저조도 이미지 향상
이미지 노이즈 제거

최석재 *lingua@naver.com*

이미지 해상도

- 여기에서는 OpenCV를 이용하여 이미지의 해상도를 다루어본다

- 구글 드라이브에 접속한다

- from google.colab import drive
- drive.mount('/content/gdrive’)

- %cd /content/gdrive/MyDrive/pytest_img/opencv/ # 경로 변경

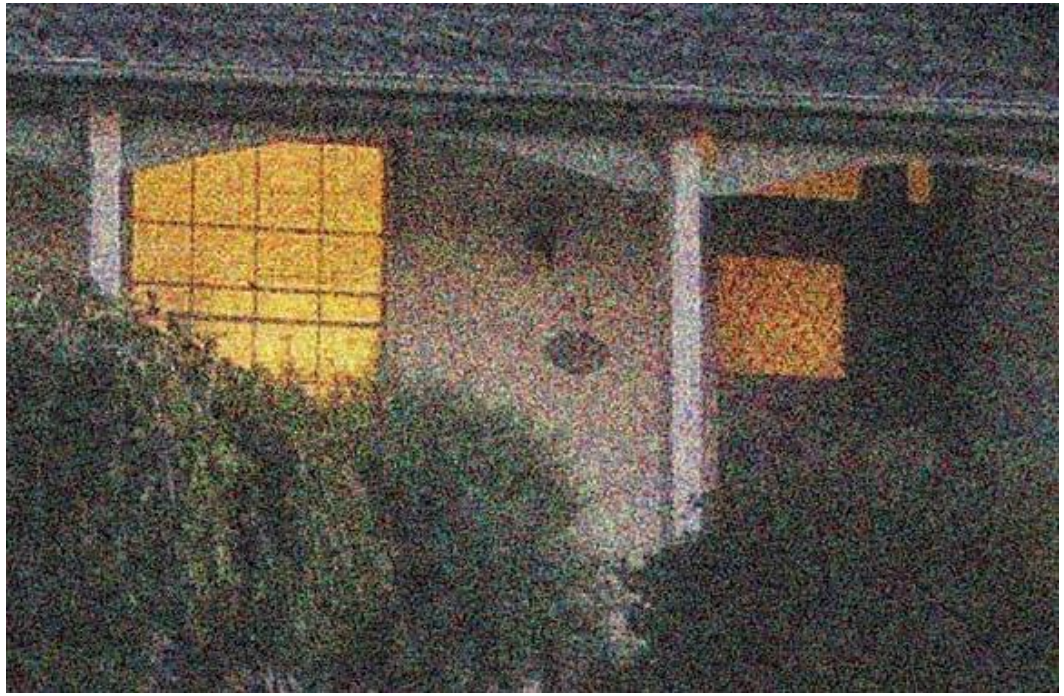
원본 이미지

- 원본 이미지를 그려본다
- 노이즈가 많은 이미지이다

- import cv2
- from google.colab.patches import cv2_imshow

- img = cv2.imread("noise_house.png")

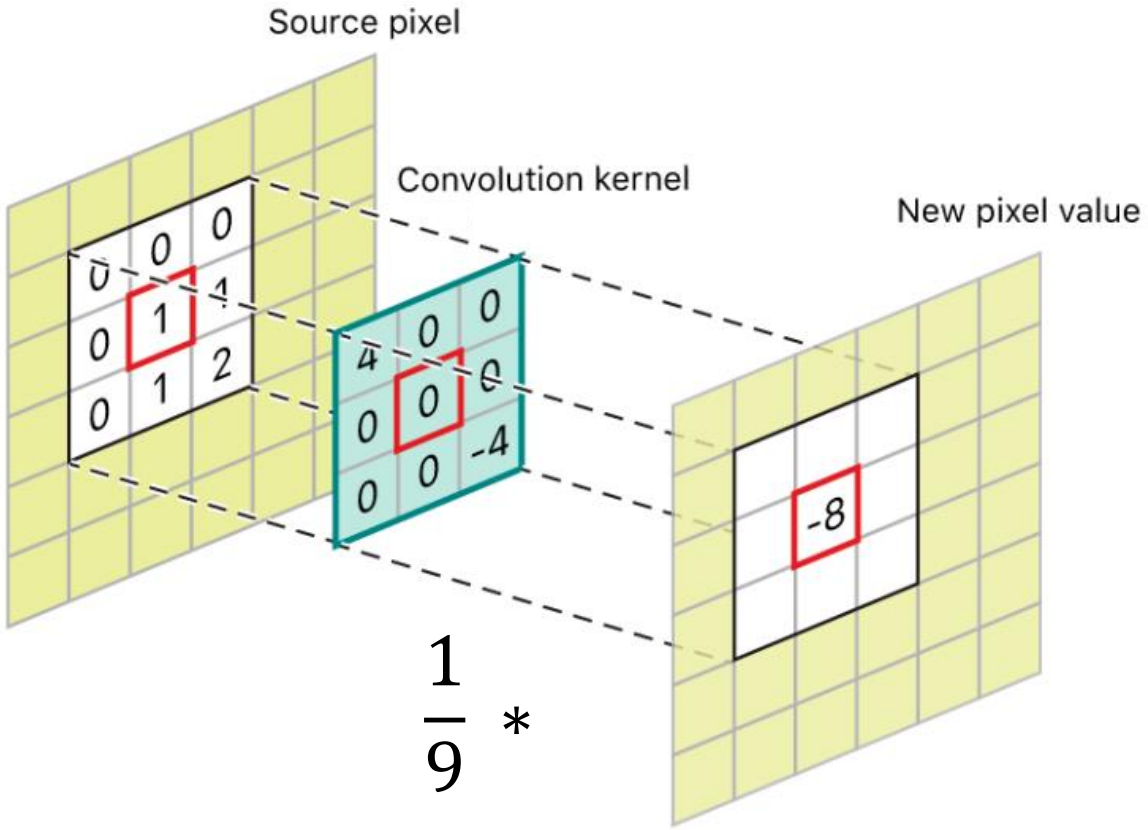
- cv2_imshow(img) # 이미지 그리기
- cv2.waitKey(0)
- cv2.destroyAllWindows()



이미지 흐리게 하기

- 이미지를 흐릿하게 하는 기본 원리는 커널을 전체 이미지에 합성곱 연산을 시킨 후
- 픽셀 평균값을 대입하는 것

- 각 픽셀과 커널의 곱셈과 합산을 거친 값을
- 커널 원소의 개수로 나누어 평균을 구한다



$$\frac{1}{9} *$$

이미지 흐리게 하기 – blur

$$\frac{1}{9}\begin{matrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{matrix}$$

사용하는 커널 (3x3 커널의 경우)

- blur()는 이미지 각 픽셀들의 평균값을 계산하여 이미지를 흐리게 한다
- 커널의 너비와 높이를 조절하여 이미지의 흐림 정도를 조절한다
- 커널이 커질수록 이미지 픽셀을 평균화시키는 범위가 커져 흐린 정도가 커진다

- img_blur = cv2.blur(img, ksize=(5, 5)) # 이미지를 흐리게 한다

- cv2_imshow(img_blur)
- cv2.waitKey(0)
- cv2.destroyAllWindows()

커널을 직접 만들기

- 커널을 직접 만들어서 진행할 수도 있다
- 앞과 동일한 커널을 사용해본다

평균 블러 커널 생성

- import numpy as np
- kernel_size = 5
- kernel = np.ones((kernel_size, kernel_size), np.float32) / (kernel_size**2) # 셀 개수로 나눔
- print(kernel)

이미지와 필터의 합성곱 연산을 수행하는 OpenCV의 filter2D로 블러 필터 적용

- img_blur = cv2.filter2D(src=img, ddepth=-1, kernel=kernel)

- cv2_imshow(img_blur)
- cv2.waitKey(0)
- cv2.destroyAllWindows()

ddepth는 출력 이미지의 정보량으로, 일반적으로는 -1을 지정하여 원본 이미지와 같게 한다
이 외에 cv2.CV_8U(8-bit Unsigned Integer. 0~255), cv2.CU_16U(16-bit Unsigned Integer. 0~65535)
cv2.CV_32F(32-bit Floating Point)로 처리할 수도 있다

이미지 흐리게 하기 – medianBlur

- 평균이 아닌, 중간값을 사용하는 medianBlur()를 사용할 수도 있다
- 특정 가중치 커널을 사용하지 않고, 단순히 주어진 커널 크기 내의 픽셀 값들을 정렬한 후,
- 중앙값을 사용하여 현재 픽셀의 값을 대체한다

- img_blur = cv2.medianBlur(img, ksize=5)

- cv2_imshow(img_blur)
- cv2.waitKey(0)
- cv2.destroyAllWindows()

이미지 흐리게 하기 – Gaussian

$$\frac{1}{16}\begin{matrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{matrix}$$

사용하는 커널 (3x3 커널의 경우)

- 가우시안 분포(정규분포)를 기반으로 한 가중치를 사용하는 GaussianBlur()를 사용할 수도 있다
  - 가우시안 블러는 중심에서 멀어질수록 가중치가 감소하는 정규 분포를 따르는 커널을 사용
  - 가우시안 커널은 중앙 픽셀에 더 많은 가중치를 주고, 멀어질수록 가중치를 줄인다
- 이미지를 평탄화하기 때문에 노이즈를 제거하면서 블러를 주는 효과를 갖는다
  - 필수 매개변수 sigmaX는 X 방향의 표준편차로서, 보통 0을 지정하여 자동 계산되게 한다
  - sigmaY도 있으나 일반적으로는 sigmaX와 같은 값을 사용한다 (0)

- img_blur = cv2.GaussianBlur(img, ksize=(5, 5), sigmaX=0)

- cv2_imshow(img_blur)
- cv2.waitKey(0)
- cv2.destroyAllWindows()

노이즈 제거 – Gaussian

- GaussianBlur()의 매개변수를 적절히 선택하면 노이즈만 줄일 수 있다
- ksize와 sigmaX를 작은값~큰값으로 조절하며 적정값을 본다
- 두 값을 키울수록 더 많은 픽셀이 평균화되어 노이즈 제거 효과가 강해지면서 블러 효과도 강해진다

- img_blur = cv2.GaussianBlur(img, ksize=(7, 7), sigmaX=3.0) # ksize는 홀수를 사용

- cv2_imshow(img_blur)
- cv2.waitKey(0)
- cv2.destroyAllWindows()

•**sigmaColor만 사용**: 공간 정보를 무시하게 되므로, 같은 색상 값이라면 이미지의 모든 부분이 혼합된다. 가장자리가 뭉개져서 이미지의 디테일이 손상될 수 있다
•**sigmaSpace만 사용**: 색상 정보를 무시하게 되므로, 공간적으로 가까운 픽셀들끼리만 혼합된다. 블러링 효과가 충분히 나타나지 않을 수 있다

노이즈 제거 – bilateralFilter

- bilateralFilter()는 노이즈 제거에 조금 더 좋은 성능을 보인다
- 평탄화할 때 색상은 물론 공간적 거리를 같이 고려하기 때문에
- 이미지의 가장자리는 보존하면서 노이즈를 제거한다. 단, 속도가 느리다

d: 고려할 픽셀 주변 직경(-1은 자동계산), sigmaColor: 클수록 색상이 비슷한 필터를 더 멀리까지 찾아 평탄화한다, sigmaSpace: 클수록 더 멀리까지 평탄화

- img_blur = cv2.bilateralFilter(img, d=-1, sigmaColor=75, sigmaSpace=75)

- cv2_imshow(img_blur)
- cv2.waitKey(0)
- cv2.destroyAllWindows()

이미지 선명하게 하기

- 이미지를 선명하게 할 때도 합성곱 연산을 사용한다
- 중앙 픽셀의 값을 강조하고, 주변 픽셀의 값을 감소시키는 커널을 사용한다
- 이미지의 가장자리 및 세부 사항이 강조되게 한다

$$\begin{matrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{matrix}$$

중앙에 큰 가중치를,
주변엔 음의 가중치로
이미지 대비를 증가시킴

$$\begin{matrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{matrix}$$

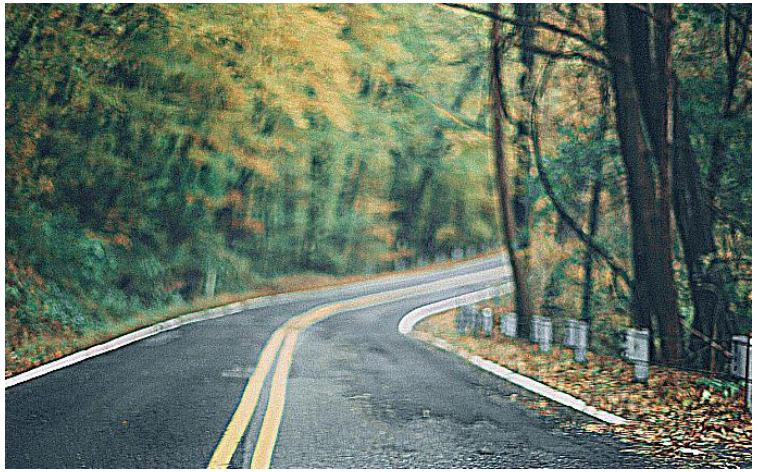
중앙에 적당히 큰 가중치를 두고
네 개 픽셀에만 -1을 주어 균형을 잡음
이미지 대비로 에지를 파악하면서도
0값으로 주변 픽셀의 값을 주지 않아
노이즈를 줄이는 효과

원본 이미지

- img = cv2.imread("blurred.png")

- cv2_imshow(img)
- cv2.waitKey(0)
- cv2.destroyAllWindows()

샤프닝 필터 사용 1



- 샤프닝 필터를 이용하여 직접 이미지를 선명하게 할 수 있다

- import numpy as np

샤프닝 커널 정의
- kernel = np.array([[-1, -1, -1],
  [-1, 9, -1],
  [-1, -1, -1]])

- sharpened_image = cv2.filter2D(img, ddepth=-1, kernel=kernel) # 필터 적용
  ddepth=-1은 원본과 같게 함
- cv2_imshow(sharpened_image)
- cv2.waitKey(0)
- cv2.destroyAllWindows()

샤프닝 필터 사용 2

- import numpy as np

샤프닝 커널 정의

- kernel = np.array([[0, -1, 0],
  [-1, 5, -1],
  [0, -1, 0]])

- sharpened_image = cv2.filter2D(img, ddepth=-1, kernel=kernel) # 필터 적용

ddepth=-1은 원본과 같게 함

- cv2_imshow(sharpened_image)
- cv2.waitKey(0)
- cv2.destroyAllWindows()

언샤프 마스킹

- Unsharp Masking은
- 원본 이미지에서 낮은 주파수를 갖는 블러 처리된 이미지를 빼면
- 원본 이미지의 높은 주파수 성분(세부사항과 가장자리)이 강조되는 원리를 이용하는 방법이다

가우시안 블러 적용

- blurred = cv2.GaussianBlur(img, (9, 9), 10.0)

원본 이미지에서 블러 이미지를 뺀 이미지 생성

- sharpened_image = cv2.addWeighted(src1=img, alpha=1.5, src2=blurred, beta=-0.5, gamma=0)

src1: 원본 이미지, alpha: src1에 대한 가중치,
src2: 블러처리된 이미지, beta: src2에 대한 가중치,
gamma: 밝기 조절을 위한 추가 상수값(일반적으로 0)

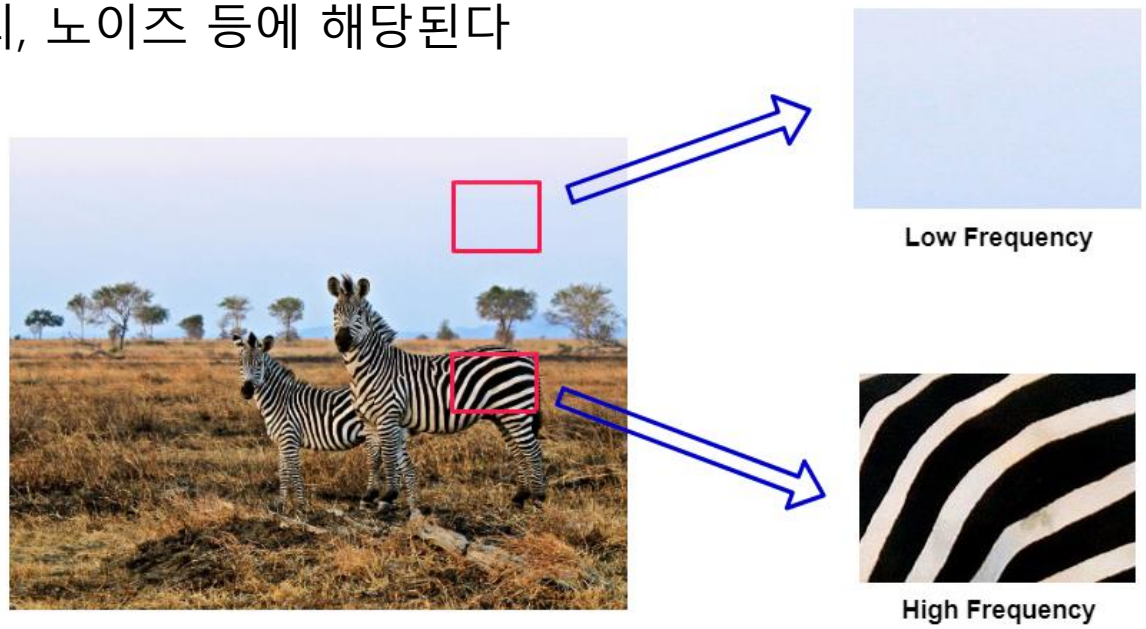
- cv2_imshow(sharpened_image)
- cv2.waitKey(0)
- cv2.destroyAllWindows()

이미지와 주파수

- 이미지의 각 픽셀값을 시간적인 신호로 간주하면 주파수로 변환할 수 있다

- 픽셀값의 변화가 적은 부분은 이미지의 기본적인 형태나 부드러운 영역이 되며,
- 이들은 낮은 주파수를 갖게 된다

- 픽셀값의 변화가 큰 부분은 세부적인 사항, 가장자리, 노이즈 등에 해당된다
- 이들은 높은 주파수를 갖게 된다



대비 높이기

- 객체의 형태가 두드러지도록 이미지의 대비를 높일 수 있다

- OpenCV의 equalizeHist() 함수는 이미지의 픽셀 강도 분포를 조정하여,
- 밝기 값이 고르게 분포하도록 만든다
- 그 결과 저조도에서 고조도에 이르기까지 다양한 밝기가 사용되어
- 전체적으로 이미지의 대비가 높아지도록 한다

- 밝기 값을 얻기 위하여 그레이스케일로 변환한다

원본 이미지

- img = cv2.imread("airplane.png")

- cv2_imshow(img)
- cv2.waitKey(0)
- cv2.destroyAllWindows()

흑백 이미지 대비 높이기

- 이미지를 그레이스케일로 읽은 뒤, cv2.equalizeHist() 를 적용한다

- img = cv2.imread("airplane.png", cv2.IMREAD_GRAYSCALE)
- img_enhanced = cv2.equalizeHist(img)

- cv2_imshow(img)
- cv2.waitKey(0)
- cv2.destroyAllWindows()

컬러 이미지 대비 높이기

- 품질은 다소 떨어지지만 컬러 이미지에도 적용할 수 있다
- 먼저 밝기와 색상을 표현하는 YUV 포맷으로 변환한 뒤,
- 색상 채널은 변경하지 않고 밝기 채널만 함수를 적용하고, 다시 컬러 포맷으로 변환한다

- img = cv2.imread("airplane.png")
- img_yuv = cv2.cvtColor(img, cv2.COLOR_BGR2YUV)

- img_yuv[:, :, 0] = cv2.equalizeHist(img_yuv[:, :, 0])
- img_bgr = cv2.cvtColor(img_yuv, cv2.COLOR_YUV2BGR)

- cv2_imshow(img_bgr)
- cv2.waitKey(0)
- cv2.destroyAllWindows()

세번째 차원의 순서는 Y(0, 밝기), U(1, 청색), V(2, 적색)

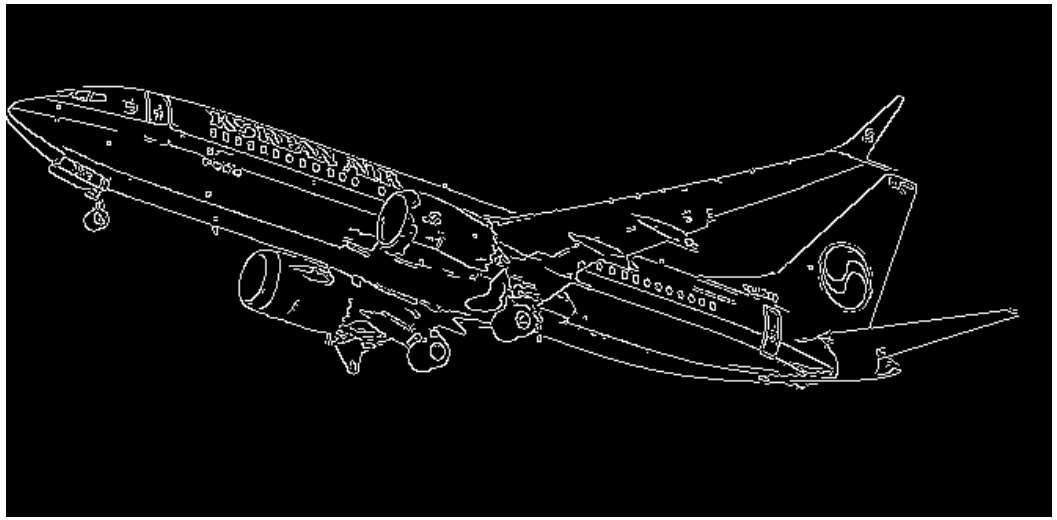
윤곽선 감지하기

- 이미지의 경계는 급격한 픽셀 값의 변화를 통해 파악한다
- 이미지 경계 감지는 캐니 에지 검출기를 많이 사용한다
- 컬러 이미지 그대로 사용해도 되지만, 좋은 결과를 위해 흑백 이미지로 읽은 뒤 사용한다

- img = cv2.imread("airplane.png", cv2.IMREAD_GRAYSCALE)
- img_edges = cv2.Canny(img, threshold1=50, threshold2=150)

threshold1: 에지 강도 하한선, threshold2: 에지 강도 상한선

- cv2_imshow(img_edges)
- cv2.waitKey(0)
- cv2.destroyAllWindows()



윤곽선 감지하기

- 가우시안 블러로 노이즈를 제거하고 검출하면 더 좋은 효과를 볼 수 있다

- img = cv2.imread("airplane.png", cv2.IMREAD_GRAYSCALE)
- img_blur = cv2.GaussianBlur(img, ksize=(5, 5), sigmaX=1.5) # 가우시안 블러
- img_edges = cv2.Canny(img_blur, threshold1=50, threshold2=150)

- cv2_imshow(img_edges)
- cv2.waitKey(0)
- cv2.destroyAllWindows()