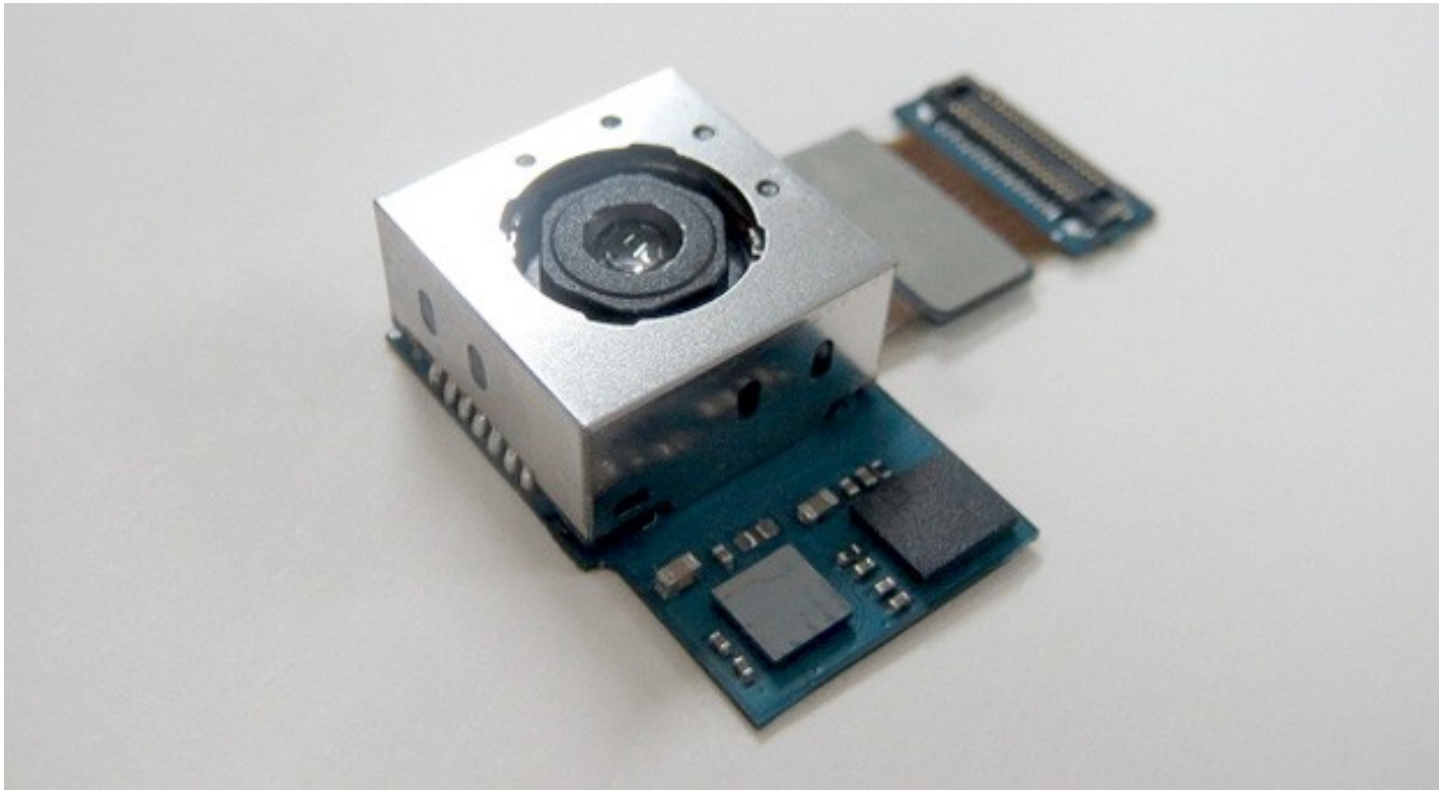
Seeing with clojure

Dave Snowdon
@davesnowdon

https://github.com/davesnowdon/cljex-2015-opencv

Most general purpose sensor



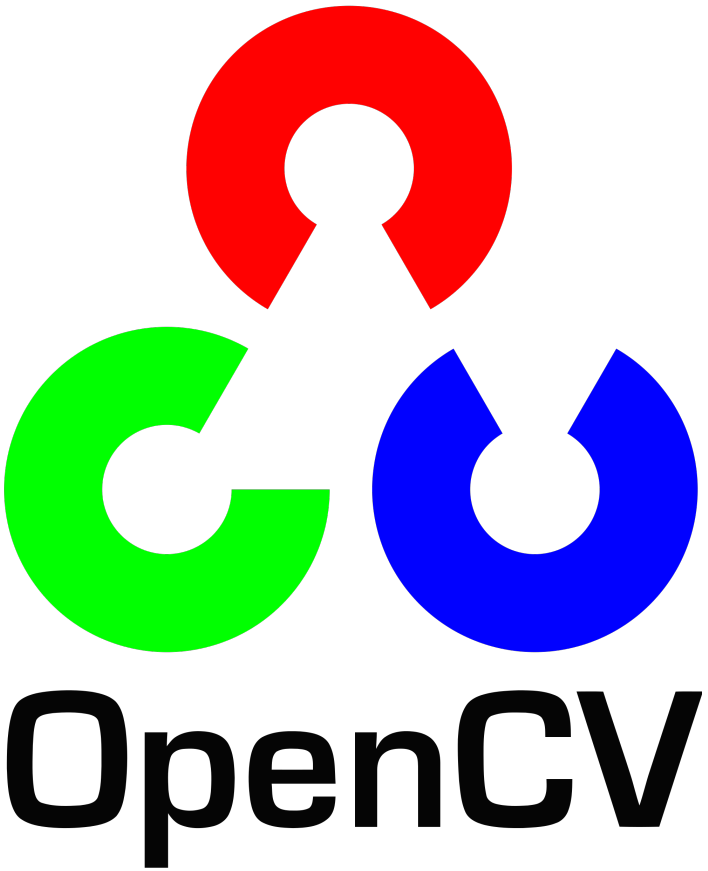
With great power comes...


name
using
specified
scale
better
pp
line
Object
Set
Training
samples
angle
See
Size
2001
two
also
Haar
Example
info
case
2
50
TBB
work
30
file
form
used
text
25
box
45
way
need
24
list
Viola
key
w
area
img
01
test
one
vec
use
Haarlike
ie
7
h
45°
Image
140
four
faces
positive
may
called
correspond
bg
Cascade
like
Boosted
sum
pixel
data
negative
Jones
sums
Fast
idea
every
range
rectangle
company
detected

This is hard

Can somebody please do this for me?

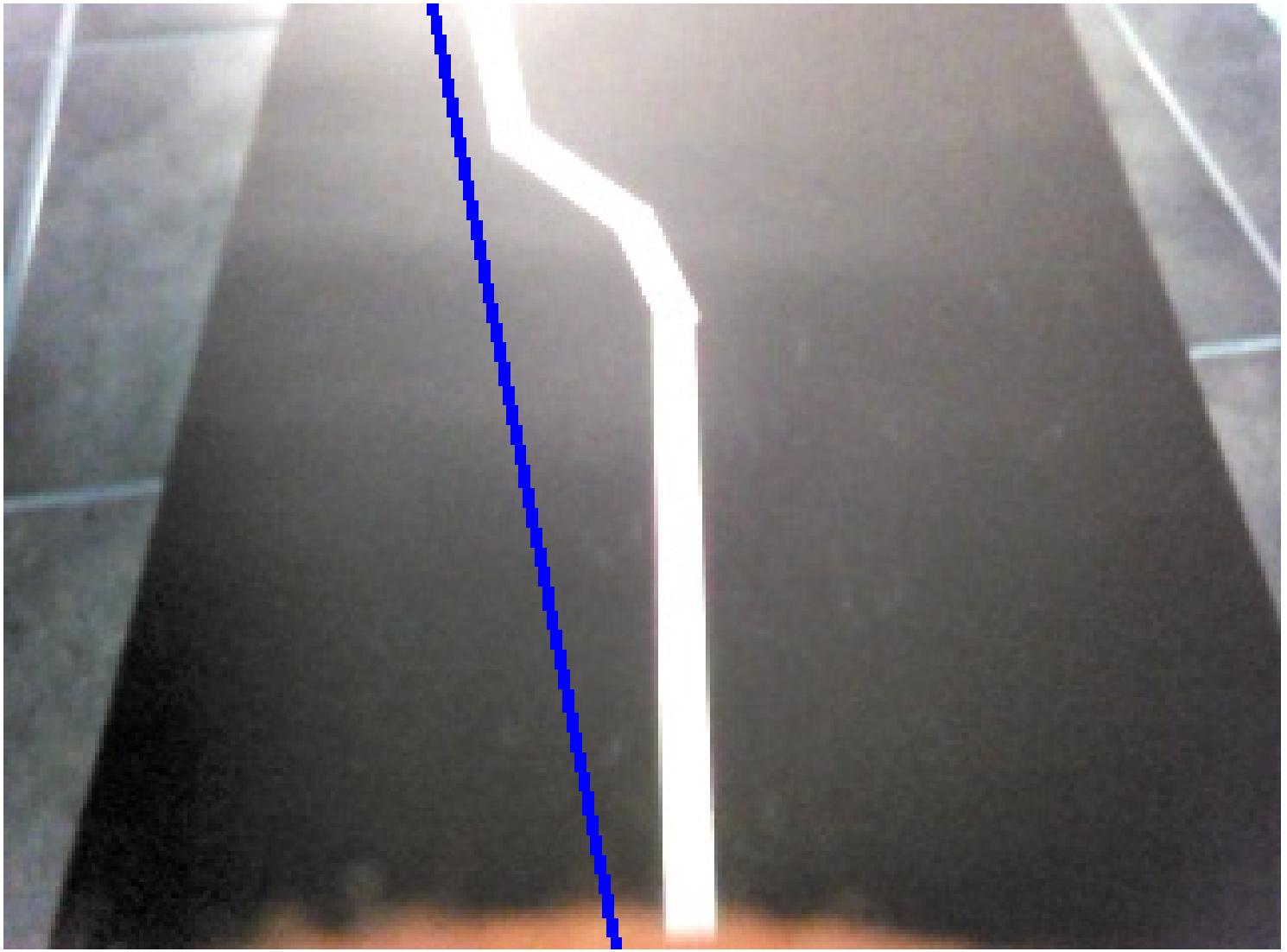
openCV



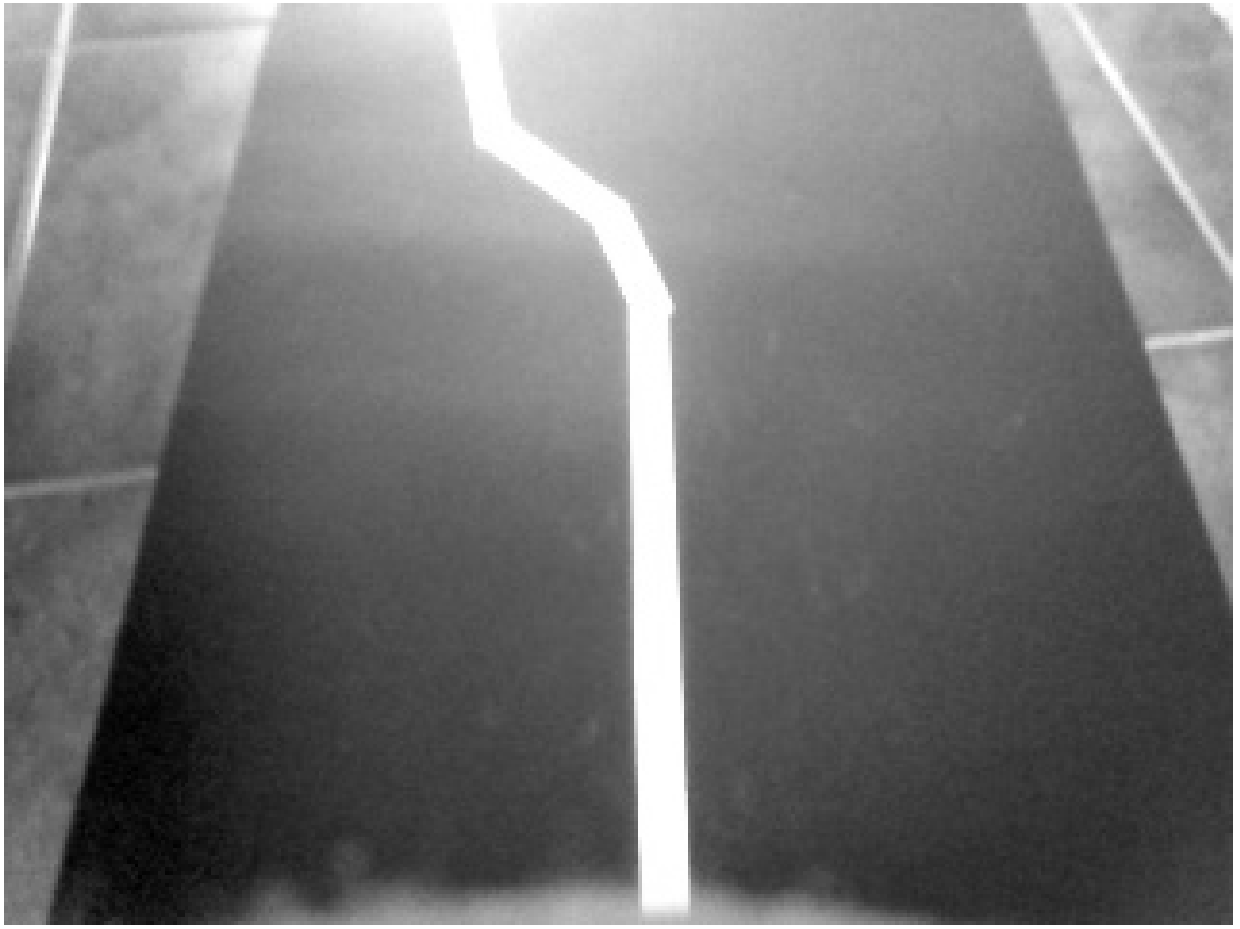
- Around since 1999
- Open source
- Multi-platform: windows, linux, mac, iOS, android, RaspberryPi
- C++, python, java

Line following

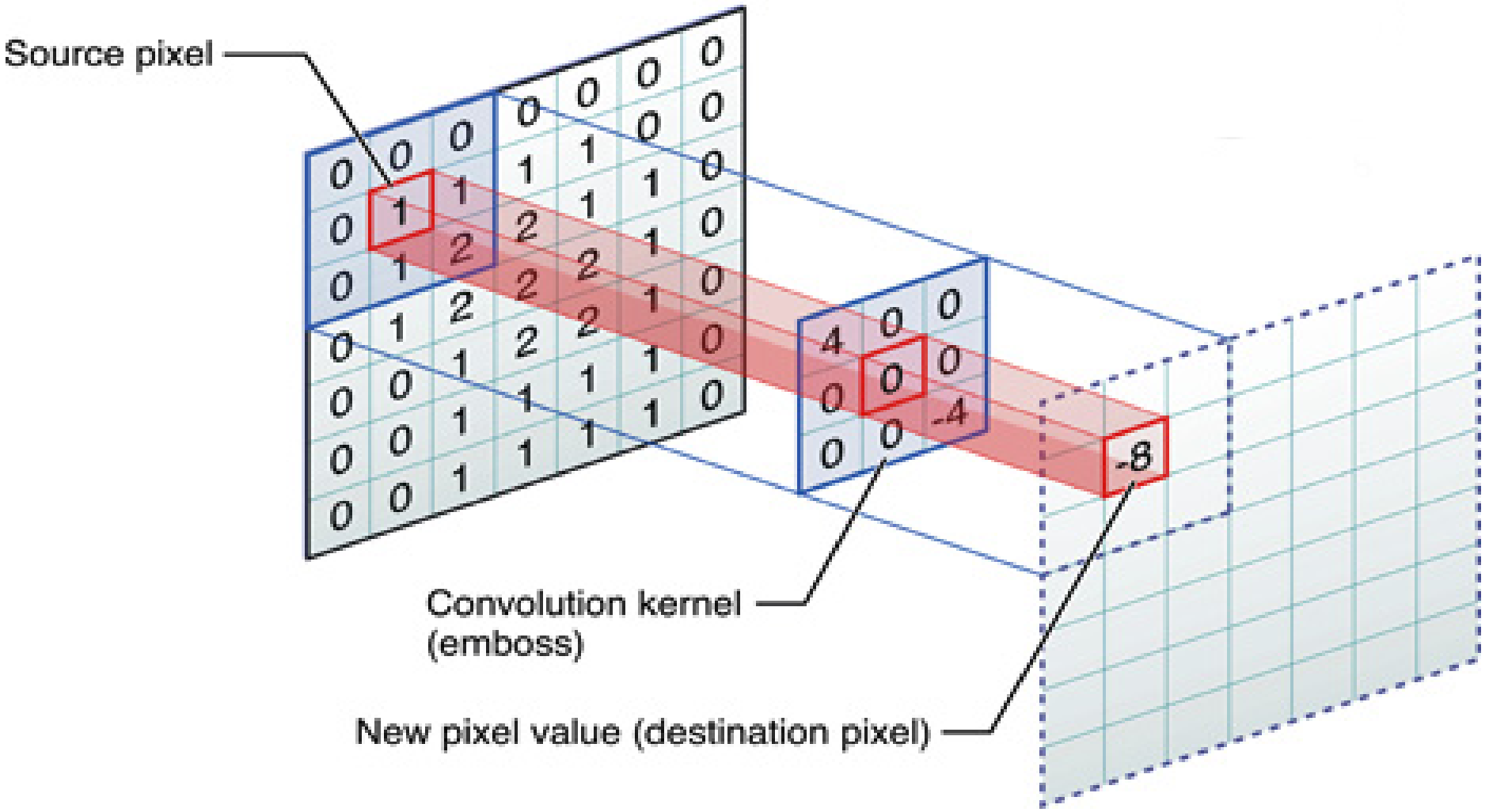
Demo: (show-line image)



(Imgproc/cvtColor src dest Imgproc/COLOR_BGR2GRAY)

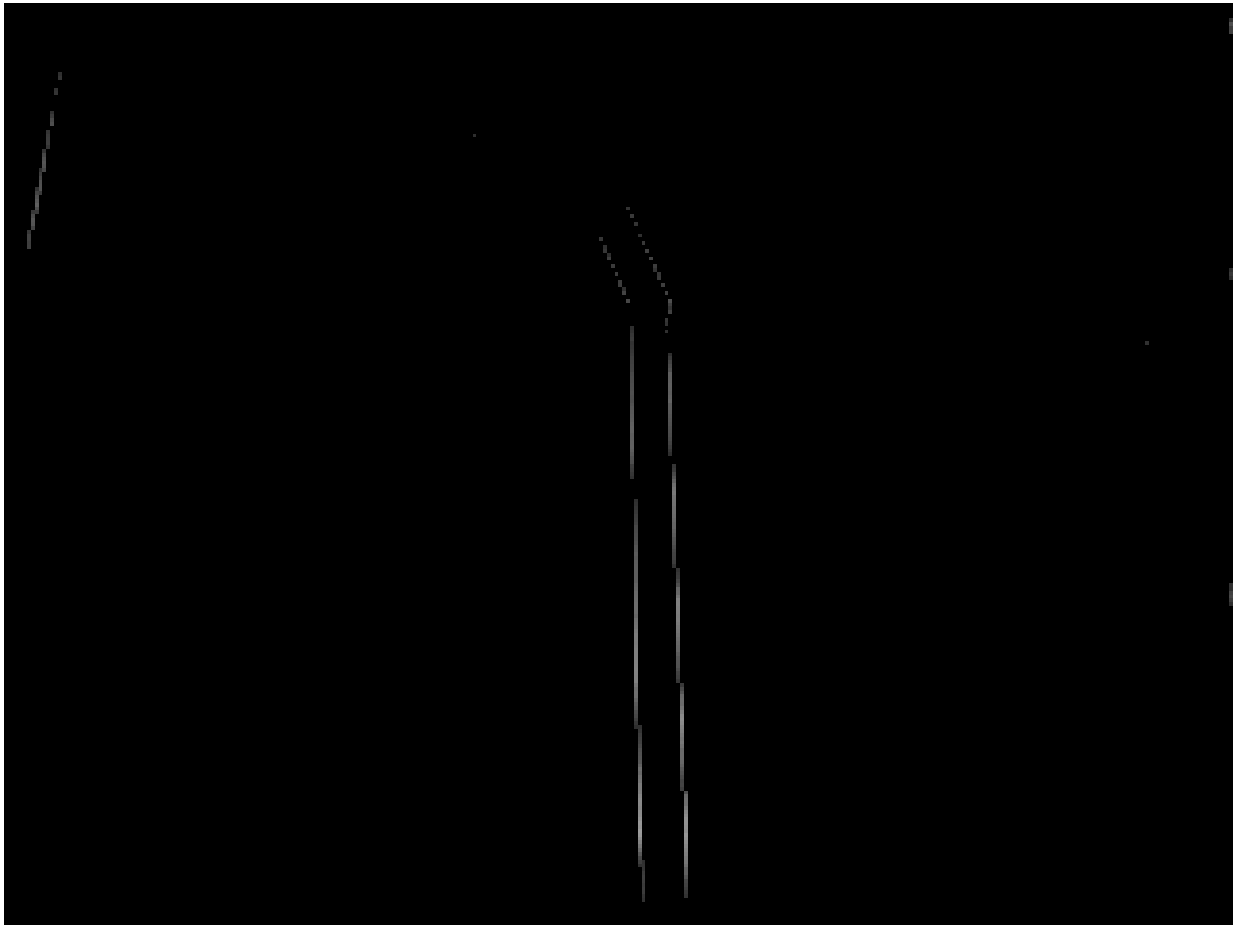


Convolution!



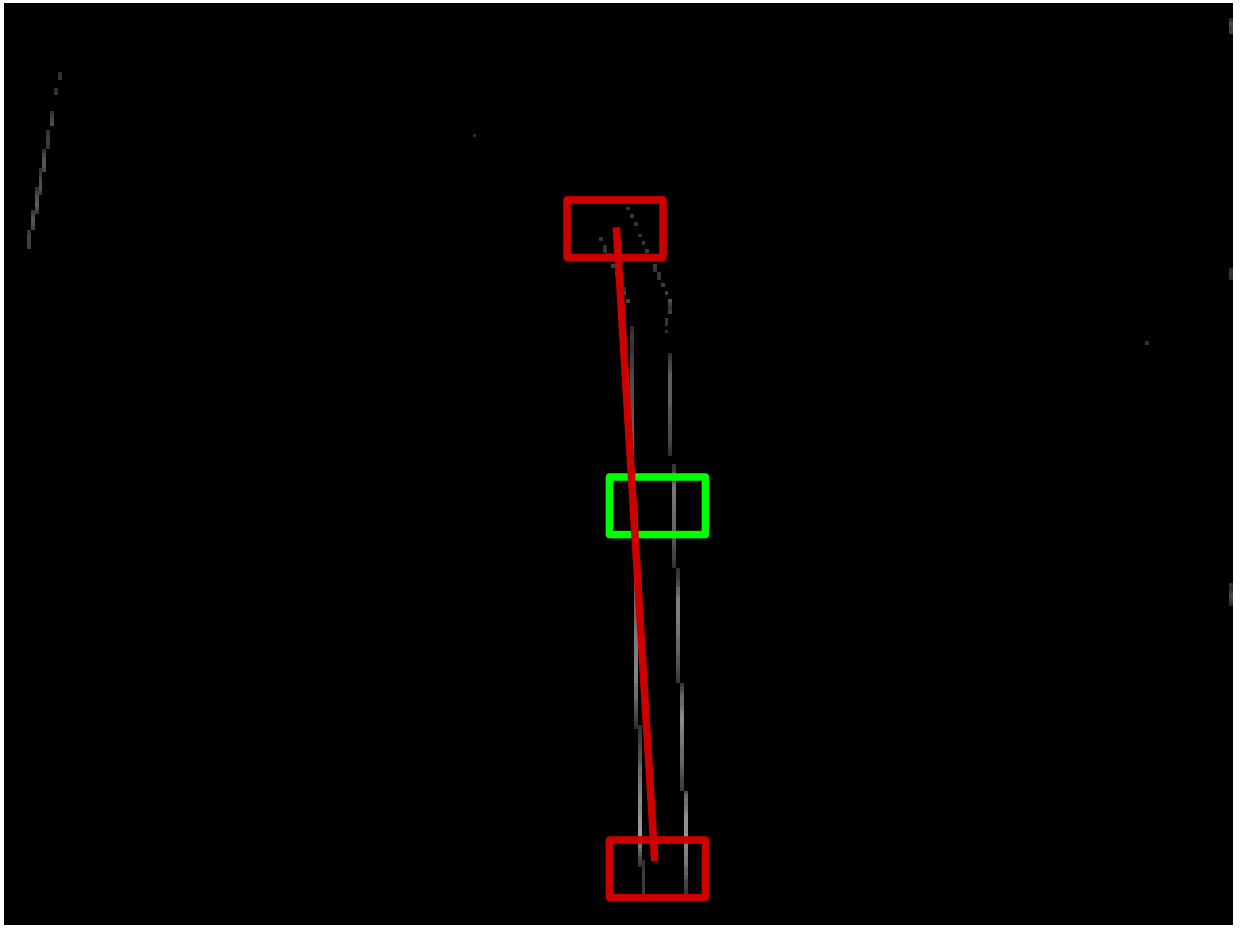
https://developer.apple.com/library/mac/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionOperations.html

(Imgproc/filter2D gray fil -1 kernel)
(Imgproc/threshold fil thresh 45.0 255
Imgproc/THRESH_TOZERO)

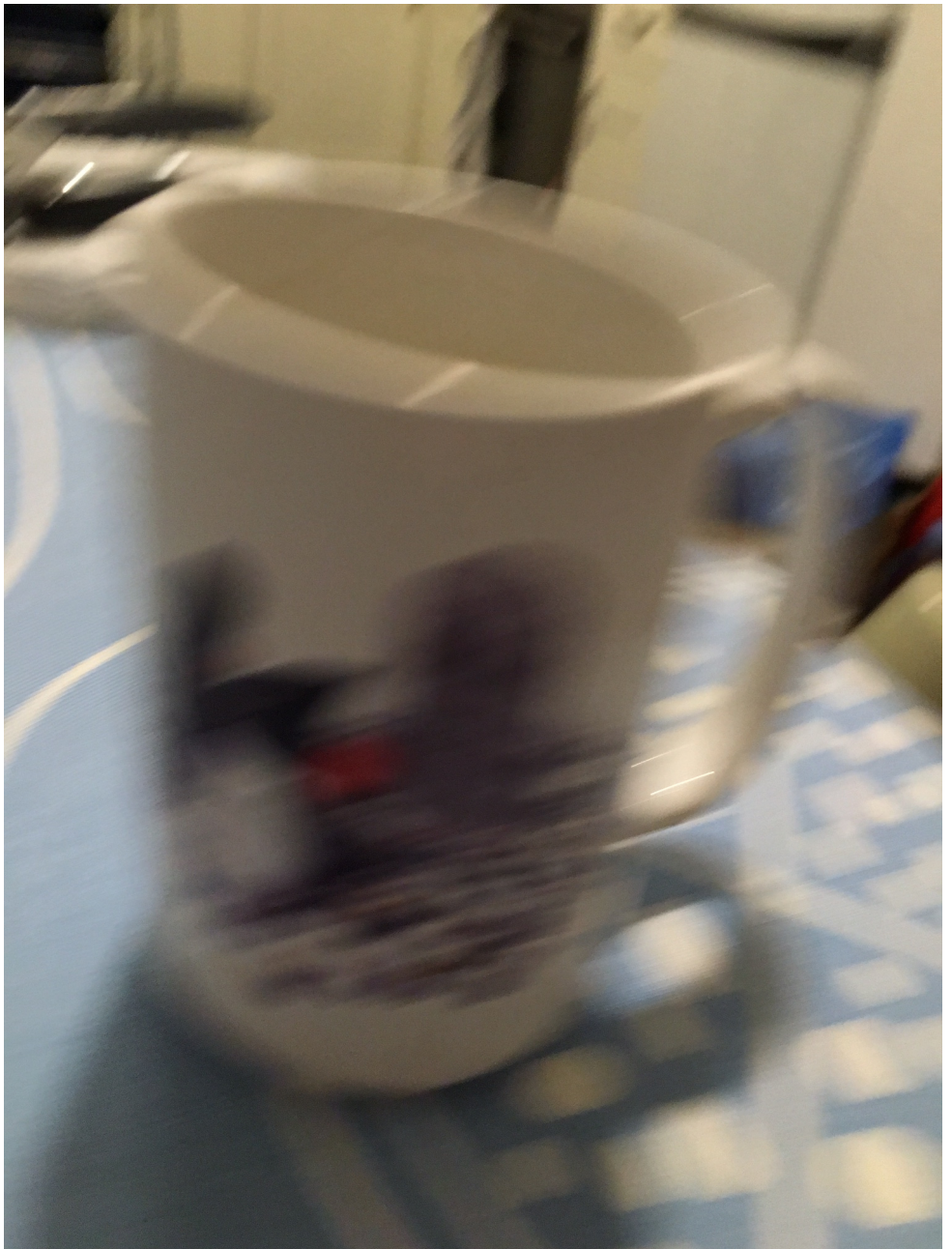


Kernel: [-1.0 2.0 -1.0]

Sample top, middle & bottom of line



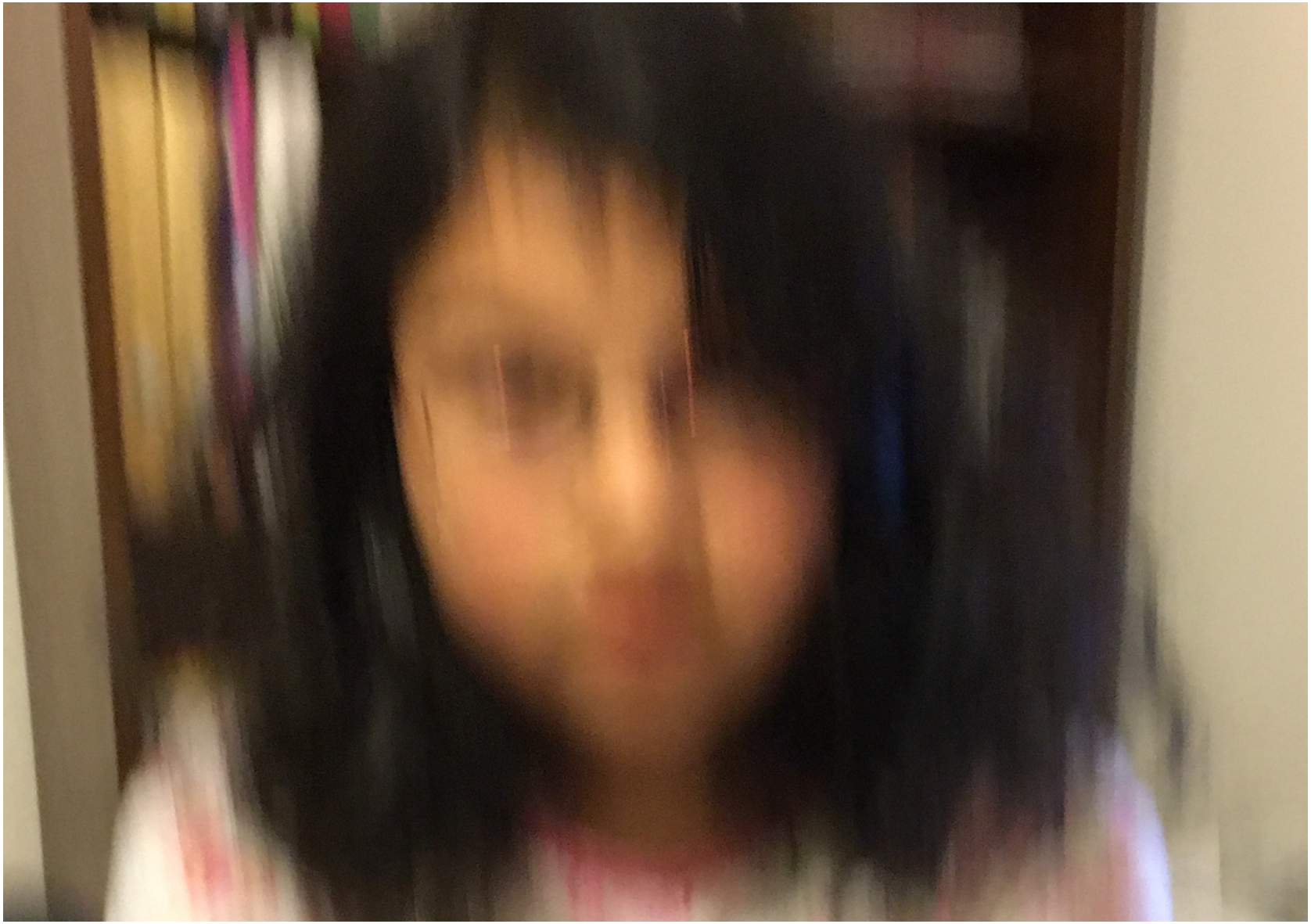
Blur Detection

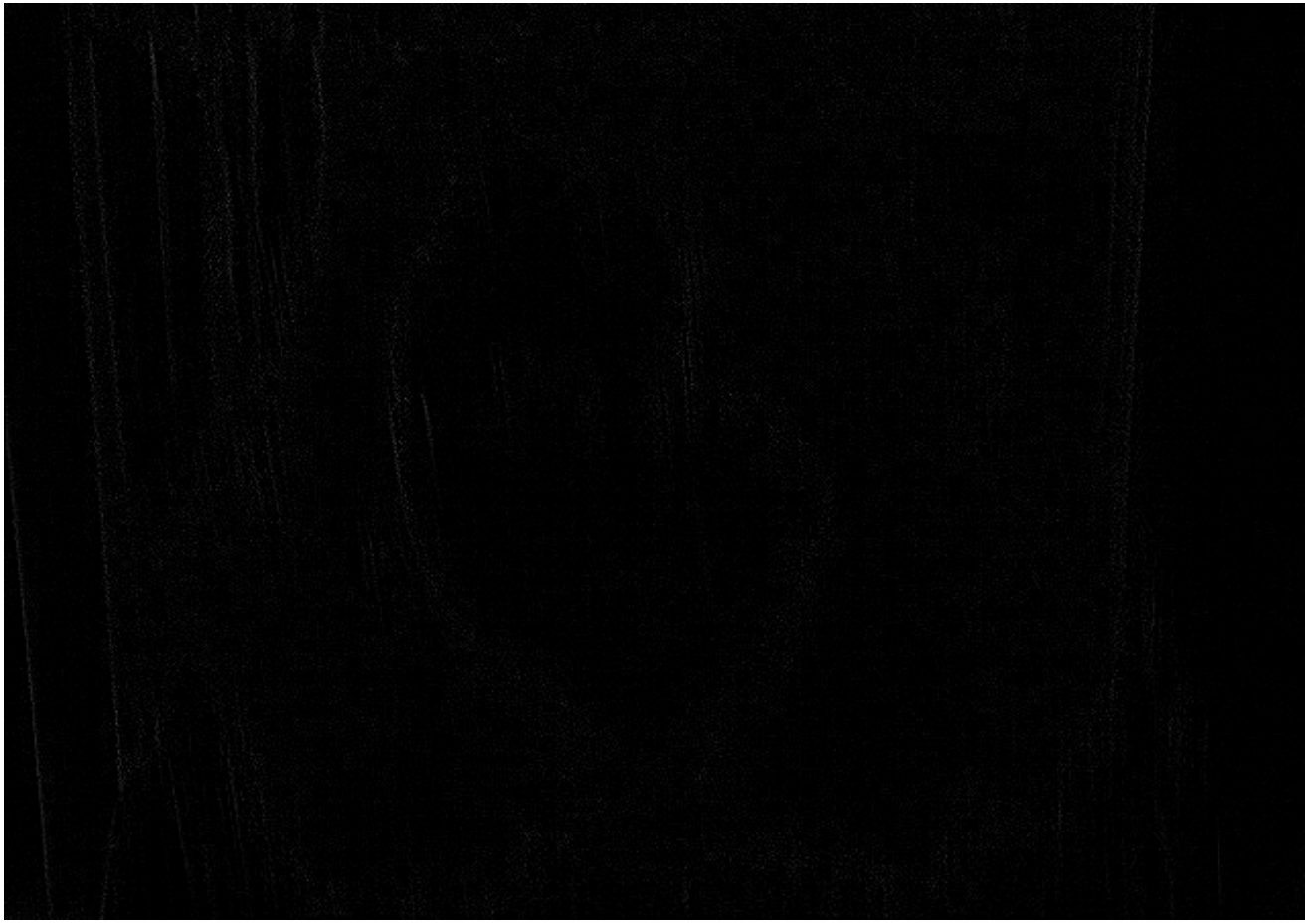


Demo: (is-image-blurry image)

(Imgproc/Laplacian src dest CvType/CV_64F)
(Core/meanStdDev dest mean stddev)

Variance: 195.92





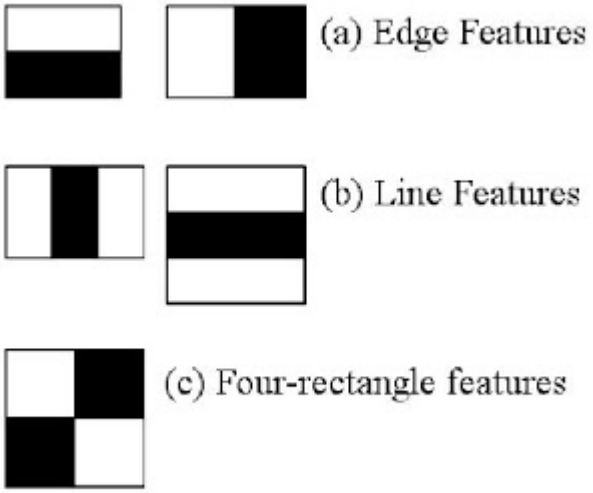
Variance: 58.5

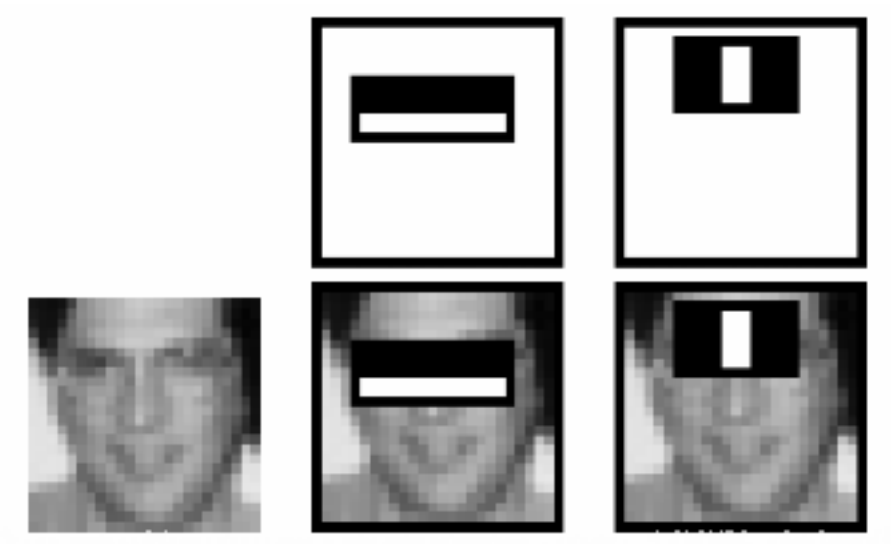
Face detection

Demo: (show-faces-and-eyes img)

(def clr (CascadeClassifier. “haarcascade_frontalface_default.xml”))
(.detectMultiScale clr img result)

Haar Classifier





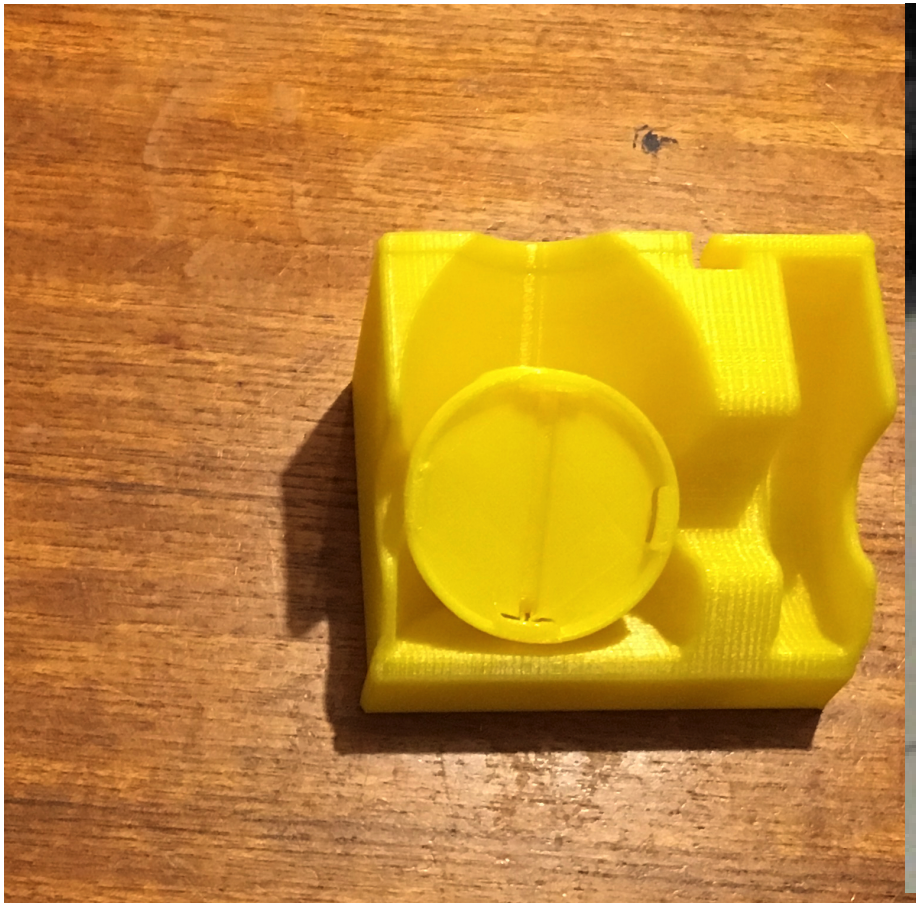
Each feature is a single value obtained by subtracting sum of pixels under white rectangle from sum of pixels under black rectangle.

http://docs.opencv.org/master/d7/d8b/tutorial_py_face_detection.html

Blob detection

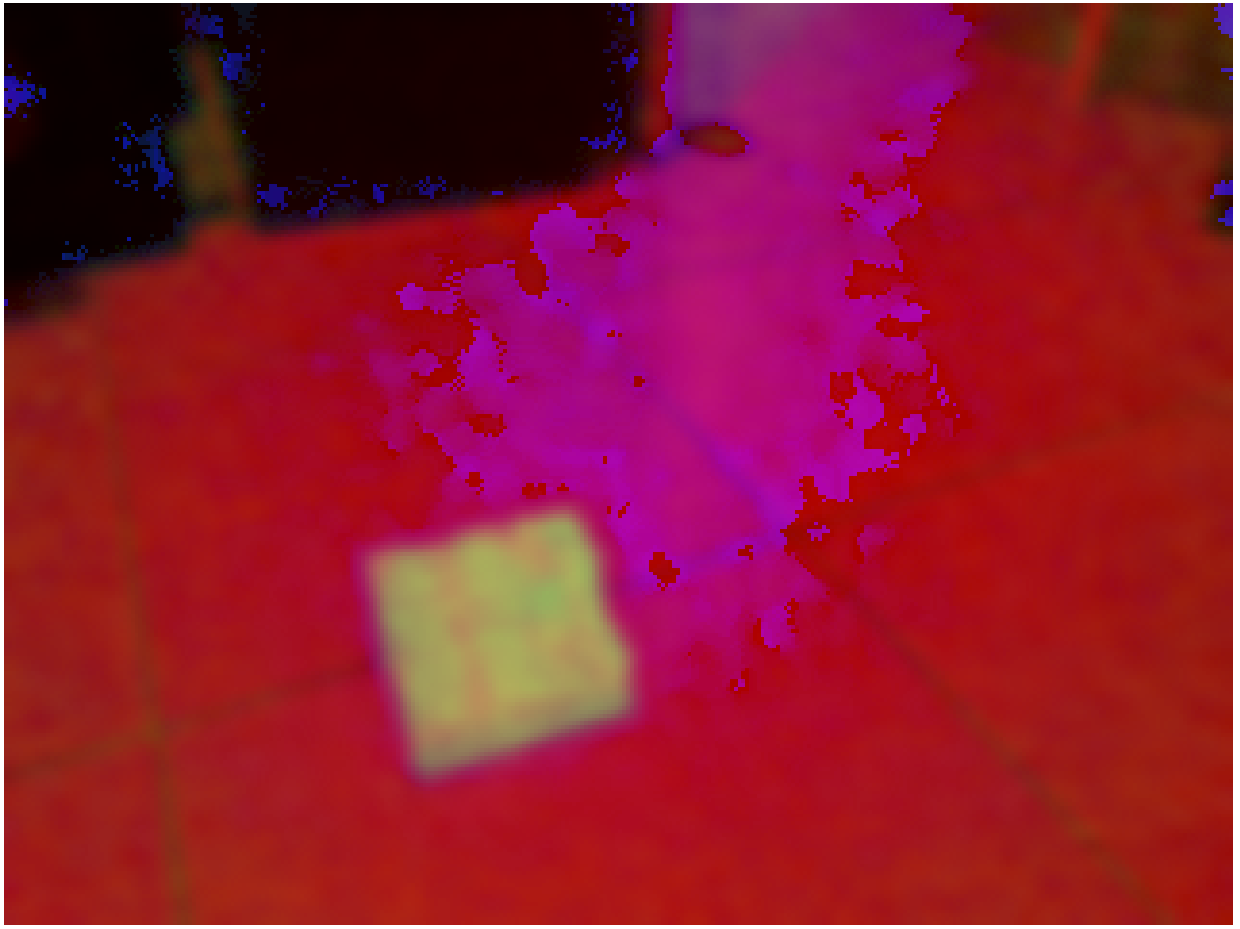
Demo: (show-blob blob-img
(Scalar. 84 80 80) (Scalar. 104 255 255))

Lighting can mess you up

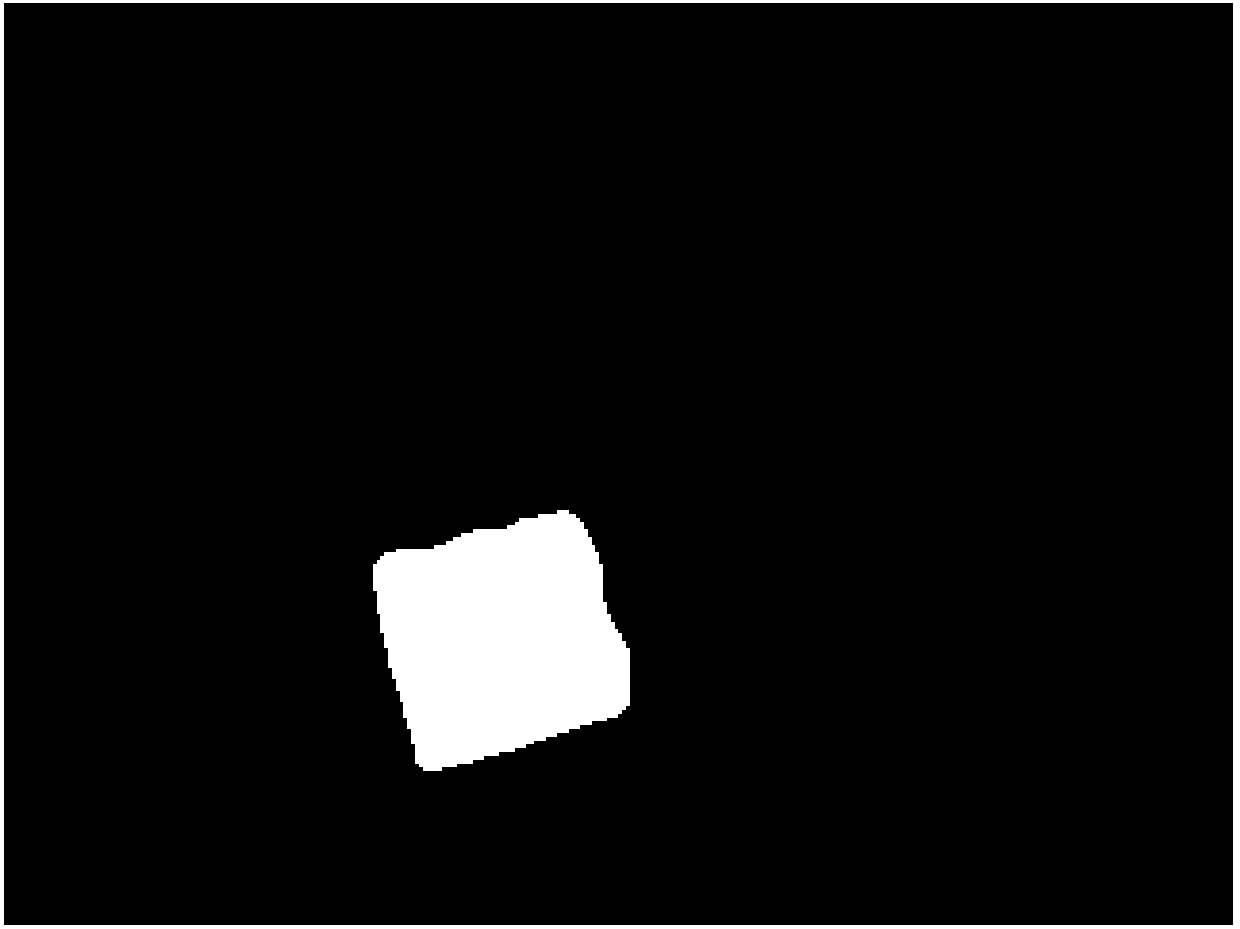


(Imgproc/GaussianBlur src dest (Size. 11 1) 0.0)

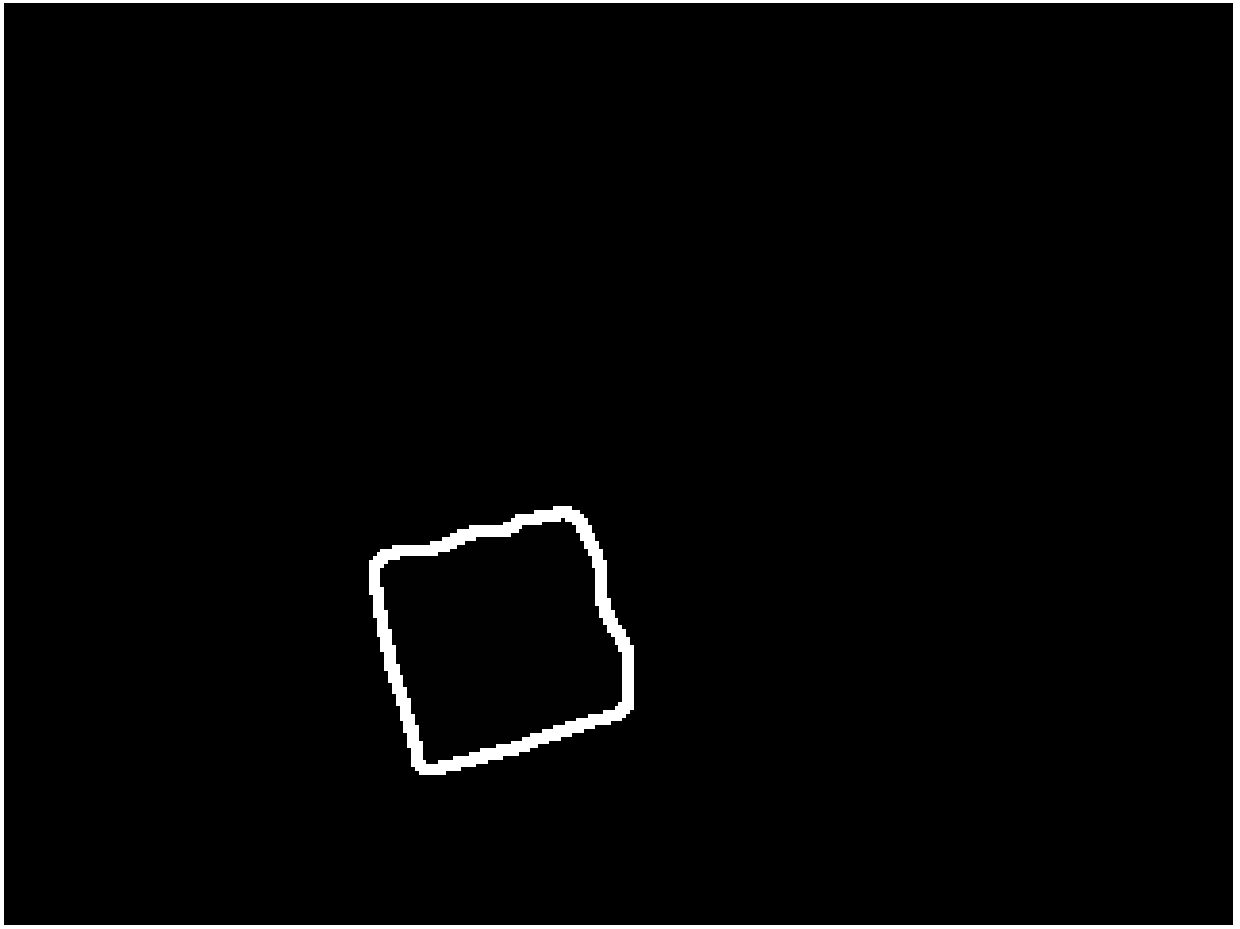
(Imgproc/cvtColor src dest Imgproc/COLOR_BGR2HSV)



(Core/inRange src low high dest)



(Imgproc/findContours img result (Mat.)
Imgproc/RETR_EXTERNAL
Imgproc/CHAIN_APPROX_SIMPLE)



Summary

- Powerful, but very imperative style
  - wrapper functions make it more palatable
- Demo code & wrappers:
  https://github.com/davesnowdon/cljex-2015-opencv
- Vision library probably worth a look
  - http://nakkaya.com/vision.html
- See also
  - http://opencv.org/documentation.html
  - http://www.pyimagesearch.com/

Demo functions

https://github.com/davesnowdon/cljex-2015-opencv

```
(def img (read-image “filename.jpg”))

(write-image “filename.png” img)

(show-line img)

(is-image-blurry img)

(show-faces-and-eyes img)

(def low-value (Scalar. hue saturation  value))
(def high-value (Scalar. hue saturation value))
(show-blob img low-value high-value)
```