

Hacking the Visual World with OpenCV

Computer Vision in Python

Nicholas Dahm / Dr. Nick / Aussie Nick

TrademarkVision

Code & slides available at:

<https://github.com/das-intensity/presidential>

January 25, 2017

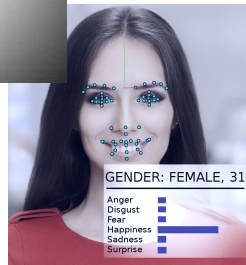
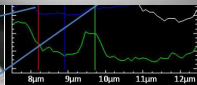
1 What is Computer Vision?

2 Example: The Presidential Look

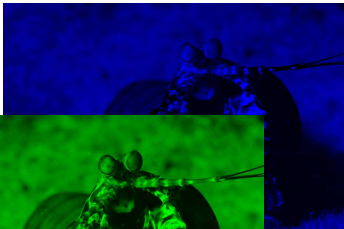
1 What is Computer Vision?

2 Example: The Presidential Look

What is Computer Vision?



What is the data?



How to Computer Vision a thing?

- ▶ **Object detection** - where are the faces in this image?
- ▶ **Object recognition** - is that face John Smith?
- ▶ **Object classification** - which animal is this?
- ▶ **Image matching** - how similar are these images?
- ▶ **Image retrieval** - what images look similar to this?
- ▶ **Object tracking** - watch where that car goes
- ▶ **Scene reconstruction** - create a 3D model from this video
- ▶ **Image restoration** - remove the wrinkle from this scanned photo

TrademarkVision - image retrieval

TrademarkVision

NEW SEARCH NICHOLAS

Add keywords / codes to search

Ranking: VC-26.15.99 VC-29.01.04 VC-29.01.06 VC-01.05 VC-29.01.95 VC-29.01.96 AU-BLUE AU-GLOBE

Search options

Image: Exact 11 Similar 34 Other 18989

0 / 19034 displaying 50 results

Monitor Report

Results: Trademarks | App Stores









































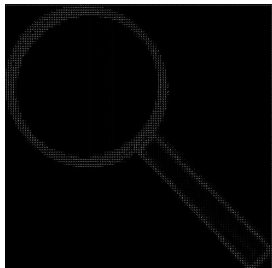
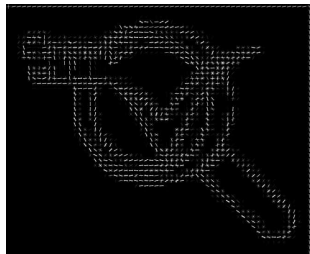
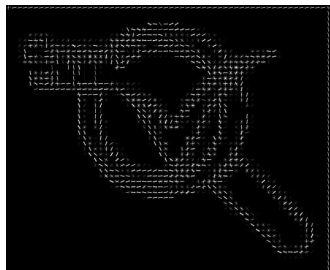
 US-85300329	 US-78757614	 US-85300327	 US-85490612	 US-86634197	 US-78757492	 US-86634181	 US-85548247	 AU-1734641	 AU-1734643
 EU-4891371	 NZ-826296	 US-85008587	 NZ-743235	 EU-14792469	 AU-1369087	 NZ-760700	 AU-1099868	 EU-14792444	 EU-9282881
 EU-12473765	 EU-12473757	 CA-1486469	 EU-15021884	 EU-15021926	 US-87271572	 US-87005614	 NZ-1035657	 AU-1745966	 NZ-1035655
 AU-1745969	 CA-1769244	 US-79062681	 US-85474105	 US-85474140	 EU-2663789	 EU-894626	 US-85474336	 US-75004683	 US-85474056

Image features - HOG



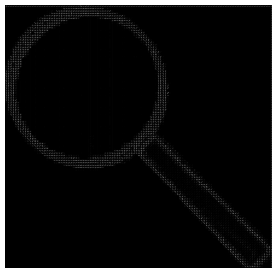
Comparing image features



→ (0.20, 0.04, 0.07, 0.12, ..., 0.11)

Diff = (0.07, 0.14, 0.06, 0.08, ..., 0.01)

$$L_1 = 0.52$$



→ (0.13, 0.18, 0.01, 0.04, ..., 0.12)

Computer Vision in Python

What Computer Vision tools are available for us in Python?

► **OpenCV**

- Large library of computer vision functions
- Many algorithms for all types of tasks
- Python wrapper for underlying C++ library
- Stores images as NumPy arrays for easy manipulation

► **NumPy**

- Extensive matrix manipulation library
- Incredibly fast when using matrix operations

► **PIL**

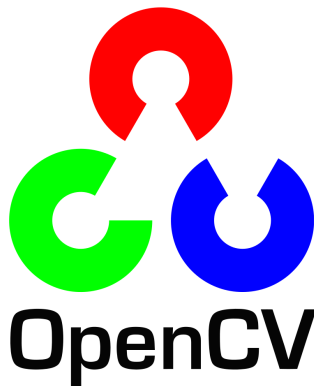
- Python Imaging Library (or newer Pillow fork)
- Various image operations, including some not in OpenCV
- Quick translations to/from OpenCV's NumPy format

1 What is Computer Vision?

2 Example: The Presidential Look

Prerequisites

- ▶ Python (2.7 recommended)
- ▶ OpenCV (3 recommended)
- ▶ NumPy
- ▶ PIL (optional)



A Wild Squirrel Appears!

```
import cv2
import numpy as np

# Download goo.gl/nXaoEf as squirrel.png

sq = cv2.imread('squirrel.png')
print sq.shape # (140, 160, 3)
cv2.imshow('squirrel', sq)

sq_gray = cv2.cvtColor(sq, cv2.COLOR_BGR2GRAY)
print sq_gray.shape # (140, 160)
cv2.imshow('squirrel gray', sq_gray)
cv2.waitKey()
```



- ▶ You used computer vision... it's super effective!

OpenCV Squirrel Windows



PIL <-> OpenCV

```
from PIL import Image
pilsq = Image.open('squirrel.png')

# PIL -> cv2
sq2 = np.asarray(pilsq)
sq2 = cv2.cvtColor(sq2, cv2.COLOR_RGB2BGR)

# cv2 -> PIL
pilsq2 = cv2.cvtColor(sq2, cv2.COLOR_BGR2RGB)
pilsq2 = Image.fromarray(pilsq2)
```

I wanna see myself!

```
print 'starting webcam...'
cam = cv2.VideoCapture(0)
ret, frame = cam.read()
assert ret # fails if couldn't read from webcam
print 'webcam res:', frame.shape

while True:
    ret, frame = cam.read()

    cv2.imshow('cam', frame)
    key = cv2.waitKey(1)
    if key != -1:
        break
```

Hi everybody!



I still want the squirrel though!

```
sq_gray_mask = sq_gray > 0
sq_mask = np.array(
    [[v, v, v] for v in row] for row in sq_gray_mask
)
```

```
while True:
    ret, frame = cam.read()

    # overlay the squirrel
    np.copyto(
        frame[-sq.shape[0]:, 0:sq.shape[1], :],
        sq,
        where=sq_mask
    )

    cv2.imshow('cam', frame)
```

A “Squirrel Productions” film



Detecting faces

```
cas = cv2.CascadeClassifier(  
    '/usr/share/opencv/haarcascades/'  
    + 'haarcascade_frontalface_default.xml'  
    )  
  
### inside while loop, after cam.read() ###  
frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)  
faces = cas.detectMultiScale(  
    frame_gray,  
    scaleFactor=1.1,  
    minNeighbors=5,  
    minSize=(30, 30),  
    flags = cv2.CASCADE_SCALE_IMAGE  
    # OpenCV 2.4 -> cv2.cv.CV_HAAR_SCALE_IMAGE  
    )  
# faces is list of (x, y, width, height)
```

Drawing faces

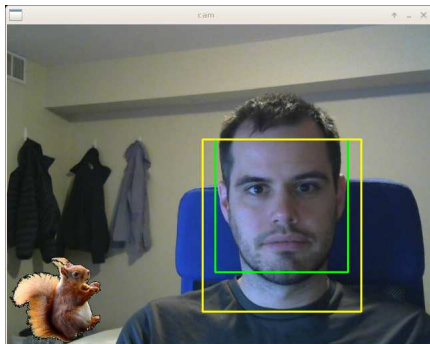
```
for (x, y, w, h) in faces:
    cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

    # Make the box a bit bigger (see why later)
    #- first save originals
    x0 = x; y0 = y; w0 = w; h0 = h

    h += int(h * 0.3)
    x -= int(w * 0.1)
    w += int(w * 0.2)
    if y < 0 or y+h > frame.shape[0]: continue
    if x < 0 or x+w > frame.shape[1]: continue

    cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 255),
        2)
```

FaceBox.. the latest social media craze



Find the non-presidential skin...

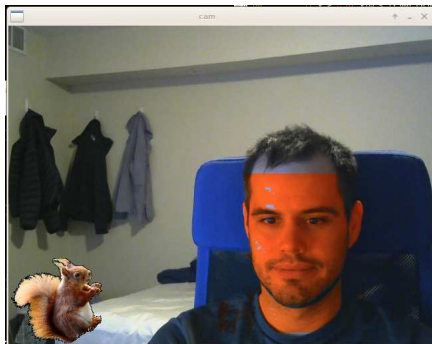
```
### inside faces loop ###
# Find the skin pixels
frame_hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
face_hsv = frame_hsv[y:y+h, x:x+w, :]

hsv_min = np.array([0, 0, 40])
hsv_max = np.array([200, 120, 220])
face_inrange = cv2.inRange(face_hsv, hsv_min, hsv_max)
face_inrange = np.array([[v, v, v] for v in row] for
    row in face_inrange), dtype=bool)
#print 'face_inrange: %s - %s' % (face_inrange.shape,
    face_inrange.dtype)
```

Upgrade it to a more presidential colour

```
### inside faces loop ###
# Set a more presidential color
pres_skin = [0.2, 0.65, 1.7]
face_bgr = frame[y:y+h, x:x+w, :] * pres_skin
face_bgr = np.minimum(face_bgr, 255)
face_bgr = np.array(face_bgr, dtype='uint8')
#print 'face_bgr: %s - %s' % (face_bgr.shape,
    face_bgr.dtype)
np.copyto(frame[y:y+h, x:x+w, :], face_bgr,
    where=face_inrange)
```

Orange is the presidential colour



Let get some presidential hair!

```
### inside faces loop ###
# Obtain some presidential hair
hair = sq[30:,0:70]
hair = np.rot90(hair, 3)
hair_mask = sq_mask[30:,0:70]
hair_mask = np.rot90(hair_mask, 3)

hair_scale = w0 / float(hair.shape[1])

hair = cv2.resize(hair, None, fx=hair_scale,
                  fy=hair_scale)
```

Wear the presidential hair!

```
### inside faces loop ###
# Wear the presidential hair!
hair_mask = np.array(hair_mask, dtype='uint8')
hair_mask = cv2.resize(hair_mask, None, fx=hair_scale,
                       fy=hair_scale)
hair_mask = np.array(hair_mask, dtype=bool)

hair = np.array(np.minimum((hair * 0.7) + 100, 255),
                dtype='uint8') # Optional

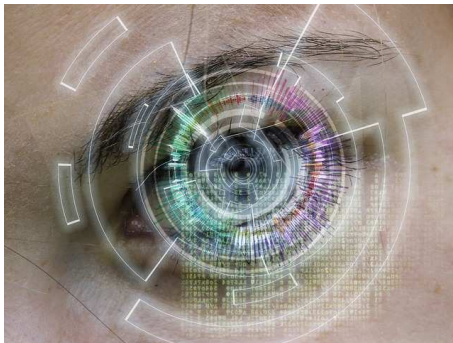
if y0 - hair.shape[0] < 0: continue
np.copyto(frame[y0-hair.shape[0]:y0,
               x0:x0+hair.shape[1], :], hair, where=hair_mask)
```

The most presidential look, period!



Thanks for listening!

Never let the visual reality get in your way again!



github.com/das-intensity
[linkedin.com/in/nicholasdahm](https://www.linkedin.com/in/nicholasdahm)
[meetup.com/members/195619601](https://www.meetup.com/members/195619601)

Want to Computer some Visions?



- ▶ **Computer Vision Interns**
- ▶ **Experienced Python Django Developer**
- ▶ **DevOps & Data Engineer**

Contact: ask@trademark.vision

Info: trademark.vision/about/trademarkvision-careers