Partage de connaissance

Cristian Perez

28 février 2017

Anatomy of an image classifier

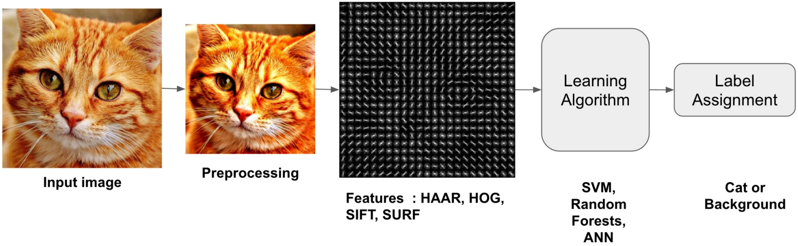
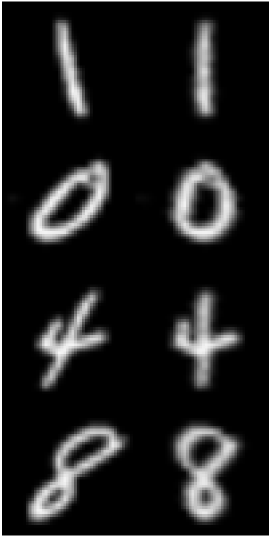


Figure 1:

Preprocessing

Deskew

Align an image to a reference assits the classification algorithm 1, 2.



Deskewing simple grayscale images can be achieved using image moments (distance and intensity of pixels).

```
def deskew(img):
    m = cv2.moments(img)
    if abs(m['mu02']) < 1e-2:
        # no deskewing needed.
        return img.copy()
    # Calculate skew based on central momemts.
    skew = m['mu11']/m['mu02']
    # Calculate affine transform to correct skewness.
    M = np.float32([[1, skew, -0.5*SZ*skew], [0, 1, 0]])
    # Apply affine transform
    img = cv2.warpAffine(img, M, (SZ, SZ), flags=cv2.WARP_I
    return img
```

Not that easy for fishes

Figure 3:

Histogram equalization

Increase image contrast using the image's histogram.

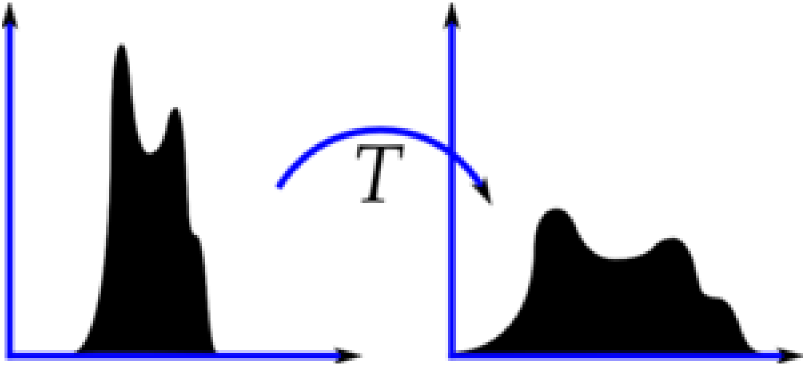


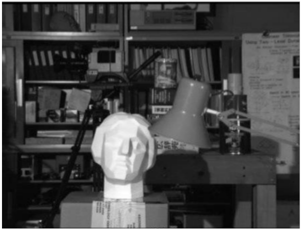
Figure 4:

Palette change by a transformation function which maps the input pixels in brighter region to the output pixels in full region.

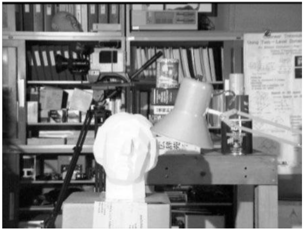
```
img = cv2.imread('wiki.jpg',0)
equ = cv2.equalizeHist(img)
res = np.hstack((img,equ)) #stacking images side-by-side
cv2.imwrite('res.png',res)
```

- Histogram equalization considers the global contrast of the image
- The background contrast improves after histogram equalization, but the face of statue lost most of the information there due to over-brightness.

Original image



After global histogram equalization



1

Adaptive Histogram Equalization

- Histogram is equalized inside blocks.
- Histogram would confine to a small region (unless there is noise).
- If noise is there, it will be amplified. To avoid this, contrast limiting is applied.
- If any histogram bin is above the specified contrast limit (by default 40 in OpenCV), those pixels are clipped and distributed uniformly to other bins before applying histogram equalization.
- After equalization, to remove artefacts in tile borders, bilinear interpolation is applied.

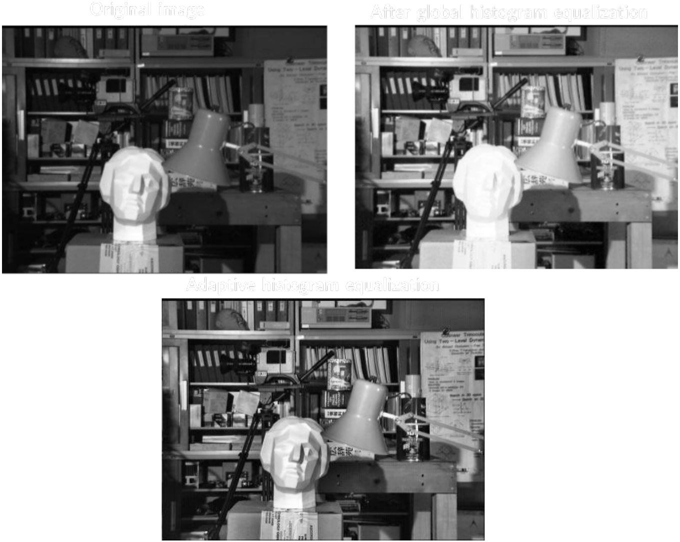

Original image
After global histogram equalization
Adaptive histogram equalization

Figure 6:

Example using fishes

Gray scale, doc histrogram opencv.

```
# Read image
src = cv2.imread("img_07473.jpg", cv2.IMREAD_GRAYSCALE);
hist = cv2.calcHist(src,[0],None,[256],[0,256])
# Histogram equalization
equ = cv2.equalizeHist(src)
equ_hit = cv2.calcHist(equ,[0],None,[256],[0,256])
# Create a AdaptativeHistogramEqualization object
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(4,4))
cl1 = clahe.apply(src)
cl1_hist = cv2.calcHist(cl1,[0],None,[256],[0,256])
```

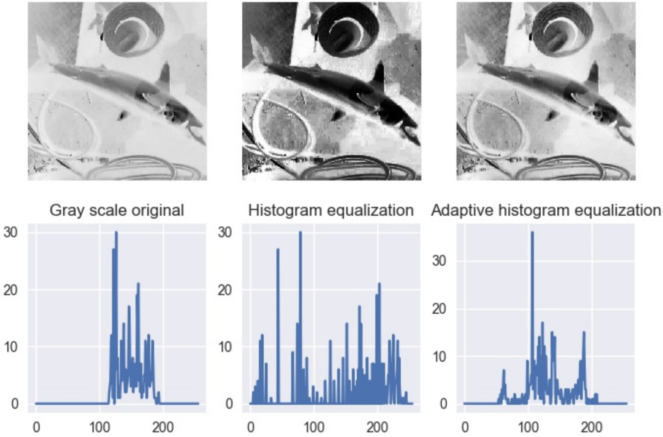

Gray scale original
Histogram equalization
Adaptive histogram equalization
30
20
10
0
0
100
200

Figure 7:

Histogram equalization for color images

- Histogram equalization is a nonlinear process.
- The concept of histogram equalization is only applicable to the intensity values in the image.
- Convert it to a color space where intensity is separated from the color information (i.e. **YUV** color space)
- Equalize the Y-channel and combine it with the other two channels.

```
# Read image
img = cv2.imread('img_07473.jpg')
img_yuv = cv2.cvtColor(img, cv2.COLOR_BGR2YUV) # transform
hist = cv2.calcHist(img_yuv,[0],None,[256],[0,256]) # hist
# equalize the histogram of the Y channel
img_yuv[:,:,0] = cv2.equalizeHist(img_yuv[:,:,0])
# convert the YUV image back to RGB format
img_output = cv2.cvtColor(img_yuv, cv2.COLOR_YUV2RGB)
equ_hit = cv2.calcHist(img_output,[0],None,[256],[0,256])
# Create a CLAHE object (Arguments are optional).
clahe = cv2.createCLAHE(clipLimit=10.0, tileGridSize=(4,4))
img_yuv = cv2.cvtColor(img, cv2.COLOR_BGR2YUV)
img_yuv[:,:,0] = clahe.apply(img_yuv[:,:,0])
cl1 = cv2.cvtColor(img_yuv, cv2.COLOR_YUV2RGB)
cl1_hist = cv2.calcHist(img_yuv,[0],None,[256],[0,256])
```

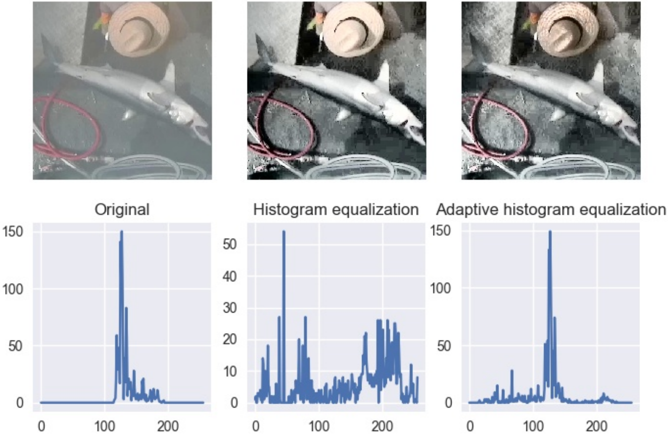
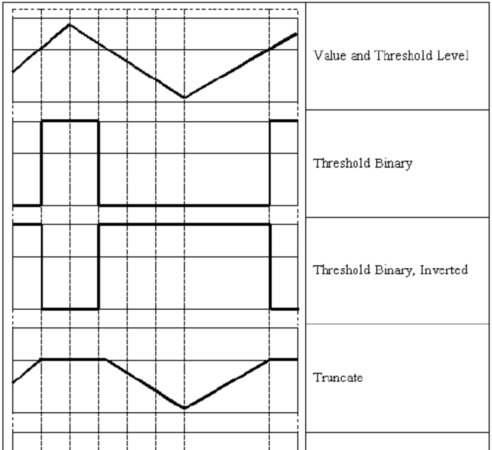

Original
Histogram equalization
Adaptive histogram equalization
150
100
50
0
0
100
200
50
40
30
20
10
0
0
100
200
150
100
50
0
0
100
200

Figure 8:

Image thresholding

- Method of image segmentation
- If pixel value is greater than a threshold value, it is assigned one value, else it is assigned another value.



```
# Thresholding with threshold value set 127
th, dst = cv2.threshold(src,127,255, cv2.THRESH_BINARY);
cv2.imwrite("opencv-thresh-binary.jpg", dst);

# Thresholding using THRESH_TOZERO
th, dst = cv2.threshold(src,127,255, cv2.THRESH_TOZERO);
cv2.imwrite("opencv-thresh-tozero.jpg", dst);

# Thresholding using THRESH_TOZERO_INV
th, dst = cv2.threshold(src,127,255, cv2.THRESH_TOZERO_INV
cv2.imwrite("opencv-thresh-to-zero-inv.jpg", dst);
```

Adaptive thresholding

- The algorithm calculate the threshold for a small regions of the image.
- Different thresholds for different regions of the same image
- Gives better results for images with varying illumination.

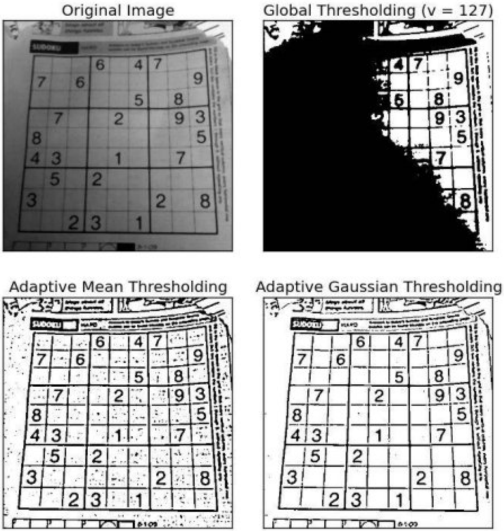

Original Image
Global Thresholding (v = 127)
Adaptive Mean Thresholding
Adaptive Gaussian Thresholding

Figure 10:

Otsu's Binarization

- Automatically finds a threshold value which lies in between two peaks such that variances to both classes are minimum
- Otsu

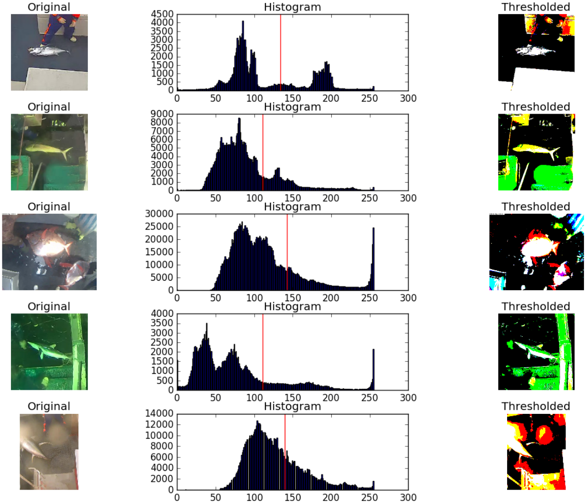


Figure 11:

Feature Extraction

Understanding features

Find the exact location of these patches in the original image.

Feature definition

- Piece of information which is relevant for solving the computational task related to a certain application.
- Specific structures in the image such as points, edges or objects.
- The result of a general neighborhood operation or feature detection applied to the image.
- Concept is very general and the choice of features in a particular computer vision system may be highly dependent on the specific problem at hand.

Feature extractor

- A feature descriptor is a representation of an image that simplifies the image by extracting useful information and throwing away extraneous information.
- A feature descriptor converts an image of size `width x height x 3` (channels) to a feature vector. (For HOG, the input image is of size `64 x 128 x 3` and the output feature vector is of length 3780)

Scale-Invariant Feature Transform (SIFT)

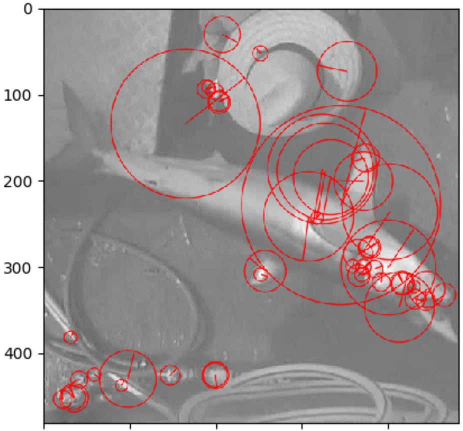
- Extract keypoints and compute its descriptor
- Invariant to uniform scaling, orientation and illumination changes
- Orientation is assigned to each keypoint to achieve invariance to image rotation
- Descriptors are vectors of 128 values, calculated from orientation histogram over the neighbourhood, docs.opencv.

```
img = cv2.imread('img_00898.jpg', 0)
sift = cv2.xfeatures2d.SIFT_create()
kp = sift.detect(img)
img2 = cv2.drawKeypoints(img,kp,None,(255,0,0),4)
plt.imshow(img2)
plt.savefig("sift.png")
```

Speeded Up Robust Features (SURF)

- In 2006, it is a speeded-up version of SIFT.
- Rely on determinant of Hessian matrix for both scale and location.

```
img = cv2.imread('img_07473.jpg',0)
surf.setHessianThreshold(1000)
kp, des = surf.detectAndCompute(img,None)
img2 = cv2.drawKeypoints(img,kp,None,(255,0,0),4)
```



Histogram of Oriented Gradients (HOG)

- The distribution of directions of gradients are used as features
- Gradients of an image are useful because the magnitude of gradients is large around edges and corners
- The gradient removes a lot of non-essential information (e.g. constant colored background)

```
# Calculate gradient
gx = cv2.Sobel(im, cv2.CV_32F, 1, 0, ksize=1)
gy = cv2.Sobel(im, cv2.CV_32F, 0, 1, ksize=1)
# Python Calculate gradient magnitude and direction ( in de
mag, angle = cv2.cartToPolar(gx, gy, angleInDegrees=True)
```

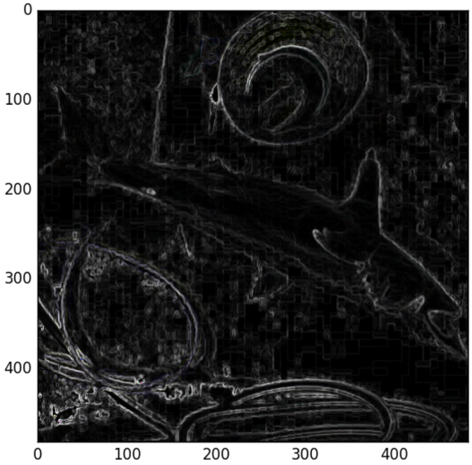


Figure 15:

Object detection

Libraries

- Dlib Object_detector
- Opencv Cascade Classfier
- Deep learning

Conclusions

- Image preprocessing can significantly increase the performance of a classification algorithm.
- A feature descriptor represents a simplified version of an image by extracting useful information and throwing away extraneous information.
- Using feature description increases training speed compared with raw images.