고려대학교
KOREA UNIVERSITY

Pixel 히스토그램 / 기타 학습 내용

PARALELL ALGORITHM LAB

Mail

고려대학교

목차

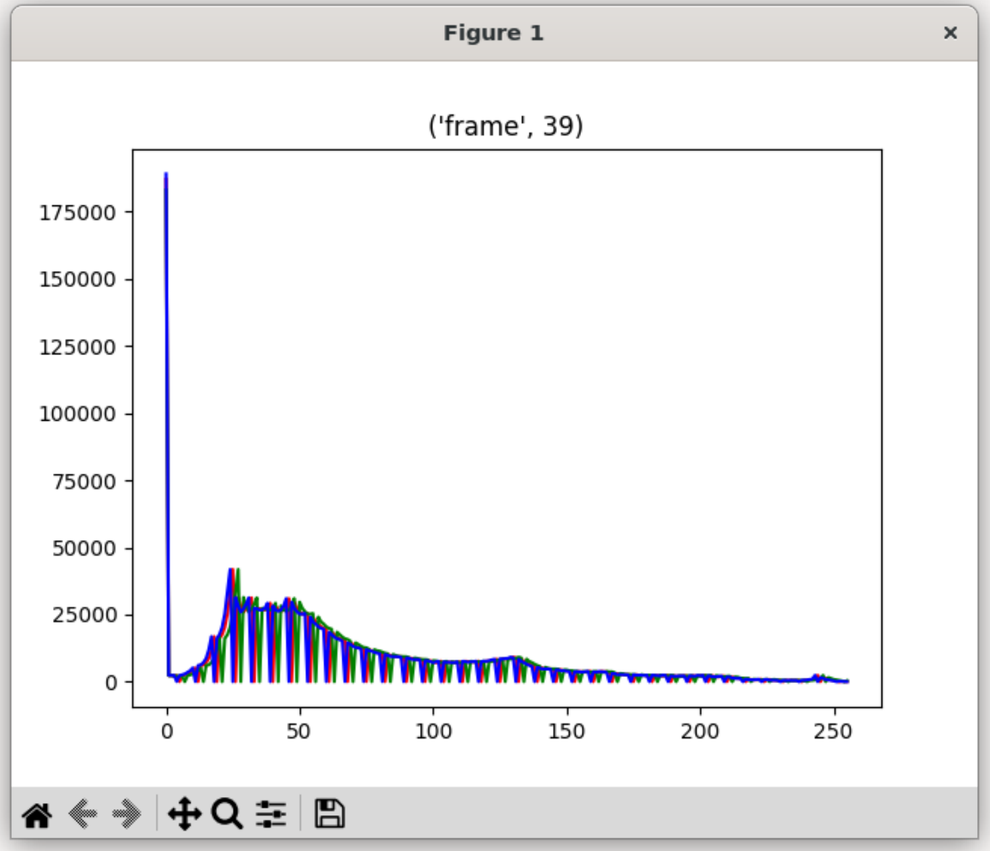
- Pixel 정보 히스토그램 그리기
- 기타 학습 내용 (기초 용어)

Pixel 정보 히스토그램 그리기

비디오의 전체 프레임 – RGB 채널 분리

시각화 과정에서 오차가 생기는 것인지 채널 별로 완전히 동일한 값이 아니라 오차가 있는 것을 확인할 수 있었음

```python
# import video file
cap = cv2.VideoCapture('../../datasets/pig/04_25_warp_images_KeyFrame.mp4')

while(cap.isOpened()):
    # read video file frame by frame
    ret, frame = cap.read()

    if ret == True:
        # BGR to RGB
        B, G, R = cv2.split(frame)

        # dist of pixels
        hist_R = cv2.calcHist([R], [0], None, [256], [0, 256])
        hist_G = cv2.calcHist([G], [0], None, [256], [0, 256])
        hist_B = cv2.calcHist([B], [0], None, [256], [0, 256])

        # plot
        plt.plot(hist_R, color='red')
        plt.plot(hist_G, color='green')
        plt.plot(hist_B, color='blue')

    else:
        break
```

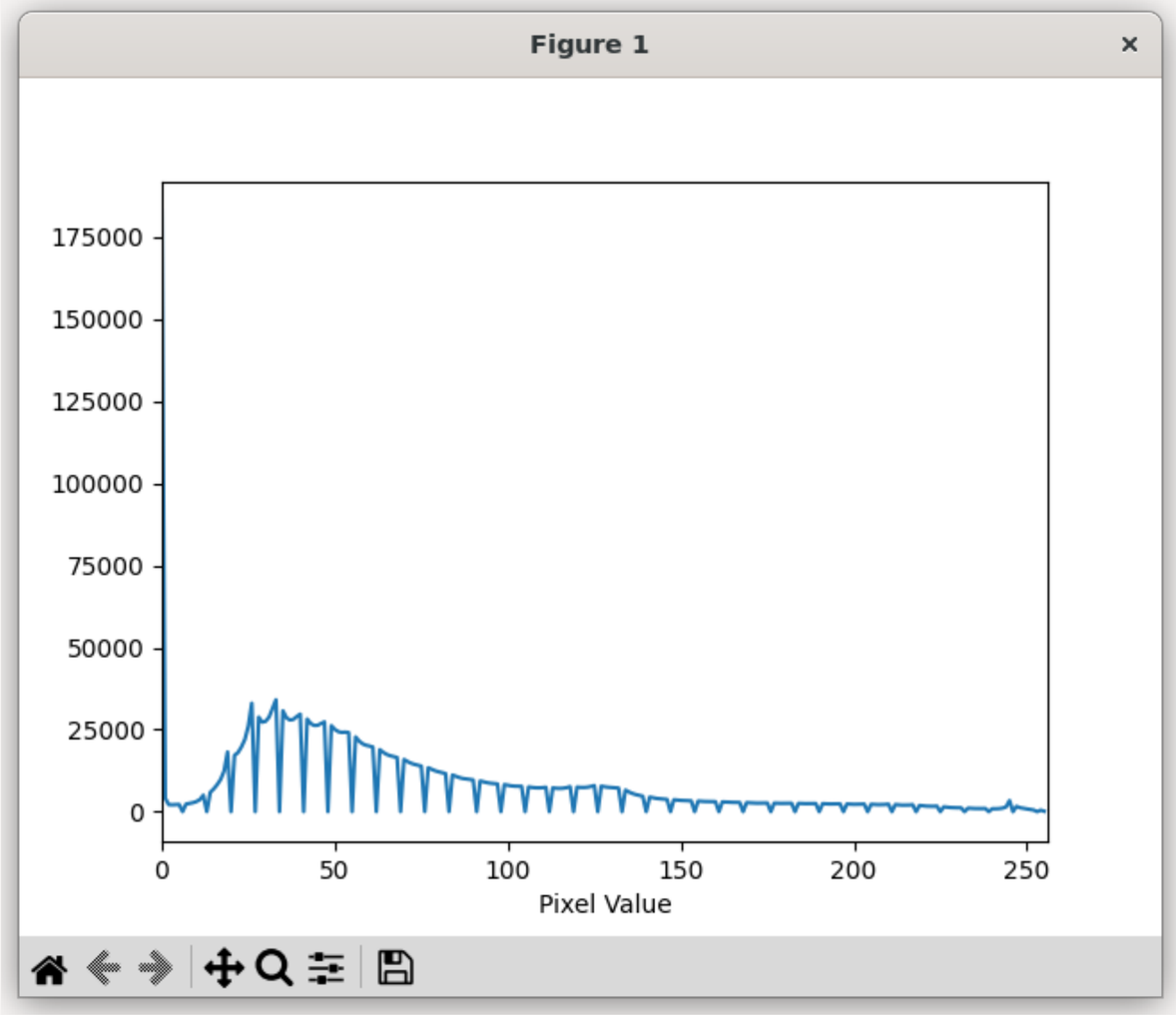


Pixel 정보 히스토그램 그리기

고려대학교

Gray scale, 비디오 평균 픽셀

```python
# import video file
cap = cv2.VideoCapture('../../datasets/pig/04_25_warp_images_KeyFrame.mp4')
frame_cnt = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))

hist_acc = np.zeros([256, 1])

while(cap.isOpened()):
    # read video file frame by frame
    ret, frame = cap.read()

    if ret == True:
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        # dist of pixels
        hist = cv2.calcHist([frame], [0], None, [256], [0, 256])
        hist_acc += hist

    else:
        break
```

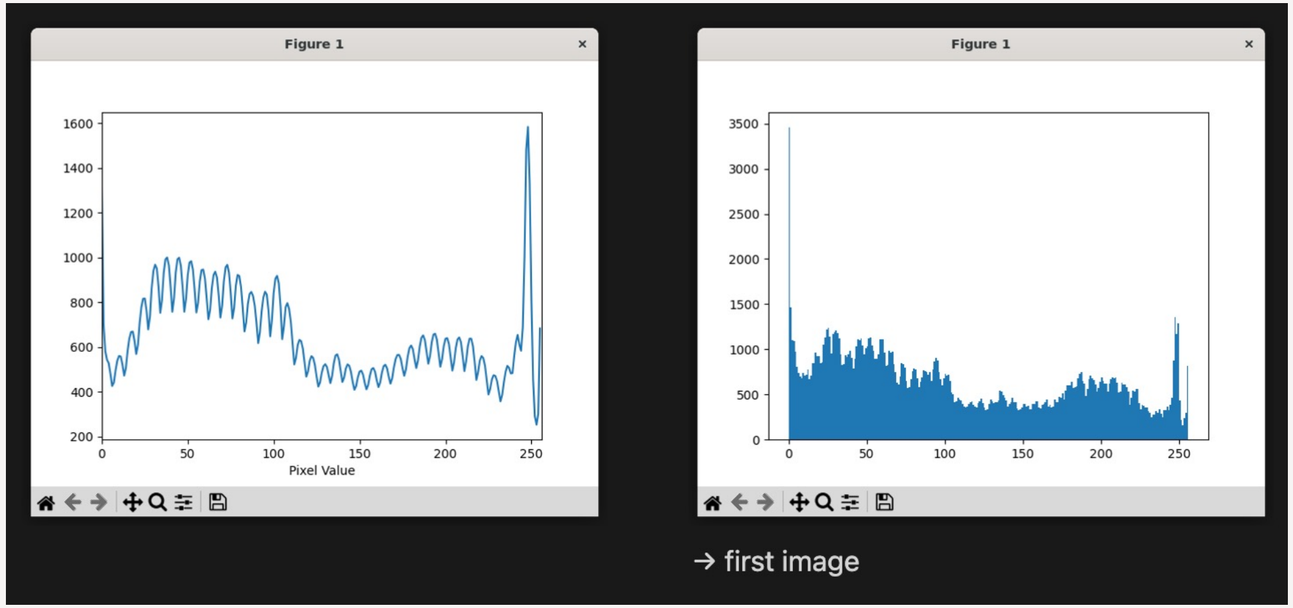


Pixel 정보 히스토그램 그리기

고려대학교

Gray scale, 폴더 내의 이미지 평균 픽셀

```python
# import folder
folder_path = '../../datasets/pig/jochiwon_08_14_keyFrame'
files = os.listdir(folder_path)

cap = cv2.VideoCapture('../../datasets/pig/04_25_warp_images_KeyFrame.mp4')
frame_cnt = len(files)

hist_acc = np.zeros([256, 1])

for file in files:
    # read file
    file_path = os.path.join(folder_path, file)
    image = cv2.imread(file_path)

    image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # dist of pixels
    hist = cv2.calcHist([image], [0], None, [256], [0, 256])
    hist_acc += hist

plt.plot(hist_acc / frame_cnt)
plt.show()
```



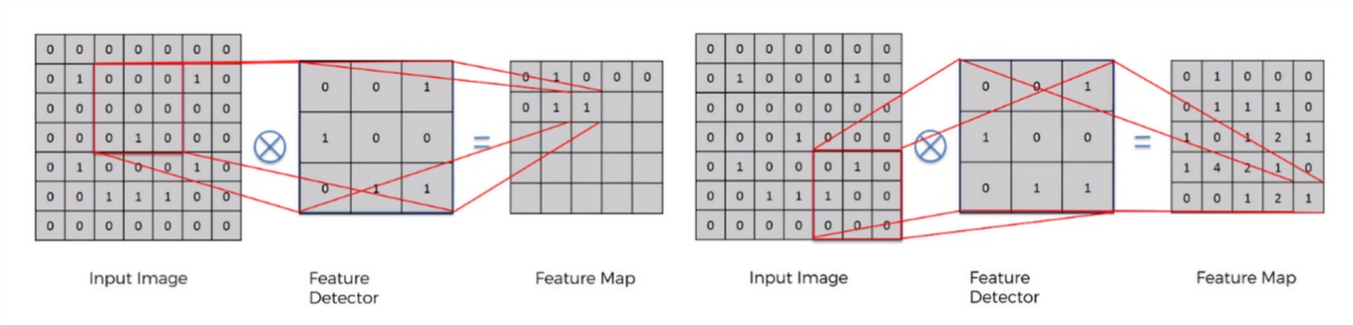
기타 학습 내용 (기초 용어)

Convolution

`f*g`

CNN에서는 큰 의미보다는 이미지 내에서 feature를 뽑기 위한 용도의 연산으로 쓰임

`Feature Detector` = `Kernel` = `Filter` 가 Input Image의 모든 영역을 훑으면서 특정한 Feature를 뽑게 되는 것, 완성된 행렬 = `Feature Map`



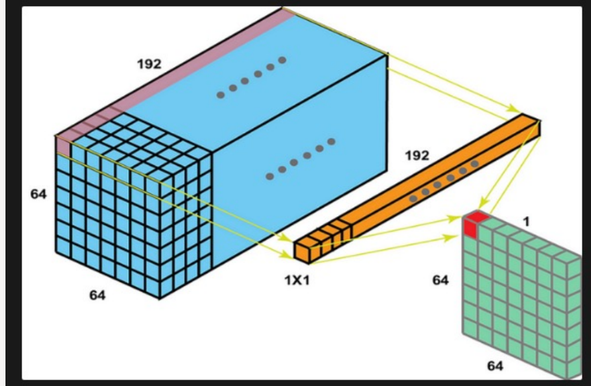
출처 | Deep Learning A-Z

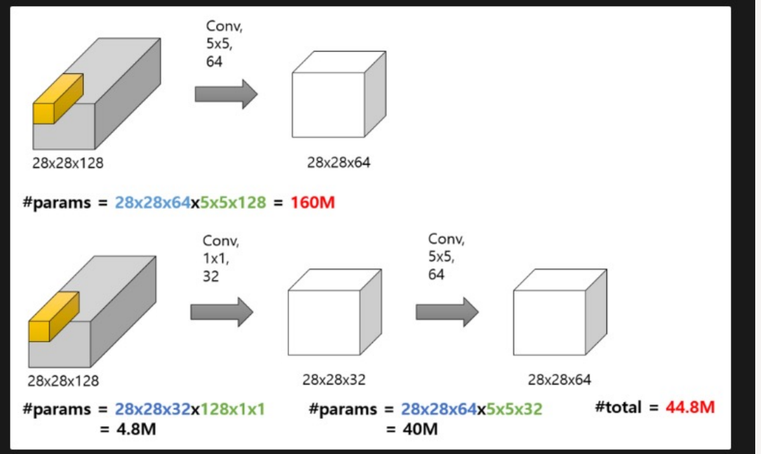
→ Input Image와 Feature Map의 크기는 7x7에서 5x5로 줄어들어 이후 연산에서 속도적 이점을 얻을 수 있음 (output의 크기는 `[(input_size - kernel_size + 2*padding) / stride] + 1`) * *stride = 한 번에 움직이는 크기*

→ 이렇게 뽑아낸 Feature Map에서도 중요한 Feature를 뽑아내기 위해 학습하는 역할을 수행하는 곳이 `Convolutional Layer`

기타 학습 내용 (기초 용어)

1x1 Convolution





(`1` x `1` x `#channel`)의 필터를 `#filter` 개 사용하여 convolution 연산을 진행하는 것

이때 `#channel` 은 입력 단의 channel 수, `#filter` 는 원하는 출력 단의 channel 수로 지정

- Channel의 수를 줄일 수 있음 (유지도 가능)
- 계산량 감소
- 사용되는 파라미터 수가 감소 → 모델을 더욱 깊게 구성 → 기존보다 많은 수의 비선형성 활성화 함수를 사용 → 모델은 점점 더 구체적인 패턴을 파악할 수 있어 성능 향상

```python
import torch
import torch.nn as nn

class Conv1x1Example(nn.Module):
    def __init__(self, in_channels, out_channels):
        super(Conv1x1Example, self).__init__()

        # 1x1 컨볼루션 정의: 입력 채널 수에서 출력 채널 수로 조절
        self.conv1x1 = nn.Conv2d(in_channels, out_channels, kernel_size=1)

        self.relu = nn.ReLU()

    def forward(self, x):
        x = self.conv1x1(x)
        x = self.relu(x)
        return x

model = Conv1x1Example(in_channels=64, out_channels=32)

input_tensor = torch.randn(1, 64, 56, 56)
output_tensor = model(input_tensor)

print("입력 텐서 크기:", input_tensor.shape) # 1, 64, 56, 56
print("출력 텐서 크기:", output_tensor.shape) # 1, 32, 56, 56

# 1x1 convolution이므로 same padding 적용을 안해도 width, height는 유지됨
```