Making the Computer See

Computer Vision For Everyday Applications

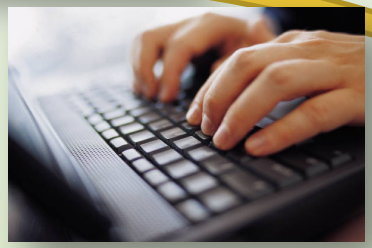
Martin Jul

NDC Oslo, June 5th, 2014

Our Habitual World…

Human Data Processing

Unstructured Data



Producer: MF
Wine: Pinot Noir
North Coast
Vintage: 2011

Structured Data

Machine Data Processing

Fun?

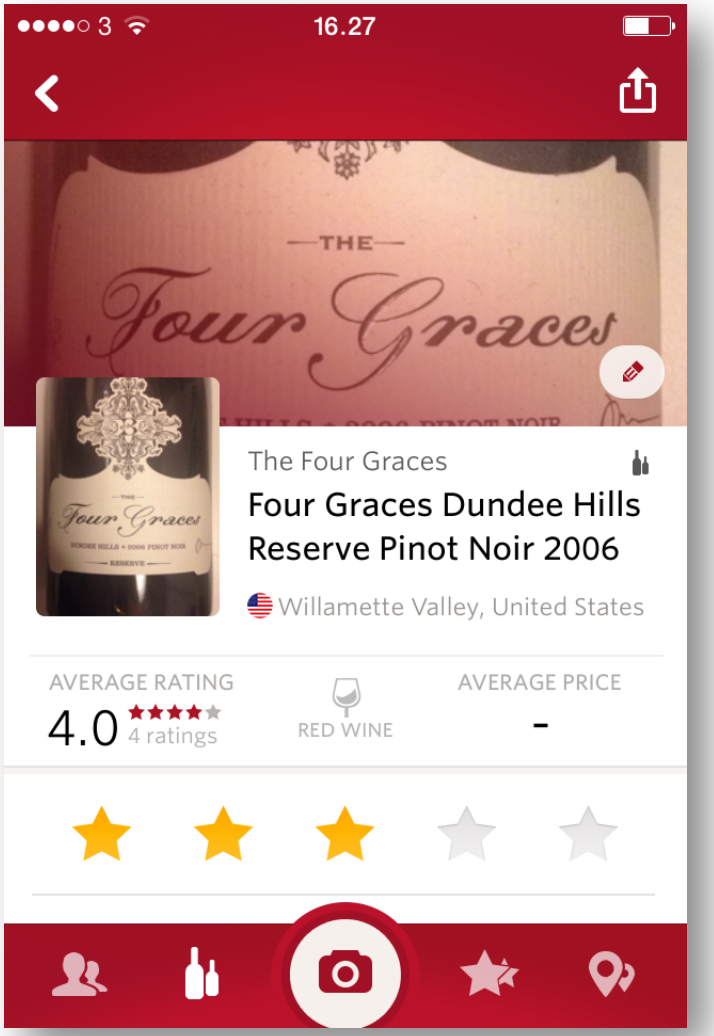

Orange
04:07 PM
66 %
Messages
Crash 1337
Edit
Do you like your new touch-screen keyboard?
it's %$!# slow!
Send

Fun?

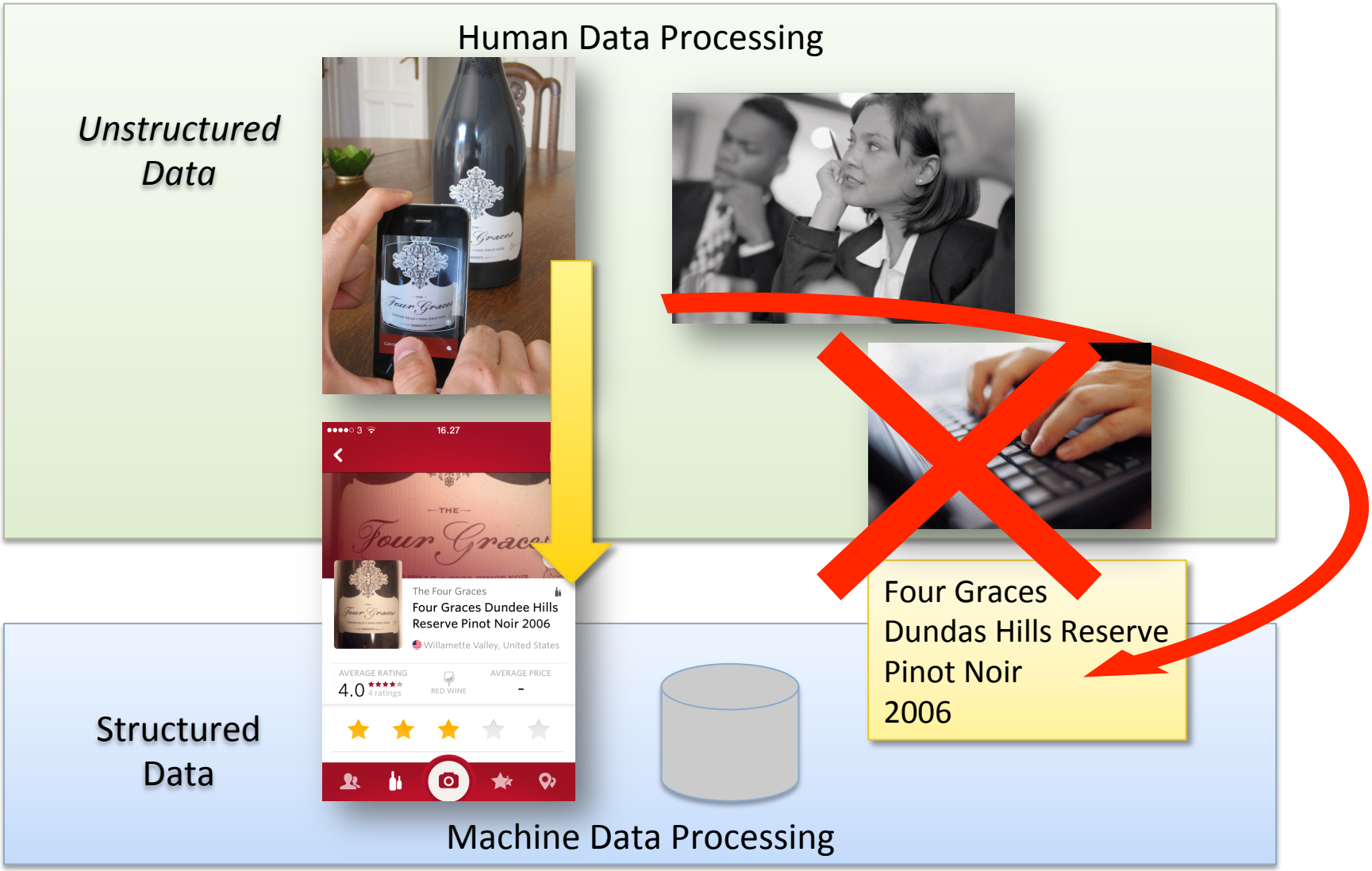
http://en.m.wikipedia.org

Fun!


THE
Four Graces
DUNDEE HILLS • 2006 PINOT NOIR
RESERVE
Cancel


3
16.27
THE
Four Graces
The Four Graces
Four Graces Dundee Hills Reserve Pinot Noir 2006
Willamette Valley, United States
AVERAGE RATING
4.0
4 ratings
RED WINE
AVERAGE PRICE
-

“Software is Eating the World”



Agenda

- Historical Context
- Books and Cars – the 20 mill. kroner problem
  - Recognizing Shapes and Geometry
  - Reading text from images
- Pizza is Here
  - Object Detection
- Wine
  - Computer Vision and Machine Learning

A long time ago in a galaxy far, far away....

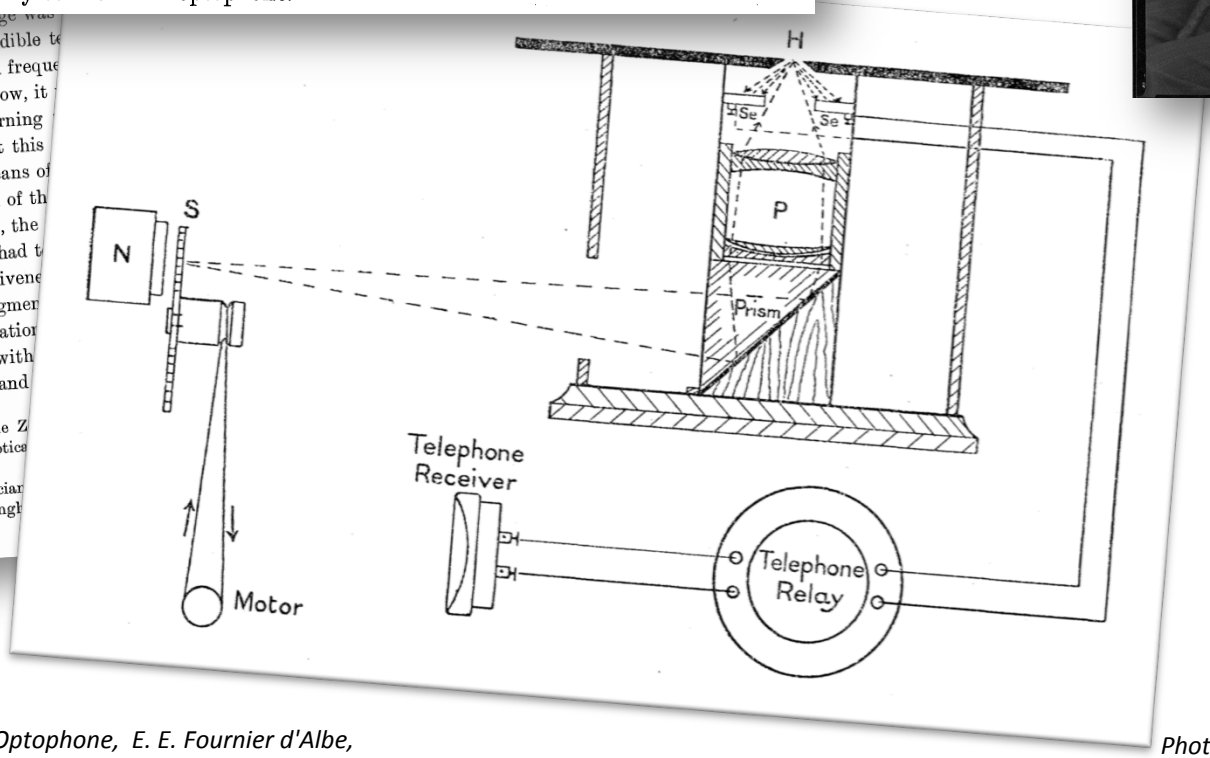
1910s

On a Type-reading Optophone.

By E. E. Fournier d'Albe, D.Sc. (Lond. and Birm.).

(Communicated by Sir Oliver Lodge, F.R.S. Received May 2,—Read May 28, 1914.)

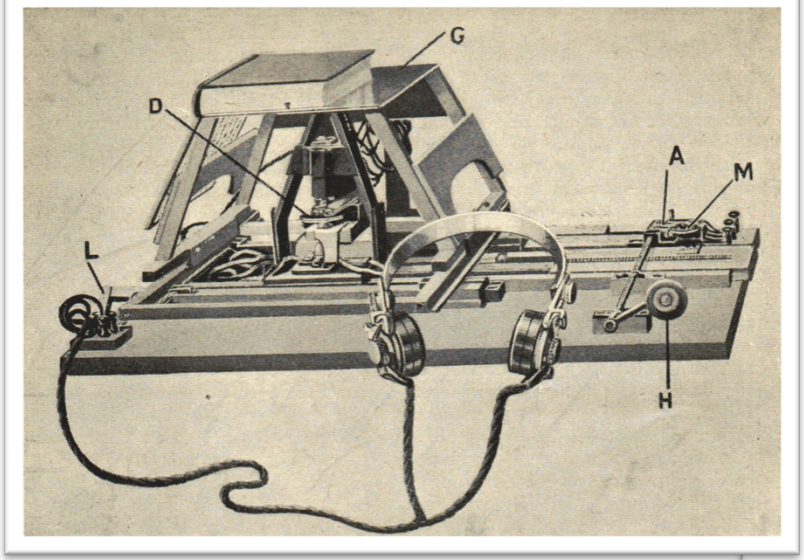
The production of sounds directly or indirectly due to the incidence of light is the general function of instruments of the type of Graham Bell's "photophone." An instrument designed to solve the more special problem of substituting the sense of hearing for the sense of sight is more appropriately termed an "optophone."



On a Type-Reading Optophone, E. E. Fournier d'Albe, Proc. R. Soc. Lond. A 1914 90 (published 1 July 1914)

Photo from National Portrait Gallery (1932) http://www.npg.org.uk/collections/search/person/mp97681/edmund-edward-fournier-dalbe

1920s



https://en.wikipedia.org/wiki/Optophone

SEVENTY-SIXTH YEAR

SCIENTIFIC AMERICAN

THE WEEKLY JOURNAL OF PRACTICAL INFORMATION

VOLUME CXXIII. NUMBER 19

NEW YORK, NOVEMBER 6, 1920

15 CENTS A COPY
20 CENTS IN CANADA

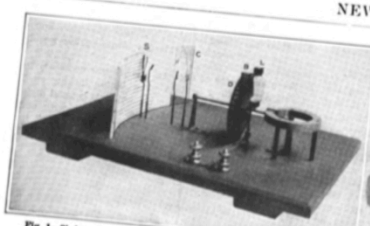


Fig. 1—Skeleton apparatus showing the principle of the "black-sounding" optophone

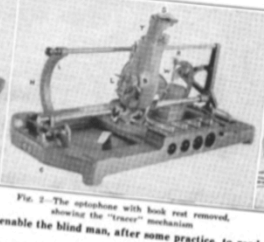


Fig. 2—The optophone with book rest removed, showing the "tracer" mechanism

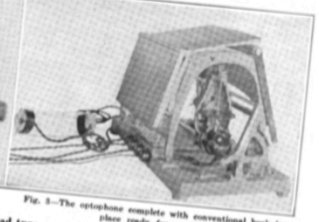


Fig. 3—The optophone complete with conventional book in place ready for reading

Simple and standard forms of optophone which enable the blind man, after some practice, to read type matter by means of sound

The Type-Reading Optophone

An Instrument Which Enables the Blind To Read Ordinary Type

IT would be difficult to conceive of a more remarkable electrical device than the type-reading optophone, which enables the blind to read ordinary type. This device, be it understood, is not altogether new as far as its basic principles are concerned. As far back as August 3rd, 1912, the SCIENTIFIC AMERICAN SUPPLEMENT carried a description of the first optophone which in its earliest form was an instrument for converting light into sound in such a manner that the blind were enabled to see by ear, so to speak.

The optophone was the invention of Dr. E. E. Fournier d'Albe of Birmingham University. Dr. d'Albe later developed a type-reading instrument by which the blind could read large print. Recently the instrument has been greatly improved in coöperation with Messrs. Barr and Stroud of Glasgow, the well-known instrument makers. In its present form it can be adjusted for any ordinary type, but it is well to direct attention to the fact that the blind person making use of the optophone must have a very fine musical ear in order to grasp the tone combinations emitted by the instrument and translate them into reading matter.

The general principle of the apparatus is shown in Fig. 1, wherein a siren disk, *D*, is run at about 30 revolutions a second by means of the small magneto-electric motor shown. It contains five circles of square perforations, the innermost circle having 24 perforations, the outermost 42, the other circles being intermediate and corresponding to the relative frequencies of certain notes of the diatonic scale. A line of light in a radial direction is provided by the straight-filament lamp *L*, and the image of the filament of this lamp is thrown upon the print by a system of three lenses on the other **side of** the selenium tablet *S*. The axis of the concavo-convex lens *C* is slightly tilted out of the axis of the other lenses for a purpose which is specified below. The general result of the optical system is to give a line of luminous dots on the print, each dot having a different musical frequency. The light constituting these dots is diffusely reflected back on to the selenium, which is put in circuit with a battery and a high-resistance telephone receiver. Those dots which fall on white paper produce a note of their own musical frequency in the telephone, while those which fall on black are extinguished, so to speak. We thus get what may be called a "white sounding" optophone, in which the black letters are read by the notes omitted from the scale rather than by the notes which remain sounding. All the reading demonstrations hitherto undertaken have been given with a "white-sounding" optophone, with the most satisfying results.

A modification of this principle, we learn from the SCIENTIFIC AMERICAN MONTHLY [successor to the SCIENTIFIC AMERICAN SUPPLEMENT] for October, has been introduced by Messrs. Barr and Stroud in consultation with Dr. d'Albe. This is the provision of a second selenium preparation in the form of a cylindrical rod, the top of which can be seen at *B*. This rod receives the light reflected by the concave surface of the lens *C*, which produces a real image of the line of dots on a generator of the cylindrical rod, and by turning this rod about its axis the image can be made more or less effective as desired. By balancing the effect on *B* against the effect on *S*, when white paper alone is exposed, a silence can be produced in the telephone, and the effect of the passage of a black letter is to make a sound which varies in accordance with the formation of the letter. This is the principle of what may be called a "black-sounding" optophone, and although its advantage over the "white-sounding" type has yet to be proved, there is little doubt that the learning of the alphabet sounded on the new principle will be easier.

The present construction adopted by Messrs. Barr and Stroud is shown in Fig. 2. Here the disk, lamp, lenses, and selenium, as well as the motor, are all mounted in the swinging "tracer," which can be brought over to the right by means of the reading-handle *H*. It then returns to the left with a slow, silent and steady motion regulated by the worm gearing *W*, which drives a small paddle inserted in a viscous liquid. This paddle can be inserted more or less deeply into the liquid by the regulating nut *R*, and such is the range of adjustment possible that a line can be read in anything from five seconds to five minutes, according to the proficiency of the reader. When the line is read, the next line is brought into focus by the change-bar *C*, which works a friction grip inside the bar on which the "tracer" is pivoted, and can be adjusted for any desired line space by means of the screw attached to the change bar. A lever attached to the "tracer" enables the operator to reverse this motion or to release the whole "tracer" from the friction gear, so that it may be quickly brought to the top of the page.

The straight-filament lamp is inserted at *L*, where it is held by a spring clip, and whence it can easily be removed for renewal even by a blind operator. The balancer is inserted at *B*, and can be adjusted for silence by means of the small handle shown.

Fig. 3 shows the apparatus from the top page end and with telephone and flexible connections attached, as well as the book-rest *V* holding a book. The adapters of the flexible connections are all of different sizes and fit into different-sized holes in such a manner that they cannot be wrongly inserted.

The operator of the optophone reads by the tones and combinations of tones emitted by the telephone receiver. The numbers of perforations in the siren disk are in proportion to the notes *G*, *C*, *D*, *E*, *G*, (sol, do, re, me, sol) of the musical scale. The spot of light corresponding to low *G* falls on the lowest point of such letters as *J*, *p*, *y*, etc., the high *G*′ falling on the tops of capitals and high letters. The three intermediate spots cover the body of the lower case letters. With a "black-sounding" optophone the letter *V* is represented by the motif (high) sol, me, re, do, re, me, (high) sol.

In practice it is found that, with the new apparatus, the various adjustments for size of type, length of line, and line interval are quite easily made by blind persons, and that the instrument, with all its delicate adjustments, can remain in use for a long time without anything getting out of order. As for speed of reading, we are told that Miss Mary Maeson, the blind girl who has been giving demonstrations with the optophone for some time, now reads habitually at a speed of about 25 words a minute with a "white-sounding" optophone. But again we repeat, a so-called musical ear is a very necessary qualification.

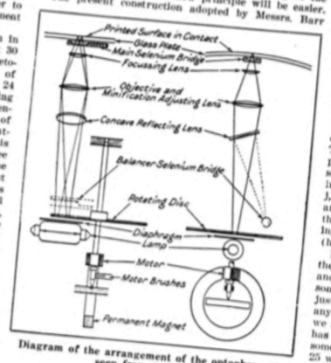
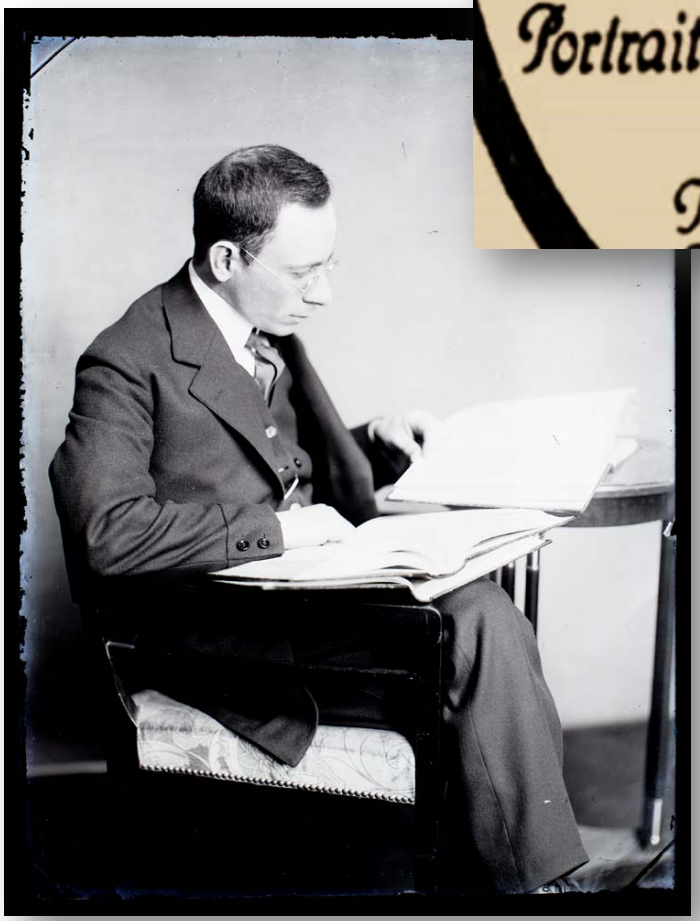


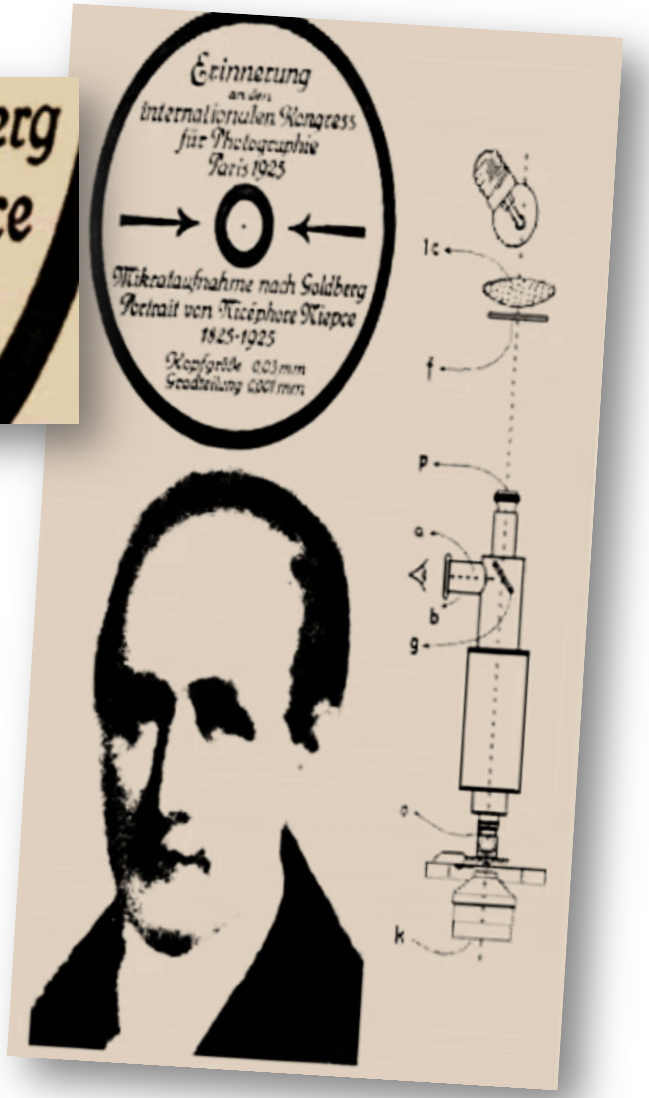
Diagram of the arrangement of the optophone as seen from two sides

Source: http://babel.hathitrust.org/cgi/pt?id=mdp.39015024546411;view=1up;seq=543

1920s



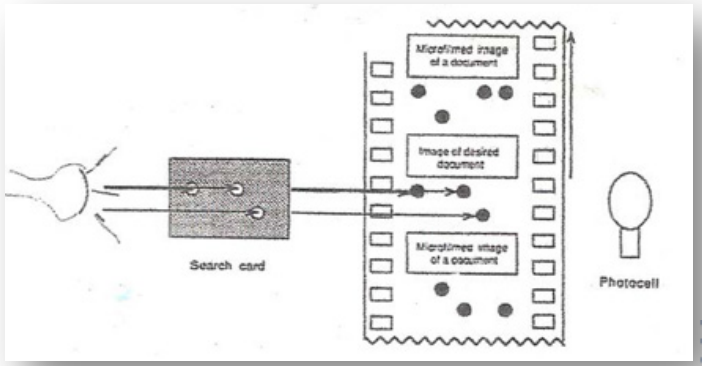
Mikrataufnahme nach Goldberg
Portrait von Nicéphore Niepce
1825-1925
Kopfgröße 0,03 mm



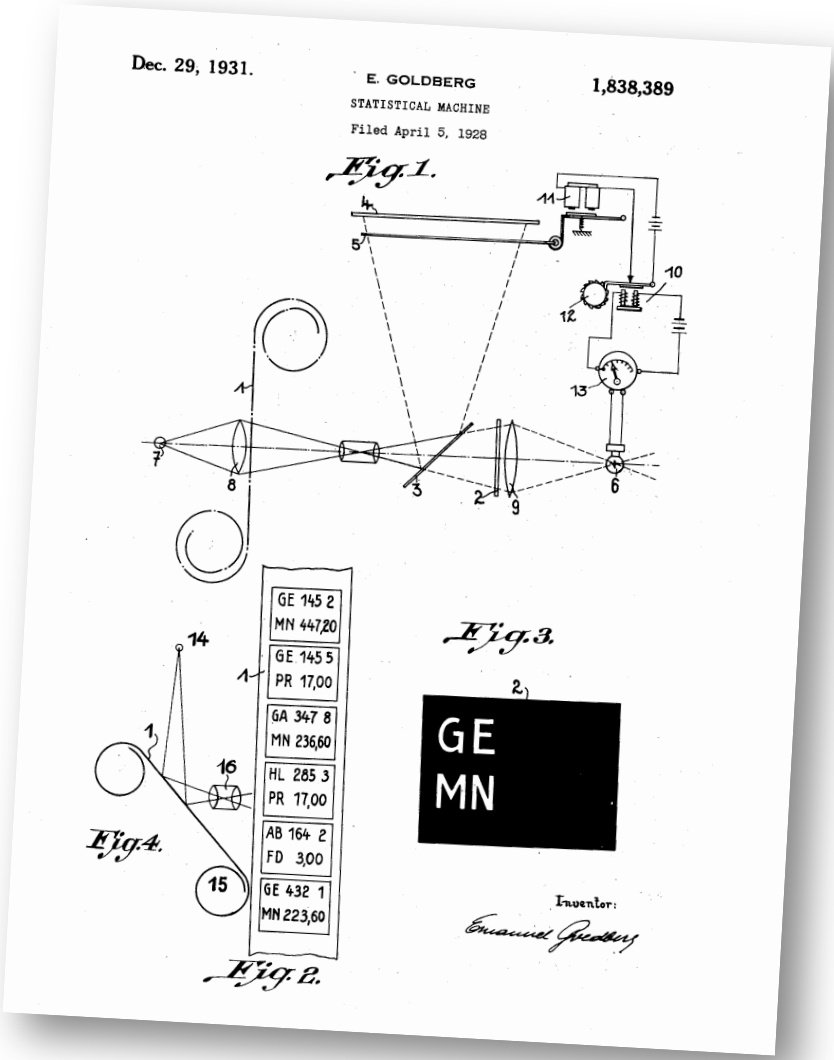
Emanuel Goldberg (1881-1970), photo from 1907,
http://zeisshistoricasociety.org/emanuel-goldberg/

1920s – 1930s

Abb. 7-8: Modell der Statistischen Maschine – von oben und der Seite (Foto M. Turkiewicz 2013, HAW Hamburg)



Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

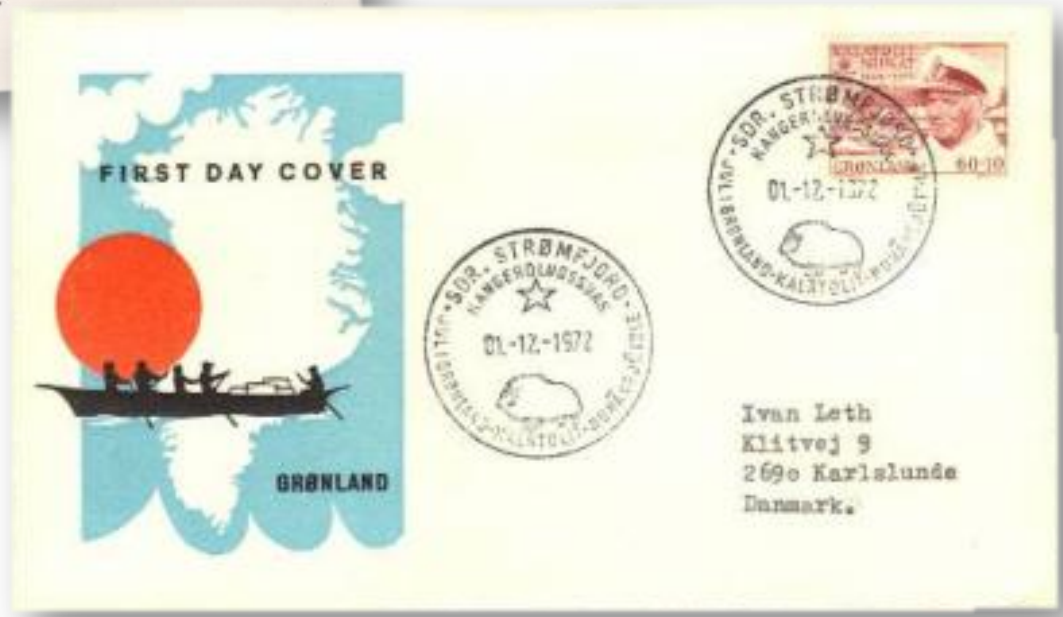


Demonstration model built by HAW Hamburg students: http://www.searchstudies.org/tl_files/ADSDS/Projektbericht_Statistische_Maschine_Gurevich_Heuer_Turkiewicz.pdf

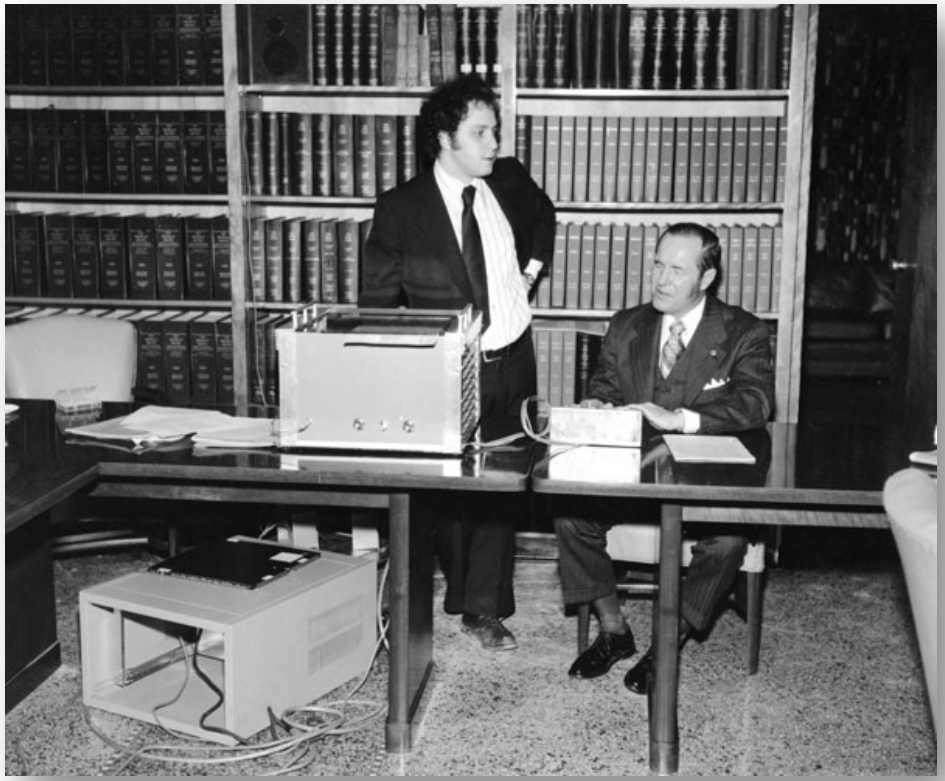
Patent info: http://patft.uspto.gov patent 1 838 389

1960s


Thule
1910-1960
THULE·1910-1960
KNUD RASMUSSEN
30
GRØNLAND
DUNDAS
KNUD RASMUSSENS
GAMLE BOPLADS THULE
24. 11. 1960
FRIMÆRKETS UDGIVELSESDAG
DAH 59
First day cover
Mrs. Clara R. Schaefer,
1733 North Michigan Ave,
Saginaw, Mich.
U. S. A.


FIRST DAY COVER
GRØNLAND
SDR. STRØMFJORD
01-12-1972
Ivan Leth
Klitvej 9
269o Karlslunde
Danmark.

1970s



Kurzweil Reading Machine (1976)
64 KB RAM
¼ MHz CPU
Flatbed CCD Scanner
Software:

- Omnifont OCR
- Text-to-speech

<u>Price in today's money:</u>
195,900 USD (2014 at CPI infl.)
1.16 million NOK

January 1977 – Kurzweil unveils the Reading Machine at Iowa Commission for the Blind
http://www.iowablindhistory.org/blindhistory/tools-and-technology-optical-character-recognition-scanners

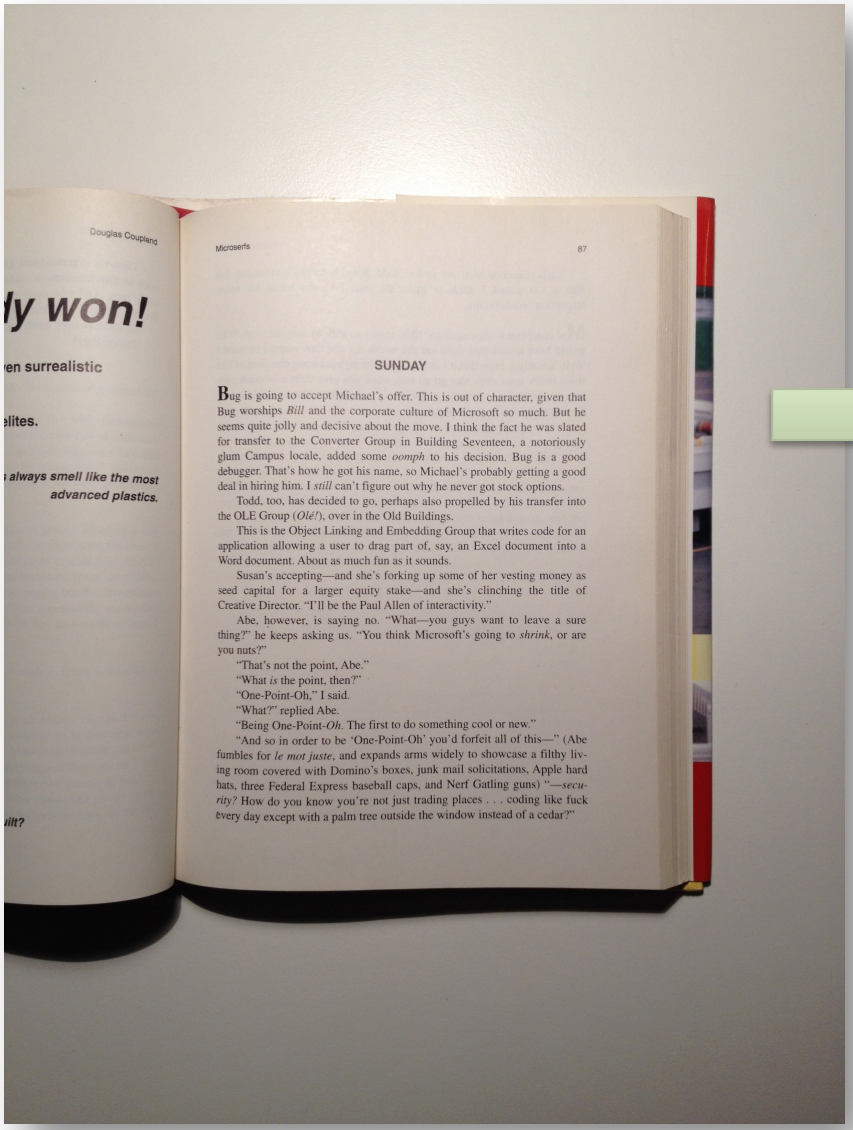
1980s

1987 - HP introduces the desktop scanner, ScanJet
http://www.hpmuseum.net/display_item.php?hw=295

OCR

OPTICAL CHARACTER RECOGNITION

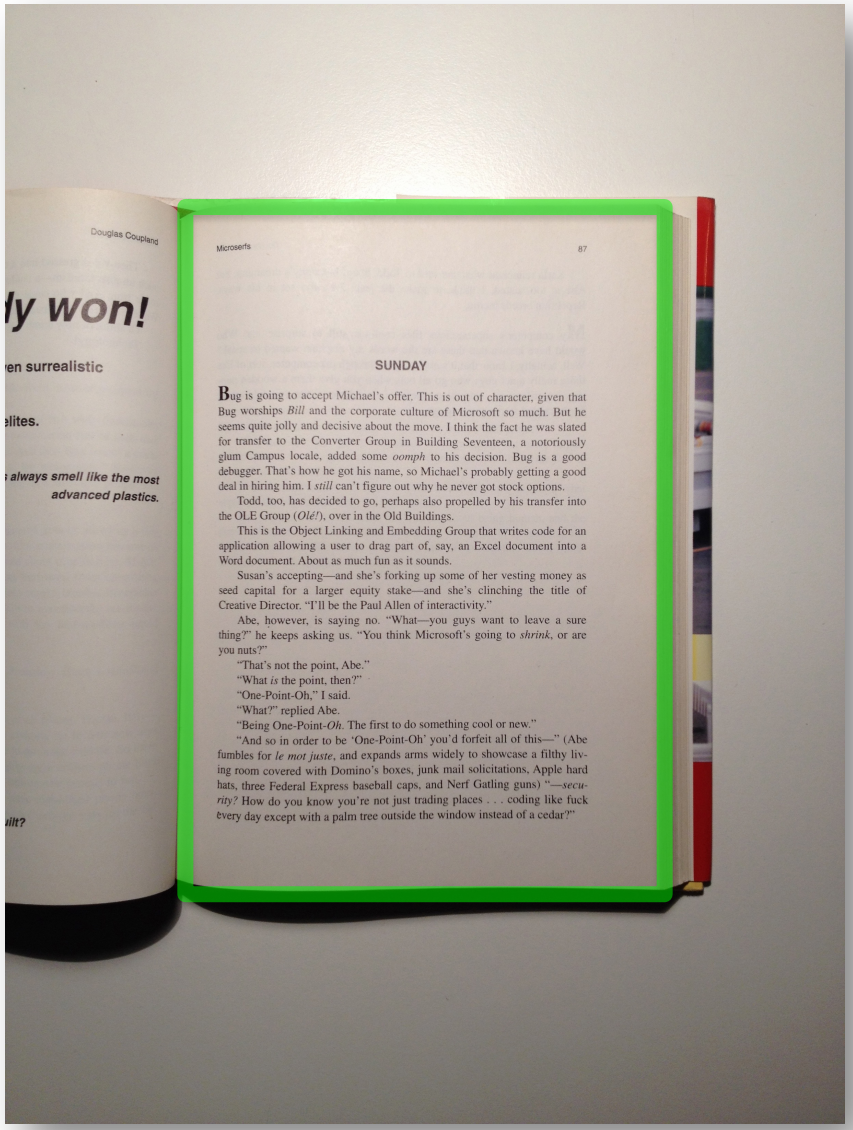
Reading Text From Images



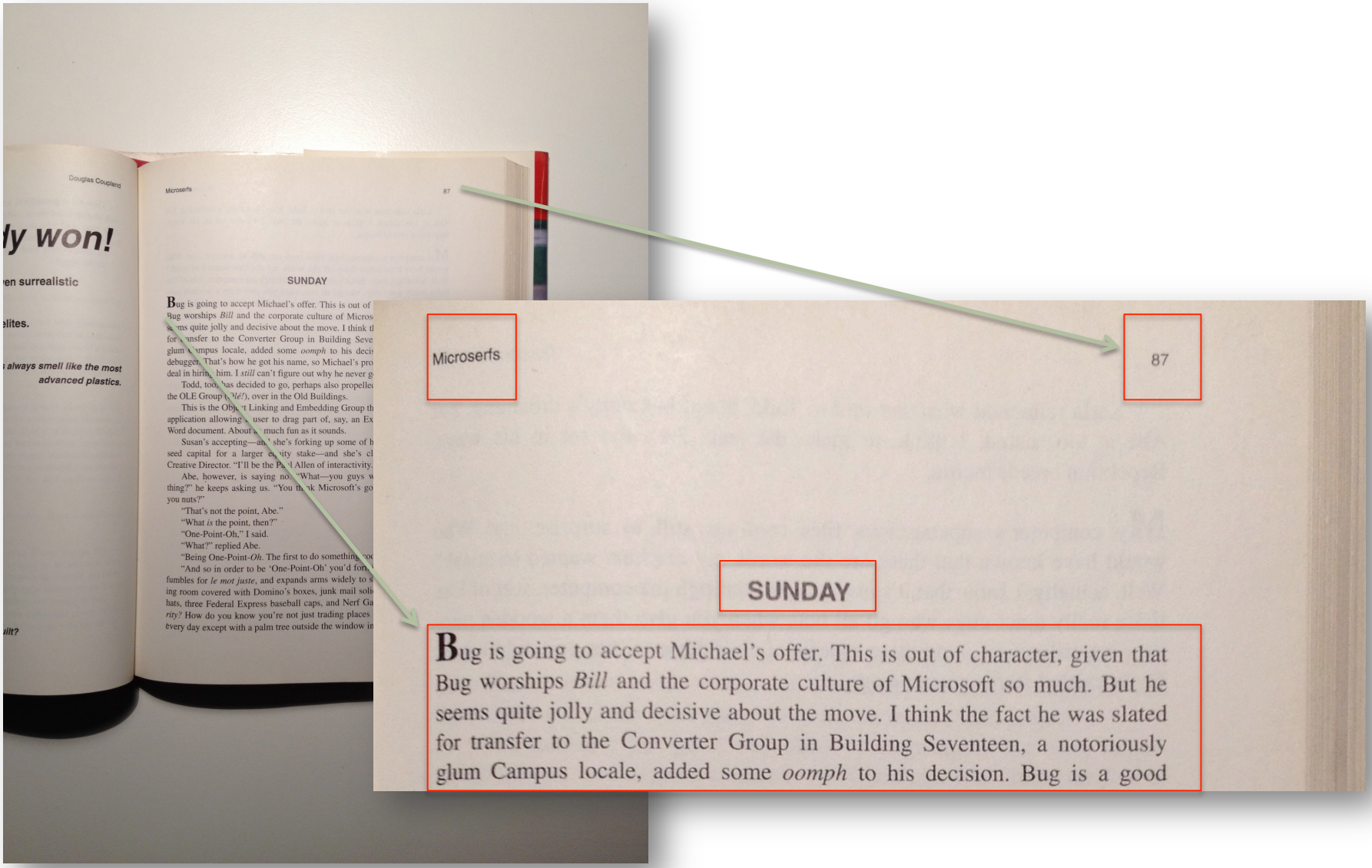

```
SUNDAY

Bug is going to accept Michael's
offer.
```

Illustration: Douglas Coupland, "Microserfs",Harper Collins, New York, NY, USA, 1995

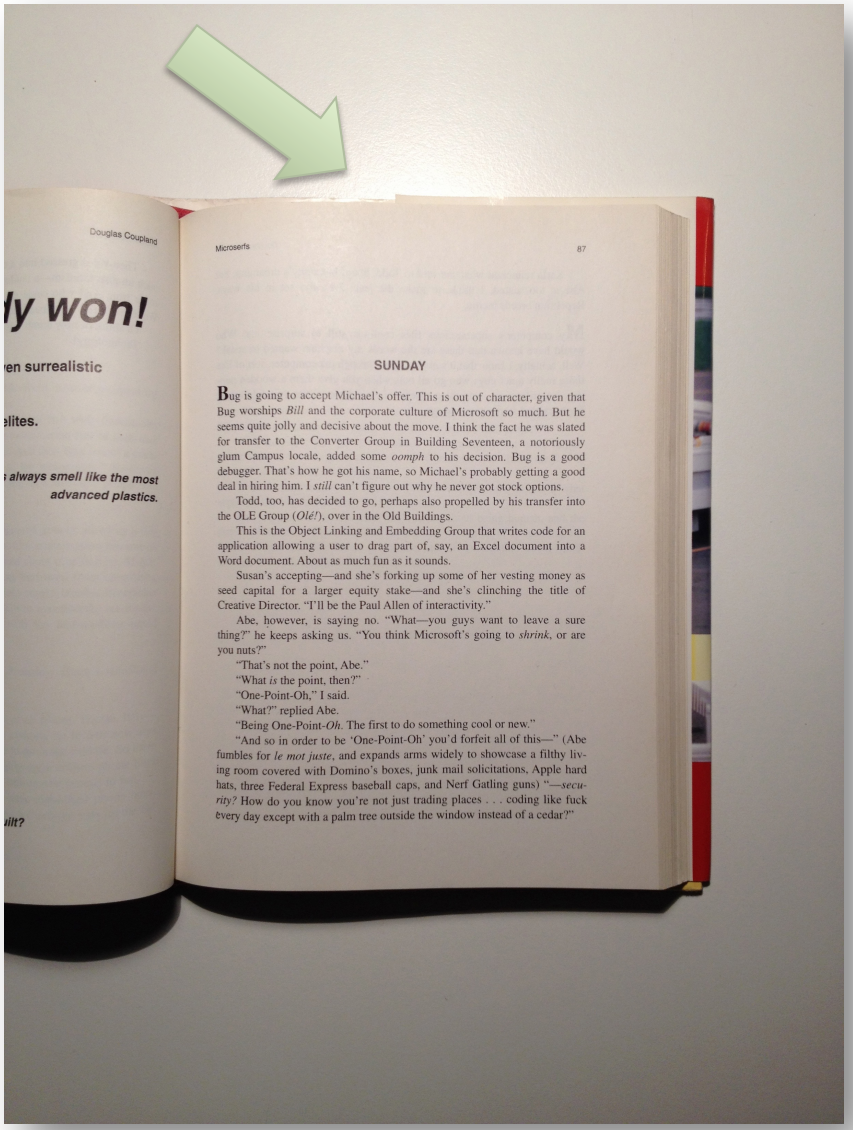
Where is the Page?

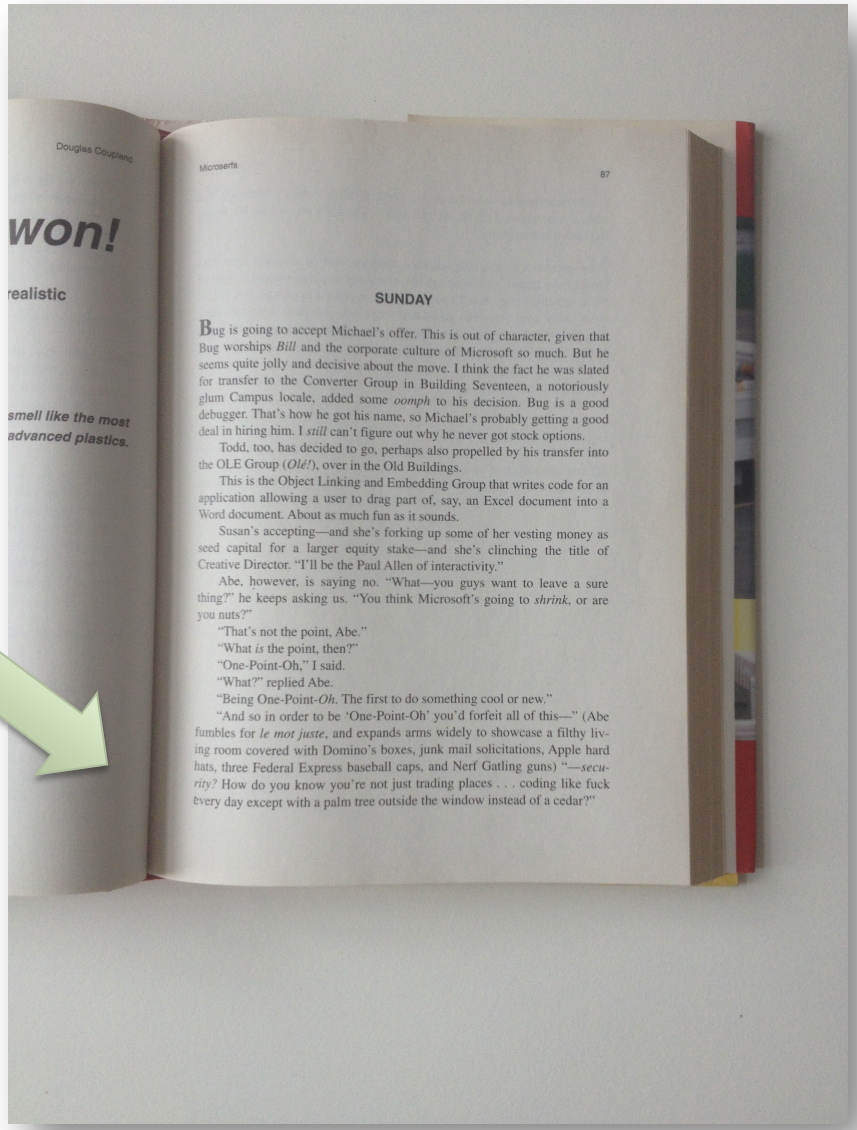


Multiple Texts and Fonts



White Isn’t


Douglas Coupland
Microserfs
87
SUNDAY
Bug is going to accept Michael’s offer. This is out of character, given that Bug worships *Bill* and the corporate culture of Microsoft so much. But he seems quite jolly and decisive about the move. I think the fact he was slated for transfer to the Converter Group in Building Seventeen, a notoriously glum Campus locale, added some *oomph* to his decision. Bug is a good debugger. That’s how he got his name, so Michael’s probably getting a good deal in hiring him. I *still* can’t figure out why he never got stock options.
Todd, too, has decided to go, perhaps also propelled by his transfer into the OLE Group (*Olé!*), over in the Old Buildings.
This is the Object Linking and Embedding Group that writes code for an application allowing a user to drag part of, say, an Excel document into a Word document. About as much fun as it sounds.
Susan’s accepting—and she’s forking up some of her vesting money as seed capital for a larger equity stake—and she’s clinching the title of Creative Director. “I’ll be the Paul Allen of interactivity.”
Abe, however, is saying no. “What—you guys want to leave a sure thing?” he keeps asking us. “You think Microsoft’s going to *shrink*, or are you nuts?”
“That’s not the point, Abe.”
“What *is* the point, then?”
“One-Point-Oh,” I said.
“What?” replied Abe.
“Being One-Point-*Oh*. The first to do something cool or new.”
“And so in order to be ‘One-Point-Oh’ you’d forfeit all of this—” (Abe fumbles for *le mot juste*, and expands arms widely to showcase a filthy living room covered with Domino’s boxes, junk mail solicitations, Apple hard hats, three Federal Express baseball caps, and Nerf Gatling guns) “—*security?* How do you know you’re not just trading places . . . coding like fuck every day except with a palm tree outside the window instead of a cedar?”


Douglas Coupland
Microserfs
87
SUNDAY
Bug is going to accept Michael’s offer. This is out of character, given that Bug worships *Bill* and the corporate culture of Microsoft so much. But he seems quite jolly and decisive about the move. I think the fact he was slated for transfer to the Converter Group in Building Seventeen, a notoriously glum Campus locale, added some *oomph* to his decision. Bug is a good debugger. That’s how he got his name, so Michael’s probably getting a good deal in hiring him. I *still* can’t figure out why he never got stock options.
Todd, too, has decided to go, perhaps also propelled by his transfer into the OLE Group (*Olé!*), over in the Old Buildings.
This is the Object Linking and Embedding Group that writes code for an application allowing a user to drag part of, say, an Excel document into a Word document. About as much fun as it sounds.
Susan’s accepting—and she’s forking up some of her vesting money as seed capital for a larger equity stake—and she’s clinching the title of Creative Director. “I’ll be the Paul Allen of interactivity.”
Abe, however, is saying no. “What—you guys want to leave a sure thing?” he keeps asking us. “You think Microsoft’s going to *shrink*, or are you nuts?”
“That’s not the point, Abe.”
“What *is* the point, then?”
“One-Point-Oh,” I said.
“What?” replied Abe.
“Being One-Point-*Oh*. The first to do something cool or new.”
“And so in order to be ‘One-Point-Oh’ you’d forfeit all of this—” (Abe fumbles for *le mot juste*, and expands arms widely to showcase a filthy living room covered with Domino’s boxes, junk mail solicitations, Apple hard hats, three Federal Express baseball caps, and Nerf Gatling guns) “—*security?* How do you know you’re not just trading places . . . coding like fuck every day except with a palm tree outside the window instead of a cedar?”

Straight Lines Aren't

Abe, however, is saying no. "What—you guys want to leave a sure thing?" he keeps asking us. "You think Microsoft's going to *shrink*, or are you nuts?"

"That's not the point, Abe."

"What *is* the point, then?"

"One-Point-Oh," I said.

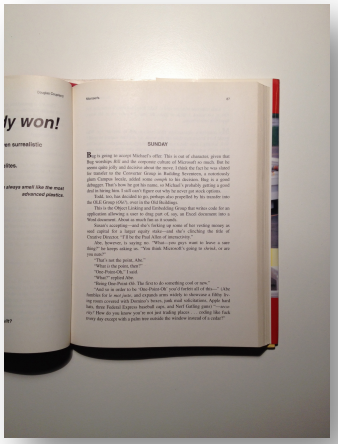
"What?" replied Abe.

"Being One-Point-*Oh*. The first to do something cool or new."

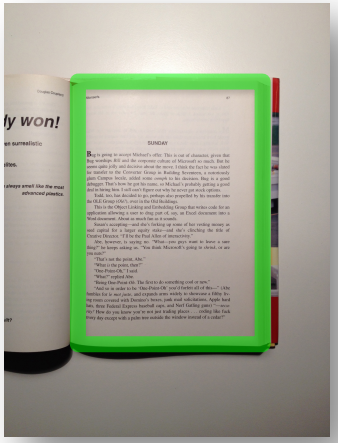
"And so in order to be 'One-Point-Oh' you'd forfeit all of this—" (Abe fumbles for *le mot juste*, and expands arms widely to showcase a filthy living room covered with Domino's boxes, junk mail solicitations, Apple hard

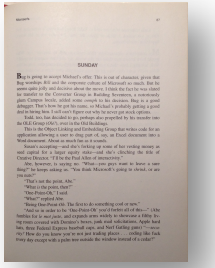
OCR Pipeline

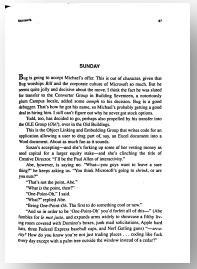
Acquisition



Preprocessing







Magic

Text output

```
"That's not the point, Abe."
"What is the point, then?"
"One-Point-Oh," I said.
"What?" replied Abe.
"Being One-Point-Oh. The first to do something cool or new."
```

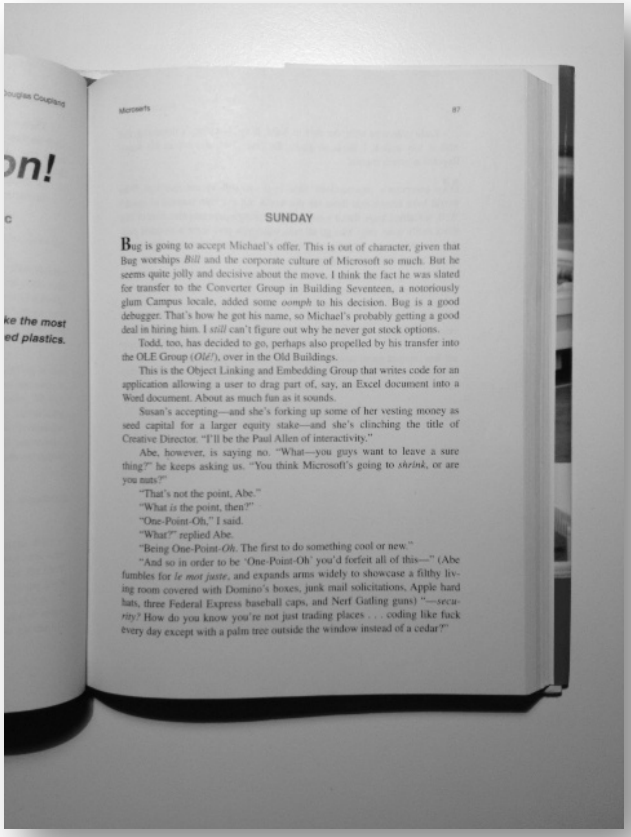
OpenCV – Computer Vision Library

- 2500 CV Algorithms
- BSD license
- All main platforms
- CUDA, OpenCL acceleration
- Realtime, multi-core

- http://opencv.org

Where is the Page?
Trick 1: Reduce Size and Colours

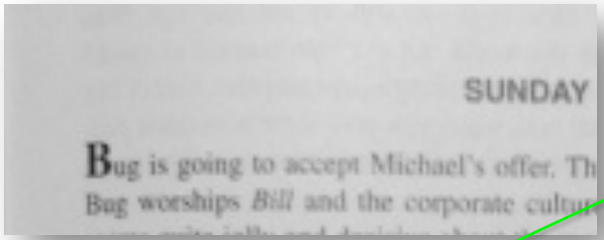
```
def find_page_with_canny(img):
    '''Find the page in the image.'''
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    show('1: Input image', gray)
```

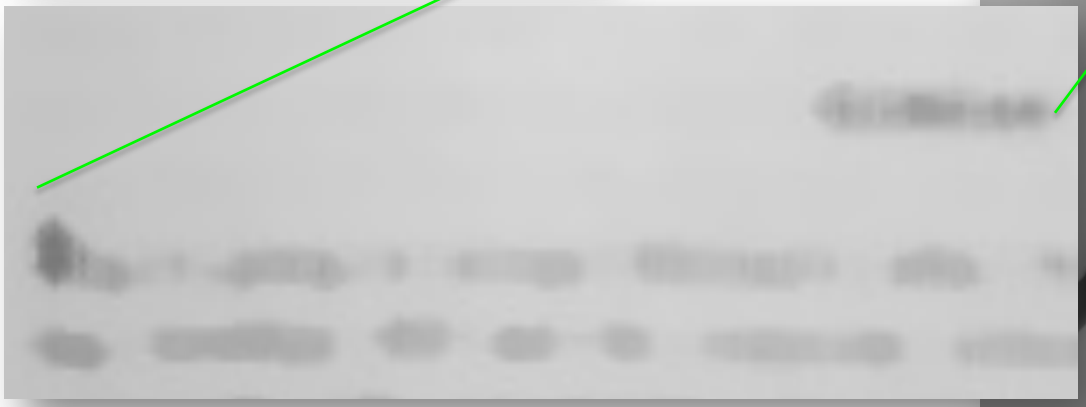


Example code: src/books.py

Where is the Page?
Trick 2: Remove Noise and Details

```
def find_page_with_canny(img):
    '''Find the page in the image.'''
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    show('1: Input image', gray)
    # Remove high-frequency noise (e.g. letters)
    median = cv2.medianBlur(gray, 7)
    show('2: Median filtered', median)
```

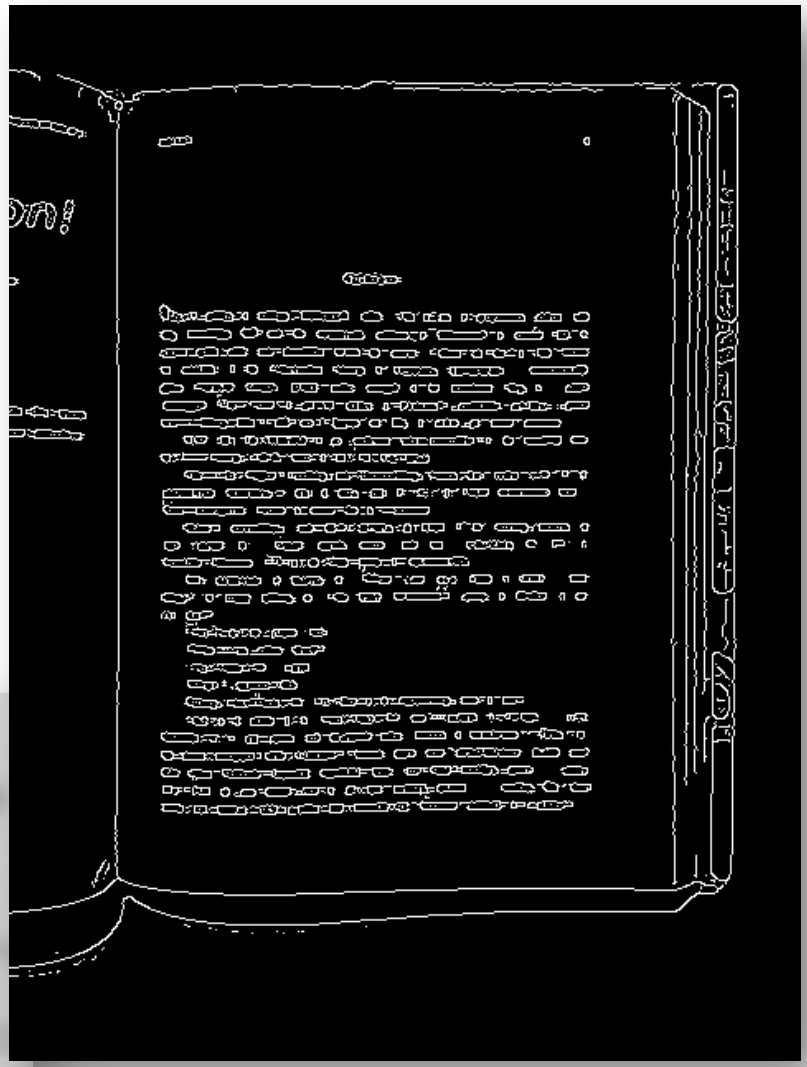




Example code: src/books.py

Trick 3: Find Edges

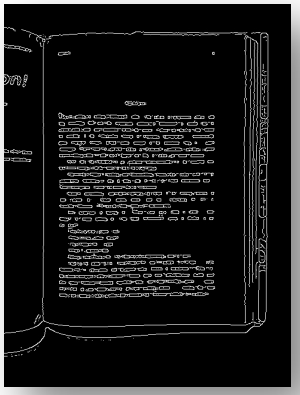
```
def find_page_with_canny(img):
    '''Find the page in the image.'''
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    show('1: Input image', gray)
    # Remove high-frequency noise (e.g. letters)
    median = cv2.medianBlur(gray, 7)
    show('2: Median filtered', median)
    edges = cv2.Canny(median, 10, 20,
                      apertureSize=3)
    show('3: Edges', edges)
```

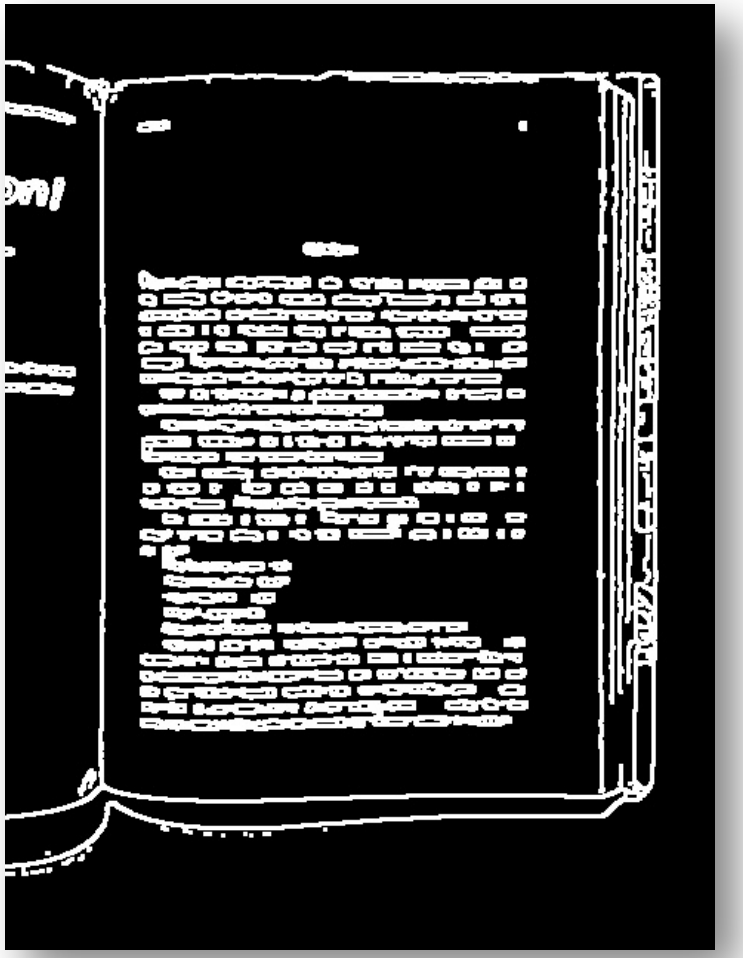


Example code: src/books.py

Trick 4: Dilate to Thicken

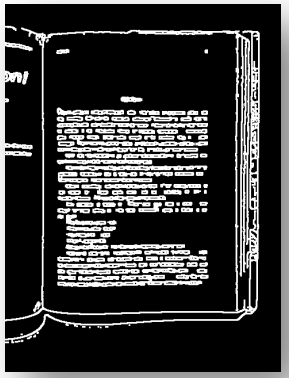
```
def find_page_with_canny(img):
    '''Find the page in the image.'''
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    show('1: Input image', gray)
    # Remove high-frequency noise (e.g. letters)
    median = cv2.medianBlur(gray, 7)
    show('2: Median filtered', median)
    edges = cv2.Canny(median, 10, 20, apertureSize=3)
    show('3: Edges', edges)
    dilated = cv2.dilate(edges,
                         kernel=None,
                         iterations=1)
    show('4: Dilated Edges', dilated)
```

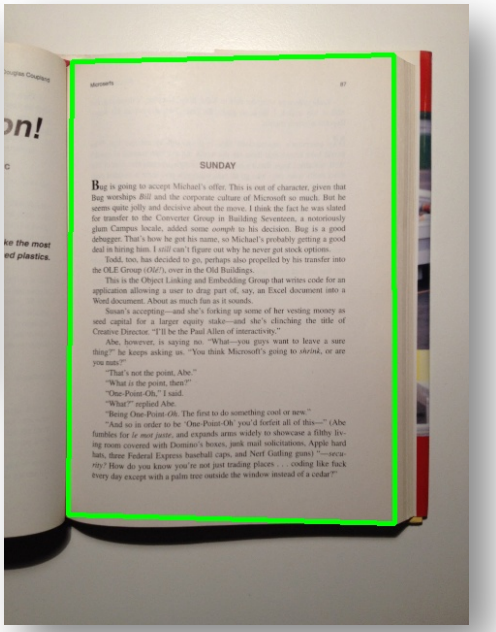




Example code: src/books.py

Trick 5: Find Contours





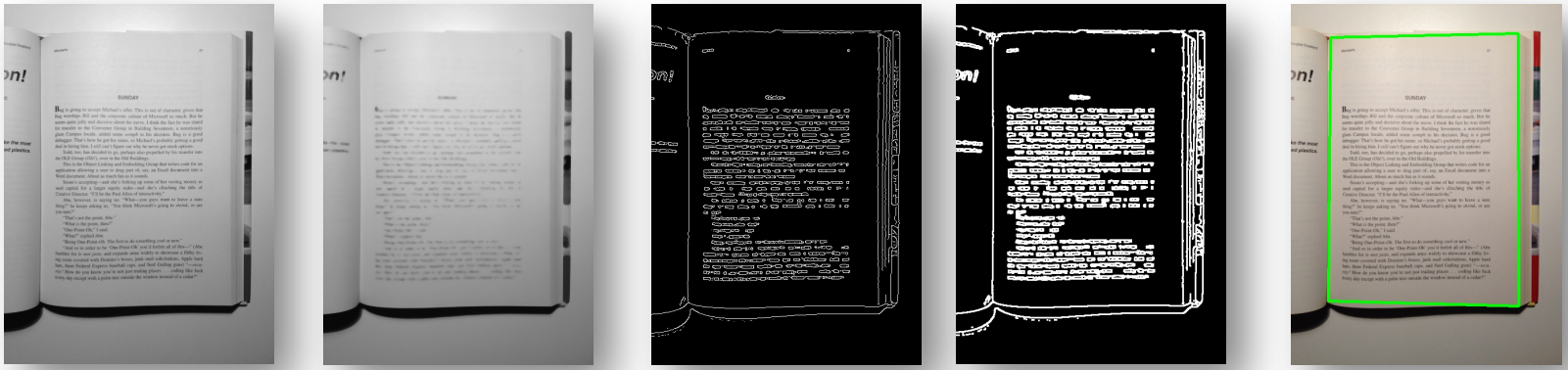
```
def find_page_with_canny(img):
    '''Find the page in the image.'''
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    show('1: Input image', gray)
    # Remove high-frequency noise (e.g. letters)
    median = cv2.medianBlur(gray, 7)
    show('2: Median filtered', median)
    edges = cv2.Canny(median, 10, 20, apertureSize=3)
    show('3: Edges', edges)
    dilated = cv2.dilate(edges, kernel=None, iterations=1)
    show('4: Dilated Edges', dilated)

    min_page_area = img.shape[0] * img.shape[1] / 4
    quads = find_quadrilaterals(dilated, min_area=min_page_area, cos_epsilon=.8)
    quads_found = draw_contours(img, quads)
    show('5: Page found', quads_found)
```

```
def find_quadrilaterals(img, min_area, cos_epsilon):
    quads = []
    contours, hierarchy = cv2.findContours(img.copy(), cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
    for cnt in contours:
        cnt_len = cv2.arcLength(cnt, True)
        poly = cv2.approxPolyDP(cnt, 0.02*cnt_len, True)
        if len(poly) == 4 and cv2.contourArea(poly) > min_area and cv2.isContourConvex(poly):
```

Example code: src/books.py

Where is the Page?

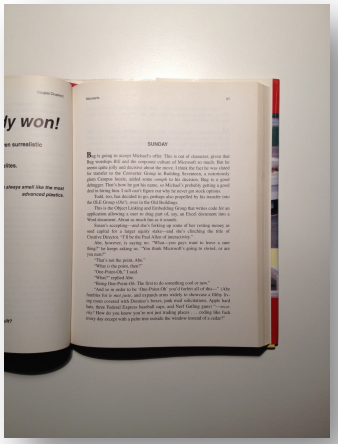
```
def find_page_with_canny(img):
    '''Find the page in the image.'''
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    median = cv2.medianBlur(gray, 7)
    edges = cv2.Canny(median, 10, 20, apertureSize=3)
    dilated = cv2.dilate(edges, kernel=None, iterations=1)
    min_page_area = img.shape[0] * img.shape[1] / 4
    quads = find_quadrilaterals(dilated, min_area=min_page_area, cos_epsilon=.8)
    quads_found = draw_contours(img, quads)
```



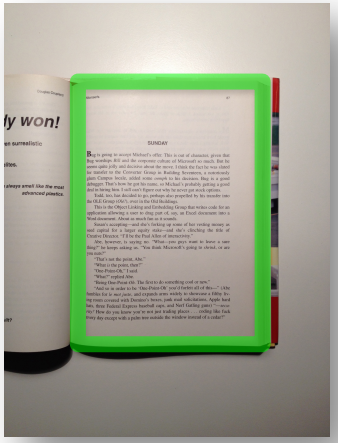
Example code: src/books.py

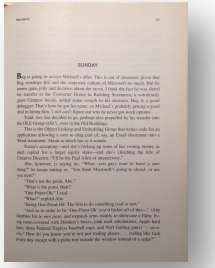
OCR Pipeline

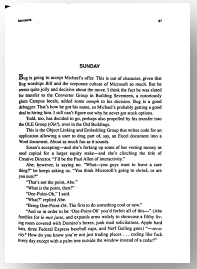
Acquisition



Preprocessing





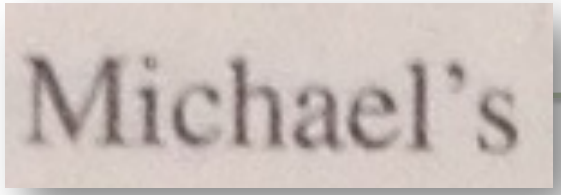


Text output

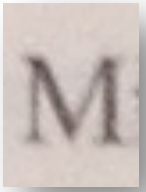
```
"That's not the point, Abe."
"What is the point, then?"
"One-Point-Oh," I said.
"What?" replied Abe.
"Being One-Point-Oh. The first to do something cool or new."
```

Magic

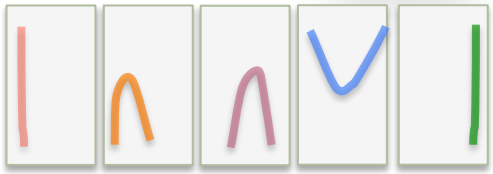
Magic Explained



Adaptive Filter to Black/White



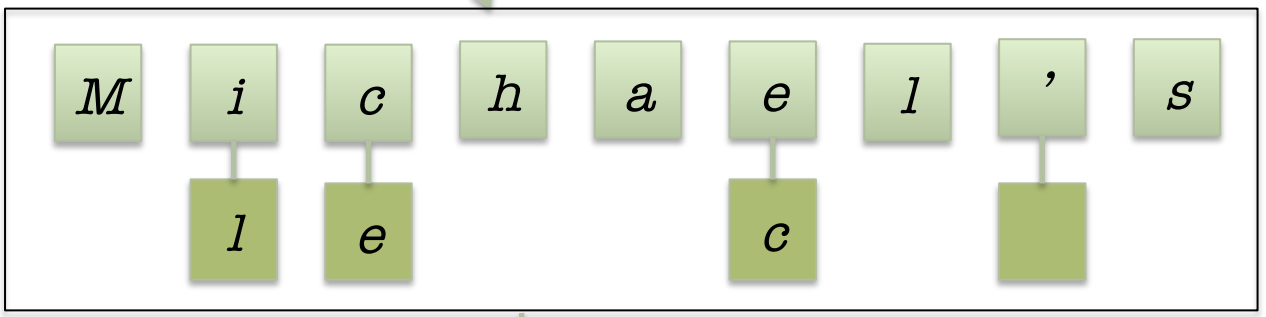
Component shape extraction



Character / font Training data

Shape matching

Probabilities



Language Training data – Dictionaries *etc.*

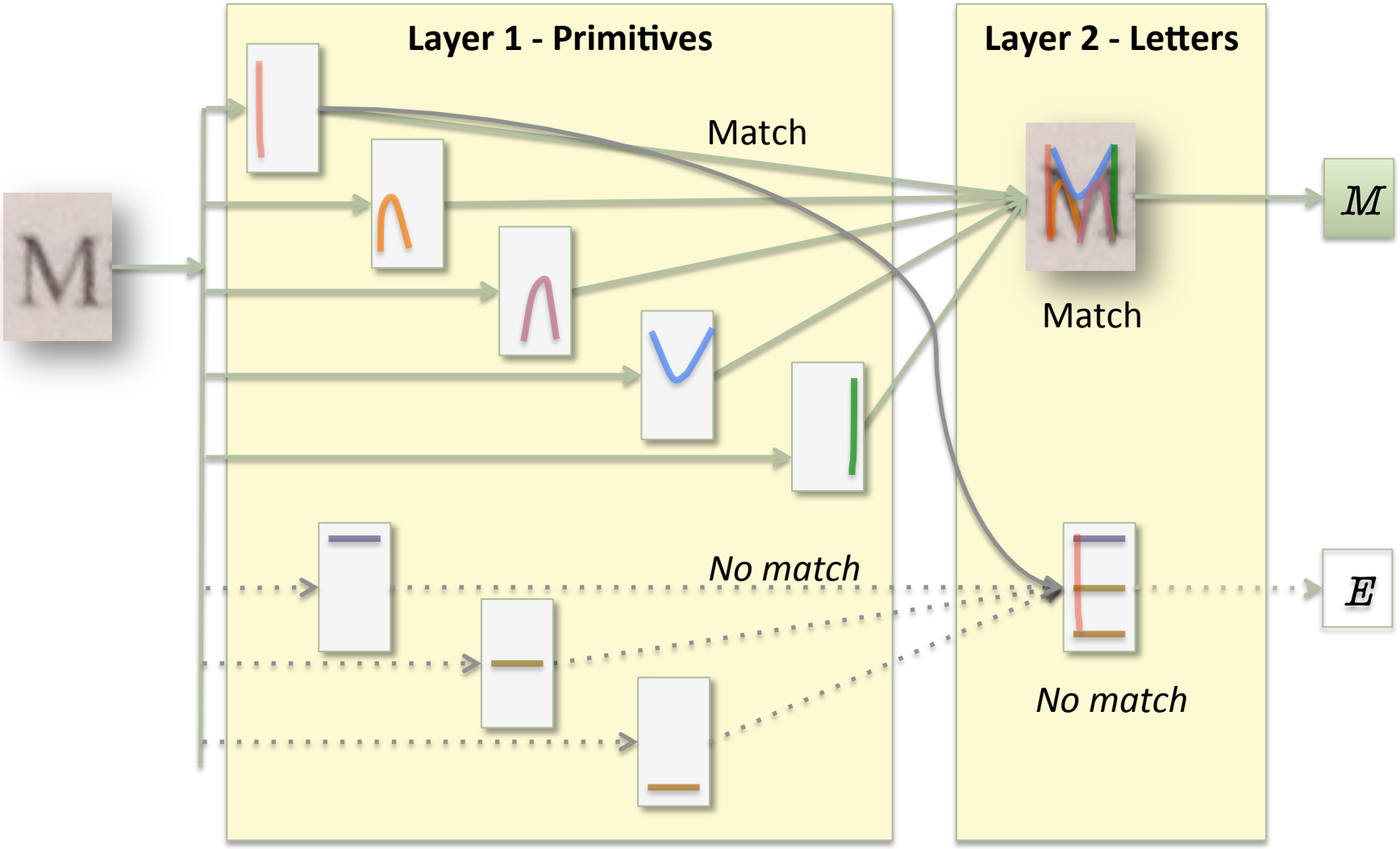
lang=*ENG*

Word matching

Michael's

Statistical confidence

OCR Shape Matching



Tesseract OCR Library

Free, open-source software

Developed by HP Labs 1985-1995

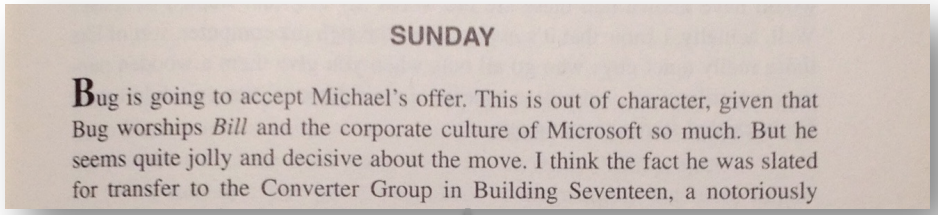
Now sponsored by Google

1987 - HP introduces the desktop scanner, ScanJet
http://www.hpmuseum.net/display_item.php?hw=295

- C++
- Windows, Mac, Linux, Android, iOS
- Wrappers for .NET, Python, …
- Apache License 2.0
- http://code.google.com/p/tesseract-ocr/

OCR - Magic Happens Here

```
def ocr_text(img):
    '''Perform OCR on the image.'''
    tr = Tesseract(lang='eng')
    pil_image = pil.Image.fromarray(img)
    tr.set_image(pil_image)
    utf8_text = tr.get_text()
    return utf8_text
```



SUNDAY

Bug is going to accept Michael's offer. This is out of character, given that Bug worships *Bill* and the corporate culture of Microsoft so much. But he seems quite jolly and decisive about the move. I think the fact he was slated for transfer to the Converter Group in Building Seventeen, a notoriously

SUNDAY

Bug is going to accept Michael‘s offer. This is out of character, given that Bug worships Bill and the corporate culture of Microsoft so much. But he seems quite jolly and decisive about the move. I think the fact he was slated for transfer to the Converter Group in Building Seventeen, a notoriously

Example code: src/books.py

Application: Feature Extraction

CARS

Pipeline

Find Plate

Read Plate

VX 59 134

Look up plate

Danish Police (2013)
400,000 “wanted” plates in Denmark
Automatic scanning and lookup from police cars

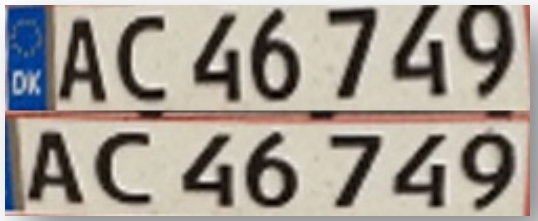
Vision pilot project: 20 million DKK (22 M NOK)

(Norway already has this)

Find the Plate

- Find rectangles in multiple representations
  - Find edges at various grayscale levels (threshold to B/W)
  - Detect edges, extract rectangles as before
  - Plates have (approximately) 5:1 proportions

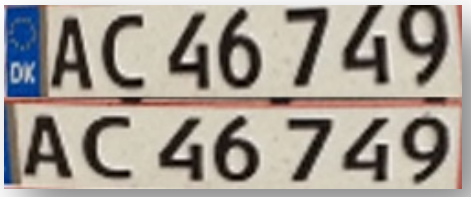
Two candidate plate regions found

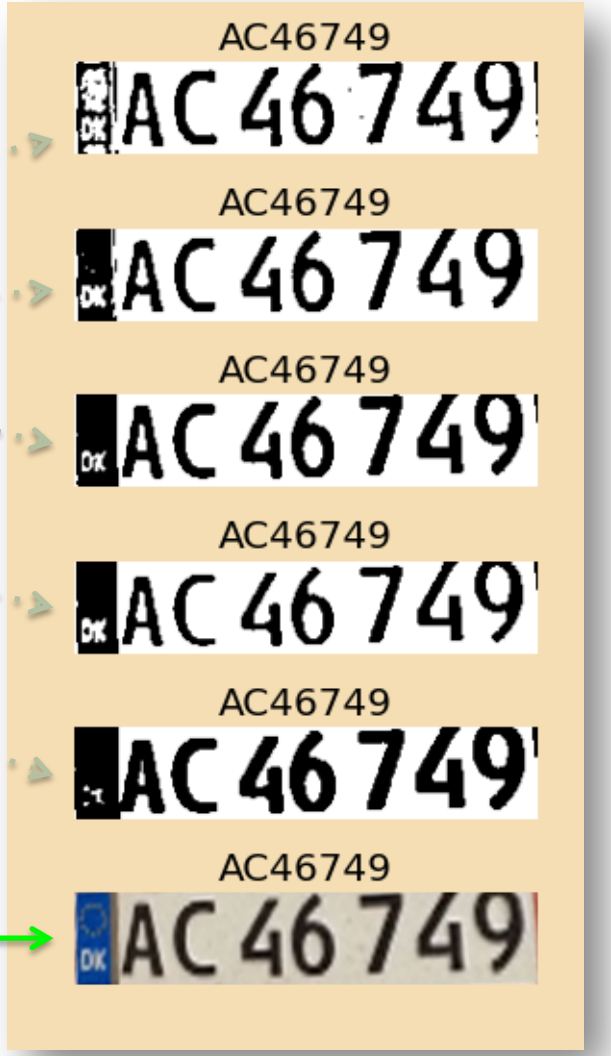


Example code: src/cars.py

Read the Plate

- Transform perspective to “head on”
- Filter plate to different representations
  - Adaptive threshold
  - Manually threshold at middle gray
  - Black/white with optimal threshold (“OTSU”)
  - Erode/dilate letters
- OCR all
- Match OCR texts to “2 letters, 5 digits”
- Select most frequently matched

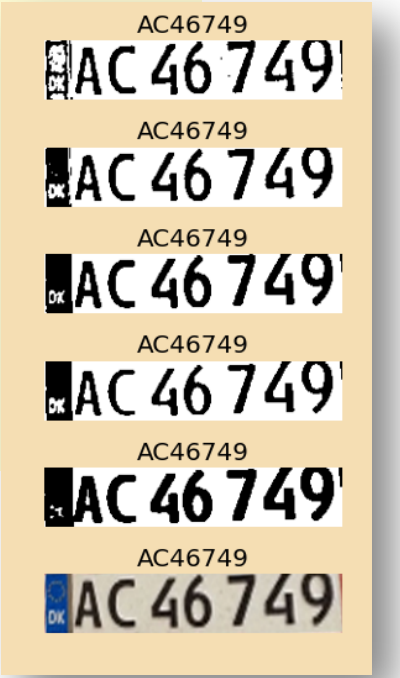




Example code: src/cars.py

Plate Filtering

```
def match_plates(title, candidate_plate_images):
    '''Returns a list of the possible matches for the plate.'''
    matches = []
    n = 0
    for cp in candidate_plate_images:
        # Filter to different representations
        n = n+1
        gray = cv2.cvtColor(cp, cv2.COLOR_BGR2GRAY)
        adaptive = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_MEAN_C,
                                         cv2.THRESH_BINARY, 11, 15)
        ret_val, th = cv2.threshold(gray, 100, 255, cv2.THRESH_BINARY)
        t_otsu, th_otsu = cv2.threshold(gray, 100, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
        ret_val, th_otsu = cv2.threshold(gray, t_otsu, 255, cv2.THRESH_BINARY)
        # thin the black parts (letters and noise)
        dilated = cv2.dilate(th_otsu, kernel=(5,5), iterations=2)
        # thicken the letters
        eroded = cv2.erode(dilated, None)
        image_variants = [cp, adaptive, th, th_otsu, dilated, eroded]
        image_matches = [m for m in map(ocr_plate, image_variants)]
        if PLOT_MATCHES:
            plot_image_variants_and_matches("%s matches %d" % (title, n),
                                            image_variants, image_matches)
        matches +=  [m for m in image_matches if m]
    return matches
```



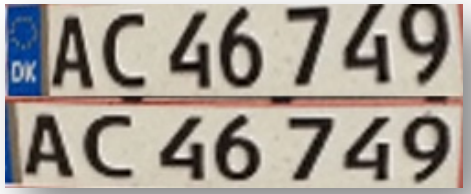
Example code: src/cars.py

OCR Code

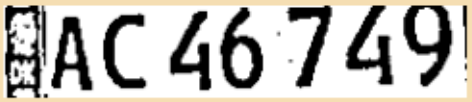
```
# --------------------------------------------------------------
# Text recognition
# --------------------------------------------------------------

def ocr_text(img):
    tr = Tesseract(lang='eng')
    tr.clear()
    pil_image = pil.Image.fromarray(img)
    # Turn off OCR word dictionaries
    tr.set_variable('load_system_dawg', "F")
    tr.set_variable('load_freq_dawg', "F")
    tr.set_variable('-psm', "7") # treat image as single line
    tr.set_variable('tessedit_char_whitelist', "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789")
    tr.set_image(pil_image)
    utf8_text = tr.get_text()
    return unicode(utf8_text)

def ocr_plate(img):
    text = ocr_text(img)
    match = re.search(r"[A-Z][A-Z] *\d{2} *\d{3}", text)
    result = None
    if match:
        result = match.group()
        # Canonicalize to no spacing
        result = result.replace(' ', '')
    return result
```

Example code: src/cars.py

Number Plate Recognition



AC46749



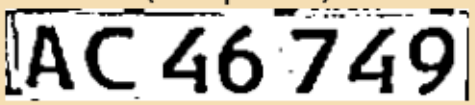
AC46749

AC46749

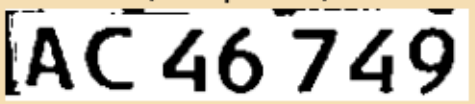
AC46749

AC46749

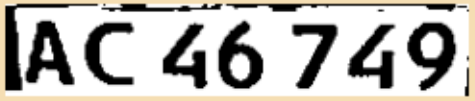
(No plate)



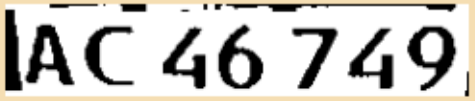
(No plate)



AC46749



AC46749



AC46749

Example code: src/cars.py

Number Plate Recognition

(No plate)

(No plate)

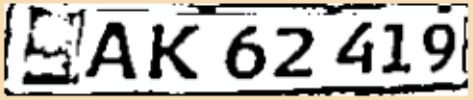
(No plate)

(No plate)

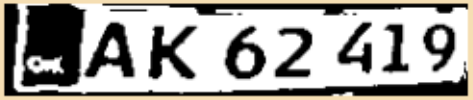
(No plate)

(No plate)

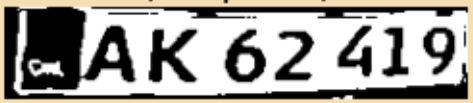
TK62419



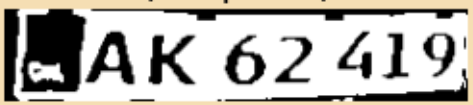
(No plate)



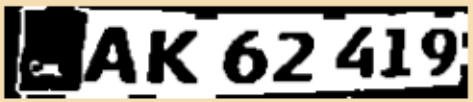
(No plate)



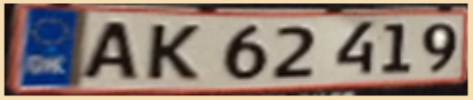
(No plate)



AK62419



(No plate)



Example code: src/cars.py

#FAIL

Feature Recognition

PIZZA

How would you do it?

Pizza time!

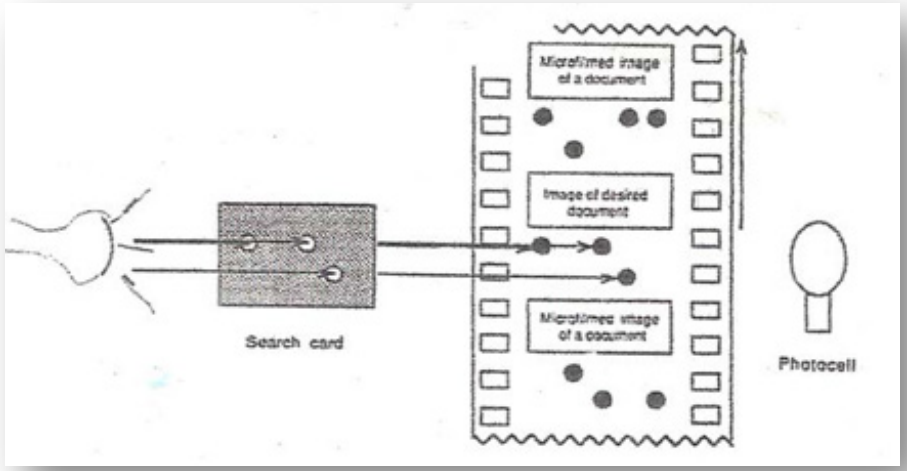
20 Questions…

- Quadrilateral found?
  - Is it dominantly white?
  - Percentage of surface
    - Black
    - Orange
    - Grey
  - Skin tones on edge of quad?
- *Is there a face shape?*
- *Is face above and near quad?*
- …

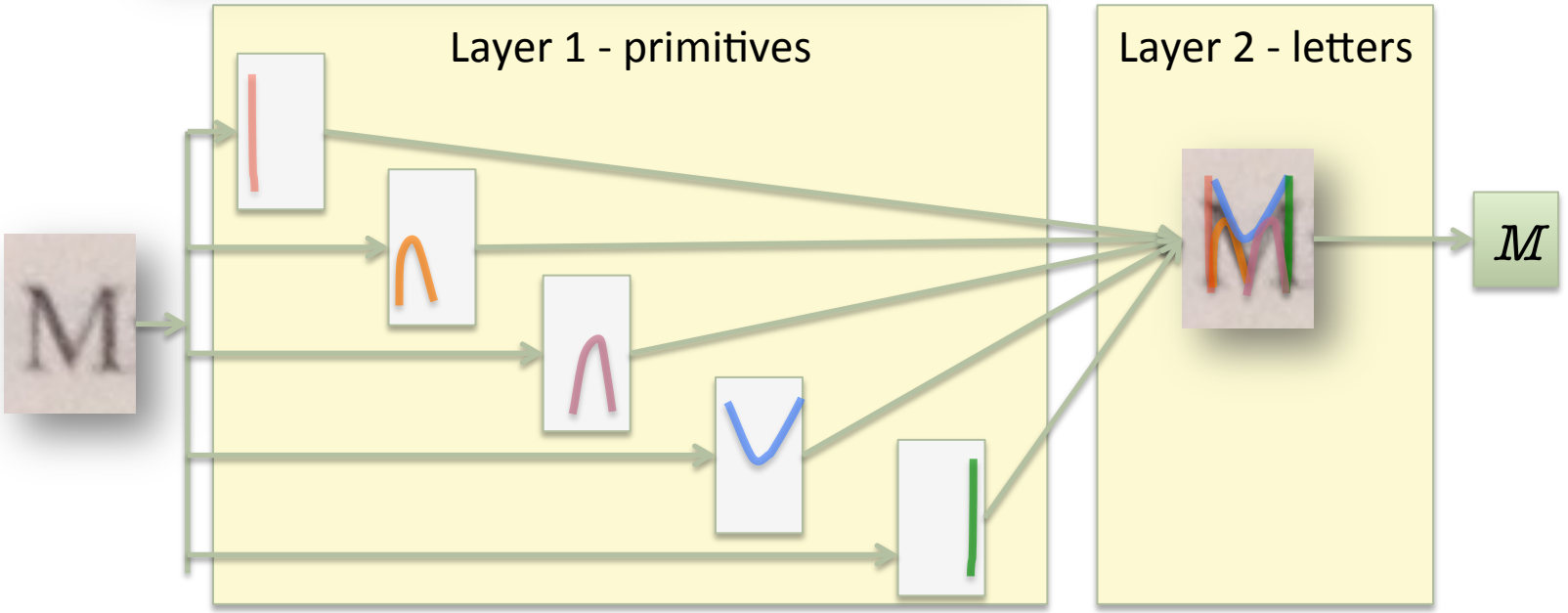
➔ Probably a pizza box

1920s Revisited

1920s Template matching

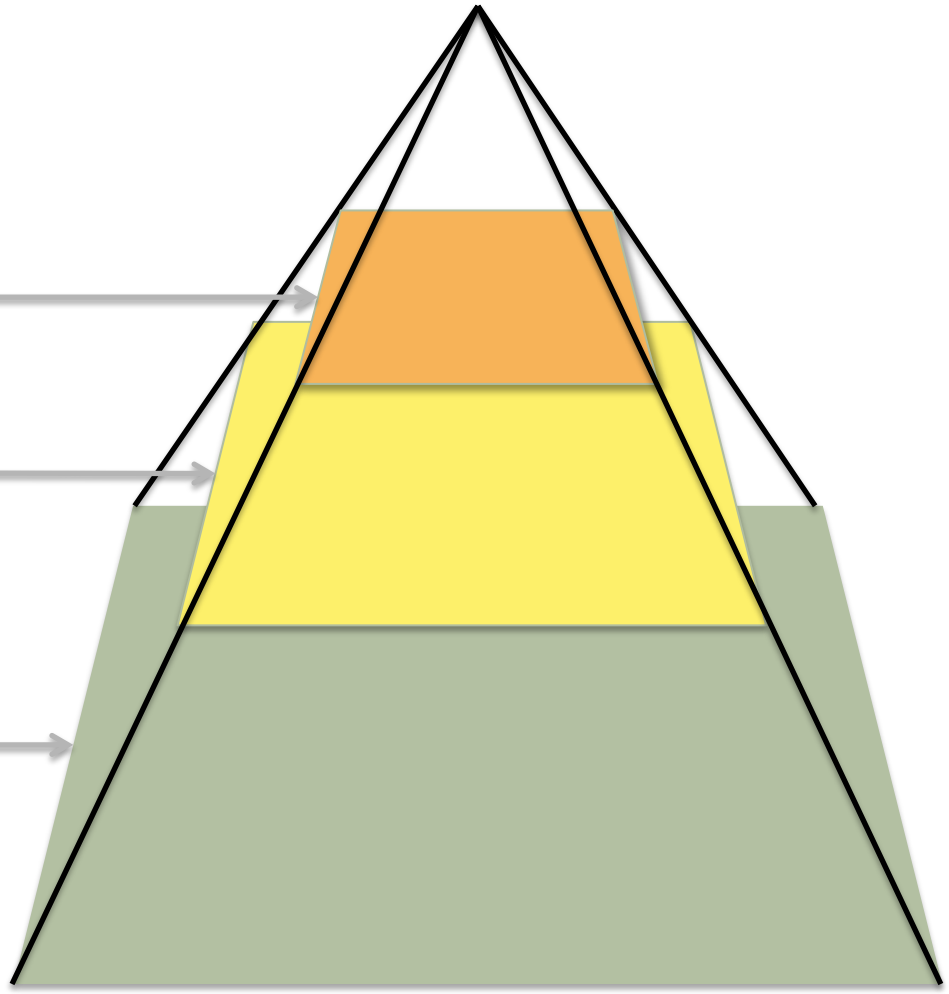


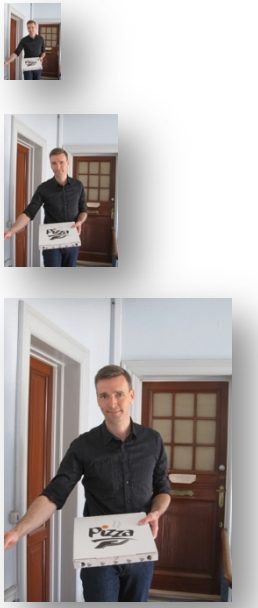
OCR Shape Matching



Known Object Detection

Scale: Pyramid or Scale-Space

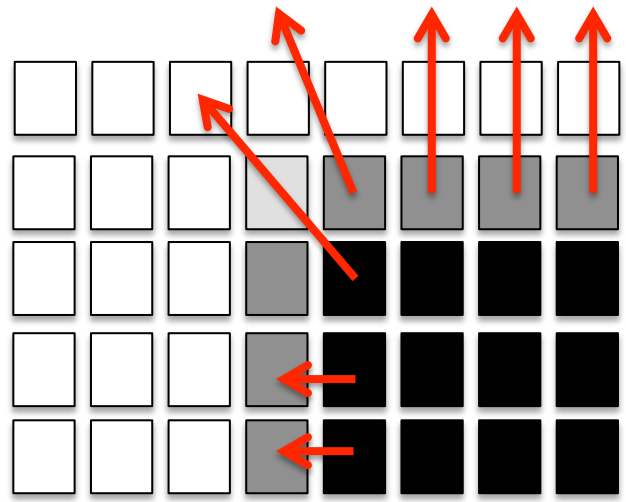


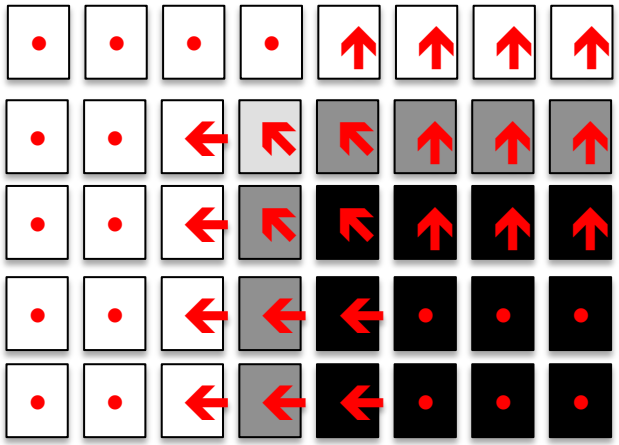


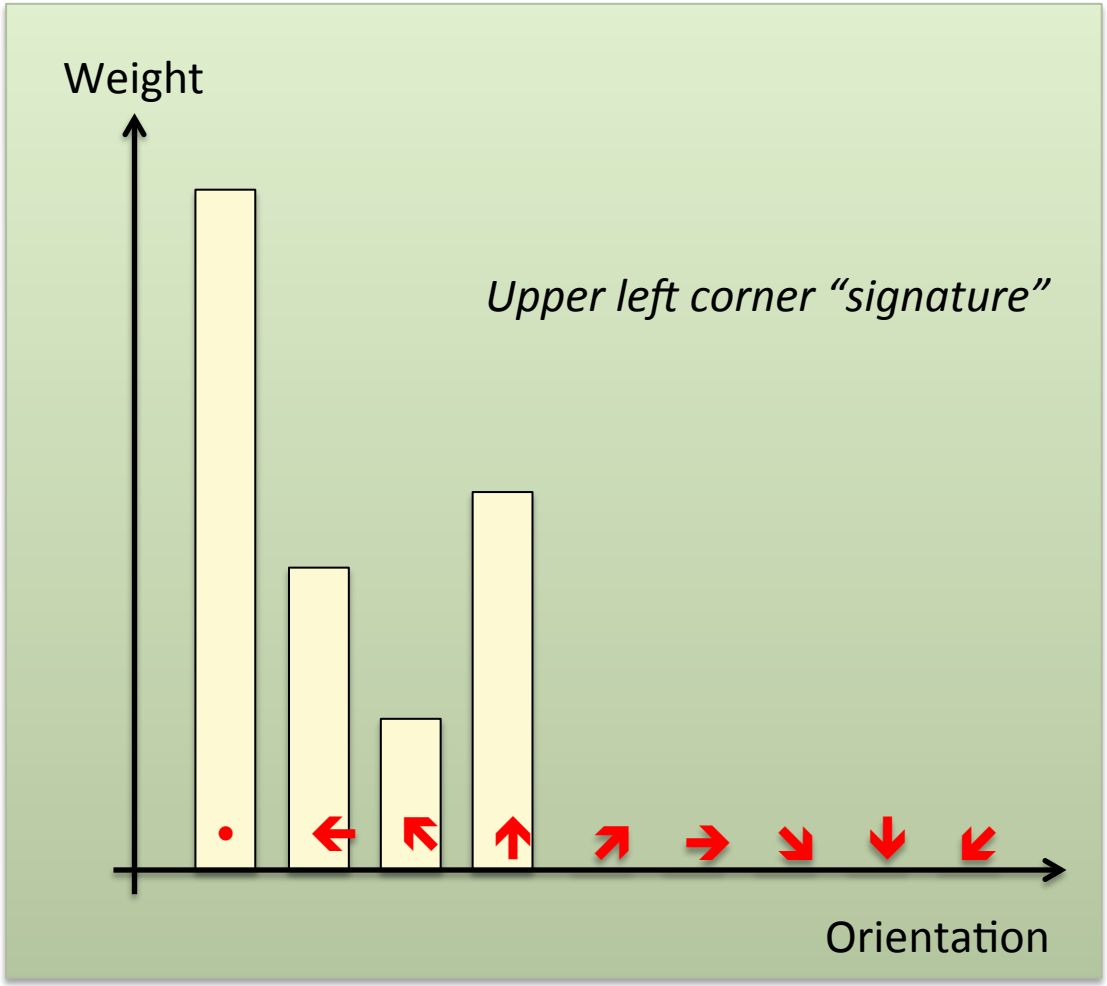
"Cheap" to repeat at various scales

Half size = ¼ pixels

HOG –
Histogram of Oriented Gradients

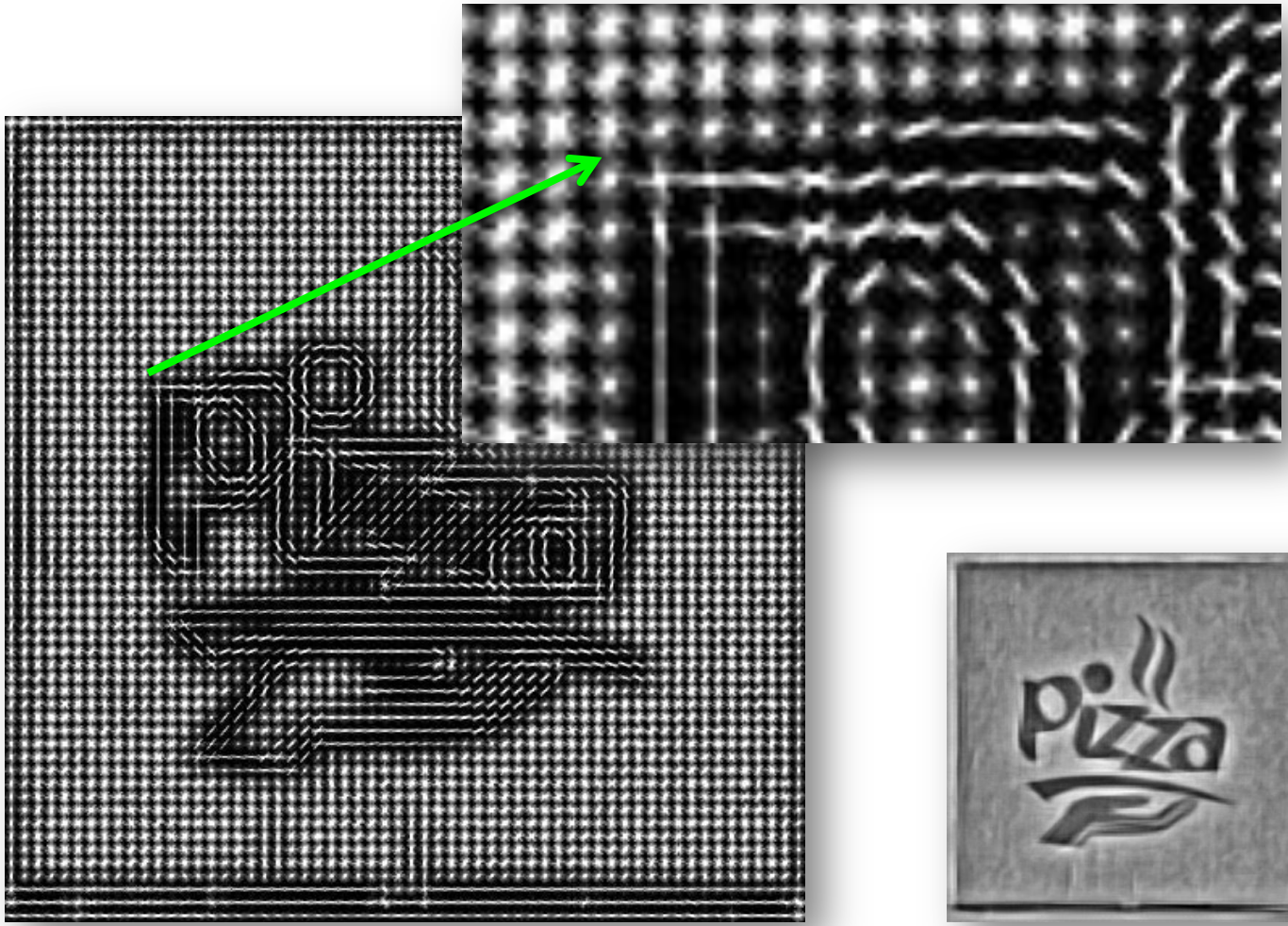






HOG Example

Input

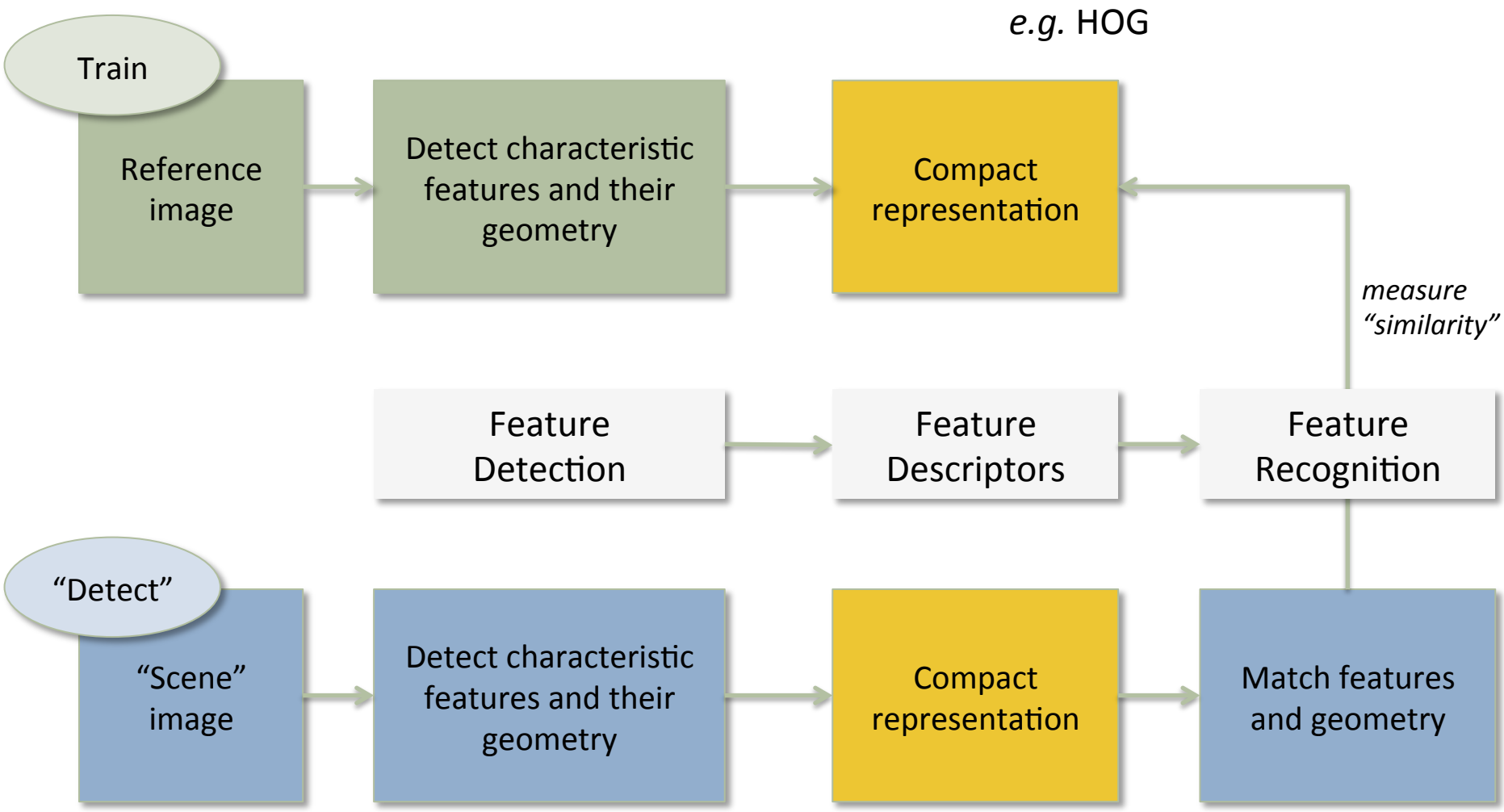


HOG

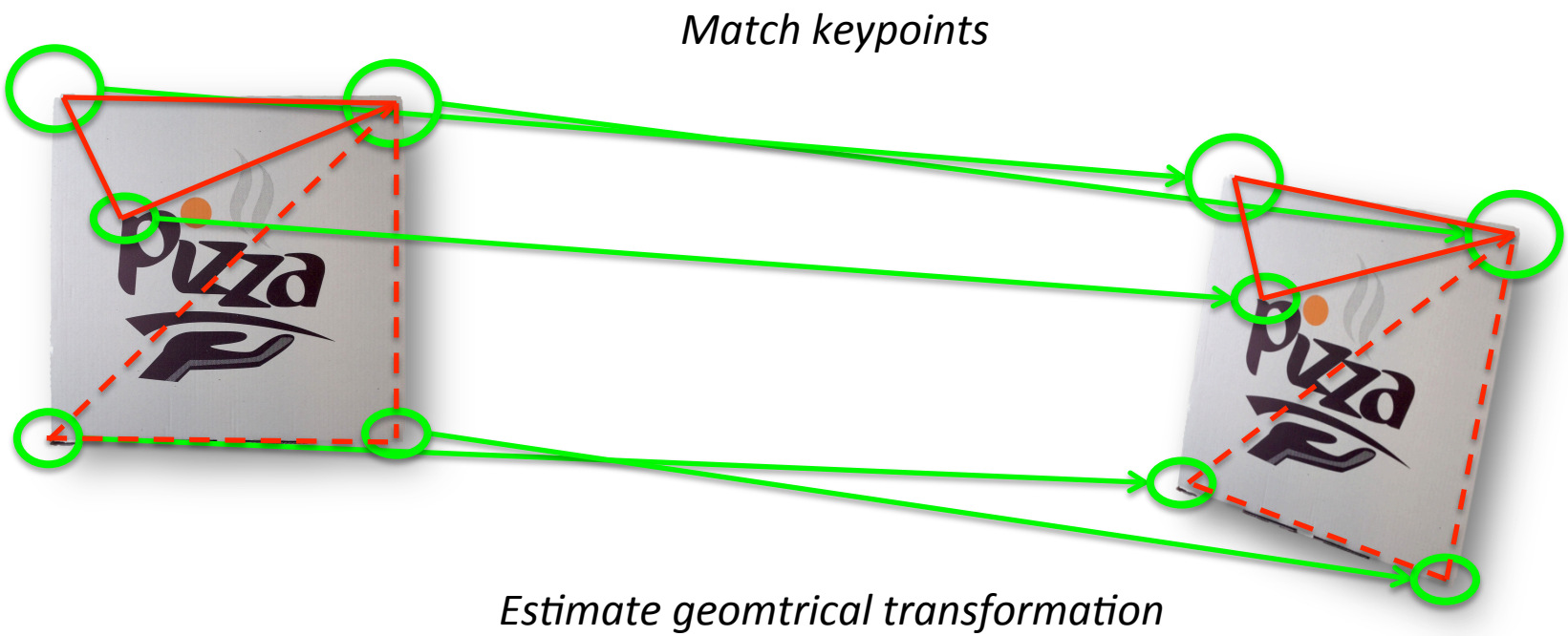
HOG reverse

Images generated with HOG visualizer at http://web.mit.edu/vondrick/ihog/index.html

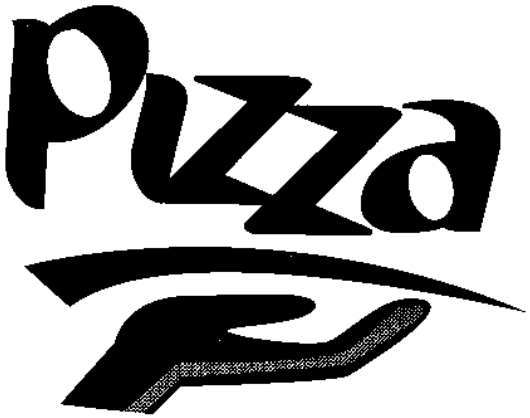
Feature Recognition



Feature Detection and Recognition



Features Detection - Keypoints



Object

Small keypoints only

All keypoints

Example code: src/pizza.py

Features Detection - Keypoints

Example code: src/pizza.py

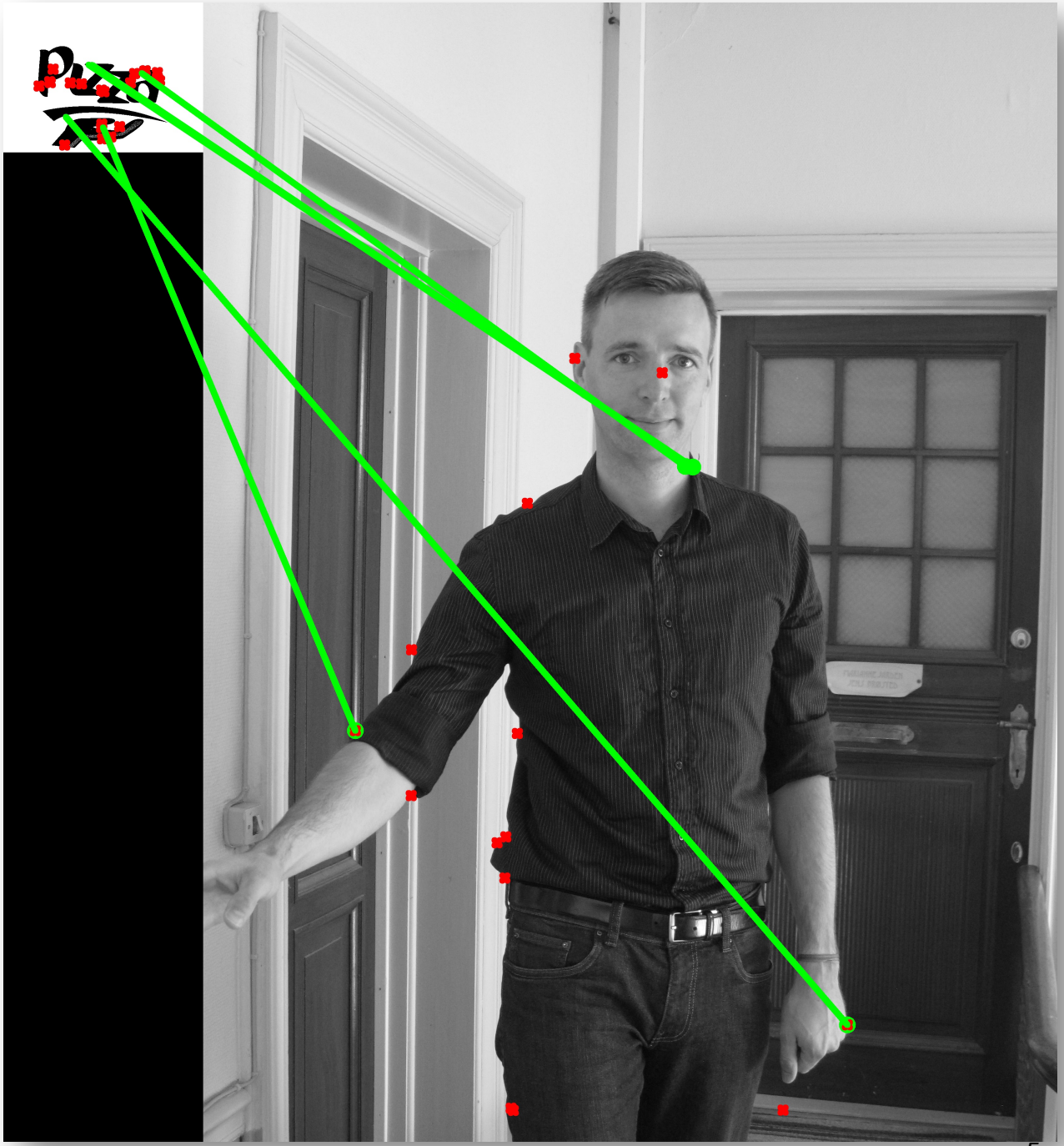
Feature Recognition

Pizza Code

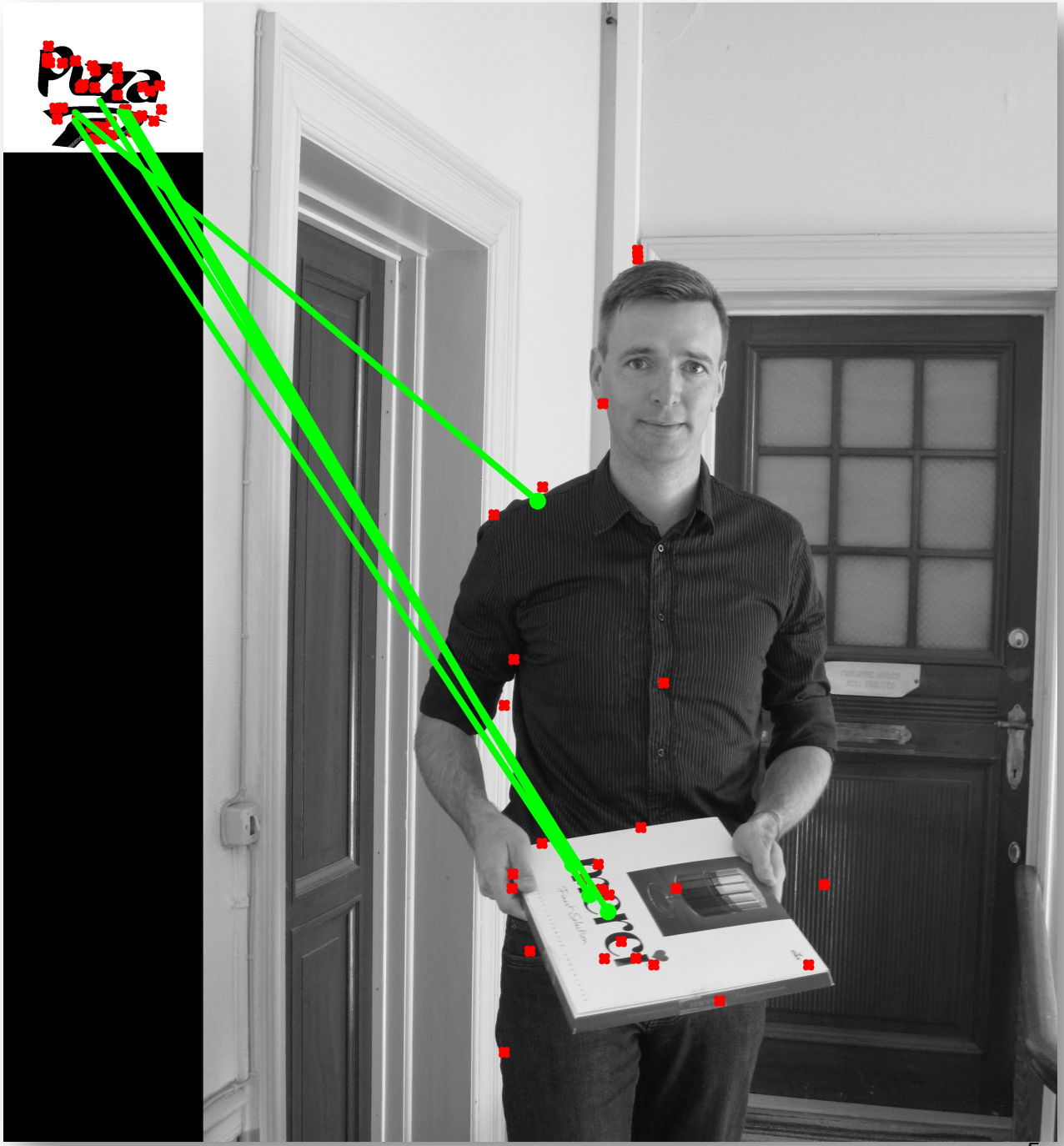
```
detector, norm = cv2.SURF(), cv2.NORM_L2
bf_matcher = cv2.BFMatcher(norm)

# Detect features in the reference object
object_keypoints, object_descriptors = detector.detectAndCompute(object_image, None)
```

```
# Detect features in the scene and match
scene_keypoints, scene_descriptors = detector.detectAndCompute(scene_image, None)

raw_matches = matcher.knnMatch(object_descriptors, scene_descriptors, k=2)
p1, p2, kp_pairs = filter_matches(object_keypoints, scene_keypoints, raw_matches)

MIN_MATCH_COUNT = 10

if len(kp_pairs)>MIN_MATCH_COUNT:
    H, status = cv2.findHomography(p1, p2, cv2.RANSAC, 5.0)
    match_lines_image = draw_matches(object_image, scene_image, kp_pairs, status, H)
    show('Match lines', match_lines_image)
```

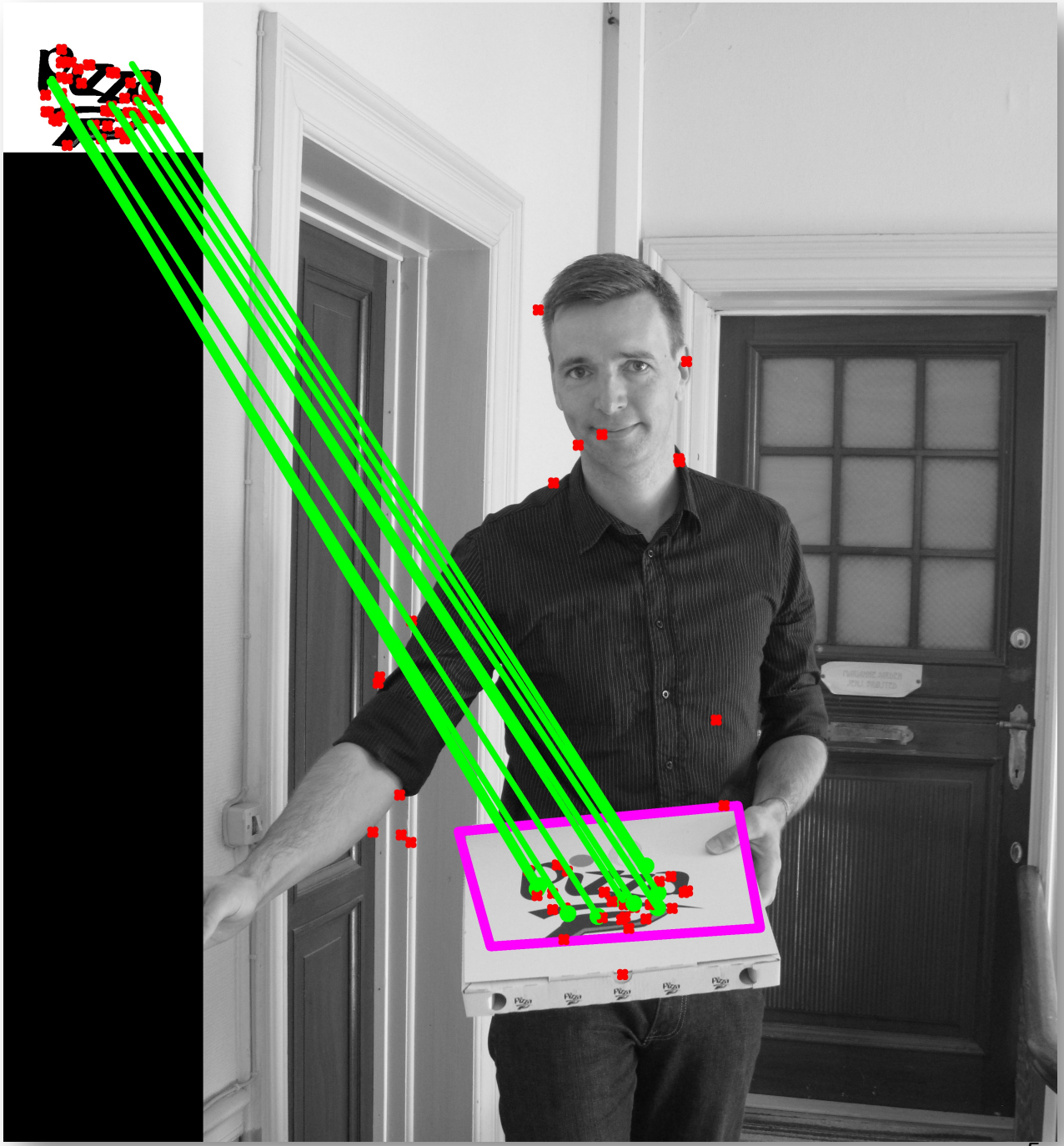
Example code: src/pizza.py



Example code: src/pizza.py

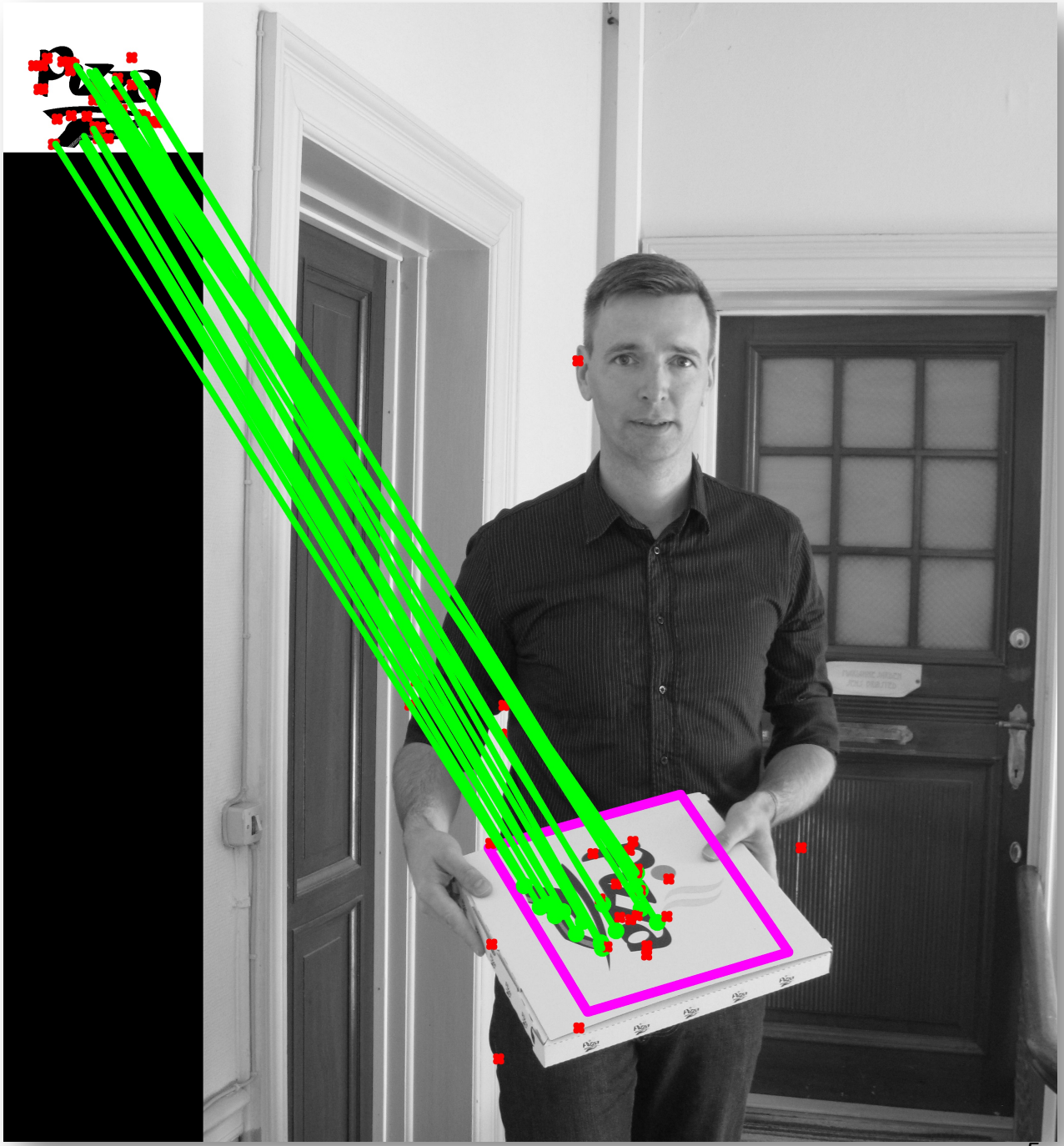


Example code: src/pizza.py



Example code: src/pizza.py

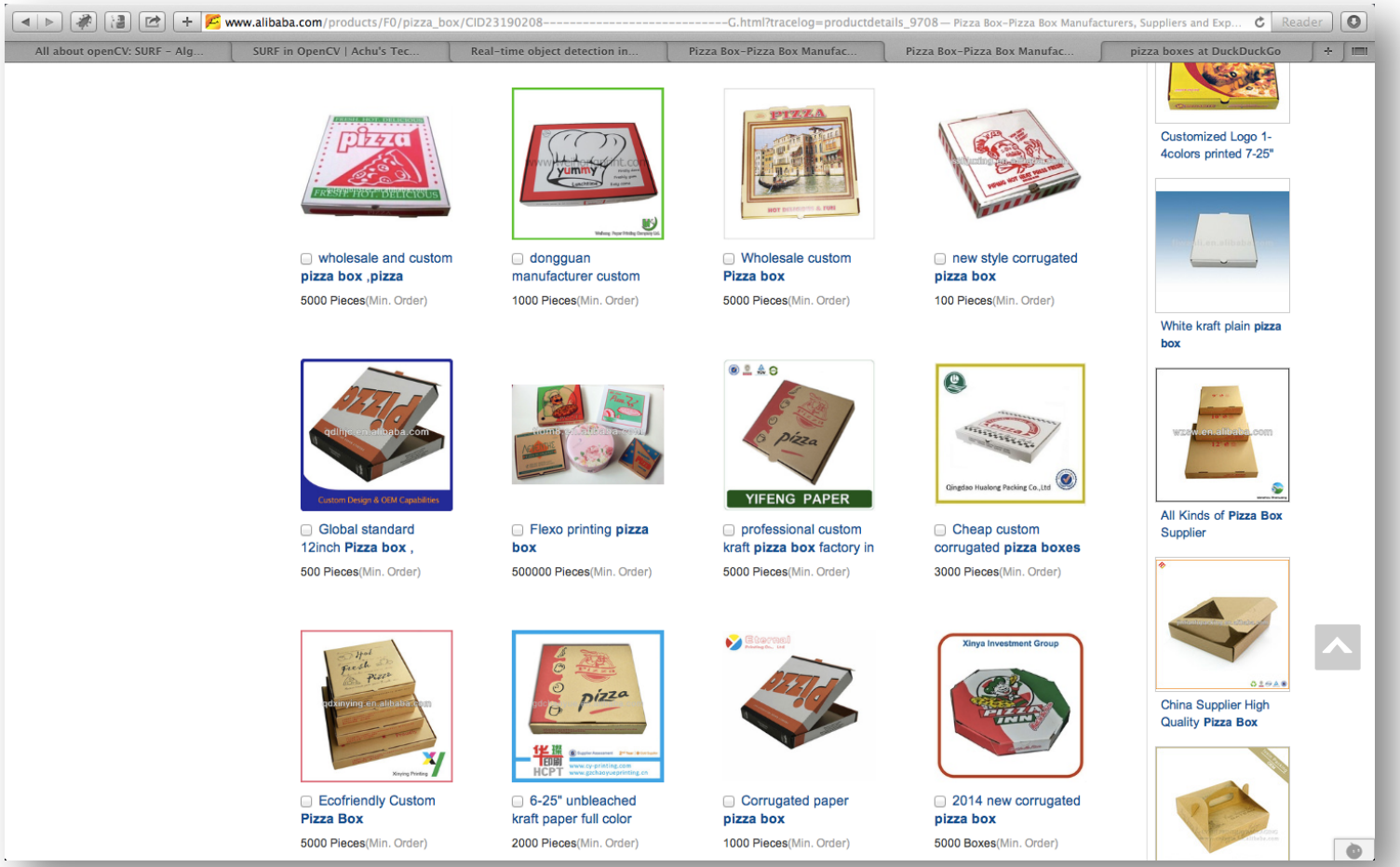
Example code: src/pizza.py



Example code: src/pizza.py

General Object Recognition

- Recognize the class of ***all possible*** pizza boxes

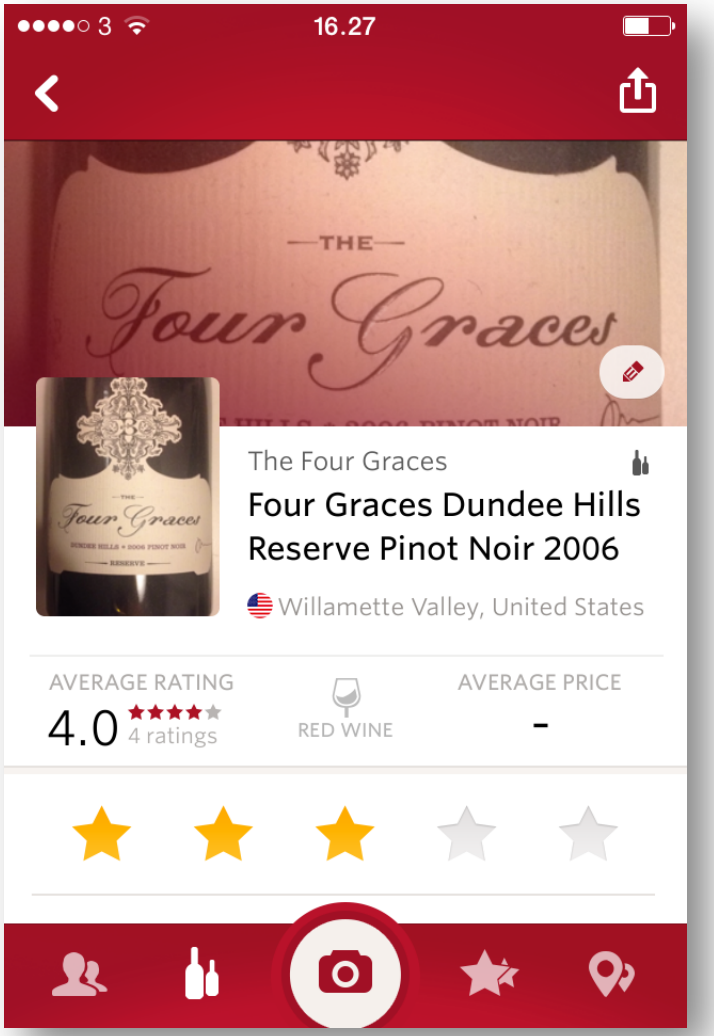


Search for "Pizza Boxhttp://www.alibaba.com/products/F0/pizza_box/CID23190208----------G.html?tracelog=productdetails_9708

Visual Queries and Machine Learning

WINE

How Would You Do It?

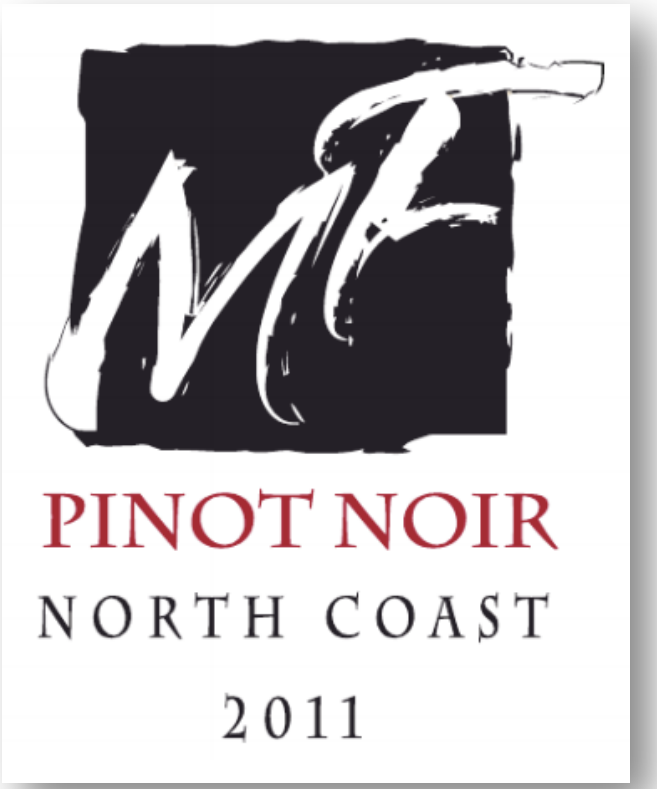


Complexities

Partial OCR "likely"

curved lines

Photo



Official Logo, MF: Matthew Fritz
From http://www.mfwine.com

Complexities

Distinct shape

Wavy text – or pattern?

OCR'able?

Artistic border
- or text?

How Would You Do It?

Example filters (feature detectors)

(Partial) OCR, *e.g.*

- Letter frequencies (language)
- Letter n-grams
- Words (may be misread/missing)

"2011"
(2,0,1,1)
(2,0,1) (0,1,1)
(2,0) (0,1) (1,1)
(2) (0) (1) (1)

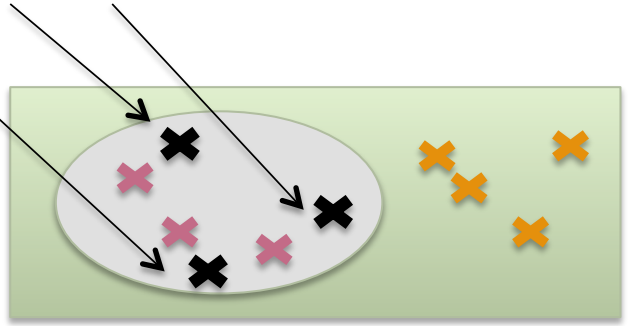
"2004"
(2,0,0,4)
(2,0,0) (0,0,4)
(2,0) (0,0) (0,4)
(2) (0) (0) (4)

Global features

e.g. Mean/dominant colours

Best match against

- Local features (Feature matching)
- Classifier (trained from known images)
- Database of producer names (known wines)
- Match - nearest neighbour



Wine Matching Pipeline

Image Space

Filter Bank

Characteristic colours

75% *10%* *1%*

Keypoints(*)

OCR letters (*)

(*) Next Stage Filter Bank

Feature Space

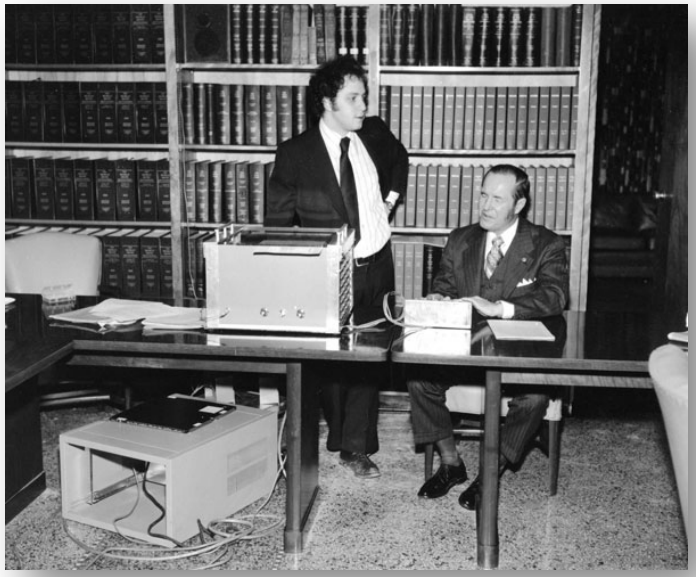
Wines to consider

Known Wines

Match against known labels

Big Data in Your Pocket

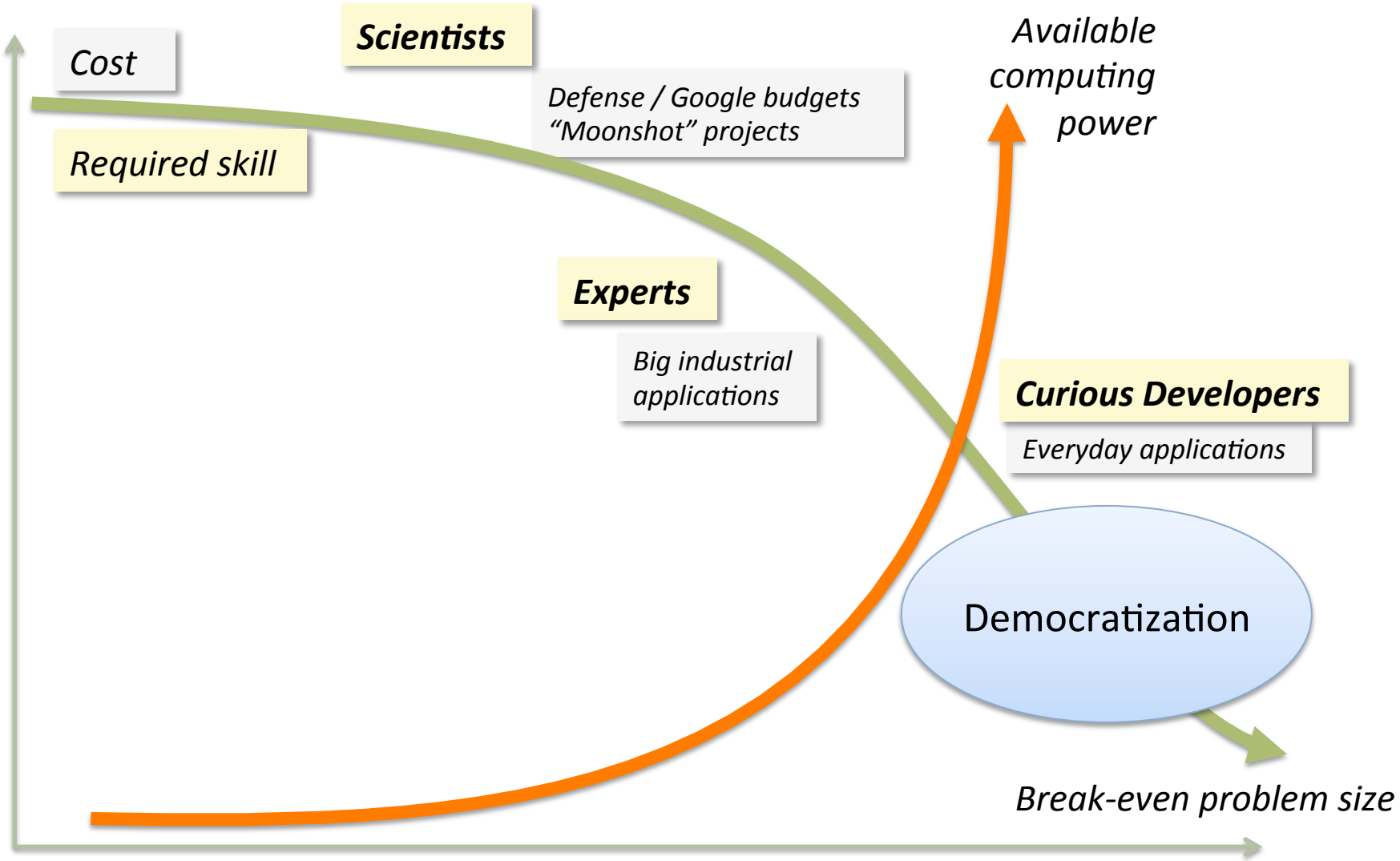
Kurzweil Reading Machine (1976)
64 KB RAM
¼ MHz CPU
Flatbed CCD Scanner
Software: Omnifont OCR, text-to-speech



January 1977 – Kurzweil unveils the Reading Machine at Iowa Commission for the Blind
http://www.iowablindhistory.org/blindhistory/tools-and-technology-optical-character-recognition-scanners

Samsung Galaxy S5 (2014)
2 GB RAM
2.5 GHz quad-core CPU
16 megapixel camera

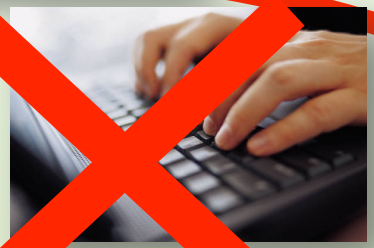
Computer Vision

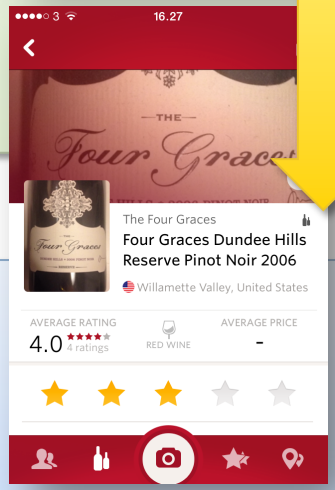


Re-imagine Everything

Human Data Processing

Unstructured Data





Four Graces
Dundas Hills Reserve
Pinot Noir
2006

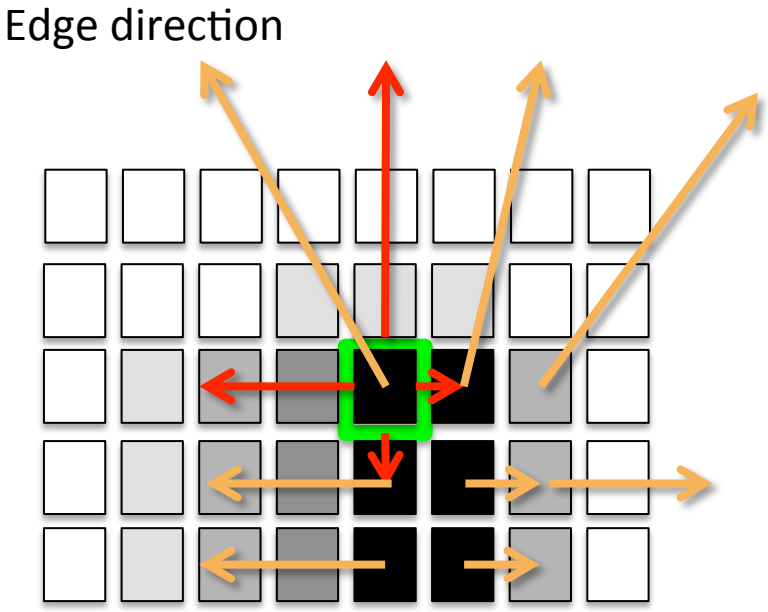
Structured Data

Machine Data Processing

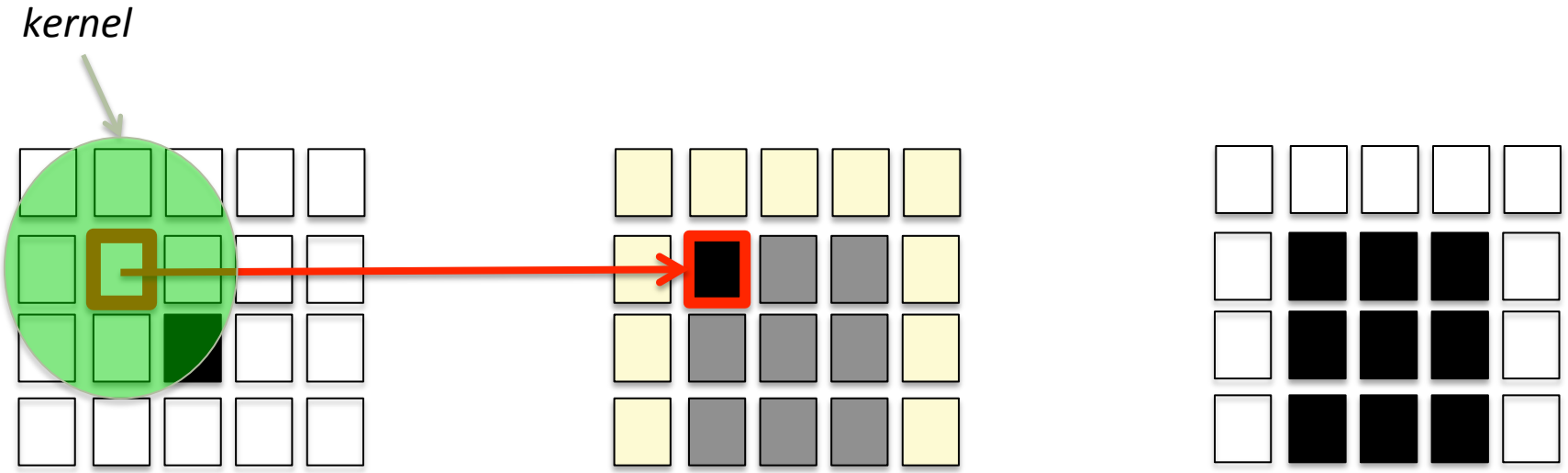
Questions?

- Get in touch!
  - Martin Jul / martin@mjul.com / @mjul
- Get the slides and the code
  - https://github.com/mjul/making-the-computer-see-ndc-2014

Canny Edge Detection

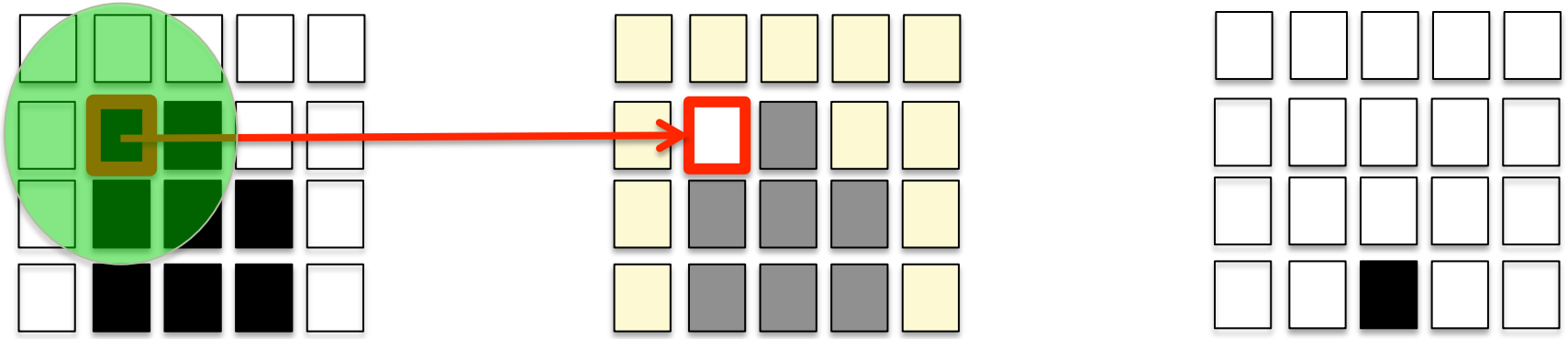


Dilation and Erosion



Erosion – *min* over kernel neighbourhood

(8-bit grayscale: white = 255, black = 0)



Dilation – *max* over kernel neighbourhood

Machine Learning Wines

Filter Bank

Statistical model

Query
Best match for these features...

GPS info
OCR words, n-grams
Distinct shapes, colours
Object recognition

Known wines same user

Typical wine near…

Database of known wines

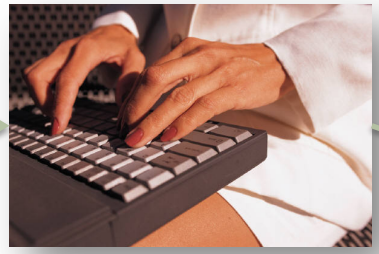
Classification

Train

Database of classified images

Machine Learning

World: Unstructured Data



"compression"

"Exact"
low redundancy

Query
where name='Foo'

Web pages
Mail
Music
Forms
Images
Maps
Scanner data
Surveillance
Quality control
…

Filter Bank

"compression"

Statistical model

Probabilistic
Representation,
Higher redundancy

Query
Best match for these features...

Learning | Classification | Recommendation | …