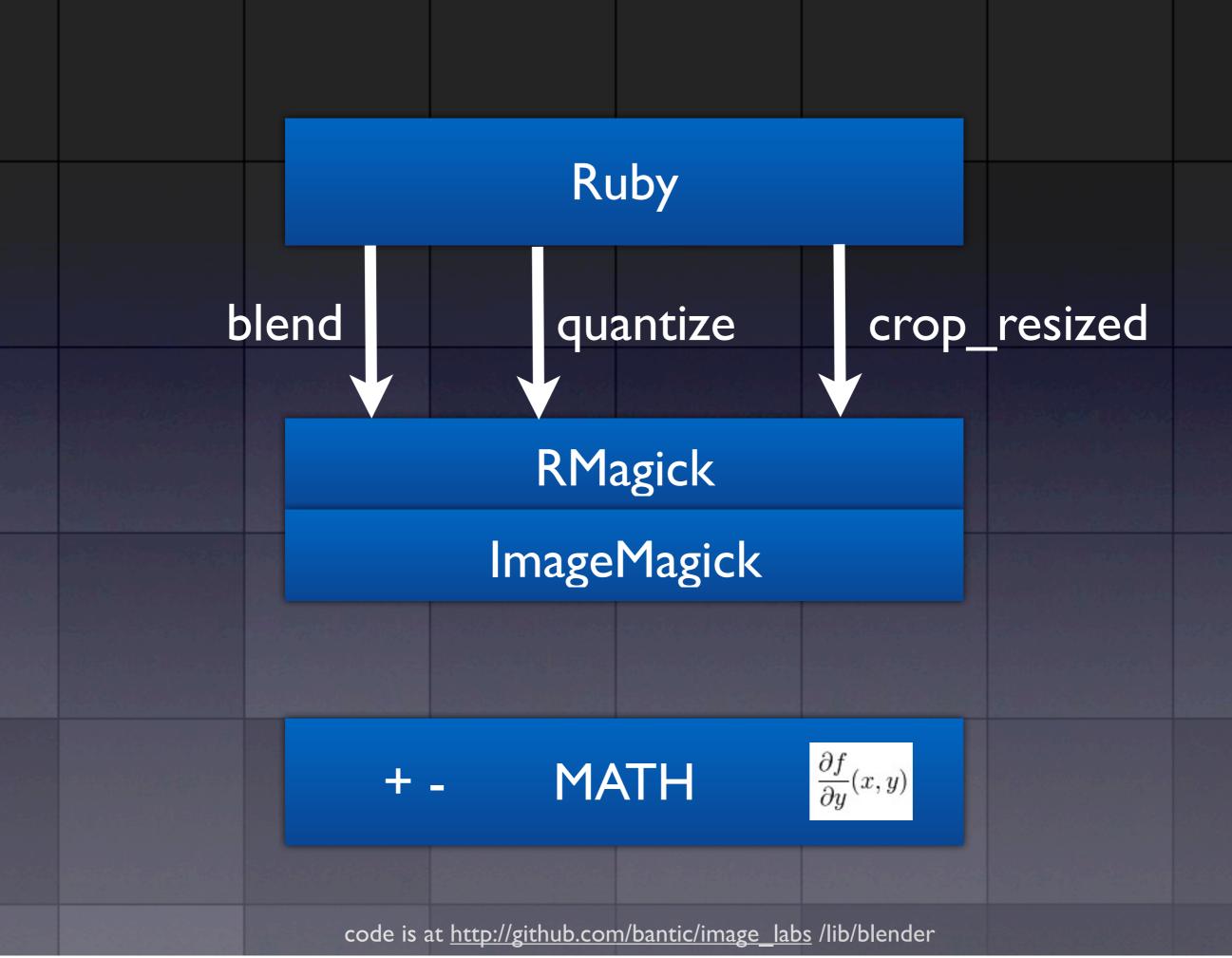


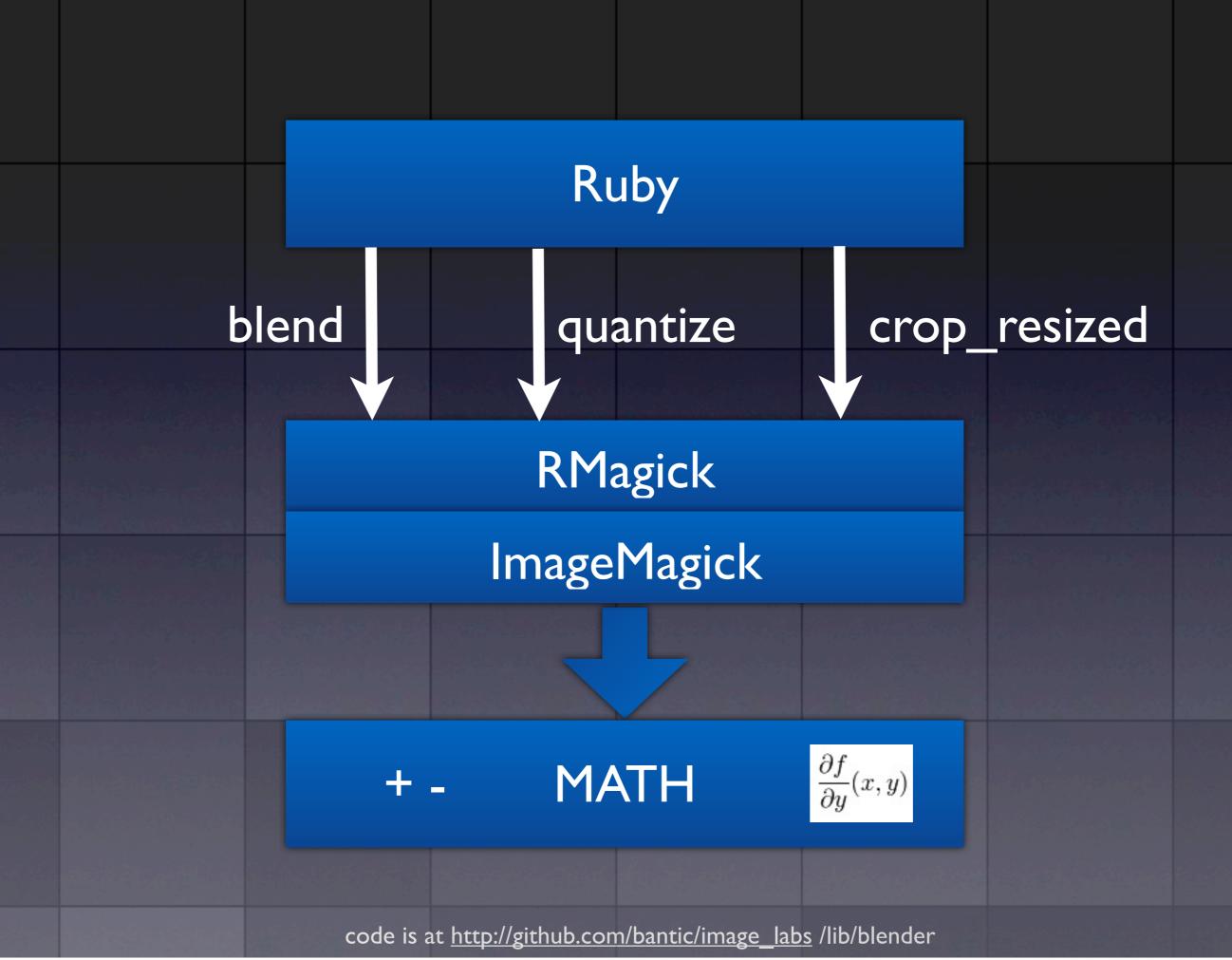


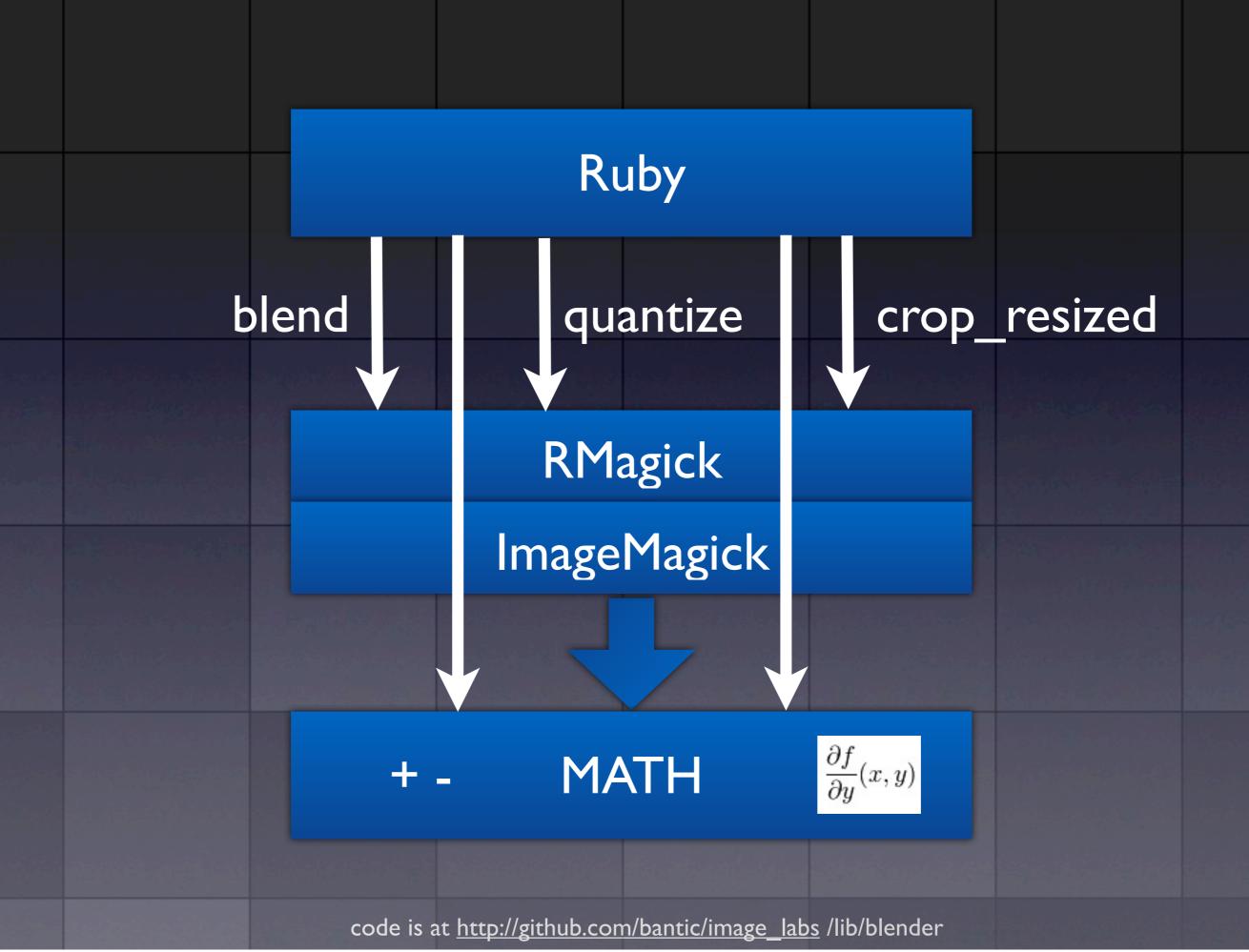












#### state of the art



 RMagick • PhotoMosaic • SeamCarving • Face Detection with OpenCV

#### code

 available on github: <u>http://github.com/bantic/</u> <u>image\_labs</u>

 libraries for PhotoMosaic creation, layer blending, Seam Carving, etc.

code is at <a href="http://github.com/bantic/image\_labs">http://github.com/bantic/image\_labs</a>

# 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 <a href="http://github.com/bantic/image\_labs">http://github.com/bantic/image\_labs</a>

# RMagick Glossary

• Pixel.new(r,g,b)

ImageList.new(path\_to\_image)

• pixel iterators

Draw object

code is at <a href="http://github.com/bantic/image\_labs">http://github.com/bantic/image\_labs</a>

#### flickr credit: mgrenner57

3, 151, 157, 152, 149, 148, 148, 150, 152, 155, 146, 159, 161, 167, 164, 156, 158, 152, 155, 159, 151, 154, 155, 159, 161, 164, 168, 167, 3, 183, 187, 194, 200, 203, 202, 203, 208, 214, 220, 224, 226, 228, 232, 228, 223, 223, 228, 230, 231, 233, 234, 226, 232, 235, 236, 229, 14, 218, 216, 217, 212, 217, 214, 208, 211, 213, 207, 208, 206, 204, 212, 213, 206, 210, 207, 206, 204, 210, 204, 204, 205, 201, 203, 133 .38, 142, 151, 132, 161, 240, 108, 154, 156, 151, 182, 200, 106, 203, 138, 142, 200, 80, 192, 144, 88, 170, 105, 123, 145, 51, 107, 121, 7 9, 33, 25, <mark>31</mark>, 33, 29, 28, 29, 27, 23, 24, 23, 26, 26, 27, 26, 27, 28, 26, 35, 34, 43, 48, 49, 35, 35, 48, 53, 40, 45, 49, 45, 47, 43, 39 56, 58, 56, 54, 63, 62, 61, 68, 69, 67, 66, 161, 142, 74, 51, 110, 206, 72, 74, 94, 93, 107, 120, 106, 105, 110, 118, 115, 203, 190, 10 **33**, 110, 110, 97, 42, 53, 49, 59, 55, 55, 56, 55, 57, 54, 47, 51, 60, 62, 69, 66, 70, 58, 57, 71, 57, 52, 51, 51, 51, 57, 54, 52, 53, 5 , 57, 53, 56, 64, 67, 61, 55, 57, 66, 65, 68, 68, 69, 72, 70, 68, 73, 77, 74, 73, 71, 72, 77, 75, 79, 81, 83, 83, 91, 93, 88, 97, 99, 100 51, 63, 65, 69, 67, 70, 67, 64, 64, 67, 61, 35, 17, 20, 30, 27, 23, 27, 41, 38, 40, 32, 35, 35, 37, 38, 37, 37, 35, 36, 33, 38, 37, 48, 5 9, 13, 15, 29, 21, 7, 11, 24, 21, 9, 12, 5, 10, 15, 9, 56, 14, 8, 13, 4, 20, 17, 14, 19, 20, 19, 4, 20, 41, 26, 31, 23, 19, 36, 25, 29, 2 17, 37, 49, 48, 44, 59, 78, 38, 14, 19, 20, 14, 15, 10, 14, 5, 39, 76, 46, 35, 42, 43, 31, 60, 99, 107, 54, 10, 22, 19, 15, 9, 5, 16, 7, 37, 47, 40, 34, 51, 40, 19, 16, 25, 17, 15, 28, 30, 20, 10, 20, 5, 21, 37, 36, 36, 69, 69, 64, 65, 42, 29, 25, 37, 28, 11, 13, 18, 12, 1 , 88, 82, 85, 74, 75, 78, 73, 75, 78, 76, 70, 72, 73, 65, 65, 70, 69, 68, 70, 64, 64, 70, 67, 61, 60, 44, 43, 48, 61, 71, 71, 70, 69, 72, 7. 7. 7. 6. 7. 7. 7. 6, 5, 6, 6, 8, 7, 4, 6, 6, 6, 6, 6, 6, 6, 6, 5, 7, 5, 6, 6, 6, 6, 5. 6. 5, 6, 5, 5, 6, 6, 5, 5, 5. 6. 6. 7. 5. 6, 4, 6, 7, 7, 6, 6, 7, 5, 6, 6, 5, 6, 6, 6, 6, 6, 7, 6, 8, 6, 8, 7, 8, 6, 8, 10, 8, 9, 13, 11, 8, 10, 10, 9, 8, 9, 9, 9, 10, 11, 14, 12, 12, 12 35, 53, 54, 42, 47, 43, 36, 40, 52, 63, 49, 42, 48, 64, 64, 57, 52, 50, 59, 65, 63, 80, 75, 64, 62, 75, 90, 80, 63, 72, 75, 70, 73, 93, 4, 127, 122, 101, 86, 96, 94, 96, 83, 93, 112, 112, 112, 118, 131, 133, 122, 112, 94, 109, 110, 113, 112, 114, 127, 119, 113, 126, 127, 1 124, 123, 125, 131, 129, 128, 135, 142, 138, 129, 131, 131, 137, 136, 138, 134, 136, 146, 153, 155, 150, 152, 152, 151, 148, 163, 160, 1 184, 194, 189, 191, 195, 195, 194, 188, 189, 194, 200, 195, 197, 200, 200, 211, 211, 210, 206, 205, 211, 206, 205, 209, 204, 212, 215, 8, 239, 239, 236, 240, 237, 235, 234, 233, 232, 229, 225, 230, 228, 225, 222, 221, 219, 219, 221, 249, 214, 212, 213, 210, 206, 204, 205, 151, 124, 113, 131, 148, 239, 110, 178, 186, 155, 169, 218, 87, 201, 163, 116, 186, 164, 129, 185, 80, 106, 166, 84, 145, 75, 68, 148, 7 27, 35, 31, 30, 28, 29, 30, 28, 25, 26, 27, 26, 26, 23, 22, 24, 23, 26, 24, 27, 28, 34, 37, 40, 40, 45, 52, 62, 68, 71, 54, 7 19, 25, 29, 74, 127, 111, 44, 44, 53, 49, 57, 52, 57, 58, 58, 63, 118, 139, 143, 78, 83, 160, 224, 192, 175, 77, 72, 147, 183, 150, 137 6, 35, 154, 116, 93, 132, 141, 159, 120, 77, 133, 133, 81, 31, 18, 24, 19, 31, 33, 39, 35, 40, 44, 40, 39, 43, 43, 42, 47, 57, 61, 59, 63 49, 47, 44, 39, 37, 35, 37, 39, 40, 46, 46, 53, 60, 58, 57, 61, 68, 77, 75, 75, 62, 97, 99, 90, 91, 93, 68, 166, 71, 78, 66, 64, 57, 78, 74, 70, 77, 74, 70, 67, , 71, 74, 69, 78, 70, 71, 68, 69, 65, 62, 65, 63, 53, 35, 29, 26, 30, 32, 36, 65, 42, 39, 42, 38, 39, 3 70. 72 , 5, 3, 36, 41, 34, 36, 32, 29, 38, 27, 16, 41, 1, 1, 12, 13, 41, 9, 10, 15, 3, 17, 20, 27, 26, 13, 14, 12. 13. 9. 7. 7. 9. 6. 8 18. 77, 54, 56, 64, 50, 34, 51, 49, 41, 40, 33, 26, 24, 7, 13, 29, 45, 43, 63, 64, 66, 11, 12, 28, 16, 8, 29, 22, 4, 39, 34, 38, 5 6 ... , 177, 55, 63, 52, 32, 31, 37, 35, 19, 14, 20, 13, 13, 26, 28, 17, 6, 8, 2, 26, 54, 54, 59, 67, 43, 40, 14, 19, 34, 36, 58, 89 11 18, 67, 67, 62, 71, 68, 67, 70, 64, 70, 67, 66, 69, 65, 60, 64, 68, 64, 64, 68, 69, 66, 62, 59, 56, 60, 57, 58, 60, 57, 55, 56 65. 62. 7, 6, 6, 6, 6, 6, 6, 6, 6, 7, 5, 5, 6, 7, 7, 7, 6, 6, 5, 6, 6, 7, 7, 4, 6, 5, 6, 6, 6, 6, 5, 5, 6, 7, 5, 5, 6, 5, 8, 7, 5, 8, 8, 9, 8, 10, 8, 9, 10, 9, 8, 7, 8, 8, 7, 7, 7, 9, 10, 9, 8, 9, 9, 9, 9, 11, 10, 12, 10, 12, 11, 12, 10, 10, 12, 11, 12, 11, 12, 15, 1 S. te 1, 42, 44, 43, 54, 53, 52, 55, 56, 57, 60, 69, 72, 63, 66, 63, 70, 74, 77, 75, 78, 75, 74, 77, 86, 81, 84, 81, 84, 83, 80, 79, 79, 83, 87, A, 120, 121, 116, 114, 116, 108, 102, 106, 113, 117, 114, 94, 96, 102, 100, 102, 46, 36, 134, 148, 139, 116, 90, 102, 94, 89, 111, 116, 1 1, 107, 110, 116, 115, 120, 131, 154, 165, 175, 170, 166, 165, 167, 176, 171, 170, 168, 166, 171, 174, 173, 179, 179, 180, 184, 190, 184, 1, 212, 213, 213, 219, 216, 219, 228, 222, 226, 227, 228, 230, 231, 229, 221, 229, 230, 229, 232, 240, 234, 238, 243, 241, 239, 233, 236, 30, 229, 232, 232, 232, 233, 225, 222, 214, 212, 214, 205, 204, 204, 203, 201, 199, 198, 196, 197, 197, 195, 192, 193, 194, 196, 196, 197 204, 207, 209, 216, 219, 210, 207, 213, 215, 217, 217, 220, 221, 219, 222, 223, 221, 221, 220, 223, 231, 229, 232, 239, 218, 223, 238, 24 218, 244, 155, 56, 51, 59, 108, 79, 123, 82, 150, 117, 116, 121, 119, 102, 107, 101, 119, 96, 130, 185, 127, 126, 140, 204, 88, 200, 141 9, 65, 63, 114, 61, 84, 52, 56, 78, 50, 61, 42, 45, 65, 46, 46, 31, 41, 43, 38, 36, 31, 40, 39, 28, 38, 28, 34, 25, 30, 26, 30, 28, 25, 2 33, 23, 19, 27, 29, 22, 28, 29, 23, 24, 30, 29, 32, 34, 30, 32, 32, 31, 43, 55, 22, 25, 18, 25, 16, 19, 57, 125, 127, 54, 46, 45, 51, 56, 175, 216, 200, 237, 213, 139, 194, 150, 143, 215, 217, 178, 157, 147, 164, 193, 91, 112, 157, 174, 123, 139, 105, 92, 135, 183, 152, 31 36. 39 43. 41. 40. 46. 46. 44. 41. 39. 40. 42. 40. 38. 44. 40. 40. 44. 41. 48. 44. 41. 44. 50. 54. 49. 53. 55. 55. 56. 57. 53. 49.

## RMagick Pixel by Pixel

out

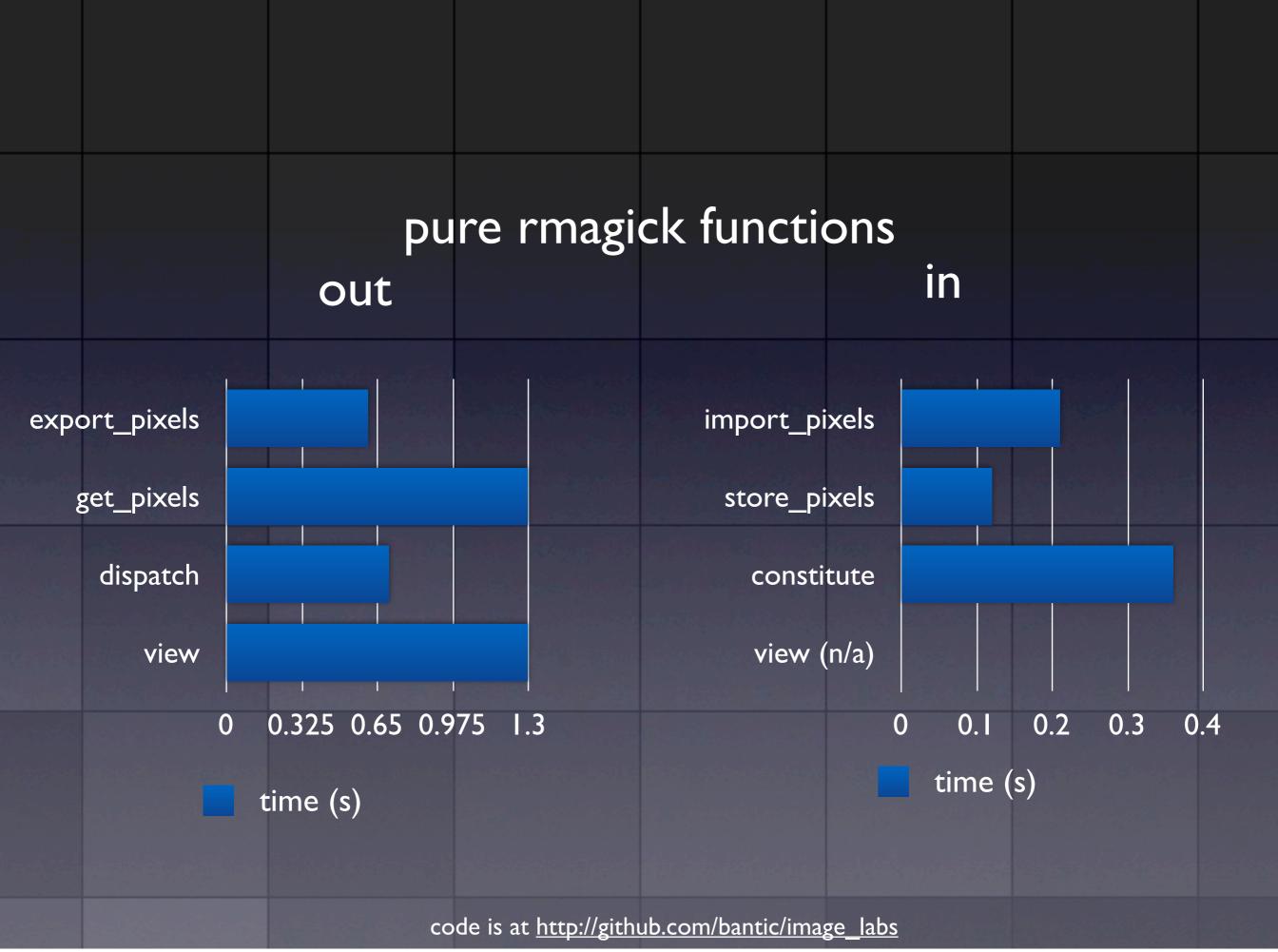
in

- Image#export\_pixels
- Image#get\_pixels
- Image#dispatch

- Image#import\_pixels
- Image#store\_pixels
- Image#constitute

#### Image#view (never use)

code is at <a href="http://github.com/bantic/image\_labs">http://github.com/bantic/image\_labs</a>



# 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

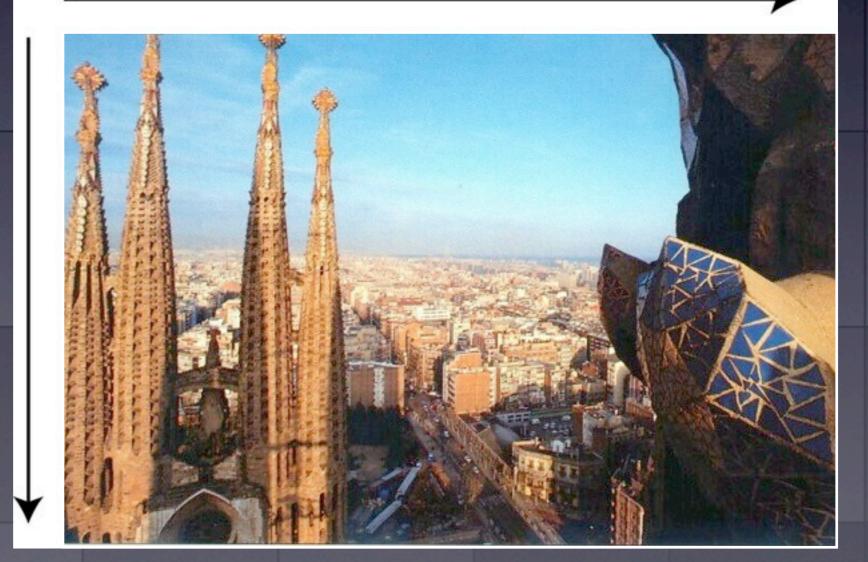
code is at <a href="http://github.com/bantic/image\_labs">http://github.com/bantic/image\_labs</a>

# RMagick Pixel by Pixel

x width columns

height rows

0,0

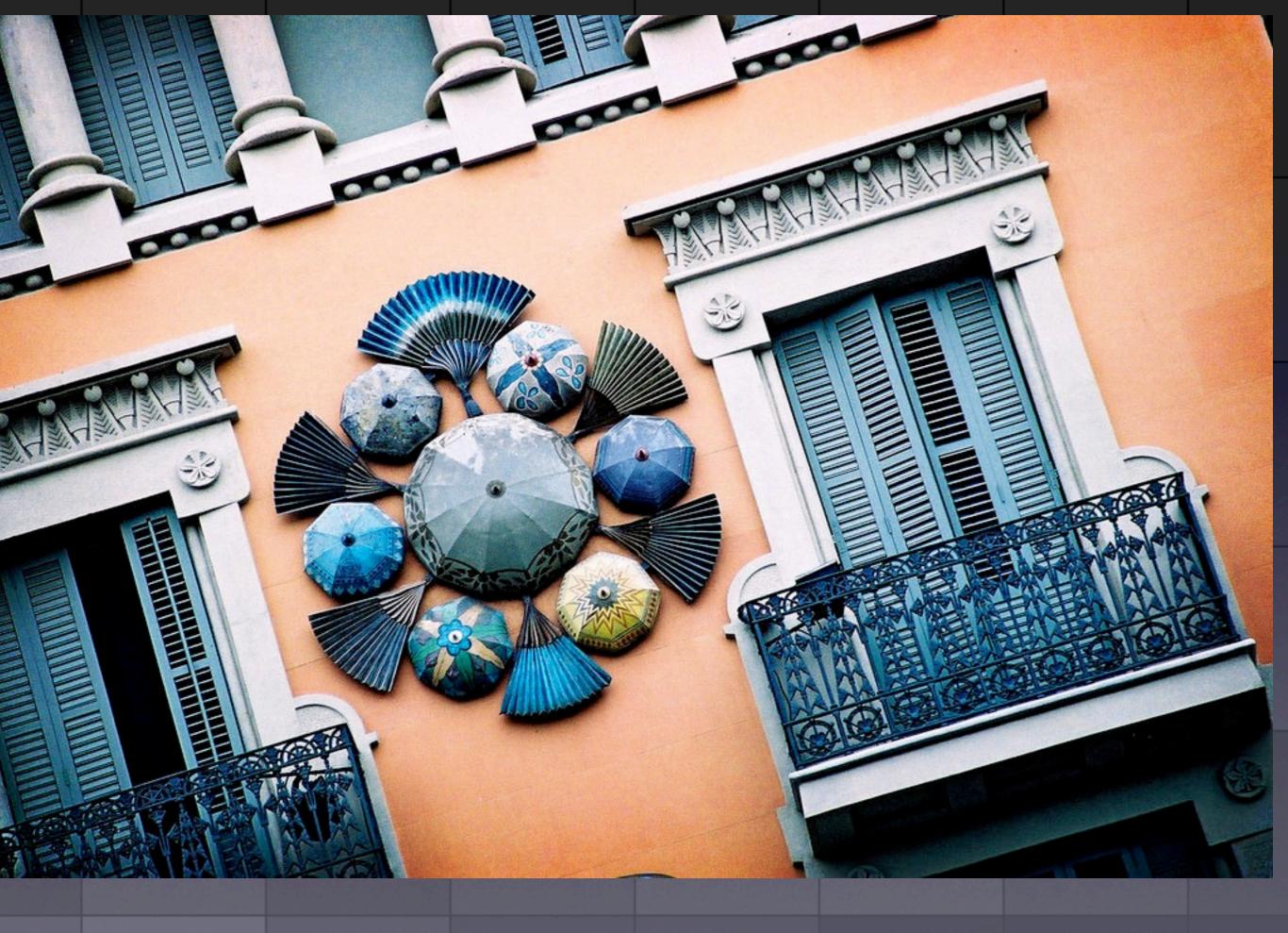


 RMagick PhotoMosaic • SeamCarving Face Detection with OpenCV RMagick
PhotoMosaic
SeamCarving
Face Detection with OpenCV

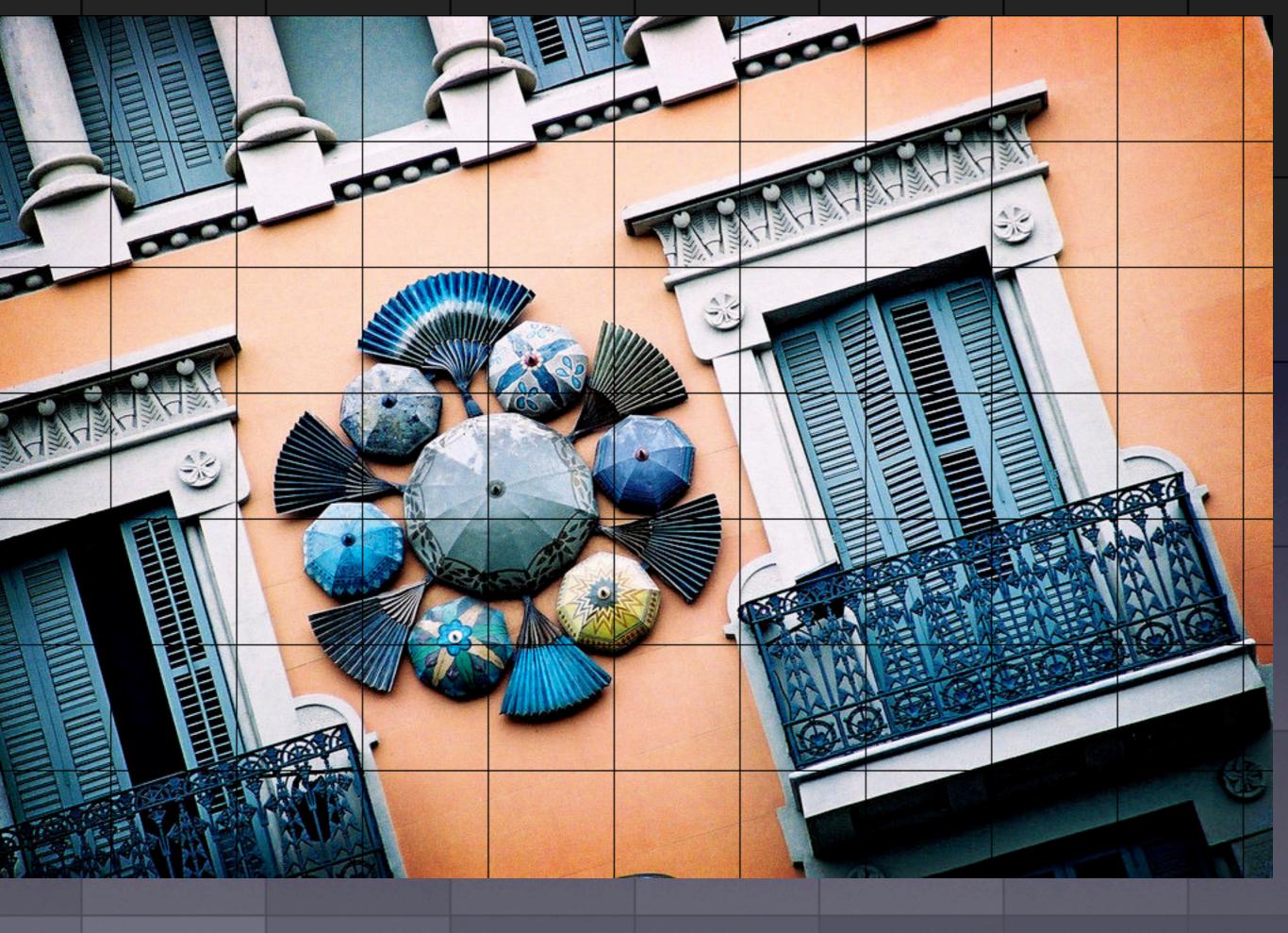
#### create a photomosaic

analyze regions of the image for intensity
replace them with new "pixels:"
images of the same intensity

code is at <a href="http://github.com/bantic/image\_labs">http://github.com/bantic/image\_labs</a> /lib/mosaicer



code is at <a href="http://github.com/bantic/image\_labs">http://github.com/bantic/image\_labs</a> /lib/mosaicer

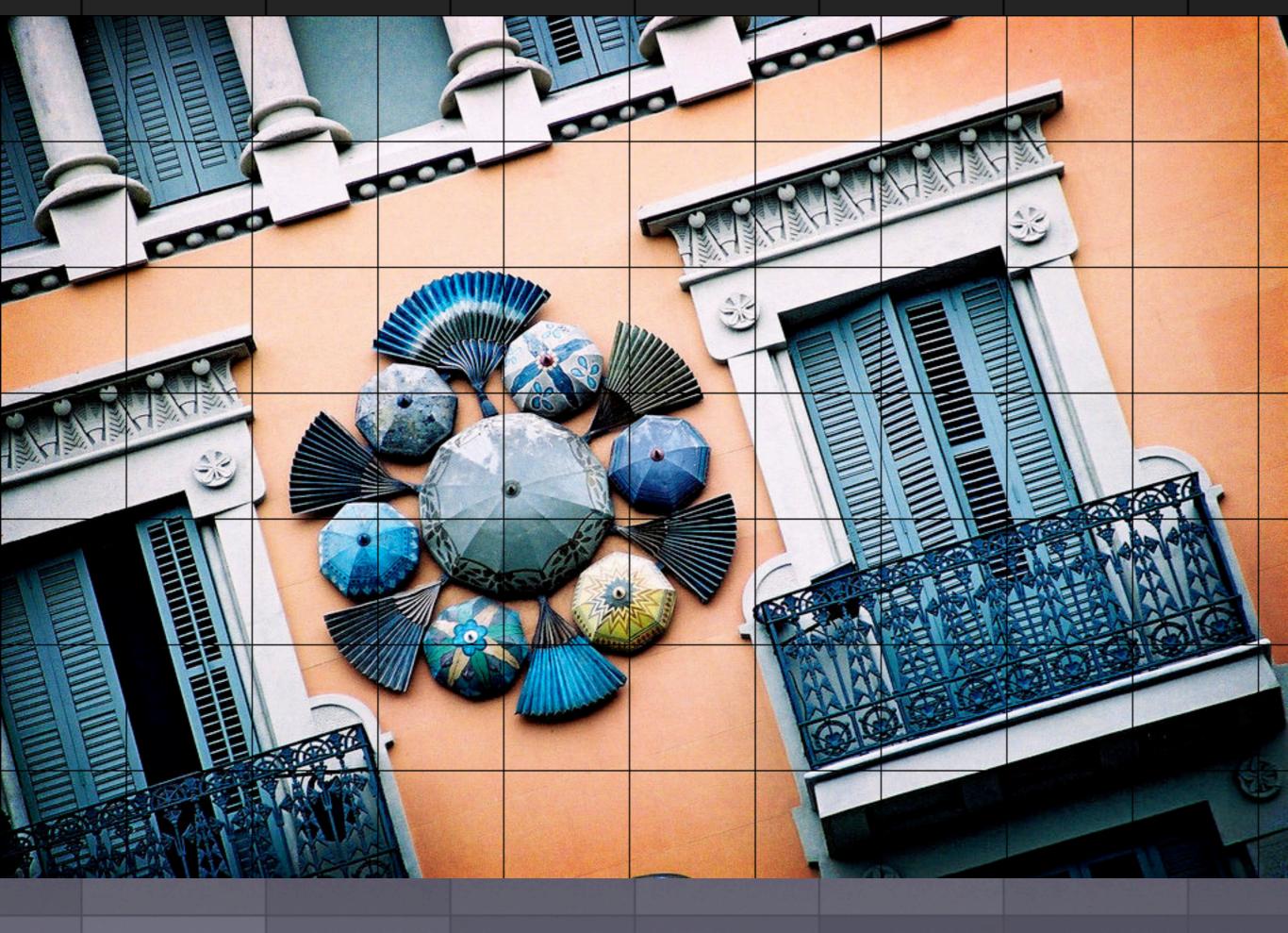


>> cfour:image\_labs(master) cory\$ bin/rmagicksh
>> img = img("images/barcelona-casa.jpg")
>> pm = PhotoMosaicer.new(img, 100, 100)
>> pm.create\_photo\_mosaic!

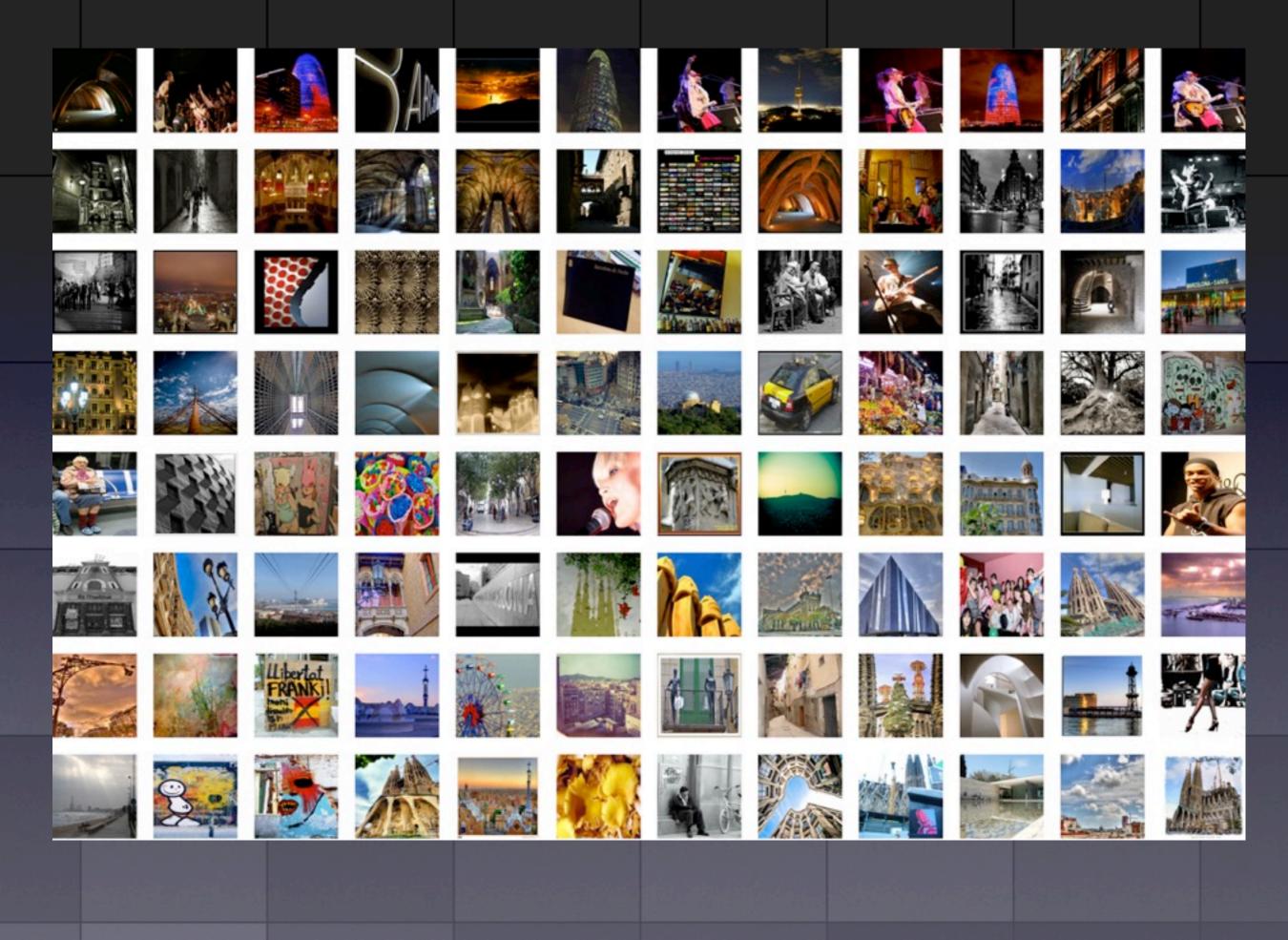
```
def intensity_for_image_portion(x_offset, y_offset, columns, rows)
    intensity = 0
    pixel_count = 0
    x_offset.upto(x_offset + columns - 1) do |column|
    y_offset.upto(y_offset + rows - 1) do |row|
        break if column >= input_image_pixels.size
        break if row >= input_image_pixels.first.size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

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

end

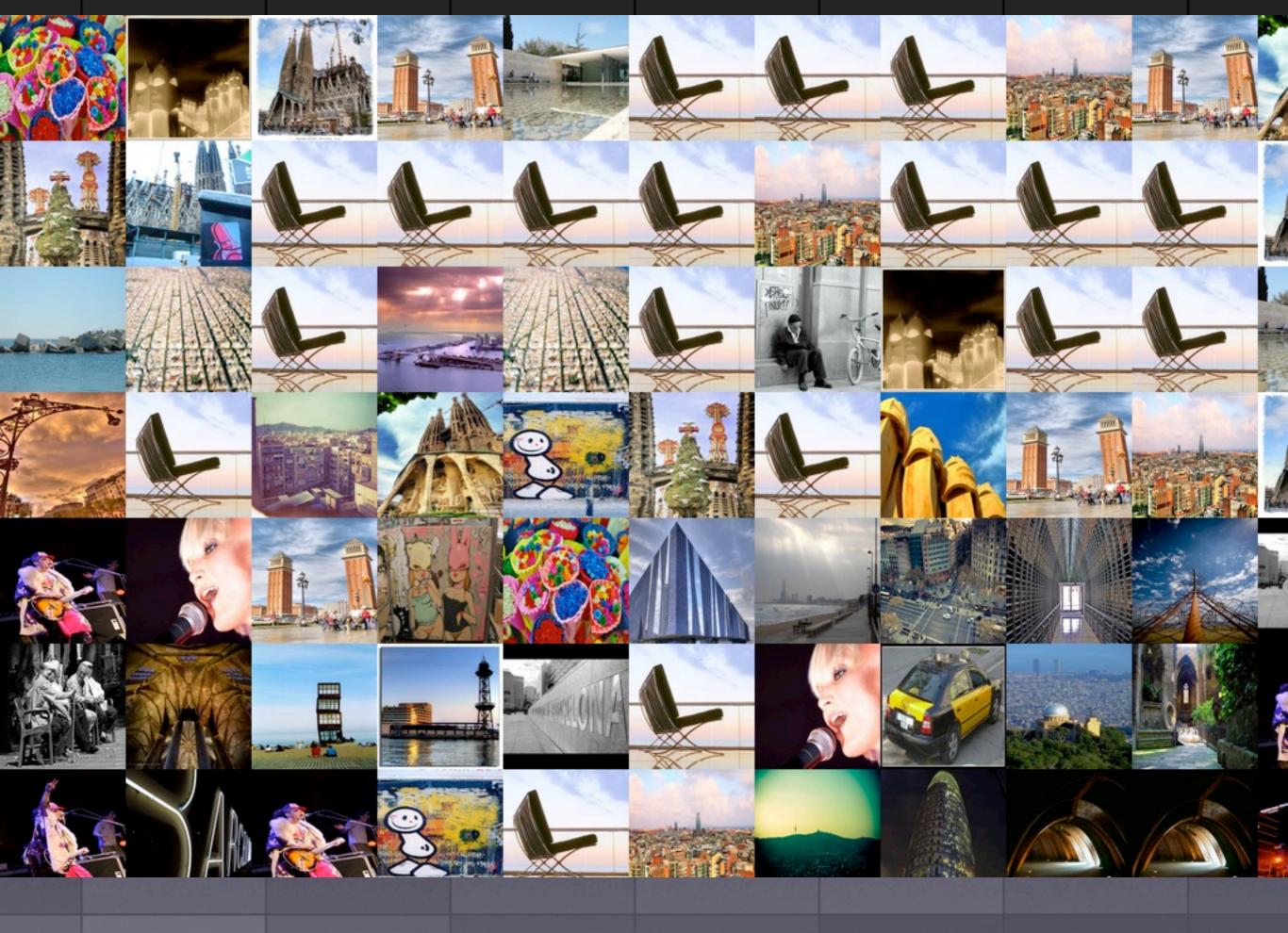


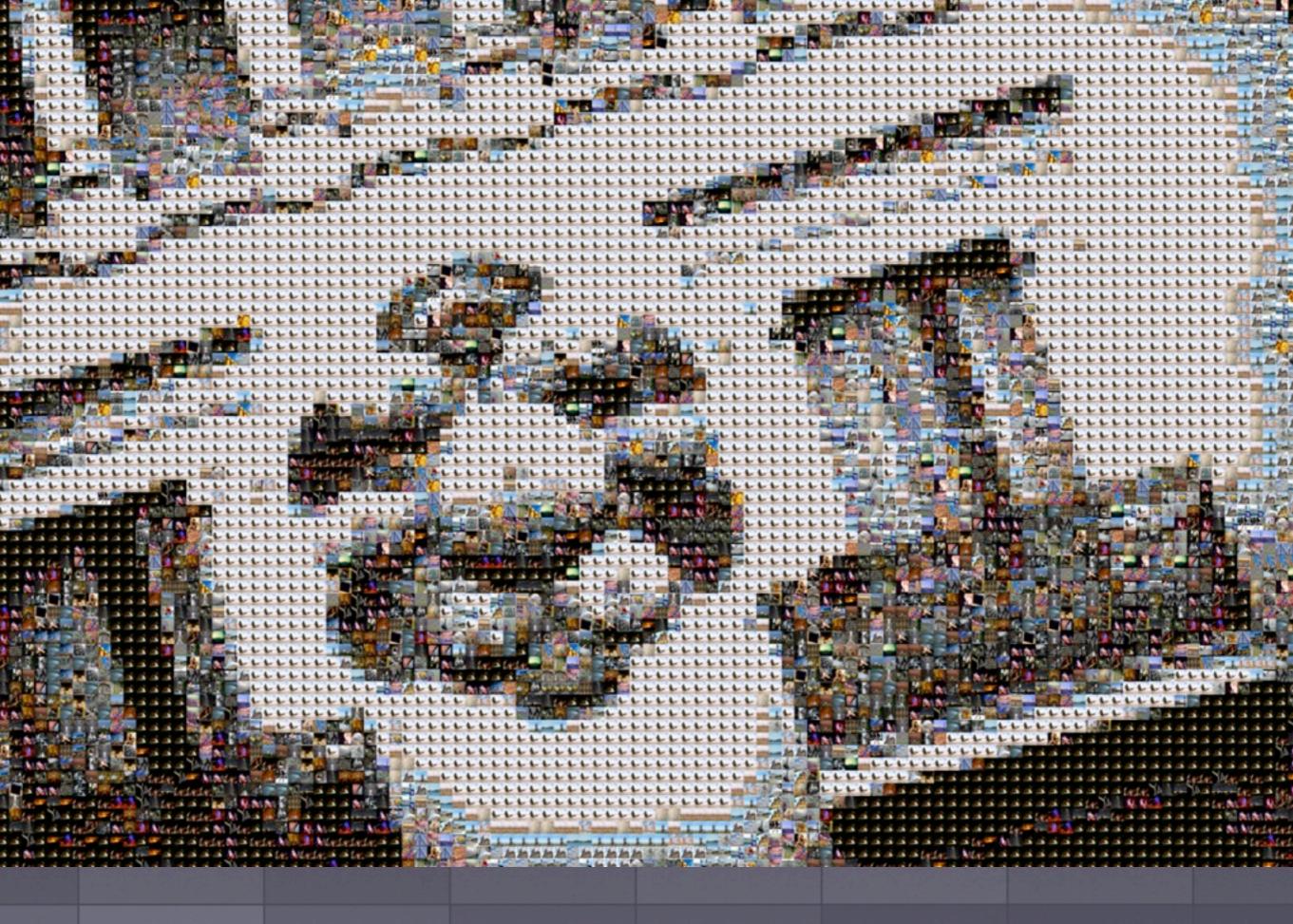
code is at <u>http://github.com/bantic/image_labs</u> /lib/mosaicer									



```
>> pm = PhotoMosaicer.new(img, 100, 100)
>> pm.source_images_dir = "images/barcelona-thumbnails"
>> pm.create photo mosaic!
   def calculate_source_image_intensities!
     @scaled_images = ImageList.new(*Dir.glob(tmp_dir + "/*"))
     @intensities =
     @image_hash = {}
     @scaled_images.each do limgl
       img_intensity = img.intensity
       @intensities << img_intensity
       @image_hash[img_intensity] = img
     end
     @intensities.sort!
    end
  # Find the closest image in our hash of source images
  nearest_intensity = @intensities.detect {|i| intensity <= i} || @intensities.last</pre>
  nearest_intensity_image = @image_hash[nearest_intensity]
```

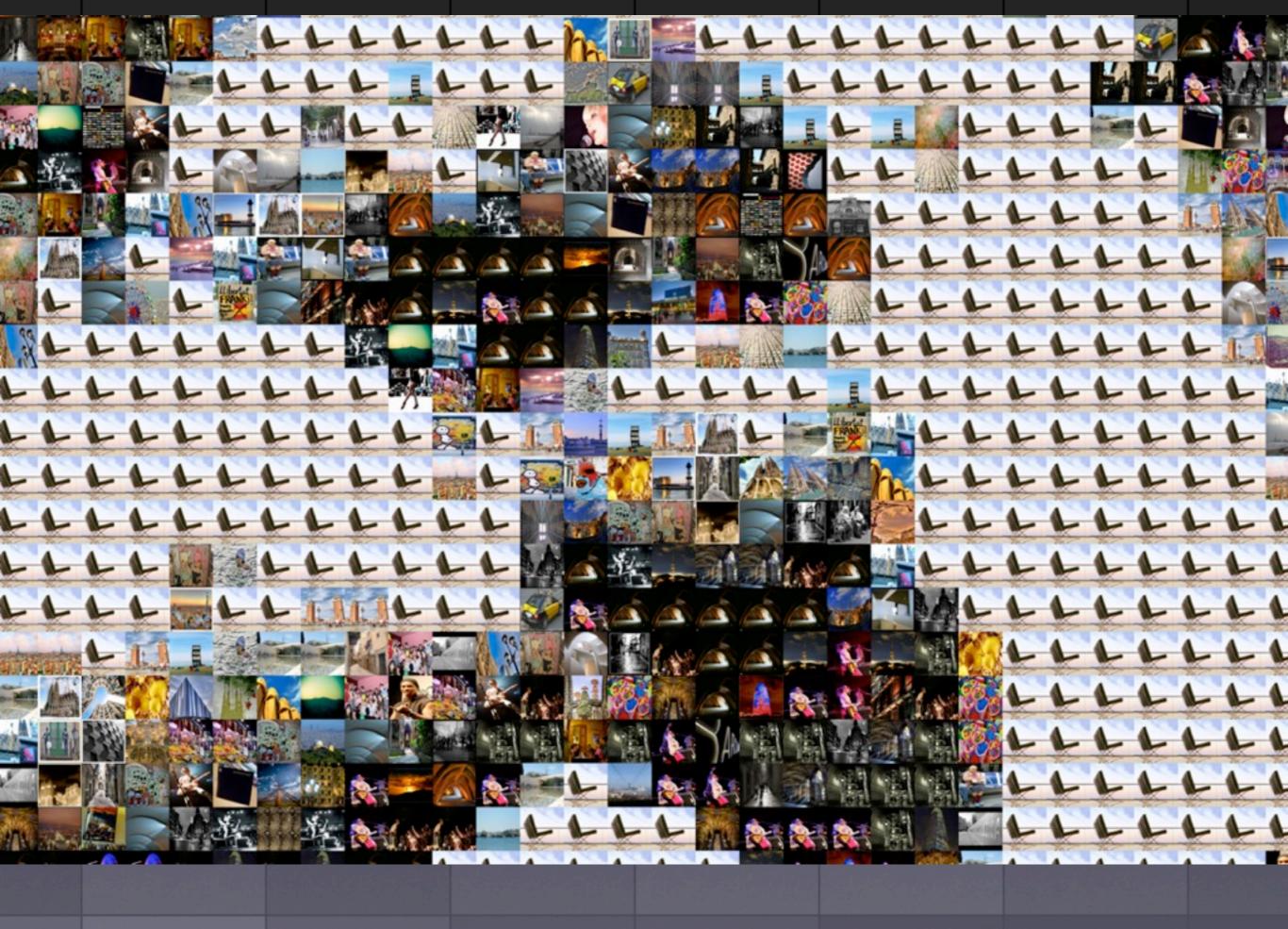
code is at <u>http://github.com/bantic/image_labs</u> /lib/mosaicer									

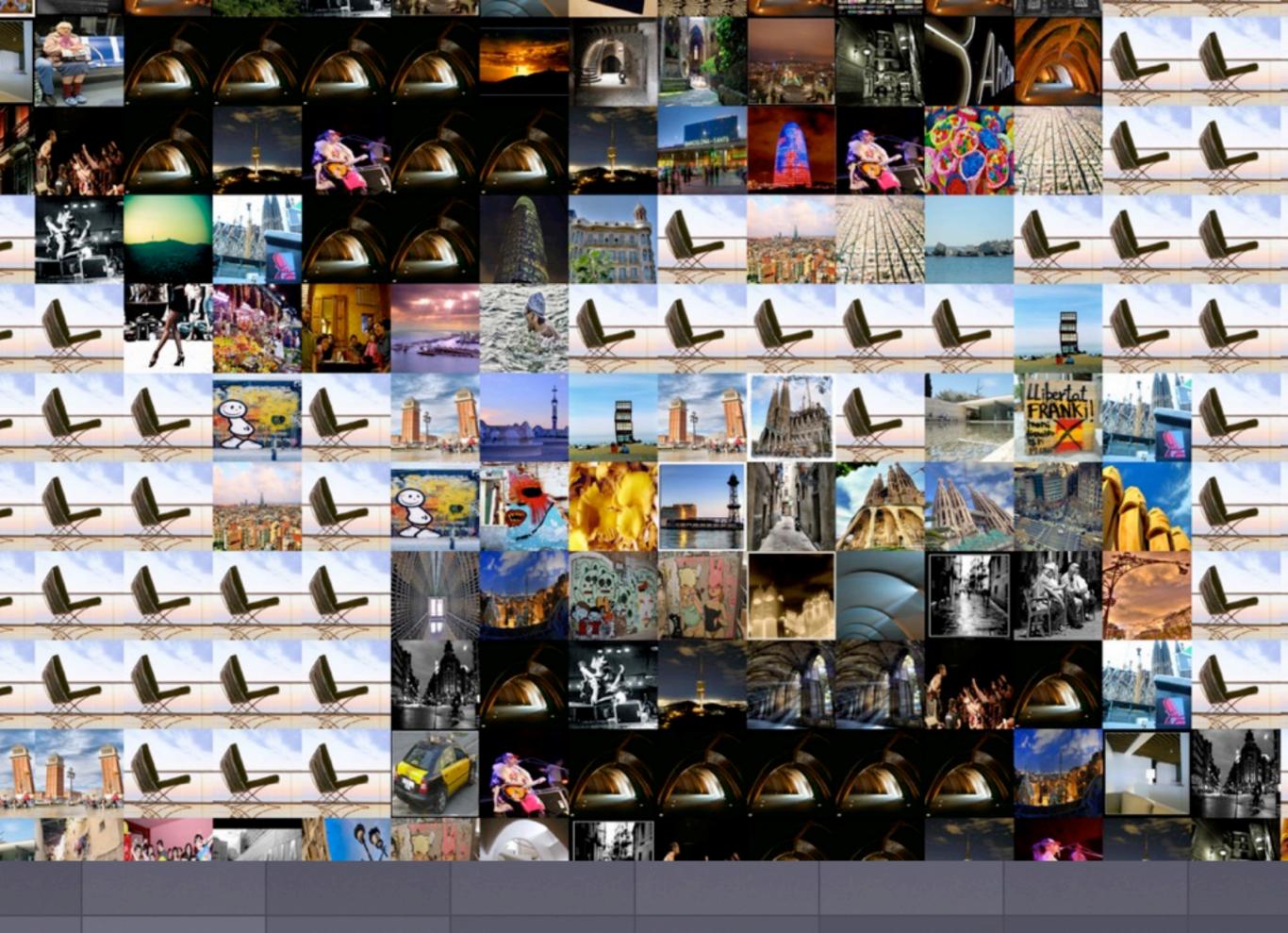


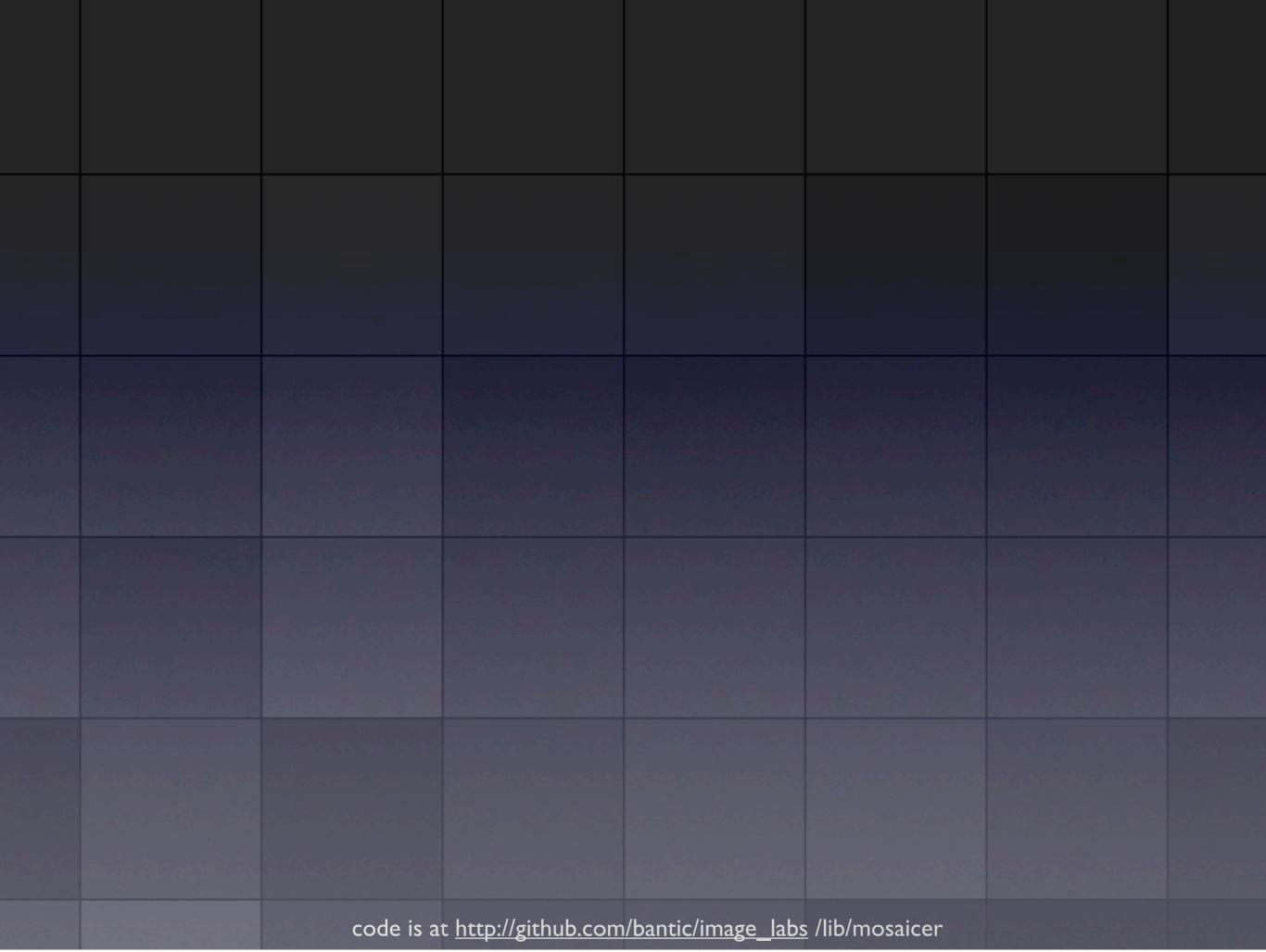


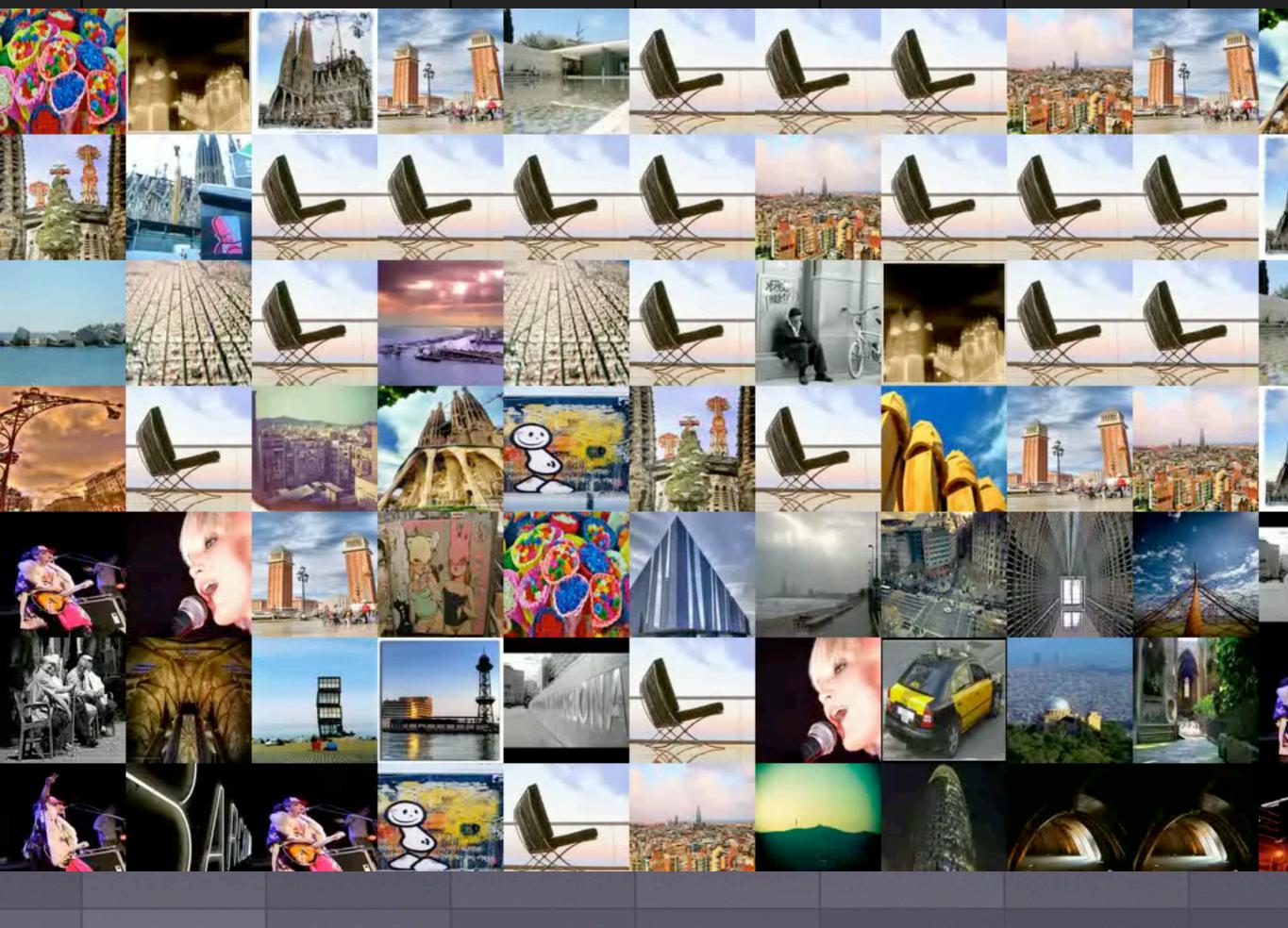
LLL LL 1.5 0 LL -AN 198 A 60 A at - St . . L

code is at <a href="http://github.com/bantic/image\_labs">http://github.com/bantic/image\_labs</a> /lib/mosaicer

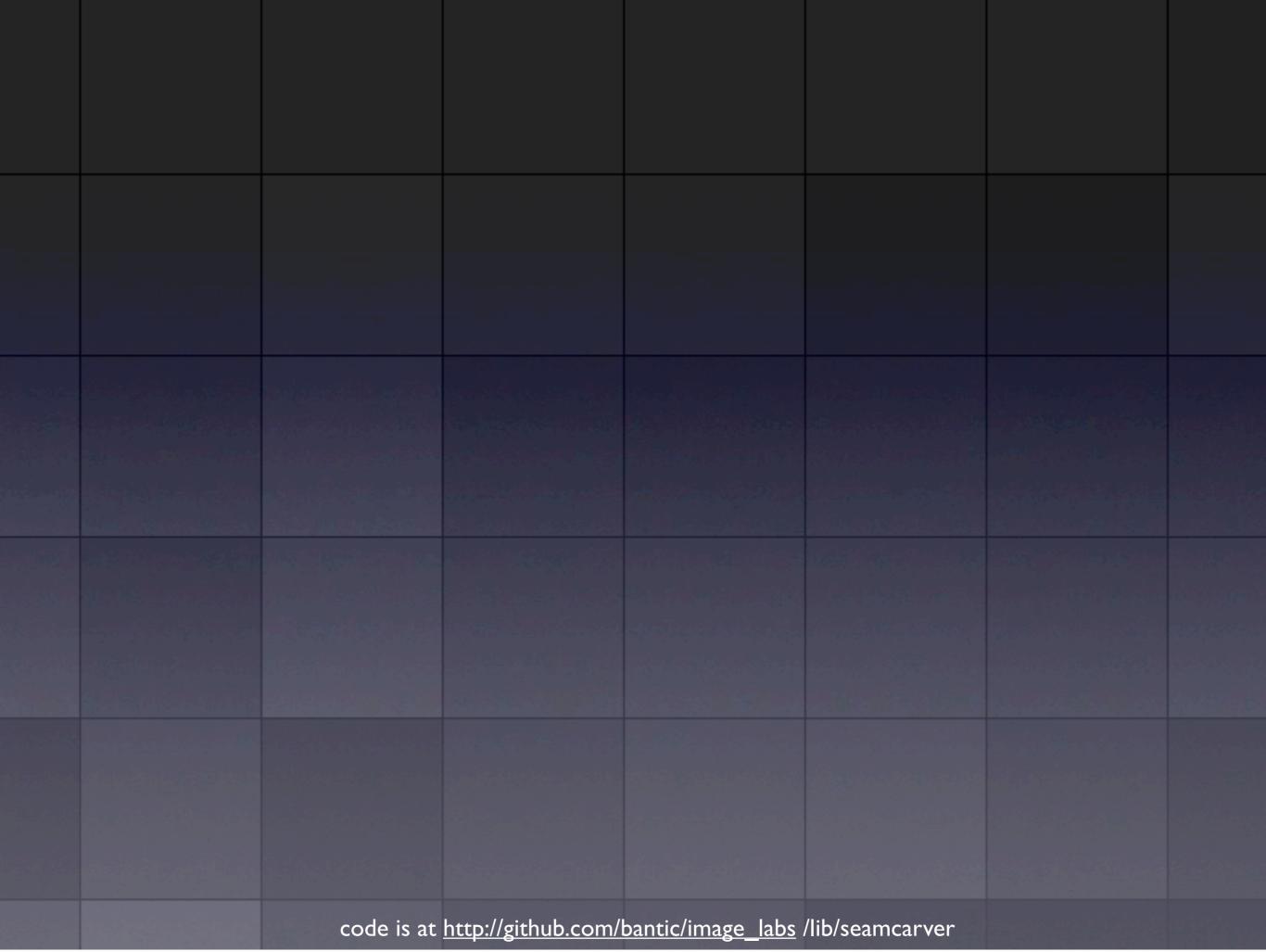








 RMagick PhotoMosaic • SeamCarving Face Detection with OpenCV





## 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 <a href="http://github.com/bantic/image\_labs">http://github.com/bantic/image\_labs</a> /lib/seamcarver



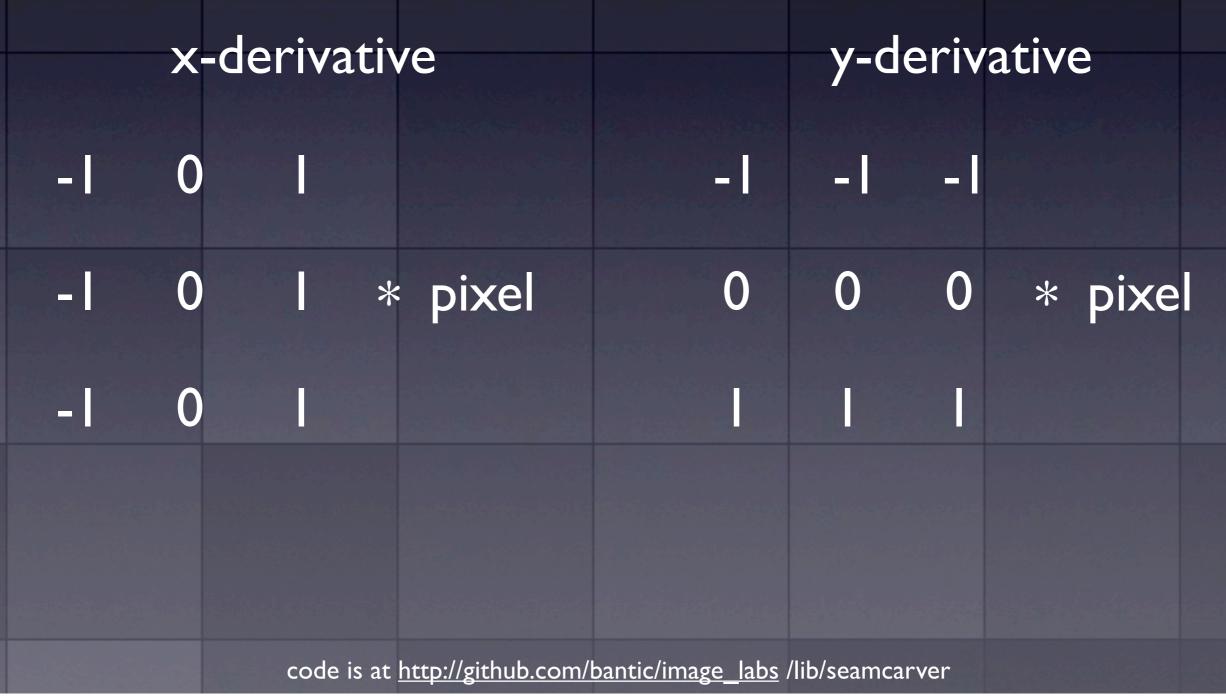
## Edge Detection

Find places where the pixels change suddenly

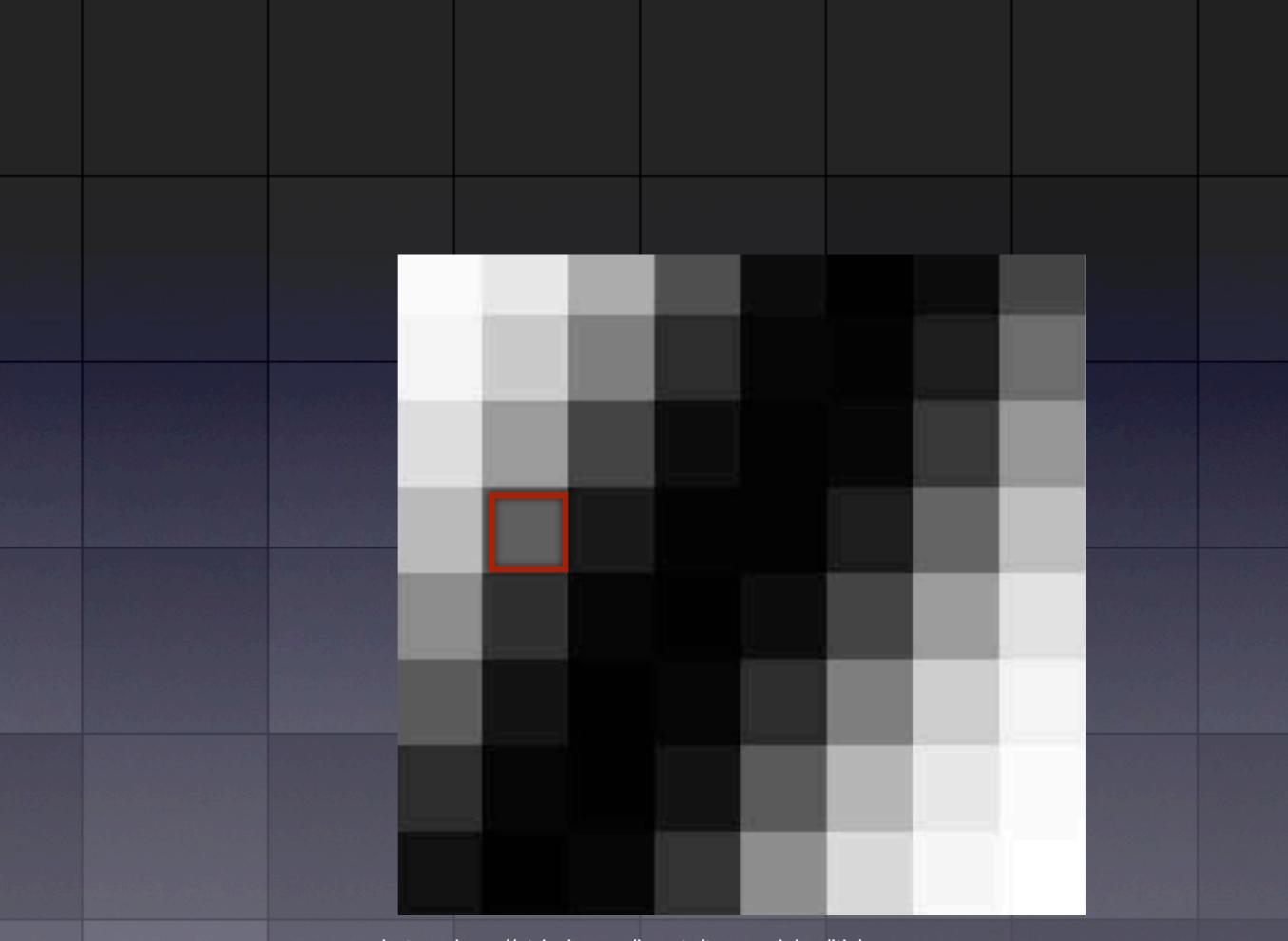
- Look in the X- and Y- directions
- Put those together = edges!

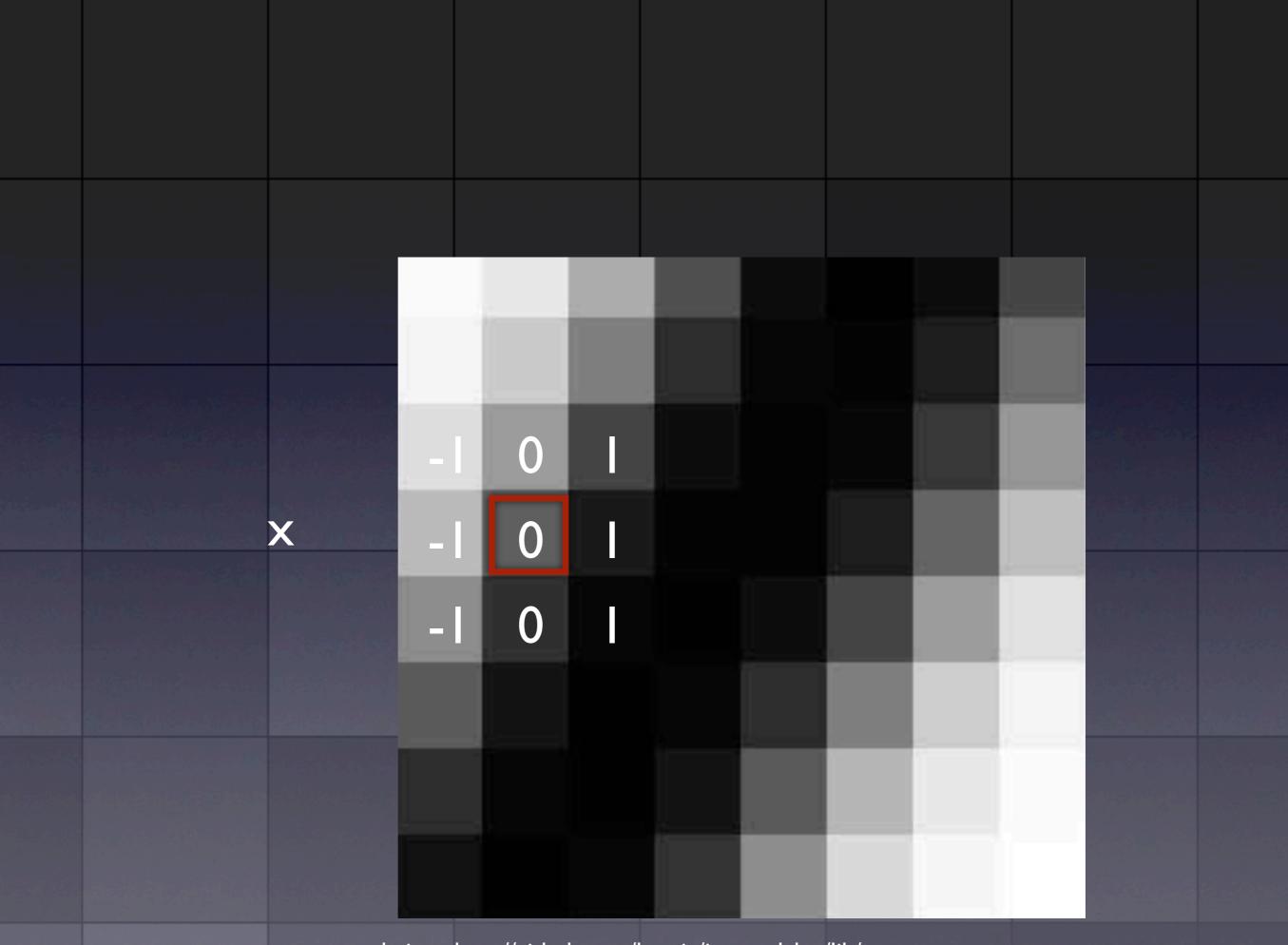
code is at http://github.com/bantic/image\_labs /lib/seamcarver

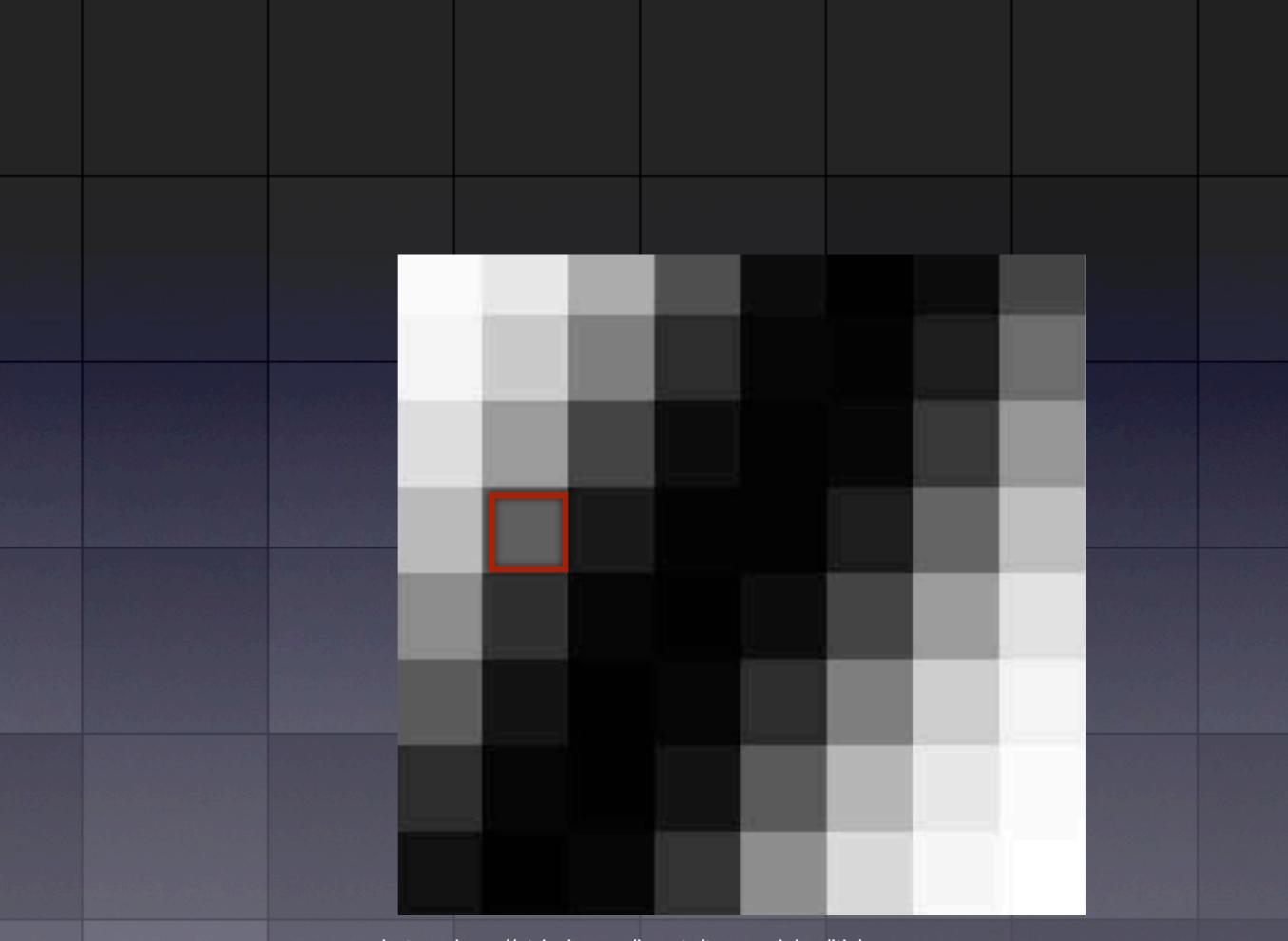


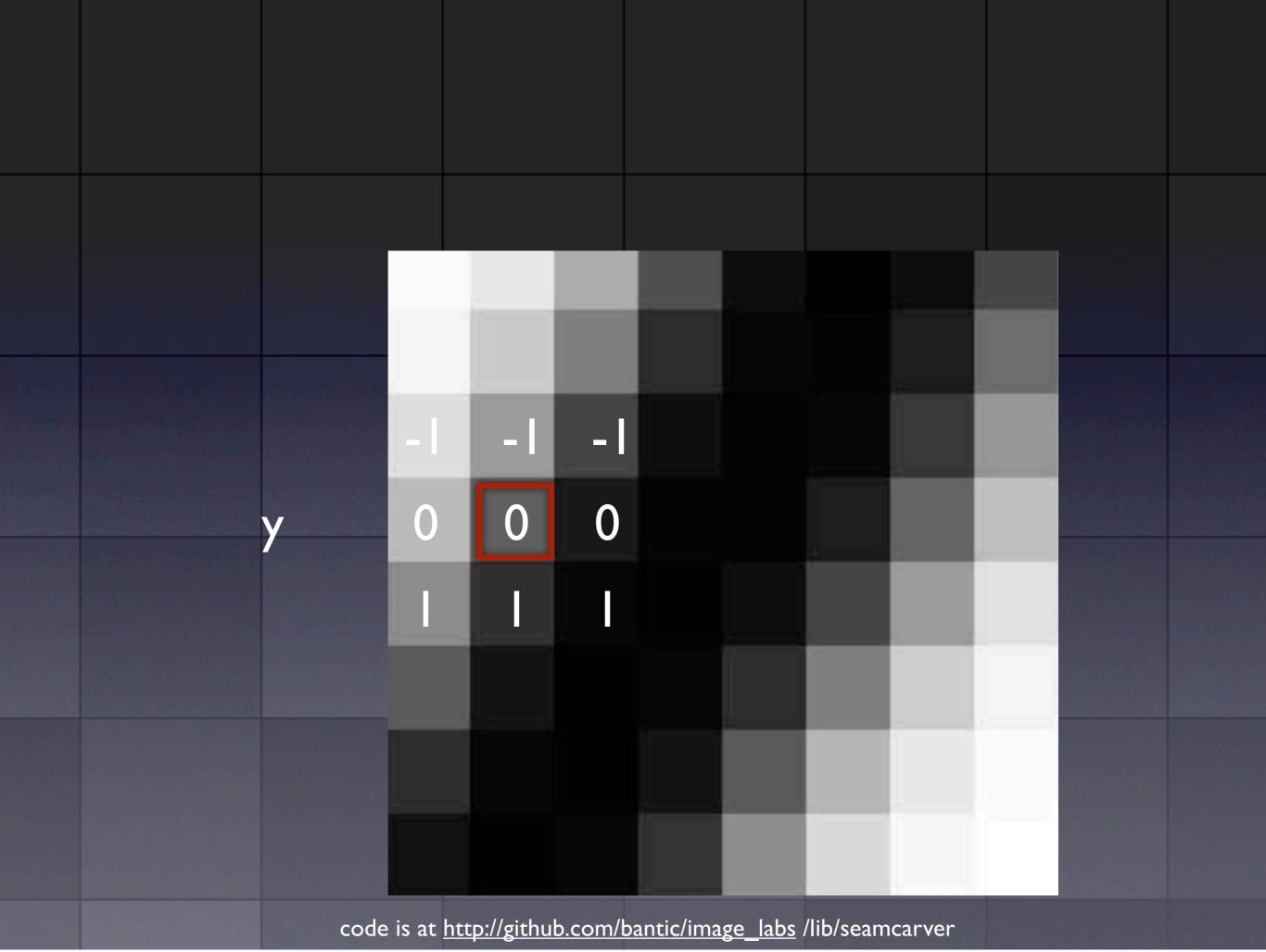




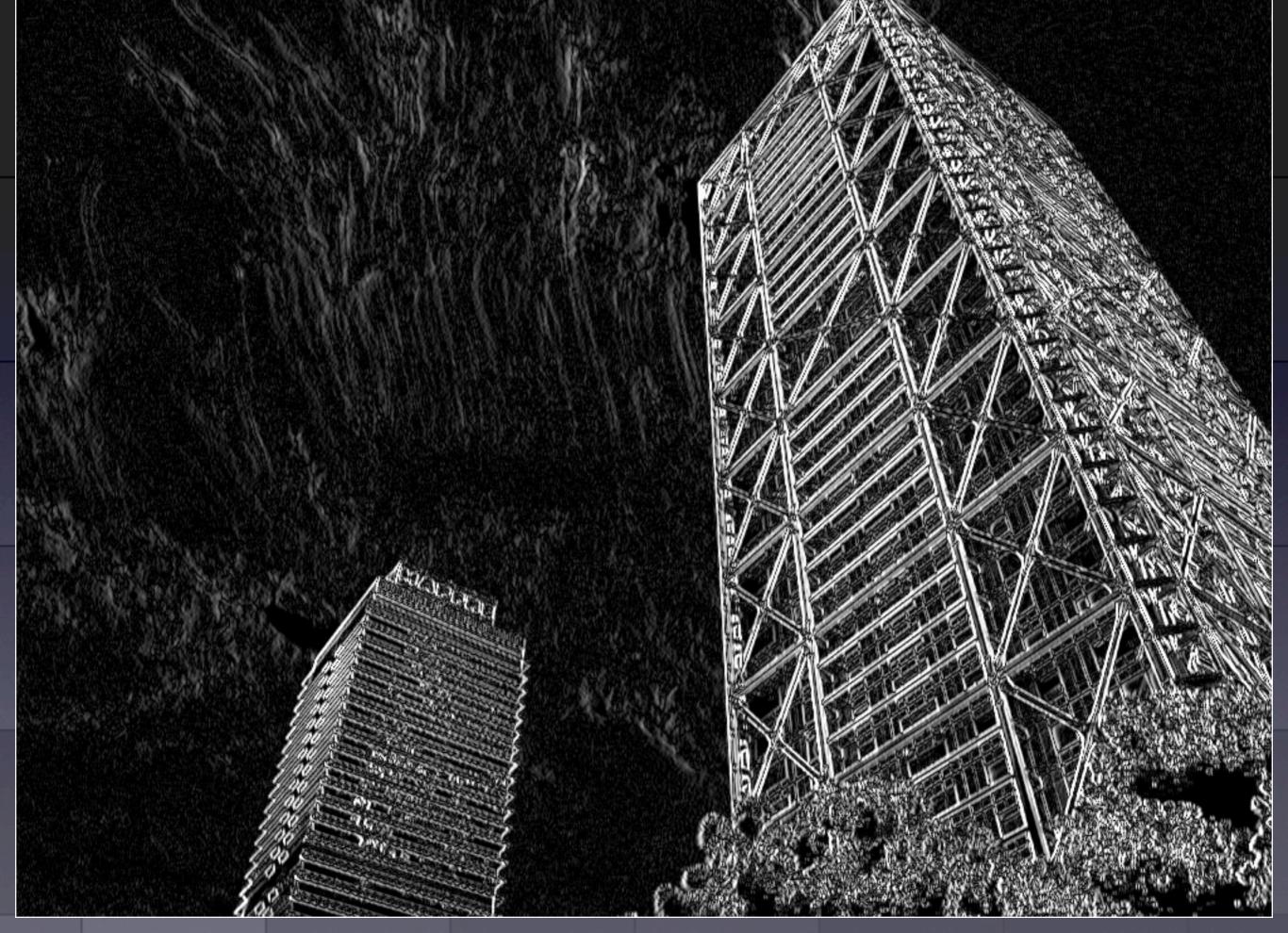




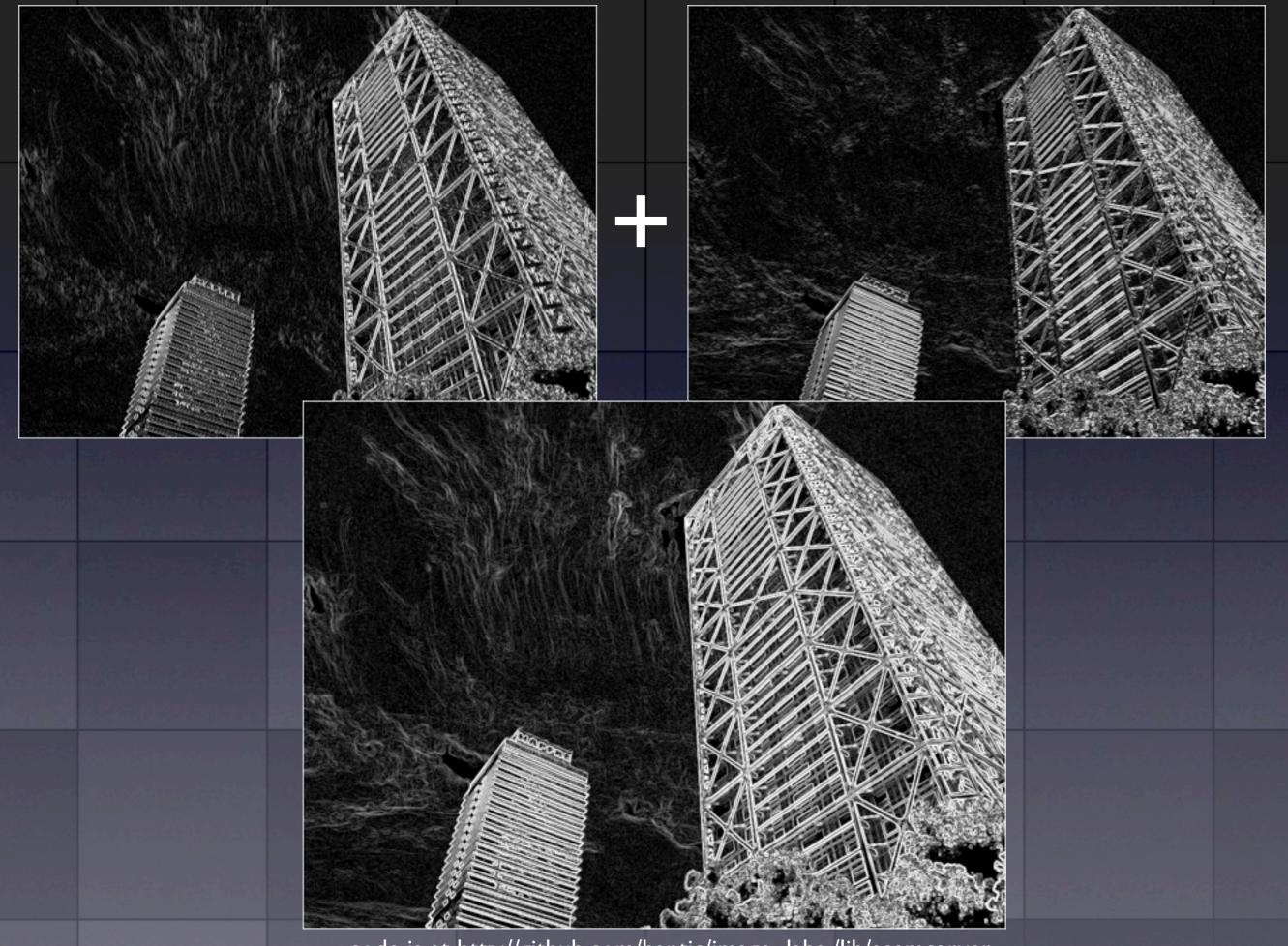










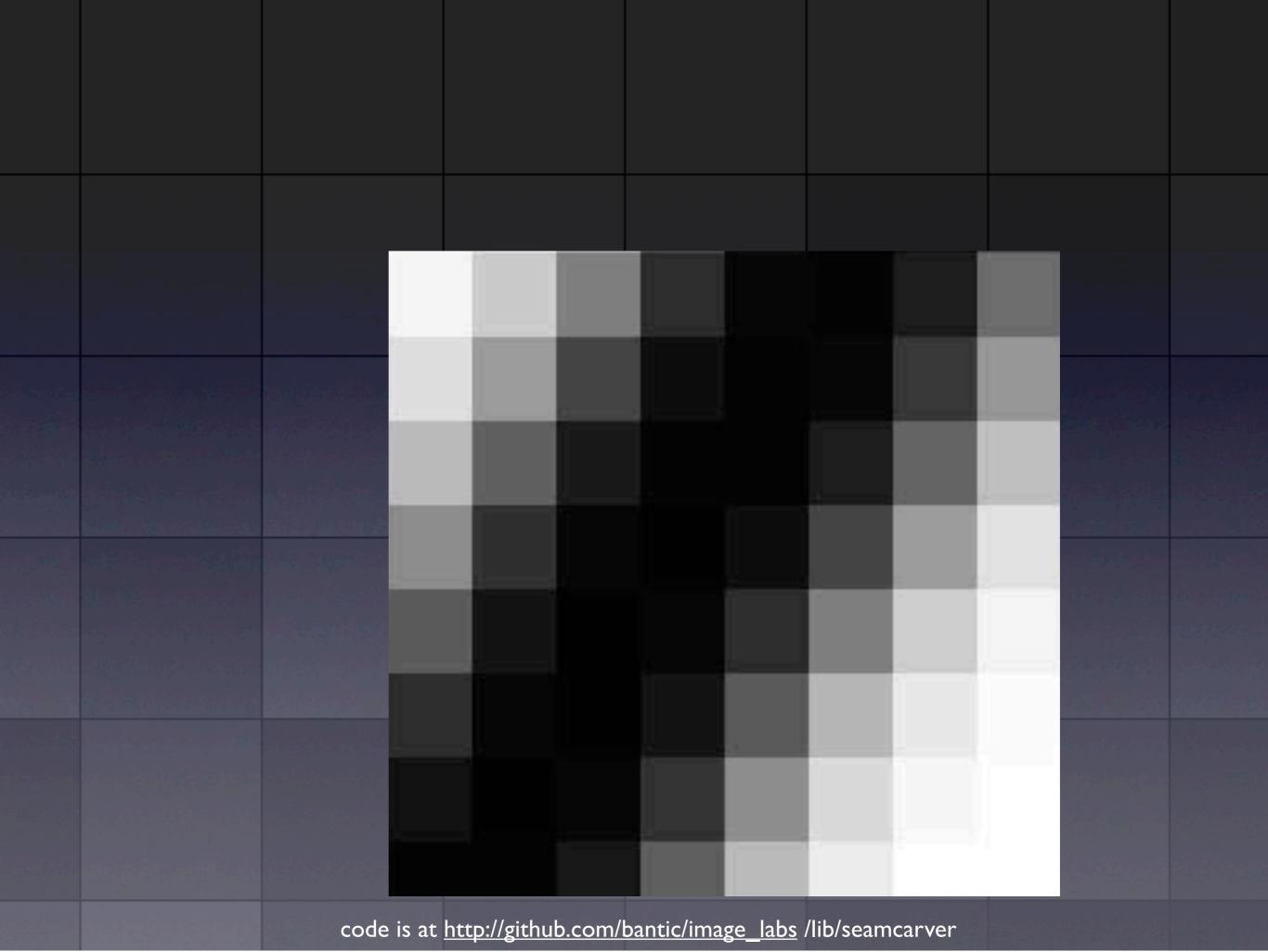


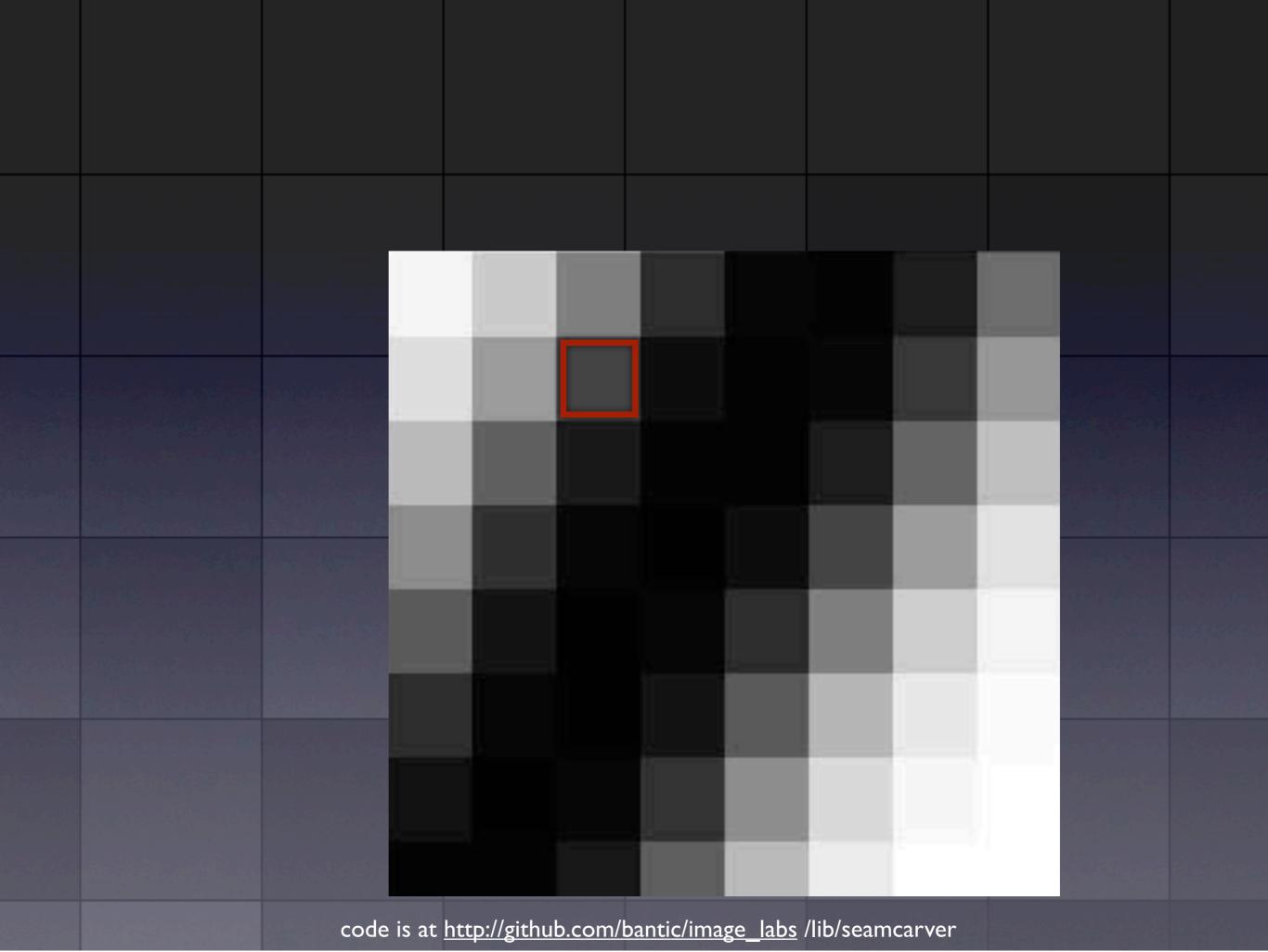
## energy map

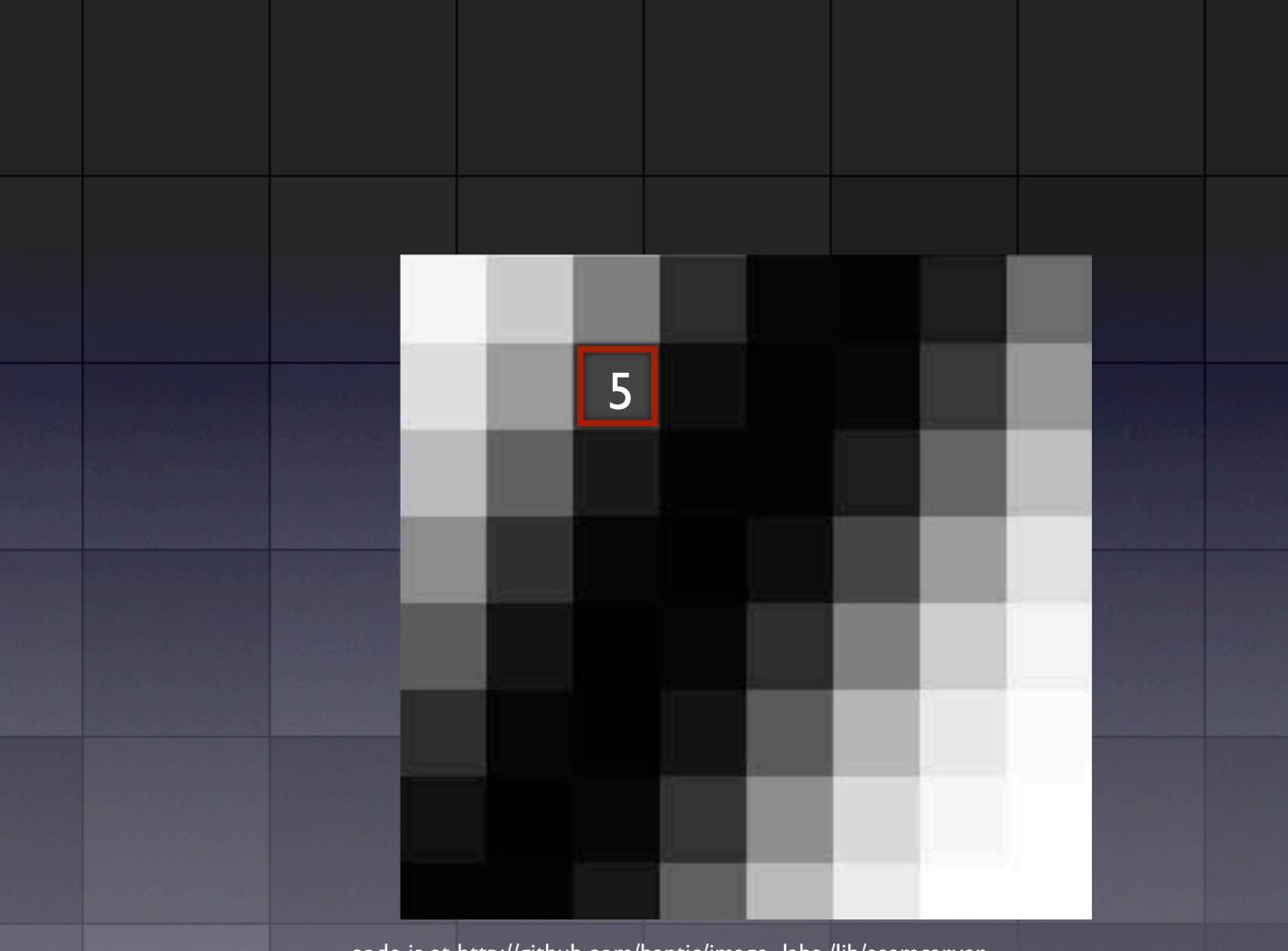
find the cheapest 8-connected path

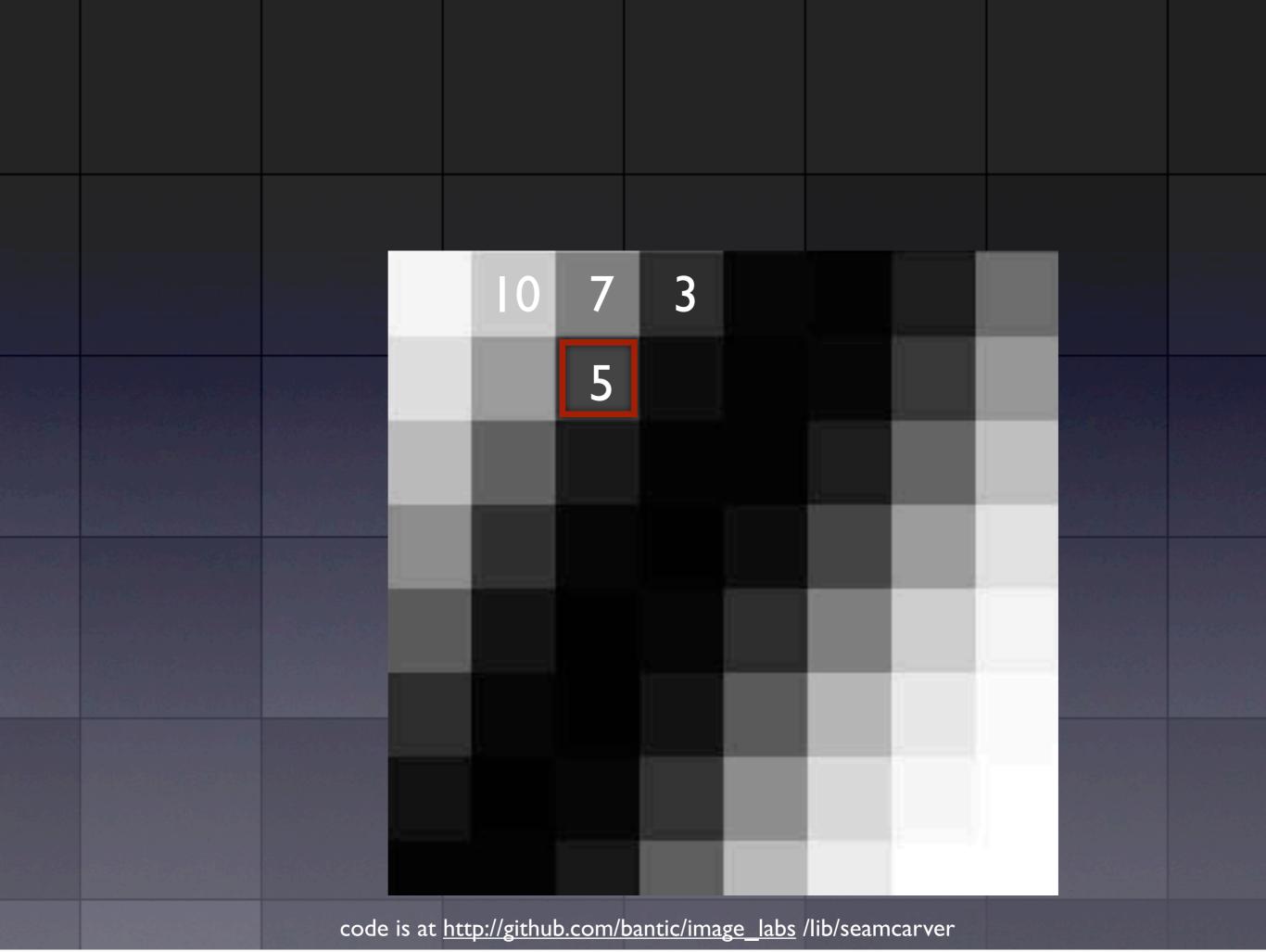
- each pixel is assigned an energy value based on the cheapest energy path to that pixel
- dynamically calculate values pixel-by-pixel in a row
- repeat with the next row

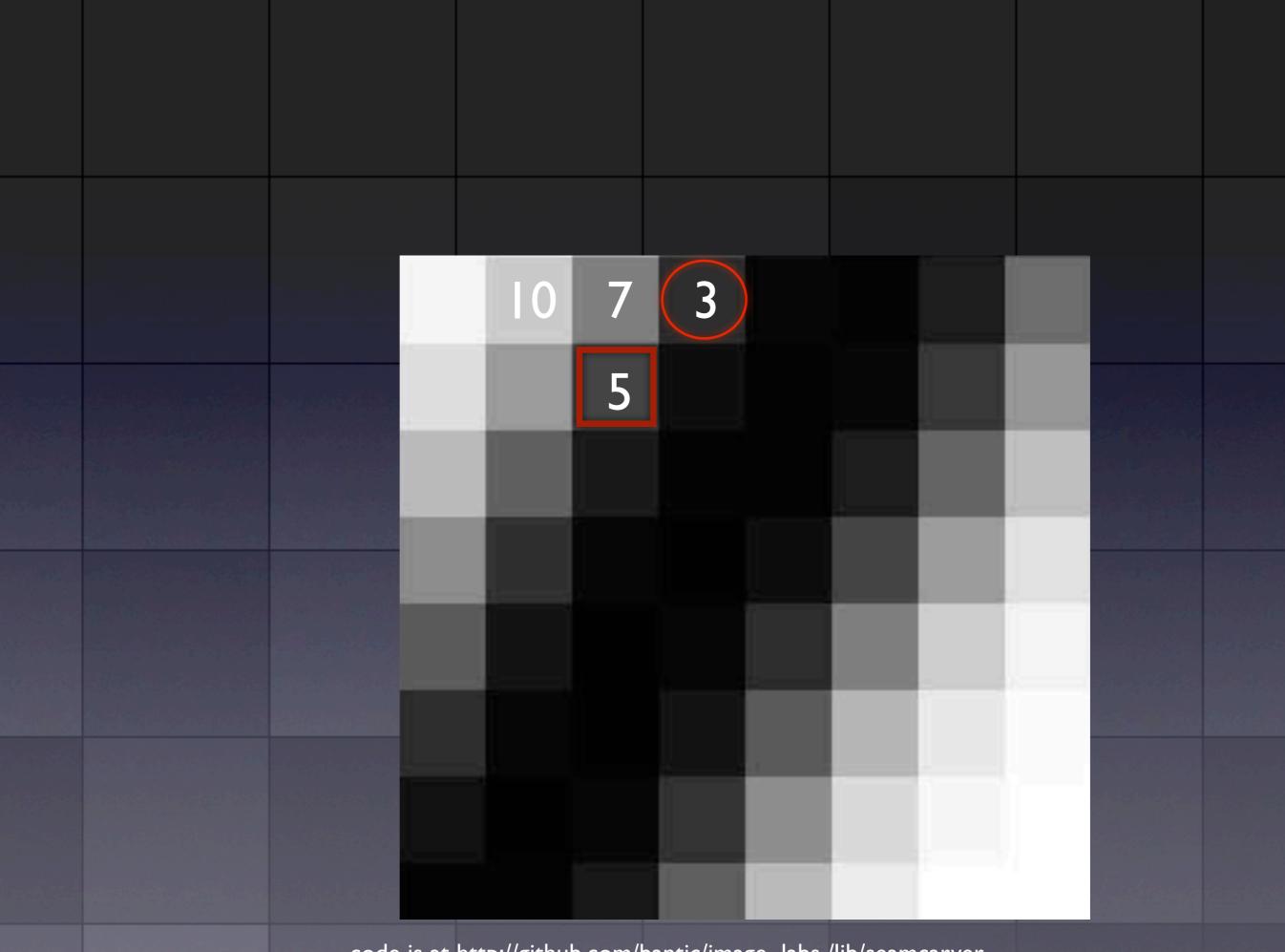
code is at <a href="http://github.com/bantic/image\_labs">http://github.com/bantic/image\_labs</a> /lib/seamcarver



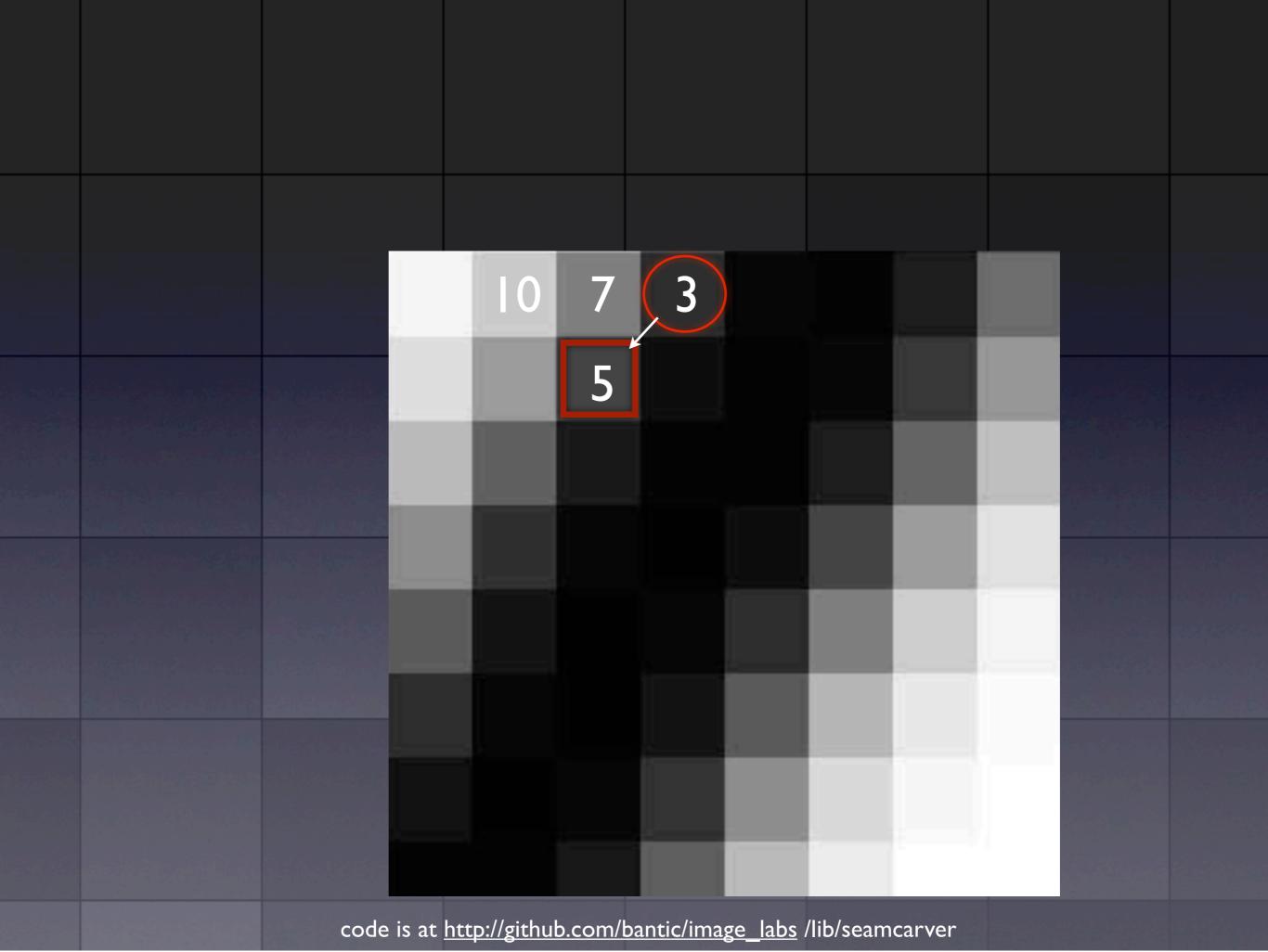


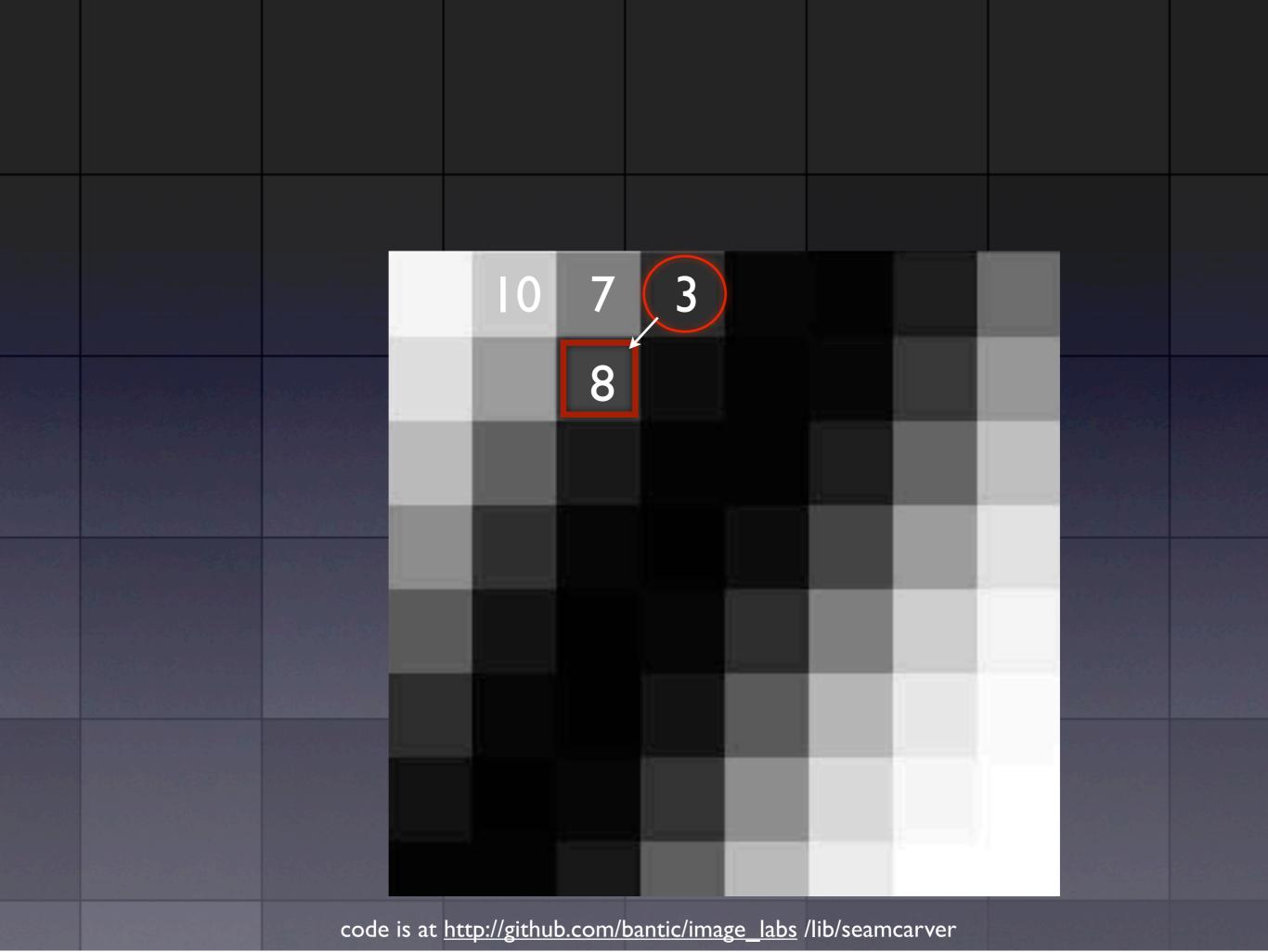






code is at http://github.com/bantic/image\_labs /lib/seamcarver



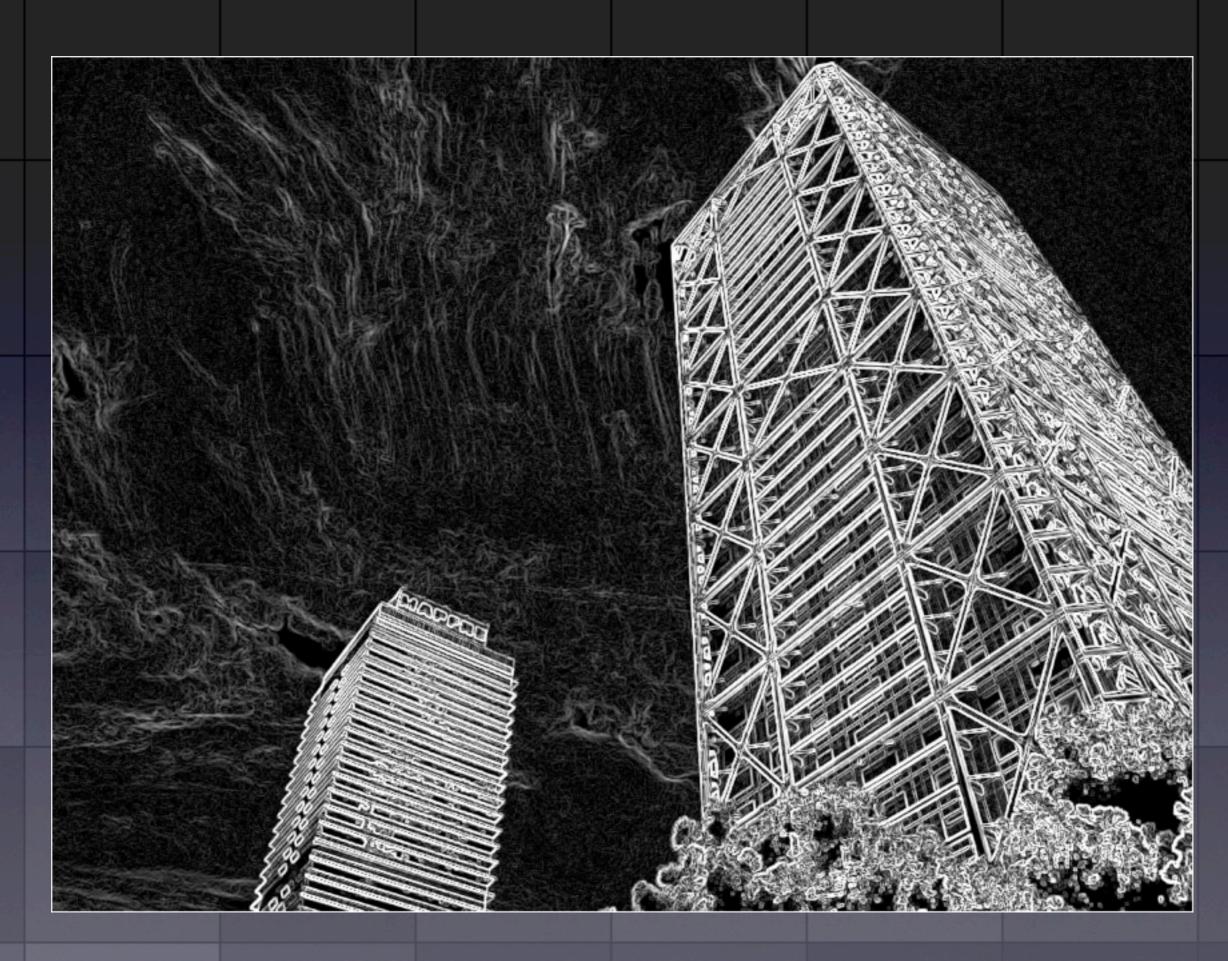


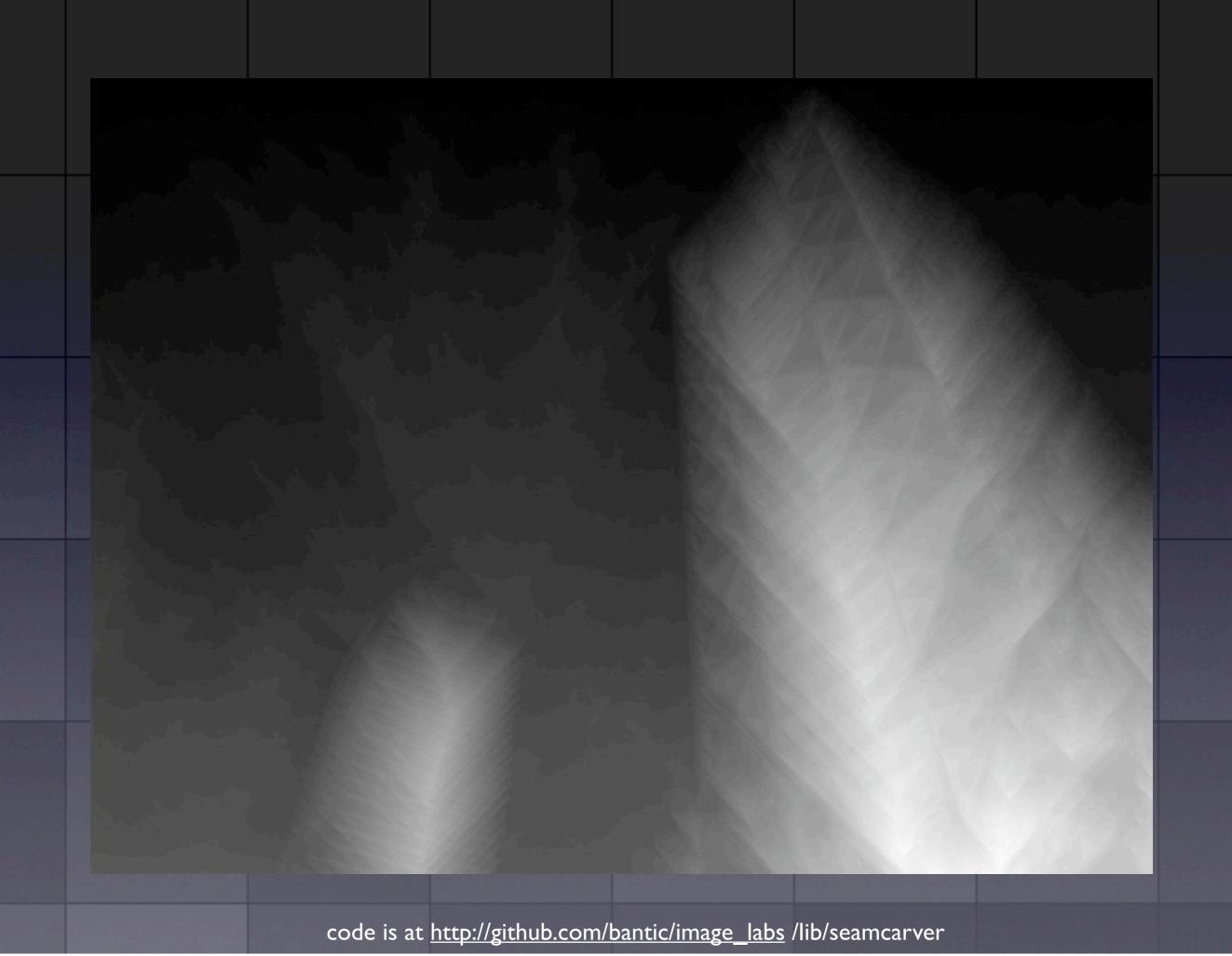
### calculate energy map

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
             # right edge, only two pixels below
  else
    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 <a href="http://github.com/bantic/image_labs">http://github.com/bantic/image_labs</a> /lib/seamcarver
```

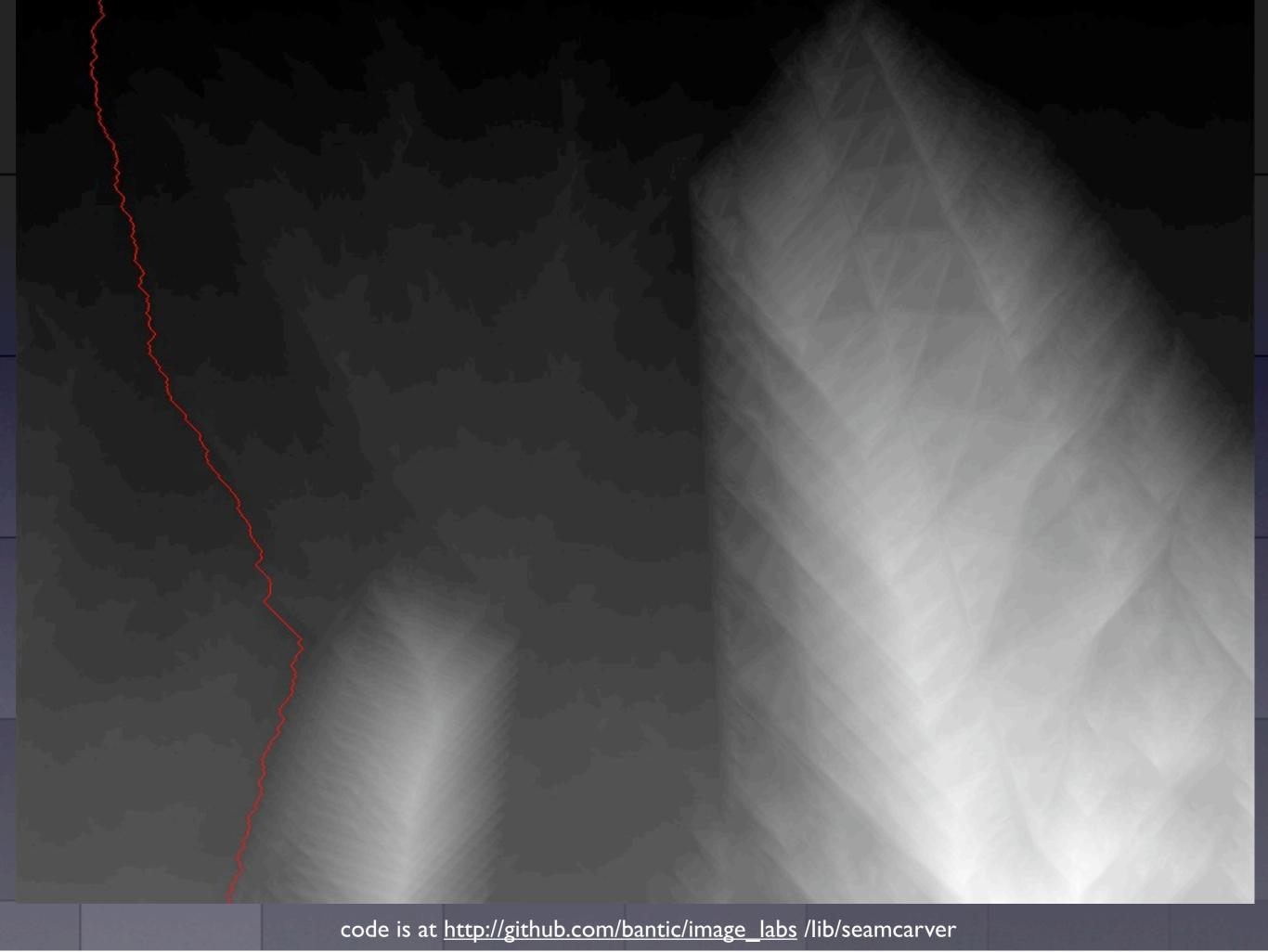


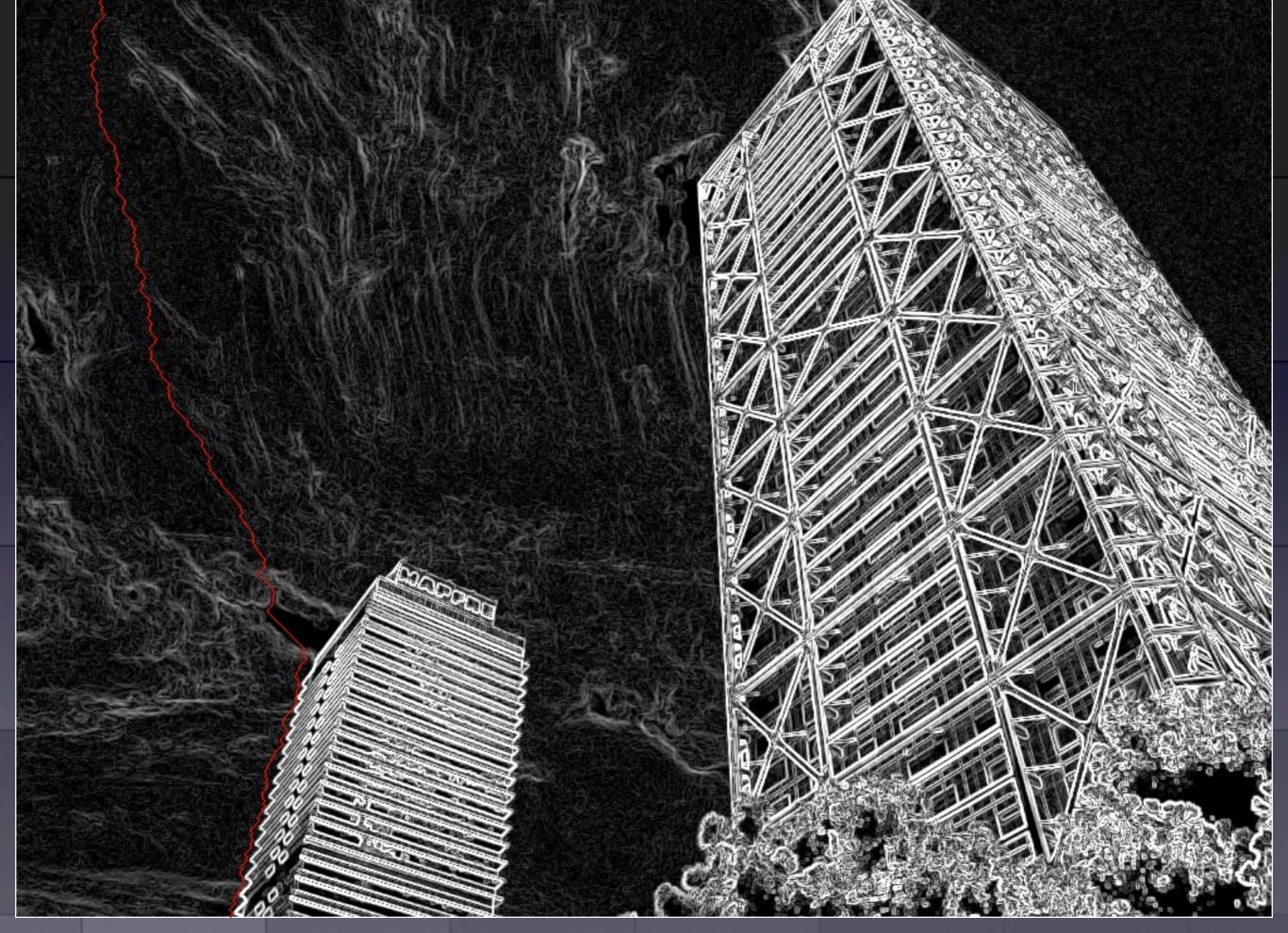




# find seam

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

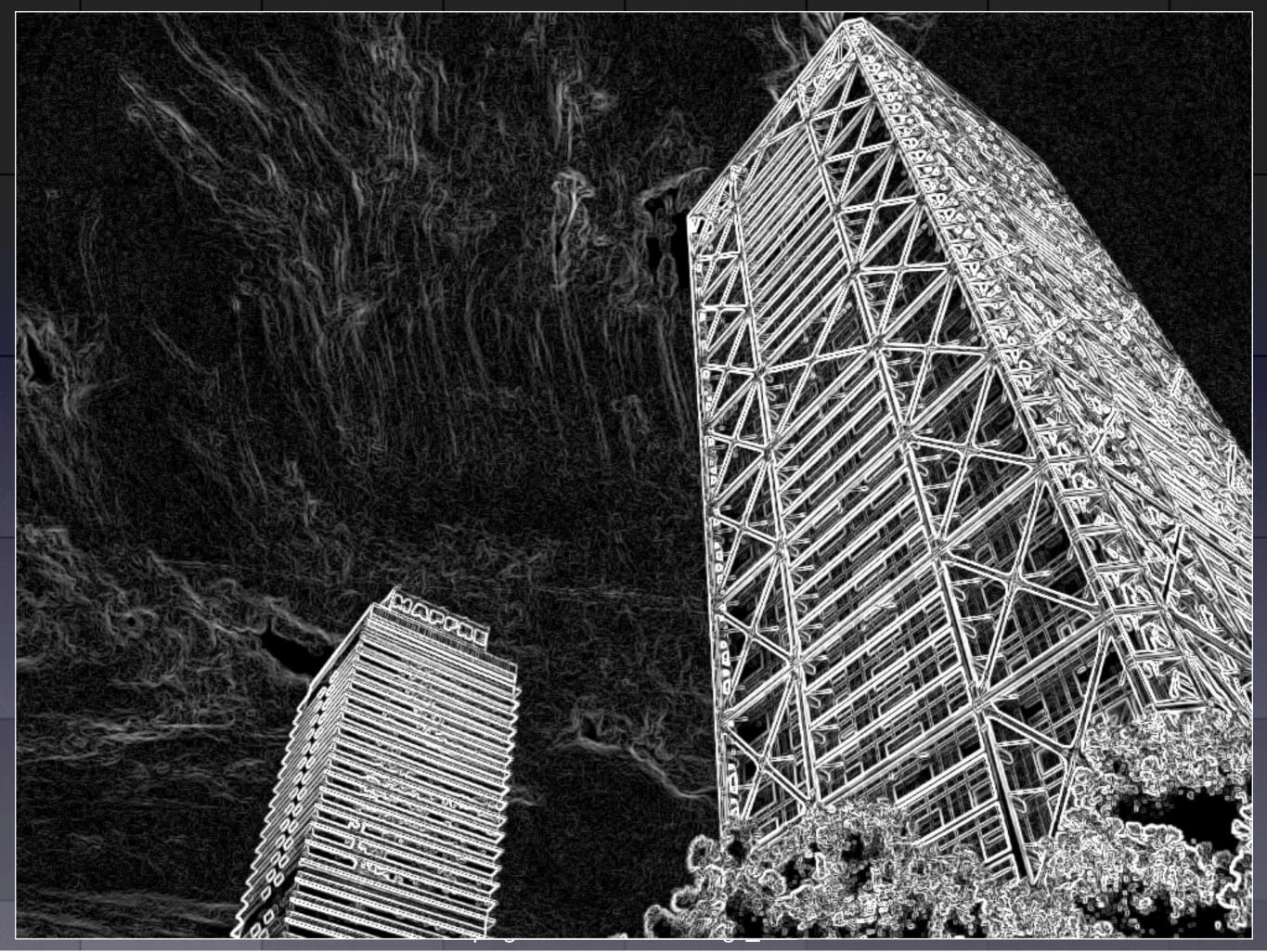


















RMagick
PhotoMosaic
SeamCarving
Face Detection with OpenCV

# 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-opency and http://github.com/bantic/image\_labs => /lib/facer

## Face Detection

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

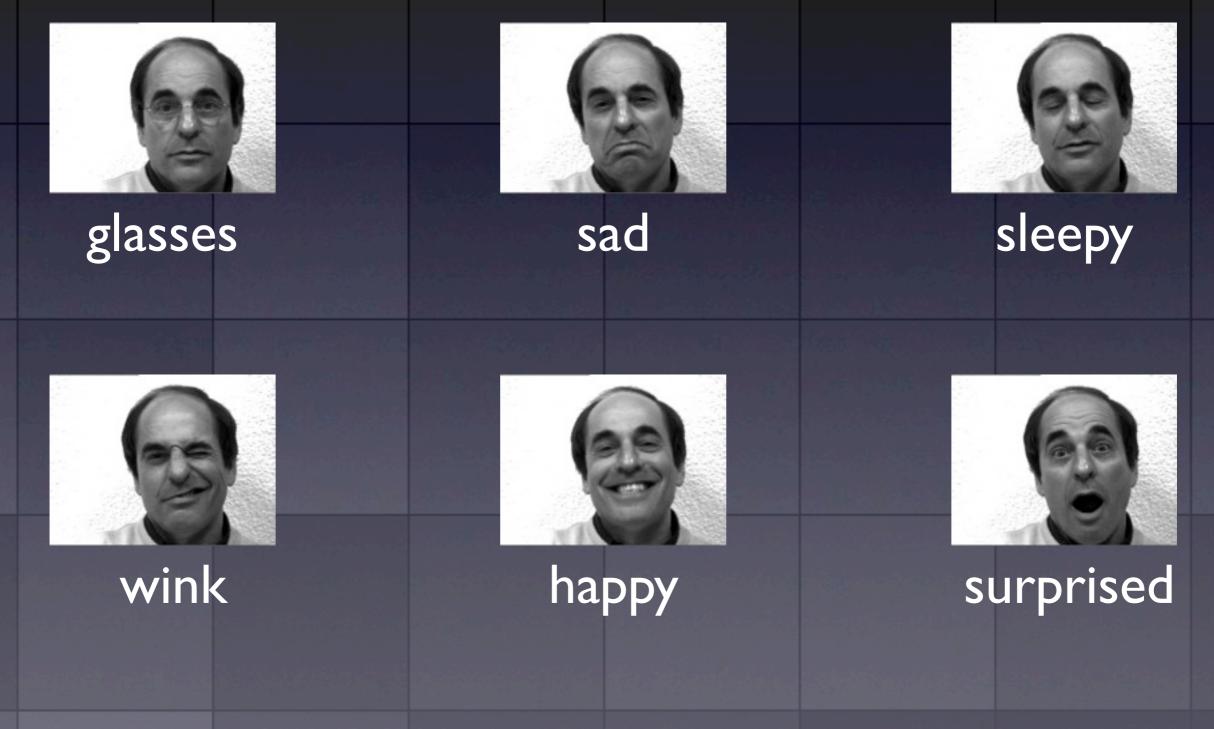
code is at <a href="http://github.com/bantic/ruby-opency">http://github.com/bantic/image\_labs</a> /lib/facer

## Face Detection

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

code is at <a href="http://github.com/bantic/ruby-opency">http://github.com/bantic/image\_labs</a> /lib/facer

## Face Detection



code is at <a href="http://github.com/bantic/ruby-opency">http://github.com/bantic/image\_labs</a> => /lib/facer

\$ bin/rmagicksh

>> require 'opencv'

>> cv\_img = OpenCV::lpllmage.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 <u>http://github.com/bantic/ruby-opencv</u> and <u>http://github.com/bantic/image\_labs</u> => /lib/facer

\$ bin/rmagicksh

>> require 'opencv'

>> cv\_img = OpenCV::lpllmage.load("images/face.ipg")

>> 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 <a href="http://github.com/bantic/ruby-opency">http://github.com/bantic/image\_labs</a> => /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 IrectI
    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::0verComposite0p)
  end
 rm_img
```

code is at <a href="http://github.com/bantic/ruby-opency">http://github.com/bantic/image\_labs</a> => /lib/facer

Tuesday, May 12, 2009

end

#### >> 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::0verCompositeOp)
end
rm_img
end
```

code is at <u>http://github.com/bantic/ruby-opencv</u> and <u>http://github.com/bantic/image\_labs</u> => /lib/facer

#### >> Facer.put\_funny\_hat\_on\_celebrity



code is at <a href="http://github.com/bantic/ruby-opency">http://github.com/bantic/image\_labs</a> => /lib/facer

#### Thank you

### http://github.com/bantic/image\_labs

#### http://github.com/bantic/ruby-opencv

### http://delicious.com/eoligarry/image\_labs